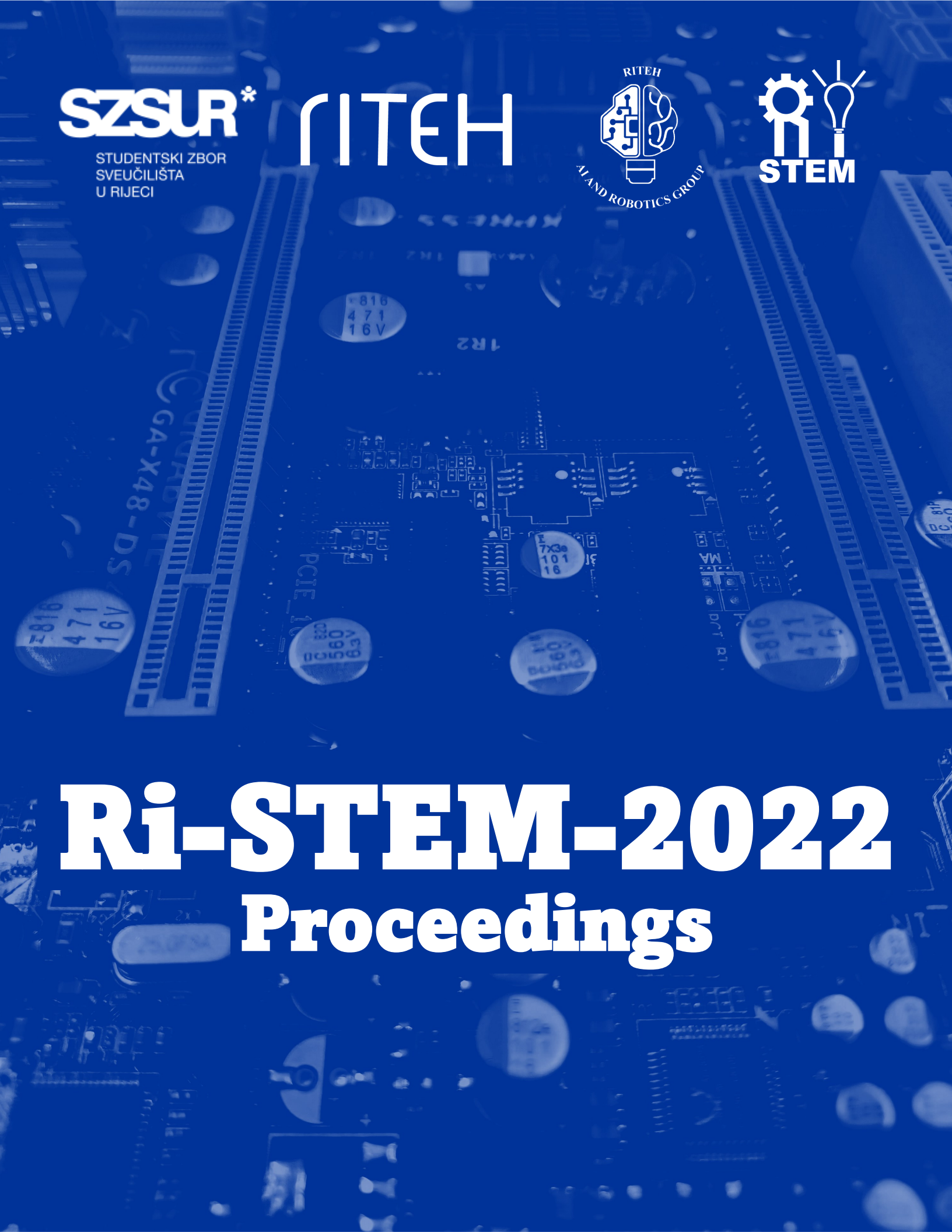
RI-STEM-2022 Proceedings



Ri-STEM-2022

Proceedings

International Scientific Student Conference

RI-STEM 2022
Rijeka, Croatia

Proceedings

Editors:

Sandi Baressi Šegota
Ivan Lorencin
Nikola Anđelić
Zlatan Car

RI-STEM 2022

Proceedings of the International Scientific Student Conference

Scientific Committee

prof. dr. sc. Zlatan Car	Croatia
prof. dr. sc. Nenad Filipović	Serbia
prof. dr. sc. Zoran Marković	Serbia
Ing. Jan Kudláček, Ph.D.	Czechia
izv. prof. dr. sc. Dubravko Franković	Croatia
izv. prof. dr. sc. Miroslav Malinović	Bosnia and Herzegovina
assist. Prof. dr. sc. Themis Exarchos	Greece
doc. dr. sc. Vedran Mrzljak	Croatia
doc. dr. sc. Igor Poljak	Croatia
dr. sc. Nikola Anđelić	Croatia
dr.sc. Ana Zulijani	Croatia
dr. sc. Ivan Lorencin	Croatia

Organizational Committee

Asim Kurahović, BS mech.	Bosnia and Herzegovina
Denis Andrić, BS comp.	Germany
Daniel Štifanić, mag. ing. el.	Croatia
Jelena Musulin, mag. ing. el.	Croatia
Matko Glučina, mag. ing. el.	Croatia
Sandi Baressi Šegota, mag. ing. comp.	Croatia
Tijana Šušteršič, MSc mech. eng.	Serbia
Anđela Blagojević, MSc el. eng. comp. scien.	Serbia
Klara Smolić, dr. med.	Serbia
Ariana Lorencin, mag. obs.	Croatia
stud. Denis Mijolović, univ. bacc. ing. el.	Sweden
stud. Arian Čarapina	Croatia
Stud. Eugen Šegota	Croatia
Marko Šegon, mag. phys.	Czechia
Simon Lysdahlgaard, dr. med.	Denmark
Georgios Kalykakis, PhD candidate	Greece

Sponsors and Supporters

Studentski Zbor Sveučilišta u Rijeci
Student Council of University of Rijeka

Tehnički Fakultet Rijeka
Faculty of Engineering Rijeka

Turistička Zajednica Općine Medulin
Medulin Riviera

Studentski Zbor Tehničkog Fakulteta u Rijeci
Student Council of Faculty of Engineering Rijeka

Riteh AI and Robotics Group

Foreword

Respected colleagues and dear friends,

It is with great happiness and honor that I write the foreword to the second RI-STEM conference. The idea of the RI-STEM conference is to provide the students of any study level with a space that will enable them to experience the publication and presentation process of a scientific article – an experience which will undoubtedly serve them well in their futures, whether they pic a professional or academical path.

This year, we bring you 30 papers from students of various STEM fields. People from various countries across Europe – Croatia, Bosnia and Herzegovina, Serbia, Chechia, Denmark, Greece and Germany; have come together to enable the knowledge and experience exchange at this years RI-STEM conference, and they have my sincerest gratitude.

I hope that we will continue with this great experience and enable even more students to gain the publication experience through RI-STEM. Multiple accounts of students who have shown how the RI-STEM publication helped them in improvement of their job applications, or served as proof of practical knowledge, is a good motivation to try and make RI-STEM a mainstay conference.

With that in mind, I thank everyone involved in RI-STEM – members of organizational and scientific committees, sponsors, and most of all – students submitting their papers. We hope to see you again next year.

Kindest regards,

Sandi Baressi Šegota
President of the organizational committee

Contents

ISBN: 978-953-8246-26-5

Urinary Bladder Cancer Diagnostics Using Machine Learning Techniques

Ivan LORENCIN[1], Josip ŠPANJOL[2,3], Zlatan CAR[4]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; ilorencin@riteh.hr
[2] Faculty of Medicine - University of Rijeka, Braće Branchetta 20/1, 51000, Rijeka, Croatia; josip.spanjol@medri.uniri.hr
[2] Clinical Hospital Centre, Krešimirova ul. 42, 51000, Rijeka
[4] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; car@riteh.hr

Abstract: Urinary bladder cancer is one of the most common malignancies of the urinary system. Due to the high metastatic potential and recurrence rate, artificial intelligence (ML) and machine learning (ML) methods are introduced to increase the diagnostic performances. In this paper, a brief overview of the application of ML methods on two data sets is provided. In the case of this research, cystoscopic images and images collected by using computerized tomography (CT) are used. From the obtained results, it can be seen that utilization of ML methods enables higher results both in the case of classification and semantic segmentation.

Keywords: artificial intelligence, clinical support systems, cystoscopy, computerized tomography, convolutional neural network, data augmentation, edge detectors, meta-heuristic algorithms, transfer learning, urinary bladder cancer

1. Introduction

Urinary bladder cancer is one of the most common malignancies of the urinary system. The root of cancer lays in the mutation of the bladder mucosa cells. The main characteristics of urinary bladder cancer are high reoccurrence rate and high metastatic potential [1,2]. For these reasons, it can be concluded that an accurate and fast diagnostic procedure is of vital importance. With aim of developing faster and more accurate diagnostic procedures, artificial intelligence (AI) and machine learning (ML) are introduced [3-5]. In this paper, an overview of the AI utilization for urinary bladder cancer diagnostics is provided. The presented diagnostic procedures are based on two different diagnostic procedures: a cystoscopy and computerized tomography.

2. Cystoscopy

The first used data set is the data set collected during optical cystoscopy and examination of urinary bladder mucosa with a confocal laser endomicroscope. The images of urinary bladder mucosa are divided into four different classes:

- High-grade carcinoma,
- Low-grade carcinoma,
- Carcinoma in-situ, and
- Healthy mucosa

An example of each class is given in Figure 1.

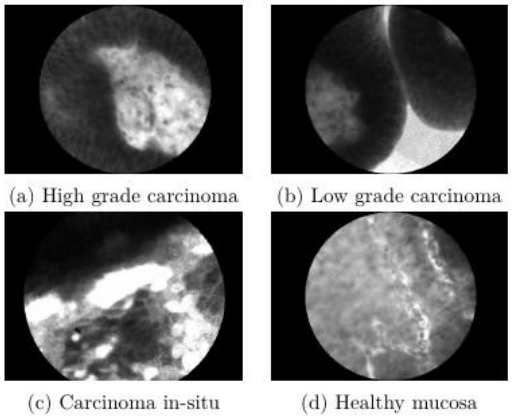
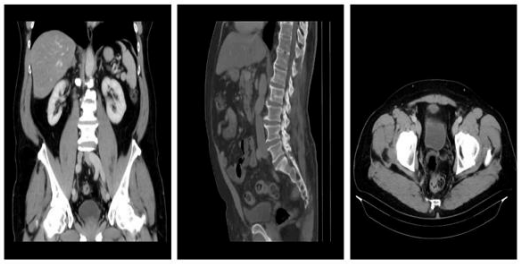


Figure 1: Example of each class from cystoscopy data set

The presented data set is used to recognize the grade of urinary bladder cancer by using AI-based classification algorithms [6].

3. Computerized Tomography

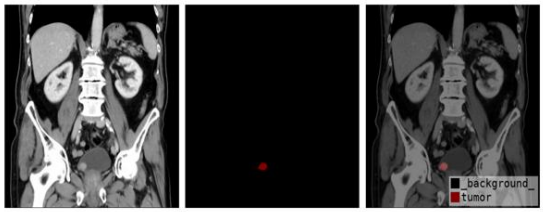
The other data set used in this research is the data set collected by using Computerized Tomography. The images collected with CT are captured in three planes (frontal, horizontal, and sagittal) and are divided into 6 classes (images without a bladder, healthy bladder, unilateral bladder wall thickening, circular bladder wall thickening, exophytic formation, and invasion outside the contour of the bladder) [7]. An example for each plane used in the research is given in Figure 2.



(a) Frontal plane (b) Sagittal plane (c) Horizontal plane

Figure 2: Example of each plane from CT data set

The aim of this data set is to develop a system for semantic segmentation of urinary bladder cancer masses from CT images. A brief process of semantic segmentation is given in Figure 3.



(a) CT image without annotation (b) Annotation mask (c) Image with superposed annotation mask

Figure 3: Semantic segmentation process

4. Methods overview

In this section, a brief description of the used AI methods is provided.

4.1 Classification

For the classification of images contained in CT and cystoscopy data sets, several standard CNN architectures are used [8]:

- AlexNet
- VGG-16
- ResNet50
- ResNet101
- ResNet152
- InceptionV3
- Inception-ResNet

The aforementioned CNN architectures are also used as pre-trained backbones for the implementation of transfer learning methodology in U-net for semantic segmentation.

4.2. Semantic segmentation

For the purposes of semantic segmentation, a standard U-net architecture is used. A schematic representation of the used U-net architecture is given in Figure 4.

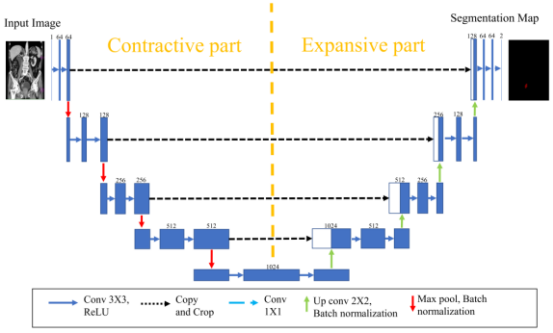


Figure 4: Schematic representation of U-net architecture

4.3. Hybrid models

Alongside standard AI-based algorithms, several approaches for hybrid models are developed as well. The aim of hybrid model utilization is to increase not only classification or semantic segmentation performances, but also generalization performances. In this section, a brief description of the hybrid models is presented.

4.3.1. Data Augmentation

With aim of performance-boosting, data augmentation is used. For the case of cystoscopy data sets, both geometrical and GAN-based augmentation procedures are used. For the case of CT data set, only geometrical augmentation is used.

4.3.2. Edge detectors

With aim of performance-boosting, edge detector-based hybrid models are also proposed. For this purpose, two different edge detection methodologies are utilized. The first approach is to utilize gradient edge detectors. In this research, three different gradient kernels are used [9,10]:

- Roberts edge detector,
- Sobel edge detector, and
- Prewitt edge detector.

Alongside gradient edge detector, Laplacian edge detector is used as well.

4.3.3. Transfer Learning

One of the modern approaches in machine learning is certainly transfer learning. In this research, transfer learning methodology is applied to both classification and semantic segmentation. The transfer learning approach includes the implementation of standard CNN architectures pre-trained by using ImageNet data set. For the case of classification problems, pre-trained architectures with added fully connected layers are used. On the other hand, in the case of U-net architecture, pre-trained CNN architectures are used as contractive parts of a U-net architecture [11-13].

4.3.4. Model selection using meta-heuristic approach

Alongside presented methods, meta-heuristic algorithms are used for model selection of the proposed classification and semantic segmentation algorithms. In this research, two meta-heuristic algorithms are used:

- Genetic algorithm (GA) and
- Discrete Particle Swarm algorithm (D-PS).

Both algorithms are executed by using an adapted fitness function that includes both classification and segmentation, as well as generalization performance. This transfer function can be defined as:

$$\boldsymbol{F(\mu)} = \overline{\boldsymbol{\Psi(\mu)}}\mathbf{1} - \boldsymbol{\sigma}\big(\boldsymbol{\Psi(\mu)}\big), \qquad (\mathbf{1})$$

where:

- $F(\mu)$ – fitness function,
- $\Psi(\mu)$ – evaluation measure, and
- σ – standard deviation .

Mean evaluation measure and standard deviation are measures derived from the k-fold cross-validation procedure used during training [14].

5. Results

In this section, a brief overview of the achieved results is provided.

5.1. Results achieved with Cystoscopy data set

If the results achieved by using AlexNet-based hybrid models on the cystoscopic data set are compared, it can be noticed that the highest classification and generalization performances are achieved if meta-heuristic algorithms are used. Furthermore, it can be seen that the sightly lower performances are achieved if the edge detector-based hybrid model and GAN-based augmentation are used, as presented in Figure 5.

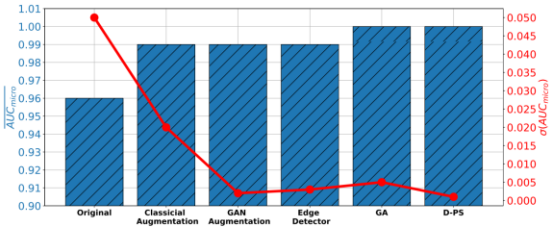


Figure 5: Classification of cystoscopic images using AlexNet

The similar results can be observed if VGG-16 architecture is used. In this case, the highest performances are achieved if D-PS is used for the model selection, as presented in Figure 6.

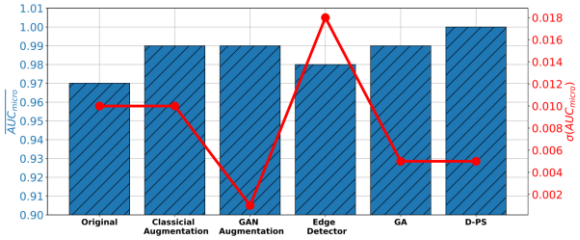


Figure 5: Classification of cystoscopic images using VGG-16

5.2. Results achieved with CT data set

If the results achieved on the CT data set are compared, it can be seen that the highest semantic segmentation and generalization performance are achieved if a general semantic segmentation system is used. In this case, the highest results are achieved if the U-net model is selected by using the D-PS algorithm. Furthermore, it can be seen that similar results are achieved if one semantic segmentation system is used for each diagnosis. In the case of horizontal and frontal planes, the highest results are achieved if U-net models are selected by using D-PS. On the other hand, in the case of the sagittal plane, the highest performances are achieved if the transfer learning methodology is used. If one semantic segmentation system is used for each diagnosis separately, significantly lower performances are achieved, as presented in Figure 6.

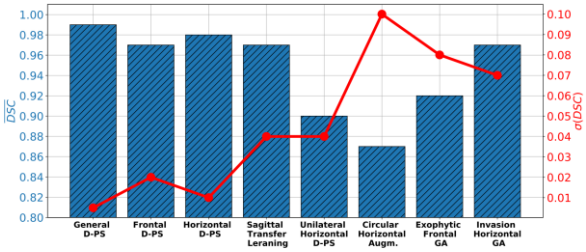


Figure 6: Results achieved with each semantic segmentation methods

6. Conclusions

From the presented results it can be seen that the higher performances are achieved in the case of both data sets if a form of the hybrid model is used. Such a property is particularly emphasized in the case of meta-heuristic algorithms and data augmentation. In final, it can be stated that there is a possibility for the implementation of ML methods in urinary bladder cancer diagnostics. The utilization of ML methods enables higher results both in the case of classification and semantic segmentation.

Acknowledgement

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the

grant 2021-1-HR01-KA220-HED-000031177, and University of Rijeka scientific grant uniri-tehnic-18-275-1447.

Bibliography

[1] Duty, B. D., and M. J. Conlin. "Principles of urologic endoscopy." Campbell-Walsh Urology, 11th ed.; Elsevier: Philadelphia, PA, USA (2016): 136-152.
[2] Lerner, Seth P., et al. "Fluorescence and white light cystoscopy for detection of carcinoma in situ of the urinary bladder." Urologic Oncology: Seminars and Original Investigations. Vol. 30. No. 3. Elsevier, 2012.
[3] Ikeda, Atsushi, et al. "Support system of cystoscopic diagnosis for bladder cancer based on artificial intelligence." Journal of endourology 34.3 (2020): 352-358.
[4] Eminaga, Okyaz, et al. "An Efficient Framework for Video Documentation of Bladder Lesions for Cystoscopy: A Proof-of-Concept Study." (2022).
[5] Checcucci, Enrico, et al. "Applications of neural networks in urology: a systematic review." Current Opinion in Urology 30.6 (2020): 788-807.
[6] Lorencin, Ivan, et al. "On urinary bladder cancer diagnosis: Utilization of deep convolutional generative adversarial networks for data augmentation." Biology 10.3 (2021): 175.
[7] Lorencin, Ivan, et al. "Utilization of Convolutional Neural Networks for Urinary Bladder Cancer Diagnosis Recognition From CT Imagery." 2021 IEEE 21st International Conference on Bioinformatics and Bioengineering (BIBE). IEEE, 2021.
[8] Alzubaidi, Laith, et al. "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions." Journal of big Data 8.1 (2021): 1-74.
[9] Lorencin, Ivan, et al. "Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis." Artificial Intelligence in Medicine 102 (2020): 101746.
[10] Lorencin, Ivan, et al. "Edge detector-based hybrid artificial neural network models for urinary bladder cancer diagnosis." Enabling AI Applications in Data Science. Springer, Cham, 2021. 225-245.
[11] Baressi Šegota, Sandi, et al. "Semantic Segmentation of Urinary Bladder Cancer Masses From CT Images: A Transfer Learning Approach." Biology 10.11 (2021): 1134.
[12] Wurm, Michael, et al. "Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks." ISPRS journal of photogrammetry and remote sensing 150 (2019): 59-69.
[13] Sharma, Suvash, et al. "Semantic segmentation with transfer learning for off-road autonomous driving." Sensors 19.11 (2019): 2577.
[14] Lorencin, Ivan. An Intelligent System for Urinary Bladder Cancer Diagnostics. Diss. University of Rijeka. Faculty of Engineering, 2022.

ISBN: 978-953-8246-26-5

Detection of gas molecules using graphene based nanosensor

Nikola ANĐELIĆ [1*], Marko ČANAĐIJA [1], Zlatan CAR [1]

[1] Affiliation: Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, email: nandelic@riteh.hr, markoc@riteh.hr, car@riteh.hr

Abstract: The presented research is derived from a PhD thesis by dr. Anđelić. The following paper describes the process by which the graphene-based nanosensor was modelled and how Genetic Programming was applied for the process of local parameter determination.
Keywords: genetic programming, graphene, nanosensorics

Nanosensor for mass detection is a mechanical sensor that proved to be an excellent candidate in the detection of atoms and molecules [1]. The working principle of these sensors is the frequency shift method [2] which is based on the difference between the natural frequency of graphene with and without added mass in the form of atoms and molecules bonded with carbon atoms.

Natural frequencies, and the frequency shift method used in atom/molecule detection will be analysed using molecular dynamics (MD) simulation and non-local plate theory. Numerical results obtained using MD (natural frequencies) will be used to adjust the non-local parameter value in non-local thin plate theory in such a way that the frequency from non-local thin plate theories will be equated with the natural frequencies from MD by modifying the non-local parameter value. This will create a dataset for the implementation of the genetic programming algorithm in order to establish approximate correlations between the input data (mechanical characteristics of graphene, graphene dimensions, temperature and natural frequencies) with the output data (non-local parameter). Since most sensors have a specified temperature range and pressure at which the sensor can perform detection, it was initially assumed that graphene as a sensing element of the nanosensor can detect gas molecules in the range from 233.15 to 313.15 K and at a pressure in the range from 0 to 1 bar.

Before investigating the natural frequencies of one layer of graphene, the mechanical and thermodynamic characteristics of this material were obtained using MD with REBO interatomic potential [3]. The obtained mechanical and thermodynamic parameters were used in non-local thin plate theory to determine the natural frequencies of single-layer graphene sheet (SLGS) as well as to examine the influence of temperature, pressure, size of SLGS, a variation of non-local graphene parameter on natural frequencies caused by gas molecules attached to the surface of SLGS [4].

MD was used to determine the natural frequencies of SLGS, to examine the influence of SLGS size on natural frequencies as well as the absolute and relative frequency shift caused by attached gas molecules to the central SLGS atom using the displacement excitation method. The results of the above analyses in MD and non-local thin plate theory showed that the size of SLGS has the greatest influence on the natural frequencies of SLGS while temperature has a very small influence on the natural frequencies. MD simulations with NPT ensemble showed that the pressure oscillates a lot during equilibration and vibration simulation and therefore the influence of pressure was omitted from further analyses.

Analyses performed using non-local thin plate theory also showed that the value of the non-local parameter has a large influence on the natural frequencies of SLGS. The mechanical characteristics are temperature dependent and with increasing temperature, the value of these parameters with small oscillations gradually decreases.

Physical data of 3 molecules of chemical weapons of mass destruction were used to investigate the possibility of detecting gas molecules using non-local theory and MD. Both theories have shown that SLGS can detect gas molecules using the absolute and relative frequency shift method. One of the main shortcomings of the non-local theory of elasticity is the unknown value of the non-local parameter, and in many studies, its value is set in a arbitrary range [5]. Therefore, the goal is to apply a genetic programming algorithm [6] to obtain a symbolic expression by which its value could be determined. The genetic programming algorithm was used to determine the symbolic expression that would connect the input values with mathematical functions, namely: mechanical parameters (modulus of elasticity, shear modulus, Poison coefficients and dimensions of graphene), operating conditions (temperature) and natural frequencies obtained by MD s a non-local parameter that represents the output value of this symbolic expression. For a genetic programming algorithm to be able to generate a symbolic expression, it is necessary to develop a data set on which the algorithm can be trained and tested. Three temperatures (233.15, 273.15, and 313.15 [K]) were used to generate the data set in MD and non-local theory. Graphene models with dimensions ranging from 20×10 to 40×20 [nm] were used for these simulations. The natural frequencies obtained by MD were used as reference values for tuning the natural frequencies in non-local theory so that the value of the non-local parameter was adjusted so that the natural frequency value was equivalent to those frequencies obtained in MD. Based on the obtained parameters using MD and non-local theory, a set of data was created which was used in the genetic programming algorithm to determine the equation by which the value of the non-local parameter could be determined.

The equation for determining the value of a non-local parameter was chosen based on the highest achieved value of the R^2 correlation coefficient which is equal to 0.9688. The obtained equation for determining the non-local parameter value was used to determine the absolute and relative frequency shift caused by the mass attached in the centre of the fully clamped graphene sheet. The obtained results were compared with the averaged absolute and relative frequency shift values obtained using MD simulations. The results of the comparison showed that on

average the calculated values of absolute and relative frequency shift are 5 % lower than those obtained using MD simulations.

Acknowledgments

This research has been (partly) supported by the CEEPUS network Ciii-HR-0108, European Regional Development Fund under the grant K.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447, Croatian Science Foundation under the project IP-2019-04-4703 and University of Rijeka under the project number uniri-technic 18-37.

References

[1] Schedin, Fredrik, et al. "Detection of individual gas molecules adsorbed on graphene." Nature materials 6.9 (2007): 652-655.

[2] Anđelić, N., M. Čanađija, and Z. Car. "Nems Resonator For Detection Of Chemical Warfare Agents Based On Single Layer Graphene Sheet." International Conference on Innovative Technologies, IN-TECH 2019, Belgrade, Serbia

[3] Brenner, Donald W. "Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films." Physical review B 42.15 (1990): 9458.

[4] Anđelić, Nikola, Zlatan Car, and Marko Čanađija. "NEMS resonators for detection of chemical warfare agents based on graphene sheet." Mathematical Problems in Engineering 2019 (2019).

[5] Fazelzadeh, S. Ahmad, and Esmaeal Ghavanloo. "Nanoscale mass sensing based on vibration of single-layered graphene sheet in thermal environments." Acta Mechanica Sinica 30.1 (2014): 84-91.

[6] Koza, John R., and Riccardo Poli. "Genetic programming." Search methodologies. Springer, Boston, MA, 2005. 127-164.

ISBN: 978-953-8246-26-5

Application of convolutional neural networks for detection and classification of brain tumors

Natalija Ivoš [1*]

[1*] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Crotia, nivos@riteh.hr .

Abstract: A brain tumor is a collection of abnormal cells in brain which damages healthy brain cells. Diagnosis and identifying the exact type and degree of tumor in early stages play an important role in choosing the relevant treatment plan. Recent progress in deep learning field has helped the health industry in diagnosing many diseases. The convolutional neural network (CNN) is most widely and most commonly used algorithm for visual learning and image recognition. Proposed paper includes a deep neural network approach and a CNN based model to classify magnetic resonance imaging (MRI) into four categories. The data group was taken from publicly available Kaggle site which contains 7022 MRI images. The proposed system achieved satisfactory results in terms of multiclass classification.

Keywords: Artificial intelligence, Brain Tumour, Classification, Convolutional Neural Network, Deep Learning.

1. Introduction

Cancer is one of the leading causes of death, and brain tumours are one of the deadliest. The complex structure of the human brain complicates the diagnosis of brain tumours and therefore magnetic resonance imaging (MRI) is used to obtain high-quality images. Artificial intelligence and deep learning are primarily used in image processing to segment, identify, and classify MRI images. Classification of brain tumour images is an important part of medical image processing because it helps in making an accurate diagnosis and further planning treatment. Some of the international articles we reviewed on the detection and classification of brain tumours using in-depth learning are S. Basheera and M.S.S.Ran who proposed a method for brain tumours classification where the tumour is initially segmented from an MRI image and then classified using a pre-trained convolutional neural network (CNN) [1]. Deep learning is one of the machine learning methods that uses neural network architecture with possible hundreds of hidden layers between the input and output layers. CNNs are currently the most widely used neural networks in the field of machine learning, used in the field of disease diagnosis and classification based on medical images, especially CT and MRI images as it does not require pre-processing or extraction features before the training process [2,3]. In general, CNN consists of many layers: input layer, convolutional layer, RELU layer, fully connected layer, classification layer, and output layer [4,5]. The main goal of this research is to apply CNN for detection and classification of brain tumors that could help physicians with earlier detection and treatment.

2. Methodology

In this study, images of brain tumours are stored as databases and four directories are created, each consisting of images for a specific class of glioma, meningioma, pituitary and no tumour. The database is divided into training and testing sets, where 70% of the data is used in the training phase and the rest is used in the testing phase. Figure 1 shows a brain tumor for each class.

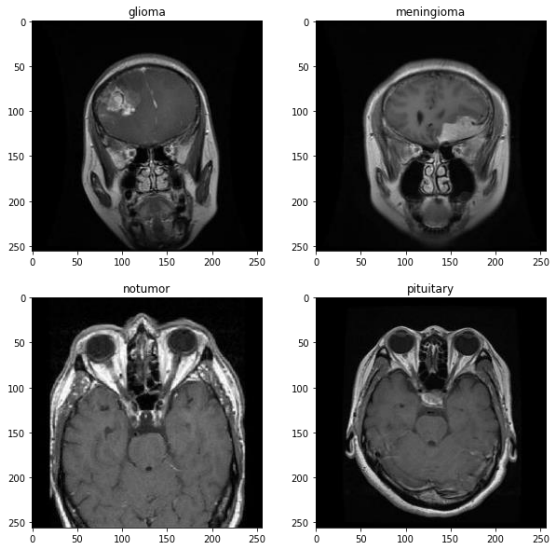


Figure 1. *Brain tumor for each class.*

The dataset consists of 7022 MRI images. The dataset has a solid number of samples per class and is relatively balanced which further facilitates the development of the model itself. All images were digitized at a resolution of 256 × 256 pixels. Table 1 lists the number of images of each class in the dataset.

Table 1. *Distribution of the dataset in four classes*

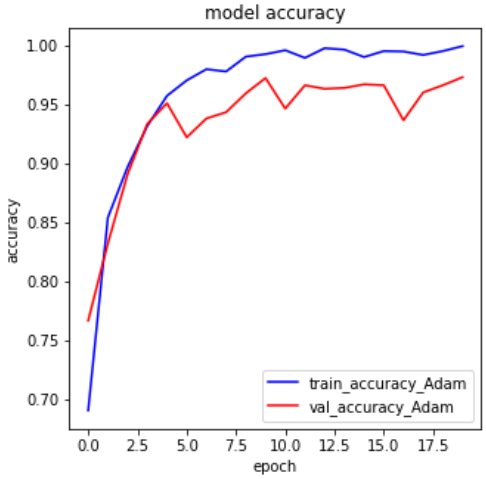
Tumour Type	Number of Images
Glioma	1621
Meningioma	1645
Pituitary	1999
No Tumor	1757
Total:	**7022**

In this paper, the simple CNN model is utilized in order to perform a classification of brain tumour MRI images into four different classes. Initially, one convolution layer of 16 filters having a filter size 3 x 3 was added. The reason for setting a small number of filters as 16 is to detect edges, corners and lines. Afterwards, a maximum unification layer with 2 x 2 filter was added to get the maximum summary of that image, then the number of convolutional layers and the number of filters was increased to 32, 64 and 128, which have the same 3 x 3

filter. This combines these small samples as the number of filters increases and finds larger samples like squares, circles, etc. Layers of maximum shrinkage were applied to them to get the maximum. Finally, a fully connected dense layer of 256 neurons along with a softmax output layer was applied that calculates the probability score for each class and classifies the final decision labels either that the MRI input image contains tumor or does not contain tumor. The Rectified Linear Unit (ReLU) activation function in each convolutional layer was applied. The activation function converts the input weighted sum into the output of that node. The ReLU is often used in the hidden layers of a CNN [6,7].

3. Results and Discussion

Experimental evaluations were performed to determine the significance and accuracy of the proposed CNN model. The model was trained for 20 epochs with a batch size of 64. The experiment was performed using TensorFlow and Keras library in Python with GPU support. Most of the neuron cells based techniques, including CNN, uses gradient drop to lower the error rate for the training process and to reform internal parameters. The gradient descent is a first order optimization algorithm and its derivate gives direction and increases or decreases the error function. Information directs the error function, changing it down to the local minimum [8]. The orthodox gradient descent technique calculates the gradient of the entire training data, which makes its process computationally slow. Algorithms such as Adam, Stochastic Gradient Descent (SDG) and Root Mean Square Propagation (RMSprop) have been developed to solve this problem. When using Adam as an optimizer the initial validation accuracy is below 0.75, but after one epoch the validation accuracy increases sharply to almost 0.85. In the same way, the initial loss of validation is below 0.7, but after one epoch the loss decreases below 0.4. There is a positive trend toward improving accuracy and reducing losses, as shown in Figure 2. Initially, the accuracy is low but it is gradually improving to 97%. The results in terms of train - validation accuracy and loss for the models trained with SGD and RMSprop optimizers are shown in Figure 3, and Figure 4.



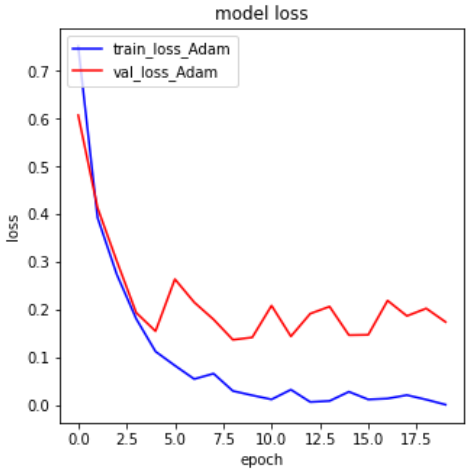
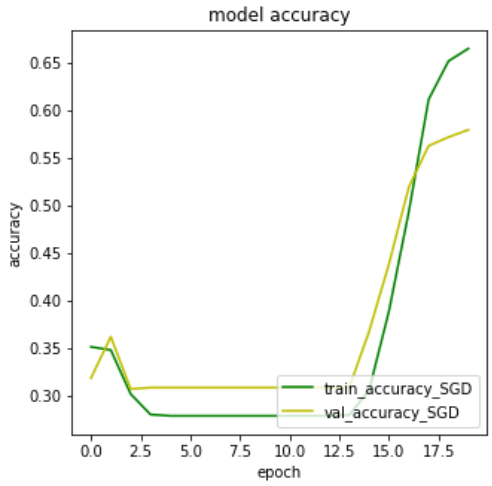


Figure 2. *Model accuracy and model loss for training and validation set using the Adam optimizer.*



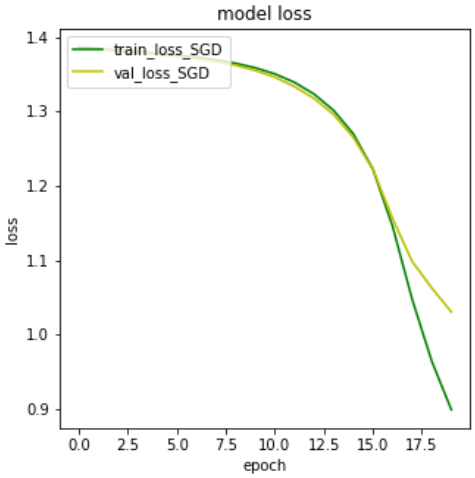


Figure 3. *Model accuracy and model loss for training and validation set using the SDG optimizer.*

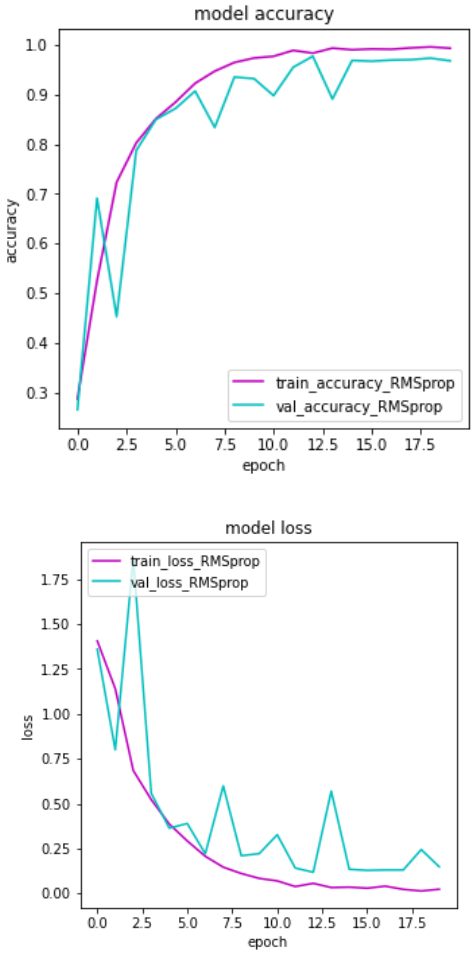


Figure 4 *Model accuracy and model loss for training and validation set using the RMSprop optimizer.*

The accuracy of the SGD optimizer (at the fixed number of epochs) is lower with a value of 63% while the loss is higher with a value of 1.3. The accuracy and loss graphs in the case of Adam and RMSprop optimizers are almost equal, with an accuracy of 97%, and 96.8%, respectively, and a loss of 0.1748, and 0.148, respectively

4. Conclusion

This paper is a comparative analysis of different optimization algorithms used in the proposed CNN architecture to perform and evaluate the performance of a four-class brain tumor classification. The analogy was made on a publicly available set of MRI brain images. Both, quantitative and graphical results show that all the optimizers work consistently, but the Adam works faster. CNN is designed to minimize or sometimes revoke data in pre-processing steps and is typically used to process raw images. It is based on two processes: convolution which is performed using trainable filters with a predetermined specification that is set during the training phase and downsampling in the training phase to achieve high accuracy [9,10].

Acknowledgments

I thank assistant Daniel Štifanić for mentorship during the work on this project and article.

Reference

[1] S. Bacheera, M. S. S. Ram, Classification of brain tumors using deep features extracted usding CNN, J. Phys., 1172(2019), 012016.

[2] 16. Lotan, Eyal, et al. State of the art: Machine learning applications in glioma imaging. American Journal of Roentgenology, Vol 212.

[3] Sasikala, M., and N. Kumaravel. A wavelet-based optimal texture feature set for classification of brain tumours. Journal of medical engineering & technology, Vol 32.

[4] Alquran H, Alqudah AM, Abu-Qasmieh I, Al-Badarneh A, Almashaqbeh S. ECG classification using higher order spectral estimation and deep learning techniques. Neural Network World, Vol 29.

[5] Mohsen, Heba, et al. Classification using deep learning neural networks for brain tumors. Future Computing and Informatics Journal, Vol 3.

[6] V. Nair, G. Hinton, *Rectified linear units improve restricted boltzmann machines*, ICML, 2010.

[7] Stefan Bauer et al, 'Multiscale Modelig for Image Analysis of Brain Tumor Studies', IEEE Transactions on Biomedica Engineering.

[8] De S., Mukherjee A., Ullah E. Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration. *arXiv*. 20181807.06766 [Google Scholar]

[9] Sasikala, M., and N. Kumaravel. Wavelet based automatic segmentation of brain tumors using optimal texture features. 4th Kuala Lumpur International Conference on Biomedical Engineering 2008. Springer, Berlin, Heidelberg, 2008.

[10] Mathur, Neha, et al. Detection of Brain Tumor in MRI Image through Fuzzy-Based Approach. High-Resolution Neuroimaging-Basic Physical Principles and Clinical Applications.

ISBN: 978-953-8246-26-5

Therapeutic Hypothermia: a Survey

Ariana Lorencin [1*]

[1] Clinical Hospital Center Rijeka, Krešimirova ul. 42, 51000, Rijeka, email. ariana.rabac@gmail.com

Cerebral damage caused by asphyxia is a medical and socio-economic problem. Asphyxia is a prepartum, intrapartum, or postpartum disorder that manifests as a disorder in the exchange of gases in the placenta and fetal lungs. There is mild and severe asphyxia. Asphyxia has a high perinatal mortality. It can cause various damages and conditions such as epilepsy, cerebral palsy, ataxia, visual disturbances, hearing disorders, mental retardation, motor developmental delay, and other conditions. Hypoxic-ischemic encephalopathy (HIE) is a brain lesion. It develops due to a lack of oxygen and circulatory disorders as a result of severe labor asphyxia. There are two stages: the primary and the secondary stage. The clinical picture of the secondary stage is neonatal convulsions. Between the primary and secondary stage, there is a so-called. latent phase (up to 6 hours of life). At this stage, therapeutic hypothermia should be started as a method of treatment. Therapeutic hypothermia is the induction of hypothermic newborn hypothermia. It should be started before the onset of the secondary stage of brain damage, in the latent phase. That period is until the 6th hour of the newborn's life. There are two methods of therapeutic hypothermia, which are selective hypothermia of the head and hypothermia of the whole body. Selective hypothermia of the head is performed at 34-34.5 degrees Celsius and whole-body hypothermia at 33-33.5 degrees Celsius. It lasts up to 72 hours. During and after the implementation of therapeutic hypothermia, intensive monitoring of vital parameters, application of analgesia and sedation, and monitoring of biochemical parameters and electrical activity of the brain is required. After performing therapeutic hypothermia, the body is gradually warmed to physiological temperature. Following evidence of efficacy, this form of treatment was included in the recommendation of the ILCOR guidelines for neonatal resuscitation.

Keywords: Asphyxia, Hypoxic-ischemic encephalopathy, Therapeutic hypothermia

1. Introduction

Cerebral damage caused by asphyxia is a medical and socio-economic problem. Timely diagnosis and treatment are a great challenge for perinatologists and pediatricians. Difficulties in estimating the severity of postpartum asphyxia create uncertainty in assessing the risk of damage and the causal link with later life damage in newborns. We consider such newborns to be neuro-risk newborns. Hypoxic-ischemic encephalopathy is the most common cause of postpartum asphyxia. It is important to diagnose postpartum asphyxia and hypoxia-ischemic encephalopathy promptly and to start treatment in neonatal intensive care units promptly in order to prevent the risks of developing these lesions. Understanding the etiology and pathogenesis of hypoxia-ischemic encephalopathy at the molecular and cellular levels has enabled new approaches in the treatment of such a condition. In addition to conservative drug treatment and fluid replacement, therapeutic hypothermia is used as a method of treating hypoxia-ischemic encephalopathy and has become the standard in clinical practice [1].

2. Asphyxia and hypoxic-ischemic encephalopathy

Asphyxia is a prepartum, intrapartum, or postpartum disorder that manifests as a disorder in the exchange of gases in the placenta and fetal lungs. It reduces the perfusion of oxygen through the fetal organs. Asphyxia leads to hypoxia (decreased oxygen), hypoxemia (decreased blood pressure), hypocapnia (increased carbon dioxide in the arterial blood), and fetal acidosis. Asphyxia has high perinatal mortality. According to the World Health Organization, 4 million children worldwide die each year from birth asphyxia, and 38% of them die before the age of 5 due to the consequences of childbirth asphyxia. Risks of postpartum asphyxia can be prolonged (long) and recurrent (rapid) labor, prolonged amniotic fluid leakage, meconium amniotic fluid, high maternal age, multiple pregnancies, poorly controlled pregnancies, fetal growth retardation (IUGR), oxytocin use in labor, preeclampsia, maternal and fetal anemia, premature placental abruption, and placenta previa. Childbirth asphyxia can cause a variety of impairments and conditions such as epilepsy, cerebral palsy, ataxia, visual disturbances, hearing impairment, mental retardation, motor retardation, and other conditions. Childbirth asphyxia is divided into mild and severe [2,3].

Hypoxic-ischemic encephalopathy (HIE) is a brain lesion. It develops due to a lack of oxygen and circulatory disorders as a result of severe labor asphyxia. Hypoxic-ischemic encephalopathy is diagnosed in 1-8 / 1000 live births in developed countries. Risks are the aforementioned risks of asphyxia itself such as long or fast birth, meconium amniotic fluid, preeclampsia, fetal growth retardation (IUGR), placental disorders (placenta previa, abruption), maternal age, multiple pregnancies, etc. idiopathic. Hypoxic-ischemic encephalopathy is an acute brain injury and neuronal necrosis that results in death and numerous injuries. Hypoxic-ischemic encephalopathy has two stages, the primary and secondary stage. Between the primary and secondary stage there is a so-called latent phase where cerebral oxidative metabolism is close to normal and brain edema has subsided (up to 6 hours of age). At this stage, therapeutic hypothermia should be started as a method of treatment. For a long time, hypoxic-ischemic encephalopathy was treated only supportively. In recent years, therapeutic hypothermia has been used as a method of early treatment. Therapeutic hypothermia has shown good results and has become the standard in clinical practice worldwide [1,4-6].

3. Therapeutic hypothermia

Treatment of asphyxiated neonates includes resuscitation, general systemic measures, and neuroprotective measures or procedures. Nonprotection is a set of treatment measures aimed at preserving the integrity of the central nervous system and preventing damage to the central

nervous system. Currently, the only clinically accepted method of treatment is moderate therapeutic hypothermia. Therapeutic hypothermia is the induction of hypothermic newborn hypothermia. Therapeutic hypothermia should be initiated before the onset of the secondary stage of brain damage, in the latent phase. This period is until the 6th hour of the newborn's life [1].

