Učenje bez pokušaja: tehnike, pristupi i primjene

Ivanović, Paula

UNIVERSITY OF RIJEKA

FACULTY OF ENGINEERING

Undergraduate University Study of Computing

Final Work

ZERO SHOT LEARNING: TECHNIQUES, APPROACHES, AND APPLICATIONS

Rijeka, September 2024.

Paula Ivanović

0069093498

UNIVERSITY OF RIJEKA

FACULTY OF ENGINEERING

Undergraduate University Study of Computing

Final Work

ZERO SHOT LEARNING: TECHNIQUES, APPROACHES, AND APPLICATIONS

Supervisor: doc. dr. sc. Goran Mauša

Rijeka, September 2024.

Paula Ivanović

0069093498

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

UČENJE BEZ POKUŠAJA: TEHNIKE, PRISTUPI I PRIMJENE

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2024.

Paula Ivanović

0069093498

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 17.03.2024.

Zavod: Zavod za računarstvo

Predmet: Uvod u objektno orijentirano programiranje

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Paula Ivanović (0069093498)**

Studij: Sveučilišni prijediplomski studij računarstva (1035)

Zadatak: **Učenje bez pokušaja: tehnike, pristupi i primjene / Zero shot learning: techniques, approaches, and applications**

Opis zadatka:

Proučiti koncept strojnog učenja bez pokušaja te karakteristični kontekst njegove primjene. Usporediti učenje bez pokušaja, učenje s malim brojim pokušaja te učenje s prijenosom znanja u kontekstu dostupne količine podataka. Objasniti način rada različitih modela strojnog učenja koji koriste učenje bez pokušaja te njegove prednosti i nedostatke. Preuzeti skup podataka za karakteristični problem predviđanja, provesti učenje i testiranje odabranog modela te vrednovati njegove performanse.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Predsjednik povjerenstva za
završni ispit:
prof. dr. sc. Miroslav Joler

Mentor:
izv. prof. Goran Mauša

IZJAVA

Sukladno članku 7. Pravilnika sveučilišta u Rijeci o izradi završnih radova, završnih ispita i završetku preddiplomskih sveučilišnih studija Tehničkog fakulteta, ja, Paula Ivanović, studentica preddiplomskog sveučilišnog studija računarstva, izjavljujem da sam samostalno izradila završni rad.

Rijeka, rujan 2024.

_____

Paula Ivanović

0069093498

ACKNOWLEDGEMENTS

Želim se zahvaliti svojoj obitelji na bezuvjetnoj podršci i vječnom navijanju prije svakog kolokvija i ispita. Također, posebno hvala mojim curama, ženama RITEH-a, veselom kvartetu i brojne druge nazive koje imaju, zbog njih i zbog bezbroj sati učenja na pozivu sam dobila priliku da pišem završni rad. Iskreno hvala svim prijateljima i obitelji, i mom Tysonu, koji su pozorno slušali moje studentske žalbe. Hvala i Bogu na svakoj uslišanoj molitvi prije ispita. Hvala i laptopu i računalu koji su jedva preživjeli ove 3 godine.

Izrazitu zahvalnost zaslužuje i moj mentor doc. dr. sc. Goran Mauša, za podršku i vodstvo tijekom cijelog projekta i završnog rada.

Najveću zahvalu i ljubav zaslužuje moj nećak Dorian, kojega ću za par godina učiti što je zero-shot learning.

CONTENT

1. INTRODUCTION 1
2. ZERO SHOT LEARNING 3
2.1. Phases of zero-shot learning 4
2.2.1. The training phase 5
2.2.2. The prediction phase 6
2.2.3. The evaluation phase 6
2.2. Uses for ZSL 8
3. MACHINE LEARNING TECHNIQUES 11
3.1. K-nearest neighbour 11
3.2. Attribute-based classification and prediction 13
3.2.1. Direct attribute prediction (DAP) 13
3.2.2. Indirect attribute prediction (IAP) 14
3.3. Neural Networks 16
3.4. Support Vector Machines 19
3.5. Transfer learning and domain adaptation 21
3.6. Comparison of learning approaches 22
4. ZERO-SHOT LEARNING MODEL 26
4.1. Case study 26
4.2. Training and testing 27
4.3. Evaluation using Top-1 accuracy 29
5. CONCLUSION 32
REFERENCES 34
FIGURES 38
SUMMARY 39
SAŽETAK 40

1. INTRODUCTION

Intelligence is often defined as the ability to acquire and apply knowledge and skills. Although the definition is controversial because of the broadness the term covers, it is generally accepted that intelligence consists of logic, problem-solving, critical thinking, emotional knowledge, and understanding. Throughout history, humans have discovered intelligence in all parts of nature, from animals and plants to fungi. Discovering intelligence in different sections led to discovering different types of intelligence, such as cellular, spatial, and naturalistic intelligence. With recent technological advances, a new type of intelligence is becoming prominent.

Artificial intelligence (AI) is a term used to refer to intelligence found in computers or other machines. Alan Turing, known as the father of the modern computer, was the first to propose a test of machine intelligence in his research from 1950 titled “Computer Machinery and Intelligence”. Another key figure is John McCarthy, who, along with his colleagues, organised a research project on artificial intelligence in 1956 at the Dartmouth workshop, which is the founding event of artificial intelligence as a field. Since then, artificial intelligence has developed in a variety of industries, along with the development of multiple branches in the field. The main branches consist of computer vision, natural language processing, neural networks and deep learning, machine learning, and robotics. All the branches share the same key aspects of AI, which are learning, reasoning and decision-making, problem solving, and perception. Each aspect allows AI systems to collect and manipulate data and enhance their functions with logical rules, probabilistic models, and decision-making algorithms.

To be intelligent, a system that is in a changing environment should have the ability to learn, [1]. Machine learning is one of the quickest-developing AI branches, which enables systems to learn from experience and improve without being explicitly programmed. It involves algorithms that analyse data, recognise patterns, and make decisions with minimal human intervention. With the provided algorithms, a self-learning model is created and can predict outcomes and classify information. All the predictions are based on trained data that is provided, and using deep data analysis, it becomes possible to recognise patterns. Although a problem arises when the model is faced with unfamiliar data and cannot recognise patterns. When unseen classes show up, the model does not have needed pretraining, and prediction will not be of value. Because of this, over the years, machine learning has developed branches to further improve this type of learning. This thesis will focus on a branch of machine learning, zero-shot machine learning, which provides a solution for this problem.

Zero-shot learning (ZSL) is a method where a model can handle tasks it was not specifically trained for. Using its understanding of the related broader topics, the model can identify or predict new things based on what it already knows. As a newer branch of machine learning, the concept is being explored in different contexts and industries. Currently, zero-shot learning is prominent in natural language processing, for example to classify the sentiment of a sequence of text, [2], computer vision for image recognition, [3], recommender systems for generalisation, [4], and robotics to gain a sequence of actions necessary for the robot to complete the task, [5]. It is applied to numerous industries that have massive amounts of data, such as healthcare, [6], finance, [7], automotive & transportation, [8], media & entertainment, [4], etc. Along with ZSL, there are other branches this thesis will mention.

One of them is few-shot learning (FSL), which enables models to learn and make accurate predictions with only a small amount of labelled training data, [9]. It focusses on generalising from a limited number of examples by using prior knowledge or similar instances to effectively recognise new categories or tasks.

The following thesis provides an extensive review of zero-shot learning, studying its models and approaches. It will also include a comparative analysis of zero-shot learning with few-shot learning and bring out their differences and applications.

2. ZERO SHOT LEARNING

Zero-shot learning (ZSL) is used for building models that can recognise unseen classes based on their descriptions. To be able to recognise an unseen class, this method uses its understanding of similarities between known and unknown categories and makes an educated guess based on learnt attributes. For recognition in ZSL, prior training is not needed, which gives this type of machine learning a promising yet challenging future. Zero-shot learning aims to replicate human intelligence and intuition. The idea behind zero-shot learning is how humans can naturally find similarities between data classes in the same way, training the machine to identify, [9].

Zero-shot learning mostly focusses on image recognition problems where the objects only appear in the central part of the image. However, some problems are more complex, therefore the idea of zero-shot detection. Zero-shot detection is an extension of ZSL that is proposed to simultaneously locate and recognise unseen objects that belong to new categories not encountered during training, [10]. Usually, zero-shot detection is used in more complex scenarios where images contain various objects.

Another subsection of zero-shot learning is Generalized zero-shot learning (GZSL). This extension aims to classify seen and unseen classes simultaneously. ZSL assumes that the model will only encounter samples from unseen classes during testing. This is not practical and does not reflect real-word applications where the model may encounter both seen and unseen classes. GZSL tackles this limitation by allowing the recognition of both seen and unseen classes during the testing phase, [11]. To do this, GZSL leverages the knowledge learnt from seen classes and transfers it to recognise unseen ones, often with the help of semantic embeddings (e.g., attributes, word vectors, or contextual information).

To illustrate the basic concept of ZSL, consider a scenario where a newly discovered animal, for example, a zebra, is described. Although the specific appearance of a zebra is unknown, knowledge of what a horse looks like can be utilized. If a zebra is described as a horse with black and white stripes, it becomes possible to recognise it based on this description. That is the premise of ZSL. This example provides a basic understanding of how zero-shot learning works. The model understands the concepts of 'horse' and 'stripes,' enabling it to make the connection. This process is further illustrated in Figure 2.1, [9]. In this figure, the “Seen Class” refers to categories of data; in the example given, the data consists of images of horses, which the model has been trained on. “Unseen Class” represents categories that the model has not encountered during training, which,

in this case, are images of zebras. Semantic attributes are descriptive features that are associated with each class, which the model uses along with seen classes to process and learn patterns. These attributes are often essential for effectively recognizing unseen classes, although alternative methods that do not rely on them exist. Knowledge transfer plays a crucial role in this process. It refers to the model's capability to apply the knowledge it has gained from the seen classes and generalise it to recognise and understand unseen categories. The model doesn't simply memorise the seen classes; rather, it learns underlying patterns and semantic relationships that can be extended to new, unseen classes. Lastly, interface is the process where the model utilises learnt attributes and knowledge transfer to classify and understand unseen classes.