To start with the therapeutic hypothesis, we must meet all the criteria. We must first prove birth asphyxia. Then we must clinically prove hypoxia-ischemic encephalopathy and do an EEG. After proving labor asphyxia and hypoxia-ischemic encephalopathy, we begin therapeutic hypothermia. There are two methods of therapeutic hypothermia and these are selective hypothermia of the head and hypothermia of the whole body. Selective hypothermia of the head is carried out at 34-34.5 degrees Celsius and whole-body hypothermia at 33-33.5 degrees Celsius. Servo-controlled devices are available for therapeutic hypothermia, as presented in Figure 1.

Figure 1. *Apparatus for therapeutic hypothermia*

They contain a cap or wrapper with coolant, as presented in Figure 2.

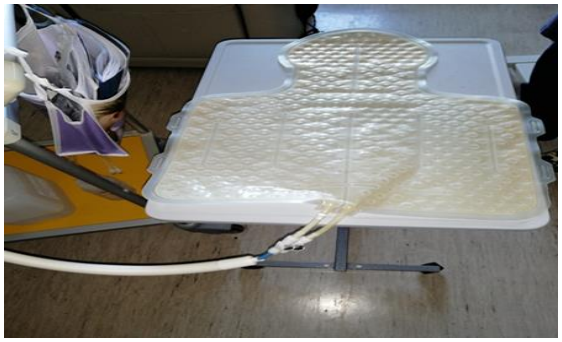


Figure 2. Mattress for therapeutic hypothermia

The duration of therapeutic hypothermia usually lasts up to 72 hours. After the implementation of therapeutic hypothermia, the organism is gradually warmed up to physiological temperature. Intensive monitoring of vital parameters, application of analgesia and sedation, and monitoring of biochemical parameters and electrical activity of the brain are required during and after therapeutic hypothermia [1].
The neuroprotective mechanism of therapeutic hypothermia involves depression of neuronal metabolism, thus maintaining ATP levels and inhibiting secondary energy crisis. Inflammatory responses, convulsions (number and duration), and the formation of oxygen free radicals are reduced. However, ultimately the neuroprotective mechanism of therapeutic hypothermia has not been proven [1].

3.1. Proof of effectiveness

Numerous randomized studies have demonstrated a statistically significant reduction in mortality and neurodevelopmental abnormalities. The results are better in the treatment than in the treatment of severe hypoxic-ischemic encephalopathy. Following evidence of efficacy, this form of treatment was recommended by the ILCOR guidelines for neonatal resuscitation. It is important to know that therapeutic hypothermia should be anticipated and started during the resuscitation in the sense that the heat source on the resuscitation table should be switched off. In this way, the asphyxiated newborn begins to passively subcool by a natural mechanism. During the evaluation, a lower body temperature should be maintained until a decision on further treatment is made. Hyperthermia can exacerbate the degree of damage and increase mortality, so even maternal febrile birth is associated with neurological damage to the child [7].

3.2. Application

Once we have defined the criteria for performing therapeutic hypothermia, the newborn must be intubated, sedated, on a respirator, monitored, and must have an umbilical catheter, rectal probe, urinary catheter, and orogastric tube installed. During all these interventions, we must make sure that the heater on the resuscitation table is turned off to achieve passive hypothermia of the newborn. We need to do an ultrasound of the brain and heart, and X-ray processing of the newborn. Then we have to do laboratory findings, and microbiological processing. When we have done all that, we start with therapeutic hypothermia. Place the newborn on a mattress filled with coolant on a hypothermia table. We place a rectal probe to measure the temperature to a depth of 5 cm. We place a skin probe on the newborn's chest. The newborn is placed on continuous monitoring where we monitor vital functions (blood pressure, saturation, inhalation) throughout the procedure. Every 1 hour. Tube patency and height are monitored. Enteral intake of electrolytes, hydration and breast milk, and painkillers. This is most often the use of Morphine. We enter anticonvulsants and ordinate antibiotic therapy enterally. We monitor diuresis through a urinary catheter and control CBP, ABS, electrolytes, lactates, and glucose. During the process of therapeutic hypothermia, brain activity is monitored by continuous EEG recording. Monitoring of the newborn on therapeutic hypothermia is continuous and constant. We provide health care (dressing, turning) and nursing documentation. Therapeutic hypothermia (lasting 72 hours) is followed by gradual warming of the newborn (14 hours).

4. Conclusion

It can be concluded that in developed countries the incidence of severe birth asphyxia is relatively low (1–8 per thousand live births) but that its consequences are very worrying due to high mortality and various developmental disorders. Cerebral damage caused by asphyxia is a medical and socio-economic problem. Timely diagnosis and treatment is a great challenge for perinatologists and pediatricians. Difficulties in estimating the severity of postpartum asphyxia create uncertainty in assessing the risk of damage and causal links with later life defects in newborns. The prognosis of the neurodevelopmental source is based on the clinical picture of moderate or severe hypoxia-ischemic encephalopathy. numerous biochemical markers, morphological changes in the brain, and the results of neurophysiological tests.

Understanding the pathophysiology of brain damage has also determined the prognostic significance for the use of therapeutic hypothermia. It is important to understand the pathophysiology in monitoring patients undergoing neuroprotective treatment to stop the process of secondary neuronal degradation in the latent phase, after successful primary resuscitation. So far, moderate controlled hypothermia has shown the best results in clinical practice and has become the standard in clinical treatment.

Reference

[1] Juretić, Emilja, and Damir Lončarević. "Perinatalna asfiksija." Medix: specijalizirani medicinski dvomjesečnik 19.104/105 (2013): 163-171.

[2] Aslam, Hafiz Muhammad, et al. "Risk factors of birth asphyxia." Italian journal of pediatrics 40.1 (2014): 1-9.

[3] McGuire, William. "Perinatal asphyxia." BMJ clinical evidence 2007 (2007).

[4] Vannucci, Robert C. "Hypoxic-ischemic encephalopathy." American journal of perinatology 17.03 (2000): 113-120.

[5] Shankaran, Seetha, et al. "Whole-body hypothermia for neonates with hypoxic–ischemic encephalopathy." New England Journal of Medicine 353.15 (2005): 1574-1584.

[6] Cornette, L. "Therapeutic hypothermia in neonatal asphyxia." Facts, views & vision in ObGyn 4.2 (2012): 133.

[7] Perlman, Jeffrey M., et al. "Neonatal resuscitation: 2010 international consensus on cardiopulmonary resuscitation and emergency cardiovascular care science with treatment recommendations." Pediatrics 126.5 (2010): e1319-e1344.

ISBN: 978-953-8246-26-5

Cardiac arrhythmia prediction based on machine learning

Mario Negovetić[1*]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, mario.negovetic1@gmail.com

Abstract: Today, arrhythmia is one of the diseases that can be easily diagnosed, but also successfully treated with the necessary medication. The use of artificial intelligence (AI) in medicine speeds up the process of identifying diseases. Therefore, this paper will present some of the main algorithms in the field of machine learning (ML) that can be used to draw various conclusions, as well as diagnose diseases. Although there is no optimal algorithm that will bring the result with 100 percent accuracy, we have tools that speed up our decisions. I have used five algorithms in the programming language Python, of which the best results were with SVC and Gradient Boosting. The main problem with machine learning in this dataset is mostly uneven data and too much dominance of one class.

Keywords: Accuracy, Algorithms, Arrhythmia, Machine Learning, Python.

1. Introduction

Artificial intelligence allows us to develop tools for rapid assessment and prediction of future conditions. This is the main reason why it has become increasingly popular in medicine. This project is primarily focused on cardiac arrhythmia and its diagnoses.

The Electrocardiogram (ECG) is the electrical representation of the contractile activity of the heart [1], and it is one of the most important techniques in cardiology because it allows for following different waveforms as signals from the heart. The ECG contains record from six limb leads, which are written in characters as I, II, III, aVR, aVL, aVF, and from six chest leads written as V1, V2, V3, V4, V5, V6 [2]. Although AI is slowly beginning to be used in the field of cardiology, expert cardiologists are still necessary to diagnose the disease. The use of AI would speed up the process of diagnosing disease and treating it, but in some cases, it might also allow for a more precise definition of the patient's condition.

There are a lot of papers on this topic and they are mostly based on varying applications of different algorithms. For example, Chan et al. (2020) [2] used convolution neural network on a dataset with 6.877 records and got an F1-score of 0.84. Guvenir et al. (1997) [3] used the VF15 algorithm and got a result of 62 % accuracy.

In this paper, will present five machine learning algorithms that could be used to diagnose cardiac arrhythmia. The algorithms used are as follows: Decision Trees, Random Forest, Gradient Boosting, Support Vector Machine (C-Support Vector Classification) and K-Nearest Neighbour. I tried to show that we can speed up the process of detecting diseases in medicine, with an important prerequisite, which is a large set of data with different classifications.

2. Methodology

The dataset that is used is from the website UCI Machine learning repository [4]. It consists of 452 instances and 279 attributes. The distribution of data is uneven as shown in Table 1. The main aim is to distinguish between the normal condition and types of cardiac arrhythmia, and then to classify it in one of the 16 groups [3]. Data pre-processing before applying machine learning methods is an important step. It was noticed that there are some fields inside the columns (feature, variables) where data was missing and the "?" sign was presented. All of these characters have been replaced with NaN values so as to allow for statistical calculation i.e., all empty fields have been replaced with the average value of that column. These instances occurred in 5 columns. Subsequently, the data filtering was done in such a way that if it is noticed that 99 percent or more of the values in one column are the same, it means that this column does not provide relevant information and it has to be dropped. Number of columns that were deemed unimportant is 64 and they have been dropped. An important step was to also scale all the data in unit variance with the statistical StandardScaler() method.

The application of algorithms is made on two types of datasets. The first set consisted of data on which the specified filtering was done, while additional filtering with correlation was done on the second set of data. Everywhere the correlation between the feature and target class was lower than 0.1, that feature was dropped from the set. By applying this filtration, I dropped 157 columns. For machine learning five different algorithms were used, from library scikit-learn, to solving problems of a multi-classification nature.

The first algorithm is Decision Trees. It has a leaf node labelled with class or with a structure consisting of a test node linked to two or more subtrees [5]. A test node calculates results based on features and make decisions on which side to go [6]. If a leaf node does not have subtrees then it provides the predicted class. The main goal of the decision tree is for a sample to be sorted into a class using one or more decision functions [7].

The next algorithm is Random Forest. As observed by Oshiro et al. (2012) [8], it is a method which constructs many decisions trees which will be used to classify a new instance by a majority vote. Each node in a tree uses a subset of the best features which is randomly selected and makes a new node [9].

K-Nearest Neighbour is a popular supervised algorithm in classification because of its simple implementation and significant classification performance. Outputs are vectors, and they are placed in space. Following that, the new entry sample will be classified based on the classification of its neighbours [10].

According to Guelman (2012) [11] the Gradient Boosting is an iterative supervised ML algorithm that combines simple parameterized functions with poor performance to produce a highly accurate prediction rule. It is very important that datasets go through data pre-processing before creating a model. Also, the base estimator is usually Decision Trees.

C-Support Vector is a part of Support Vector Machine algorithms where the aim is to find hyperplane in N-dimensional space. Its benefit is in choosing the right kernel for different problems [12].

Each algorithm was used in the same way. Firstly, I separated the datasets into train (75 %) and test (25 %) sets. Then, the GridSearchCV() method was used for hyperparameter tuning and choosing a set of optimal hyperparameters for learning algorithm.

Table 1. *Class distribution of dataset*

Class	Num. of instances	ratio
Normal	245	0.54
Ischemic changes	44	0.09
Old Anterior Myocardial Infarction	15	0.03
Old Inferior Myocardial Infarction	15	0.03
Sin. tachycardy	13	0.03
Sin. bradycardy	25	0.06
PVC	3	0.007
Supraventricular Premature Contraction	2	0.004
Left bundle branch block	9	0.02
Right bundle branch block	50	0.11
1. degree AV block	0	0
2. degree AV block	0	0
3. degree AV block	0	0
Left ventricule hypertrophy	4	0.009
Atrial Fibrillation or Flutter	5	0.01
Others	22	0.05

3. Results and Discussion

As the output of the measurability of the algorithm, I calculated accuracy, F1 score, precision and recall score confusion matrix and ROC-AUC score. Figure 1. shows the result of the ROC-AUC score for each algorithm, separating the correlation dataset and the normal dataset. The Gradient Boosting Classifier model has the best score, which is 0.7967 in the normal dataset and its accuracy is 67.26 %. The C-Support Vectore Classification has a 0.7895 in dataset with the correlation filter, and its accuracy is 63.72 %. The lowest performing model was built with the KNN algorithm where with the normal dataset, it has a 56.64 % accuracy and the ROC-AUC score of 0.67, while with the correlation dataset, the accuracy is 53.98 % and the ROC-AUC is 0.72.

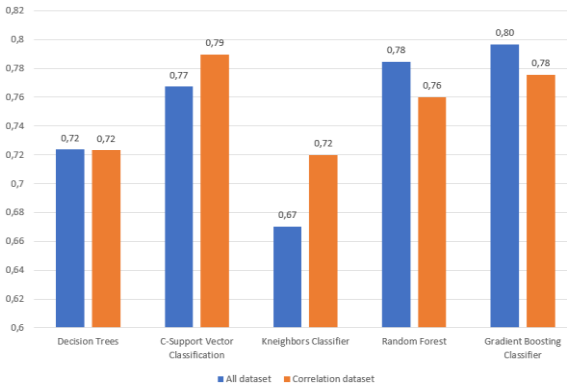


Figure 1. *ROC-AUC scores for all algorithms*

The F1 score is one of the common measures to rate how successful a classifier is, and it is a combination of precision and recall metrics. For an average parameter, I used *'macro'* due to the uneven distribution data. As can be seen in Figure 2, the results are almost the same, however, two algorithms have minimal benefits. The model from the Gradient Boosting Classifier has the best score from 0.3690 in normal dataset, while the Random Forest has, in dataset with the correlation filter, 0.2931. The worst model was built with the KNN algorithm. All these results are very bad as there was no model that has been able to train to have high precision in inference.

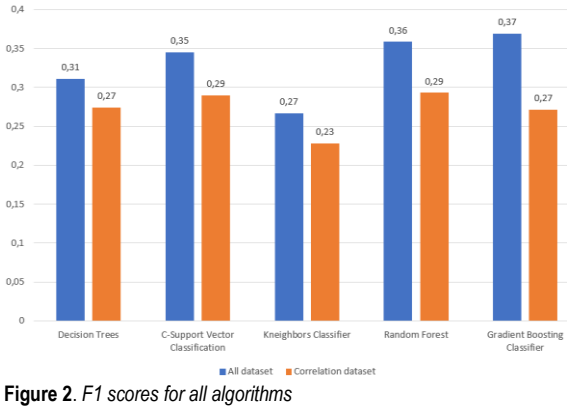


Figure 2. *F1 scores for all algorithms*

4. Conclusion

The problem of poor performance in this study is primarily related to the dataset. There is always the option of artificially expanding an already existing dataset but in this case, it is not good option. There is too much data concentration in one class (Normal class), while the others have little to no data. Therefore, the model fails to recognize other classes due to there not being enough samples to learn which features result with other class.
The three algorithms that stood out during the process are Gradient Boosting, C-Support Vector and Random Forest. The results would certainly be even better if a larger dataset was used.

Acknowledgments

I thank assistant dr. sc. Ivan Lorencin for his mentorship during the work on this article.

Reference

[1] Isin, Ali, and Selen Ozdalili. "Cardiac arrhythmia detection using deep learning." Procedia computer science 120 (2017): 268-275.
[2] Chen, Tsai-Min, et al. "Detection and classification of cardiac arrhythmias by a challenge-best deep learning neural network model." Iscience 23.3 (2020): 100886.
[3] Guvenir, H. Altay, et al. "A supervised machine learning algorithm for arrhythmia analysis." Computers in Cardiology 1997. IEEE, 1997.
[4] "Arrhythmia Data Set", from Internet, https://archive.ics.uci.edu/ml/datasets/arrhythmia, 20. May 2022.
[5] Quinlan, J. Ross. "Learning decision tree classifiers." ACM Computing Surveys (CSUR) 28.1 (1996): 71-72.
[6] Podgorelec, Vili, et al. "Decision trees: an overview and their use in medicine." Journal of medical systems 26.5 (2002): 445-463.
[7] Swain, Philip H., and Hans Hauska. "The decision tree classifier: Design and potential." IEEE Transactions on Geoscience Electronics 15.3 (1977): 142-147
[8] Oshiro, Thais Mayumi, Pedro Santoro Perez, and José Augusto Baranauskas. "How many trees in a random forest?." International workshop on machine learning and data mining in pattern recognition. Springer, Berlin, Heidelberg, 2012.
[9] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.
[10] Mucherino, Antonio, Petraq J. Papajorgji, and Panos M. Pardalos. "K-nearest neighbor classification." Data mining in agriculture. Springer, New York, NY, 2009. 83-106.
[11] Guelman, Leo. "Gradient boosting trees for auto insurance loss cost modeling and prediction." Expert Systems with Applications 39.3 (2012): 3659-3667.
[12] Novakovic, J., and A. Veljovic. "C-support vector classification: Selection of kernel and parameters in medical diagnosis." 2011 IEEE 9th international symposium on intelligent systems and informatics. IEEE, 2011.

ISBN: 978-953-8246-26-5

Country-level factor influence on COVID-19 excess mortality rates determined using Random Forest algorithm

Sandi BARESSI ŠEGOTA[1*], Anđela BLAGOJEVIĆ[2,3], Tijana ŠUŠTERŠIČ[2,3], Jelena MUSULIN[1], Daniel ŠTIFANIĆ[1], Matko GLUČINA[4], Ivan LORENCIN[1], Nikola ANĐELIĆ[1], Nenad FILIPOVIĆ[2,3], Zlatan CAR[1]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, sbaressisegota@riteh.hr, jmusulin@riteh.hr, dstifanic@riteh.hr, ilorencin@riteh.hr, nandelic@riteh.hr, car@riteh.hr
[2] Faculty of Engineering, University of Kragujevac, Sestre Janjić, 34000 Kragujevac, Serbia, tijanas@kg.ac.rs, andjela.blagojevic@kg.ac.rs, fica@kg.ac.rs
[3] Bioengineering Research and Development Centre (BioIRC), Prvoslava Stojanovića 6, 34000 Kragujevac, Serbia
[4] University of Rijeka, Trg Braće Mažuranića 10, 51000 Rijeka, Croatia, matko.glucina@riteh.hr

Abstract: The reaction to and the effect severity on the population of COVID-19 varied across different countries. This indicates that there are country-level factors which influenced the severity COVID-19 effect on the populace. The goal of this research is to determine some of these factors using Our World in Data COVID-19 dataset. The performance of the country is evaluated using excess mortality (EM), while the inputs selected for the analysis are stringency index (SI), population (P), population density (PD), median age (MA), percent of population aged 65 or older (P65), percent of population aged 70 or older (P70), GDP per capita (GDP), percentage of population living in extreme poverty (EP), death rate from cardiovascular diseases (CDR), prevalence of diabetes withing population (DP), percentage of female smokers (FS), percentage of male smokers (MS), availability of handwashing facilities (HWF), hospital beds available per thousand (HB), life expectancy (LE), and human development index (HDI). The inputs are evaluated using Random Forest (RF) algorithm via two metrics – Mean Decrease in Impurity (MDI) and Feature Permutation (FP). Results show that the SI is the most influential factor when evaluated using RF.

Keywords: artificial intelligence, COVID-19, coronavirus modelling, feature importance, feature permutation machine learning, mean decrease in impurity, random forest regressor

1. Introduction

COVID-19 pandemic has gripped the world since early 2020, causing widespread issues relating to the disease cause by the SARS-COV-2 virus. While the influence of the virus is still present, with infections continuing to happen, the power the virus once held over people is lowering. Still, the raise of a new pandemic in the future is only a matter of time – and it is important to note that the data on how countries dealt with the coronavirus pandemic is key on determining which policies have shown to be appropriate and which of the countries may be more vulnerable based on certain country level factors. Artificial intelligence has been shown to be one of the key techniques used in modelling the COVID-19 pandemic [1, 2], with various techniques being applied – from neural networks [3] and genetic programming [4, 5] and tree-based algorithms [6, 7].

One of the algorithms that have shown to perform well in determining the influence of the individual parameters on the outputs is the RF algorithm [8, 9]. This research will apply the RF algorithm on the publicly available Our World in Data COVID-19 dataset [10].

2. Methodology

First step is the processing of the dataset. The input factors of the dataset are selected as they follow: Stringency Index – which is a mix of 9 measures that define how stringent were the applied measures to deal with the pandemic at a country level (SI) [11], population (P), population density (PD), median age (MA), percent of population aged 65 or older (P65), percent of population aged 70 or older (P70), GDP per capita (GDP), percentage of population living in extreme poverty (EP), death rate from cardiovascular diseases (CDR), prevalence of diabetes withing population (DP), percentage of female smokers (FS), percentage of male smokers (MS), availability of handwashing facilities (HWF), hospital beds available per thousand (HB), life expectancy (LE), and human development index (HDI) [12]. The output used to determine the success of the country dealing with the pandemic is excess mortality (EM) – number of deaths from all causes which are higher then the expected during an event (in this case the COVID-19 pandemic) [13]. The data is processed by removing values inside the rows that did not contain the listed values. This results in a dataset with 725 data points.

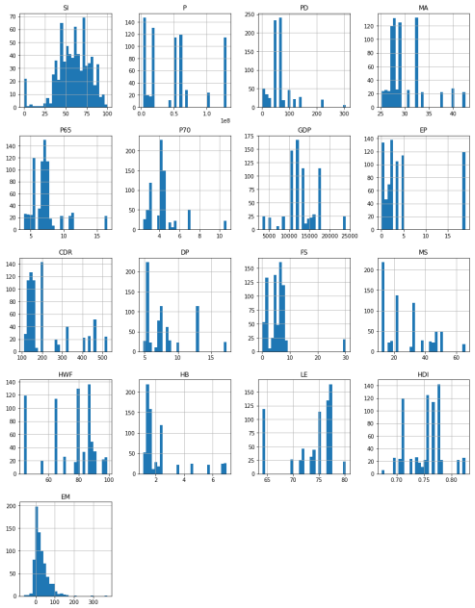


Figure 1. *Histograms of the data within the dataset*

The distributions of each of the variables are given in Figure 1, with the output variable being separated in the last row. The figure shows that the data has different

ranges which means that the dataset needs to be scaled prior to its use in the further research.
Figure 2 shows the correlation between each of the inputs to the output variable.

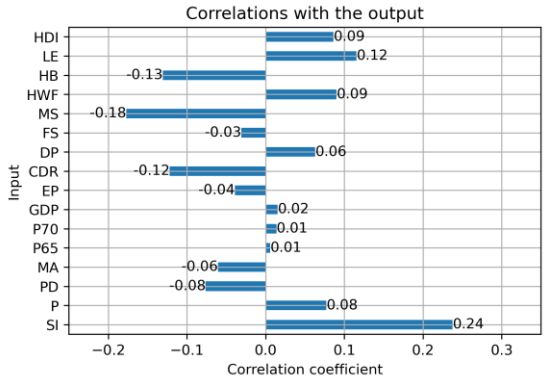


Figure 2. *Correlation coefficients between the inputs and the output variable.*

Correlation coefficients show that the SI and LI have the highest positive coefficient to the EM, with the MS, HB and CDR having the highest negative coefficient correlation.
The feature importance analysis is performed using a RF regressor. RF regressor will consist of many decision trees (in the presented case, this number is 10,000) that will each aim to regress the output value. As each of these models is a so-called open box model, the influences of the individual features in each of the decision trees can be analysed. Then we can evaluate the importance of each feature by determining how much more precise the trees that used the feature in question have been, compared to the ones without – or how much did the model error when a certain feature was used. This method is called Mean impurity decrease (MDI), as it calculates the lowering of the impurity (error) of the model [14].
To confirm the results obtained using MDI evaluation another approach was also used. In this approach the feature importance is determined by comparing the model trained with a dataset as-is, to a model trained with a dataset in which one of the inputs was randomly shuffled. If the score is significantly lower when using a dataset with a single random shuffle, it indicates that the feature in question has a high-performance influence. On the other hand, if the shuffling of the feature has no influence on the performance than the indication is that the feature does not matter within the trained model. This approach is referred to as Feature Permutation (FP) [15].

3. Results and Discussion

Figures 3. And 4. Demonstrate the results obtained using the RF, measured by MDI and FP respectively. As it can be seen in the figure 3. the main influence when determined using MDI is SI, by a large margin. The second feature with the highest performance is EP. None of the other feature stand out with any significance as other inputs show a similar, relatively low, importance when evaluated for feature importance using the RF algorithm with the MDI metric.

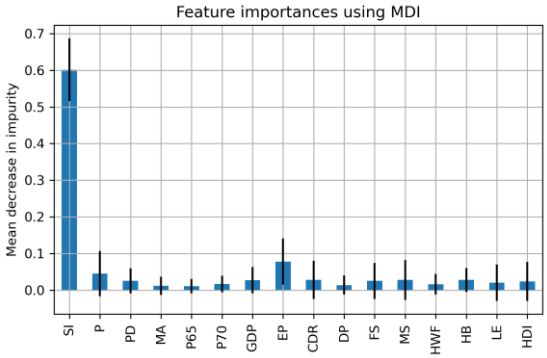


Figure 3. *Feature importance determined using MDI metric*

The data shown in figure 4., the feature importance using FP metric, is in agreeance with the MDI metric – confirming the obtained results.

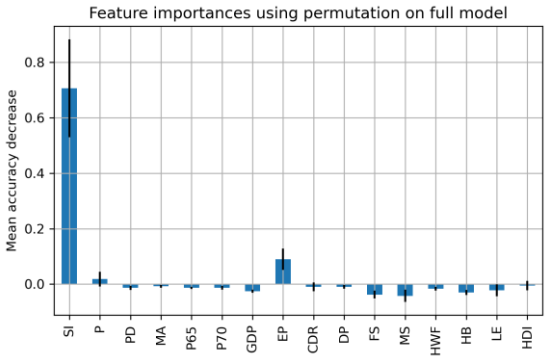


Figure 4. *Feature importance determined using FP metric*

4. Conclusion

The calculated models show the high importance of EP and SI factors relative to the other analysed inputs. Still, the obtained results need to be considered with the data in mind. It is not likely that the higher SI causes a higher EM – but the opposite may be truth, where the higher EM causes the decision makers to raise the stringency of the anti-covid measures. On the other hand, other factors which show a higher importance may influence the EM as shown.
The sort of analysis performed in the presented research has value, as it can suggest which of the factors should be addressed by decision makers in the case of a pandemic. Still, a deeper analysis needs to be performed before taking the results at face value – as other, unconsidered factors, may have important influence, or the dynamic nature of decision making within a pandemic may cause the output to influence the values which were used as inputs.
Future work in this field may be considered, especially focusing on previously mentioned limitations – through a deeper analysis of the existing factors and the inclusion of other values that may influence – such as health care index of a given country.

Acknowledgments

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, University of Rijeka scientific grant uniri-tehnic-18-275-1447.

References

[1] Musulin, Jelena, et al. "Application of artificial intelligence-based regression methods in the problem of covid-19 spread prediction: A systematic review." *International Journal of Environmental Research and Public Health* 18.8 (2021): 4287.
[2] Blagojević, Anđela, et al. "Use of Regressive Artificial Intelligence and Machine Learning Methods in Modelling of COVID-19 Spread (COVIDAi): Project Review." *Ri-STEM-2021* (2021): 1.
[3] Car, Zlatan, et al. "Modeling the spread of COVID-19 infection using a multilayer perceptron." *Computational and mathematical methods in medicine* 2020 (2020).
[4] Anđelić, Nikola, et al. "Estimation of COVID-19 epidemic curves using genetic programming algorithm." *Health informatics journal* 27.1 (2021): 1460458220976728.
[5] Anđelić, Nikola, et al. "Estimation of covid-19 epidemiology curve of the united states using genetic programming algorithm." *International Journal of Environmental Research and Public Health* 18.3 (2021): 959.
[6] Šušteršič, Tijana, et al. "Epidemiological Predictive Modeling of COVID-19 Infection: Development, Testing, and Implementation on the Population of the Benelux Union." *Frontiers in public health* (2021): 1567.
[7] Blagojević, Anđela, et al. "Artificial intelligence approach towards assessment of condition of COVID-19 patients-Identification of predictive biomarkers associated with severity of clinical condition and disease progression." *Computers in biology and medicine* 138 (2021): 104869.
[8] Baressi Šegota, Sandi et al. "Cancer Rates per Country - Determining the Importance of Country Level Factors using Random Forest Regressor" 1st Serbian Conference on Applied Artificial Intelligence (2022)
[9] Zhao, Yifan, et al. "Classification of Zambian grasslands using random forest feature importance selection during the optimal phenological period." *Ecological Indicators* 135 (2022): 108529.
[10] Mathieu, Edouard, et al. "A global database of COVID-19 vaccinations." *Nature human behaviour* 5.7 (2021): 947-953.
[11] Gros, Daniel, Alexandre Ounnas, and Timothy Yu-Cheong Yeung. "A new COVID policy stringency index for Europe." *Covid Economics* (2021): 115.
[12] Ladi, Tahmineh, Asrin Mahmoudpour, and Ayyoob Sharifi. "Assessing impacts of the water poverty index components on the human development index in Iran." *Habitat International* 113 (2021): 102375.
[13] Rivera, Roberto, Janet E. Rosenbaum, and Walter Quispe. "Excess mortality in the United States during the first three months of the COVID-19 pandemic." *Epidemiology & Infection* 148 (2020).
[14] Han, Hong, Xiaoling Guo, and Hua Yu. "Variable selection using mean decrease accuracy and mean decrease gini based on random forest." 2016 7th ieee international conference on software engineering and service science (icsess). IEEE, 2016.
[15] Altmann, André, et al. "Permutation importance: a corrected feature importance measure." Bioinformatics 26.10 (2010): 1340-1347.

ISBN: 978-953-8246-26-5

On Cervical Cancer Diagnostics Using Machine Learning

Matko GLUČINA [1*], Ariana LORENCIN [2], Ivan LORENCIN[3]

[1] University of Rijeka, Braće Mažuranića Square 10, 51000, Rijeka, Croatia, email: matko.glucina@uniri.hr
[2] Clinical Hospital Center Rijeka, Krešimirova ul. 42, 51000, Rijeka, email. ariana.rabac@gmail.com
[3] University of Rijeka, Faculty of Engineering, Vukovarska ul. 58, 51000, Rijeka, email: ilorencin@riteh.hr

Abstract: This paper presents the classification of cervical cancer using the Multilayer Perceptron into a publicly available dataset Cervical cancer (Risk Factors) taken from the UCI Machine learning repository consisting of 36 attributes with 859 instances. The target values were Hinselmann, Schiller,Cytology, and biopsy by processing the dataset and using the GridSearch method, the MLP algorithm gives a high percentage of accuracy for all four outputs, but an insight into other metrics shows that additional research elaboration is needed. based on the f1 score the best result for Hinselmann is 0.17, Schiller in the amount of 0.4, Cytology 0.29, and Biopsy 0.19 which indicates more than enough of a poor-quality model for implementation in real conditions.

Keywords: Artificial intelligence, Cervical cancer, Machine learning, Multilayer perceptron, Neurons.

1. Introduction

Cervical cancer is in most cases cell carcinoma resulting from infection with human papillomavirus or adenocarcinoma. This type of cancer is the third most common gynecological cancer. The average age at the time of diagnosis is about 50 years but also can occur in the 20s. Risk groups for cervical cancer are mostly younger women who frequently change partners, have early sexual intercourse, people with a history of HPV infection, and smoking. In most cases, there are no major symptoms until cancer has progressed. The first symptom of advanced cancer is usually postcoital vaginal bleeding. The diagnosis is made using the Papanicolaou test, biopsy (colposcopy), and finally cytologic analysis. Stages are determined clinically most often by FIGO guidelines. Treatment usually includes surgical resection, radiation, and chemotherapy. Healing and prognosis depend on the stage of cancer and the treatment measures taken [1,2]. There have been studies that have implemented ML for diagnostics, ranging from bladder cancer [3,4] to COVID-19 [5-7]. In this paper, a diagnostic approach based on the utilization of machine learning (ML) methods is proposed. The main question in this research is: is it possible to implement ML methods, mainly artificial neural networks (ANNs) Is it the diagnostics of cervical cancer? In the section methodology, a brief description of the used data set and developed algorithms is given. Furthermore, information about targeted hyper-parameters is provided. After the methodology, achieved results are presented and discussed. In the end, a conclusion is derived.

2. Methodology

Multilayer Perceptron (*MLP*) is a neural network that contains the ability to learn the relationship between linear and nonlinear elements it is inspired by the structure of the human brain and is one of the ML methods that is trained using a supervised learning approach. The human brain is a complex system that is virtually impossible to model, much information in human brain research is unknown but nevertheless serves as an inspiration in many scientific fields for its ability to develop intelligence [8].

During the development of the *MLP* algorithm itself, every scientific advancement in the field of deep learning did not start with demanding structures, on the contrary, it started with simpler structures, i.e., in this case, it started from a piece of *MLP* called a neuron [9]. As mentioned earlier, the human brain is made up of different types of cells called neurons. Each of them is interconnected and thus forms a neural network. When modeling the Artificial Neural Network (*ANN*), the chemical process, neuron frequencies, various brain features, and many other things that are not understood or explained by medicine are ignored. Considering before mentioned claims, Figure 1. shows a basic neuron consisting of several parts among which are: dendrites, cell body, axon hillock, and axon.

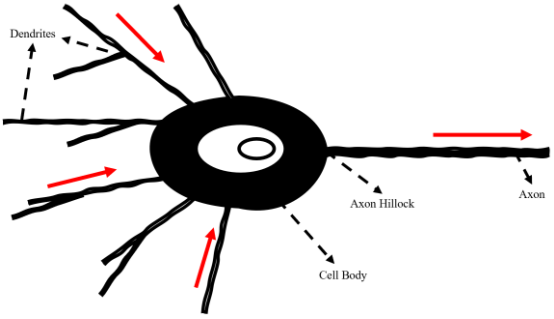


Figure 1. *Natural neuron scheme*

The multilayer perceptron has many neurons inside of its structure, it has feed-forward *ANN* with the characteristic of multiple inputs to the algorithm, while the output value is only one. As shown in Figure 2, the given statement is explained [10]. Fascinating results in the field of classification and regression can be achieved by using *MLP*. In addition to a detailed analysis of each individual neuron, proper parameterization can provide quality models that can be useful [11].

In the last few decades, the number of cervical cancer patients in developed countries, including Croatia, has been declining. However, according to Global Cancer Observatory (*GLOBOCAN*) estimates for 2020, cervical cancer is still in the high fourth place in terms of frequency (6.5% of all cancer cases) and mortality (7.7% of all

cancer deaths) in women worldwide [12]. Therefore, it is extremely important to pay attention to this problem for the early prevention of cervical cancer. For this purpose, a dataset [13] from the UCI Machine learning repository was used for this research. The data set consisted of 36 variables collected from 858 patients. The variables are as follows: age, number of sexual partners, first sexual intercourse (age), number of pregnancies, smokes, smokes (years), smokes (packs/year), hormonal contraceptives, hormonal contraceptives (years), IUD, IUD (years), STDs, STDs (number), STDs:condylomatosis, STDs: cervical condylomatosis, STDs: vaginal condylomatosis, STDs:vulvo-perineal condylomatosis, STDs: syphilis, STDs: pelvic inflammatory disease, STDs: genital herpes, STDs: molluscum contagiosum, STDs: AIDS, STDs: HIV, STDs: hepatitis B, STDs: HPV, STDs: number of diagnoses, STDs: time since the first diagnosis, STDs: time since the last diagnosis, Dx: Cancer, Dx: CIN, Dx: HPV, Dx, Hinselmann, Schiller, cytology, biopsy. Where the target variables are Hinselmann, Schiller, cytology, and biopsy while the rest are the values of the input variables.

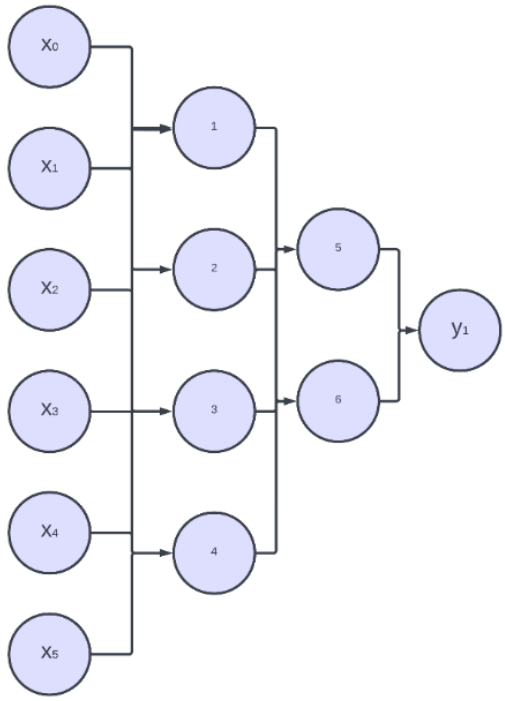


Figure 2. *MLP neural network with three inputs and a single output with a hidden layer that contains three neurons*

Colposcopy is a medical procedure that observes the cervix, vagina, and vulva under high magnification using a specially designed moving microscope called a colposcope. Such a method of diagnosis is called Heinselmann diagnostics (first to describe colposcope). There is also a Schiller diagnosis where by applying Lugol's solution (iodine solution in potassium iodate) glycogen from surface cells binds iodine from the solution to the surface of the squamous epithelium, and the epithelium turns dark brown - known as iodine positive reaction which excludes with the existence of epithelial cell atypia. The abnormal epithelium is usually white and does not receive iodine - these are the so-called iodine-negative changes. The biopsy is a method of diagnosing cervical cancer that takes a piece of tissue that we send for cytological analysis. The cytological analysis includes the examination of cervical cells under a microscope and determination of cervical cancer as well as the stage of the disease [14].

However, the dataset is not complete, in the translation of individual patients no data were available and for this reason, it was necessary to "clean" the dataset in order to train the *MLP* algorithm. By deleting rows that were not complete in the sense that they did not contain all the data in each column they were deleted thus reducing the dataset to 669 patients with all data. *GridSearch* (*GS*) was used for more precise results, and *Cross Validation* (*CV*) was used to confirm the output metrics, which legitimizes the result [15]. The parameter space used in the *GS* parameter optimization method to obtain a consistent solution is listed in Table 1.

Table 1. *GS Hyperparameter space used in this research*

Hyperparameter	Value
Number of hidden layers	(50,50,50), (50,100,50), (100,10),(5,5,5)
Activation	'identity', 'logistic', 'tanh', 'relu'
Solver	'lbfgs', 'sgd', 'adam'
Alpha	[0.0001, 0.05,0.001,0.1,0.008,0.5
Learning rate	'constant','adaptive','invscaling'
Tolerantion	$1x10^{-4}$
Maximum iteration number	200,400,1000,2000

When running the program code of the algorithm, five *CVs* are validated. This confirms the validity obtained results by avoiding the overfitting problem.

3. Results and Discussion

The results achieved by MLP application are presented in Table 2, Table 3, Table 4, and Table 5. Table 2 represents the output for Hinselmann, Table 3 for Schiller, Table 4 for Cytology, and Table 5 for Biopsy. Before the presentation of the results, the values evaluated are the following mean value accuracy, the standard deviation of accuracy, mean value of f1 score, the standard deviation of f1 score, the mean value of ROC-AUC metric, and the standard deviation of ROC-AUC metric.

Table 2*. Results from GS for Hilsemann output*

Result metric	1st result	2nd result	3rd result
Mean test accuracy	0.95	0.93	0.93
Std test accuracy	0.02	0.04	0.02
Mean_test_f1	0.17	0.16	0.15
Std test f1	0.45	0.27	0.4
Mean test roc auc	0.55	0.44	0.64
Std test roc auc	0.29	0.23	0.22

It can be seen from Table 2 that the results for Hilsemann vary in the vicinity of some areas. Accuracy is around 0.93, its deviation is around 0.03, f1 score is around 0.16

and deviation f1 is around 0.4, RoC has values highest 0.64 while minimum 0.44.. Hyper parameters used for best result were: 'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (5, 5, 5), 'learning_rate': 'adaptive', 'max_iter': 400, 'solver': 'sgd', 'tol': 0.0001}

***Table 3**. Results from GS for Schiller output*

Result metric	1st result	2nd result	3rd result
Mean test accuracy	0.9	0.94	0.91
Std test accuracy	0.02	0.02	0.04
Mean_test_f1	0.14	0.13	0.12
Std test f1	0.14	0.53	0.22
Mean test roc auc	0.56	0.66	0.57
Std test roc auc	0.1	0.19	0.31

The results for Schiller are slightly more diverse compared to Hilsemann, in all three cases different deviations in the first and second result are 0.02, f1 score has different value in all three cases and deviations are smaller. the best hyperparameters were: 'activation': 'tanh', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50, 50), 'learning_rate': 'adaptive', 'max_iter': 1000, 'solver' : 'lbfgs', 'tol': 0.0001.

***Table 4**. Results from GS Citology output*

Result metric	1st result	2nd result	3rd result
Mean test accuracy	0.86	0.88	0.85
Std test accuracy	0.07	0.09	0.07
Mean_test_f1	0.29	0.26	0.24
Std test f1	0.2	0.24	0.27
Mean test roc auc	0.53	0.65	0.55
Std test roc auc	0.23	0.21	0.23

Looking at Table 4 it is possible to see the metrics for the Cytology output. The accuracy is lower compared to the previous two outputs, but nevertheless, the f1 score is higher, but unfortunately, the ROC-AUC is lower compared to the previous two cases. Best parameters for Cytology output were: 'activation': 'logistic', 'alpha': 0.05, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'constant', 'max_iter': 2000, 'solver': 'lbfgs', 'tol': 0.0001.

***Table 5**. Results from GS Biopsy output*

Result metric	1st result	2nd result	3rd result
Mean test accuracy	0.91	0.94	0.94
Std test accuracy	0.04	0.03	0.03
Mean_test_f1	0.19	0.13	0.13
Std test f1	0.26	0.53	0.53
Mean test roc auc	0.57	0.62	0.63
Std test roc auc	0.17	0.26	0.25

Comparing the previous three cases, the *GS* method gave the best results for Biopsy output, but they are still not nearly as good as this model. Accuracy is around 0.9, accuracy deviation is around 0.04, f1 score is quite low and its deviation is quite high, ROC-AUC values are still insufficient as well as its deviations. Best hyperparameters were: 'activation': 'tanh', 'alpha': 0.008, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'adaptive', 'max_iter': 400, 'solver': 'lbfgs', 'tol': 0.0001

It is evident for all four output values that the results are not in line with expectations. Accuracy is the highest metric but looking at other metrics as an f1 score, RoC-AuC shows that the results are quite low. Looking at the trend of growth and decline of metrics, it is evident that the results do not tend to improve the output metrics.

4. Conclusion

Based on the given results and the implementation of the research the results are unsatisfactory. Accuracy is high in all four cases, but one metric is not sufficient to accept this type of model. There are potentially several reasons why this is the case. One of them is the quality of the dataset, i.e. the dataset consists of many bool values that amount to class zero. The algorithm automatically does not have enough data to be able to train according to satisfactory values. In addition to the above reason, the potential problem may be the selection of the wrong parameters, given the limited computer resources, select those parameters that the local computer can withstand without stopping the training process of the algorithm. Further research can be improved by increasing the dataset with better quality data and training the algorithm on a high-performance computer, We encourage future researchers to choose other methods besides the *MLP* method such as Support Vector classification or any other classification algorithms. The answer to the given question from the introduction is: the algorithm did not meet the stated goal and the model is not of sufficient quality for implementation, the quality of the dataset greatly affects the final output, and selecting multiple input variables does not guarantee the safety of obtaining a quality model.

Acknowledgments

This research has been (partly) supported by the CEEPUS network Ciii-HR-0108, European Regional Development Fund under the grant K.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447.

Reference

[1] Petignat, Patrick, and Michel Roy. "Diagnosis and management of cervical cancer." Bmj 335.7623 (2007): 765-768.

[2] Cohen, Paul A., et al. "Cervical cancer." The Lancet 393.10167 (2019): 169-182.

[3] Lorencin, Ivan, et al. "Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis." Artificial Intelligence in Medicine 102 (2020): 101746.

[4] Ikeda, Atsushi, et al. "Support system of cystoscopic diagnosis for bladder cancer based on artificial intelligence." Journal of endourology 34.3 (2020): 352-358.

[5] Lorencin, Ivan, et al. "Automatic evaluation of the lung condition of COVID-19 patients using X-ray images and convolutional neural networks." Journal of Personalized Medicine 11.1 (2021): 28.

[6] Blagojević, Anđela, et al. "Artificial intelligence approach towards assessment of condition of COVID-19 patients-Identification of predictive biomarkers associated with

severity of clinical condition and disease progression." Computers in biology and medicine 138 (2021): 104869.

[7] Vaishya, Raju, et al. "Artificial Intelligence (AI) applications for COVID-19 pandemic." Diabetes & Metabolic Syndrome: Clinical Research & Reviews 14.4 (2020): 337-339.

[8] Taud, Hind, and J. F. Mas. "Multilayer perceptron (MLP)." *Geomatic approaches for modeling land change scenarios*. Springer, Cham, 2018. 451-455.

[9] Zorins, Aleksejs, and Peteris Grabusts. "Artificial neural networks and human brain: Survey of improvement possibilities of learning." *ENVIRONMENT. TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference*. Vol. 3. 2015.

[10] Car, Zlatan, et al. "Modeling the spread of COVID-19 infection using a multilayer perceptron." *Computational and mathematical methods in medicine* 2020 (2020)..

[11] Baressi Šegota, Sandi, et al. "Frigate speed estimation using CODLAG propulsion system parameters and multilayer perceptron." *NAŠE MORE: znanstveni časopis za more i pomorstvo* 67.2 (2020): 117-125.

[12] Nicula, Florian AI, et al. "Challenges in starting organised screening programmes for cervical cancer in the new member states of the European Union." *European journal of cancer* 45.15 (2009): 2679-2684.

[13] Cervical cancer (Risk Factors) Data Set https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29

[14] Tsikouras, Panagiotis, et al. "Cervical cancer: screening, diagnosis and staging." J buon 21.2 (2016): 320-325.

[15] Lorencin, Ivan, et al. "On urinary bladder cancer diagnosis: Utilization of deep convolutional generative adversarial networks for data augmentation." Biology 10.3 (2021): 175.

ISBN: 978-953-8246-26-5

Predicting fertility problems in men using MLP and SVM

Antonia Burić[1*], Anja Režek[2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, aburic@riteh.hr.
[2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, arezek@riteh.hr.

Abstract: The purpose of this article is to predict fertility problems in men using previously given dataset and two classifier methods, MLP and SVM. Some of Python libraries which have been used during this research and are worth mentioning are sklearn.neural_network, sklearn.model_selection and sklearn.metrics. The neural network architecture which was used is a multilayer perceptron (MLP) and a *support-vector machines* (*SVM), and for training is* used trial and error method, in that method parametars are randomly chosen for code implementation. Also, the fertility quality is evaluated using AUC ROC metrics. After implementation it can be concluded that training takes longer with increasing the number of neural networks hidden layers. However, this reduces the potential error. After working on this project, satisfactory results were obtained with an accuracy of 87% MLP method and 93% SVM method. In conclusion, the statement was made that the initial hypothesis was fulfilled and that the SVM method achieves the better results faster and easier than the other method.

Keywords: Artificial intelligence, Machine Learning, MLP, SVM, AUC ROC metrics, Multilayer Perceptron

1. Introduction

The problem to be solved is the prediction of men's fertility using a multi-layered perceptron. The research [3] showed that fertility rates have dramatically decreased in the last two decades, especially in men. It has been described that environmental factors, as well as life habits, may affect semen quality. Artificial intelligence techniques are now an emerging methodology as decision support systems in medicine. By correctly adjusting the parameters of the multilayer perceptron, more accurate results are obtained.
The data used in this project contains ten groups of parameters, which are:

- Season in which the analysis was performed
- Age at the time of analysis
- Childish diseases
- Accident or serious trauma
- Surgical intervention
- High fevers in the last year
- Frequency of alcohol consumption
- Smoking habit
- Number of hours spent sitting per day
- Output, Diagnosis normal (N) or altered (O)

Based on the given data, the main goal is to train the neural network to predict fertility of men with as much accuracy as possible. The assumption is that better results will be obtained by the SVM method.

2. Methodology

The neural network must be viewed as a system with input and output data. The multilayer perceptron is the most known and most frequently used type of neural network.
This architecture is called feedforward (Fig.1). Multilayer perceptron is a type of a feed-forward neural network characterized by a single output, multiple inputs and one or more hidden layers [8]. Figure 1. explains the architecture in more detail: Multilayer perceptron is a neural network learning algorithm with a teacher (supervised learning) that learns a function
f(·): R^m→R^o
on a given set of data where m represents the number of input dimensions, and o represents the number of output dimensions. For a given set of input data: X= x_1, x_2, x_m and a given set of output data y, a neural network can learn an approximate nonlinear function for classification or regression.

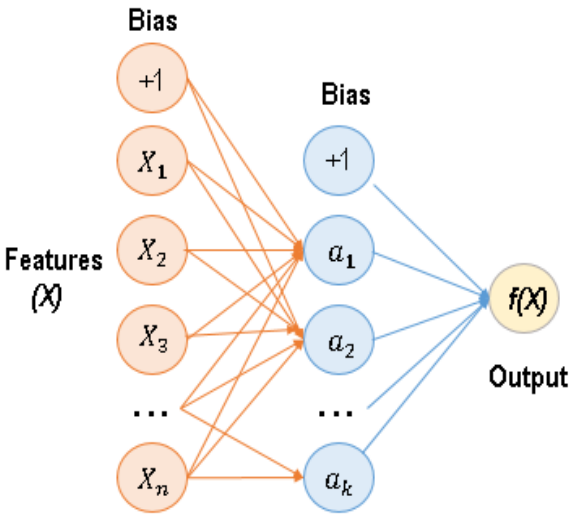


Figure 1. *MLP – Multi Layer Perceptron*

MLP is widely used for solving problems that require supervised learning as well as research into computational neuroscience and parallel distributed processing. Applications include speech recognition, image recognition and machine translation.

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and

outlier's detection. However, it is mostly used in classification problems. Originally, SVM is a binary classifier that works by identifying the optimal hyperplane and correctly divides the data points into two classes. There will be an infinite number of hyperplanes and SVM will select the hyperplane with maximum margin. The margin indicates the distance between the classifier and the training points (support vector). Figure 2 illustrates the basic idea of support vector machine [6].

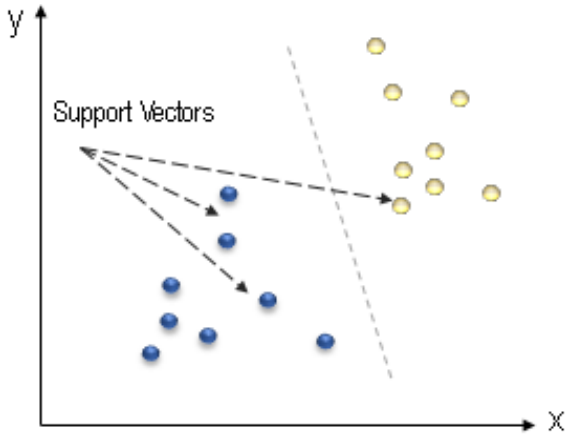


Figure 2. *SVM – Multi Layer Perceptron*

A number of techniques can be used to expand the classifier from binary to multiclass.

AUC ROC is one of the most important evaluation metrics for any classification model's performance.

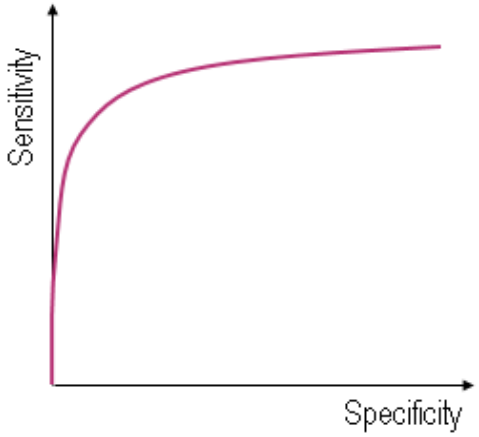


Figure 3. *AUC ROC Curve*

ROC (*Receiver Operating Characteristic*) Curve tells us about how good the model can distinguish between two things. Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two [10].

On the figure 3. is a sketch with a basic idea of the model predicting correct and incorrect values with respect to the threshold set. The more the AUC ROC value moves away from 0.5 to 1 or 0, the accuracy gets better and better. If it's 1, that's an ideal situation.

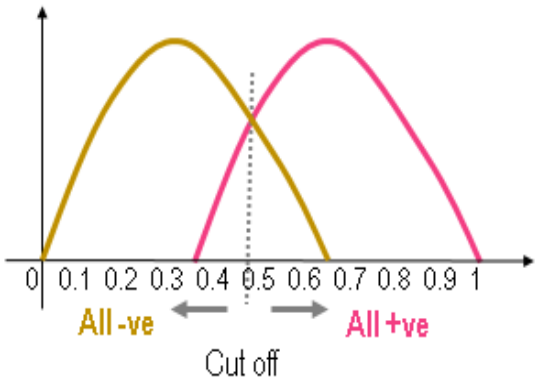


Figure 4: *Representation of AUC ROC idea*

4. Results and Discussion

During the neural network training 75 percent of the data has been taken and the remaining data (25 percent) for testing. The method used in neural network training is a trial and error method. With random selection of parameters required for network training, satisfactory results were obtained.

The best results are shown in Table 1 and Table 2. The tables below serve to show the best results obtained by randomly selected parameters of each of the classification methods used. In the MLP method, four parameters were selected to implement the classification: Hidden layer, Activation, Solver and Max_iter. In the SVM method, randomly selected parameters are: C = l, kernel = i, gamma = j, shrinking = k, probability = m, max_iter = p, decision_function_shape = n. Defaulted value parametars which are used are: coef0 = 0.0, degree = 3, cache_size = 200, class_weight = None, verbose = 0.

Table 1. *The best results obtained with MLP classifier*

Hidden layer	Activation	Solver	Max_iter	AUC
6	tanh	lbfgs	500	0,8696
6	identity	sgd	10	0,8478
6	relu	lbfgs	150	0,7283
6	tanh	sgd	10	0,7065

Despite the fact that the training lasts longer, in this case it is noticeable that the best result was obtained with a larger number of iterations.

Table 2. *The best result obtained with SVM classifier*

C	1	0.5	1	1
Kernel	linear	linear	linear	linear
Gamma	scale	scale	scale	scale
Shrinking	1	1	1	0
Probability	0	0	1	0
Dec.fun.shape	ovo	ovr	ovo	ovo
Max_iter	10	10	10	10
AUC ROC	0.9248			

In addition to the parameters that are constant, the values listed in the table have changed. The table shows some of the combinations that gave the highest accuracy. It can be noticed that the combinations are quite similar. For a dozen combinations, one accuracy was obtained, much higher accuracy than the MLP method, amounting to 93%. But all the other combinations gave very little accuracy, amounting to about 0.5 or exactly 0.5, which is extremely bad for the AUC metric.

4. Conclusion

The aim of the paper, which was to obtain high accuracy of fertility assessment in men, was successfully fulfilled, with an accuracy of 87% in the case of working with the MLP method and 93% accuracy with the SVM method. The obtained results satisfied the initial thesis. The SVM method made it extremely easy and faster to achieve an extremely good result.

Acknowledgments

We thank assistant dr. sc. Ivan Lorencin for his mentorship during the work on this project and article.

Reference

[1] Lorencin, Ivan, et al. "On Urinary Bladder Cancer Diagnosis: Utilization of Deep Convolutional Generative Adversarial Networks for Data Augmentation." *Biology* 10.3 (2021): 175. https://doi.org/10.3390/biology10030175

[2] Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. Artificial Intelligence in Medicine, 102, 1017467.

[3] David Gil, Jose Luis Girela, Joaquin De Juan, M. Jose Gomez-Torres, and Magnus Johnsson. Predicting seminal quality with artificial intelligence methods. Expert Systems with Applications, 39(16):12564 â€“ 12573, 2012

[4] Understanding Support Vector Machine(SVM) algorithm from examples (along with code) - September 13, 2017, accessed: 15.05.2022 https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[5] Xiaowei Li, Wen-Jing Lu, Ya'nan Li, Fujian Wu, Rui Bai, Shuhong Ma, Tao Dong, Hongjia Zhang, MLP-deficient human pluripotent stem cell derived cardiomyocytes develop hypertrophic cardiomyopathy and heart failure phenotypes due to abnormal calcium handling, 13 August 2019

[6] Vapnik V 1995 The Nature of Statistical Learning Theory New York Springer Verlag

[7] Amin Kahrizi, Hosein Hashemi: Neuron curve as a tool for performance evaluation of MLP and RBF architecture in first break picking of seismic data, September 2014

[8] Murtagh, F. (1991). Multilayer perceptrons for classification and regression. Neurocomputing, 2(5-6), 183-197. https://doi.org/10.1016/0925-2312(91)90023-5

[9] Dejan Radulović , Dino Negovanović: Gait speed prediction based on walking parameters using MLPRegressor

[10] Jocelyn D'Souza: Let's learn about AUC ROC Curve!, accessed:15.05.2022 https://medium.com/greyatom/lets-learn-about-auc-roc-curve-4a94b4d88152

ISBN: 978-953-8246-26-5

Urinary bladder cancer detection using YOLOv5 algorithm

Tajana Cvija [1*], Karlo Severinski [2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, tcvija@riteh.hr
[2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, kseverinski@riteh.hr

Abstract: Computer vision is a branch of artificial intelligence that is experiencing accelerated growth in a wide variety of application areas, including medicine, with object detection being one of its primary tasks. One of the algorithms that enables accurate real-time object detection is the YOLO (You Only Look Once) algorithm which reframes object detection as a regression problem. In this paper the fifth version of YOLO, YOLOv5, is used to train a model which task is detecting tumor cells on the urinary bladder. The model is successfully trained and tested on CT and MRI scans of the abdomen resulting in 92.5% accuracy. In the future it can be used with diagnostic devices to perform real-time tumor detection.

Keywords: artificial intelligence, computer vision, object detection, urinary bladder cancer, YOLOv5 algorithm

1. Introduction

Urinary bladder cancer is one of the most common genitourinary malignant diseases. [1] It is usually detected in its early stages and, hence, treatable and curable. The risk of reocurrence remains, so regular checkups and proper diagnosis are mandatory. The grand goal of future medicine is in using the potential of rapidly developing artificial intelligence (AI) to tailor medical diagnosis, decisions, health practices and therapies to the individual patient. [2, 3] A branch of AI that is experiencing accelerated growth in a wide variety of application areas, including medicine, is computer vision (CV). [4, 5] CV allows computers and systems to extract, process and interpret useful information from digital images, videos, and other visual input. [6, 7] One of its primary tasks is object detection which consists of detecting and classifying instances of semantic objects of a certain class in digital images and video. [8, 9, 10] To be able to do so it often uses deep learning methods and algorithms based on neural networks. [3, 9, 10] One of the algorithms that enables accurate real-time object detection is the YOLO (You Only Look Once) algorithm. It exists since 2015 and reframes object detection as a regression problem using a single neural network. [8, 11] The algorithm divides the input images into grids, which each cell has a different detection task. [12] It has been upgraded to five versions with the latest being YOLOv5, and performs with great accuracy and speed. [8] It is written using Python language and Pytorch framework. The YOLOv5 architecture contains four architecture with YOLOv5s as the basic one. [11]

In this paper, YOLOv5s is used to train a model tasked to detect tumor cells on the urinary bladder on frontal CT and MRI scans of the abdomen. The used dataset consisted of mentioned scans stored in .jpg format and corresponding labels created with Labellmg, a free software which enables annotation of image and video data and stores it in .txt format supported by YOLO algorithms. [13]

2. Methodology

As previously stated, the image dataset in .jpg format used in object detection training consisted of 360 frontal CT and MRI scans that showed tumor cells on the urinary bladder of various subjects. Figure 1. a) and Figure 1. b) represent frontal CT and MRI scans of the abdomen, respectively.

a)

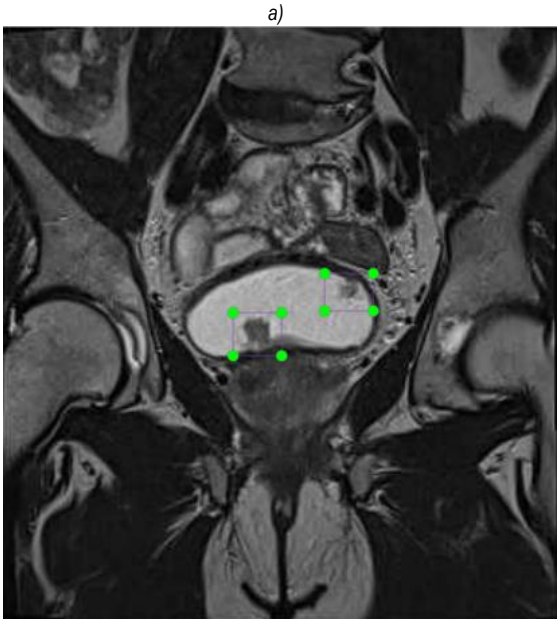
b)

Figure 1. *Frontal abdomen a) CT scan, b) MRI scan*

Before starting the training each image had to be annotated using already mentioned Labellmg software. The annotation was performed by using bounding boxes

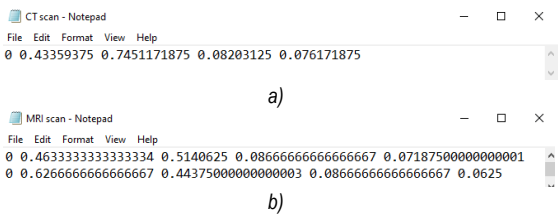
around areas containing tumor cells as shown in Figure 2 that results in corresponding .txt files given in Figure 3.

a)



b)

Figure 2. *Bounding boxes around tumor cell areas on the frontal abdomen a) CT scan, b) MRI scan*



a)

b)

Figure 3. *.txt files corresponding to bounding boxes around tumor cell areas on the frontal abdomen a) CT scan, b) MRI scan*

The images alongside with their labels were then used in training of the YOLOv5 model.

The open source code for YOLOv5 algorithm with a pretrained model for transfer learning was downloaded from GitHub platform and modified to fit one class object detection on grayscale input images. [14] It was implemented and executed in a Google Colaboratory notebook which offers free and powerful GPU with time-limited access.

3. Results and Discussion

The training was performed for 1800 epochs resulting in loss function given in Figure 4.

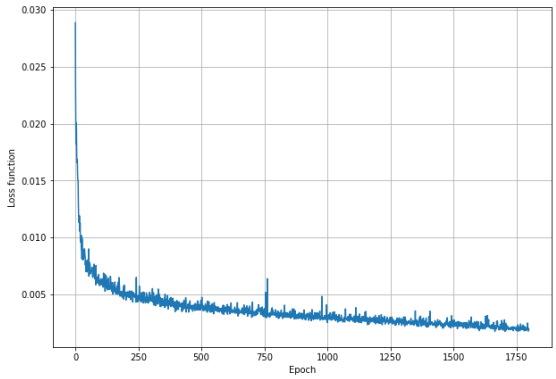


Figure 4. *Loss function*

The trained model's parameters (weights) were saved in a .pt format file and tested on another 40 frontal abdomen scans, which the model has not "seen" before. The accuracy of tumor cells detection was 92.5%. Examples of successful detection are given in Figure 5.

4. Conclusion

Concluding on the obtained results. The training of YOLOv5 model used for detection of the urinary bladder cancer has been successful and resulted in 92.5% detection accuracy which is considered as satisfactory in medical application of detection algorithms. Henceforward, this method can be used with horizontal and sagittal scans of the abdomen to obtain a further insight into patient's condition and even implemented with MRI and CT devices to enable real-time tumor cells detection.

Acknowledgments

First, we want to thank the assistant Ivan Lorencin, dr. sc., who helped us during the making of this paper and provided us with advice and data. We also thank everyone else who made this conference and our participation in it possible.

a)

b)

Figure 5. *Tumor cells detection on frontal abdomen a) CT scan, b) MRI scan*

References

[1] D. S. Kaufman, W. U. Shipley, A. S. Feldman, "Bladder cancer", *The Lancet*, 2009, https://doi.org/10.1016/S0140-6736(09)60491-8

[2] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, R. Socher, "Deep-learning-enabled medical computer vision", *npj Digital Medicine 4,* article 5, January 2021 https://doi.org/10.1038/s41746-020-00376-2

[3] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, H. Müller, "Causability and explainability of artificial intelligence in medicine", *WIREs Data Mining and Knowledge Discovery*, April 2019 https://doi.org/10.1002/widm.1312

[4] S. E. Umbaugh, "Computer vision in medicine: color metrics and image segmetation methods for skin cancer diagnosis", *University of Missoury - Rolla*, thesis, January 1990 https://dl.acm.org/doi/10.5555/101483

[5] A. N. Ramesh, C. Kambhampati, J. R. Monson, P. J. Drew, "Artificial intelligence in medicine", *Annals of the Royal College of Surgeons of England,* September 2004, https://doi.org/10.1308/147870804290

[6] I. H. Sarker, "AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems", *SN Computer Science*, 2022. https://doi.org/10.1007/s42979-022-01043-x

[7] G. Yang, W. Feng, J. Jin, Q. Lei, X. Li, G. Gui, W. Wang, "Face Mask Recognition System with YOLOV5 Based on Image Recognition", *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020 https://ieeexplore.ieee.org/document/9345042

[8] D. Thuan, "Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm", *Oulu University of Applied Sciences*, Thesis, 2021 https://www.theseus.fi/handle/10024/452552

[9] D. Andina, A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review", *Coputational Intelligence and Neuroscience of Missoury* , February 2018 https://doi.org/10.1155/2018/7068349

[10] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompy, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, C. I. Sanchez, "A survey on Deep Learning in Medical Image Analysis", *Med. Image Anal. 42*, 2017 https://doi.org/10.1016/j.media.2017.07.005

[11] A. Mohiyuddin, A. Basharat, U. Ghani, V. Peter, S. Abbas, O. Bin Naeem, M. Rizwan, "Breast Tumor Detection and Classification in Mammogram Images Using Modified YOLOv5 Network", *Computational and Mathematical Methods in Medicine*, 2022 https://doi.org/10.1155/2022/1359019

[12] W. Wu, H. Liu, L. Li, Y. Long, X. Wang, J. Li, Y. Chang, "Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image", *PLoS ONE 16(10): e0259283*, October 2021 https://doi.org/10.1371/journal.pone.0259283

[13] T. Lin, "Labelimg", accessed April 2022 https://github.com/tzutalin/labelImg

[14] G. Jocher: "YOLOv5", May 2020 https://github.com/ultralytics/yolov5

ISBN: 978-953-8246-26-5

Approximation of temperature in PMSM using a multilayer perceptron

Tomislav Štimac [1*], Nikola Anđelić [2]

[1 2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; tstimac95@gmail.com, nandelic@riteh.hr

Abstract: PMSM is a popular motor used in many industry applications. It is important to have correct temperature measures or estimations to fully exploit its overload potential and protect motor lifespan. In this paper MLP was used to predict stator yoke temperature from all other parameters of the dataset. It was shown that using the simple structure of MLP regression led to good estimates of the stator yoke temperature.
Keywords: Artificial intelligence, Machine learning, Multilayer Perceptron, PMSM overload, PMSM temperature estimation,.

1. Introduction

The permanent magnet synchronous motor (PMSM) is very popular and is used in many industrial applications due to its superior torque and power density. Preventing motor failure and ensuring long time usage along with high effectiveness is the key and that leads to temperature problems. In the stator there is a problem with deteriorating protective coating which can lead to partial discharge, reduction of lifespan and faults. Rotor problems are related to the decrease of PM flux which can lead to irreversible demagnetization of the magnets [1]. And finally, sensors that are hard to implement, are usually irreplaceably embedded and their functionality degrades. Motors are not cheap and easy to get so it is crucial to use as much overload potential as possible without damaging or shortening their lifespan. That is the reason for researching newer accurate real-time estimates of the temperature [2].
In this paper, the authors used a dataset related to [2] that contains many measurements related to specific parts of the PMSM that are shown in figure 1. Specific interest was in stator yoke whose temperature was estimated.

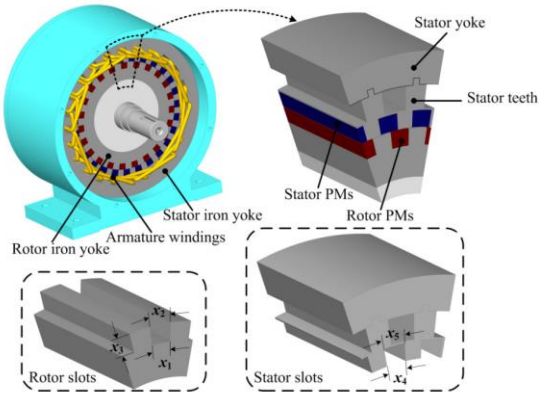


Figure 1. *Pm motor*

There are a lot of temperature estimation techniques but among the most successful are lumped-parameter thermal networks (LPTNs) which label equivalent circuit diagrams approximating inner heat transfer built on thermodynamic theory. In [2] is presented that using a neural network (NN) can show better results. Recurrent NNs and convolutional NNs (CNNs) are used because they achieve high estimation accuracy without having to retrieve domain expertise. That leads to the author's idea 'Is it possible to implement multilayer perceptron (MLP) for temperature estimation of PMSM?'. In this paper, it will be shown how MLP was configured and tested and how the temperature was estimated. Inspiration to use MLP was from successful implementations of neural networks reported in [4,5,6,7].

2. Materials and methods

The data set [2] comprises several sensor data collected from PSMS deployed on a test bench. Test bench measurements were collected by the LEA department at Paderborn University. The dataset consists of 185 h multivariate measurements sampled at 2Hz. The motor is torque-controlled while its speed is determined by a speed-controlled load motor. Temperatures were measured with embedded thermocouples, and the rotor temperature is represented by the average PM surface temperature across four sensors. The data set has 13 columns and each of them represents a variable, while each row (1330816) represents one snapshot of sensor data at a certain time step. Since sample data is 2 Hz there is one row per 0.5 s. Used data set is large and contains a lot of data. In table 1 all variables are listed by name along with basic static analysis. Voltages u_q and u_d represent the d and q component of voltage in the d-q system, similarly, i_q and i_d represent current. Motor speeds are shown in rotations per minute and torque in Nm. All other variables are the temperatures of different parts of the motor. Temperatures of all parameters vary from room temperature to over 100 degrees which is expected under load.

Table 1. *Dataset variable explanation and static analysis.*

NAME	UNIT	MIN	MAX	MEAN	STDEV
U_q	V	-25.3	133	54.3	44.2
coolant	°C	10.6	102	36.2	21.8
Stator winding	°C	18.6	141	66.3	28.7
U_d	V	-132	131	-25.1	63.1
Stator tooth	°C	18.1	112	56.9	23
Motor speed	RPM	-276	6000	2200	1860
I_d	A	-278	0.05	-68.7	64.9
I_q	A	-293	302	37.4	92.2
pm	°C	20.9	114	58.5	19
Stator yoke	°C	18.1	101	48.2	20
Ambient	°C	8.78	30.7	24.6	1.93
torque	Nm	-246	261	31.1	77.1

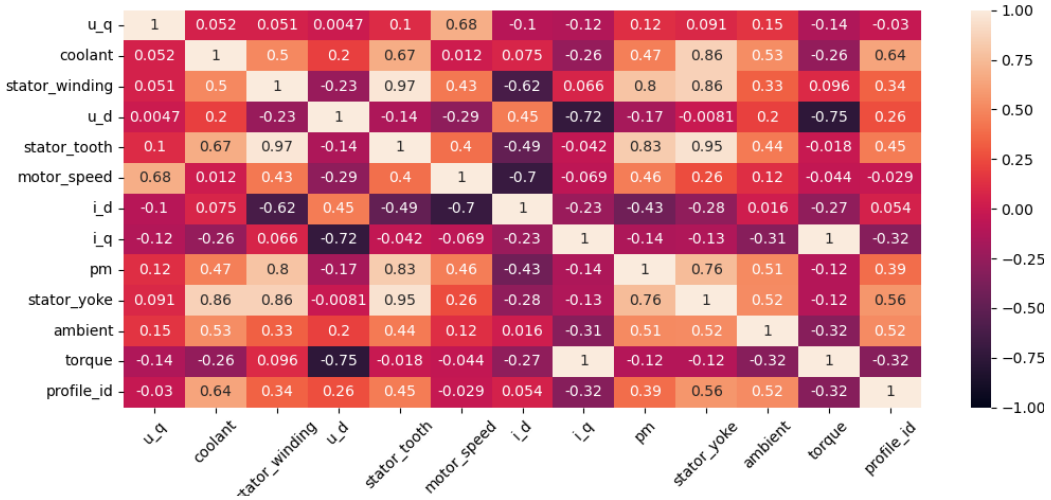


Figure 2. *Correlation Heatmap*

The next step in analyzing the data set is a correlation. Correlation is a statistical measure that expresses the strength of the relationship between two variables. Positive correlation occurs when two variables move in the same direction, and negative is when they move in the opposite direction. It is important to use correlation to determine the cause and effect relationship between variables in the data set. Using Seaborn [8] correlation heatmap from figure 2 was created. The correlation was calculated using Pearson and Spearman correlation coefficients that gave identical results. Heatmap helps visualize the strength of relationships between numerical variables that give us insight into how to move on with this paper. It was detected that stator_yoke has a good correlation with many variables so it was chosen as a variable that will be predicted with MLP regressor.

MLP is a feed-forward artificial neural network (ANN) that consists of three-layer types populated with neurons. The first layer is the input layer while the last layer is the output layer while all other layers in between are hidden layers. Feed-forward means that inputs are combined with the initial weights in a weighted sum and subjected to the activation function. Weights are coefficients that represent summated values from neurons in previous layers and are fed to the next layer. The main thing is an iterative adjustment of weights in the network which is called backpropagation. In each iteration, after the weighted sums are forwarded through all layers error is calculated. Then backpropagation happens and weights of the first hidden layer are updated according to the error. The described process happens until the error hasn't changed more than a threshold compared to the previous iteration [4,5,7]. To learn and optimize MLP scikit-learn was used [9,10]. In this paper, Adam was the main solver for weight optimization. It worked well on a relatively large dataset which is expected according to [10]. Other solvers were tried but Adam gave consistently good results. Choosing other parameters of MLP regressor was at random from a defined range. Table 2 has presented all parameters and their range that were used in training. Activation is a parameter that describes the activation function for the hidden layer. Identity or linear activation function, logistic is a sigmoid function, tanh the hyperbolic tan function, and ReLU the rectified linear unit function. The initial learning rate controls the step size in updating the weights. Alpha or L2 regularization parameter removes a small percentage of weights at each iteration. Beta1 and beta2 are parameters used only for Adam and they represent exponential decay rate for estimates of the first and second moment vector. According to [11] most important configuration parameters for Adam are alpha, beta1 and beta2. Their recommended values are in the default column of table 2.

Table 2. *Hyperparameters and their values.*

HYPERPARAMETER	LOWER BOUNDARY	UPPER BOUNDARY	DEFAULT
Number of Hidden layers	0	5	1
Number of neurons	30	150	100
Activation	Identity, Logistic, Tanh, ReLU		
Learning rate	Adaptive, Constant, Invscaling		Constant
Initial learning rate	0.0001	0.01	0.001
Alpha	0.00001	0.01	0.0001
Max iteration	10000	100000	200
Beta 1	0	1	0.9
Beta 2	0	1	0.999
No change	10	50	10

It is essential to evaluate the accuracy of the model. In this paper, three parameters were used. According to [12] formulas and explanations were given. R-Squared or coefficient of determination represents the coefficient of how well the values fit compared to the original values. The value ranges from 0 to 1 which is interpreted as percentages. A higher value represents a better model. It is defined by the formula (1) where $\hat{y}$ is predicted value of y, and $\bar{y}$ is mean value of y.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}, \qquad (1)$$

Mean absolute error (MAE) represents the difference between the original and predicted values averaged by the absolute difference over the dataset. It is defined by the formula (2) where variables have the exact meaning as in formula (1)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|, \qquad (2)$$

Root mean squared error (RMSE) is the error rate by the square root of MSE. Defined by the formula (3). For MAE and RMSE result can be any positive number, closer to zero represents a better result.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}, \qquad (3)$$

3. Results and discussion

Training was done using Spyder from Anaconda navigator on a desktop pc with 16 GB of RAM and AMD FX 8350 8 core processor. Training data used 30% of the dataset and MLP regressor had random selection of hyperparameters from table 2. Training did well but took fairly long and a lot of results got high evaluation parameters. Many models got R^2 of over 0.95, MAE and RMSE values lower than 1. Many of successful models had zero hidden layers and maybe the most successful model used recommended parameters for MLP regression using solver Adam. Result of that model is shown in figure 3. It is visible that predicted data from the test data set follows real data great, and there is little to none deviation to real data. Most successful models with their hyperparameters and evaluation parameters are in table 3. Model shown on figure 2 is Model 2 from table 3. In general best models had one hidden layer with 60-110 neurons. Model closer to the default MLP parameters for Adam performed better.

Table 3. *Models with hyperparameters*

HYPERPARAMETERS	MODEL 1	MODEL 2	MODEL 3
Hidden layers + neurons	()	(100,)	(144, 31)
Activation	relu	/	relu
Solver	adam	Adam	adam
Alpha	0.00608	0.0001	0.00957
Learning rate	constant	/	adaptive
Learn rate init	0.0010886085113552774	0.001	0.0046202847761008655
Max iter	34225	69043	21804
Shuffle	True	True	True
Beta 1	0.49079915156267995	0.9	0.7133305486357291
Beta2	0.564733181031462	0.99	0.4625096 20204692
No change	26	36	22
R^2	0.9970873778636159	0.9986171197189755	0.9973551046008468
MAE	0.8299110002763045	0.5147797298115492	0.7550857408894963
RMSE	1.078256671839985	0.7429718734802531	1.0275059175083538

3. Conclusion

Starting hypothesis 'Is it possible to implement multilayer perceptron (MLP) for temperature estimation of PMSM?' was proven. Using MLP for temperature estimation was successful and it is obvious that using simpler methods can be effective. In the future it is possible to do more testing with less input variables and to test estimation of other variables in the rotor which are much harder to measure than on the stator of the motor.

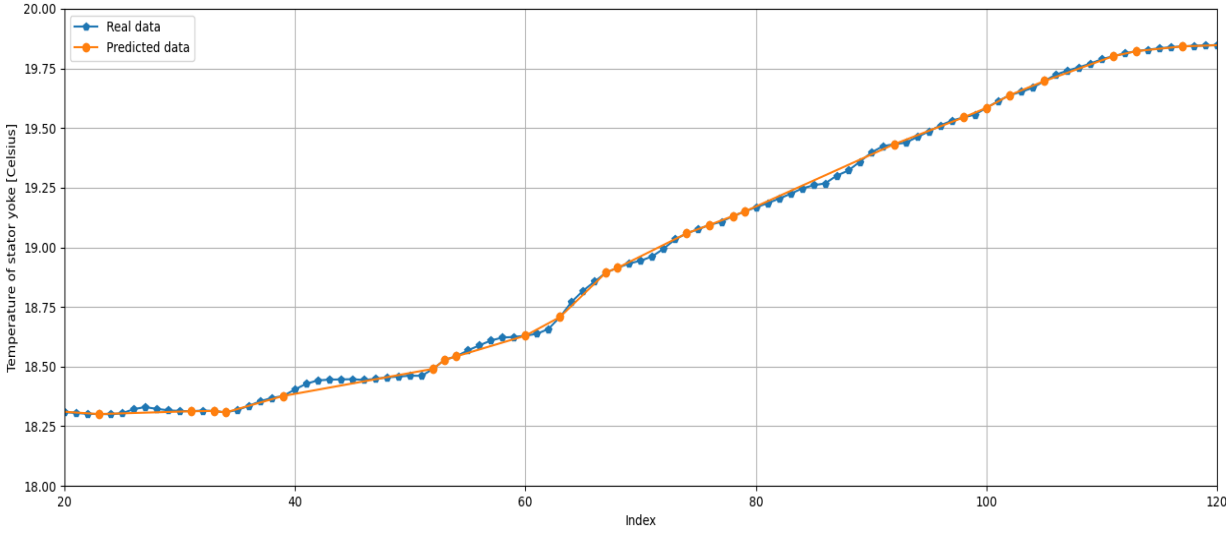


Figure 2. *Predicted data of test data set compared to real data*

Reference

[1] Gaona, D., Wallscheid, O., & Böcker, J. (2017, December). Fusion of a lumped-parameter thermal network and speed-dependent flux observer for PM temperature estimation in synchronous machines. In 2017 IEEE Southern Power Electronics Conference (SPEC) (pp. 1-6). IEEE.

[2] Kirchgässner, W., Wallscheid, O., & Böcker, J. (2020). Estimating electric motor temperatures with deep residual machine learning. IEEE Transactions on Power Electronics, 36(7), 7480-7488.

[3] Jian, L., Shi, Y., Wei, J., Zheng, Y., & Deng, Z. (2015). Design optimization and analysis of a dual-permanent-magnet-excited machine using response surface methodology. Energies, 8(9), 10127-10140.

[4] Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. Artificial Intelligence in Medicine, 102, 101746.

[5] Car, Z., Baressi Šegota, S., Anđelić, N., Lorencin, I., & Mrzljak, V. (2020). Modeling the spread of COVID-19 infection using a multilayer perceptron. Computational and mathematical methods in medicine, 2020.

[6] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Marine objects recognition using convolutional neural networks. NAŠE MORE: znanstveni časopis za more i pomorstvo, 66(3), 112-119.

[7] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Multilayer perceptron approach to condition-based maintenance of marine CODLAG propulsion system components. Pomorstvo, 33(2), 181-190.

[8] https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/; last visited 20.05.2022

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html?highlight=mlp%20regressor#sklearn.neural_network.MLPRegressor; last visited 20.05.2022

[11] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/; last visited 20.05.2022

[12] https://www.datatechnotes.com/2019/02/regression-model-accuracy-mae-mse-rmse.html; last visited 20.05.2022

ISBN: 978-953-8246-26-5

Artificial intelligence models for the prediction of NOx emissions in gas turbines

Matko GLUČINA[1*], Vedran MRZLJAK[2], Igor POLJAK[3] , Zlatan CAR[2]

[1] University of Rijeka, Braće Mažuranića Square 10, 51000, Rijeka, Croatia, email: matko.glucina@uniri.hr.
[2] University of Rijeka, Faculty of Engineering, Vukovarska ul. 58, 51000, Rijeka Croatia, email: vmrzljak@riteh.hr, car@riteh.hr
[3] Department of maritime sciences, University of Zadar, Ulica Mihovila Pavlinovića bb 23000 Zadar, Croatia email:ipoljak1@unizd.hr

Abstract: Gas emissions from power plants and gas turbines use fossil fuels as their energy source and have a detrimental effect on people and the environment, one of the harmful gases obtained by burning fossil fuels is Nitrogen Oxide which can have harmful effects on humans. This research boils down to the use of artificial intelligence algorithms from the scikit-learn library that can potentially predict future NOx values from a gas turbine. The research was conducted on a publicly available data set from a gas turbine in Turkey. Linear models Linear regression, Ridge, Stochastic gradient descent, ExtraTreesRegessor, GradientBoostingRegressor, and RandomForestRegressor were used, with the best result obtained using GradientBoostingRegressor, with R^2 and MSE being 0.9 and 12.60. The conclusion is that a simple approach could not provide sufficient quality results for the use of the model.

Keywords: Artificial intelligence, Gas Turbine Power Plants, Machine Learning, $NO_{X,}$ Regression Prediction

1. introduction

Electricity is one of the most important energies of today without which today's world would be unthinkable. Almost every company, household institution, etc. uses at least one electrical appliance, be it a computer or a light bulb, and of course, the question arises of electricity supply, ie what is a quality way of producing electricity that will meet the needs of today. There are various ways to produce electricity such as fossil fuel power plants, and renewable energy sources such as solar, wind, water, and biomass, but not all are effective and applicable 365 days a year. Due to the unprofitability of investing in renewable energy sources, today's consumer world is still based on conventional fossil power plants on coal and gas, which of course emit a variety of harmful gases like Nitrogen Oxide (NO_x) and Carbon monoxide (*CO*) that cause ozone holes and climate change [1]. Due to climate change, numerous studies are being done, so Mahmut Dirik in [2] uses a hybrid algorithm, in this case, an Adaptive neuro-fuzzy inference system combined with a genetic algorithm (*ANFIS - GA*) for the potential prediction of *CO* and NO_x emissions where with a coefficient of determination he obtained results which varies between 0.79933 and 0.90363 for the data separated into test and training data with different rates. The minimum values for the metrics Mean squared error (*MSE*), Root mean square error (*RMSE*), the standard deviation of the error (*STD*), and mean absolute percentage error (*MAPE*) were 24.8379, 4.9838, 4.9839, and 5.1660 and the minimum errors, ie values in the test set were 6.5961, 5.1571, 5.157, and 5.3695, respectively [2]. Research made by Alan Rezazadeh [2], presents the K-Nearest-Neighbor (*KNN*) algorithm for potentially predicting *NOx* emissions from natural gas electrical generation turbines. The research indicates the importance of electricity itself and maintaining electrical plants because the age of individual elements and structures greatly affects the degradation of equipment quality. Research also indicates the importance of a quality dataset in this case *KNN* performed with accurate prediction rates with a comparably bigger dataset of this kind of problem. Derrick Adams at. El. in their research [3] developed Deep Neural Network (*DNN*) and Least Squares Support Vector Machine (*LSSVM*) algorithms for predicting Sulphur oxide and Nitrogen oxide (*SOx-NOx*) emissions in coal-powered power plants. The results show that training without possible assumptions can improve the accuracy of the testing phase by a minimum of 10%, and the coefficient of efficiency can value up to 0.8925 and 0.994 for SO_x and NO_x respectively.

In this study, the aim is to try to find an approach simpler and more conventional than the computer-intensive algorithms for predicting NOx emissions. Several methods were used, of which the one that gave the best results in the elimination system was taken in the end. The hypotheses of this paper are as follows:

- is there a regression model that can accurately predict NO_X emissions based on a given dataset,
- whether the dataset depends on the outcome of the algorithm results and
- is the obtained model used to predict the values of NO_X and CO?

2. Methodology

This chapter covers the methodology used for this research. Machine learning (*ML*) is formulated as "minimizing the problem" of losing function versus a given set of examples (training set). This feature expresses the discrepancy between the values predicted by the model being trained and the expected values for each example case. The goal is to train a model with the ability to correctly predict a set of cases that are not present in the training set. The method by which it is possible to distinguish different categories of an algorithm is the type of result expected from a particular machine learning algorithm. Among the main categories it is possible to find:

- classification: inputs are divided into two or more classes and the learning system must produce a model that can assign one or more classes among those available to the input.

These types of tasks are usually solved using supervised learning techniques. An example of classification is the assignment of one or more labels to an image based on the object or objects contained therein,

- regression: conceptually like the classification with the difference that the output has a continuous and not discrete domain. It is usually managed under the supervision of learning. An example of regression is estimating the depth of a scene from a color image display.

The domain of the output in question is almost infinite and is not limited to a certain discrete set of possibilities. Since regression models are required for prediction, they can be found in the Python software library *scikit-learn*, which offers multiple selections of regression methods [4]. An example of a regression line can be found in Figure 1.

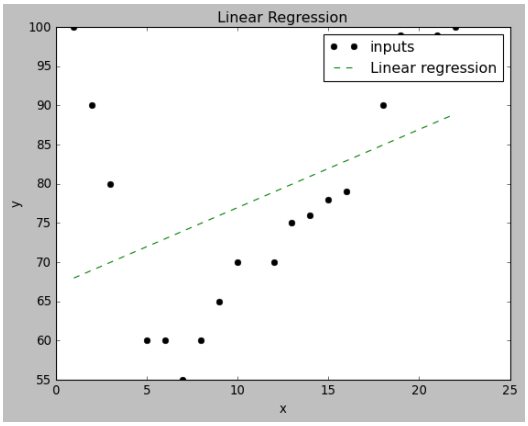


Figure 1. *Example of a linear regression line for predicting future values in the dataset*

Figure 1 shows several elements of linear regression. The linear regression consists of the direction E_y shown in Equation 1, this direction passes through the smallest distance between mutually defined points from the data set.

$$Ey = b_0 + b_1 x \quad (1)$$

If E_y is a linear function of x, then the coefficient next to x b_1 is the coefficient of the direction which is the graph of the function E_y. In other words, the direction can be written algebraically, in the form of equation (1). Where x is the argument of the function, b_1 is the direction coefficient, and b_0 is the segment on the y axis [4]. In addition to the linear regression, more complex regression methods combine higher-degree polynomials to better approximate the future values of previously given points. Furthermore, a polynomial regression from Figure 2 [5] is reported based on an unevenly distributed data set. Figure 2 represents the process of minimizing the error by adding the degree of the polynomial when approximating the points, the figure shows 3 polynomials from the first to the third degree. The first degree (green dashed line) poorly approximates values, the second degree (blue dotted line) affects some values, but not enough for quality approximation, and the third degree (red solid line) affects almost completely without error all values of the data set [5].

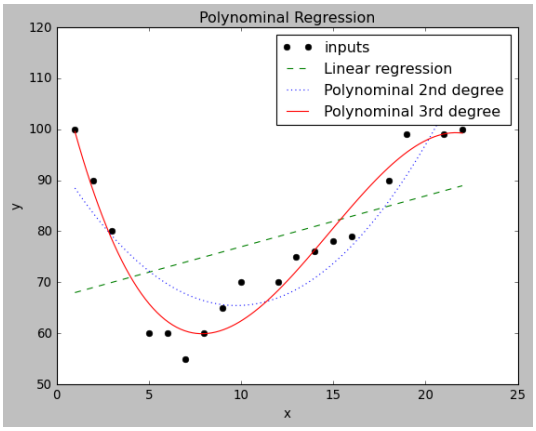


Figure 2. *Representation of polynomial regression*

The first degree (green dashed line) poorly approximates values, the second degree (blue dotted line) affects some values, but not enough for quality approximation, and the third degree (red solid line) affects almost completely without error all values of the data set [5]. The above methods are trivial, the conventional approach is shown in Figures 1 and 2, but more complex solutions are needed for some challenges. They are achieved using hybrid methods (merging multiple artificial intelligence (*AI*) algorithms into one). Thus, in the case of [6] Ivan Lorencin at. el. uses an approach to estimate power output in a power plant using a genetic algorithm (*GA*) and a Multilayer Perceptron *(MLP*). The research aimed to increase the performance of *MLP* compared to the heuristic algorithms used in previous research [6].

This research was approached in the way that, as mentioned above, an attempt was made to try from the simplest method to a hybrid; ensemble method to obtain a regression model for the approximation of *NOx* emissions. A publicly available dataset [7] from the UCI Machine learning repository entitled "Gas Turbine *CO* and *NOx* Emission Data Set Data Set" was used in this study. The data set contains 36733 instances of 11 sensor measures aggregated over one hour, from a gas turbine located in Turkey to study flue gas emissions, namely *CO* and *NOx*. Measured values from the dataset are:

- Ambient temperature (*AT*) [°C],
- Ambient pressure (*AP*) [mbar],
- Ambient humidity (*AH*) [%],
- Air filter difference pressure (*AFDP*) [mbar],
- Gas turbine exhaust pressure (*GTEP*) [mbar],
- Turbine inlet temperature (*TIT*) [°C],
- Turbine after temperature (*TAT*) [°C],
- Compressor discharge pressure (*CDP*) [mbar],
- Turbine energy yield (*TEY*) [MWH],
- Carbon monoxide (*CO*) [mg/m^3] and
- Nitrogen oxides (*NOx*) [mg/m^3].