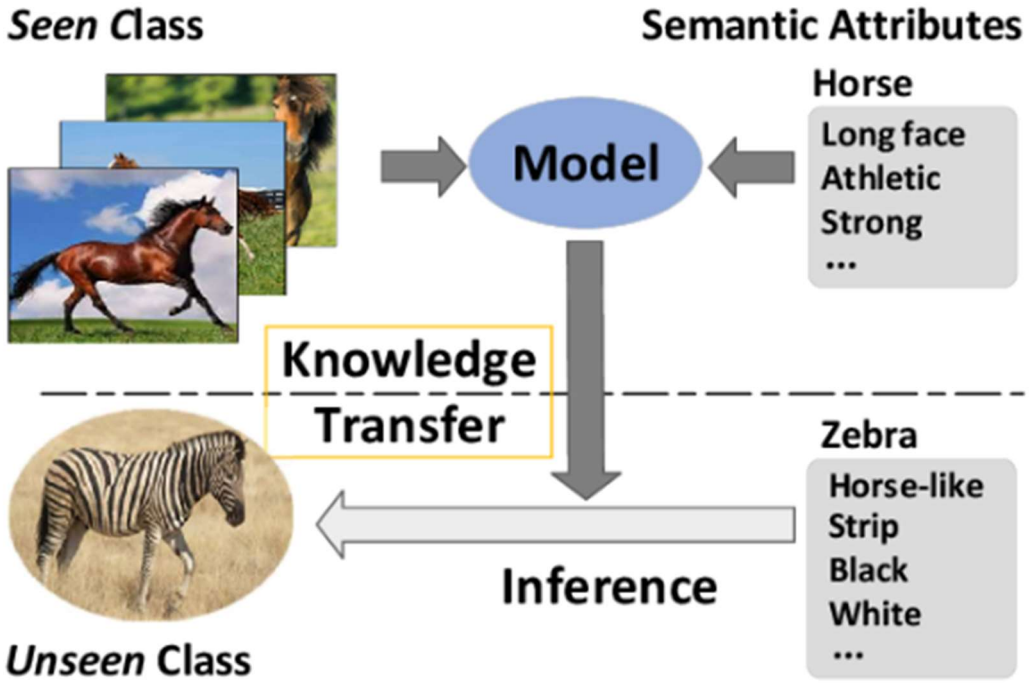

Figure 2.1: *Zero-Shot Learning Overview, [9]*

2.1. Phases of zero-shot learning

For a zero-shot learning model to be effective, it needs to be composed of several key phases. Each phase contributes to the model's overall effectiveness. Figure 2.2, [11], will help provide a better understanding of each phase. The figure depicts a basic ZSL model, which consists of a training phase and a prediction phase. In the training phase, the model uses given data and gains knowledge of the attributes. In the prediction phase or the inference phase, the knowledge is used to categorise

instances of the new class. Together, these phases form the backbone of a zero-shot learning model. The careful design and execution of both the training and prediction phases are critical to the model's ability to perform in real-world scenarios, where the ability to recognise and classify new, unseen objects or classes is highly valuable.

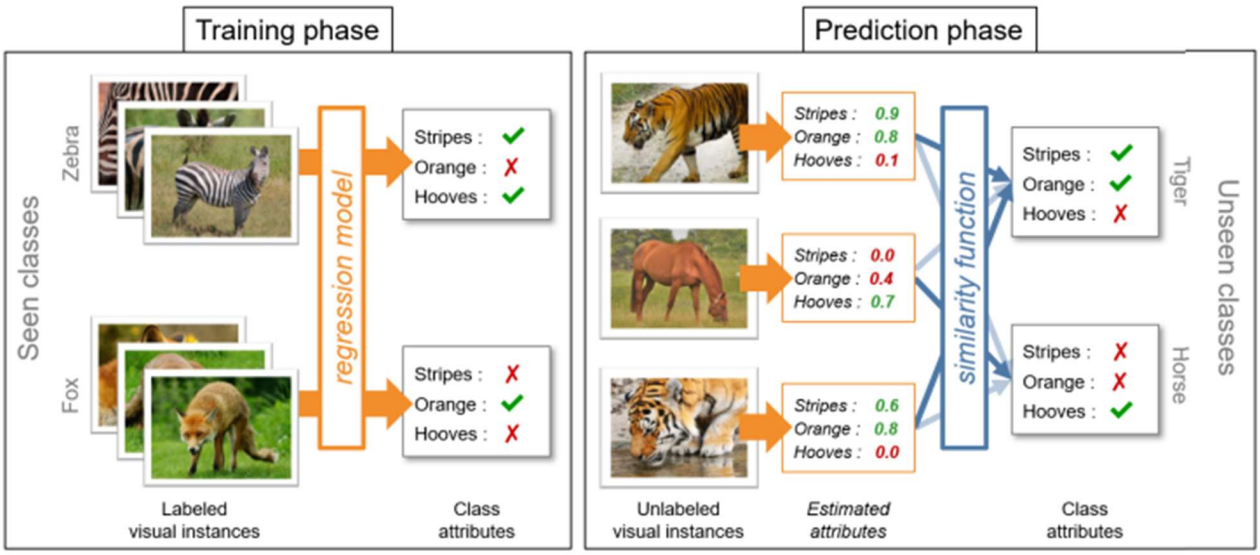

Figure 2.2: *Phases of a basic zero-shot learning model, [11]*

2.2.1. The training phase

The training phase involves learning from a set of labelled examples where classes are known and well-defined, [12]. During this phase, the model learns to map visual features of the input data to a semantic space. In figure 2.2 on the left side, we see examples of seen classes, Fox and Zebra. These are classes for which the model has available labelled data, which consist of a visual example and corresponding attributes. In this case, class Zebra is defined as a class whose objects have stripes, have hooves, and are not orange. With this information, a regression model identifies the relationship between labelled visual instances, images provided, and an class attributes, target or semantic attributes using the seen class. This learnt mapping of relationships allows the model to estimate these attributes for new, unseen images during the prediction phase.

2.2.2. The prediction phase

The model's task in the prediction phase is to classify data into classes not included in the training phase, [12]. Since these classes are unknown to the model, it relies on the semantic attributes learnt during training. In figure 2.2 on the right side, the model encounters unseen classes, images of what is known to us as a horse and tiger. The model estimates the attributes for these unknown classes. As shown in the figure, the tiger is estimated to have “stripes” with a high probability of 0.9 and “orange” with 0.8, but a low probability for “hooves” with 0.1. Using a similarity function, the model compares the estimated attributes of unseen classes with the attributes of seen classes. This comparison helps the model determine which unseen class is most similar to the seen classes, enabling it to make accurate predictions by leveraging the learnt relationships between seen and unseen classes. For example, the tiger's attributes (high likelihood of "stripes" and "orange") closely match the attributes of the unseen class "Tiger," enabling the model to classify the image correctly.

2.2.3. The evaluation phase

Often an optional third phase is implemented, the evaluation and refinement phase. This phase is not depicted on figure 2.2, but it would follow the prediction phase. In it, the model’s performance is assessed. Based on the evaluation results, the model may undergo refinement, which could involve adjusting the semantic space, improving the similarity function, or fine-tuning the model to enhance accuracy and robustness. This phase is crucial for optimising the model's ability to correctly classify unseen classes while minimising errors.

In Zero-Shot Learning (ZSL) and Generalized Zero-Shot Learning (GZSL), evaluation methods are critical to understanding the effectiveness of the models. There are basic evaluation methods used, like classification accuracy and per-class accuracy. Although GZSL is an extension of the ZSL, it uses different evaluation methods.

Classification accuracy measures the accuracy of different components. There are two main branches: micro-averaging and macro-averaging, [13]. Firstly, an understanding of precision and recall is needed. Precision is a measure of result relevancy; it shows how often a machine-learning model is correct when predicting the target class. Recall is a measure of how many truly relevant

results are returned, it shows whether a machine-learning model can find all objects of a target class. Micro-average accuracy measures the overall accuracy across all test instances, all true positive, false positive and false negative predictions across classes are summed up and precision and recall are calculated jointly, [13]. The equations are show in equation 2.1 and 2.2.

$$Precision = \frac{TP_1 + TP_2 + \cdots TP_N}{TP_1 + FP_1 + TP_2 + FP_2 + \cdots TP_N + FP_N} \quad (2.1)$$

$$Recall = \frac{TP_1 + TP_2 + \cdots TP_N}{TP_1 + FN_1 + TP_2 + FN_2 + \cdots TP_N + FN_N} \quad (2.2)$$

where:

TP stands for True Positive cases,

FP stands for False Positive cases,

FN stands for False Negative cases,

N stands for the number of classes.

Macro-average accuracy measures the average accuracy per class; this is especially important in ZSL when classes are unbalanced. Macro-average accuracy is used to ensure that the model performs well across all classes rather than just a few dominant ones. This method prevents the model from being biased toward classes with more instances. Simply described, first the metric is measured by class, then it is averaged across classes, [13]. The basic equations for this method are shown in equation 2.3 and 2.4.

$$Precision = \frac{P_1 + P_2 + \cdots P_N}{N} \quad (2.3)$$

$$Recall = \frac{R_1 + R_2 + \cdots R_N}{N} \quad (2.4)$$

where:

P stands for Precision of a class,

R stands for Recall of a class,

N stands for the number of classes.