A correlation heatmap in Figure 3. describes the relationships between values in each dataset. This means that it describes how one variable affects another, with a value of 1 indicating that the reference value correlates with the target value, and -1 denoting the opposite effect. For example, *TEY* highly correlates with *CDP* which is obvious because with greater air intake turbine can yield more energy. If the vital values of *CO* and NO_X are observed, no value directly correlates with other system variables, on the contrary, more variables do not correlate than correlate, which is a great challenge in predicting future NO_X values. The data set needed to be scaled because of extreme peaks in values for almost every variable. This makes it easier for the algorithm to load data to predict future values in its training set, values are limited to -1 to 1 instead of hitherto unbalanced ratios. After performing the previous actions, the dataset is ready for implementation into *AI* regression algorithms.

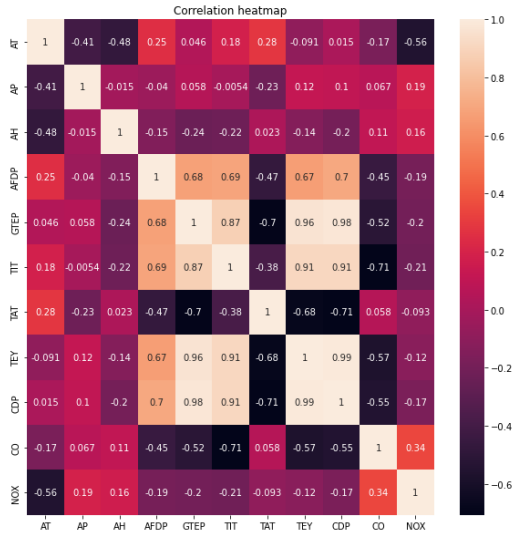


Figure 3. *Correlation heatmap of all variables from a given dataset*

Dataset is scaled and adjusted with *StandardScaler* from the *scikit-learn* package. The last step in regression research is by using *ensemble* methods of regression. *Ensemble* methods aim to combine the prediction of several basic estimators built with given learning algorithms, thus improving the generalization of robustness relative to a single estimator. There are two categories of ensemble methods: The method comes down to taking two estimators and looking at the mean value of each. On average, a combined appraiser is usually better than any single base appraiser because its variant is reduced. Some examples are: *RandomForestRegressor*, *ExtraTreeRegressor, and BaggingRegressor*. In contrast, in reinforcement methods, basic estimators are built sequentially, and attempts are made to reduce the bias of the combined estimator. The motivation is to combine a few weak models to produce a powerful ensemble. A summary of the methodology is as follows: first conventional and trivial methods like linear regression were used, then variations on a linear method like Ridge regression were used, and then *ensemble* regression methods were used.

3. Results and Discussion

After a detailed training of the algorithms for adjusting the parameters, the results are obtained. The assumption is that the simplest methods will get the worst results, which is true in this case. While hope, that more complicated *ensemble* methods will get better metrics than similar linear models still exist. Specific metrics are used to display the results, in this case, R^2 and *MSE* were used, which was defined in the introductory chapter of this article. At least 2 metrics need to be considered when calculating metrics. The reason for this is the "misconception" of the accuracy of the results, for example, R^2 can be shown to be 1 or close to 1 if the trend of the curve is similar, i.e., the same as the movement of points but for example, can be completely dislocated in relation to the actual result. Starting from linear regression models, the results are as follows:

- *linear regression,*
- *ridge linear model and*
- *stochastic gradient descent (SGD) regression*.

After that, *ensemble AI* algorithms were used:

- *ExtraTreesRegressor,*
- *GradientBoostingRegressor and*
- *RandomForestRegressor*.

Table 1. *shows the results for R^2 and MSE for all used algorithms*

Algorithm	R^2	MSE	CV
Linear Regression	0.55	58.80	NO
Ridge	0.54	58.79	NO
RidgeCV	-31.32	4318.13	YES
SGD	0.56	58.90	NO
ExtraTreesRegressor	0.62	12.74	YES
GradientBoostingRegressor	0.90	12.60	YES
RandomForestRegressor	0.88	15.44	YES

Table 2. Hyperparameters used for best-given results

Algorithm	Best hyperparameters
RidgeCV	‘alphas’ : [1.0], ‘fit_intercept’: [False], ‘scoring’:[‘negative mean squared error’]
ETR	'criterion’: ['squared_error'], 'splitter': [16], 'max_depth': [16], 'min_sample_split': [6], 'min_samples_leaf':[16], 'n_estimators': [100], 'criterion': ['mse']
GBR	n_estimators':[250], 'learning_rate': [0.05], 'max_depth':[6], 'min_samples_leaf’: [3,], 'max_features’: [0.5]}
RFR	'n_estimators': [1000], 'max_depth': [3], 'min_samples_split': [3]

It is possible to read some results from Table 1. Thus, starting from linear regression models, it is evident from the metrics that the results are not even close to

satisfactory. Thus, for example, basic linear regression gives R^2 of 0.55 and MSE of 58.80. This is a rather poor result since R^2, i.e., the predicted regression line, shows 50% different, i.e., wrong values than they should be. The mean square error is 58.80 in this case MSE for the line is calculated as the average of the sum of squares for all data points. Ridge has similar results as linear regression. R^2 is 0.53 while MSE is 58.79. The reason for this is some similarities in each of the algorithms. *Ridge* with *CrossValidation* is the worst of all regression models offered by *scikit-learn* for solving this kind of problem. The R^2 score is negative, which means that the complete approach to solving this challenge is just a waste of time. It does not predict any value from a given dataset and its square error value is very high. After that, the last in the trivial linear model's *SGD* algorithm has a value of R^2 and MSE in the amount of 0.56 and 58.90. The results are also unsatisfying and not usable in any way. By switching to ensemble methods, better results are achieved. For example, *ExtraTreeRegressor* (*ETR*) results in R^2 and MSE of 0.62 and 12.74. The results compared to the linear models are better but further research is needed to improve the results and get a model satisfactory for NO_X prediction. *GradientBoostingRegressor* (*GBR*) gets the best results of all the artificial intelligence algorithms used so far. R^2 is 0.9 while MSE is 12.60. However, it is evident from the square error that additional elaboration of the problem is needed, which would give even better results. The last in a series of algorithms used is *RandomForestRegressor* (*RFR*) which received R^2 and MSE in the amount of 0.88 and 15.44 which is high but not high enough for use as a real model prediction. All ensemble models were validated using 3 k-fold Cross Validation.

4. Conclusion

Emission emissions prediction is a major challenge that needs to be addressed as soon as possible. Emissions of harmful gases affect the harmony of life, and it is necessary to stabilize it as soon as possible. Based on research and the use of some *AI* models to attempt to predict NO_X emissions, inadequate results have been obtained for real-world use and real-world problems. The best result was given by *GBR* in the amount of R^2 and MSE 0.90 and 12.60, but the results should be even more accurate and of better quality, than obtained. As for further research, engineering scientists are invited to get better results than shown. The proposal is to redistribute the data set used in this paper. In this paper, a whole set of data of 5 years was considered and a model was trained. The results can potentially vary from those obtained. In addition, it is recommended to take other similar algorithms and get even better results, for example, take the *ExtreemeGradientBoost* (*XGBoosst*) algorithm and train it with more diverse hyper parameters. Or use a different approach like *Artificial Neural Network* [9] or different *ML* and Evolutionary Computing methods to solve this challenge [10].

Acknowledgments

I would like to take this opportunity to thank my close and beloved people who supported me in this entire research, without whom this research would not have been possible.

This research has been (partly) supported by the CEEPUS network Ciii-HR-0108, European Regional Development Fund under the grant K.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447

Reference

[1] Dirik, Mahmut. "Prediction of NOx emissions from gas turbines of a combined cycle power plant using an ANFIS model optimized by GA." Fuel 321 (2022): 124037. https://doi.org/10.1016/j.fuel.2022.124037

[2] Rezazadeh, Alan. "Environmental Pollution Prediction of NOx by Predictive Modelling and Process Analysis in Natural Gas Turbine Power Plants." Pollution 7.2 (2021): 481-494. https://dx.doi.org/10.22059/poll.2021.316327.977

[3] Adams, Derrick, et al. "Prediction of SOx–NOx emission from a coal-fired CFB power plant with machine learning: Plant data learned by deep neural network and least square support vector machine." Journal of Cleaner Production 270 (2020): 122310. https://doi.org/10.1016/j.jclepro.2020.122310

[4] Hao, Jiangang, and Tin Kam Ho. "Machine learning made easy: a review of scikit-learn package in a python programming language." Journal of Educational and Behavioral Statistics 44.3 (2019): 348-361. https://doi.org/10.3102/1076998619832248

[5] Ostertagová, Eva. "Modelling using polynomial regression." Procedia Engineering 48 (2012): 500-506 https://doi.org/10.1016/j.proeng.2012.09.545

[6] Lorencin, Ivan, et al. "Genetic algorithm approach to the design of multi-layer perceptron for combined cycle power plant electrical power output estimation." Energies 12.22 (2019): 4352.

[7] Gas Turbine CO and NOx Emission Data Set Data Set https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set

[8] Rathore, Santosh Singh, and Sandeep Kumar. "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems." *Knowledge-Based Systems* 119 (2017): 232-256. https://doi.org/10.1016/j.knosys.2016.12.017

[9] Strušnik, Dušan, and Jurij Avsec. "Artificial neural networking and fuzzy logic exergy controlling model of combined heat and power system in thermal power plant." *Energy* 80 (2015): 318-330. https://doi.org/10.1016/j.energy.2014.11.074

[10] Musulin, Jelena, et al. "Application of artificial intelligence-based regression methods in the problem of covid-19 spread prediction: A systematic review." *International Journal of Environmental Research and Public Health* 18.8 (2021): 4287.

ISBN: 978-953-8246-26-5

Estimation of excitation current of synchronous motor with multilayer perceptor and support vector machines

Jura Modrić[1], Ivan Zoričić[1]

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia)

Abstract: In this article, we have discussed working principles of synchronous motors, and estimation for its parameters, specifically excitation current as targeted value for estimation.

Keywords: MLP, SVR, Synchronous motors, Cross validation, python

1. Introduction

Synchronous motors are used as servo drives in applications such as computer peripherals equipment, robotics, and adjustable-speed drives in variety of applications such as load-proportional capacity-modulated heat pumps, large fans, and compressors.

In a synchronous motor rotor normally rotates at the same speed as the revolving field in the machine. The stator is like that of an induction machine consisting of a cylindrical iron frame with windings, usually three phases, located in slots around the inner periphery. Main difference is in the rotor, which normally contains an insulated winding connected through slip rings or other means to a source of direct current.

It is difficult to represent relationships between SM parameters mathematically, as they are complex and nonlinear. The task is to create the strong models to estimate the excitation current of SM.
Two models are trained and represented their result in later section, MLP and SVR. There is a brief explanation of the methods, comparison of result and accuracy of the estimator models in the next chapters. [1][2]

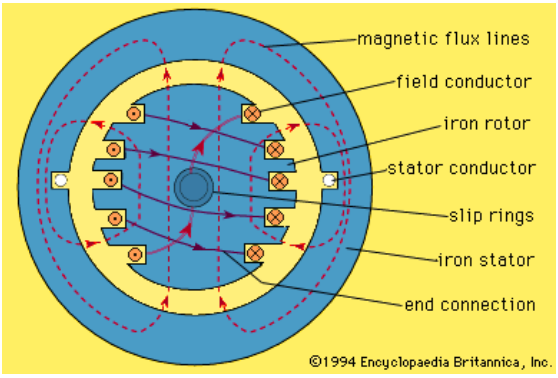


Figure 1. Synchronous motor scheme

2. Methodology

Dataset consists of load current, power factor, power factor error, excitation current and rate of change of excitation current. By analysing the columns of the dataset with Pearson correlation coefficient, one can see strong positive correlation with itself for each feature on diagonal of the matrix. If only the first row is observed, we can see correlation between the Iy (load current) and all other columns or features in our dataset. Power factor has a negative 0.04 corelation with Iy, which implies virtually no correlation between parameters. As power factor error and power factor have negative one correlation, so are the Iy and e correlations the same as Iy and Pf, but negative.
Iy and If(excitation current, our target) have mild correlation of approximately 0.4 which suggest rather weak correlation.

Table 1. *Dataframe preview*

#	Iy	PF	E	DIF	IF
1	3	0.68	0.32	0.372	1.552
2	3	0.7	0.3	0.36	1.54
…	…	…	…	…	…
556	6	0.99	0.01	0.16	1.34

As our target and derivative of target have strong correlation, or linear relationship (which is expected from theory), dIf or rate of change of If is omitted from feature selection. Thus, power factor, power factor error and load current were selected as input features for AI algorithms.

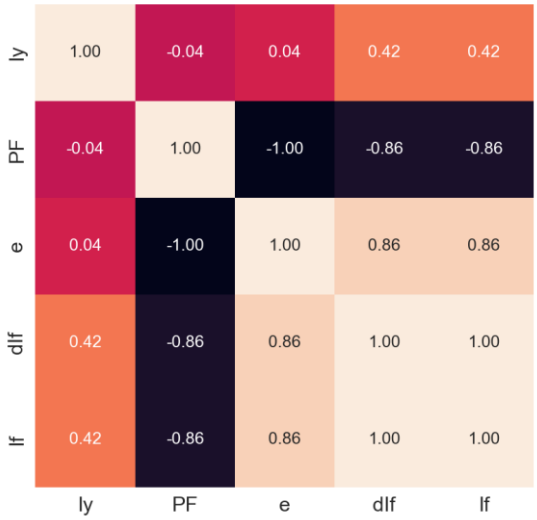


Figure 2. Feature correlation matrix

Next step is feature scaling, so the algorithm would preform faster and more accurately. Feature scaling, when not always necessary, is indeed desired. Depending on the distribution of values for each feature, we choose normalisation or standardisation. For gaussian distribution of values, we use standardisation, which is a method that converts dataset values in a way that their mean value is centred around zero and values. As the plots suggest, we can se the "shift" of graphs to the left, centred around zero. The plot in figure x shows all the features standardized and put in the same array (array of arrays), where inner array contains 3 features for each datapoint.

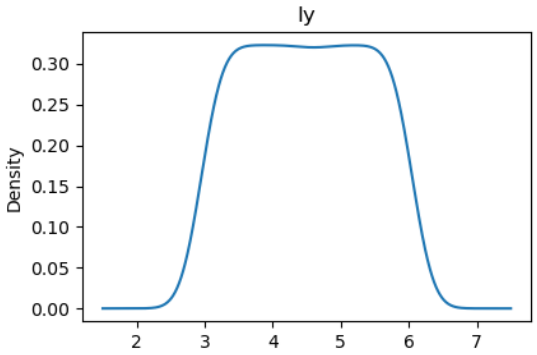


Figure 3. Load current distribution

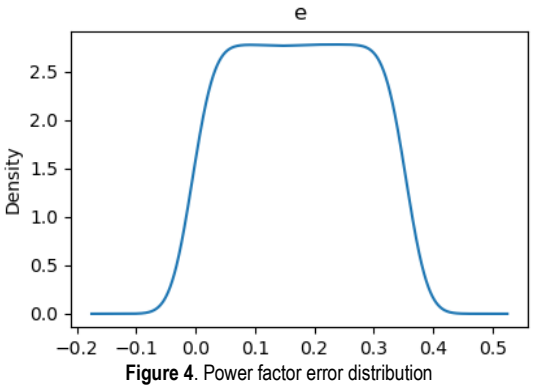


Figure 4. Power factor error distribution

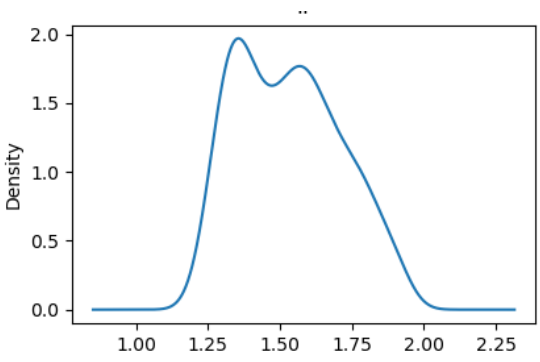


Figure 5. Field current distribution

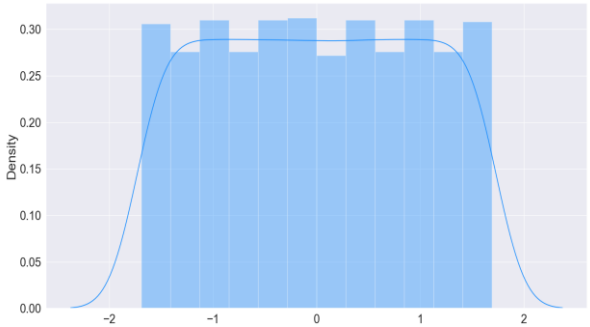


Figure 6. Scaled features distribution

Support vectors regression method is a regression method. The idea is to fit a regression line (in linear regression), on top of the training data points with minimal error. We define a "tube" around the regression line, with parallel lines at some offset. This offset, called epsilon, is the margin at which there is no penalty for data point that is not on the regression line, but is inside the epsilon margin.

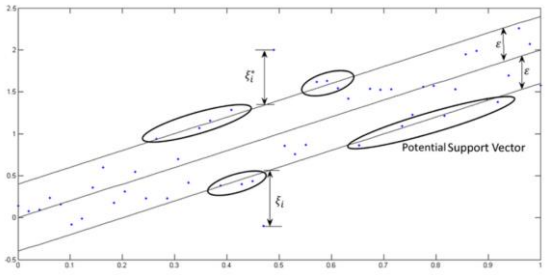


Figure 7. *Support vector principle*

MLP on the other hand works differently. It is an artificial neural network that consists of input, hidden and output layer of neurons. There can have more than one hidden layer. Every neuron has an activation function and is connected to every neuron in the next layer.

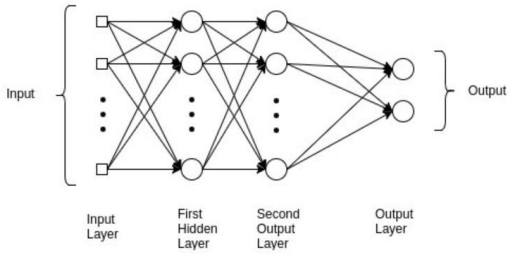


Figure 8. MLP architecture

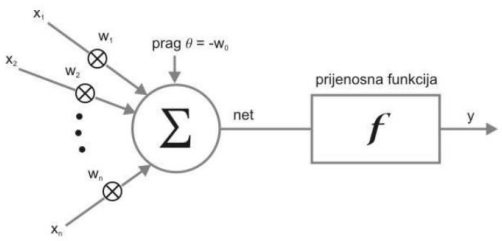


Figure 9. Neuron

$$y = f(\sum_{k=0}^{n} w_k x_k) \ \#(1)$$

In fig.9 f(net) is the activation function used in block "f". The function can be linear, sigmoid, tanh, ReLU most commonly, but there are others. [3]

Activation functions determine the output of neuron based on the summed inputs it gets from the previous layer. By modifying the weights of connection between neurons, we get different output every iteration, and thus converge to minimal error or model.

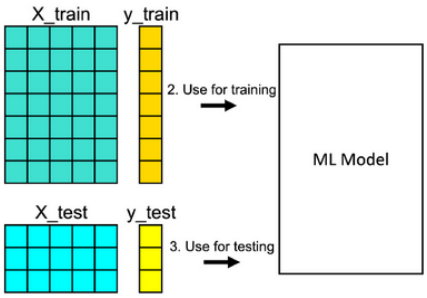


Figure 10. *Train test split*

Next step is to configure the regressors. It has several options (hyperparameters), such as: number of hidden layers, type of activation function, solver, learning rate, early stopping and similar for MLP, and kernel type, degree of polynomial (if using poly as kernel), gamma (if using rbr, poly, sigmoid as k.), parameter "C" or regularization parameter, epsilon, or "margin" for vectors around centre value. [4]
The hyperparameter tuning was performed by grid search algorithm. For each regressor, we have entered parameters intuitively, respecting the general practice and most common use cases for each hyperparameter. After few iterations parameter grid was updated with new values, close to those chosen in the previous iteration, and deleting the ones which haven't been chosen in last few iterations.
That successively lead to smaller and smaller error for the current models.

Grid search with 5 folds and negative mean squared error as scoring parameter was used for tuning the hyperparameters. Learning curve for both models is shown in figures below, together with scaling and accuracy of models.

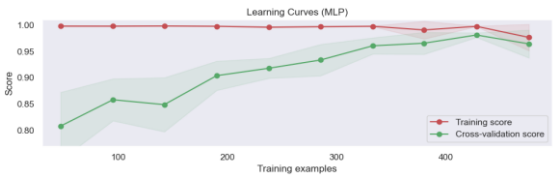


Figure 11. *Learning curve MLP*

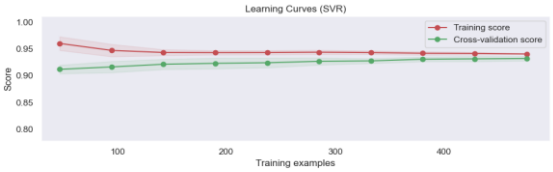


Figure 12. *Learning curve SVR*

MLP demonstrates better accuracy of the model. It has lower errors than SVR, as shown in the graphs.

3. Results and Discussion

After fitting the results in our regression models, the model was trained. The first graph shows the plot of real current, measured experimentally, in blue line. Other lines, as described in the legend, show predictions by each prediction model. At the first glance, we can see that MLP is the closest to real values of excitation current. Polynomial SVR shows values that are little over the targeted value, while RBF SVR shows under-values.

For evaluation metrics, generally when talking about regression problems mean error, mean absolute error, and R^2 are most common indicators for accuracy. Those are the metrics by which we'll be comparing SVR to MLP.

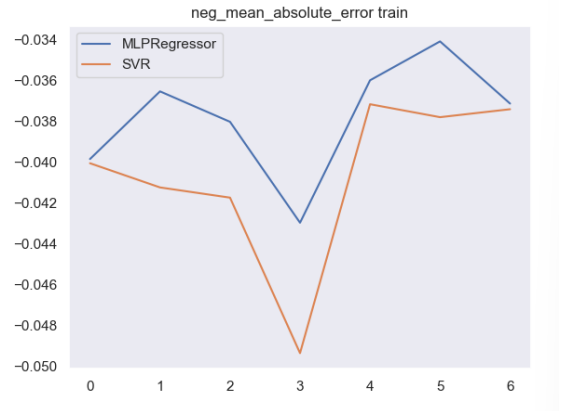


Figure 13. Feature correlation matrix

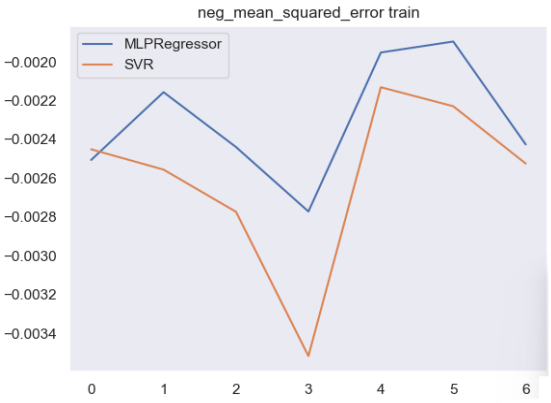


Figure 14. Feature correlation matrix

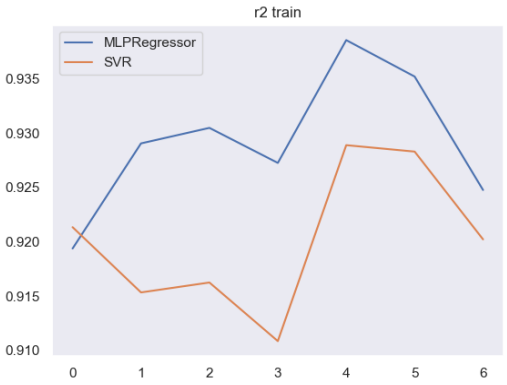


Figure 15. Feature correlation matrix

All used metrics report MPL having smaller error. It outperforms SVR by just small amount. Learning curves for each model suggest conversion at around 0.93 for both models, with MLP being better by just 0.02 points. We could say that SVR is a little more robust and faster because of simplicity.

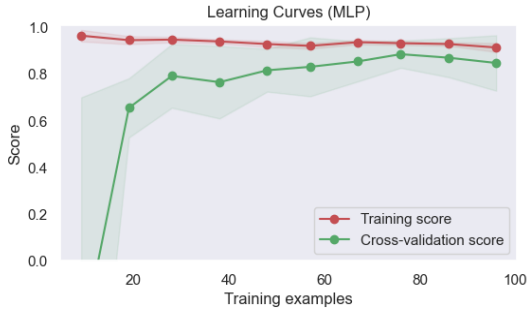


Figure 16. Feature correlation matrix

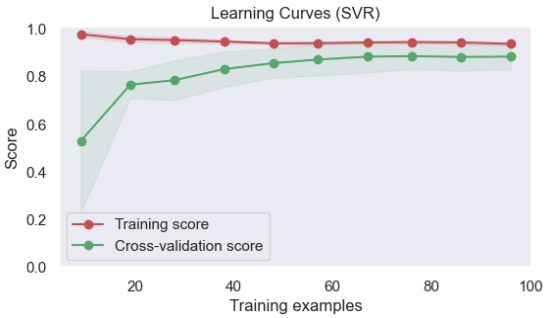


Figure 17. Feature correlation matrix

Fit times from 0.4 to 0.6 for MLP in comparison to 0.0025 to 0.0050 seconds for SVR is quite dramatical difference.

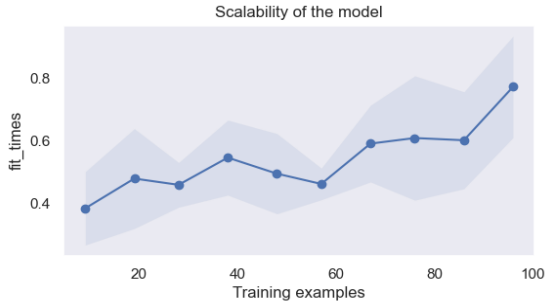


Figure 18. Feature correlation matrix

Figure 19. Feature correlation matrix

Below charts plot fitting times versus accuracy score. It is clear that MLP achieves satisfactory accuracy at 0.5 seconds, when SVR does the same at 0.005 seconds, or 100 times faster.

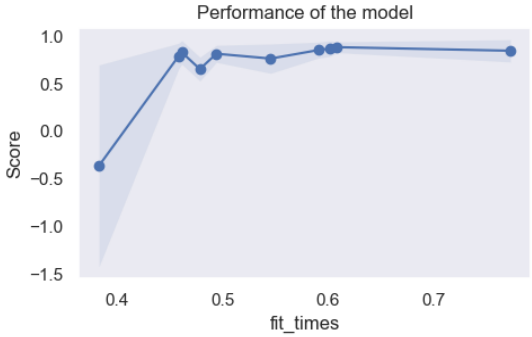


Figure 20. Feature correlation matrix

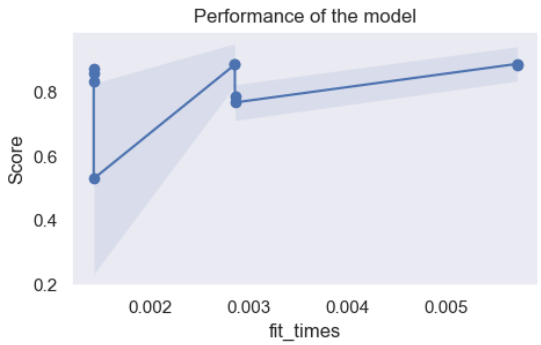


Figure 21. Feature correlation matrix

4. Conclusion

This was very rewarding research, in terms of knowledge gained. We discussed some properties of AI methods, working of SM. We tested the models, compared them and plotted the results. We tried out different tools for estimating AI performance.

Reference

[1] Kahraman, H. T. (2014). Metaheuristic linear modeling technique for estimating the excitation current of a synchronous motor. Turkish Journal of Electrical Engineering & Computer Sciences, 22(6), 1637-1652.

[2] Kahraman, H. T., Bayindir, R, & Sagiroglu, S. (2012). A new approach to predict the excitation current and parameter weightings of synchronous machines based on genetic algorithm-based k-NN estimator. Energy Conversion and Management, 64, 129-138.

[3] Scikit-learn. Scikit-learn python library documentation

[4] Car, Zlatan Lectures in Evolution robotics

ISBN: 978-953-8246-26-5

Fuel Consumption Using Multilayer Perceptron

Karla Šešelja [1], Marko Škara [2]

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, kseselja@riteh.hr
[2] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, mskara@riteh.hr

Abstract: This paper presents an analysis of passenger car fuel consumption using a multilayer perceptron. The main goal is to approximate the consumption that will later help in the design of internal combustion engines, which will reduce CO_2 emissions. The dataset used in this paper is publicly available and contains 393 data points. The quality of solution is estimated by the coefficient of determination (R^2) and the mean absolute error (MAE). The best result of $R^2 = 0.88$ and MAE = 2 MPG was achieved with four hidden layers containing 128 neurons each, tanh activation function, adaptive learning rate of 0.0001, L2 regularization of 0.1 and LBFGS solver.

Keywords: Artificial intelligence, Artificial neural networks, Fuel consumption, Machine learning, Multilayer perceptron

1. Introduction

Fuel consumption of cars with internal combustion engines (ICE) is one of biggest problem in the automotive industry. In addition to the economic factor of reducing fuel use, which is non-renewable energy source. There is also environmental problem, global warming. Car manufacturers need to reduce CO_2 emissions so that engines have same or even higher efficiency then today's engines.

Numerous works have been done to estimate fuel consumption using an artificial neural network (ANN). It was used in modelling aircrafts [1], ships [2], trucks [3] and even tractors [4]. The advantage of artificial neural networks is high accuracy with minimal error. The essence of this paper is to determine fuel consumption using dataset that contains passenger car parameters and multilayer perceptron.

2. Methodology

Data set used in this paper is publicly available and contains the information for creating the predictive model [5]. 393 data variables can be divided in two types. The inputs will be independent and on the other hand outputs will be dependent variables. Table 1. shows names, types and uses of variables in the data set.

Table 1. *Characteristics of data set*

NUMBER	NAME	TYPE	USE
1	MPG	Continuous	Output
2	Cylinders	Multi-valued discrete	Input
3	Displacement	Continuous	Input
4	Horsepower	Continuous	Input
5	Weight	Continuous	Input
6	Acceleration	Continuous	Input
7	Model year	Multi-valued discrete	Input
8	Origin	Multi-valued discrete	Input

Multilayer perceptron (MLP) is the machine learning (ML) algorithm used in this paper. It is a type of artificial neural network (ANN) that is trained using forward propagation processes. It is usually used for this type of problem when it is necessary to get some kind of output. The model consists of neurons placed in layers, using fully connected architecture in which every neuron in one layer is connected to all neurons in subsequent layer, using weighted connections [6]. Each MLP consists of at least three layer of neurons. First ones are input variables and the last one is output variable. In between comes a hidden layer of neurons that consists of at least one layer. Example of this network is shown in Figure 1.

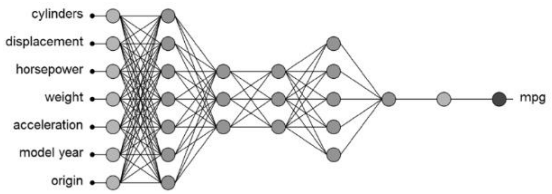


Figure 1. Example of an *MLP ANN used for this problem [7]*

Hyperparameters are values which describe the general architecture of the neural network used to train this model. They need to be varied to achieve the best regression models for each case. Hyperparameters adjusted during the training for this paper were [8]:

- Hidden layers - the number of hidden layers and number of neurons in each layer.
- Activation function – type of function used within all of the model's neurons
- Solver – algorithm used for weigh adjustment during training
- Initial learning rate (α) – determines initial speed of ANN
- Learning rate type - adjustment of learning rates through the training. Constant keeps the learning rate the same as starting throughout training, while adaptive and inverse scaling adjust it depending on the weight gradient value

- L2 – regularization parameter

Hyperparameters used in this paper and their possible values are represented in Table 2.

Table 2. *Hyperparameters and their values*

HYPERPARAMETERS	VALUES	NUMBER
Hidden layers	(8), (16), (32), (64), (128), (8,8), (16, 16), (32, 32), (64, 64), (128, 128), (8, 8, 8), (16, 16, 16), (32, 32, 32), (64, 64, 64), (128, 128, 128), (8, 8, 8, 8), (16, 16, 16, 16), (32, 32, 32, 32), (64, 64, 64, 64), (128, 128, 128, 128), (128, 128, 128, 128, 128)	21
Activation function	Linear, ReLu, Tanh, Logistic	4
Solver	Adam, LBFGS	2
Initial learning rate	0.1, 0.01, 0.001, 0.0001	4
Learning rate type	Constant, Adaptive, Invscaling	3
L2	0.5, 0.1, 0.01, 0.0001	4

After training the network, models were evaluated using two metrics: coefficient of determination (R^2) and mean absolute error (MAE).

Coefficient of determination is value that defines how well is the variance of the real results represented by predicted results [9]. Its value is in range between 0 and 1 and it is calculated using:

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}\left(y_i - \frac{1}{n}\sum_{i=0}^{n} y_i\right)^2}, \qquad (1)$$

where y_i contains real data and $\widehat{y_i}$ contains predicted data. Each prediction corresponds with real data for each i. Mean average error calculates the difference for each element of the two solution vectors (real and predicted data), measures their absolute error per each element. It can be calculated by:

$$MAE = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i|, \qquad (2)$$

3. Results and Discussion

Out of 8064 solutions, 4636 solutions are in range [0;1] for coefficient of determination. Out of the 4636 solutions 407 solution are higher than 0.85. Best found MLP solution is with $R^2 = 0.88 \pm 0.08$ and $\text{MAE} = 2.07 \pm 0.88$. Hyperparameter settings of best solution is given in Table 3.

Table 3. *Hyperparameters for best found MLP*

HYPERPARAMETERS	VALUE
Hidden layers	(128, 128, 128, 128)
Activation function	Tanh
Solver	LBFGS
Initial learning rate	0.0001
Learning rate type	Adaptive
L2	0.1

Graphs down below were made in Excel and they show hyperparameter settings for solutions better than 0.85. As said before there were 407 solutions higher than 0.85. Figure 2. shows that ReLu activation function was used most, by 359 of the networks. Tanh was used by 29, logistic by 19 and identity by none.

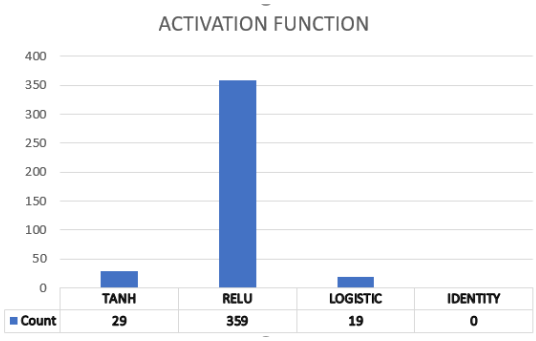


Figure 2. Distribution of activation function for values higher than 0.85

Figure 3. shows distribution of learning types used by networks. 144 networks used constant learning type which is the most. Invscaling type was used by 135 and adaptive type was used by 128 networks.

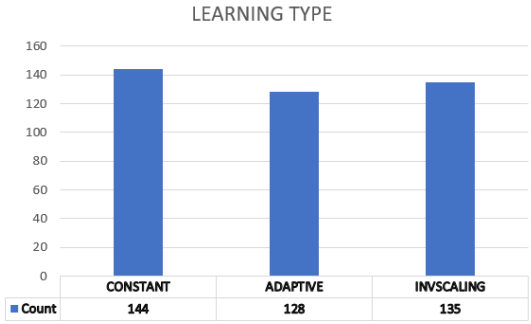


Figure 3. Distribution of learning type for values higher than 0.85

Figure 4. shows distribution of initial learning rates used by networks. Most used solution was 0.1 with 135 networks using it. Other values are relatively equally divided.

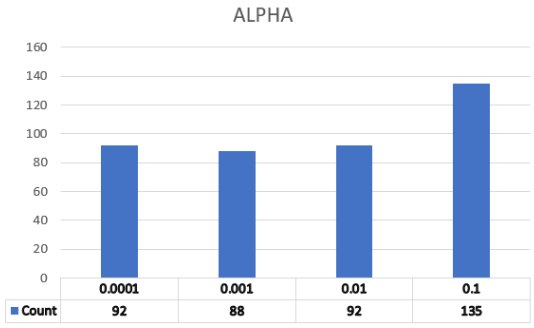


Figure 4. Distribution of initial learning rates for values higher than 0.85

Figure 5. shows distribution of regularization parameters used by networks. All solutions are used around 100 networks. Least used one was 0.5 by 95 and most used was 0.0001 by 105 networks.

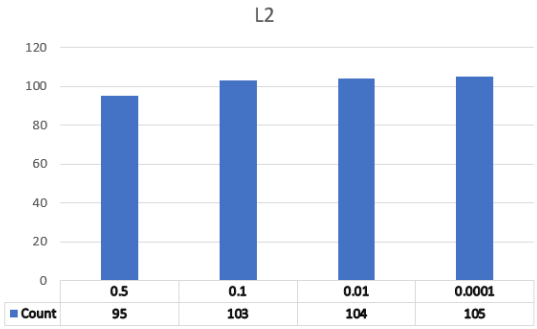


Figure 5. Distribution of regularization parameters for values higher than 0.85

Figure 6. shows distribution of solvers used by networks. Only 1 network used Adam solver while the remaining 406 networks used the LBFGS solver.

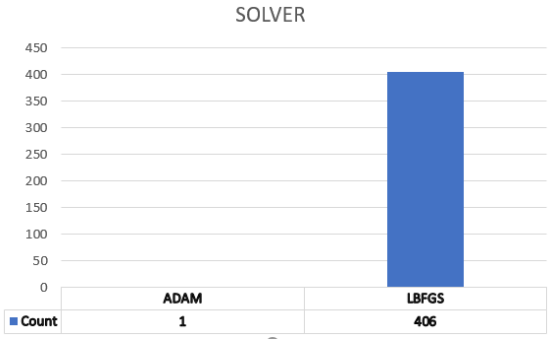


Figure 6. Distribution of used solvers for values higher than 0.85

Figure 7. shows distribution of hidden layers. Least used were single hidden layers, but moving towards 3 or 4 hidden layers can be seen increase of usage by networks. Most used hidden layers are ones with 128 neurons. 4 hidden layers with 128 neurons was used most and single hidden layer with 8 and 16 neurons were not used at all.

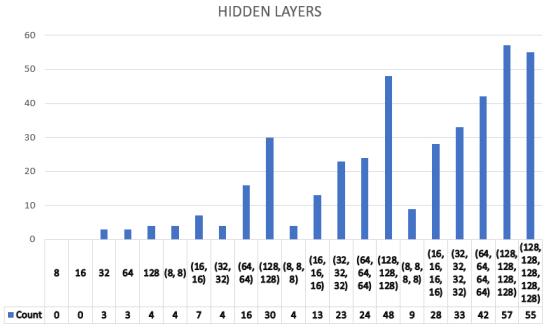


Figure7. Distribution of hidden layers for values higher than 0.85

4. Conclusion

This paper presents the MLP ANN for fuel consumption prediction. The results are satisfactory but there is always room for improvement. Best result of coefficient of determination is $R^2 = 0.88 \pm 0.08$ and for mean average error is $\mathrm{MAE} = 2.07 \pm 0.88$. This result was achieved using following hyperparameter settings: 4 hidden layers with 128 neurons each, Tanh activation function, adaptive learning rate of 0.0001, L2 of 0.1 and LBFGS solver.

Future work should include more hidden neural networks with more neurons, database expansion or the use of different methods to achieve even better results.

Acknowledgments

We thank Sandi Baressi Šegota mag. ing. comp. for his mentorship and help during the work on this article.

Reference

[1] Khan, Waqar Ahmed, et al. "A novel self-organizing constructive neural network for estimating aircraft trip fuel consumption." *Transportation Research Part E: Logistics and Transportation Review* 132 (2019): 72-96. https://www.sciencedirect.com/science/article/pii/S1366554519303138

[2] Jeon, Miyeon, et al. "Prediction of ship fuel consumption by using an artificial neural network." *Journal of Mechanical Science and Technology* 32.12 (2018): 5785-5796. https://link.springer.com/article/10.1007/s12206-018-1126-4

[3] Siami-Irdemoosa, Elnaz, and Saeid R. Dindarloo. "Prediction of fuel consumption of mining dump trucks: A neural networks approach." *Applied Energy* 151 (2015): 77-84. https://www.sciencedirect.com/science/article/pii/S0306261915005279

[4] Borges, Pedro HM, et al. "Estimation of fuel consumption in agricultural mechanized operations using artificial neural networks." *Engenharia Agrícola* 37 (2017): 136-147. https://www.scielo.br/j/eagri/a/5z3p8WQc9tJfDJy4qH4wsqy/?format=html&lang=en

[5] Luis Candanedo, luismiguel.candanedoibarra '@' umons.ac.be, University of Mons (UMONS). http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

[6] Baressi Šegota, Sandi, et al. "Frigate speed estimation using CODLAG propulsion system parameters and multilayer perceptron." *NAŠE MORE: znanstveni časopis za more i pomorstvo* 67.2 (2020): 117-125. https://hrcak.srce.hr/file/345685

[7] Jamala, Mohammed N., and Samy S. Abu-Naser. "Predicting MPG for automobile using artificial neural network analysis." *International Journal of Academic Information Systems Research (IJAISR)* 2.10 (2018): 5-21. https://www.researchgate.net/profile/Samy-Abu-Naser/publication/328430242_Predicting_MPG_for_Automobile_Using_Artificial_Neural_Network_Analysis/links/5bcdf021299bf1a43d99370e/Predicting-MPG-for-Automobile-Using-Artificial-Neural-Network-Analysis

[8] Baressi Šegota, Sandi, et al. "Dynamics Modeling of Industrial Robotic Manipulators: A Machine Learning Approach Based on Synthetic Data." *Mathematics* 10.7 (2022): 1174. https://www.mdpi.com/2227-7390/10/7/1174/htm

[9] Baressi Šegota, Sandi, et al. "Improvement of marine steam turbine conventional exergy analysis by neural network application." *Journal of Marine Science and Engineering* 8.11 (2020): 884. https://www.mdpi.com/2077-1312/8/11/884/pdf

ISBN: 978-953-8246-26-5

Condition-Based Maintenance of Naval Propulsion Systems: A Brief Review

Igor POLJAK [1], Darin MAJNARIĆ[2], Vedran MRZLJAK[3], Ivan LORENCIN[4]

[1] Maritime Department, University of Zadar, Mihovila Pavlinovića 1, 23000 Zadar, Croatia; ipoljak1@unizd.hr
[2] Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Ul. Ivana Lučića 5, 10000 Zagreb, Croatia; darin.majnaric@gmail.com
[3] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; vmrzljak@riteh.hr
[4] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; ilorencin@riteh.hr

Abstract: A standard maintenance methodology used in the majority of modern technical systems is corrective maintenance. This methodology is based on the removal of the malfunction by repairing or replacing broken components. Such an approach can be expensive due to the prolongate diagnostic and repair procedures. An alternative approach is to use predictive methodology: condition-based maintenance (CBM). In this paper, a brief review of relevant papers in the field of CBM of marine propulsion systems is presented. At the end, all presented methodologies are compared.

Keywords: CODLAG propulsion system, Condition-based maintenance, Diesel engine, Machine learning, Statistical modeling

1. Introduction

During the exploitation of any technical system, maintenance is one of the key actions required to expand the exploitation possibility of the system and to prolongate its lifetime. A standard maintenance methodology used in a majority of modern technical systems is corrective maintenance. This methodology is based on the removal of the malfunction by repairing or replacing broken components. Such an approach can be expensive due to the prolongate diagnostic and repair procedures. Furthermore, this methodology can cause more severe damage to the system components. An alternative approach is to use predictive methodology based on the condition of specific outputs of the system [1]. Such an approach is also called condition-based maintenance. One of the key applications of condition-based maintenance (CBM) is in the marine environment [2]. Due to the high power and maintenance costs, CBM has a high implementation possibility in marine propulsion systems [3].

In this paper, a brief review of relevant papers in the field of CBM of marine propulsion systems is presented. CBM methodologies are presented on both diesel and CODLAG propulsion systems. In the end, all presented methodologies are compared. According to the obtained knowledge, a conclusion regarding the applicability of CBM in naval propulsion systems will be given.

2. Diesel Engines

Alongside standard gas and steam propulsion plants, the utilization share of diesel engines is rapidly increasing [4]. In several published papers, CBM of a marine diesel engine was performed. Wang et. al [5] have used stochastic filtering to estimate the state of the fleet that consists of several marine diesel engines. As input parameters, metal concentrations in oil samples were observed. The authors have concluded that such an approach has a high implementation possibility. Cvrk and Ilijević [6] have proposed a methodology based on the measurements of key parameters of a marine diesel engine by using the electronic device "PREMET". Furthermore, the visual control of the cylinders was used. The authors have concluded that by using such an approach, an extension of the period between two overhauls is enabled. Furthermore, by using such an approach maintenance costs are reduced. Awang et. al [7] have proposed CBM of a 2-stroke marine diesel engine based on ultrasound monitoring. The method, combined with wave spectrum analysis is used for the early identification of excessive friction. The authors have concluded that by combining CBM and wave spectrum analysis, identification of engine performances can be performed without the need for major and complex diagnostic procedures. Coraddu et. al [8] have proposed the utilization of the Digital Twin of the ship for the estimation of the speed loss due to marine fouling. The proposed ship uses two four-stroke diesel engines with a rated power of 3840 *kW*. The achieved results are pointing towards the conclusion that such an approach can be used to achieve a higher quality of speed loss prediction.

3. CODLAG propulsion system

In recent years, more complex propulsion systems based on the combination of different components are emerging. One of these propulsion systems is a combined diesel-electric and gas (CODLAG) propulsion system. Such a system uses a combination of electrical power produced by diesel generators and power produced by gas turbine [9]. A schematic representation of such a propulsion system is presented in Figure 1.

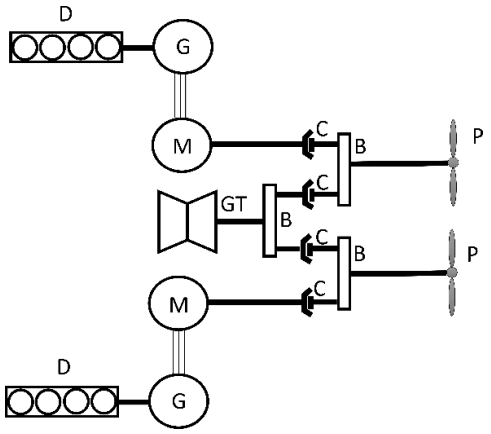


Figure 1: Scheme of a CODLAG propulsion system (GT – gas turbine; M – electrical motor; G – electrical generator; D – diesel engine; B – gearbox; C – clutch; P – Frigate propeller)

In the last couple of years, there have been studies that proposed CBM of a CODLAG propulsion system. Cipollini et. al [10] have developed a CBM system based on several state-of-the-art machine learning (ML) models were used. As input to custom regression models, sensor data is collected from a vessel that is characterized by a CODLAG propulsion system. Such an approach has been shown to be reliable and effective. The same group of authors [11] has proposed an approach based on the utilization of a minimal feedback method to develop a user-friendly data collection procedure. Coraddu et. al [12] proposed an ML framework for CBM of CODLAG marine propulsion system. In this paper, an approach based on the estimation of the state of a CODLAG propulsion system installed as propulsion of a frigate is proposed. Such an approach has shown the possibility of ML utilization for CODLAG CBM. This claim can be supported with error rates lower than 0.1. The used data set is collected from simulation and published openly. By using the data set, multiple ML regression methods were examined. The authors in [13] have used genetic programming algorithm to estimate the state of a frigate and to perform CBM. By using this approach, symbolic expressions are developed. The developed symbolic expressions have high regression performances. Such a conclusion can be derived from high R2 scores, higher than 0.99. The authors in [14] have proposed a similar approach for CBM, which is also based on the utilization of genetic programming and symbolic expressions. In this research, CODLAG parameters and decay actors were used to estimate the ship speed change. In this case, the lowest error rate of 0.6729 is achieved. Furthermore, in this case, R2 values up to 0.99 were achieved.

4. Methods overview

From the presented overview, it can be seen that the stochastic modeling is showing high performances in CBM of diesel engine for propulsion. Furthermore, it can be seen that in the case of CODLAG propulsion system, ML methods have an important place in CBM methodology. Furthermore, it can be seen that genetic programming and resulting symbolic expressions can also be successfully used for CBM of a CODLAG propulsion system. An overview is presented in Table 1.

Table 1: Comparison of achieved results

Propulsion system	Paper	Method
Diesel engine	[5]	Stochastic filtering
	[6]	"PREMET"
	[7]	Ultrasound Signal
	[8]	Twin Ship model
CODLAG propulsion system	[10]	Machine learning regression
	[11]	Machine learning regression
	[12]	Machine learning regression
	[13]	Genetic programming
	[14]	Genetic programming

5. Conclusions

In this paper, a brief overview of the CBM methods in naval propulsion is presented. It can be concluded that CBM methodology can be applied to diesel-based propulsion, as well as to more complex propulsion systems such as CODLAG. To increase propulsion robustness and reduce maintenance costs, computer-aided CMB should be included in the standard maintenance procedure of naval propulsion systems.

Acknowledgement

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, and University of Rijeka scientic grant uniri-tehnic-18-275-1447.

Reference

[1] Ahmad, Rosmaini, and Shahrul Kamaruddin. "An overview of time-based and condition-based maintenance in industrial application." Computers & industrial engineering 63.1 (2012): 135-149

[2] Mérigaud, Alexis, and John V. Ringwood. "Condition-based maintenance methods for marine renewable energy." Renewable and Sustainable Energy Reviews 66 (2016): 53-78.

[3] Lorencin, Ivan, et al. "Multilayer perceptron approach to condition-based maintenance of marine CODLAG propulsion system components." Pomorstvo 33.2 (2019): 181-190.

[4] Fernández, I. A., Gómez, M. R., Gómez, J. R., Insua, A. A. B.:Review of propulsion systems on LNG carriers, Renewable and Sustainable Energy Reviews 67, p. 1395–1411, 2017.(https://doi.org/10.1016/j.rser.2016.09.095)

[5] Wang, Wenbin, B. Hussin, and Tim Jefferis. "A case study of condition based maintenance modelling based upon the oil analysis data of marine diesel engines using stochastic filtering." International Journal of Production Economics 136.1 (2012): 84-92.

[6] Cvrk, Sead, and Dragan Ilijević. "Application Of Diagnostics as a Basis of Condition Based Maintenance of the Marine Propulsion Diesel Engine." Brodogradnja: Teorija i praksa brodogradnje i pomorske tehnike 71.3 (2020): 119-134.

[7] Awang, Mohd Naim, et al. "Main Propulsion Marine Diesel Engine Condition Based Maintenance Monitoring Using

Ultrasound Signal." Advancement in Emerging Technologies and Engineering Applications. Springer, Singapore, 2020. 175-187.
[8] Coraddu, Andrea, et al. "Data-driven ship digital twin for estimating the speed loss caused by the marine fouling." Ocean Engineering 186 (2019): 106063.
[9] Baressi Šegota, Sandi, et al. "Frigate speed estimation using CODLAG propulsion system parameters and multilayer perceptron." NAŠE MORE: znanstveni časopis za more i pomorstvo 67.2 (2020): 117-125.
[10] Cipollini, Francesca, et al. "Condition-based maintenance of naval propulsion systems with supervised data analysis." *Ocean Engineering* 149 (2018): 268-278.
[11] Cipollini, Francesca, et al. "Condition-based maintenance of naval propulsion systems: Data analysis with minimal feedback." *Reliability Engineering & System Safety* 177 (2018): 12-23.
[12] Coraddu, Andrea, et al. "Machine learning approaches for improving condition-based maintenance of naval propulsion plants." Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment 230.1 (2016): 136-153.
[13] Anđelić, Nikola, et al. "Estimation of gas turbine shaft torque and fuel flow of a CODLAG propulsion system using genetic programming algorithm." Pomorstvo 34.2 (2020): 323-337.
[14] Anđelić, Nikola, et al. "Use of Genetic Programming for the Estimation of CODLAG Propulsion System Parameters." Journal of Marine Science and Engineering 9.6 (2021): 612.

ISBN: 978-953-8246-26-5

Prediction Of Appliance Energy Consumption

Tomo Galović [1], Mario Bugarin [2]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, tgalovic@riteh.hr
[2] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, mbugarin@riteh.hr

Abstract: This paper describes the problem of predicting device consumption. The problem was to determine the most optimal neural network hyperparameters for predicting device consumption based on known data set. The introduction outlines the reasons for the need to predict device consumption and two basic ways to predict. The method of calculating hyperparameters and determining the best values of the results obtained by the neural network are described in the second chapter. Cross validation was also used in the calculations. The third chapter contains the results of the comparison of matching between real and calculated values with and without cross-validation, and presents the hyperparameters of the above.

Keywords: Neural network, hyperparameters, consumption, cross-validation, prediction R squared, mean absolute error.

1. Introduction

Predicting the consumption of appliances is important for two fundamental reasons. The first reason is the planning of the budget for the payment of electricity bills, while the second is related to the production of electricity. Electricity consumption is related to the type and number of electrical devices, and the way they are used by consumers [1]. Two different models can be used to predict energy consumption. The physical model uses thermodynamic rules for modeling and analysis, and is based on building and environmental parameters. The model for predicting electrical appliances consumption using artificial intelligence relies on sets of collected data in a certain period. Using the above the neural network can be created [2]. In this article, the prediction of the power consumption of device using artificial intelligence will be performed on collected data set. Data set will be used to train and test the neural network. Results with and without cross-validation will also be presented.

2. Methodology

Using the data set [3] neural network training with and without cross-validation was performed. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations [4]. Neural network training and testing was performed in the Python programming language. Neural network architecture used was *Multi Layer Perceptron* (*MLP*). MLP is a type of multilayer neural network consisting of two or more layers of interconnected neurons [5]. The input data set consists of the values of atmospheric conditions inside and outside the building, while the output data set consists of the device consumption values. The phyton function MLPRegressor, which defines the MLP neural network, was used in the program code. Function attributes are hyperparameters such as activation function, number of iterations, or number of neurons. Hyperparameters are shown on Figure 1.

```
params_dict = [{'hidden_layer_sizes': [(20, 20, 20, 20), (20, 20), (40, 40, 40, 40), (40, 40)],
               'activation': ['relu','identity','logistic','tanh'],
               'solver': ['adam', 'lbfgs'],
               'learning_rate':['constant','adaptive','invscaling'],
               'learning_rate_init': [0.1,0.01,0.5, 0.0001],
               'alpha': [0.001,0.01,0.0001, 0.00001],
               'max_iter': [10000]}]
```

Figure 1. *Hyperparameters values for combinations*

The hyperparameters of the neural network were changed during training and different combinations were tested. After training and testing the network, the results were obtained. Different hyperparameters and methods with or without cross-validation lead to better or worse matches. Based on the levels of coincidence of the output values of the neural network with the actual ones, the values of the hyperparameters were selected. The accuracy of neural network prediction was measured using a goodness of fit factors as coefficient of determination (R^2) and median absolute error regression loss (MAE). The coefficient of determination is calculated according to the expression:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}, \qquad (1)$$

while MAE counts as:

$$MAE = \frac{\sum_{i=1}^{n}|Y_i - \hat{Y}_i|}{n}, \qquad (2)$$

where symbols are:

n – number of measurements
Y_i – real measured value
$\hat{Y}_i$ – predicted value
$\bar{Y}$ – mean value [1]

The results are more accurate the closer R^2 is to 1 and MAE is closer to 0.

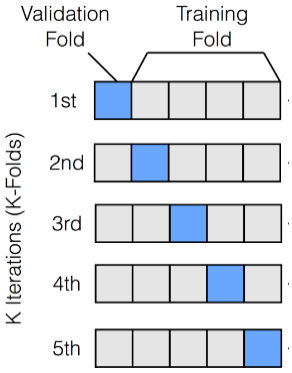


Figure 2. *Cross validation method [5]*

Cross validation method is shown by Figure 2. From the display we see that network training is performed in multiple iterations. In each subsequent iteration, the training data sets and the test data set change. Thus, the network is trained with a larger amount of data. In this way, an effort is made to achieve greater invariance of the neural network.

3. Results and Discussion

Based on the comparison of R^2 obtained for different values of hyperparameters, hyperparameters were selected for whose values R^2 takes the value closest to 1. This represents a very good match between the output values of the neural network and the actual values of the observed data.
The best results for the case without cross-validation were obtained for the following hyperparameter values:

Table 1. *Hyperparameters without cross-validation*

Hyperparametar	Value
Activation function	Rectified Linear Unit
Alpha	0.0001
Hiden layer sizes	40,40,40,40
Learning rate	Adaptive
Learning rate init	0.01
Maximum of iterations	10000
Solver	lbfgs

Table 1 shows that the neural network consists of four hidden layers. Each layer has 40 neurons for which the goodness of fit measure values are shown in Table 2.

Table 2. *Goodness of fit measure factors without cross-validation*

R^2	MAE
0.9763625	-0.1843809

The best results using the cross-validation were obtained for the following hyperparameter values:

Table 3. *Hyperparameters using cross validation*

Hyperparametar	Value
Activation function	Sigmoid
Alpha	0.0001
Hiden layer sizes	40,40
Learning rate	Constant
Learning rate init	0.0001
Maximum of iterations	10000
Solver	adam

Table 3 shows that the network consists of 2 hidden layers. Each of the layers has 40 neurons for which the best values of goodness of fit measure are shown in Table 4.

Table 4. *Goodness of fit measure factors using cross validation*

R^2	MAE
0.9704483	-0.2079654

4. Conclusion

Based on the obtained results, it can be concluded that the predictions of neural networks whose hyperparameters are presented in Chapter 3 obtained with and without cross-validation give very accurate results. Consumption of the device is predicted with great accuracy. A comparison of the results obtained with and without cross-validation shows that for the case without cross-validation R^2 takes a value closer to 1, which leads to the conclusion that the model without cross-fitting is more accurate. The presumed reason for this is the higher number of repetitions during training with cross-validation and potential overfitting. Increasing the input variables could increase the accuracy of the device consumption forecast.

Reference

[1] Luis M. Candanedo, Véronique Feldheim, Dominique Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy and Buildings, Volume 140, 2017, Pages 81-97, ISSN 0378-7788, https://doi.org/10.1016/j.enbuild.2017.01.083.

[2] Kadir Amasyali, Nora M. El-Gohary, A review of data-driven building energy consumption prediction studies, Renewable and Sustainable Energy Reviews, Volume 81, Part 1, 2018, Pages 1192-1205, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2017.04.095.

[3] Luis Candanedo, luismiguel.candanedoibarra '@' umons.ac.be, University of Mons (UMONS). http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

[4] Šegota, Sandi Baressi, et al. "Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator." International Journal of Advanced Robotic Systems 18.4 (2021): 1729881420925283. https://doi.org/10.1177/1729881420925283

[5] Car, Zlatan, et al. "Modeling the spread of COVID-19 infection using a multilayer perceptron." *Computational and mathematical methods in medicine* 2020 (2020).. https://doi.org/10.1155/2020/5714714

[6] From internet, PyTorch K-Fold Cross-Validation using Dataloader and Sklearn, August 29, 2021, https://androidkt.com/pytorch-k-fold-cross-validation-using-dataloader-and-sklearn/

ISBN: 978-953-8246-26-5

Determination of steel hardness after hardening

Dani Šestan[1], Dean Stjepanović[2]

[1] Faculty of Engineering – University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, dsestan@riteh.hr.
[2] Faculty of Engineering – University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, dstjepanovic@riteh.hr.

Abstract: This paper presents a Multilayer Perceptron (MLP) artificial neural network method for the purpose of determining steel hardness after hardening. The hardness of steel is important for many processes which describe the properties of it such are resistant to scratching, penetration, plastic deformation and dents. The goal is to obtain the highest possible hardening efficiency of steel for minimizing injection heat into the process, in order to make steel as resistant as possible to the said loads. The best result achieved by MLP with two hidden layers containing 10 neurons each, logistic activation function, adaptive learning rate and solver LBFGS was an accuracy of 97.499999% with standard deviation of 0.027386.

Keywords: Artificial intelligence, artificial neural networks, multilayer Perceptron, steel hardness

1. Introduction

Hardening is a heat treatment where steels are heated to their appropriate hardening temperature (usually between 750°C and 1200°C, depending on the composition of the material), held at temperature and then rapidly cooled ("quenched") in water or oil. Therefore, heated steel is soaked at a lower temperature ("tempering") which develops the final mechanical properties of steel. Hardening purpose is to develop the optimum combination of strength, toughness and hardness in a steel and to offer a route of savings in material and weight to the component designer [1]. The aim of this article is to examine the hardening efficiency of steel based on the input data such as family of steel, shape, product type etc; using multilayer perceptron artificial neural network. The purpose is to improve the properties of the steel itself and to minimize injection heat to the process.

2. Methodology

Used data set is publicly available [4] and contains the information for calculating the efficiency of hardening. Data set contains 798 variables divided in 38 attributes.
Multilayer perceptron is a feed forward neural network which consists of input layer, one or more hidden layers and output layer shown on Figure 1 [2, 3]. For the input signal to be processed enter the input layer, task like prediction or classification is executed by the output layer. The real computational mechanism of MLP is a hidden layer. An arbitrarily selected number of hidden layers is located between input and output layer. The data flow in MLP is forward, from input to output layer.

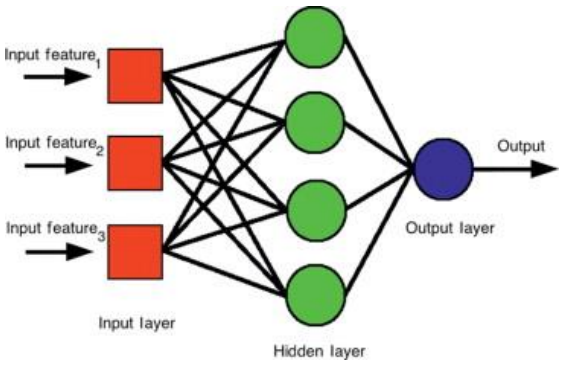
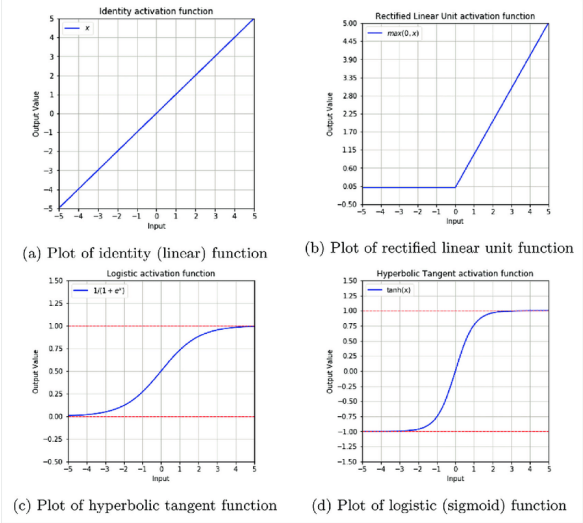


Figure 1. *Multilayer perceptron neural network*

Various activation functions were used during neural network learning. Some of them are ReLU, logistic (sigmoid), tanh and identity (linear) activation function. Table 1 presents the definition of mentioned activation functions and in Figure 2 are plotted.

Table 1. *Activation function*

Name	f(x)
Sigmoid	$\frac{1}{1+e^{-x}}$
Tanh	$\frac{2}{1+e^{-2x}} - 1$
ReLU	$x, x \geq 0$ $0, x < 0$
Identity	$x, x \in R$



(a) Plot of identity (linear) function (b) Plot of rectified linear unit function

(c) Plot of hyperbolic tangent function (d) Plot of logistic (sigmoid) function

Figure 2. *Plotted activation functions ((a) – identity, (b) – ReLU, (c) – tanh, (d) – logistic)*

3. Results and Discussion

By changing the hyperparameters, different results were obtained. Some of them were acceptable and some were not. The following few tables show the best result for certain hyperparameters.

Table 2. *Results for hyperparameters: activation: logistic, alpha: 0.1, hidden layer sizes: (10, 10), learning rate: adaptive, initial learning rate: 0.5, max iteration: 10000, solver: LBFGS*

accuracy	std
0.974999	0.027386
0.932499	0.062449
0.867499	0.112472
0.922499	0.057879
0.864999	0.179164

Table 3. *Results for hyperparameters: activation: logistic, alpha: 0.5, hidden layer sizes: (10, 10, 10), learning rate: adaptive, initial learning rate: 0.0001, max iteration: 10000, solver: LBFGS*

accuracy	std
0.877499	0.011
0.9475	0.048476
0.9475	0.029154
0.909999	0.076485

Table 4. *Results for hyperparameters: activation: logistic, alpha: 0.1, hidden layer sizes: (10, 10), learning rate: adaptive, initial learning rate: 0.5, max iteration: 10000, solver: LBFGS*

accuracy	std
0.9375	0.05244
0.922499	0.0886
0.9475	0.064031
0.9225	0.048476
0.954999	0.084557

Table 5. *Results for hyperparameters: activation: logistic, alpha: 0.1, hidden layer sizes: (12, 12), learning rate: adaptive, initial learning rate: 0.5, max iteration: 10000, solver: LBFGS*

accuracy	std
0.925	0.05244
0.9375	0.035553
0.9325	0.06819
0.93	0.04062

4. Conclusion

Due to the simplicity and relatively short learning time, the method of multilayer perceptron (MLP) was used to the calculate the hardening efficiency. The main goal of this article was to obtain satisfied hardening efficiency of steel. This was achieved by selecting the sigmoid activation function that gave the best result during network learning. Once it was found that the sigmoid activation function gave the best results, other hyperparameters were changed to achieve a hardening efficiency greater than 92 percent, which was successful. Future work should include more hidden neural networks with more neurons, database expansion or the use of different methods to achieve even better results.

Acknowledgments

We thank Sandi Baressi Šegota mag. ing. comp. for his mentorship during the work on this article.

Reference

[1] Wallwork, Harden & Temper, https://www.wallworkht.co.uk/content/harden_and_temper/ (04.05.2022.)

[2] Car, Zlatan, et al. "Modeling the spread of COVID-19 infection using a multilayer perceptron." *Computational and mathematical methods in medicine* 2020 (2020).. https://doi.org/10.1155/2020/5714714

[3] ScienceDirect, Multilayer Pereceptron, https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron (04.05.2022.)

[4] David Sterling and Wray Buntine, http://archive.ics.uci.edu/ml/datasets/Annealing

ISBN: 978-953-8246-26-5

Assembling humanoid robot (InMoov)

Karlo Džafić [1]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia), kdzafic@riteh.hr.

Abstract: InMoov is an open source humanoid robot whose pieces can be printed with a 3D printer. It is necessary to give an overview of the development of humanitarian robots and briefly describe the main representatives. In the final paper, it is necessary to present the basics of knowledge from 3D printing, programming in ROS, which are used in the construction of humanoid robots.

Keywords: Arduino, Artificial intelligence, Machine Learning, ROS, Robotics.

1. introduction

InMoov robot is the first life-size Open Source 3D-printed robot. Repeatable on any home 3D printer with an area of 12x12x12 cm, it is designed as a development platform for universities, laboratories, hobbyists, but above all for manufacturers. His concept, which is based on sharing and community, gives him the honour of being reproduced for countless projects around the world. Gael Langevin is a French sculptor and designer. He has been working for the biggest brands for more than 25 years. InMoov is his personal project, launched in January 2012 as the first open source prosthetic arm, it has led to projects like Bionico, E-Nable and many others [1][2][3][4].

2. Methodology

The method used to start making InMoov is to use a 3D printer with at least 12 x 12 x 12 print volumes. Some parts can exceed a few millimetres without more than requiring reprinting of the parts themselves. In the project itself, I used a type of plastic, Polylactic Acid, also known as PLA. To make it easier to organizationally print parts of a humanoid robot, the parts are divided into 8 categories that make up:

1. Hands and palms
2. Biceps
3. Shoulders and torso
4. Head, eyes and neck
5. Upper abdomen
6. Middle abdomen
7. Lower abdomen
8. Back

The next step is the hardware and the electronics themselves. As for the main electronics used for the two Arduino Mega [5] microcontrollers and the servo motors HK15298B, Hitec HS805BB and MG996. Their role is to control the operation of the servo motors via the Arduino and achieve dynamic movement of the robot itself. The map of the servo motor connections with the Arduino microcontroller is shown in Figure 1. It should be mentioned that in order to interact with the robot in use, there are two webcams, speakers and a microphone that receive commands from the man himself. MyRobotLab [6], Python scripts and the Robot Operating System (ROS) were used for the software part of the project. Considering InMoov is a combination of servo, Arduino, microphone, camera, kinect and computer. With MyRobot Lab we have a service for InMoov and it consists of many other peer services and allows easy initialization and control of these subsystems [7].

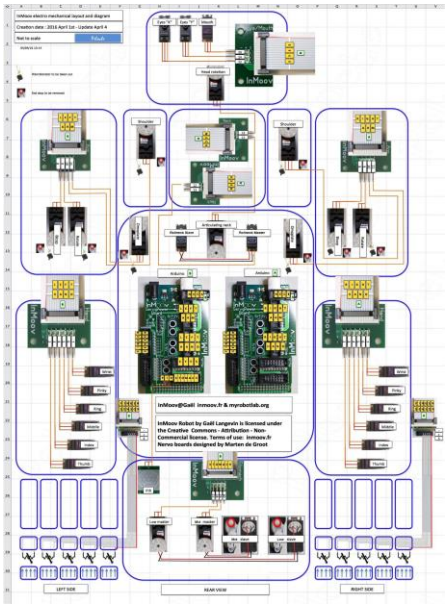


Figure 1. *Map connection V2*

3. Results and Discussion

The results show the parts made so far, namely the arms, the whole head and the whole abdomen. Each part when made must be tested with Arduino microcontrollers and servo motors using test scripts written in Python and in an Arduino IDE based on C and C ++. The reason for this is that avoiding the testing of individual parts can result in plastic cracking and the whole process must be done again, which immediately means extending the design and paying for expensive parts, which we want to avoid. In the continuation of this chapter, there are pictures of what the humanoid robot itself looks like in reality. In Figure 2. we can see an example of connecting 5 servo motors for all 5 fingers. A signal pin is connected to each servo motor from which it goes to the microcontroller. Also, each servo motor requires a separate 5v DC power supply.

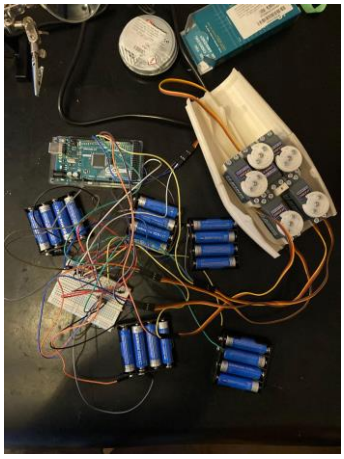


Figure 2. *Connections for testing hand*

Figure 3. shows fully made and assembled two hands. The transmission of the rotation of the servo motor is manifested in the movement of the fingers via braided fish lines that are otherwise used for fishing.

Figure 3. *Fully assembled hands and finger*

Perhaps the most attractive among humans is the very head of the InMoov robot. Inside the head there is a whole set related to the eyes as well as servo motors that move the jaw of the robot. You can see what the head itself looks like in Figure 4.

Figure 4. *Fully assembled head*

4. Conclusion

The main conclusions of this paper are that through the InMoov project we can get a better insight into the complexity of making a humanoid robot and how automation itself consists of several parts of technical areas to have one automated system. At the time of writing, we still have parts such as the shoulders and torso and the back itself to have a fully humanoid robot. Future plans for the InMoov project are certainly a focus on setting the robot's legs and dedicating work to robot stability and implementing knowledge so that the InMoov robot can walk.

Acknowledgments

I would like to thank professor Zlatan Car and Ivan Lorencin for their mentorship during the work on this article and for giving me the opportunity to present my work that I have been working on for 3 years .

Reference

[1] LANGEVIN, Gael. InMoov-Open Source 3D printed life-size robot. pp. URL: http://inmoov.fr, License: http://creativecommons. org/licenses/by–nc/3.0/legalcode, 2014.

[2] Escribà Montagut, Gerard, et al. Inmoov robot: building of the first open source 3D printed life-size robot. 2016. Bachelor's Thesis.
http://inmoov.fr/

[3] VALENCIA, Nicolás Ortiz, et al. Movement detection for object tracking applied to the InMoov robot head. In: 2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA). IEEE, 2016. p. 1-6.
https://ieeexplore.ieee.org/abstract/document/7743328

[4] RODRÍGUEZ OLIVERA, Carlos Tomás, et al. 3D printed robot. 2018
https://uvadoc.uva.es/handle/10324/36846

[5] Arduino Mega 2560 Rev3
http://store.arduino.cc/products/arduino-mega-2560-rev3

[6] MyRobotLab
http://myrobotlab.org/

[7] MyRobotLab - InMoov
http://myrobotlab.org/service/inmoov

ISBN: 978-953-8246-26-5

Calculating the shortest path using Breadth-First Search

Matija Kušan [1*], Grgur Bosiljevac [2]

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, gbosiljevac@riteh.hr, mkusan@riteh.hr

Abstract: This paper tries to solve the problem of calculating the shortest path on a map using a BFS algorithm. The problem is solved using Python. The goal was to create a matrix map that represents occupied and unoccupied space, and then draw the shortest path through unoccupied space. The given end result was satisfactory as it did actually draw the shortest possible path.

Keywords: Algorithm, Breadth-First Search, graph traversal, layer, node, Python

1. introduction

This article will discuss the implementation of Breadth-first search, or BFS for short, in calculating the shortest possible path on a set map, or a graph. Breadth-first search (BFS) is an important building block of many graph algorithms, and it is commonly used to test for connectivity or compute the single-source shortest paths of unweighted graphs [1]. For example, the nature of the relationship between two vertices in a semantic graph can be determined by the shortest path between them using BFS [6]: any path to the solution that is of length n will be found when the search reaches depth n, and this is guaranteed to be before any node of depth < n is searched [2]. The shortest path is calculated by implementing the graph traversal technique, where each node is visited exactly once (if we let the process run until completion and if the graph is connected) [4]. In addition to BFS, there are other techniques that vary in the order in which they visit the nodes; examples include Depth-First Search (DFS), Forest Fire (FF) and Snowball Sampling (SBS) [4]. Breadth-first search is often much more efficient than depth-first search, because the latter cannot detect duplicate nodes representing the same state and generates all paths to a given state [5].

2. Methodology

As already stated in the introductory segment, the goal of this project was to draw the shortest path on a premade graph. Mainly, this would be used to draw the shortest path on a map for a robot to travel. The algorithm (BFS) efficiently visits and marks all the key nodes in a graph in an accurate breadthwise fashion. It selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected one [3]. Once the algorithm visits and marks the starting node, it moves towards the nearest unvisited nodes and analyses them. Once visited, all nodes are marked. These iterations continue until all the nodes of the graph have been successfully visited and marked [3]. This process is shown in figure 1.

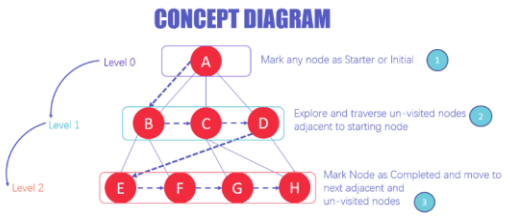


Figure 1. *Concept diagram of a BFS algorithm [3]*

The software package used to draw the graph on which the mentioned path is drawn is Spyder, using Python. Firstly, a map needs to be drawn. The method used to draw a map in this article is using a matrix. Unoccupied space is marked as 0, and occupied space (barricades) as well as the edges of the map are marked as 1. Secondly, graph traversal is in play. It requires the algorithm to visit, check, and/or update every single un-visited node [3]. Two points in the graph are selected as the start and end point. The algorithm starts from the start point. It is important that nodes do not get visited twice, hence, the queue is introduced. Four neighbouring nodes around the current node are checked, and if they are unvisited, they are then marked and enqueued so that they do not get visited twice. Thirdly, the shortest path is chosen using the algorithm.

3. Results and Discussion

Drawing the map was the first step of this process. As already mentioned, the map is in a matrix form, where unoccupied space is marked as 0, while occupied space (barricades) and the edges of the map are marked as 1. The map is shown in figure 2. The black spaces represent barricades, the white spaces represent unoccupied space, and, for the sake of transparency, red spaces represent the start (upper left) and end (bottom right) points. The next step is to do a graph traversal by an algorithm that had already been explained earlier. The result is shown in figure 3.

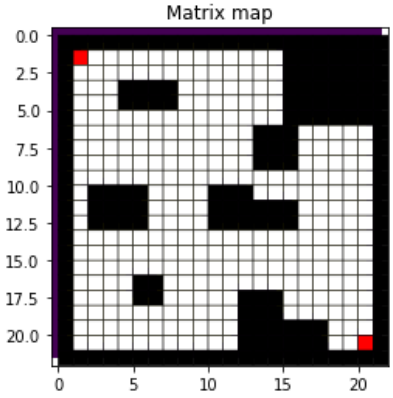


Figure 2. *Map overlay for drawing the shortest path, drawn in Spyder*

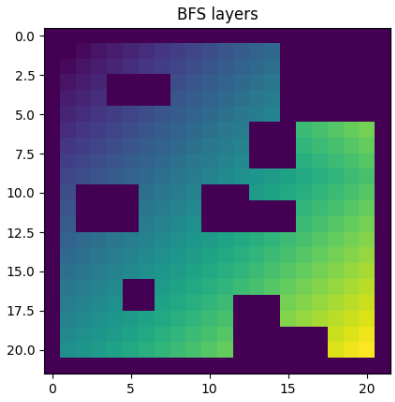


Figure 3. *BFS graph traversal by layers, drawn in Spyder*

As the queue progresses, layers have an increasingly higher value, as it is shown in figure 3. Layers that have higher values have their colour more yellow than those with lower values. Finally, it is time to calculate and draw the shortest path on a given map. The algorithm starts from the starting node. It is checking 8 nodes around it, picking a node with a higher queue number, preferably moving diagonally from itself. This process is repeated until the current node coordinates match the destination node coordinates. Th results of this process are shown in figure 4.

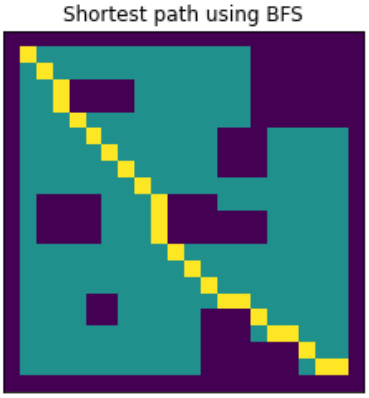


Figure 4. *Shortest path calculation using BFS, drawn in Spyder*

4. Conclusion

From looking at the calculated shortest path in figure 4, it can be concluded that the algorithm works relatively well, and since it's checking its surroundings in the graph traversal part, it should be acknowledged that the algorithm should work perfectly fine in an unknown environment. This work on the implementation of BFS could be used as a sidenote, or even a base for some more advanced methods such as a Voronoi Graph.

Acknowledgments

We want to thank the assistant Sandi Baressi Šegota mag. ing. comp. for his mentorship during our work on this article. We would also like to thank all those who organized this conference and enabled us students to participate with our work.

Reference

[1] S. Beamer, K. Asanovic, D. Patterson et al, "Direction-Optimizing Breadth-First Search" *Electrical Engineering and Computer Sciences Department, University of California, Berkeley,* 2012., pp. 1-10 https://doi.org/10.1109/SC.2012.50

[2] Bundy, A., Wallen, L. et al, "Breadth-First Search". In: Bundy, A., Wallen, L. (eds) Catalogue of Artificial Intelligence Tools. *Symbolic Computation. Springer, Berlin, Heidelberg* https://doi.org/10.1007/978-3-642-96868-6_25

[3] From web: https://www.guru99.com/breadth-first-search-bfs-graph-example.html

[4] M. Kurant, A. Markopoulou and P. Thiran, "On the bias of BFS (Breadth First Search)," *2010 22nd International Teletraffic Congress (ITC 22)*, 2010, pp. 1-8 https://doi.org/10.1109/ITC.2010.5608727.

[5] Korf, Richard E., and Peter Schultze., "Large-scale parallel breadth-first search." *AAAI. Vol. 5. 2005.*, pp. 1380-1385 https://www.aaai.org/Papers/AAAI/2005/AAAI05-219.pdf

[6] Yoo, Andy, et al. "A scalable distributed parallel breadth-first search algorithm on BlueGene/L." SC'05: *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. IEEE*, 2005. https://www.osti.gov/servlets/purl/919215

ISBN: 978-953-8246-26-5

Laser sensor error by distance

Karla Šešelja, Marko Škara

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, kseselja@riteh.hr
[2] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, mskara@riteh.hr

Abstract: This paper presents the basic operating principles of laser sensors and program that simulates a laser sensor error by distance. Two types of data were used, compared at the same time. Default parameters are distance, sensor error, measurement time and sampling frequency. Based on the obtained data, sensor variances were determined at two distances. Also in this paper, the Python code and the results of the obtained work will be presented. The conclusion contains comments on the results and the advantages and disadvantages of laser sensors.

Keywords: laser, mean value, measurement error, Python, sensor, standard deviation, variance.

1. Introduction

There is a wide variety of applications for optical sensors like laser sensors. They are used in many sectors of industry and research for distance measurement. Particularly in confined spaces non-contacting sensor systems are the best solution.
In this paper it will be discussed and simulated effect of measurement error on laser sensors. Laser distance sensors (figure 1.) measure distances and allow it to take measurements at great distances. These distance sensors work on the basis of the Time-Of-Flight (ToF) principle, which means that the sensor emits a laser beam and receives the reflection from it. The time that elapses between sending and receiving the laser light ensures that the laser distance sensor can internally determine the distance. The distance over which the measurements can be taken differs per series.

Figure 1. *Laser distance measuring sensor.[1]*

There are different principles for measuring laser distance sensors, such as:

- Pulse laser distance sensor – It is a pulse laser with a very short duration, when it reached the target part of the energy will be reflected back. The reflected pulsed laser is called an echo. The echo returns to the rangefinder and is received by a photoelectric detector. According to the interval between the main wave signal and the echo signal, that is, when the laser pulse travels from the laser to the target to be measured, the distance of the target to be measured can be calculated.
- Phase laser distance sensor - the phase change of the modulated signal is used when the laser is propagated in space. According to the wavelength of the modulated wave, the distance represented by the phase delay is calculated. The indirect method of phase delay measurement is used instead of directly measuring the time required for the round trip of the laser to achieve distance measurement (Figure 2). The accuracy of this method can reach the millimeter level.

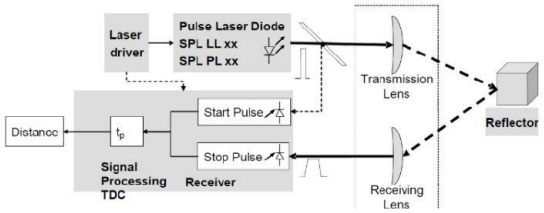


Figure 2. *Working principle of laser distance measuring device [2]*

- Triangulation laser distance sensor – this measurement principle is that the light emitted by the laser is focused on the surface of the measured object after being focused by the condensing lens, and the receiving lens receives the scattered light from the incident light spot and images it on the photoelectric position detector on the sensitive side. When the object moves, the relative distance of the object movement is calculated by the displacement of the light spot on the imaging surface. The resolution of triangulation laser ranging is very high, which can reach the order of microns.

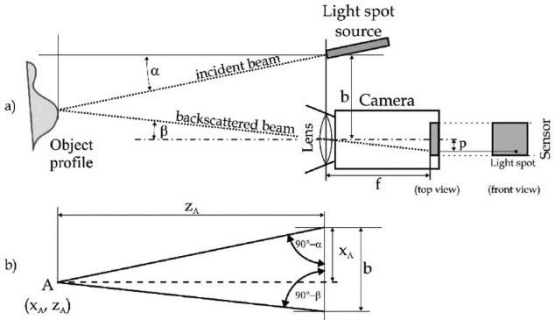


Figure 3. *Triangulation principle [2]*

- Interferometric laser distance sensor - By moving the measured target and measuring the coherence, the distance increment measurement is completed by counting, so the sensitivity of

the interferometric measurement is very high, which can reach the nanometer level.

Laser distance sensors are excellent for the use in engineering for quality control and process monitoring. They are also suitable for automation, chemical industry, medical technology and special machine construction.

In the next part it will be simulated a 1% measurement error on distance of 10m and 2% measurement error for distances greater than 10m. This simulation is necessary to determine effectiveness of laser sensors on multiple distances.

2. Methodology

The work in the program is divided into two parts. For a dimension up to ten meters the percentage error is 1%, and for dimensions larger than 10 meters the percentage error is 2%. After importing necessary libraries such as numpy and matplotlib it is necessary to define basic parameters such as:

- Simulation time: 10s
- Sampling frequency 100Hz

For the first example distance is set to ten meter with percentage error of 1%. Maximum error value is product of distance and percentage error. To simulate measurement error, a random number was generated with value between negative and positive maximum error value of 1%. Also, *for* loop was used to simulate sensor measurement. With default time and sampling frequency we get the result of 1000 samples of measured values. Then, these 1000 samples are shown in a form of graph. From this graph it is calculated mean value, standard deviation and variance which will be later used for assessing the accuracy of laser sensors.

For the second example distance is over ten meters with percentage error of 2%. In this part, a list was created of

```
import numpy as np
import matplotlib.pyplot as plt
f=100    #frekvencija 100Hz
T=1/f    #perioda
t=10     #vrijeme uzorkovanja
n=T*t
#-----------------------
d1=10        #10
pogreska_1=0.01
max_pogreska_1=d1*pogreska_1
A_1=list()

d2=12        #>10
A_2=list()
l_2=list()
matrix=list()
```

random values between 0.1 and 100. Next step, zero values were substituted with one and zero was substituted with minus one. List with measurement values (0.1 – 100) were multiplied with list of 1's and -1's to get negatives as well as positive values. Using the second *for* loop the simulation was completed and the results were presented in a form of graph.

```
#-----------------------
for i in range(0,100):
  A_1.append(d1+np.random.uniform(-
max_pogreska_1,max_pogreska_1))
  l_2.append(np.random.uniform(0.1,1))
  matrix.append(np.random.randint(-1,1))
  if (matrix[i]==0):
     matrix[i]=1.0
  if (matrix[i]==-1):
     matrix[i]=-1.0
n=np.arange(0,10,0.1)
#figure 1
plt.figure()
plt.plot(n,A_1)
plt.grid()
plt.show()
plt.xlabel('t [s]')
plt.ylabel('d - distance [m]')
plt.title('Percentage error 1% and d=10')
average=sum(A_1)/len(A_1)
variance=np.var(A_1)
standard_deviation=np.std(A_1)
print(average)
print(variance)
print(standard_deviation)
od=np.multiply(l_2,matrix)
for i in range(0,100):
  A_2.append(d2+d2*od[i])
#figure 2
plt.figure()
plt.plot(n,A_2)
plt.grid()
plt.show()
plt.xlabel('t [s]')
plt.ylabel('d - distance [m]')
plt.title('Percentage error 2% and d>10')
average2=sum(A_2)/len(A_2)
variance2=np.var(A_2)
standard_deviation2=np.std(A_2)
print(average2)
print(variance2)
print(standard_deviation2)
```

3. Results and Discussion

After analyzing the obtained results, it is evident that the results are changing and they are not completely accurate. Accuracy can depend on various influences which cannot be controlled as we can see on Figure 4. and Figure 5.

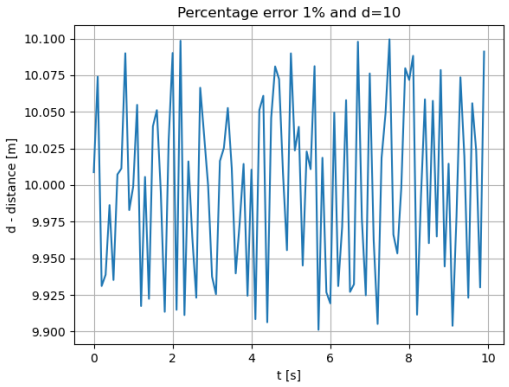


Figure 4. *Simulation results for distance of 10 meter and percentage error 1%.*

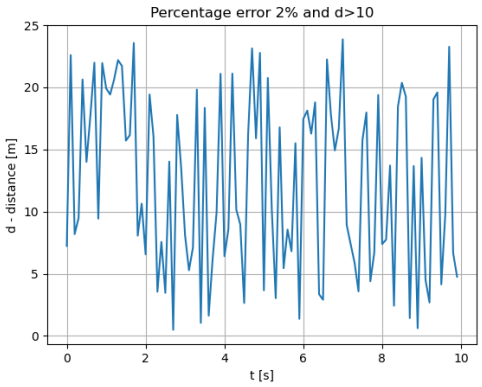


Figure 5. *Simulation results for distance greater than 10 meter and percentage error 2%.*

Table 1. *Simulation results.*

Distance	10[m]	>10[m]
Mean value	9.9979	11.9165
Standard deviation	0.0032	52.9767
Variance	0.0562	7.2785

4. Conclusion

Laser sensors are typically more expensive than analogue measuring devices simply because of the technology involved. They can be delicate because very precise calibration must be maintained in order for them to work properly. By analysing the given results, higher measurement distance yields higher measurement errors as we can see in Table 1. Mean value of these two distances are within reasonable range, but more noticeable errors occur while analysing standard deviation and variance.
However, laser sensor can be extremely accurate. More than anything else, using measuring devices that utilize lasers can improve the precision of your operation in ways that other measuring devices are simply incapable of. Also, they are often very light and easy to use.

Acknowledgments

We thank Sandi Baressi Šegota, mag. ing. comp. for his mentorship and help during the work on this article.

Reference

[1] „Laser distance sensor“, https://www.top1sensor.com/product/skd-20s-laser-distance-sensor/
[2] „Laser sensor“, https://www.apogeeweb.net/electron/The-Function-And-Principle-Of-Laser-Sensor.html
[3] Dwayne C. Carmer And Lauren M. Peterson „Laser Radar in Robotics“
[4] “The Pros and Cons of Lasers in Measuring Devices” https://www.srpcontrol.com/the-pros-and-cons-of-lasers-in-measuring-devices/
[5] Baressi Šegota, Sandi, „Senzori u robotici“
[6] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah „A Low-Cost Laser Distance Sensor“
[7] The Python Language Reference https://docs.python.org/3/reference/

ISBN: 978-953-8246-26-5

Calculation and visualization of mobile robot kinetic energy

Anja Režek [1*], Antonia Burić [2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia,arezek@riteh.hr.
[2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, aburic@riteh.hr.

Abstract: This paper presents calculation and visualization of mobile robot kinetic energy. Using the equations, the values of kinetic energy of linear motion, kinetic energy of robot rotation, kinetic energy of both wheels rotation and total kinetic energy were first calculated. Visualization of the calculated kinetic energies as a function of wheel speeds are shown in the form of 3D graphs. Both the calculation and visualization were done by using Python programming language. It can be concluded that, for the given robot configuration, the highest influencing factor are the kinetic energies of the wheel rotation.

Keywords: Kinetic energy, mobile robot, Python, visualization

1. Introduction

Dynamic robot modelling is a key step in the mobile robot design [1]. One of the main parts of this process is the calculation of the associated kinetic energies, produced by the robot motion [2]. For this reason, this paper discusses the process of calculating and visualizing the various kinetic energies produced by the motion of a mobile robot. In this paper, the methodology which includes the used kinetic energy model, and the used mobile robot parameters are given first, followed by the presentation and discussion of the obtained visualizations. Finally, the conclusions are drawn.

2. Methodology

In Table 1 are given initial parameters for the calculation of kinetic energies.