Evaluation in GZSL is usually done through two main categories: Unseen accuracy and Seen accuracy. Unseen accuracy presents the accuracy on unseen classes when the candidate classes include either both seen and unseen classes or only the unseen classes. Even though there are no examples from unseen classes provided during training, the model uses shared attributed (or other forms of knowledge transfer) to make predictions about these classes. The accuracy is evaluated on the unseen classes during testing, which the model has inferred based on the learned attributes from the seen class. Seen accuracy presents the accuracy of seen classes when the candidate includes either both seen and unseen classes or only the unseen classes. These categories can be used in evaluation methods such as the harmonic mean, which is a more balanced measure that penalises models that perform well on seen classes but poorly on unseen classes, or vice versa.

2.2. Uses for ZSL

Zero-shot learning provides significant advantages in versatility and efficiency for industries managing large datasets, dynamic environments, or limited traditional training data. By enabling models to predict tasks and recognise categories they have not been explicitly trained on, ZSL proves invaluable across various industries.

One of the industries that uses ZSL is retail and e-commerce. The main use of this concept is enhancement of customer experience and optimisation of product discovery and recommendation. Retailers often use ZSL to recommend new products to customers based on their preferences, even if the model hasn't seen the products prior. Performance of recommender systems (RS) relies heavily on the amount of training data available; on the other hand, zero-shot learning promises some degree of generalisation from an old dataset to an entirely new dataset, [4]. $ZESR_{EC}$ is a dubbed zero-shot recommender algorithm that is trained on an old dataset and generalized to a new one where there are neither overlapping users nor overlapping items, [4]. With this it is possible to successfully recommend items in the zero-shot setting, and it is possible to learn interpretable user behaviour patterns, which can be generalized across datasets.

Another example of ZSL use is found in the automotive industry. With the rise of self-driving cars, ZSL is needed for recognition and proper reaction to new or unusual objects without needing extensive labelled data for each new type of object. Here ZSL contributes to user safety. A model can be used that can detect objects under limited data and has strong open-set detection and zero-

shot generalisation capabilities. GroundingDINO is a zero-shot object detection model based on image-text cross-attention mechanism to corner case detection in autonomous driving. Using predefined text prompts a model is guided to achieve zero-shot object detection even if they are not present in the training data, [8]. An image backbone is used for image feature extraction, a text backbone is used for text feature extraction and a feature enhances is used for fusing these features. A language-guided query selection module starts the queries, and a cross-modality decoder improves the predicted bounding boxes using the combined features, [8].

The most popular industry where ZSL is found is technology and software development. Mostly, ZSL models can be found in Natural Language Processing, or NLP, which is used in chatbots and virtual assistants to understand and respond to user questions and tasks they have not been trained for.

Most importantly, in the healthcare sector, ZSL has shown a promising future in enhancing diagnostic capabilities and simplifying drug discovery. Evolution of drug-resistant microbial species is one of the major challenges to global health, [6].

AMP0, [6], is a zero- and few-shot learning model that predicts the peptide activity. Antimicrobial peptides have different biological activity against microbes (bacteria, viruses, and fungi). Because of the intense experimenting needed to gain knowledge on the possible antimicrobial peptides that can affect microbes, we can use zero-shot learning to create baseline predictions. To gain the most optimal results, a large database for testing is needed. The DBAASP is a database that offers information about the antimicrobial activity and structure of peptides. This database contains around 12984 peptide sequences. There are some attributes that are necessary to be able to make a prediction. One of these attributes is MIC, a number that represents the minimum inhibition concentration against different targeted microbe species. This number defines whether the peptide is antimicrobial (<25) or non-antimicrobial (>1000). To create a prediction, the AMP0 uses vectors for representing a peptide sequence attributed to the microbial species, a target variable that represents the MIC of the peptide sequence against the species, a learnable parameter, and the prediction problem. The prediction problem can simply be defined as a mathematical function between the vector for peptide sequence, the vector attribute for microbial species, and the learnable parameter, [6]. The baseline models used are Radial Basis Function SVM, Gradient Boosted Tree classifier (XGBoost), neural networks, and k-nearest neighbour classification. The basic ZSL scheme used is inspired by the model Romera Parades & Torr. A joint feature representation is constructed by concatenating peptide and species features, where a corresponding label would be set on -1 for non-antimicrobial and +1 for antimicrobial. For each model

hyperparameters were optimized through cross-validation. To test different models a percentile-wise Rank Frequency Per Peptide (RFPP) is used to predict peptide-target species interactions. A lower RFPP score indicates better prediction accuracy, this is shown in figure 2.3. For instance, the random classifier, which generates random predictions, has a median RFPP of 75, meaning that for 50% of peptides, the correct target species is within the top 75 out of 336 predictions. In comparison, XGBoost and SVM perform significantly better, with median RFPPs of 50 and 9, respectively. However, the proposed ZSL model outperforms all other methods, achieving an RFPP of 1.0 at the 75th percentile, indicating that for 75% of peptides, the model's top prediction is correct, [6].

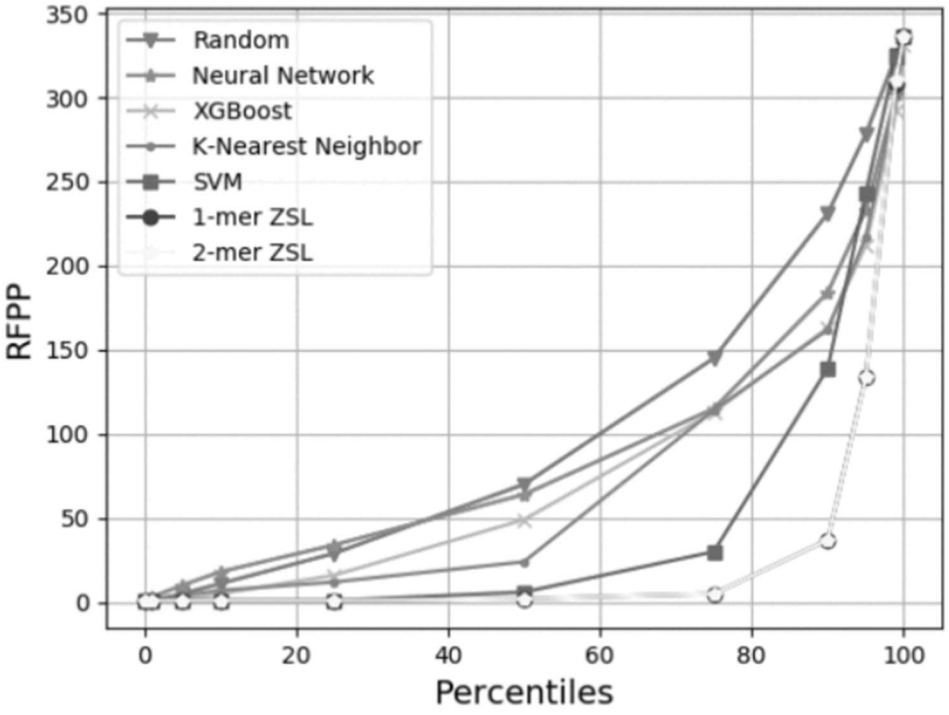

Figure 2.3*: Percentile-wise RFPP Scores, [6]*

AMP0 offers prediction of effectiveness of certain peptides against a new microbial species by modelling the microbial species as a class represented by the attribute vector of genomic sequence. Using zero-shot learning, we can encounter unseen classes and still give satisfactory predictions without the need to do an experiment. This reduces the cost and can further speed up a new biological discovery.

3. MACHINE LEARNING TECHNIQUES

Since ZSL has a wide range of uses and capabilities, there are different techniques used for classification and prediction. They are methods commonly used in zero-shot learning to tackle the problem of learning to recognise unseen classes or tasks without any training examples. Each of these techniques offers different approaches to address the challenges of zero-shot learning, such as leveraging nearest neighbour methods, attribute-based classification, neural networks, semantic word networks, and transfer learning/domain adaptation strategies.

3.1. K-nearest neighbour

One of the fundamental and intuitive methods of classification is k-nearest neighbour (kNN). Although this method is not designed for ZSL models, it is the easiest to understand. Understanding this method is needed because it provides insight into how similar methods might be adapted, since this is the most intuitive way of thinking. The principle is straightforward; a sample is classified based on the majority class among its k-closest neighbours.

The similarity between classes is typically determined by Euclidean or Hamming's distance. Euclidean distance, in Euclidean space, is defined by the length of a straight-line segment that would connect two points, [14]. Meanwhile, Hamming's distance measures the number of differing bits between two binary strings or vectors of equal length, which can be useful in specific contexts like binary attribute comparisons. For example, the difference between 1000 and 0110 is three. This distance can be portrayed by a 4-bit binary tesseract, as shown on figure 3.1, where the shortest path, the least number of lines between each point, is the distance between them, [15].