Table 1. *Initial parameters*

Robot weight [kg]	1
Robot diameter [m]	0.1
Moment of inertia (I_A=I_C=const.)	1
Distance between the wheel and the center [m]	0.2
Wheel diameter [m]	0.1

The formulas used to calculate kinetic energies are shown below.
In Equation (1) formula for kinetic energy of linear robot motion is shown [3]

$$T_l = \frac{m}{2}\left(\frac{r^2}{4}\left(\dot{\theta}_R + \dot{\theta}_L\right) + \frac{r^2 d^2}{D^2}\left(\dot{\theta}_R + \dot{\theta}_L\right)^2\right), \qquad (1)$$

where m is robot weight in kilograms, r and D are wheel and robot diameters in meters, d is distance between the wheel and the center in meters and $\dot{\theta}_R, \dot{\theta}_L$ are right and left wheel speeds.
Formula for kinetic energy of robot rotation is given in Equation (2) [3]

$$T_r = \frac{r^2}{2D^2} I_A\left(\dot{\theta}_R - \dot{\theta}_L\right)^2, \qquad (2)$$

where I_A is the moment of inertia of the wheel which is assumed to be a constant in this research.
Kinetic energy of both wheels rotation is calculated like shown in Equation (3) [4]

$$T_{kr} = \frac{1}{2} I_0\left(\dot{\theta}_R + \dot{\theta}_L\right)^2, \qquad (3)$$

where I_0 is the moment of inertia of the robot which is assumed to be a constant in this paper.
Total kinetic energy of a mobile robot is equal to the sum of previously mentioned kinetic energies, as shown in Equation (4) [4]

$$T = T_l + T_r + T_{kr}. \qquad (4)$$

It is also necessary to define wheel speeds. As shown in Figure 1, 1000 values ranging from 0 to 10 were taken.

```
print("Creating linspaces for speeds.")
v_r = np.linspace(0,10,1000)
v_l = np.linspace(0,10,1000)
```

Figure 1. Defining wheel speeds

For the purpose of creating 3D graphs, it is necessary to make a transformation in the *X*, *Y*, *Z* axis, using the command *meshgrid* from the *numpy* library, as shown in Figure 2.

```
print("Creating linspaces for speeds done.\nCreating meshes.")
X, Y = np.meshgrid(v_r, v_l)
print("Meshes done.\nCalculating z-axis.")
Z1 = T_l(X, Y)
Z2 = T_r(X, Y)
Z3 = T_kr(X, Y)
Z4 = T(X, Y)
```

Figure 2. Transformation of X, Y, Z axis

Figure 3. shows the part of the code in which the necessary parameters and graph names are set and this action is repeated for each of the kinetic energies.

```
#plot T_l
fig = plt.figure(figsize=(12,8), dpi=100)
ax = plt.axes(projection='3d')
ax.set_xlabel("$v_r$")
ax.set_ylabel("$v_l$")
ax.set_zlabel("$T_{l}$")
ax.set_title("Dependance of $T_{l}$ on $v_r$ and $v_l$.")
ax.plot_surface(X, Y, Z1, rstride=10, cstride=10,cmap='viridis',
edgecolor='none')
```

Figure 3. Plotting of 3D graph

3. Results and Discussion

Figure 4 displays the dependence of linear motion energy on the wheel speeds of the mobile robot. It can be seen that the linear motion energy increases when one of the speeds increases while the other is low. When both speeds are high the linear motion energy is relatively low. Compared to the values of other kinetic energies the maximal value of linear motion energy is low.

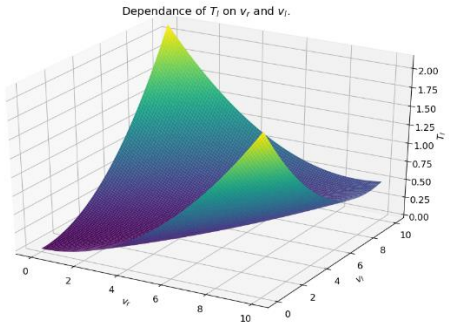


Figure 4. Kinetic energy of linear motion as a function of wheel speeds

Figure 5 displays the dependence of robot rotation energy on the wheel speeds of the mobile robot. Behavior of this graph is similar to the one in Figure 4. The robot rotation energy increases when one of the speeds increases while the other is low. When both speeds are high the robot rotation energy is relatively low. Compared to the value of linear motion energy, the maximal value of robot rotation energy, is higher, but compared to others, value is still low.

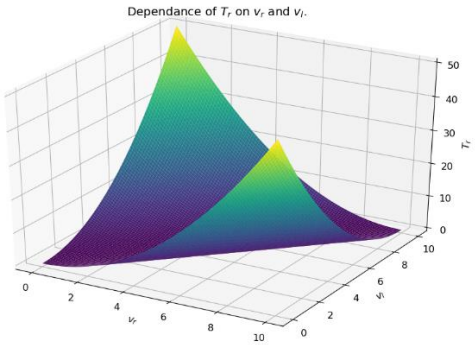


Figure 5. Kinetic energy of robot rotation as a function of wheel speeds

Figure 6 displays the dependence of both wheels rotation energy on the wheel speeds of the mobile robot. It can be seen that the both wheels rotation energy increases when both of the speeds increases. When both speeds are low the both wheels rotation energy is also low. Compared to the values of other kinetic energies the maximal value of both wheels rotation energy is high.

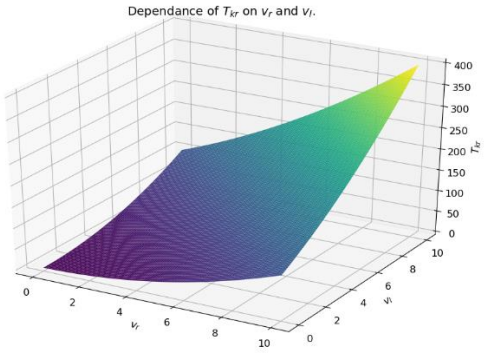


Figure 6. Kinetic energy of both wheels rotation as a function of wheel speeds

Figure 7 displays the dependence of total kinetic energy on the wheel speeds of the mobile robot. Behavior of this graph is similar to the one in Figure 6. Total kinetic energy increases when both of the speeds increases, and inversely when both speeds are low total energy is low..Compared to the values of other kinetic energies the maximal value of total kinetic energy is the highest.

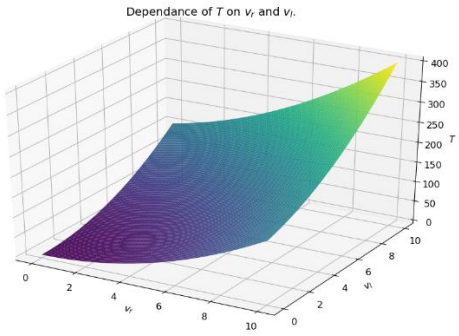


Figure 7. Total kinetic energy of a mobile robot as a function of wheel speeds

4. Conclusion

The goal of the paper, which was to visualize kinetic energies of the mobile robot, was successfully accomplished. From the obtained graphs, it can be concluded that the kinetic energy of both wheel rotation has the highest influence on the total kinetic energy, followed by the rotational kinetic energy. Finally, the linear kinetic energy has the lowest influence. This is obviously only valid for the given robot manipulator configuration, and future work may include expanding this research to include different robot configurations.

Reference

[1] Baressi Šegota, Sandi, et al. "Dynamics Modeling of Industrial Robotic Manipulators: A Machine Learning Approach Based on Synthetic Data." Mathematics 10.7 (2022): 1174

[2] Baressi Šegota, Sandi, et al. "Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms." International Journal of Advanced Robotic Systems 17.2 (2020): 1729881420908076.

[3] Thrun, Sebastian. "Probabilistic robotics." Communications of the ACM 45.3 (2002): 52-57.

[4] Kelly, Alonzo. Mobile robotics: mathematics, models, and methods. Cambridge University Press, 2013.

ISBN: 978-953-8246-26-5

Infrared sensor loss simulation

Dani Šestan[1], Dean Stjepanović[2]

[1] Faculty of Engineering – University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, dsestan@riteh.hr.
[2] Faculty of Engineering – University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, dstjepanovic@riteh.hr.

Abstract: This paper presents a simulation of infrared sensor. The operation of IR sensors and their application will be described first. Then, for a predetermined sensor temperature, the distance of the object from the sensor will increase until it is obtained at what distance at which the object is not visible. Also, it will show how the temperature of the object changes with distance from the sensor itself and the written code in Python.

Keywords: Infrared sensor, laser, loss simulation, Python

1. introduction

Today, infrared (IR) technology is being widely used in everyday's life and industry. The simplest example is communication between a TV and a remote control. Televisions use an IR sensor to receive the signal sent from the remote control. Lower power usage, simple design and convenient features are the main benefits of IR sensors. As IR radiation is not visible to the human eye, it can be found in areas of the microwave and visible spectrum. Infrared waves typically have wavelengths between 0.75 and 1000 µm. The IR spectrum itself can be divided into three regions: near (0.75 – 3 µm), mid (3 – 6 µm) and far-infrared (higher than 6µm).
An infrared sensor, shown in Figure 1, is an electronic device that emits to sense segments of the environment. It can detect movement and measure the heat of an object (in infrared spectrum, all objects radiate some kind of thermal radiation).

Figure 2. *An infrared sensor*

One IR sensor consists of five elements. These are infrared source, transmission medium, optical component, infrared detectors and signal processing. The infrared source is an IR LED that emits IR radiation that is not visible to the human eye. IR receivers or sensors detect the emitted radiation and come in the form of photodiodes or phototransistors. The photodiode's output voltage and resistance change in proportion to the IR light received. It is important that the wavelength of the receiver corresponds to the wavelength of the transmitter. On Figure 2 is shown the described working principle of the IR sensor.

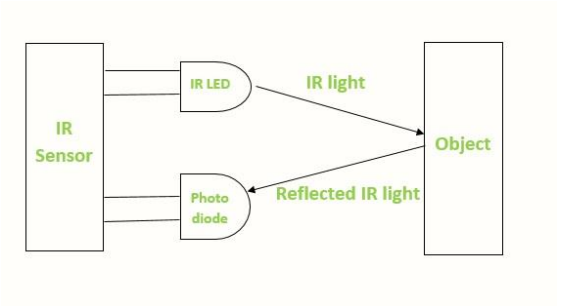


Figure 3. *Working principle of IR sensor*

The emitted radiation of the IR transmitter reaches the object, and part of the radiation is reflected back to the IR receiver. Based on the reception intensity of the IR receiver, the sensor output is defined.
It was initially mentioned that IR sensors are widely used. Some of the applications are: night vision devices, radiation thermometers, infrared tracking, meteorology, gas detectors etc.

2. Methodology

Like any body or object radiates a certain amount of thermal radiation it is possible to determine at what distance its temperature will be zero degrees Celsius. This dependence is given by the following equation (1). The calculated new temperature decreases with the square of the distance:

$$T_{new} = \frac{T_{sensor}}{r^2}, \quad (1)$$

where T_{sensor} stands for initial temperature sensor, r for object distance from the sensor and T_{new} for the new calculated temperature.
Approaching the code writting (presented on Figure 3) was easy. For the set temperature of IR sensor, distance was increased using arange command form the numpy library. Calculated data was stored in the matrix T_{new}.

```
import numpy as np
import matplotlib.pyplot as plt

T=100     #set temperature of IR sensor
r=np.arange(0.01, 100.01, 0.01)
Tnew=T/(r**2)
plt.figure()
plt.plot(r,Tnew,linewidth=6)
```

```
plt.rc('axes', titlesize=64)
plt.rc('figure', titlesize=64)
plt.xlim(0,10)
plt.ylim(0,100)
plt.xlabel('Distance [m]')
plt.ylabel('Temperature [°C]')
plt.title('Temperature depedence on distance')
plt.grid()
```

Figure 4. *The Python code*

3. Results and Discussion

After analysing the collected data, it is evident that results are not the same. As the initial temperature of the IR sensor increases, distance at which the object will be invisible also increases. Following figures (Figure 4, Figure 5 and Figure 6) graphically show dependencies for a different set sensor temperature.

With an initial IR sensor temperature of 100°C (Figure 4), the IR sensor will not see an object at approximately 10 meters.

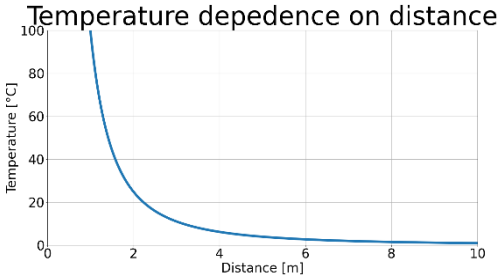


Figure 5. *Temperature dependence at 100°C*

By reducing the initial temperature of the IR sensor to 50°C (Figure 5), the distance at which the object will not be visible has also decreased (to approximately 7 meters).

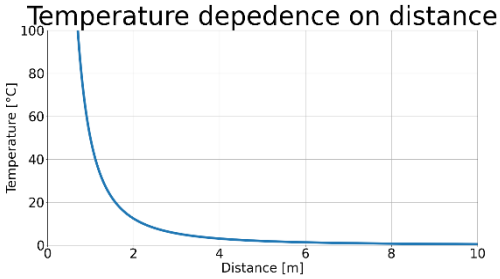


Figure 6. *Temperature dependence at 50°C*

By further reducing the initial temperature, the distance at which the object will not be visible is further reduced. Figure 6 shows the temperature dependence for the initial IR sensor temperature of 25°C.

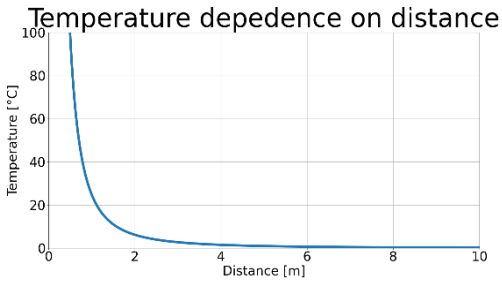


Figure 7. *Temperature dependence at 25°C*

4. Conclusion

As IR sensors are being widely used (military, meteorology, water analysis, radiation thermometers, infrared tracking, gas detectors etc.), it is important that they are as accurate and efficient as possible.

For the set IR sensor temperatures, distance of the object from the sensor is increased until the object becomes invisible. Initial temperature decreases with the square of the distance and the obtained results are graphically displayed.

For further research, external environmental influences that affect the operation and quality of IR sensors should be taken into the consideration.

Acknowledgments

We would like to thank Sandi Baressi Šegota, mag. ing. comp; for his mentorship during the work on this article".

Reference

[1] Robu.in, IR sensor Working and Applications, https://robu.in/ir-sensor-working/ (09.05.2022.)

[2] AZO Sensors, The Working Principle and Key Applications of Infrared Sensors, https://www.azosensors.com/article.aspx?ArticleID=339 (09.05.2022.)

[3] Baressi Šegota, Sandi, „Senzori u robotici"

ISBN: 978-953-8246-26-5

Ultrasonic sensor measurement error simulation

Tomo Galović [1], Teodor Grenko [2]

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, tgalovic@riteh.hr
[2] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, tgrenko@riteh.hr

Abstract: This paper will describe ultrasonic sensor, principle of operation and the principle of calculating distance using the above. There will be shown simulation of sensor measurements for different values of the distance. Second part will analyse the procedure of simulating sensor values using Phyton. Program written for purposes of this paper will be explained and described. The third part includes the results which are given in a form of graphs and table. Measurement error will be evaluated based on standard deviation, mean value and variance of given measurements. The conclusion contains comments on the results and the advantages and disadvantages of ultrasonic sensors

Keywords: Ultrasonic, transmitter, receiver, measurement error, Python, mean, standard deviation, variance

1. introduction

In this paper it will be discussed and simulated effect of measurement error on ultrasonic sensors. Ultrasonic sensors are used for sensing proximity and detecting levels with high reliability [1]. Using ultrasonic sound waves it is possible to determine distance between sensor and object [1].

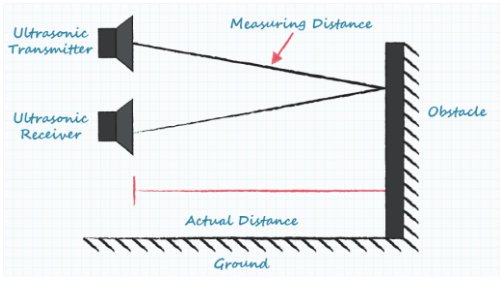


Figure 8 *Basic working principle of ultrasonic sensors*

Ultrasonic sensor is made of two parts; ultrasonic transmitter and ultrasonic receiver as shown in figure 1 [2]. Transmitter emits a chirp with a frequency range between 23kHz and 40kHz [2]. This frequency is above human hearing range and hence the term ultrasonic [2]. After transmitting, it is then measured the time it takes for sound to bounce off of the object and return to receiver. It is known that sound travels at a speed of 343 m/s and with this information it is possible to calculate the distance between sensor and object with the following equation [2]:

$$d[m] = \frac{t[s] * 343\ \left[\frac{m}{s}\right]}{2}$$

Where: d – distance between sensor and object, t – time between emitting and receiving sound. To compensate for time it takes for signal to return after bouncing off of an object, it is necessary to divide the equation with half.
In the next part it will be simulated a 2% measurement error on distance of 1m, 5% measurement error on distance of 10m and >10% measurement error for distances greater than 10m. This simulation is necessary to determine effectiveness of ultrasonic sensors on multiple distances.

2. Methodology

Sensor error percentage is divided on three parts. For dimensions under one meter percentage error is 2%, dimensions above one meter and under ten meters percentage error is 5% and for dimensions above ten meters percentage error is not constant and its value can be number between 10% and 100%.
First part of simulation which includes dimensions under one meter was realized using code that is shown on Figure 2.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt


        t=100
        f=10
        T=1/f
        d1=1
        error_1=0.02
        max_error_1=d1*error_1
        out_1=list()
        for i in range(0,1000):
            out_1.append(d1+np.random.uniform(-max_error_1,max_error_1))
        n=np.arange(0,t,T)
        plt.figure()
        plt.plot(n,out_1)
        plt.grid()
        plt.show()
        plt.xlabel('Time [s]')
        plt.ylabel('Distance [m]')
        plt.title('Distance: 1m, Percentage error = 2%')
        aver_1=sum(out_1)/len(out_1)
        std_1=np.std(out_1)
        var_1=np.var(out_1)
```

Figure 2 *Python code for simulating distance of 1m*

Firstly, it is necessary to import libraries such as *numpy* and *matplotlib*. Simulation time was 100s with the sampling frequency of 10Hz. Distance is set to one meter with percentage error of 2%. Maximum error value is product of distance and percentage error. To simulate measurement error, a random number was generated with value between negative and positive maximum error value of 2%. Next, *for* loop was used to simulate sensor measurement. With the sampling frequency of 10Hz and simulation time of 100s we are given the result of 1000 samples of measured values. These 1000 samples are then shown in a form of graph. From this graph it is calculated mean value, standard deviation and variance which will be later used for assessing the accuracy of ultrasonic sensors. Same code can be applied for distance between one and two meters where percentage error was 5%. However, for simulating distances over ten meters whose percentage error is over 10% it is necessary to alter the code, as shown in Figure 3. For the distance over ten meters measurement error is not constant and its value can vary between 10%

and 100%. Figure 3 shows implementation in Python of this effect.

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
        t=100
        f=10
        T=1/f
        d1=1
        d3=15
        out_3=list()
        limit_3=list()
        jedinice=list()
        n=np.arange(0,t,T)
        for i in range(0,1000):
           limit_3.append(np.random.uniform(0.1,1))
           jedinice.append(np.random.randint(-1,1))
           if (jedinice[i]==0):
               jedinice[i]=1.0
           if (jedinice[i]==-1):
               jedinice[i]=-1.0
        odstupanje=np.multiply(limit_3,jedinice)
        for i in range(0,1000):
           out_3.append(d3+d3*odstupanje[i])
        plt.figure()
        plt.plot(n,out_3)
        plt.grid()
        plt.show()
        plt.xlabel('Time [s]')
        plt.ylabel('Distance [m]')
        plt.title('Distance: 15m, Percentage error > 10%')
        aver_3=sum(out_3)/len(out_3)
        std_3=np.std(out_3)
        var_3=np.var(out_3)
```

Figure 3 *Python code for simulating distance greater than 10m*

First step for simulation was to create a list of random values between 0.1 and 100. It is important to notice that error could also be negative value and therefore a list of binaries (0 or 1) was created. Next, zeroes were substituted with 1 and 0 was substituted with -1. After substation, list with measurement values (0.1 – 100) were multiplied with list of 1's and -1's to get negatives as well as positive values. Using the second *for* loop the simulation was completed and the results were presented in a form of graph.

3. Results and Discussion

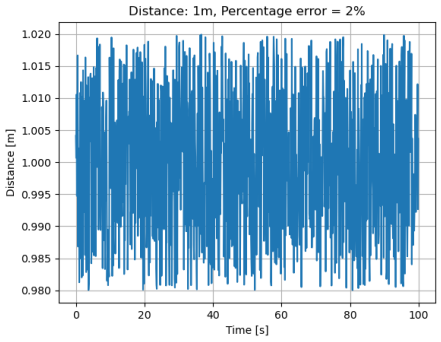


Figure 4 *Simulation results for distance of 1m*

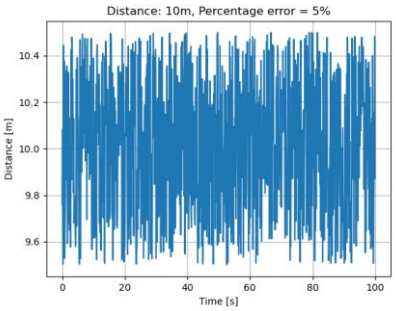


Figure 5 *Simulation results for distance of 10m*

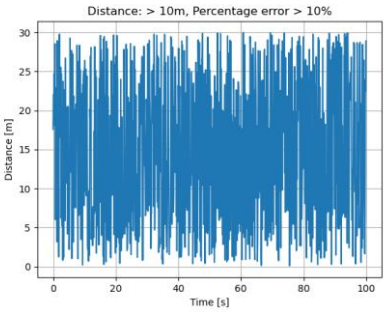


Figure 6 *Simulation results for distance greater than 10m*

Table 1. *Simulation results*

Distance	1[m]	5[m]	>10[m]
Mean	0.9995	9.9877	15.0473
Standard deviation	0.0113	0.0113	9.1111
Variance	0.0001	0.084	83.011

Analysing figures 4, 5 and 6 it is noticeable that measurement is very inconsistent. Measurements can depend on many different parameters such as humidity, air pressure, air currents and so on, which cannot be controlled and this effect is illustrated in figures above [3].

4. Conclusion

By analysing the given results, it is visible that higher measurement distance yields higher measurement errors. Best results are shown at closer range.

Mean value of all three distances are within reasonable range, however more noticeable errors occur while analysing standard deviation and variance as shown in table 1. Analysing the standard deviation and variance, we can conclude that there is greater spread of values at greater measuring distances.

In conclusion, ultrasonic sensors are cheap, simple, lightweight and with small power consumption [4]. However, their accuracy is decreased with greater distances and therefore should only be used in short distance measurements.

Acknowledgments

We thank mag. ing. comp. Sandi Baressi Šegota for his mentorship during the work on this article.

References

[1] Burnett, Roderick, „Understanding How Ultrasonic Sensors Work“ https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm
[2] Smoot, Jefff, „The Basics of Ultrasonic Sensors“ https://www.cuidevices.com/blog/the-basics-of-ultrasonic-sensors
[3] Ultrasonic Sensor Accuracy https://senix.com/ultrasonic-sensor-accuracy/
[4] Baressi Šegota, Sandi, „Senzori u robotici“
[5] The Python Language Reference https://docs.python.org/3/reference/

ISBN: 978-953-8246-26-5

Design of a DC motor mathematical model using Python

Marko MAVRINAC [1]

[1] Electroindustrial and trade school Rijeka, Zvonimirova 12, 51000 Rijeka, Croatia, mavrinac07@eios.hr

Abstract: This paper compares the design of theoretical systems and their mathematical models in Simulink and Python. An example of such a system is provided in the paper and potential uses of this design philosophy are highlighted.

Keywords: DC motor, optimization, mathematical model, Python, numpy

1. Introduction

Modern control systems used in different industries like robotics and aeronautics support such delicate and expensive systems and equipment, there is no room for error when it comes to their inner workings [1]. In these systems, variables such as reaction time, speed or power usage have very low tolerance limits in order to satisfy a particular requirement, depending on their purpose [2,3].

One could imagine it takes decades and decades of destroyed test equipment and financial strain before reaching such a level of complexity in any system, but thanks to the rapid advancement of computer technology in the recent decades, it's possible to avoid the expensive equipment loss while still being able to learn something from the designed theoretical systems.

One of the main reasons the problem can be approached this way is the use of simulations, or better said, mathematical models of these systems. These models attempt to approximate the behavior of their real-life equivalents by using sets of differential equations to describe the input-output relationships of the components in the system they're simulating.

There already exists a number of software packages designed to create these simulations, the notable one being MATLAB and its feature rich package Simulink [4]. The subject of this paper isn't MATLAB and its inner workings, but rather the advantages of designing custom mathematical models and simulations using Python (or any other programming language) with the example of the design of separately excited DC motor model.

The aforementioned software packages have well optimized models and procedures that enable the user to create fast performing simulations with accurate results. The issue is, these software packages are often clunky, take tens of gigabytes of space on the drive and have license systems that prevent hobbyists from using the software altogether, unless they're ready to pay a significant sum of money for a piece of software they likely aren't going to be making any profit from.

Any upgrades to these software packages usually come bundled in new versions. The new software versions also tend to have obsolete and removed functions, replaced by better performing ones. This is great from a development standpoint, but often means no backwards compatibility so it requires users to keep all versions of software relevant to the project installed on their computer, causing storage space problems.

2. Separately excited DC motor mathematical model design

If behavior of a system can be described using a set of differential equations, it can be modeled in any programming language that can perform numerical integration and derivation (or in worst case, these methods can be developed). In the case of Python, the *numpy* and *scipy* packages have all the required tools one would need to design a mathematical model of such a system. The *numpy* module is especially important because it's written in C which trades Python's dynamic typing in favor of a significant performance boost, especially when working with multiple large datasets.

DC motor converts electrical into mechanical energy, therefore it can conceptually be split into a mechanical and electrical subsystem [5]. Both subsystems are defined with a set of differential equations that describe the relationship between different variables inside them, such as armature current, motor torque, load torque etc.

The mechanical subsystem defines the relationship between speed ω, motor torque T_{em} and load torque as:

$$\frac{d\omega}{dt} = \frac{1}{J_{tot}}(T_{em} - T_L), \qquad (1)$$

It should be noted, in this model T_L represents both load torque and the torque caused due to the bearing friction. J_{tot} represents rotor and load inertia. Differential equation of the electrical subsystem shown in (2) defines the relationship between armature current i_a, armature voltage U_a and electromotive force e caused by the motor rotation. L_a and R_a represent armature inductance and resistance. This equation can be defined as:

$$\frac{di_a}{dt} = \frac{1}{L_a}(u_a - i_a R_a - e), \qquad (2)$$

The operation of these two subsystems is mutually dependent considering motor torque is a function of armature current and electromotive force is a function of motor's rotational speed [5]. The relationship between motor torque and armature current can be seen in (3) and

the relationship between electromotive force and rotational speed is shown in (4). These equations can be written as:

$$T_{em} = k_m \Psi i_a = K_m i_a \quad (3)$$

and:

$$e = k_e \Psi \omega = K_e \omega. \quad (4)$$

For both expressions, magnetic flux linkage Ψ is assumed constant. Graphical representation in a form of a block diagram shown in figure 1 helps visualize the system and the relationship between its subsystems. Inputs into the system are armature voltage and load torque and the output is motor's rotational speed, but several other variables can be monitored.

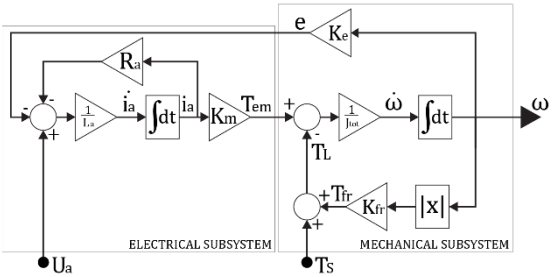


Figure 1. *Block diagram of a separately excited DC motor*

This mathematical model can be designed in Simulink, but requires a MATLAB installation. In some applications, such as when implementing optimization algorithms [5] or when requiring optimized single purpose simulations, it could be more convenient to design the model by using a programming language where all required variables are stored in large arrays which can further be used for different purposes.

In Figure 2, armature current and rotational speed are plotted for fixed armature voltage. Just like in MATLAB, there is a tradeoff between performance and accuracy. Considering all of the variables are discrete data sets, step size and simulation time determine the number of points in time where a value for each variable needs to be calculated.

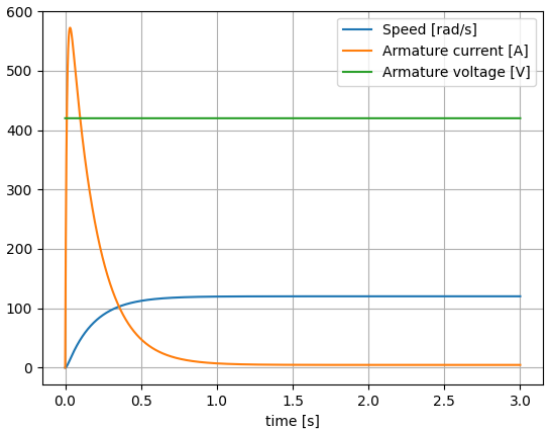


Figure 2. *Plot of DC motor variables in time*

3. Conclusion

Given the complexity of the problem, in most cases it is more productive and efficient to design mathematical models of prototypes and theoretical systems in already developed general-purpose simulation software packages. Given their general-purpose nature, in most cases these packages are designed with accuracy and ease of use in mind, not low execution time.

Optimization algorithms greatly benefit from reduced execution time achieved by removing all unnecessary parts of the model and in some cases isolating certain parts of the code which is especially easy if the model of the system already is in a form of code. The other notable feature of this workflow is absence of restraints and limits due to the working environment. This means it's possible to implement custom GUI to highlight important variables, different ODE solvers and any data analysis procedure not already implemented in the common simulation software.

References

[1] Baressi Šegota, Sandi, et al. "Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms." International Journal of Advanced Robotic Systems 17.2 (2020): 1729881420908076.

[2] Nouman, Khan, Zaman Asim, and Khan Qasim. "Comprehensive Study on Performance of PID Controller and its Applications." 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE, 2018.

[3] Sun, Xiaodong, et al. "State feedback control for a PM hub motor based on gray wolf optimization algorithm." IEEE Transactions on Power Electronics 35.1 (2019): 1136-1146.

[4] Le Roux, P. F., and M. K. Ngwenyama. "Static and Dynamic Simulation of an Induction Motor Using Matlab/Simulink." Energies 15.10 (2022): 3564.

[5] Mavrinac Marko, Zlatan Car and Mario Šercer. "Genetic Algorithm-Based Parametrization of a PI Controller for DC Motor Control." Tehnički glasnik 16.1 (2022): 16-22.

ISBN: 978-953-8246-26-5

Visualization of trilateration localization

Dorian Mohorić [1*]

[1] Faculty of Engineering – University of Rijeka, Vukovarska 58, 5100 Rijeka, Croatia, dmohoric@riteh.hr

Abstract: Trilateration is an often-used method of mobile robot localization. It uses three distances from known points to determine the location of a robot. In this paper it is described how to visualize it in Python programming language using common libraries like matplotlib. Sensor error is also considered and displayed. Resulting plot accurately portrays how trilateration is used to find the absolute location of a mobile robot.
Keywords: mobile robots, Python, trilateration, triangulation

1. Introduction

Mobile robots are increasingly used in manufacturing and service industry. The main advantage is that they can operate autonomously. For this to be achieved, mobile robots must include a localization system in order to assess the robot position as accurately as possible. There are many different solutions which are used to solve the localization problem. They are classified in two general groups: relative and absolute localization methods [1].

In relative localization, internal measurements are used to determine the robot position and orientation from a known initial position. The two main methods in this field are inertial navigation [2,3], in which the measured data are accelerations of robot points and angular velocities; and odometry [4], in which the measured data are wheel rotation angles. Odometry is a widely used technique because of its low cost, high updating rate and short path accuracy [5]. However, its unbounded growth of time integration errors with the distance travelled by the robot is unavoidable and represents a significant inconvenience [6].

In absolute localization, the position is determined from its geometric relationship with external measurements of the robot workspace. Sensors such as laser scanner, camera, ultrasound etc. provide distance or angular measurements, relative to the robot. There are several methods of absolute localization. The first group of them apply when the measurements correspond to the same robot position (geometric methods). In such a case, the robot position can be straightforwardly obtained from the measured data and its geometric relationship with the robot position. This is the case of trilateration, which determines the robot position from distance measurements associated with a set of known points or landmarks [7]. Triangulation is used when the measured data are angles [8].

In this paper visualization of trilateration is done in Python programming language with matplotlib which is used for plotting. matplotlib is a portable 2D plotting and imaging package aimed primarily at visualization of scientific, engineering, and financial data. [9].

2. Methodology

As mentioned in the introduction, Python is used in this study. Three circles are defined by their center coordinates and radii. Mathematical formulas for intersection points of two circles [10] are used and an intersection function is defined. Inputs are coordinates of two circles and their

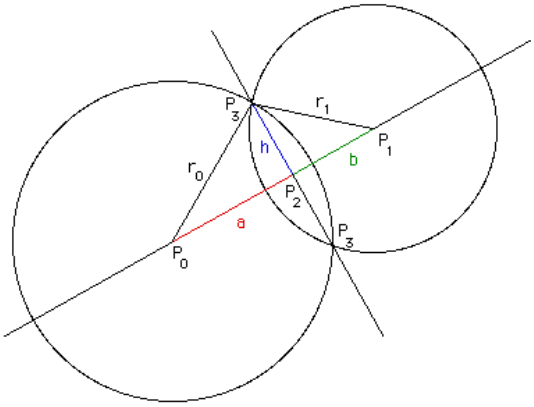


Figure 1. *Intersection points of two circles [10]*

radii, the function returns coordinates of 2 points of intersection. Figure 1. shows the notation used in the function. Distance d between the points "P_1" and "P_0" is calculated first in Eq. 1:

$$d = \sqrt{((x_1 - x_0)^2 + (y_1 - y_0)^2)}\ , \#(1)$$

Using the properties of right triangles, we can solve for length a (Eq. 2):

$$a = \frac{r_0^2 - r_1^2 + d^2}{2d}, \#(2)$$

where r_0 and r_1 are radii of circles. Length h is calculated using the Pythagorean theorem:

$$h^2 = r_0^2 - a^2, \#(3)$$

Next to last, coordinates of point "P_2" are calculated by following equations 4,5:

$$x_2 = x_0 + a \cdot \frac{x_1 - x_0}{d}\ , \#(4)$$

$$y_2 = y_0 + a \cdot \frac{y_1 - y_0}{d}, \#(5)$$

And finally, intersection coordinates of point "P_3" on either side of point "P_2" are calculated in similar way (Eq. 6,7,8,9):

$$x_3 = x_2 + h \cdot \frac{y_1 - y_0}{d}, \#(6)$$

$$y_3 = y_2 - h \cdot \frac{x_1 - x_0}{d}, \#(7)$$

$$x_4 = x_2 - h \cdot \frac{y_1 - y_0}{d}, \#(8)$$

$$y_4 = y_2 + h \cdot \frac{x_1 - x_0}{d}, \#(9)$$

The intersection function is then applied to first and second circle, and first and third circle in order to obtain two points of intersection per circle pair. Each point from one circle pair is compared to other pair. If the points match with some acceptable error, then the point where three circles intersect is found.

Visualization is done with matplotlib library. Sensor distance circle is drawn by using a cosine function on x axis, and sine function on y axis. This is repeated for sensor error tolerance circles by increasing and reducing the radius 5 percent. Area between these two circles is filled red. Circle centre is marked with a plus sign. This process is then repeated for two other circles. Finally, a blue dot is positioned on the intersection of these three circles.

3. Results and Discussion

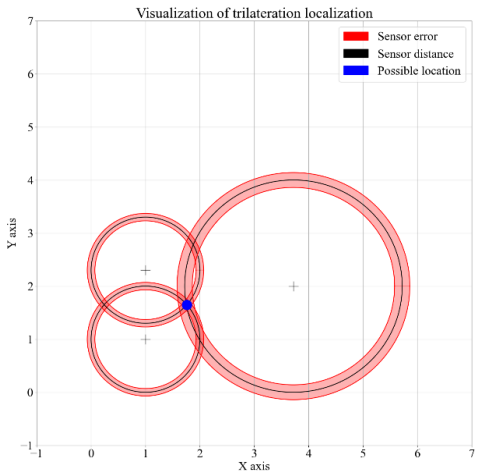


Figure 2. *Trilateration localization plot*

The resulting plot is given in Figure 2. Blue dot represents the location of the robot if it's assumed the sensors are perfectly accurate. Since that is not true, the red area that intersects is a less probable location of the robot.

4. Conclusion

Trilateration is a method of mobile robot localization. In this paper it is described how to visualize it in Python programming language with sensor error taken into account. Resulting plot truthfully portrays how trilateration is used to find the location of a mobile robot.

Acknowledgments

I would like to thank Sandi Baressi Šegota, mag. ing. comp. for the help and mentorship in the making of this paper.

Reference

[1] Borenstein, Johann et al. "Mobile robot positioning: Sensors and techniques." *J. Field Robotics* 14 (1997): 231-249. https://doi.org/10.1002/(sici)1097-4563(199704)14:4%3C231::aid-rob2%3E3.0.co;2-r

[2] Font, Josep Maria, and Joaquim A. Batlle. "Mobile robot localization. Revisiting the triangulation methods." IFAC Proceedings Volumes 39.15 (2006): 340-345. https://doi.org/10.3182/20060906-3-it-2910.00058

[3] Barshan, Billur, and Hugh F. Durrant-Whyte. "Inertial navigation systems for mobile robots." *IEEE transactions on robotics and automation* 11.3 (1995): 328-342. https://doi.org/10.1109/70.388775

[4] Wang, C. Ming. "Location estimation and uncertainty analysis for mobile robots." *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE, 1988. https://doi.org/10.1109/robot.1988.12229

[5] Font-Llagunes, Josep M., and Joaquim A. Batlle. "Consistent triangulation for mobile robot localization using discontinuous angular measurements." Robotics and Autonomous Systems 57.9 (2009): 931-942. https://doi.org/10.1016/j.robot.2009.06.001

[6] Kelly, Alonzo. "Linearized Error Propagation in Odometry." The International Journal of Robotics Research 23.2 (2004): 179–218. https://doi.org/10.1177/0278364904041326

[7] Thomas, Federico, and Lluís Ros. "Revisiting trilateration for robot localization." IEEE Transactions on robotics 21.1 (2005): 93-101. https://doi.org/10.1109/tro.2004.833793

[8] Betke, Margrit, and Leonid Gurvits. "Mobile robot localization using landmarks." IEEE transactions on robotics and automation 13.2 (1997): 251-263. https://doi.org/10.1109/70.563647

[9] Barrett, Paul, et al. "matplotlib--A Portable Python Plotting Package." Astronomical data analysis software and systems XIV. Vol. 347. 2005. https://doi.org/10.1007/978-1-4842-7410-1_16

[10] Bourke, Paul. "Circles and spheres." Paul Bourke (1992).

ISBN: 978-953-8246-26-5

Design and analysis overview of dual arm robots

Danijel Lončar [1*], Denis Mijolović [2,3]

[1] Faculty of Engineering University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, email: dloncar@riteh.hr.
[2] University of Rijeka, Faculty of Engineering, Vukovarska 58, 51000 Rijeka, Croatia, dmijolovic@riteh.hr.
[3] Uppsala University, Department of Civil and Industrial Engineering, Ångströmlaboratoriet, Lägerhyddsvägen 1, 752 37 Uppsala, Sweden, denis.mijolovic.0720@student.uu.se.

Abstract: This paper represents problem solving methods of manipulating and control of dual arm robot in real and predictable situations using simplified approach in mathematical calculations, considering eventual and very possible faults during robot's lifespan. These approaches are not reserved for dual-arm robot only but can be used on multi-arm robot as well. Accounting for present day practices and its impact on robot longevity, latest paradigms are considered – particularly, ones assessing the efficiency and plausibility of evolutionary robotics and various algorithms compared to conventional methods like Jacobian method and null space method. While Jacobian method and null space methods are methodologically credible, various dependencies and rising complexity are implied as potentially negative impact from economic and flexibility-of-use perspectives. Findings on the application of evolutionary robotics algorithms have shown to come with successful and demonstrative results for dual arm robots with higher degrees of freedom. Additionally, this may imply the positive outcomes in terms of energy saving, parts longevity, velocity of robot development, and adaptability to different scenarios.

Keywords: Dual arm robot, genetic algorithm, multi arm robot, relative jacobian method, robot trajectory planning

1. Introduction

Dual-arm robot manipulators and multiple movement freedom requires very complex mathematical management so each arm could be able to execute given task in a way that avoids collission with other arm [1]. The simplest example of using a dual-arm robot is when each hand performs the same task and does not depend on the other hand like stacking goods on, or from a conveyor belt, where a dual-arm robot typically behaves like two different one-arm robots. The situation is further complicated if two arms have to perform a process together. Two arms can perform symmetrical or mirror operations such as lifting an object and moving it, the same movements but phase or time shifted such as pulling a rope, or totally different operations, where one hand depends on the other, such as removing food rests from the plate. Additionally, collaborative systems dimensioned for complex tasks like multi-robot collaboration in autonomous construction tasks [2] or aerial or mobile manipulation [3] call for new holistic approaches that would yield better optimisation flexibility and method ubiquity.

2. Methodology

With practical applications of dual arm arobots are considered, the robot design and parameterisation get even more complex with each arm having more freedom of movement, i.e., such robot has more joints driven by a separate drive, and each segment of the arm has its own mass. When moving the arm segment from a stationary position, that part of the arm accelerates or slows down, so it is necessary to take into account the inertia of individual segments. These mass inertias are variable because it is not the same whether the segment moves closer to the axis of rotation (folded arm) or far from the same axis of rotation (stretched out arm). Additionally, things get complicated if such arm picks up or drops an object at the end where the sum of inertia for a given hand suddenly changes.

This problem can be solved by moving the "empty" arm at a higher speed than when it is under load for that difference in load mass when calculating inertia. The image of the system changes significantly if the robot constantly changes the weight of the load. One such example is picking and separation of segments of different shapes on the recycling belt. In the end, the problem is not that the robotic arm will change the inertia, but the problem is whether will it stop in a specific position without overshooting and takes additional position corrections after stopping because that requires unnecessary movements which significantly prolongs task execution time. In cases of fast processes, as with the already mentioned conveyor belt, it could happen that the robot is not able to perform any given task [4]. The risk of this issue occurring increases not only with the number of units or robotic arms used, but with additional system-wide parameters as presented by Kim et. al (2013) on an example of quadrotor with two DOF robotic arm – where the robotic manipulator, as well as its dynamics, are highly couopled with the areal vehicle.

Once other such issues like precision, performance reliability, system robustness, or energy efficiency are accounted for, one may come with a conclusion robotic system is highly contextual and grows tremendeously in complexity as collaborative tasks are included. Whilst previous examples have no direct reprecutions on human helth, once such endeavour is envisioned by NASA through their plan to lead human Martian exploration [2]. In order to facilitate the landing site, life-vital infrastructure, transporation systems, and various other moduels; NASA is incredibly reliant on the uzse of robotic technologies that ought to perform such tasks with nearly zero-margin errors [2].

With previous examples in mind, this research seeks to investigate and analyse existing research on various applications of design and analysis of dual arm robots with an aim to come with an overview of novel technologies

and methodologies within the scope of evolutionary robotics that could enhance specific processes.

3. Results and Discussion

From the presented situations, it is obvious that a very complex algorithm is needed to process the data for a given robot in order for it to be able to perform any given task. Therefore, a new way of controlling the robot is proposed by implementation of impedances with inertia, damping and stiffness on the robot using the relative Jacobian technique because the method greatly simplifies programming by considering double-arm robot as a one-arm manipulator [5]. Movements are thus defined in such a way that a end motion is defined as the relative motion of two end motions by given paths.

Any machine that has more moving parts is also more susceptible to brake down more easily. The most critical points of failure are the joints of individual segments of the robot's arm. Although the robot is seemingly operative, the increased clearances in the joints increase the inaccuracy of the robot. The closer the faulty joints are to the mount of the manipulator, the greater the error at the end of the arm. As it is not possible to predict when and where the shift in accuracy will occur, two solutions to this problem have been proposed. One of them is by limiting manipulativeness to the optimal efficiency of both robot arms using Jacobian Zero Space. The second way suggests allowing compensation of losses over the final results of the movement using zero space saturation.

Other methods of controlling the robot is such similar scenarios are evaluated through the prism of utilisation of genetic algorithm (GA) in the case of simple robotic manipulators with two DOF, to which was found such method could be plausable for the usecase of optimisation of robot manipulator paths in regard to joint torque [6]. Specifically, the application of algorithms based on evolutionary robotics principles allow for significant energy savings through optimisation of actuator torque dependant on the joint trajectory – further implying not only energy savings itself, but overall prolongation of the work life of the robotic manipulator and consequential maintenance or acquisition cost benefits for businessies using them. Related work on the use of GA are further discussed by Baressi Šegota et al. (2020), as well as comparative analysis of results obtained through the application of the beforementioned altorithm and other commonly used algorithms, such as simulated annealing (SA) and differential evolution (DE), on an example of six DOF robotic manipulator. Results obtained through the analyses showed successful optimisation of six DOF robotic manipulators using evolutionary algorithms – with best results provided by GA with random recombination [7].

4. Conclusion

While there are various complex methods of fault control, such as kinematic joint fault tolerant control is based on Jacobian method and using saturation null space method that changes the current configuration of both manipulators without affecting the desired relative and effectors distance [8], implementation of such methods for robotic dual arm manipulators with higher DOF may exceed overall calculation complexity – which may inconclusively lead to procedural errors. Evolutionary robotics and genetic algorithms have shown great results in specific scenrios and may be deemed as more cost-effective and time efficient. This is backed by findings from Baressi Šegota et al. (2020) and Stroupe et al. (2005).

Reference

[1] T. J. Tarn, A. K. Bejczy and X. Yun, “Design of dynamic control of two cooperating robot arms: closed chain formulation,” IEEE International Conference on Robotics and Automation, vol. 1, pp. 7-13, 1987.
[2] A. H. T. Stroupe, A. Okon, H. Aghazarian and M. Robinson, “Behavior-based multi-robot collaboration for autonomous construction tasks,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, 2005.
[3] S. Kim, S. Choi and H. J. Kim, “Aerial Manipulation Using a Quadrotor with a Two DOF Robotic Arm,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, 2013.
[4] D. I. Park, C. Park, H. Do, T. Choi and J. Kyung, “Design snd analysis of dual arm robot using dynamic simulation,” in International Cobference on Ubiquitous Robots and Ambient Intelligence, Korea, 2013.
[5] J. Lee, P. H. Chang and R. S. Jamisola, “Relative Impedance Control for Dual-Arm Robots Performing Asymmetric Bimanual Tasks,” IEEE Transactions on Industrial Electronics, vol. 45, no. 7, pp. 3786-3796, 2014.
[6] D. P. Garg and M. Kumar, “Optimization techniques applied to multiple manipulators for path planning and torque minimization,” Engineering Applications of Artificial Intelligence, vol. 15, no. 3-4, pp. 241-252, 2002.
[7] S. Baressi Šegota, N. Anđelić, I. Lorencin, M. Saga and Z. Car, “Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms,” International Journal of Advanced Robotic Systems, vol. 17, no. 2, 2020.
[8] A. Freddi, S. Longhi, A. Monteri`u and D. Ortenzi, “A Kinematic Joint Fault Tolerant Control Based on Relative Jacobian Method for Dual Arm Manipulation Systems,” in Conference on Control and Fault-Tolerant Systems, Spain, 2013.

ISBN: 978-953-8246-26-5

Visualization of visibility graph for path planning

Jura Modrić ,Ivan Zoričić

Faculty of Engineering – University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia

Abstract:.We discussed the topic of map planning in robotics and in general. Path planning algorithms and graph visualisation were used. Used approach and algorithms are explained in this article. Complexity and number of iterations with respect to number of nodes were discussed.
Keywords: Visibility graphs, Dijkstra, Collision detection

1. General

We are moving in a direction where robots nowadays are starting to be more and more parts of our society in every way. They are helping us in many ways, even in the ways we're maybe not aware of. Nevertheless, one of the tasks, what we humans expect from them and generally is one of the abilities that every entity has is movement. We would like to make robot move in coordination with the given task. Task it can complete my moving, considering today we have robots which are able to move in many different environments. If we can move the robot, or any path planning when thinking in general terms. GPS systems can benefit from this type of planning for that matter

2. introduction

When mobile robots receive request to move to a specific location, robots must first find optimal path to reach that location. There are multiple ways to calculate their path, but one that will be described in this article is Visibility graph.
The visibility graph is a fundamental combinatorial structure in computational geometry, that is used in computing shortest paths among polygonal obstacles in the plane and in decomposing two-dimenstional shapes into clusters.
Visibility graph is often used in robotics, and there are many examples of it being used. One of early implementations of visibility graph happened in 1969 when Nils Nillson made Shakey the Robot, which was first general-purpose mobile robot able to reason with its own actions, and it used visibility graph to map its environment.

3. Methodology

Visibility graph is often used to find shortest path among set of polygonal obstacles in the plane. This problem can be divided into two smaller problems. First is constructing visibility graph and second is applying algorithm that finds shortest path among all possible paths, then use those nodes for finding the shortest path. We ensure movement of robot outside restricted area by following only visible paths between vertices of the obstacle.
First thing is to import a map. We used matrix format, where zeros represent moveable space, ones represent obstacle, and twos are the vertices of the obstacle, or polygon that represents the shape of the obstacle. Border of the map is confined by ones, where robot shouldn't move. We specified the starting and ending point by values 3 and 4 respectively. There are 19 nodes total, including starting and ending node. Concerning code complexity with respect to the number of fields, we expect O(n^2) iterations of the algorithm for iterating through the entire matrix. The matrix is in form of nested arrays where each inner array represents row of matrix.[1]

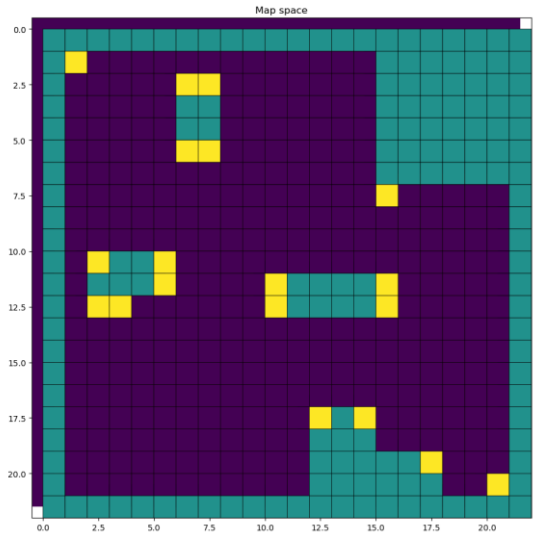


Figure 1. *Map space*

Our main algorithm, after plotting the map, iterates trough the map and finds all the vertices. This takes O(n^2) iterations to complete. After forming the list of x and y coordinates of the vertices, we proceed to analyse every pair of vertices. This requires again iterating O(n^2) times. In inner loop, we calculate distance between vertices. We can see if distance is equal to zero, which means we're referencing to same spot, and can continue to iterate. Then we check for visibility between two vertices. It is done by collision checking function described later. If the function validates no obstacle between points, we proceed to plot the path between vertices and append it to the node connection graph[1]

We defined functions to make the code more readable and easier to reuse. They are Dijkstra and A* search algorithms and collision checking for pair of vertices.
Dijkstra function accepts list of adjacent nodes. It is formatted as dictionary. Dictionary consists of key value pair, so that the key is the observed vertex, and its pair is a list of tuples which contains connected second vertex and distance to it from current point. In this way, every point has connections and distances to other points in map. Next parameters of Dijkstra are start and end node.

We first mark all nodes unvisited. At the first vertex, we assign zero values as starting and infinity as value for all other nodes. Then we visit adjacent nodes and update their distance to parent node. We go to the next node and update its distance. If the adjacent node has shorter path, we update it. This is repeated until target node is reached. After calculating the path, the optimal trajectory is displayed and plotted on the map [2][3].

Collision checking is O(n), where n is number of points we take along the line between two nodes. The algorithm is as follows.: Function takes the x and y coordinates of the first and second point. Then, we populate an array of numbers between x values of the two points, with large number of points between, as the linear complexity of algorithm doesn't greatly depend on number of points considering quadratic complexity of earlier stages of algorithm.
The equation of a two points line is used to construct x and y pairs for every x between vertices. We can then check, for every point, if we have an obstacle, or 1 in our mapped space for that specific spot on the map. One problem is, when using this approach, points that are on the same x coordinate result in division by zero in formula. To overcome this, we first check if we're dealing with that kind of vertices, and if yes, we just have to check all the points on the line between considering y coordinate as the variable. To clarify, we normally use x as variable and y as function of x, but in case of points with same x coordinates, we use y as a variable of function x. The function returns False in case of hitting the obstacle, but returns True when no obstacle was found.

Another method was tested, but not implemented. We tried using the polar coordinates sweep, where a ray would be rotated around the vertex, and for every position of that ray the algorithm checks if the ray hits any obstacle. In the end, we used first solution, gave good results.

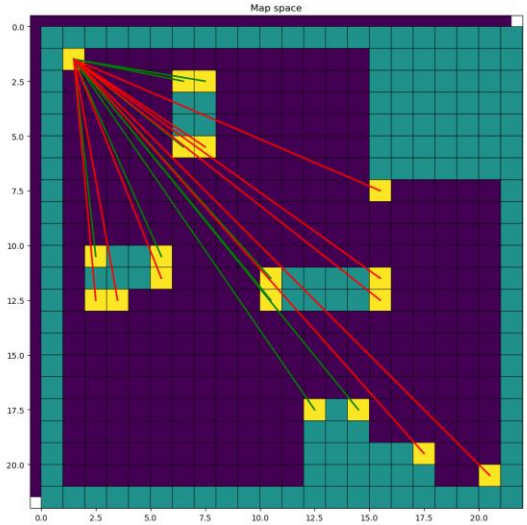


Figure 2. *First vertex with legal and illegal connections*

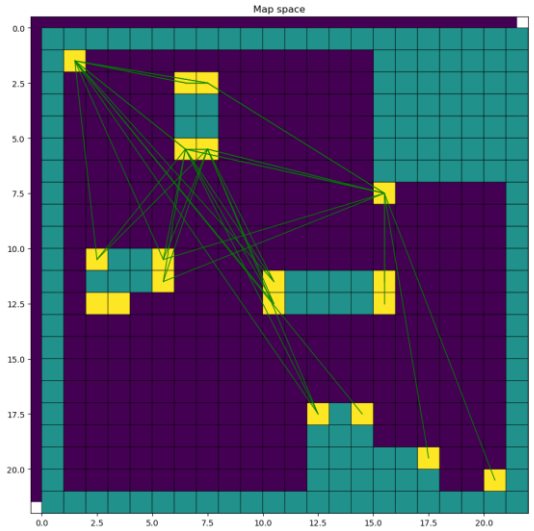


Figure 3. *First six vertices processed*

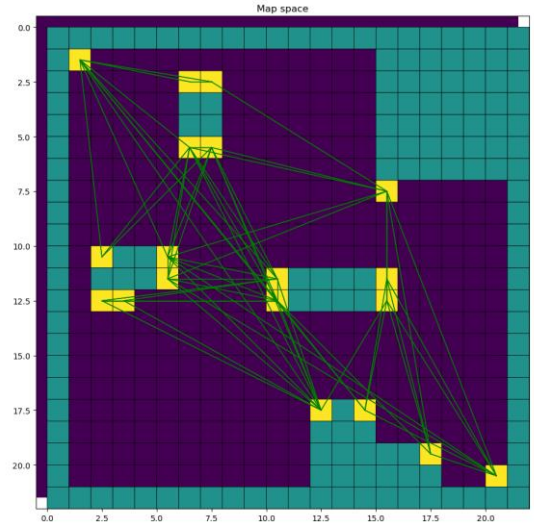


Figure 4. *All vertices processed*

It is worth mentioning that this approach works with theoretical models of robot. In real use, we can see the problem when robot has its own dimensions, and collision would have to take in account those parameters. We have neglected those as it was not in the scope of this paper, but main principle holds.

3. Results and Discussion

After executing the algorithm, full map is displayed with all the nodes that are in visible path of each other. In addition, Dijkstra algorithm provides us with the shortest way of getting from point A to point B.

The fastest path between A and B is marked with red line. Calculated path is very close to the shape of ideal, fastest path that would be straight diagonally from start to end, when obstacles would be ignored. If we assume different shape of the map, we can vary number of vertices, edges. One can easily see the applications of this type of path planning. Our testing map can be overlayed with some ground shots of various terrains, scaled.

You can imagine this map being a living room where objects are represented as obstacles, which is something that we can se implemented in robots for transporting material, automated delivery robots (warehouses, etc.). With slight altering of the code or parameters, we can achieve different behaviour, we can use the map of nodes to make the robot visit each and every one in different kinds of patterns.

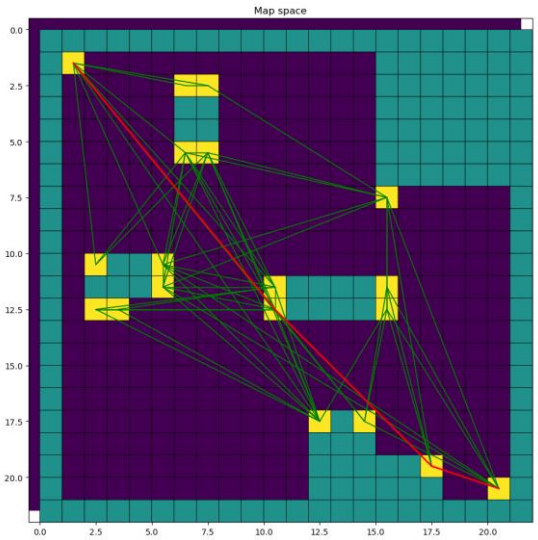


Figure 5. *Final plotted path (red) from the start vertex to end vertex*

As long as the map has collision, nodes to provide to Dijkstra algorithm, we can achieve motion with robot. If we scale this map to larger scenarios, like factory, or even city block, one can see this algorithm can be even used in small vehicles like scooters, bikes, drones, RC aircrafts, toys. It can be used in medical purposes, for delivering medical supplies, etc. It can be used in larger vehicles, like modern electric automobiles, taxi services, mail delivery, magazine delivery. It can be used in every vehicle, land, sea or air. Code can be updated with all sorts of new features, depending on the specific targeted area of implementation.

This planning can be optimised even further by modern AI methods. One can train AI model to achieve movement between vertices in a manner it is trained to. This opens very broad field of applications, including naval.

The execution times for small number of vertices is in terms of seconds. This includes the whole program, including plotting. While map traversal takes most of the computing time, once it is completed and all the vertices are mapped, we can do calculations and path planning on those mapped vertices in any way we like.
This is very rudimental but robust way of moving through space, and finding fastest path.

4. Conclusion

Topic of path planning is quite interesting, with a lot of potential applications. It can help us in everyday life, improve the quality of life and make repeating tasks more easy to manage. This article expanded a way of thinking about hardware and software for generating maps as input to our program. If we include other evolutional robotics topics, like neural networks, convolutional neural networks, reinforced learning, or even include this on swarm of robots, there is a whole new virtual space of possibilities for usage.

Acknowledgments

We thank professor Zlatan Car, assist. Sandi Baressi Šegota, for their mentorship during the work on this article.

Reference

[1] Car, Zlatan Lectures in Evolution robotics
[2] Hauser, Kris. Motion Planning in Simple Geometric Spaces
[3] Wikipedia, Dijkstra's Algorithm

ISBN: 978-953-8246-26-5

Approximation of density functional theory (DFT) ground state energies from electron densities using a multilayer perceptron

Filip Babić [1*], Nikola Anđelić [2]

[1 2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; fbabic@riteh.hr, nandelic@riteh.hr

Abstract: Density functional theory *ab initio* computations allow researchers to calculate and predict properties of many-body particle systems. Unfortunately, these calculations are computationally intensive, and research has been done on using machine learning predictive methods to bridge the gap between usefulness and execution speed. In this paper research has been done on using a multilayer perceptron to approximate DFT energies from a dataset of known DFT densities. In the methods section the dataset was described while the source from which it was generated was referenced. Following the description, the dataset was analysed and prepared for use in MLP regression by extracting highest correlating features. The MLP hyperparameter search grid was set and scoring measures were described. In the end the model with the best results was extracted while the results were listed and visualised followed by a discussion. Finally, a conclusion was made that the research was successful as a ground-state energy mapping from DFT densities is observed and the hypothesis from the introduction was confirmed.

Keywords: Artificial Intelligence, Density Functional Theory, DFT, Ground State Energy, Machine Learning, Multilayer Perceptron

1. Introduction

Interest in using machine learning (ML) for scientific research has grown exponentially in the last two decades [1][2] as more computational resources became readily available and many free and easy to use frameworks for developing and applying ML programs have been constructed with many meticulously curated databases to experiment with [3][4]. ML is a way to create input and output mappings only from acquired (sampled) data where theoretically formulating functions would be difficult [5][6]. This is the reason why ML has spread to sciences with multivariate and nonlinear data like in medicine [7][8] or in places that have complicated interactions or quantum physics problems like in material sciences [9] and computational chemistry [10]. Leveraging ML in computational chemistry has lately proved very successful [11]. In this article, the focus will be on density functional theory (DFT) which is a computational quantum chemistry model used to extract the electrical properties of many-body systems of particles [12].

As is shown in the work of Burke *et.al.* [13][14] if there is a known electron density then other electrical properties like the electron potentials and ground-state energies can be calculated or approximated. The aforementioned publications calculated DFT energies using ML to approximate exchange-correlation functionals with kernel ridge regression. The goal of this research was to see if its possible to adequately approximate water molecule energy via regression on a density dataset using a multilayer perceptron by bypassing the need for approximating the unknown exchange-correlation functional by using neural networks to create a direct mapping between the density and ground-state energy.

2. Methodology

The dataset chosen for the research is the water densities dataset from public and free repository [15] with many DFT-based datasets and is used as-is. The densities dataset was generated by Bogojeski *et.al.* and for details on its generation, we refer to their publication [14].

The water dataset contains 102 molecular geometries formatted in the Bohr model with their coupled-cluster (CC) energies, DFT energies, and DFT densities based on the PBE functional. Each geometry is given as a 3x3 matrix and has a CC and DFT energy given in kcal/mol and a DFT density given in 125000 Fourier basis coefficients per geometry. Dataset is visualised in Figure 1 where the first graph shows the density coefficients with densities overlapping for each geometry and the other two are DFT and CC energies with markers for each geometry.

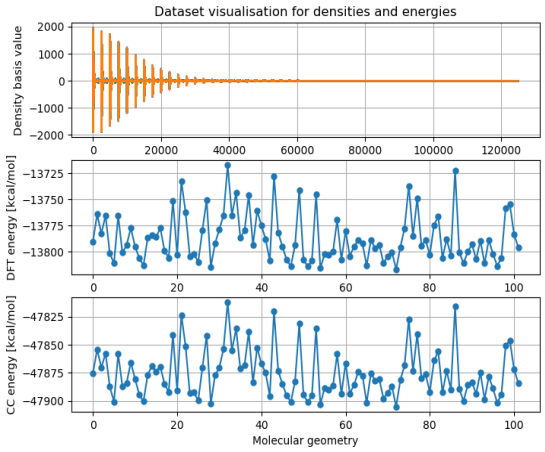


Figure 9 *Visualising densities and energies from the dataset*

The dataset used has a great number of input features (125000) but has a relatively small number of samples (102) for each feature. Because of those characteristics training of the ML model can be slow and cause input noise with features of low importance [16]. These effects require analysis of min and max values, their mean, standard deviation, and Pearson correlation between features to try and extract the most useful features. Due to the size of the density dataset, a correlation matrix isn't viable and has been performed in chunks per 5000 features due to computational restraints. The correlation of features against DFT energy was observed and if the values were between -1 and -0.6 or 0.6 and 1 those features were kept. The standard deviation of the densities and the correlation

values of the densities against the DFT energy are illustrated in Figure 2. The CC energy has an almost perfect correlation which is expected. After extraction of the highest correlating values, the size of the density dataset ended up having 7012 features for 102 samples. This final dataset is randomly split between a train (70%) and test (30%) set.

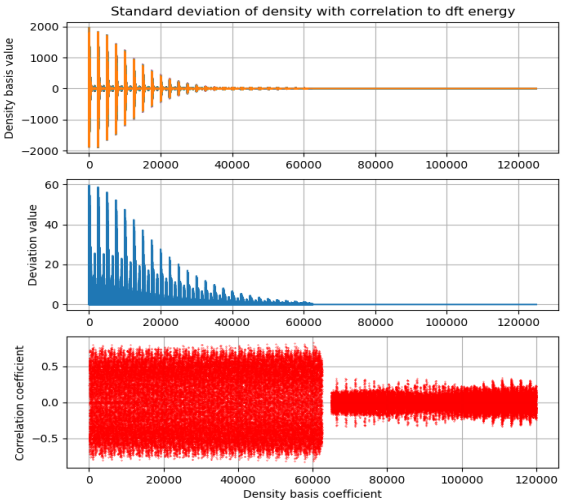


Figure 10 *Analysis of the DFT densities. First graph shows the density coefficients, second graph is the standard deviation between the densities per geometry, third graph shows correlation value of the densities against the DFT energy*

Approximation of molecular energies is done using regression analysis as a statistical model by estimating the relationship between a dependant variable (the molecular energy) and the independent variables (density features). In this case, the computational model of a machine learning framework for regression such as a multilayer perceptron (MLP) was used.

A multilayer perceptron is a type of feedforward artificial neural network (ANN) that is characterized as having three or more layers of interconnected neurons. The outermost layers are the input layer and output layer with one or more hidden layers in the middle where each neuron in the following layer is connected to every neuron in the layer before it [6][7]. The input layer has as many neurons as there are input features (7012 neurons) and the output layer consists of one neuron as the prediction output (molecular energy). An MLP works by having neurons in one layer take in the activated weighted sum of values from the neurons in the layer before it then performing their own weight multiplication and passing that value through an activation function. An activation function maps the input to the output either directly through an identity function, within some limit through logistic or tanh function or filtered with a ReLU function. Weights are coefficients that scale the summated values from neurons in previous layers and affect the outcome of activation. The neural network trains by adjusting these weights through backpropagation depending on the predicted output error [6][7].

The selection of MLP as a method is due to the easy implementation and adequate results on both linear and nonlinear types of data. Implementation of the MLP ANN has been achieved in the Python programming language using the scikit-learn library [17]. The scikit-learn library contains tools and functions that facilitate building an easy-to-use ML framework. The training was performed on a personal computer containing an AMD Ryzen 5 2600 CPU with 6 cores (12 logical) and 8 GB of RAM running the Windows 10 operating system.

To build and train an MLP model determining hyperparameters is an important step. Hyperparameters are settings to ANN models that will affect the quality and speed of model training. Determining hyperparameters in a supervised learning situation is done by setting a baseline value within a range of values and training the model. After training hyperparameters are adjusted either manually or randomly within the predetermined range then repeating the model training followed by evaluation of results. Hyperparameters used in training are listed in Table 1 with defined upper and lower boundaries for numerical values and possible choices for model settings.

Table 1. *Model hyperparameters used in the parameter search.*

Hyperparameter	lower boundary	upper boundary
Hidden layer sizes	0	10
Number of neurons	10	150
Solver	LBFGS, Adam	
Activation	Identity, Logistic, Tanh, ReLU	
Learning rate	Constant, Adaptive, Invscaling	
Initial learning rate	0.0000001	0.01
L2 regularisation parameter	0.000001	0.01

Model result evaluation is done using standard metrics of coefficient of determination (R^2) on both train and test sets and evaluating mean absolute error (MAE) and root mean square error (RMSE) on predicted output values.

The R^2 metric measures the percentage of correct predictions of the model and is defined by the formula (1) where $y^{\wedge}$ is the predicted value of y, and y^{-} is the mean value of y. The results are in the range [0,1], where 0.0 means a failed prediction result, and 1.0 is the best possible result.

$$R^2 = 1 - \frac{\sum_{i=0}^{m}(y_i - \hat{y})^2}{\sum_{i=0}^{m}(y_i - \ddot{y})^2}, \qquad (1)$$

MAE represents the average difference between the predicted and original values averaged by the absolute difference and is defined by the formula (2). Results can be any positive number where the closer it is to 0 the prediction results are closer to the original.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|, \qquad (2)$$

RMSE represents the square root of the difference between the predicted and original values by the squared average difference and is defined by the formula (3). Results are any positive number and the closer it is to 0 the prediction results are closer to the original

$$MAE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}, \qquad (3)$$

3. Results and Discussion

After hyperparameter search and test evaluations the MLP showed a very high-quality regression where best models regularly scored R^2 of more than 0.98 and MAE and RMSE values of 1.5 or lower. The best model hyperparameter values and scores are listed in Table 2. During the parameter search it was observed that having more than 3 layers had a negligible result on scoring which was also observed for values of more than 50 neurons per layer so these parameters were kept at those values to conserve computing power and lower training time.

Of the two solvers LBFGS showed lower training times and better scores which was expected as it should converge faster on smaller datasets as per scikit-learn API [3][17]. Adam solver had similar R^2 scores but worse MAE and RMSE, also it showed extremely high sensitivity to initial learning rate parameter and tended to start to oscillate with instability in the loss function so training and evaluation was difficult to perform.

The dataset has large differences in minimum and maximum values with positive and negative numbers and using standard scalers to transform the data caused much lower training scores so the dataset was left as is. Due to this logistic and tanh functions performed very poorly. Identity activation function showed best results. ReLU also performed well but had generally worse results than the identity activation function.

Results are also visualised in Figure 3 which shows predicted results in blue plotted over original values with markers in red where first two subplots contain zoomed in zones from the graph in the lower portion. The shaded area is the visualised tolerance of $y_{predicted} \pm$ RMSE which can be seen once zoomed in.

Table 2. *Hyperparameters and score metrics for best MLP model.*

hyperparameters and results	values
Hidden layer sizes and neurons	(50,50,20)
Solver	LBFGS
Activation function	Identity
Learning rate	Adaptive
Initial learning rate	0.00001
L2 regularisation parameter	0.0001
R^2 TRAIN	0.999694
R^2 TEST	0.9999
MAE	0.26274
RMSE	0.330126
Training time	~1.5 to 3 minutes

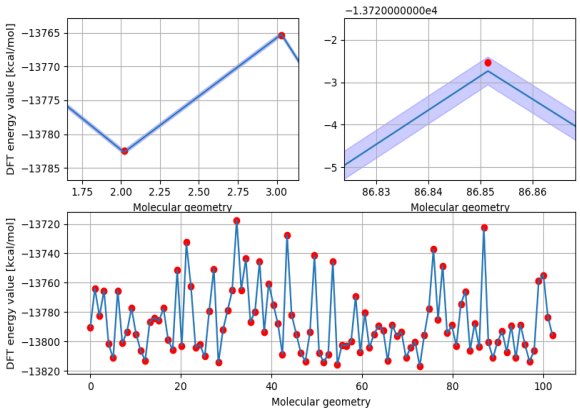


Figure 11 *Results of ground state energy predictions with predictions in blue and original values in red*

4. Conclusion

In this paper the research shows it is possible to train an MLP model to approximate ground state energies using known DFT densities of water molecule conformers in a relatively short amount of time on a personal computer with limited computational resources. The model has a limited use case since it can only offer ground state energies from a limited dataset, but it shows promise as a proof of concept that MLP could be used to bypass complicated functionals given a well-prepared dataset. Research also shows that good results can be obtained from a small sample size due to the complexity of the density features that are crucial in DFT mapping.

Further research is planned in expanding the model to other datasets with more complex molecules with bigger sample sizes to see how scalable the models are.

Acknowledgments

"We thank assistant dr.sc. Nikola Anđelić for his mentorship during the work on this article".

Reference

[1] Afiouni, Rania. "Organizational learning in the rise of machine learning." (2019).

[2] Dhanalaxmi, B. "Machine learning and its emergence in the modern world and its contribution to artificial intelligence." 2020 International Conference for Emerging Technology (INCET). IEEE, 2020.

[3] Buitinck, Lars, et al. "API design for machine learning software: experiences from the scikit-learn project." arXiv preprint arXiv:1309.0238 (2013).
[4] Asuncion, Arthur, and David Newman. "UCI machine learning repository." (2007).
[5] Janet, Jon Paul, and Heather J. Kulik. Machine Learning in Chemistry. American Chemical Society, 2020.
[6] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
[7] Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. Artificial Intelligence in Medicine, 102, 101746.
[8] Car, Z., Baressi Šegota, S., Anđelić, N., Lorencin, I., & Mrzljak, V. (2020). Modeling the spread of COVID-19 infection using a multilayer perceptron. Computational and mathematical methods in medicine, 2020.
[9] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Multilayer perceptron approach to condition-based maintenance of marine CODLAG propulsion system components. Pomorstvo, 33(2), 181-190.
[10] Prezhdo, O. V. (2020). Advancing physical chemistry with machine learning. The Journal of Physical Chemistry Letters, 11(22), 9656-9658.
[11] Snyder, J. C., Rupp, M., Hansen, K., Müller, K. R., & Burke, K. (2012). Finding density functionals with machine learning. Physical review letters, 108(25), 253002.
[12] Gal, T. (2011). The ground-state energy and external potential as functionals of the electron density and their derivatives. arXiv preprint arXiv:1108.3865.
[13] Brockherde, F., Vogt, L., Li, L., Tuckerman, M. E., Burke, K., & Müller, K. R. (2017). Bypassing the Kohn-Sham equations with machine learning. Nature communications, 8(1), 1-10.
[14] Bogojeski, M., Vogt-Maranto, L., Tuckerman, M. E., Müller, K. R., & Burke, K. (2020). Quantum chemical accuracy from density functional approximations via machine learning. Nature communications, 11(1), 1-11.
[15] http://quantum-machine.org/datasets/ ; last visited 19.5.2022
[16] Khalid, F., Hanif, M. A., Rehman, S., Qadir, J., & Shafique, M. (2019, March). Fademl: Understanding the impact of pre-processing noise filtering on adversarial machine learning. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 902-907). IEEE.
[17] https://scikit-learn.org/stable/index.html ; last visited 19.5.2022

ISBN: 978-953-8246-26-5

Determining the Influence of Hardware on the Execution Times of Trained Machine Learning Models

Sandi Baressi Šegota [1*], Matko Glučina [2], Daniel Štifanić [1], Jelena Musulin [1], Ivan Lorencin [1], Nikola Anđelić [1], Zlatan Car [1]

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, sbaressisegota@riteh.hr, dstifanic@riteh.hr, jmusulin@riteh.hr, ilorencin@riteh.hr, nandelic@riteh.hr, car@riteh.hr
[2] University of Rijeka, Braće Mažuranića 10, 51000 Rijeka, Croatia, matko.glucina@riteh.hr.

Abstract: While many discussions and observations have been made regarding the execution times of machine learning (ML) model training, not many researchers have shown concern regarding the execution times of trained models when the model are applied for inference. In this paper, the researchers observe the execution times of a realistic hybrid system consisting of a YOLOv3 based detection model and two classification models based on VGG16 and VGG19 convolutional neural networks. The provided hybrid model is tested on five different hardware configurations – single core CPU execution, eight thread CPU execution, 48 thread CPU execution, single GOU execution and five GPU execution. The goal is to compare the execution times and determine the user satisfaction and ease of use on standard consumer hardware, readily available to the end users of the project, in comparison to a high performance computing architectures. The results show that the best performing architecture is a single GPU, but even the slowest solution (single core CPU), does not show an extremely slow time when inference is applied.

Keywords: artificial intelligence, convolutional neural networks, execution timing, high performance computing, hybrid systems, machine learning, model inference

1. introduction

Training times of the machine learning (ML) models are oft discussed [1], but the training processes are often offloaded to high performance computing machines [2, 3]. When it comes to model integration, the trained models will commonly be executed on regular user-accessible machines with comparatively poorer performance. For this reason, this paper investigates the performance of the realistic developed classification models. The models are based on the convolutional neural networks, developed as a hybrid system for detection and classification of the underground infrastructure. The analysis aims to determine the performances of realistic classification models to determine the usability and user satisfaction when using developed data-driven artificial intelligence-based models. For this reason, five different device sets are used, selected in order to compare the regular PC an end user may have access to (1 CPU, 8 CPUs, 1 GPU) to a HPC environment and determining images (48 CPUs, 5 GPUs).

2. Methodology

This section will first describe the used models, followed by the process of timing.

2.1. Used models

The data used for model training are collected as part of the CEKOM SmartCity.4DII project and consists of ground penetrating radar (GPR) imagery with the annotation data provided by the domain partners, who are the infrastructure owners. Three separate AI models are developed – a model for detecting the underground infrastructure location based on the YOLOv3 [4] algorithm, and two models for the material and width classification of the detected infrastructure – based on VGG16 [5] and VGG19 [6] convolutional neural network (CNN) architectures. The models have been trained to achieve a satisfactory model performance. The models are connected within a specific script, in which the output of the detection algorithm serves as the input of both classification models.

2.2. Timing

The developed hybrid algorithm system is trained using the main node of the Z4 HPC cluster [7]. The system was selected for two reasons: the availability of all the devices that wished to be tested, and high enough amounts of memory and input-output (IO) bandwidth where it will not be bottlenecking the performance. Using all systems for tests guarantees that no biases towards execution time are introduced due to the operating system or system libraries. Timing is performed using the Linux time program for time measurements [8]. The time command returns the real, system and user times after the given command has been executed. The user and system times refer to the times the CPU instructions are ran outside and within the kernel space. The real time refers to the so-called wall time – i.e. the time elapsed for the user while the execution is in progress.

Five different device sets are used for the performance measurement. CPU execution is used in three different modes – with a single threaded execution, and a multi-threaded execution with 8 or 48 threads respectively. In addition, the model inference is also run using GPUs – with a single GPU and five GPUs. This was selected to determine the needs and performance that an end user may expect. All the executions are run with niceness set to -20 (highest priority) [9]. The models are executed on 47 different images (a single microlocation images of the underground infrastructure).

The limitations are set using two commands. For setting CPUs, the taskset command is used [10]. This command allows us to set the number of threads the command executed through it (in this instance, the inference model) will use. For setting GPUs, the environmental variable CUDA_VISIBLE_DEVICES [11] is set to either “0” (for a single GPU) or “0,1,2,3,4” (to use all five GPUs). When

the CPU execution is performed, this variable is set to "-1". This environmental variable controls which CUDA capable devices are visible to the code that is executed in the shell.

The hardware used for the timing experiment is:

- CPU – Intel Xeon Gold 6240R with 24 Cores/48 Threads @2.4 GHz (boost up to 4.00 GHz), 35.75 MB of Cache, and
- GPU – NVIDIA Quadro RTX 6000 with 4608 CUDA cores and 24 GB of RAM.

In addition to the above, the server has 768 GB of RAM and all the models are run from Intel D3-S4510 240GB SSD, in RAID 1 [12]. The operating system that is installed on the machine is Linux Ubuntu 18.04 with the kernel version 4.15.0-176.

The execution times of the code can be split into library and model loading time and the poor execution (model prediction) times. Both times are presented, by measuring the execution time of the total program, and the execution time of just the model and module loading – enabling us to determine the execution times as a difference. These results are presented in the following section.

3. Results and discussion

The first table shows the obtained timings for each of the tested devices – focusing on the total execution times. As mentioned, the focus should be given on the real time, as this is the time the user experiences when running the model.

Table 1. *The timings for each of the tested devices*

Device	Real time [s]	User time [s]	System time [s]
1 CPU	117.309	113.514	3.645
8 CPU	103.032	270.567	120.693
48 CPU	50.613	872.58	237.702
1 GPU	33.351	37.632	21.206
5 GPU	41.058	40.235	27.511

The data in question shows the significant improvement in the total execution times, where the single GPU has the lowest execution time. The time for 5 GPUs is somewhat higher – which is probably caused by longer loading times. Observing execution times, we can see that the system times significantly increase when multiple CPUs are used due to the data transfer and execution handling. User times also increase due each of the individual threads being counted separately. Higher amount of CPUs shows the lowering of the times, but it is not significant in regards to the timings.

Table 2 and Figure 2 show that the loading times are equal for the first two CPU cases, but lower when multiple CPUs are used, probably due to multiple cache chips being used for loading the data. In case of GPUs, the real times show a significant increase due to the fact that data needs to be separated and transferred to multiple GPUs.

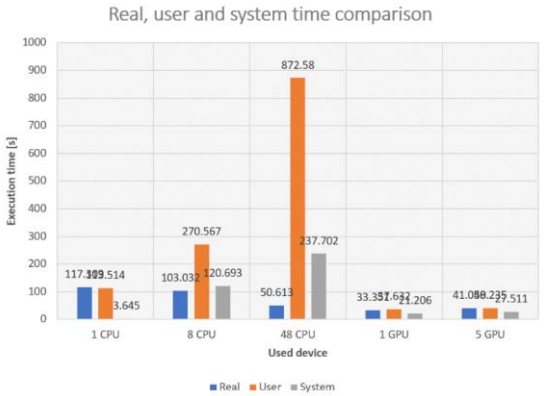


Figure 1. *Visualization of all the total measured times.*

Table 2. *The loading timings for each of the tested devices*

Device	Real time [s]	User time [s]	System time [s]
1 CPU	25.471	9.634	3.189
8 CPU	25.72	9.749	3.194
48 CPU	14.48	16.333	16.323
1 GPU	21.752	17.98	13.598
5 GPU	33.696	27.828	17.84

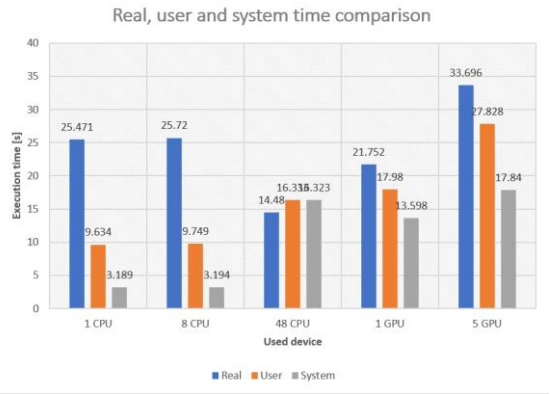


Figure 2. *Visualization of all the measured loading times.*

Table 3 and Figure 3 demonstrate the pure execution times, obtained as a difference between the total timings and the loading times. This is meant to show the time the models actually spend executing on the accelerators. This shows that the 5 GPUs are significantly the fastest in comparison to the other devices.

Table 3. *The execution timings for each of the tested devices*

Device	Real time [s]	User time [s]	System time [s]
1 CPU	91.838	103.88	0.456
8 CPU	77.312	260.818	117.499
48 CPU	36.133	856.247	221.379
1 GPU	11.599	19.652	7.608
5 GPU	7.362	12.407	9.671

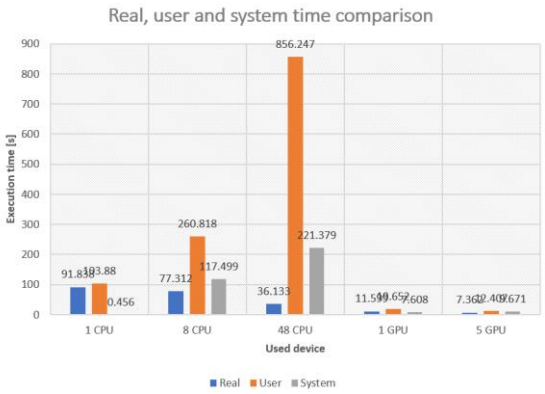


Figure 3. *Visualization of all the determined execution times.*

4. Conclusion

The data in question shows that the fastest model execution times, when observing the total real times, which is what a user would experience, is achieved when using the GPU. Still, the cost to the end user needs to be considered. Taking the cost into the account, it van be seen that all of the used devices provide satisfactory times, with a maximum total real time below 2 minutes.

Acknowledgments

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, University of Rijeka scientific grant uniri-tehnic-18-275-1447.

References

[1] Seiffert, Chris, et al. "RUSBoost: Improving classification performance when training data is skewed." *2008 19th international conference on pattern recognition*. IEEE, 2008.
[2] Car, Zlatan, et al. "Modeling the spread of COVID-19 infection using a multilayer perceptron." *Computational and mathematical methods in medicine* 2020 (2020)..
[3] Kahle, James A., Jaime Moreno, and Dan Dreps. "2.1 Summit and Sierra: designing AI/HPC supercomputers." *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019.
[4] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
[5] Lorencin, Ivan, et al. "On urinary bladder cancer diagnosis: Utilization of deep convolutional generative adversarial networks for data augmentation." *Biology* 10.3 (2021): 175.
[6] Lorencin, Ivan, et al. "Automatic evaluation of the lung condition of COVID-19 patients using X-ray images and convolutional neural networks." Journal of Personalized Medicine 11.1 (2021): 28.
[7] Musulin, Jelena, et al. "Multiclass Classification of Oral Squamous Cell Carcinoma." Ri-STEM-2021 (2021): 7.
[8] Linux manual page, "time(1)", https://man7.org/linux/man-pages/man1/time.1.html
[9] Jonk, Ruben, et al. "Timing prediction for service-based applications mapped on linux-based multi-core platforms." 2018 21st Euromicro Conference on Digital System Design (DSD). IEEE, 2018.
[10] Linux manual page, "timeset(1)", https://man7.org/linux/man-pages/man1/taskset.1.html
[11] Šego Sanders, Jason, and Edward Kandrot. CUDA by example: an introduction to general-purpose GPU programming. Addison-Wesley Professional, 2010.ta,
[12] Sandi Baressi, et al. "A Brief Note on the Influence of Storage Choices on Machine Learning Algorithm Training Times." Ri-STEM-2021 (2021): 19.

ISBN: 978-953-8246-26-5

License plate recognition using convolutional neural networks

Andrea Drndić, Matea Vučetić

[1] Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia.

Abstract: This paper describes the process of configuring and training a license plate recognition algorithm using Google Colab as the host operating system (OS) and You Only Look Once v4 (YOLOv4) with Darknet for the neural network foundation. This algorithm speeds up the recognition process and reduces the number of human resources required. The main problem with the process is that the algorithm must be able to recognize the license plate under different conditions (in the rain, at night, during movement, etc.), also the system should not be fooled by different shapes, colors, and license plate sizes depending on the country they come from. The input dataset was obtained for Google's Open Image Dataset and contained 1500 images of "Vehicle registration plates" together with their corresponding labels. After training with 6000 iterations for 24 hours, the algorithm got to an average precision of 85.23%. Better average precision could be accomplished by increasing the number of iterations and input dataset size, but that would prolong the training time by a considerable amount.

Keywords: Artificial intelligence, CNN, Licence plate, Machine Learning, YOLOv4.

1. introduction

Automatic license plate recognition (ALPR) plays an important role in numerous applications such as unattended parking lots, security control of restricted areas, traffic law enforcement, and automatic toll collection [1]. It speeds up the recognition process and reduces the number of human resources required. ALPR recognizes a vehicle's license plate number from an image or images taken in either color, black and white, or infrared camera. The main problem with license plate recognition is that the algorithm must be able to recognize the license plate under different conditions (in the rain, at night, during movement, dirty, etc.), also the system should not be fooled by different shapes, colors, and license plate sizes depending on the country they come from. What also needs to be taken into account is that the license plate may or may not be in the picture, and if it is, there could be more than one. The quality of the acquired images is a major factor in the success of the ALPR [2]. All these problems represent a great challenge for such a system, with reliability and average precision being a of upmost importance. Today we have a lot of different researches using convolutional neural networks (CNN).

Lorencin et al. (2021) [3] uses CNN to make a procedure that allows in-vivo evaluation of bladder mucosa without the need for a biopsy. The main focus in his research was focused on increasing the performance of classifiers by applying dataset pre-processing, while using established CNN architectures that have achieved high classification performance in solving various problems in practice.

Also, Narin et al. (2021) [4] in his work used pre-trained CNN-based models (ResNet50, ResNet101, ResNet152, InceptionV3 and Inception-ResNetV2) that have been proposed for the detection of coronavirus pneumonia - infected patients using chest X-ray radiographs. They implemented three different binary classifications with four classes (COVID-19, normal (healthy), viral pneumonia and bacterial pneumonia) by using five-fold cross-validation.

CNN is often used in medicine for diagnosis of various diagnoses and diseases. The main focus of this project will be to make an algorithm that will recognize license plates on selected photos with a high percent of accuracy using CNN. Will CNN be as good at recognizing license plates as it was for diseases? Is this method equally fast and efficient for that purpose?

2. Methodology

The main algorithm used in this paper is You Only Look Once v4 (YOLOv4) with Darknet [5]. YOLOv4 depending on the configuration can recognize different objects simultaneously. The Convolutional Neural network (CNN) is written in CUDA language and implemented in the Darknet software bundle to enable the use of GPU accelerated learning in YOLO implementations. The core of the YOLO target detection algorithm [6] lies in the model's small size and fast calculation speed. Instead of processing every pixel in the image, which increases the processing time, the license plate can be distinguished by its features, and therefore the system processes only the pixels that have these features [5]. YOLO uses a single bounding box regression to predict the height, width, center, and class of objects [7]. To reduce the time to train a reliable algorithm, Google Colab was used as the main GPU backend. Colab allows anyone to write and execute arbitrary python code through a browser and provides free access to GPUs for CUDA usage.

The YOLO algorithm [7] uses the following threes methods for object identification:

1. Residual blocks divide the image into various grids and every grid cell will detect objects that appear within them.
2. Bounding box regression then predicts the height, width, center, and class of the detected object.
3. Intersection over union (IOU) ensures that the predicted bounding boxes are the right size and eliminates the ones that don't fit the detected objects precisely.

Figure 1. represents the object detection flow of the YOLO algorithm, combining the above 3 steps:

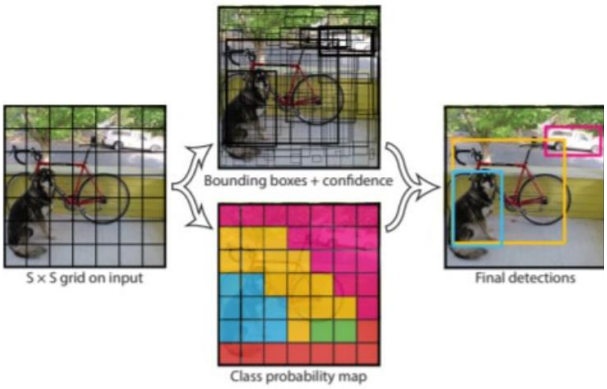


Figure 1. *An example of the YOLO algorithm flow in a given image*

After installing YOLOv4 with Darknet in Colab [8], Google Drive had to be connected to the instance for transferring data and picture samples. The object detector relies on a good dataset of images and labels for training. With some python code and Windows PowerShell, a sample of 1500 images of "Vehicle registration plates" was downloaded from Google's Open Image Dataset and correctly labeled which represent the input data [9]. In addition to the above, a further 300 images were downloaded into a separate folder serving as the validation dataset. The external validation dataset was selected in order to reduce the effect of overfitting on the test dataset during the validation [10]. In this example, it's 20% of the size of the input data (20% of 1500 = 300). The test dataset was also chosen to be 20% of the input data. Using a couple more python commands, the downloaded images and labels were formatted to YOLO format and uploaded into a separate folder in Colab.