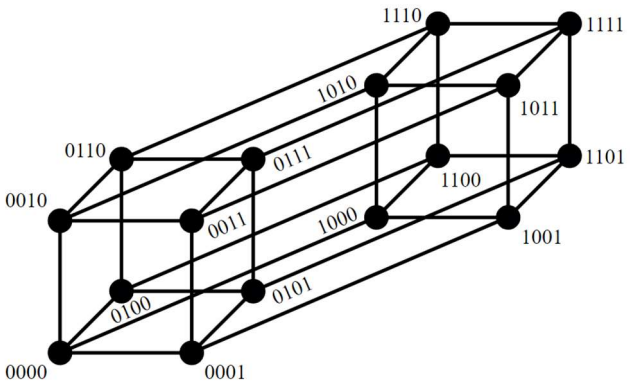

Figure 3.1*: Phases of a basic zero-shot learning model [15]*

To assess the performance of kNN, various methods are used, one of which is cross-validation. Cross-validation, or k-fold cross-validation, helps estimate the model's accuracy by splitting the data into multiple subsets and evaluating performance iteratively. An example of a basic KNN algorithm is shown in figure 3.2, [16]. Where the algorithm classifies a test sample by choosing among its k-closest neighbours in the training set. K-Fold Cross-validation is used to evaluate the model by dividing the dataset into k folds, equally sized partitions, then iteratively training and testing the model across those folds. Each fold is used as a test set once and as part of the training set k-1 times.

```
begin
  initialize distance matrix D and confusion matrix C
  set TotAcc ← 0, NumIterations to desired iterations

  calculate all pairwise distances and store in D

  for t ← 1 to NumIterations do
    set C ← 0, ntotal ← 0
    partition data into κ folds

    for fold ← 1 to κ do
      set current fold as test set, others as training
      set ntotal ← ntotal + ntest

      for i ← 1 to ntest do
        find k nearest neighbors of test sample xi
        determine most frequent label ω^ among neighbors
        update confusion matrix C for true label ω and predicted label ω^

      calculate accuracy Acc = Σ(cjj) / ntotal
      update TotAcc = TotAcc + Acc

  calculate AvgAcc = TotAcc / NumIterations
end
```

Figure 3.2 Pseudocode for k-Fold Cross-Validation

In zero-shot learning, the concept of nearest neighbours can be extended to compare features or embeddings of unseen classes with those of seen classes. For instance, if classes are represented by semantic attributes or embeddings, the task of classifying an unseen instance involves finding the nearest known class representation and assigning the class label based on the closest match. The main problem in this method is the weight of the attributes. Since the attributes are represented through a binary vector and are compared using Hamming's distance, all attributes have the same importance, which is not applicable to the real world. Zero-shot learning requires advanced

techniques to weigh attributes appropriately and ensure that the similarity measures account for their varying degrees of importance. A basic example of kNN use in zero-shot learning can be found in the paper by Weijia Shi, Julian Michael and others titled “kNN-Prompt: Nearest NEighbor Zero-Shot Inference”, where a kNN-Prompt can enable zero-shot learning by incorporating information from a large, unlabeled datastore of text, [17]. The key idea is to leverage k-nearest neighbors to make predictions based on similar contexts seen before, even without explicit examples for a given task.

3.2. Attribute-based classification and prediction

A property of an object is an attribute if a human has the ability to decide whether the property is present or not for a certain object, [18]. This definition contains properties that are visible as well as those that are not directly visible but are related to visible information. Attribute-based classification offers a solution for object classification with disjoint training and test classes.

Assume the situation of learning with disjoint training and test classes. If for each class $z \in Z$ and $y \in Y$, an attribute representation $a^z, a^y \in A$ is available, then we can learn a nontrivial classifier $\alpha: X \to Z$ by transferring information between Y and Z through A, [18]. Where A represents a set of attributes, Z and Y represent classes and X represents the input space, set of all possible feature vectors or instances that need to be classified. There are two generic methods for attribute-based classification: direct attribute prediction and indirect attribute prediction, [18]. These methods are prominent approaches within probability frameworks in which the predictions obtained in the first phase can be combined to determine the most likely target class. Both approaches employ probabilistic classifiers but differ in their focus during the learning and inference stages.

3.2.1. Direct attribute prediction (DAP)

Direct attribute prediction (DAP) offers the possibility to take the weight of the attributes into consideration. The idea is to independently predict each attribute's presence or absence for a given instance. During the testing phase, a probability classifier is learnt for each attribute. Then, in the prediction phase, the prior defined classifier is used to determine new classes. When a new instance is encountered, the model uses the classifier to predict the probability of each attribute being

present in that instance. Then predicted probabilities are combined to infer the most likely class by comparison with the knowledge gained during training. This method is illustrated in figure 3.3, in which the dark grey nodes represent components that are always observed, and the light grey nodes are observed only during the training phase, [18]. White nodes need to be predicted and have not been observed. Thick lines represent class-attribute relations, and the parameters β are learnt from attributes.

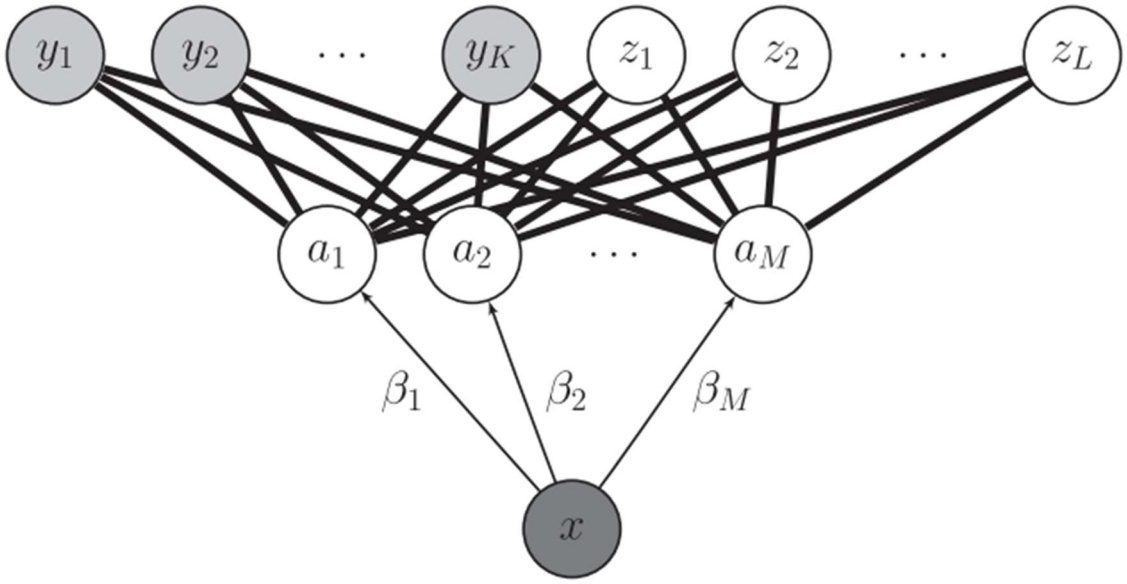

Figure 3.3: *Direct Attribute Prediction (DAP), [18]*

This method's versatility in handling various attribute and class types is one of its main advantages. The classes during testing do not need to be the same as during training, which makes this method a suitable candidate for ZSL. However, there is a limitation because this method treats each attribute as independent, which does not reflect real-world correlations between attributes. For example, attributes like “has fur” and “is mammal” are likely correlated, but DAP does not account for these correlations, leading to suboptimal predictions.

3.2.2. Indirect attribute prediction (IAP)

Indirect attribute prediction (IAP) is similar to DAP; instead of learning a probability classifier for each attribute, it learns a probability classifier for each class. IAP, depicted in figure 3.4, also uses

the attributes to transfer knowledge between classes, but the attributes form a connecting layer between two layers of labels: one for classes that are known at training time and one for classes that are not, [18]. In this framework, the model is trained to predict the likelihood of a given instance belonging to a specific class based on the attributes associated with that class. During training, IAP learns the relationships between attributes and classes for the seen data. For each class, a probabilistic model is built that captures the likelihood of observing certain attributes. During inference, when a new instance is encountered, the model predicts the probability that the instance belongs to each unseen class based on the predicted attributes.

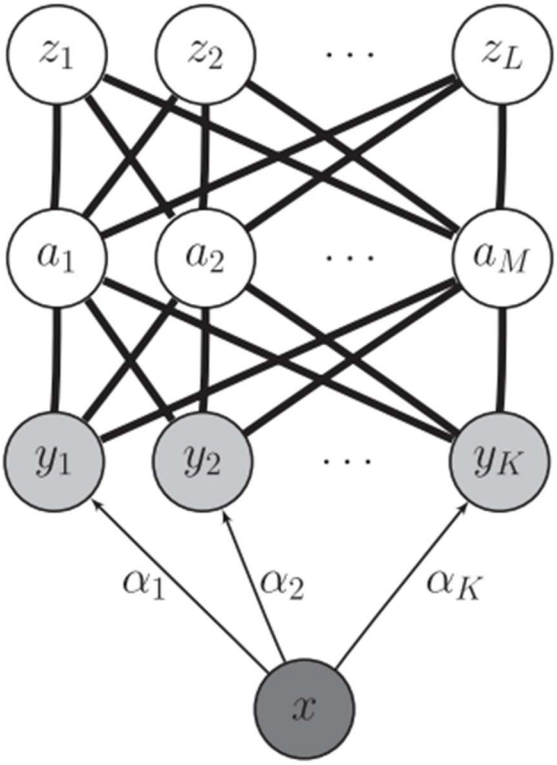

Figure 3.4 *: Indirect Attribute Prediction (IAP), [18]*

Unlike DAP, this method considers the correlations between attributes. For example, IAP can better model the relationship between correlated attributes like "has fur" and "is mammal," leading to more accurate predictions. However, IAP can be computationally expensive because it requires learning and evaluating a classifier for each class during inference. Despite this, IAP is a powerful method, especially for scenarios where attribute correlations are significant, making it a strong candidate for Zero-Shot Learning (ZSL), where the classes during testing differ from those seen during training.

3.3. Neural Networks

Neural Networks are quantitative models linking inputs and outputs adaptively in a learning process analogous to that used by the human brain, [19]. These networks are designed to model complex, non-linear relationships between input data (such as images, text, or other features) and output labels. Neural networks consist of layers of interconnected nodes (neurones), each of which processes input data and passes it on to the next layer. The first layer is called the input layer, and it receives the raw input signals. The final layer is the output layer, which produces a prediction based on the processed data.

Most neural networks use various functions, such as the activation function and logistic functions. The activation function receives an input signal and produces an output signal after a certain threshold value. If the input value is below the threshold, the output will be zero; only when the input value is above the threshold will a non-zero output value appear. That is the premise of a step function. Typically, neural networks use sigmoid functions, which are smoother and do not have a significant jump. These functions are also called logistic functions. A graph comparison between these functions can be seen in figure 3.5, [20].

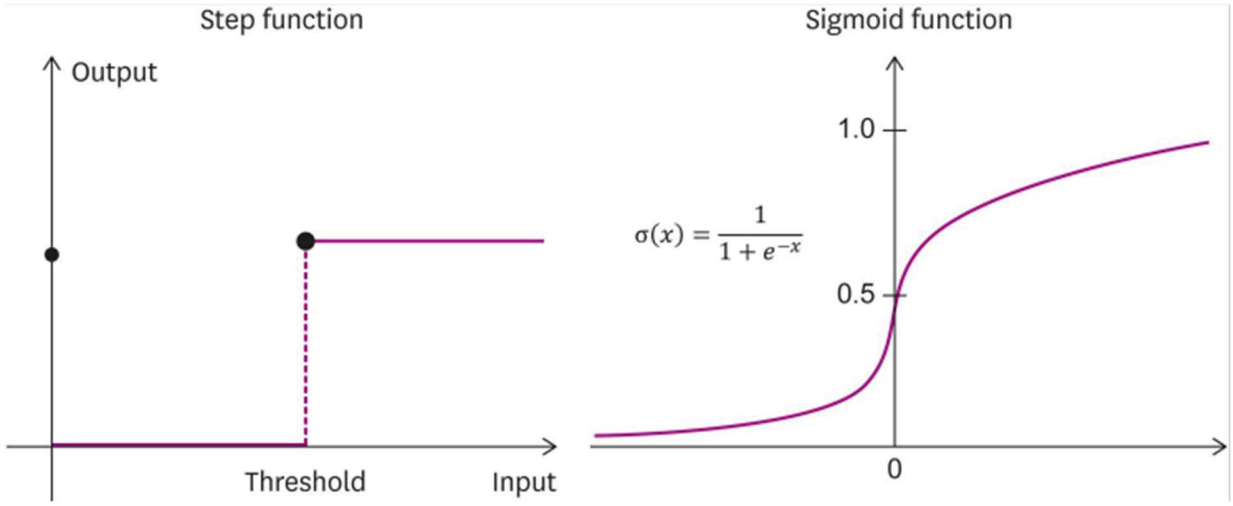

Figure 3.5 *Step function (left) and Sigmoid function (right), [20]*

The use of sigmoid functions is dominant because neural networks often use a weight modification method during the learning process. Once the weight is modified, the whole layer requires a

differentiated activation function. The basic sigmoid function can be expressed with the following equation:

$$\sigma(x) = \frac{1}{1+ e^{-x}} \qquad (3.1)$$

where:

$\sigma(x)$ represents the simoid function with x as the input,

e is Euler's number, approximately equal to 2.71828.

The learning process of an artificial neural network involves updating the connection strength (weight) of a node (neuron), [20]. Nodes receive multiple inputs, which get processed by the sigmoid function, and the result is the output value. The updating of the weights is determined by the error between the predicted output and the correct output. By using the error between the predicted value and the correct, the weight in the network is adjusted so that the error is minimised and an output close to the truth is obtained.

This concept of neural networks is similar to biological neural networks, which consist of neurones, therefore making this method a ground-breaking way of replicating the human mind in machine form. Similarities between biological and artificial neural networks are shown in figure 3.6.

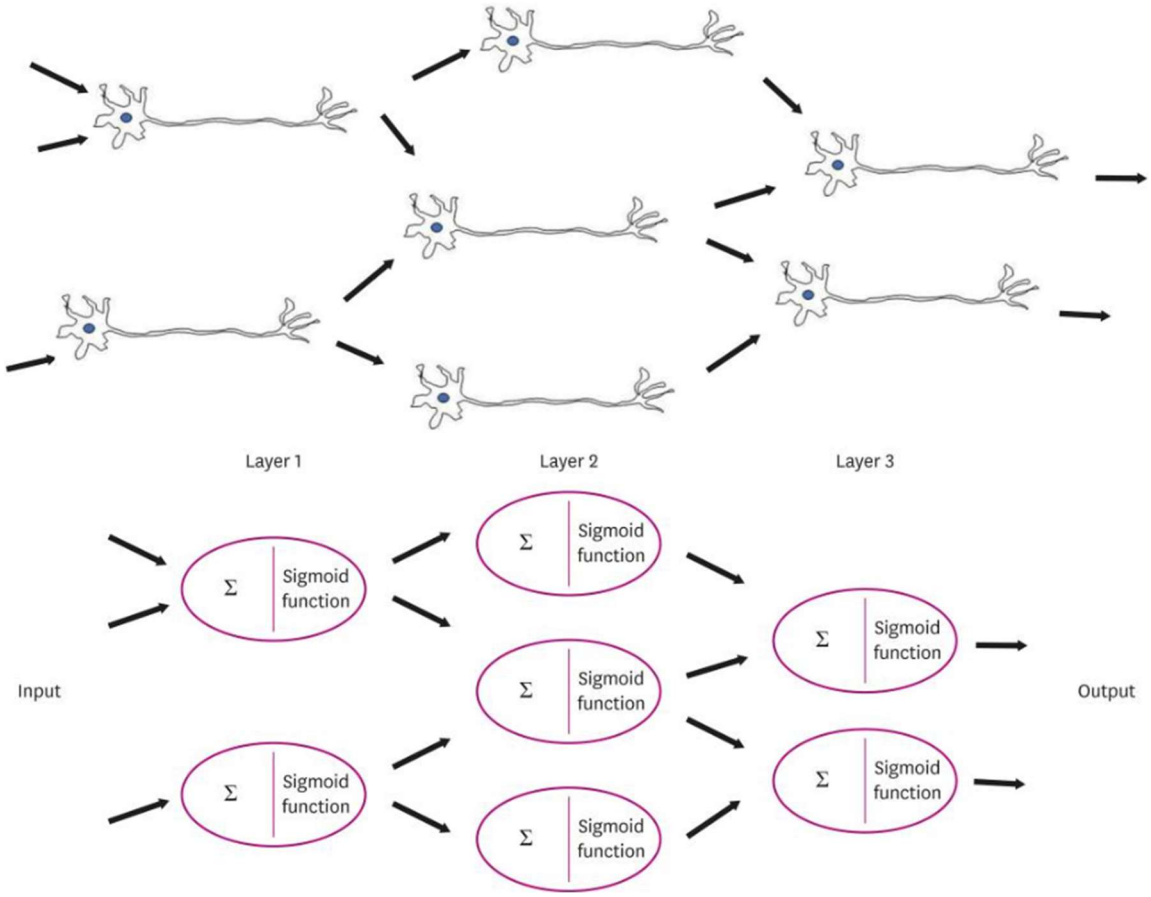

Figure 3.6 *Biological neural network (top) and multi-layer perception in artificial neural network (bottom), [20]*

In ZSL, neural networks are often used for their ability to learn high-level feature representations that can generalise well to unseen classes. These networks can be trained to map raw input data to a semantic space where each dimension corresponds to an attribute. Once trained, the network can be used to predict the attribute signatures of unseen classes, which can then be compared to the known attribute signatures to classify new instances.

The main key advantage of neural networks is their adaptability. They are adaptive in that they can learn to estimate the parameters of some population using a small number of exemplars (one or a few) at a time, [19]. However, neural networks are often seen as “black boxes” because it can be difficult to interpret how they arrive at a particular prediction; their decision-making process can be opaque.

3.4. Support Vector Machines

Support Vector Machines (SVM) are a set of related methods for supervised learning, applicable to both classification and regression problems, [21]. The key idea behind this method is to maximise the margin between the closest data points of each class, known as support vectors, and the decision boundary.

This means that SVM is finding the optimal boundary, a hyperplane, which separates different classes in the feature space. A hyperplane is a decision surface that divides the space into two parts, each containing examples from one of the classes. The margin is the minimal distance between the hyperplane and the closest data point from any class. This margin should be maximised because it implies better generalisation ability, thus reducing the risk of misclassifying. The SVM algorithm selects the hyperplane that maximises this margin, known as the maximum-margin hyperplane, using the quadratic programming optimisation problem, [21]. A quadratic programming (QP) optimisation problem is a type of mathematical optimisation problem that involves a quadratic objective function and linear constraints. It is a special case of optimisation where the objective function is quadratic and the constraints on the variables are linear.

Consider a simple scenario where we have two classes of examples in a multidimensional space. If these classes are well-separated, a reasonable hyperplane would lie somewhere between the means of these two classes. The hyperplane can be seen as a geometric comparison between the angles formed by vectors connecting new data points to the cluster means. This comparison can be computed using the dot product of vectors. A basic linear Support Vector Machine model is shown in figure 3.7, where the axes of the graph represent the features of the data points, [22]. Blue circles represent one class, and red squares represent another class. The hyperplane is the decision boundary identified by the model, shown in figure 3.7 as a solid, thick black line. The area between the dashed lines is the margin, which is meant to be maximised in order to ensure that the closest data points are as far away from the hyperplane as possible. The arrow labelled ‘w’ indicates the weight vector, which is perpendicular to the hyperplane and plays a crucial role in defining its orientation, [22].