Before running the training algorithm, it has to be configured with the right values (number of iterations, classes…). This is done in the "yolov4-obj.cfg" file and the main variables are represented in Table1.:

Table 1. *Configured variables in the YOLOv4 cfg file*

MAX_BATCHES	WIDTH	HEIGHT	STEPS
6000	416	416	8000, 9000

With YOLOv4 configured and the input dataset ready, training can be started and will last depending on the hardware used. After a little more than 24 hours, the algorithm reached the 6000 iterations limit and the custom object detector was completed.

3. Results and Discussion

Results from the YOLOv4 console showed that after only 1000 iterations, the algorithm could detect license plates with an average precision of 79.44%. After the completed 6000 iterations this percentage went up to 88.92% meaning that by increasing the number of iterations 6 times the average precision went up 9.48%.

Figure 2. *Output image of the YOLO algorithm after successful detection*

Figures 2 and 3 show the license plate bounding box on the image with the corresponding average precision of 99% and 70%. Such variety is normal because the license plate in Figure 3 is much smaller in comparison to Figure 2 with the numbers unreadable.

Figure 3. *Output image of the YOLO algorithm after successful detection*

4. Conclusion

The main task of this paper was accomplished as to build a license plate recognition algorithm. With an average precision of 85.23% it's accurate enough to recognize the license plates on the majority of images. The performance of deep learning depends on some hyper-parameter, such as the architecture of the CNN, the number of filters in the convolution layer, dropout, optimizer, and the number of training data, but increasing the input data would prolong the training time by a considerable amount [11]. Future work could include adding a secondary object class in the YOLO configuration so that besides the license plates, the system could detect the type of vehicle (car, truck, etc…) such as the one developed by the Korea Electronics Technology Institute [12]. A large number of license plate recognition (ALPR) and make and model recognition (MMR) systems have been developed to relieve human operators of the tedious task of explicitly detecting and identifying a wide range of cars [12].

Acknowledgments

We sincerely thank our mentor Matko Glučina who guided us through the whole process of understanding and training the algorithm described in this paper.

Reference

[1] Chang, Shyang-Lih, et al. "Automatic license plate recognition." IEEE transactions on intelligent transportation systems 5.1 (2004): 42-53.: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1271288

[2] Chang, Shyang-Lih, et al. "Automatic license plate recognition.": https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6213519

[3] Lorencin, I.; Baressi Šegota, S.; Anđelić, N.; Mrzljak, V.; Ćabov, T.; Španjol, J.; Car, Z. On Urinary Bladder Cancer Diagnosis: Utilization of Deep Convolutional Generative Adversarial Networks for Data Augmentation. Biology 2021, 10, 175. https://doi.org/10.3390/biology10030175

[4] Narin, A., Kaya, C. & Pamuk, Z. Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. Pattern Anal Applic 24, 1207–1220 (2021). https://doi.org/10.1007/s10044-021-00984-y

[5] YOLO algorithm for object detection: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

[6] A Review of Yolo Algorithm Developments: https://pdf.sciencedirectassets.com/280203/

[7] Introduction to YOLO Algorithm for Object Detection: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

[8] Google Colab sample project: https://colab.research.google.com/drive/1qENBftx962vFwz24KTxqPCvPD2Jv009o#scrollTo=YQUDXrhJekxl

[9] Google's Open Image Dataset: https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=segmentation&r=false&c=%2Fm%2F01jfm_

[10] Automatic Bunch Detection: https://doi.org/10.3390/agronomy12020319

[11] Triwiyanto, Triwiyanto, I. Putu Alit Pawana, and Mauridhi Hery Purnomo. "An improved performance of deep learning based on convolution neural network to classify the hand motion by evaluating hyper parameter.": https://ieeexplore.ieee.org/abstract/document/9107136

[12] Park, S.-H.; Yu, S.-B.; Kim, J.-A.; Yoon, H. An All-in-One Vehicle Type and License Plate Recognition System Using YOLOv4. Sensors 2022, 22, 921. https://doi.org/10.3390/s22030921

ISBN: 978-953-8246-26-5

Object detection model for parked car detection using YOLOv4 algorithm

Karlo Severinski [1*], Tajana Cvija [2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, kseverinski@riteh.hr, tcvija@riteh.hr.

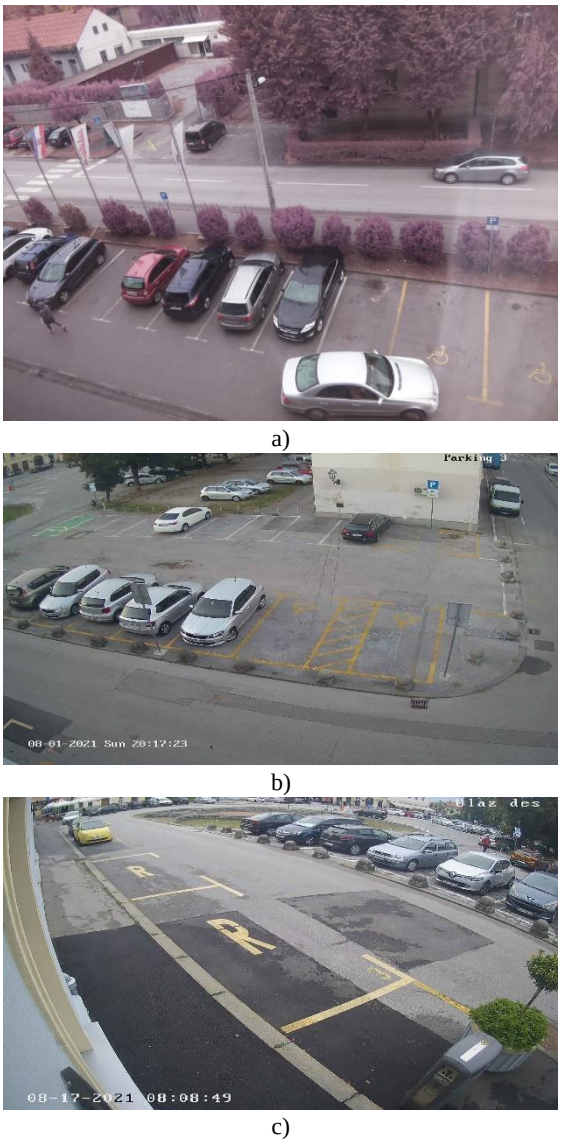
Abstract: This paper discusses the use of the YOLOv4 algorithm for parked car detection. A custom dataset of car parking lot images will be annotated and used for the training of the YOLOv4 object detection model using the transfer learning method. The goal of this paper is to create a model that detects cars from a certain camera angle, similar to the angle of a parking camera so that it can later be used in parking space occupancy detection. The dataset was labeled with the LabelImg annotation tool and the google colab notebook was used for training because of its GPU. The obtained results are presented in detail in the paper. The trained model results are satisfactory so that they can be used for the mentioned purpose of parking space occupancy detection.
Keywords: artificial intelligence, LabelImg, object detection, Python, YOLOv4

1. Introduction

Although object detection can be traced back to the years before 2014, when the standard was “traditional object detection”, today's algorithms use “deep learning-based detection” for object detection. One of such algorithms is the YOLO algorithm [1]. YOLO stands for “you only look once”, and is an algorithm that uses fixed-grid regression for object detection. This approach makes real-time and accurate object detection which can be used in various fields [2]. In this paper, one of these algorithms is going to be used for car detection on a custom dataset that consists of parking lot pictures, from ordinary security and parking cameras. Some of the similar problems that use a similar idea are face mask detection [3], bridge crack detection [4], various fruits detection [5, 6], and others. YOLOv4 algorithm is a high-efficiency model detection model that runs extremely fast, and faster than most state-of-the-art models [7, 8]. This makes it a great choice for implications such as detecting cars in a parking lot and determining the occupancy of the parking spaces. With this, the user saves a lot of time and money for the same job done by the modern parking space occupancy systems. Because of the custom dataset collected from the parking and security cameras, the dataset needs to be annotated. In the paper, the labels were made with the software called LabelImg. LabelImg is a free offline annotation tool that is used for the graphical annotation of images [9]. Since LabelImg is free and supports both image and video annotations it’s a great choice for this purpose [10]. The custom training set consists of 340 parking lot pictures and a testing set that contains 85 images. Also, for the sake of additional testing and validation of the model, another set of parking lot images was downloaded from the internet.

2. Methodology

As previously stated, the custom dataset consists of parking lot pictures. The difference between data is mostly the camera angle, but there are several different parking lots in the given dataset. The examples of the dataset are given in figure 1.



a)

b)

c)

Figure 1. a) Parking lot example 1, b) Parking lot example 2, c) Parking lot example 2 from a different camera angle

The whole custom dataset must be properly labelled so that the model can be trained. The next step is to label the dataset with LabelImg annotation tool. An example of a labelled image is shown in figure 2.

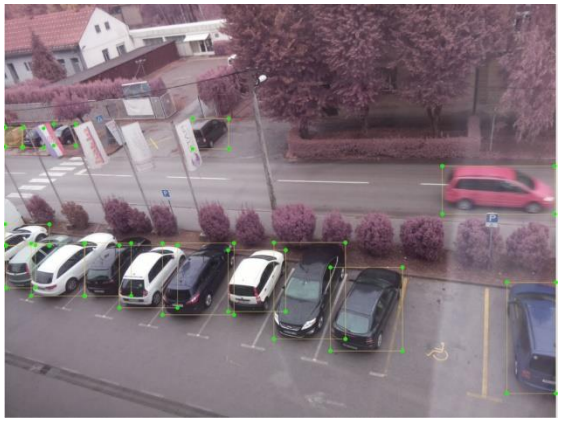
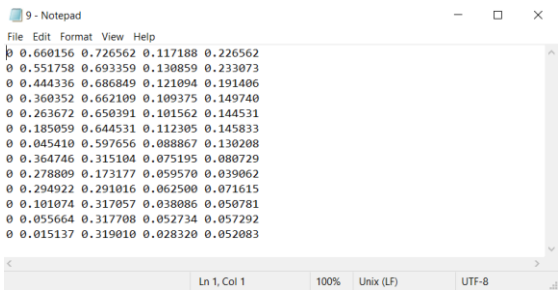


Figure 2. LabelImg labelling example

LabelImg gives coordinates of an object in a .txt format that is supported by the YOLOv4 algorithm that uses the darknet [11]. LabelImg also gives the classes .txt file that prints out all the classes labelled for a certain dataset. This can be seen in figure 3. Since here only one class is labelled, classes.txt only consists of one class.



a)

b)

Figure 3. a) coordinates .txt file, b) classes .txt file

The next step is to train the model using the given YOLO algorithm. In this paper, the google colab notebook was used for training the model, because of its free and powerful GPU. Also, the previously mentioned darknet model was used for training. It should be mentioned that the transfer learning method was used, since pretrained weights should give better end results [12].

3. Results and Discussion

The model was trained for 1800 iterations with its loss function given in figure 4. Also, in figure 5, the results of object detection are given.

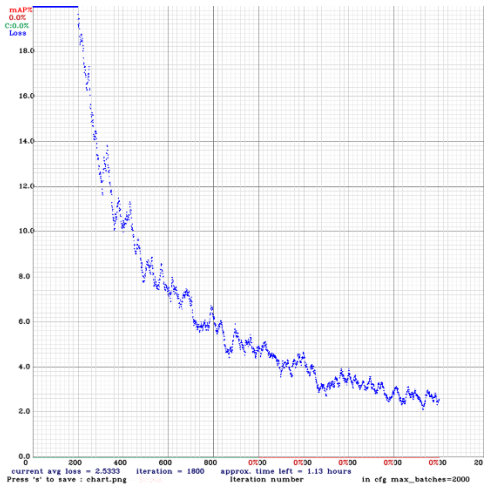
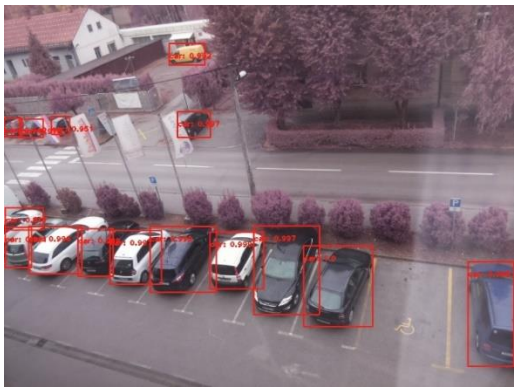
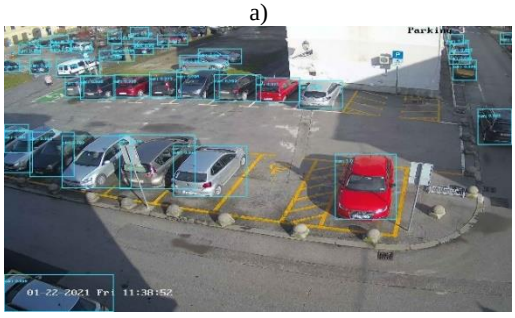


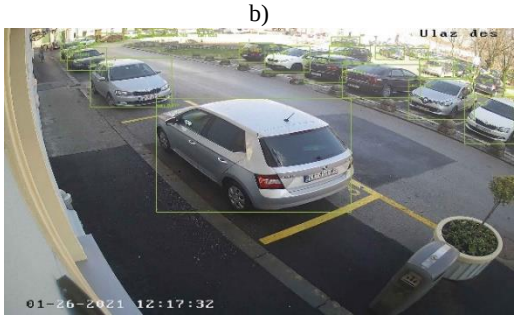
Figure 4. Loss function for the trained model



a)



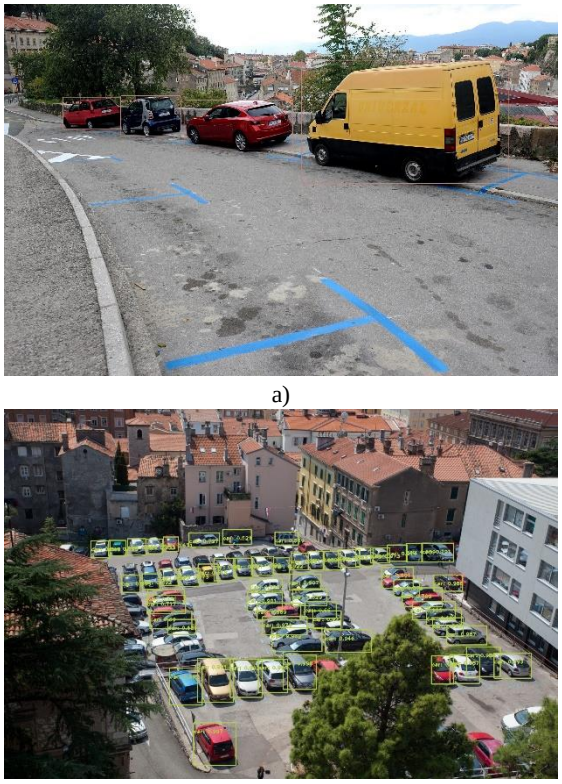
b)



c)

Figure 5. a) object detection for parking example 1, object detection for parking example 2, object detection for parking example 2 from a different camera angle

As previously mentioned, the validation dataset was collected. The model was also tested with the validation dataset to see how the model will act in the never seen environment. The results are shown in figure 6.



a)

b)

Figure 6. a) Object detection on a validation dataset example for a street parking in Rijeka (https://www.novilist.hr/rijeka-regija/rijeka/rijeka-plus-otkrio-koliko-ce-naplacivati-parking-u-ulici-franje-rackog-stanari-su-to-trazili/), b) Object detection on a validation dataset for city parking lot in Rijeka (https://riportal.net.hr/rijeka/rijeka-plus-od-subote-u-rijeci-nova-pravila-vezana-uz-placanje-dnevne-parkirne-karte/108700/)

4. Conclusion

From the images used to evaluate the model given in figures 5 and 6, the conclusion is that the trained model does a great job in determining and detecting cars both within the test set and the validation set. The transfer learning method secured better detection accuracy with fewer iterations of the algorithm needed. Since the future work on this topic is the detection of vacant and occupied parking spaces in a parking lot using a parking camera, the conclusion is that the given results are satisfying for that purpose.

Acknowledgments

We want to thank the assistant dr. sc. Ivan Lorencin mag. ing. el. and assistant Matko Glučina mag. ing. el. They helped us in the making of this paper. We hope there will be a lot of joint future work and projects. We would also like to thank the company devlab d.o.o. for providing us the custom dataset that was used in the paper. Also, we would like to thank all those who organized this conference and enabled us, students, to participate with our work.

Reference

[1] Zou, Zhengxia, et al. "Object detection in 20 years: A survey." arXiv preprint arXiv:1905.05055 (2019). https://arxiv.org/pdf/1905.05055.pdf

[2] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.

[3] Yu, Jimin, and Wei Zhang. "Face mask wearing detection algorithm based on improved YOLO-v4." Sensors 21.9 (2021): 3263. https://www.mdpi.com/1424-8220/21/9/3263

[4] Yu, Zhenwei, Yonggang Shen, and Chenkai Shen. "A real-time detection approach for bridge cracks based on YOLOv4-FPM." Automation in Construction 122 (2021): 103514. https://www.sciencedirect.com/science/article/abs/pii/S0926580520310943

[5] Gai, Rongli, Na Chen, and Hai Yuan. "A detection algorithm for cherry fruits based on the improved YOLO-v4 model." Neural Computing and Applications (2021): 1-12. https://link.springer.com/article/10.1007/s00521-021-06029-z

[6] Tian, Yunong, et al. "Apple detection during different growth stages in orchards using the improved YOLO-V3 model." Computers and electronics in agriculture 157 (2019): 417-426.

[7] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020). https://arxiv.org/pdf/2004.10934.pdf

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

[9] Lin, T.: Labelimg. https://github.com/tzutalin/labelImg Accessed 14 April 2022

[10] Reis, Pedro Miguel Lima de Sousa. "Data Labeling tools for Computer Vision: a Review." (2022). https://run.unl.pt/bitstream/10362/135873/1/TCDMAA0144.pdf

[11] Bochkovskiy, Alexey: darknet https://github.com/AlexeyAB/darknet Accessed 14 April 2022

[12] Raza, Kazim, and Song Hong. "Fast and accurate fish detection design with improved YOLO-v3 model and transfer learning." Int. J. Adv. Comput. Sci. Appl 11 (2020): 7-16. https://www.researchgate.net/profile/Kazim-Raza-5/publication/339640242_Fast_and_Accurate_Fish_Detection_Design_with_Improved_YOLO-v3_Model_and_Transfer_Learning/links/5e707084a6fdccc06e94b160/Fast-and-Accurate-Fish-Detection-Design-with-Improved-YOLO-v3-Model-and-Transfer-Learning.pdf

ISBN: 978-953-8246-26-5

KNN Classification of APS Failure Data in Scania Trucks

Denis Mijolović [12*]

[1] University of Rijeka, Faculty of Engineering, Vukovarska 58, 51000 Rijeka, Croatia, dmijolovic@riteh.hr.
[2] Uppsala University, Department of Civil and Industrial Engineering, Ångströmlaboratoriet, Lägerhyddsvägen 1, 752 37 Uppsala, Sweden, denis.mijolovic.0720@student.uu.se.

Abstract: This paper seeks to conduct and interpret k-Nearest Neighbours (abrv. kNN) supervised classification analysis on the beforementioned dataset provided by Scania CV AB. With growing relevance of machine learning in business intelligence, Scania CV AB aimed to achieve competitive edge and financial benefits through analysis of telematic data extracted from their fleet. In this research, KNN classifier is used for classification model. Prior to model training, data analysis is conducted with an aim to check for classifier imbalance that would negatively affect the results of the model. Model prediction accuracy is interpreted and discussed as the model accuracy for the best estimator is 90.19% with ± 3.42% standard deviation whilst normalised confusion matrix showed discrepancy that does back-up the accuracy score due to discrepancy in true positives and false positives. Finally, improvements to the dataset pre-processing methods are additionally proposed for future work.

Keywords: Failure detection, K-nearest neighbour, Machine Learning, K-nearest neighbour.

1 Introduction

With Machine Learning and Data Analytics becoming an integral part for leading companies in competitive industries and markets, it is nominal that successful utilisation of the latest findings and practices in machine learning yields economic and organisational benefits to those who make positive investment decisions directed at data analytics departments. These statements are backed by other research and available secondary data on industrial economics and socio-technological absorption of innovation amongst industrial actants [9]. One of the aforementioned companies is Scania CV AB, Swedish-based heavy vehicle-making company. Telematic readings have been released by the company in 2017 for purpose of data analyses and machine learning model training. According to Dua and Graff, these analyses and models could be used for improvements in predictive maintenance induced by potential failures of Air Pressure System (abrv. *APS*) and corresponding cost optimisation [10]. Dataset has two classes: *pos* (representing APS-induced failure) and *neg* (representing non-APS failure).

2 Dataset Analysis

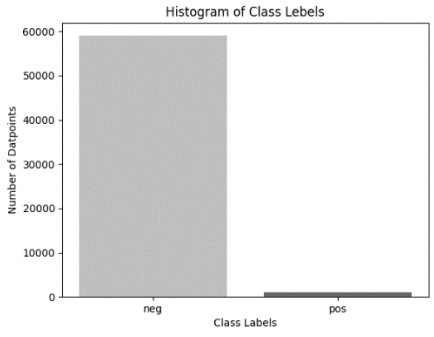


Figure 12. *Class Analysis Histogram.*

Before dwelling deeper into dataset analysis, it is worthwhile mentioning both supervised and unsupervised machine learning models are indubitably reliant on dataset quality – therefore implying necessity for data inspection and analyses prior to making final conclusions based of the output results.

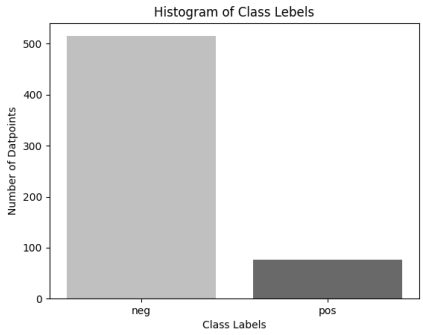


Figure 13. *Class Analysis Histogram After Data Manipulation.*

Larsen and Becker inspect this topic further through clarification of automatic machine learning excellence criteria, some of which are relevant for the future course of this article as well (17-20). Namely, criteria like *productivity* and *understanding and learning* come as relevant; as the former implies iterative process of model optimization and data imputation, whilst the later signifies the importance of framing datasets and machine learning models within the informational context of each specific use case [9]. The beforementioned criteria is used as a guideline for necessity of data pre-processing, which had to be utilised due to high data imbalance within the dataset as presented Figure 12. – showing 98.33% of dataset classes are attributed as *neg* and 1.67% as *pos* respectively. The necessity for pre-processing the data is further backed by the fact that 98.83% of dataset columns and 99.02% of dataset rows contain missing values – which could influence or even obstruct algorithms that are sensitive to non-value inputs [9]. While there are contextually applicable guidelines for highly imbalanced data scenarios that suggest the use of various data imputation methods, this paper investigates kNN classification results for the case of dropping of indices containing *not a number* values. Dataset manipulation improved data balance as shown in Figure 13. (87.14% *neg* and 12.86% *pos*)whilst no non-value data points were preserved.

3 KNN Classification Result and Discussion

Detailed kNN classification result based on processed dataset and accuracy score measurement is presented in Table 1. available in Appendix A. Best result was achieved for model execution with 3 nearest neighbours and distance-based weight function. Additionally, neither positive nor negative effects were found through alternation of algorithm method presets. KNN classification model results have thus initially shown satisfactory results according to metrics used in the method. However, kNN algorithm outputs classification results are highly susceptible to noise when fewer neighbours are used – thereby proposing a need for additional analyses of given results and the dataset itself, especially if analysed dataset has high volume and high noise [11].

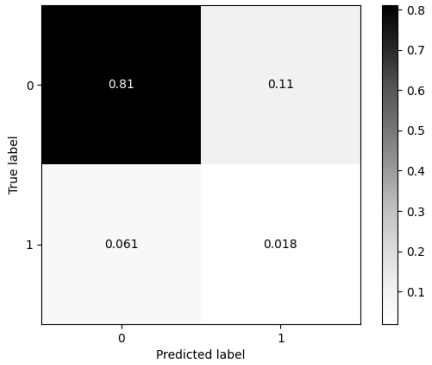


Figure 14. *Normalised Confusion Matrix of Best Estimator Prediction Model.*

To evaluate these statements within context of this paper, normalised confusion matrix is used as an additional measure of model quality, as shown on Figure 14. While the model successfully predicts true negatives, noticeable negative discrepancy between true positives and false positives shows additional actions are required in dataset balancing and algorithm parameters.

4 Conclusion and future work

Machine learning has become a gateway for companies to utilise their own data through conversion to business intelligence models that help them drive their competitive advantage [12]. Scania CV AB is amongst those companies through their offering of data-driven services [13]. The research work was carried out with the purpose of evaluation of k-nearest neighbour learning model for the case of APS failure classification at Scania CV AB. Prior to model training, initial dataset was analysed. Analysis has shown the initial dataset was highly imbalanced. With imbalanced learning being a relevant long-term issue for learning models based on everyday applications, methods for dealing with imbalanced data are continuously being developed [14].

With none of the traditional methods for dealing with imbalanced data as mentioned by Maciejewski and Stefanowski, Zhang and Zhang, or Lin, Weng and Lai used, k-nearest neighbours learning model managed to predict the outcome of test dataset with 90.19% score accuracy and ± 3.42% standard deviation. Normalised confusion matrix has been constructed to verify plausibility of given results prior to coming with a final conclusion. Findings form the confusion matrix have shown the importance of dataset pre-processing, as the overall score had significant negative discrepancy between true positives and false positives – thereby verifying claims by Itoo, Meenakshi and Singh and Ding, Chen and Dong (241-242), who signify the neccessity of additional dataset pre-processing or algorithm modification that would gain improvements for imbalanced datasets. Therefore, in future research, application of additional dataset pre-processing and imputation methods is advised to achieve plausible model results for this practical application.

Reference

[1] K. R. Larsen and D. S. Becker, “What Is Machine Learning?,” in Automated Machine Learning for Business, Oxford University Press, 2021, pp. 1-21.
[2] D. Dua and C. Graff, “Machine Learning Repository,” 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks. [Accessed 10 May 2022].
[3] A. Shokrzade, M. Ramezani, F. A. Tab and M. A. Mohammad, “A novel extreme learning machine based kNN classification method for dealing with big data,” Expert Systems With Applications, vol. 183, 2021.
[4] Microsoft Corporation, “Annual Report 2021,” Microsoft, Redmond, 2021.
[5] Scania AB, “Data-driven services,” 2022. [Online]. [Accessed 14 May 2022].
[6] H. Ding, L. Chen, L. Dong, Z. Fu and X. Cui, “Imbalanced data classification: A KNN and generative adversarial,” Future Generation Computer Systems, pp. 240-254, 2022.
[7] F. Itoo, M. Meenakshi and S. Singh, “Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection,” International Journal of Information Technology, vol. 13, p. 1503–1511, 2021.
[8] K.-B. Lin, W. Weng, K. Lai and P. Lu, “Imbalance data classification algorithm based on SVM and clustering function,” in 9th International Conference on Computer Science & Education, 2014.
[9] T. Maciejewski and J. Stefanowski, “Local neighbourhood extension of SMOTE for mining imbalanced data,” in 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 2011.
[10] L. Zhang and D. Zhang, “Evolutionary cost-sensitive extreme learning machine.,” IEEE transactions on neural networks and learning systems, pp. 3045-3060, 2016.

Appendix A

Table 1. *kNN Classification Results*

Accuracy Score	Std Accuracy Score	Algorithm	n-Neighbours	Weights
0.9019085600341831821 0	0.03419830647884587738	Auto	3	Distance
0.9019085600341831821 0	0.03419830647884587738	Ball Tree	3	Distance
0.9019085600341831821 0	0.03419830647884587738	KD Tree	3	Distance
0.9019085600341831821 0	0.03419830647884587738	Brute	3	Distance
0.8968238142714713978 5	0.03872933546642255698	Auto	5	Distance
0.8968238142714713978 5	0.03872933546642255698	Ball Tree	5	Distance
0.8968238142714713978 5	0.03872933546642255698	KD Tree	5	Distance
0.8968238142714713978 5	0.03872933546642255698	Brute	5	Distance
0.8968238142714713978 5	0.03225404456224516270	Auto	10	Distance
0.8968238142714713978 5	0.03225404456224516270	Ball Tree	10	Distance
0.8968238142714713978 5	0.03225404456224516270	KD Tree	10	Distance
0.8968238142714713978 5	0.03225404456224516270	Brute	10	Distance
0.8883777239709443884 8	0.03844926535841013127	Auto	3	Uniform
0.8883777239709443884 8	0.03844926535841013127	Ball Tree	3	Uniform
0.8883777239709443884 8	0.03844926535841013127	KD Tree	3	Uniform
0.8883777239709443884 8	0.03844926535841013127	Brute	3	Uniform
0.8849451645064805305 2	0.01714623555470045149	Auto	15	Distance
0.8849451645064805305 2	0.01714623555470045149	Ball Tree	15	Distance
0.8849451645064805305 2	0.01714623555470045149	KD Tree	15	Distance
0.8849451645064805305 2	0.01714623555470045149	Brute	15	Distance
0.8832644922375729734 7	0.01934014904361845699	Auto	5	Uniform
0.8832644922375729734 7	0.01934014904361845699	Ball Tree	5	Uniform
0.8832644922375729734 7	0.01934014904361845699	KD Tree	5	Uniform
0.8832644922375729734 7	0.01934014904361845699	Brute	5	Uniform
0.8748041589517162597 3	0.01931277384389774865	Auto	15	Uniform
0.8748041589517162597 3	0.01931277384389774865	Ball Tree	15	Uniform
0.8748041589517162597 3	0.01931277384389774865	KD Tree	15	Uniform
0.8748041589517162597 3	0.01931277384389774865	Brute	15	Uniform
0.8714285714285713302 4	0.01902845782507361222	Auto	10	Uniform
0.8714285714285713302 4	0.01902845782507361222	Ball Tree	10	Uniform
0.8714285714285713302 4	0.01902845782507361222	KD Tree	10	Uniform
0.8714285714285713302 4	0.01902845782507361222	Brute	10	Uniform

ISBN: 978-953-8246-26-5

Influence of data scaling/normalization techniques on estimation accuracies of ground-state energies using genetic programming

Nikola ANĐELIĆ [1*], Sandi BARESSI ŠEGOTA[1], Ivan LORENCIN[1], Matko GLUČINA [2]

[1] Affiliation: Faculty of Engineering - University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia, email nandelic@riteh.hr, sbaressisegota@riteh.hr, ilorencin@riteh.hr
[2] Affiliation: University of Rijeka, Trg braće Mažuranića 10, 51000 Rijeka, Croatia, email: matko.glucina@uniri.hr .

Abstract: In this paper, the influence of data scaling/normalizing techniques on the estimation accuracy of the ground state energies of molecules based on combination of C,H,N,O,P, and S atoms achieved by symbolic expressions obtained by applying genetic programming (GP) was investigated. For this investigation, a function with random selection of GP parameters was developed, to find optimal GP parameters. The methods for evaluation of obtained symbolic expressions using GP were: coefficient of determination (R^2), mean absolute error (MAE), and root mean square error ($RMSE$). The results of the research showed that the symbolic expression obtained on the data set previously processed with the Power transformer technique has the highest estimation accuracy of the basic energy states of molecules ($R^2 = 0.9845, MAE = 0.3265, RMSE = 0.45$).

Keywords: Data Scaling Techniques, Genetic programming, Ground-state energies.

1. Introduction

Today, quantum mechanical simulations of molecules and solids are common due to the advances made in the field of electronic structure theory and increased computing power. Using these simulations vast amount of chemical compounds have been studied using different electronic structure methods, and an effort has been made to collect this data for screening an extensive space of chemical compounds and materials [1]. This screening has been useful in discovering not only new systems but also for the rational design of chemicals and materials for targeted applications [2,3]. Applying artificial intelligence (AI) to these types of datasets created the possibility of predicting the electronic structures of chemical compounds. So far, various types of machine learning algorithms have been applied to create AI models which could predict the ground state energy as reported in [4,5]. However, the transformation of these types of algorithms in the mathematical equation is almost impossible. So, to obtain the symbolic expression which can easily be used for the estimation of ground-state energies the genetic programming (GP) will be utilized. GP has been successfully utilized in the energy sector [6,7], and the development of epidemiology models [8,9]. One major advantage of utilizing genetic programming is that after training the symbolic expression is obtained which could be used to estimate or eventually predict the ground-state energies. In this paper, the idea is to investigate the influence of various data scaling/normalizing techniques of input dataset values on the estimation performance of ground-state energies achieved with obtained symbolic expression using GP.

2. Methodology

The dataset used in this paper is a publicly available dataset [10] that consists of intermolecular Coulomb repulsion operators (Coulomb matrices) for each molecule and the corresponding ground state energies that were computed using density functional theory (DFT) simulations. The Coulomb repulsion operators can be written as:

$$C_{IJ} = \begin{cases} 0.5\, Z_I^{2.4} & I = J \\ \dfrac{Z_I Z_J}{|R_I - R_J|} & I \neq J' \end{cases} \quad (1)$$

where Z_I are atomic number, R_I are atomic coordinates and I, J run over atoms in each molecule. The off-diagonal terms correspond to ionic repulsion between atoms I and J and the diagonal terms are obtained from a fit of the atomic numbers to the energies of isolated atoms. The Coulomb matrices represent the full set of parameters that DFT needs as inputs (Z_I -atomic numbers, R_I the atomic coordinates). The Coulomb matrices are symmetric matrices so for each molecule only the upper triangle of the matrix is provided in the dataset. The number of input variables, in this case, is 1275 while the output variable is the ground-state energies. The entire dataset consist of 16242 data points.

As already stated in this paper the genetic programming algorithm has been used to obtain symbolic expressions which could estimate the ground state energies of chemical compounds with high accuracies. For the purposes of this investigation, random GP parameter selection has been developed in order to find optimal GP parameters. The predefined range of GP parameters from which the developed function could select parameter values is shown in Table 1.

Table 1. The predefined range of GP parameters

Gp Parameters	Lower Bound	Upper Bound
Population size	100	2000
Number of Generations	200	500
Tournament size	10	200
Tree depth	(3,6)	(7,12)
Crossover	0.95	1
Subtree mutation	0.01	1
Hoist mutation	0.001	1
Point mutation	0.001	1
Stopping criteria	1×10^{-6}	1×10^{-3}
Maximum number of Samples	0.7	1
Constant range	-10000	10000

Parsimony coefficient	1×10^{-5}	1×10^{-4}

At the beginning of GP execution, the population is created and the number population size is defined with the population size parameter value. The population is created with a ramped half-and-half method. In GP all population members are treated as tree structures so to ensure variety between population members it is necessary to define tree depth value. The GP has two termination criteria one of which is the predefined number of generations (number of maximum generations) while the other is meeting the predefined minimum fitness value (stopping criteria value) and if one member of the population satisfies the condition. In case of fulfillment of the condition, i.e. if one population member satisfies the condition, the GP execution is terminated. In all these investigations the GP executions were terminated when the maximum number of generations is reached. The tournament selection size parameter is responsible in each generation for randomly selected population members that will compete to become the parents (winners) for the next generations. On these parents, the genetic operations (crossover and mutation) will be performed. In GP there are genetic operations i.e. crossover, subtree mutation, hoist mutation, and point mutation. The sum of all these coefficients should be equal to 1. If the sum is less than 1 then some winners of tournament selection will enter the next generation unchanged. For crossover operation, we need two tournament selection winners and on both tournament winners, random subtrees are selected. From the second tournament winner (donor) the subtree is used and inserted instead of the randomly selected subtree of the first tournament winner. In subtree mutation, the random subtree is selected on the tournament selection winner and is replaced with a randomly generated subtree. In hoist mutation, the random subtree is selected on the tournament selection winner, and on that subtree random node is selected which then replaces entire subtree. In point mutation the random nodes on tournament winner are selected which are replaced with randomly generated nodes. The maximum number of samples value represents the fraction of samples that are extracted from the train dataset and used to evaluate each population member. The constant range parameter is the predefined range of constant that is used in GP during execution to generate population and in genetic operations. Sometimes during the GP execution, the size of population members can rapidly grow without any benefit to the fitness value which can lead to memory overflow or longer execution time (bloat phenomenon). The parsimony coefficient parameter value is responsible for penalizing large population members by making them less favourable for tournament selection.

Also, different normalizing and scaling techniques have been applied to the dataset input values to investigate which normalizing or scaling techniques will produce the symbolic expression with the highest accuracy. The dataset transformation methods used on the input variables were:

- Maximum absolute scaling (MaxAbsScaler) – scale each input variable by its maximum absolute value. In this estimator, the data is scaled and each input variable is translated individually such that the maximum absolute value of each feature in the dataset will be 1.0. This method does not shift/center the data, and does not destroy any sparsity.
- Minimum Maximum Scaling (MinMaxScaler) – transform input variables by scaling each input variable to a given range. This method scales and translates each dataset input variable individually such that it is in a range between 0.0 and 1.0.
- Nomalizer – is a method of normalizing input variables individually to unit norm. Each sample with at least one non zero components is rescaled independently of other samples so that its norm equals 1.0 ,
- Power transformer - method for performing data transformations to make it more Gaussian-like.
- Robust scaler– scale features using statistics that are robust to outliers.
- Standard scaler – Standardize features by removing the mean and unit scaling to unit variance.

To evaluate obtained symbolic expressions three evaluation metrics have been used, and these are: coefficient of determination (R^2), mean absolute error (*MAE*), and root mean squared error (*RMSE*).
The coefficient of determination (R^2) is the proportion of the variation in the dependent variable that is predictable from the independent variable. For a dataset having n values that are denoted as $y_1, y_2, \dots, y_n$, each associated with fitted value $f_1, \dots, f_n$. The resiudals are defined as $e_i = y_i - f_i$. If $\overline{y}$ is the mean of the observed data $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ then the variability of the dataset can be measured with residual sum of squares $SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$, and the total sum of squares $SS_{tot} = \sum_i (y_i - \overline{y})^2$. From that the coefficient of determination can be formulated as:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}. \qquad (2)$$

The range of R^2 is between 0 and 1. When predicted values exactly match the observed values then the value of R^2 is equal to 1. The value of 0 means that all predicted values are equal to mean $\overline{y}$.
The mean absolute error (MAE) can be define as the measure of errors between paired observations expressing the same phenomenon. MAE is mathematically defined as:

$$MAE = \frac{\sum_{i=1}^{n} |f_i - y_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}, \qquad (3)$$

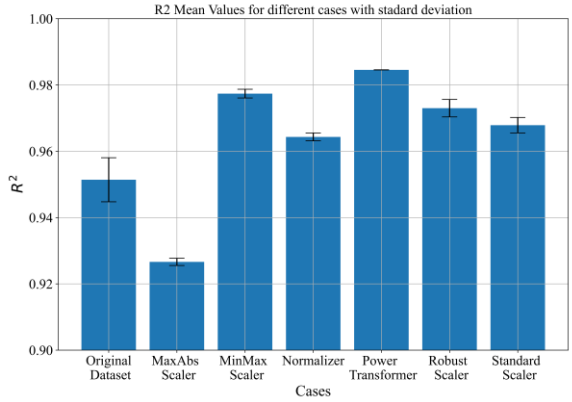
Where f_i is the prediction and y_i is the true value, and n is the number of samples in the dataset. The root mean

squared error of predicted values f_i and the dataset true values y_i can be calculated using the expression:

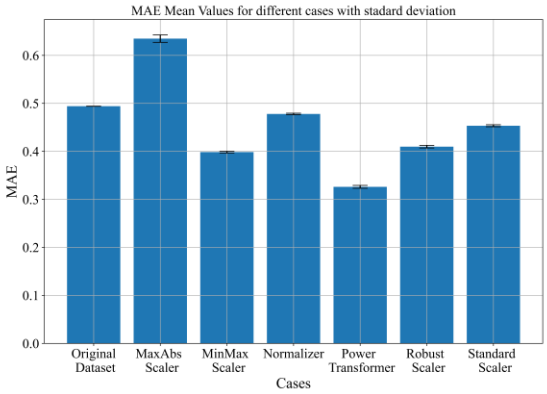
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(f_i - y_i)^2}{n}}. \quad (4)$$

3. Results and Discussion

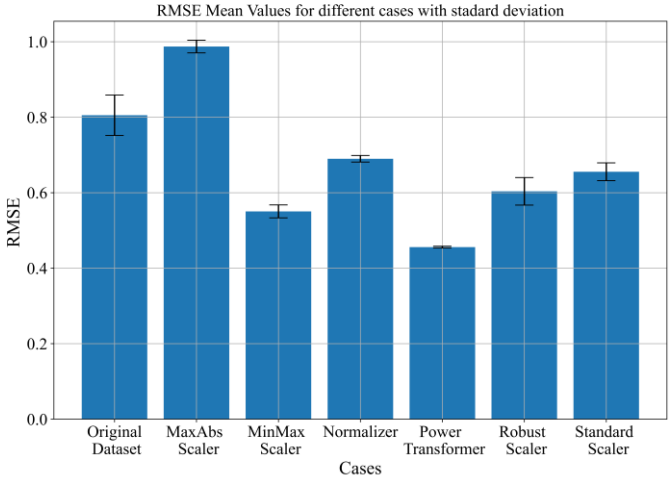
After each GP execution the symbolic expression was obtained and evaluated on train and test dataset to calculate the R^2, MAE, and $RMSE$. The idea is to investigate the difference between evaluation scores on train and test dataset, respectively. After these results were obtained on train and test dataset the mean values of R^2, MAE, and $RMSE$ and standard deviation were calculated. If the difference between evaluation results on train and test dataset is large it could potentially indicate the overtraining occurred. The mean values of R^2, MAE, and $RMSE$ as well standard deviation are shown in Figure 1 for all cases.



a) R^2 mean values for different cases with standard deviation



b) MAE mean values for different cases with standard deviation



c) RMSE mean values for different cases with standard deviation

Figure 1. *Comparison of estimation accuracies achieved with symbolic expressions obtained on differetly scaled/normalized dataset, a) R^2 values, b) MAE values, and c) RMSE values.*

To select the best estimation performance achieved by a symbolic expression is the highest value of R^2 and the lowest values of MAE and $RMSE$. The second criteria for selecting the best symbolic expression was the smallest error "bars" which in Figure 1 indicates the smallest standard deviation value. From all select symbolic expressions in each dataset normalizing/scaling case the best symbolic expression in terms of estimation performance was in the case of the Power Transformer.

The GP parameters and corresponding R^2, MAE and $RMSE$ values achieved with symbolic expressions for Power Transformer case is shown in Table 2.

Table 2. The GP parameters and estimation metric values for the best case (data transformed using power transformer)

Gp Parameters	R^2	MAE	RMSE
1675, 476, 176, (6, 11), 0.95, 0.02, 0.0086, 0.01, 4.9e-05, 0.97, (-2460.86, 1370.14), 3.04e-05	0.9845 ± 3.2621× 10^{-5}	0.326596 ± 0.0029	0.45 ± 0.0024

As seen from Table 2 the best symbolic expression was obtained with high value of population size and the population was propagated to 476 generations. In each generation 176 population members competed to become

parents of next generation. The depth of population trees was in range from 6 to 11. The dominating genetic opertor was crossover (0.95) when compared to other three mutation operators (0.02, 0.086, and 0.01). The stopping criteria was set to 4.9×10^{-5}. However, the stopping criteria was never met during GP execution so the GP was terminated after the maximum number of generation was reached. The 97% of training data was used randomly selected in each generation to evaluate each population member. The constants used to construct the population members and later in genetic operations were in range form -2460.86 to 1370.14. The parsimony coefficient was set to 3.04×10^{-5}. The value of this coefficient was very low however, the bloat phenomenon did not occur but the size of obtrained symbolic expression is large. Due to the large size of the symbolic expression the symbolic expression would not be shown.

4. Conclusion

In this paper the GP with random parameter selection and different data transformations performed on the input variables have been used to obtain symbolic expressions which could estimate the ground state energies with high accuracy. The results of the conducted investigation showed that the highest estimation accuracy was achieved in terms of R^2, *MAE* and *RMSE* values was in the case where the power transformer method was applied on input dataset variables. Conducted investigation showed that the dataset scaling/normalizing methods have influence of GP performance and in the end on the estimation accuracy of obtained symbolic expressions.

Acknowledgments

This research has been (partly) supported by the CEEPUS network Ciii-HR-0108, European Regional Development Fund under the grant K.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, Erasmus+ project WICT under the grant 2021-1-HR01-KA220-HED-000031177, project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447

References

[1] Curtarolo, Stefano, et al. "AFLOW: An automatic framework for high-throughput materials discovery." Computational Materials Science 58 (2012): 218-226.

[2] Hautier, Geoffroy, et al. "Finding nature's missing ternary oxide compounds using machine learning and density functional theory." Chemistry of Materials 22.12 (2010): 3762-3767.

[3] Pizzi, Giovanni, et al. "AiiDA: automated interactive infrastructure and database for computational science." Computational Materials Science 111 (2016): 218-230.

[4] Behler, Jörg. "Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations." Physical Chemistry Chemical Physics 13.40 (2011): 17930-17955.

[5] Hu, LiHong, et al. "Combined first-principles calculation and neural-network correction approach for heat of formation." The Journal of Chemical Physics 119.22 (2003): 11501-11507.

[6] Anđelić, Nikola, et al. "Estimation of gas turbine shaft torque and fuel flow of a CODLAG propulsion system using genetic programming algorithm." Pomorstvo 34.2 (2020): 323-337.

[7] Anđelić, Nikola, et al. "Use of Genetic Programming for the Estimation of CODLAG Propulsion System Parameters." Journal of Marine Science and Engineering 9.6 (2021): 612.

[8] Anđelić, Nikola, et al. "Estimation of COVID-19 epidemic curves using genetic programming algorithm." Health informatics journal 27.1 (2021): 1460458220976728.

[9] Anđelić, Nikola, et al. "Estimation of covid-19 epidemiology curve of the united states using genetic programming algorithm." International Journal of Environmental Research and Public Health 18.3 (2021): 959.

[10] Himmetoglu, Burak. "Tree based machine learning framework for predicting ground state energies of molecules." The Journal of chemical physics 145.13 (2016): 134101.