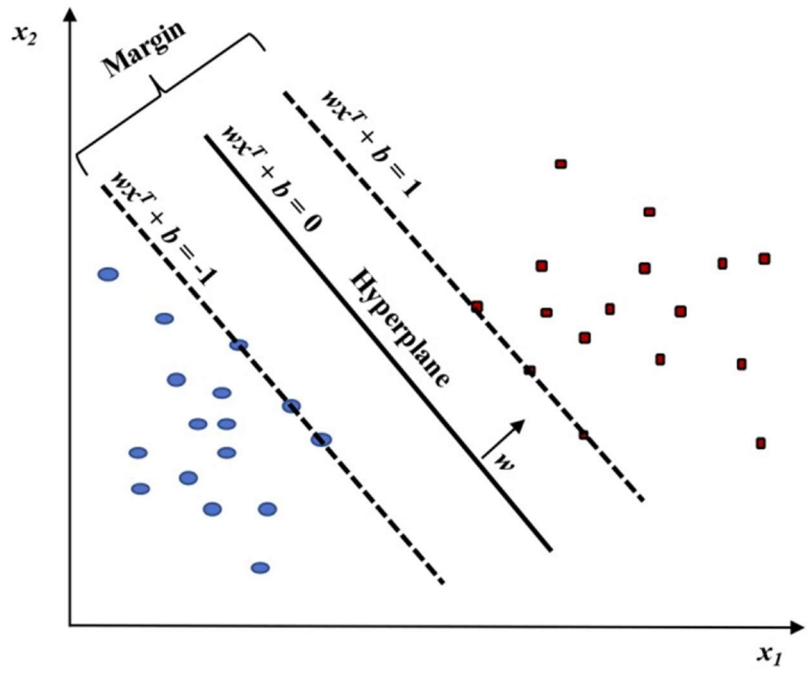

Figure 3.7 *Linear SVM model. Two classes (read versus blue) were classified, [22]*

It is important to mention that SVM as a classifier has been used in cancer classification since the high-throughput microarray gene expression data was available in the early 2000's, [22]. SVMs are particularly useful in ZSL because they can handle high-dimensional data and are effective at finding the decision boundary even in cases where the classes are not linearly separable. This is achieved by kernel functions, which map the input data to a higher-dimensional space where a linear decision boundary can be found. Common kernel functions include the Radial Basis Function (RBF) kernel and the polynomial kernel. An example of a kernel function is depicted in figure 3.8. Where data cannot be separated by linear SVM, it can be transformed and separated by a kernel function, [22].

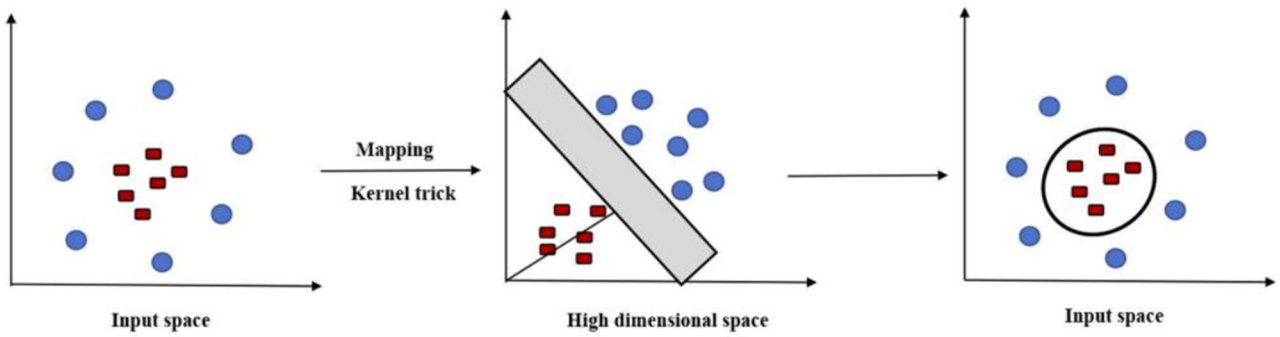

Figure 3.8 *Kernel function, [22]*

This ability to handle linear and non-linear classification makes SVM versatile tools in ZSL. However, like all the methods previously mentioned, SVMs have limitations. They can be sensitive to the choice of kernel function and hyperparameters, and finding the optimal kernel for a specific problem often requires extensive experimentation and cross-validation. Additionally, SVMs can be computationally expensive for large datasets, particularly when using non-linear kernels.

3.5. Transfer learning and domain adaptation

Transfer learning is a critical technique in ZSL, enabling models to leverage knowledge gained from previously seen classes to classify unseen classes. The basic idea behind transfer learning, widely referred to as "learning how to learn", is to reuse a model trained on one task (the source task) to perform well on a different but related task (the target task). This is done by extracting knowledge from a set of input tasks, which is later used for future tasks. It is particularly useful in ZSL, where the model must classify classes for which no labelled training data is available.

For example, a model performing zero-shot text classification might use a transformer-based model like BERT, already pre-trained on a massive corpus of language data, to convert words into vector embeddings. Likewise, a zero-shot image classification model might repurpose a pre-trained convolutional neural network (CNN) like a ResNet or U-Net, as it will already have learnt filter weights conducive to identifying important image features that could inform classification, [23].

The advantage of transfer learning in ZSL is that it allows models to generalise to new classes with minimal additional training, reducing the need for large amounts of labelled data. However, transfer learning also has limitations. The performance of the model on the target task is highly dependent on the similarity between the source and target tasks.

Zero-shot learning (ZSL) can be considered a special case of transfer learning where the source and target domains have different tasks/label spaces and the target domain is unlabelled, providing little guidance for the knowledge transfer, [24].

Domain adaptation is closely related to transfer learning but focusses on adapting a model trained on one domain (the source domain) to perform well on a different domain (the target domain). In domain adaptation, the input distributions of the source and target domains are different, but the

task or function remains the same. In transfer learning, the input distribution in both source and target tasks is supposed to be the same, whereas there are several labelling functions. Domain adaptation makes the reverse assumption; the learning function is the same but the input distributions for source and target tasks are different, [25].

Domain adaptation is a powerful tool in ZSL because it allows models to generalise across different domains, making them more robust to variations in the data. However, it also introduces additional complexity, as the model must learn to adapt to new domains without compromising its performance on the source domain. This can be challenging, especially when the domains are significantly different or when there is limited data available for the target domain.

3.6. Comparison of learning approaches

In situations where collecting large datasets is impractical, choosing a suitable machine learning approach and strategy is necessary and one of the crucial decisions. Among these strategies, the methods of zero-shot learning, few-shot learning, and transfer learning stand out as significant. Each approach has its merits and limitations, particularly when evaluated in the context of the available data. This part of the thesis will provide a general overview of the three learning approaches and a general comparison.

As previously mentioned, zero-shot learning, also known as learning without attempts, is the ability of a model to make predictions on new and unseen data. This method is usually based on generalisation and is used in situations where data is limited or non-existent. Although it has major advantages considering predictions can be made without prior data for the class or task, ZSL is highly dependent on the quality of pre-existing knowledge. If the foundational data is insufficient or lacks diversity, the model's prediction may be inaccurate, inadequate, or overly generalised.

On the other hand, few-shot learning, also known as learning with a small number of attempts, enables AI models to learn from a small number of examples, similar to a quick learner mastering a new skill with minimal practice, [26]. This method involves training a model with a limited amount of data specific to the task. Unlike zero-shot learning, during the training phase some data is needed in order to provide a prediction; for images, it is usually less than 10 samples. This approach strikes a balance between the need for task-specific data and the ability to generalise from minimal examples. Few-shot learning is particularly effective when data collection is expensive or when new categories emerge that were not present during the initial training phase.

This method has advantages in situations where data is limited but still available in small quantities during the training phase.

Figure 3.9 illustrates a FSL model focused on feature extraction for classification tasks. The model is trained on a large, labelled dataset, and then a feature extractor processes the input data, transforming raw images into relevant features that facilitate classification, [27]. A query set contains new images that need classification. By employing a similarity function to compare the features of the query images with those of the support images, the model generates predictions for the query set based on their proximity to the learnt examples.

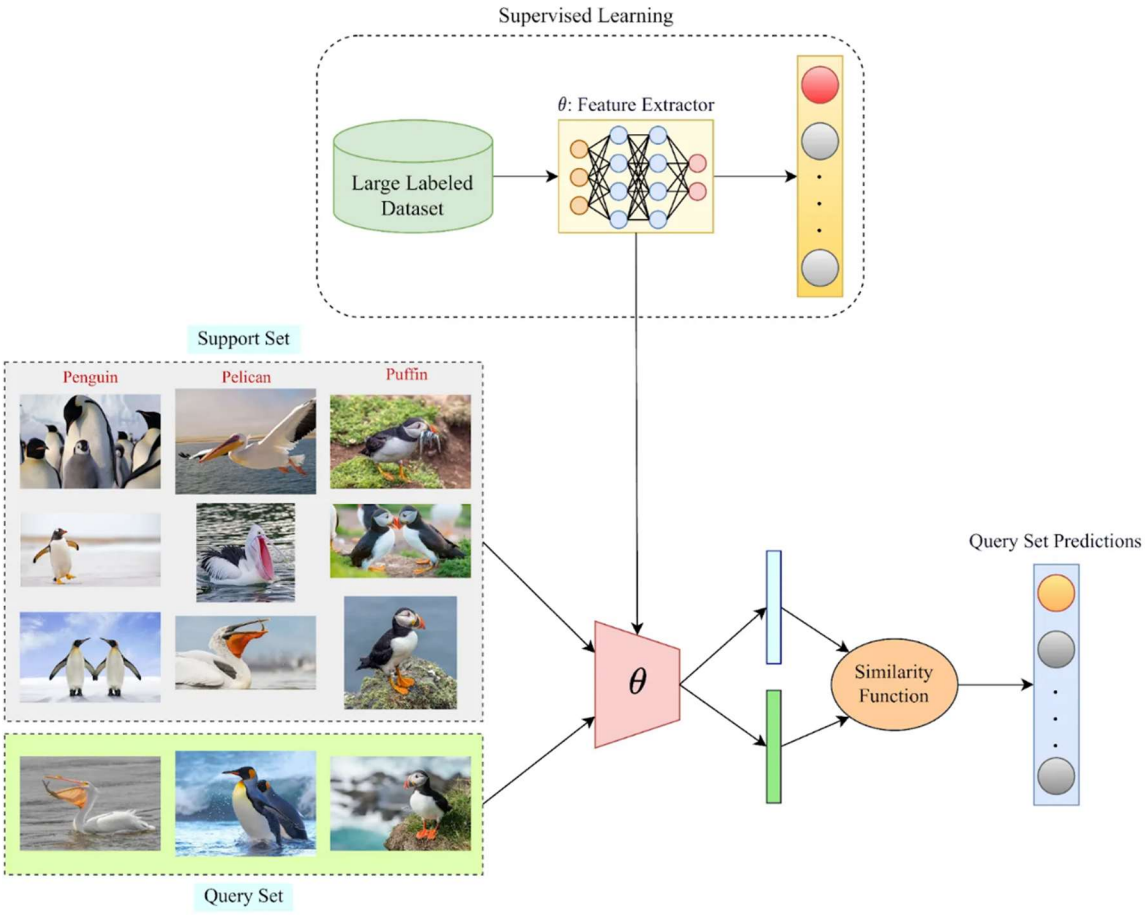

Figure 3.9 *General overview of a Few-Shot Learning framework, [27]*

A common use for both of these methods is Natural Language Processing (NLP). Where in the dynamic domain, Few-Shot Learning equips models to generalise from minimal examples, while Zero-Shot Learning enables comprehension and generation on entirely novel tasks, [28]. Both methods have unique advantages and applications.

ZSL proves to be useful in social media semantic analysis, where an unprepared model might suffer because there is no training data for this task. ZSL can analyse language used on social media, like emoticons and slang, and provide insights into public sentiment without needing prior training. Sentiment analysis is made agile and flexible by zero-shot learning, which demonstrates its ability to interpret the constantly changing fabric of human expression on social media. On the other hand, there are some challenges when applying zero-shot learning in NLP. ZSL may encounter challenges when dealing with tasks of a high degree of domain specialisation or intricate knowledge; in this case, a broad understanding of language will not be enough. Along with that, ZSL may struggle to distinguish between sarcasm and genuine text as it lacks explicit training on such subtleties. Due to the model's heavy reliance on the pretraining data, its efficacy may be hampered when used on a task requiring uncommon or developing linguistic phenomena if there are not enough pertinent instances in the pretraining data. Unlike ZSL, few-shot learning may provide more precision and accuracy since the training data will involve at least one instance of data or task.

Advantages of FSL are highlighted in scenarios where acquiring extensive labelled data is resource intensive or economically impractical and when there is a need for rapid adaptation. In essence, the strategic use of few-shot training in NLP tasks is driven by its ability to offer efficiency, rapid adaptability, and versatility, [28]. Few-shot learning faces challenges in effectively generalising from a small number of examples, particularly when those examples are of low quality or lack diversity. This can lead to difficulties in capturing complex tasks, handling ambiguity, and dealing with rare events, potentially compromising model performance. For example, in text summarisation, if the few-shot examples predominantly represent a specific writing style, the model may produce biassed or non-generalisable summaries when applied to diverse writing styles.

Along with these two methods, a common type of learning is transfer learning. Transfer learning, also known as learning with knowledge transfer, stands out as the most versatile approach, particularly when there is a wealth of data in related domains but limited data for the specific task at hand. Transfer learning allows a model to leverage the abundant data from the source domain to improve its performance on the target task, even with minimal target-specific data. This approach is highly effective in accelerating learning and improving accuracy, as the model can reuse previously acquired knowledge rather than starting from scratch.

To choose the right machine learning model, it is important to determine the problem, understand available data, evaluate multiple models, and consider computational resources. The model you

select will have a significant impact on the insights you derive from your data and ultimately determine the usefulness of a project, [29]. The type and quality of data are crucial for model selection; clean and pre-processed data improve model performance. Defining the problem and setting measurable goals eases the choosing process. Zero-shot learning should be used for predictions that need to be on tasks or classes that the model has never encountered before. When there is a small number of labelled data for a new task or class but there is generalisation needed, the ideal approach is few-shot learning. Transfer learning should be used when there is a large, pre-trained model that needs to be applied to a related but different task with limited data. This method is effective when the new task benefits from the knowledge the model has already acquired.

4. ZERO-SHOT LEARNING MODEL

The zero-shot learning model, [30], that will be used is based on two research papers: “Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly” (ZSLGBU), [31], and “An embarrassingly simple approach to zero-shot learning” (EsZSL), [25]. This model integrates ZSL techniques with residual networks, particularly ResNet, to perform tasks across various datasets. The model’s description, datasets utilised, phases, and evaluation will be discussed, along with the underlying Python code.

4.1. Case study

The model, [30], uses residual networks and attribute-based learning on a set database. Residual Network (ResNet) is a deep neural network architecture designed for improving the training of very deep networks. The core idea is the usage of residual, skip, connections, which allow the network to bypass certain layers. Therefore, it enables the model to train much deeper networks without suffering from the complexity that commonly occurs in neural networks. ResNet employs residual learning, where layers are designed to learn residual mappings with reference to the layer inputs. In this ZSL framework, the ResNet architecture is employed to extract features from images, which are then mapped onto a semantic space where unseen classes can be predicted. ResNet has been extraordinarily successful in various tasks, including image recognition, object detection, and more. Therefore, it is used in this model along with an image database. The following chapters will provide comparative analysis of several widely used datasets for zero-shot learning, including coarse-grained and fine-grained datasets, as well as a large-scale dataset.

The effectiveness of the ZSL model is evaluated across several datasets, each offering unique challenges. These datasets include CUB, aPY, AWA1, AWA2, SUN, and ImageNet:

- CUB (Caltech-UCSD Birds 200-2011) is a fine-grained dataset with 11,788 images of 200 bird species, [32]. Out of those, 5994 are used for training, and the rest for testing. Each image has detailed annotations: 1 subcategory label, 15 part locations, 312 binary attributes, and 1 bounding box. This dataset is one of the most widely used for fine-grained visual categorisation tasks; therefore, this will provide a suitable test for the model.

- aPY (Attribute Pascal and Yahoo) is a small-scale, coarse-grained dataset containing 15339 images from 3 broad categories, [33]. The categories are animals, objects, and vehicles which are further divided into a total of 32 subcategories (aeroplane, zebra, etc.), 64 attributes across 32 classes. The dataset is divided into 20 classes for training, 5 for validation, and 12 for testing.

- AWA1 (Animals with Attributes 1) is a medium-scale dataset with 30,475 images across 50 classes, annotated with 85 attributes, [31]. The dataset is split into 40 classes for training, 13 for validation, and 10 for testing. AWA2 (Animals with Attributes 2) is an extension of AWA1 which includes 37,322 images for the same 50 classes, offering a richer set of images and a more comprehensive evaluation ground for ZSL models.

- SUN [34] is a fine-grained and medium-scale dataset with respect to both the number of images and the number of classes, i.e., SUN contains 14340 images coming from 717 types of scenes annotated with 102 attributes, [31].

- ImageNet, [35], is a large-scale dataset consisting of 14 million labelled images spanning 21,000 classes, organised hierarchically. It is widely used for both zero-shot due to its diversity in granularity and significant class imbalance.

4.2. Training and testing

The basic methods used are initialisation, finding hyperparameters, training the model, and testing the model. Firstly, it is needed to define the dataset, its path, and the hyperparameters ‘alpha’ and ‘gamma’.

In the initialisation method the dataset needs to be pre-processed. This is done by normalising the data, extracting labels from precomputed features (ResNet) and performing attribute splits. Then organising it into training, validation, and test sets. The class labels are then converted into a contiguous range for easier manipulation during training. Another key point that needs to be secured is the uniqueness of classes. In a typical zero-shot learning scenario, there are no overlapping classes between the training and testing phases, i.e., the train classes are completely different from the test classes. Therefore, verification of no overlapping classes needs to be done.

Then, the signature matrix is created, which holds semantic information for each class, and the "ground truth" matrices are generated that serve as reference points during training and testing.

Hyperparameters 'alpha' and 'gamma' are used to calculate the weights for the model. These hyperparameters can be provided by the user; if not, the calculated hyperparameters are used, snippet of code in figure 4.1 shows the method used for calculation. The method searches for the optimal values of alpha and gamma by training the model on the training set and evaluating its performance on the validation set. The performance metric is the average per-class accuracy derived from the confusion matrix, which provides a summary of the prediction results for which the true values are known. The structure of a confusion matrix is shown in table 4.1.

```
def find_hyperparams(self):
    accu = 0.10
    alph1 = 4
    gamm1 = 1

    #Weights
    V = np.zeros((d_train,a_train))
    for alpha in range(-3, 4):
        for gamma in range(-3,4):
            # Model Training
            part_1 = np.linalg.pinv(np.matmul(self.train_vec, self.train_vec.transpose()) + (10**alpha)*np.eye(d_train))
            part_0 = np.matmul(np.matmul(self.train_vec,self.gt_train),self.train_sig.transpose())
            part_2 = np.linalg.pinv(np.matmul(self.train_sig, self.train_sig.transpose()) + (10**gamma)*np.eye(a_train))

            V = np.matmul(np.matmul(part_1,part_0),part_2)

            # Model Validation
            outputs = np.matmul(np.matmul(self.val_vec.transpose(),V),self.val_sig)
            preds = np.array([np.argmax(output) for output in outputs])

            cm = confusion_matrix(self.labels_val, preds)
            cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
            avg = sum(cm.diagonal())/len(self.val_labels_unseen)

            if avg > accu:
                accu = avg
                alph1 = alpha
                gamm1 = gamma
                print(alph1, gamm1, avg)
    print("Alpha and gamma:",alph1, gamm1)
    return alpha, gamma
```

Figure 4.1 *Method for hyperparameter calculation, [30]*

Table 4.1: Confusion matrix structure

Actual \ Predicted	Positive	Negative
Positive	True Positive	True Negative
Negative	False Positive	False Negative

After this method, the model and data are ready for the training phase, where a weight matrix is computed using the optimal hyperparameters. This matrix is then used to map the features into the semantic space, which allows the model to make predictions on unseen classes.

Lastly, testing in this ZSL model is conducted using the trained weights on a completely unseen test set. The model generates predictions by mapping the test data into the semantic space and calculating the outputs using the trained weights. The predicted labels are then compared with the true labels to evaluate the model's performance. The method used to compute the predictions involves a matrix multiplication operation that applies the learnt weights to the feature vectors extracted from the test images. This step is crucial as it determines the model's ability to generalise to classes it has never seen during training.

The complete process is summarised using figure 4.2, which shows that the training stage uses the matrix *S* together with training instances to learn the matrix *V*, which maps from the feature space to the semantic space. At the inference or prediction stage, that matrix *V* is used with the signatures of test classes, *S'*, to obtain the final linear model *W'*.

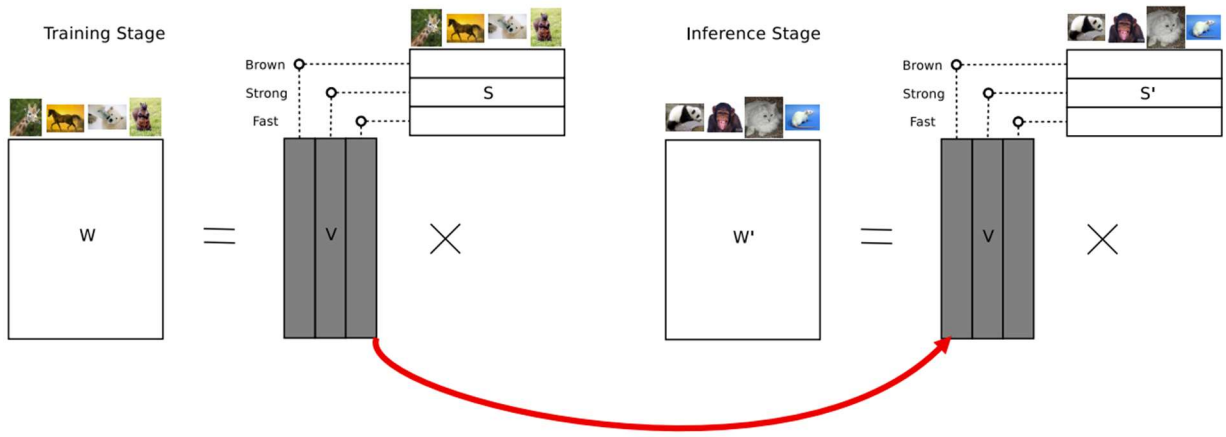

Figure 4.2 *Summary of EsZSL model, [25]*

4.3. Evaluation using Top-1 accuracy

A common evaluation criterion used is the Top-1 accuracy criteria. This metric is used particularly for image classification tasks. The prediction is accurate when the predicted class is the correct one, [31]. A per-class averaged top-1 accuracy is an important evaluation metric when the dataset is not well balanced in the number of images per class. However, top-1 accuracy can be biased

towards classes that are more common in the dataset. This means that if a model performs well on frequently seen classes but poorly on rarer ones, the overall top-1 accuracy might still appear high. To address this, accuracy can be calculated by macro-averaging, ensuring that performance on each class is given equal weight. To achieve high performance on densely populated classes, and on sparsely populated classes it is important to average the correct predictions independently for each class. This is done by the following equation:

$$acc_y = \frac{1}{\|Y\|} \sum_{c=1}^{\|Y\|} \frac{\#correct\ predictions\ in\ c}{\#samples\ in\ c} \qquad (4.1)$$

where:

acc_y represents the average per-class top-1 accuracy,

$\|Y\|$ represents the total number of classes.

c represents an index of each class.

Top-1 accuracy measures the proportion of times the model's top prediction (the class it deems most likely) is correct. When a model classifies an image, if the predicted class with the highest probability matches the true class, it counts as a correct prediction. For example, if a model is trained to classify images into categories like "dog," "cat," and "bird," and it is presented with an image of a dog, the top-1 accuracy would be 100% if "dog" is the model's most confident prediction.

Using this method, evaluation will be done on the EsZSL model and compared to the ZSLGBU model. Results from the ZSLGBU model can be found in table 4.2., along with the EsZSL model results. Comparing the results, it is noticeable that achieving similar results is possible with a simple zero-shot learning model, [30]; further adjusting the hyperparameters alpha and gamma can provide better precision and accuracy. These results are evaluated for features extracted from ResNet-50 for the proposed data split. An accuracy of 53.9%, as stated in the results for the CUB dataset, signifies that the model correctly identifies or predicts the class of 53.9% of the instances in the dataset. This means that just over half of the predictions align with the actual label. For zero-shot learning this could be reasonable, but it is important to consider the nature of the task, complexity of the dataset and performance in comparison to other models in the case of a specific

dataset. In the replicated EsZSL model, [30], a decrease in accuracy is seen almost in every database except for the aPY and SUN databases. This fluctuation in results by a margin of 1-4% may be because this model does not have the kernel implementation used in the ZSLGBU paper, [30]. For small-scale, medium-scaled, and fine-grained database, the model proposed, [30], provided an accuracy of over 50%. Meaning that more than half of the instances are correctly predicted. For small-scale and coarse-grained that accuracy has dropped to around 30%. Future work might focus on refining hyperparameters, enhancing the model architecture, or incorporating additional complementary features to further boost predictive performance.

Table 4.2: Model evaluation results, top-1 accuracy in %, [36]

Dataset	ZSLGBU	EsZSL	Hyperparameters	
			Alpha	Gamma
CUB	53.9%	53.74%	3	1
AWA1	58.2%	56.8%	3	0
AWA2	58.6%	54.82%	3	0
aPY	38.3%	38.56%	3	-1
SUN	54.5%	55.69%	3	2

5. CONCLUSION

Zero-shot learning is a significant advancement in the field of machine learning, addressing the critical challenge of recognizing and categorizing classes or tasks without prior exposure to specific examples. The method's ability to recognize and categorize classes or tasks without prior exposure to specific examples makes this method a way to replicate the human mind. With two key phases and a strong model evaluation it is possible to recreate real-world issues. Therefore, this method is becoming more representative in fields such as healthcare, retail, and e-commerce, etc.

Several techniques have been developed to implement zero-shot learning, each offering unique advantages and addressing specific challenges. Each method offers unique advantages. Attribute-based classification and prediction offer a solution for object classification with disjoint training and test classes, although it can be computationally expensive, or it might not reflect real-word correlations between attributes. Neural networks are a great way to represent the human brain, intuition, and logic in a machine form. Their adaptability provides an advantage which is not seen in other methods, although, their decision-making process can be difficult to interpret. Support Vector Machines have the ability to handle linear and non-linear classifications offering versatility, but they often require extensive experimentation and cross-validation. Lastly, transfer learning and domain adaptation allow models to generalise to new classes with minimal additional training, because the knowledge can be transferred from one model to another. However, the model's performance is then highly dependent on the similarity between the source and the target tasks.

A specific model described in this thesis demonstrates how accurate zero-shot learning is when it comes to prediction. Depending on the type of dataset; fine-grained, coarse-grained, small-scale, medium-scale or large-scale, a zero-shot model will offer different results. The model described, [30], provides a lower accuracy on small-scaled and coarse-grained datasets, around 30%. This means that just around 30% of the predictions align with the actual label. Given these nuances, this accuracy can be viewed through an optimistic lens, particularly when noting that zero-shot models are often employed in scenarios where labelled data for specific classes is scarce. Furthermore, if evaluations on the other datasets such as AWA1 or CUB provide higher accuracy levels, shows that certain datasets might be more conducive than others for zero-shot learning methods. It is important to note that traditional supervised learning methods on well-curated datasets might yield accuracies exceeding 70-90%. In contrast, zero-shot learning models, by their nature, might not reach these levels since they operate under the premise of generalization.

In conclusion, zero-shot learning enables models to generalize tasks and categorize classes they have never encountered before by leveraging semantic relationships and transferring knowledge from related domains. This approach has the potential to significantly expand the capabilities of machine learning systems. Zero-shot learning can be applied in various domains, including image recognition, natural language processing, recommendation systems, healthcare, and fraud detection, and it holds promise for driving further advancements in human discovery.

REFERENCES

[1] Alpaydin, E.: “Introduction to Machine Learning”, 3rd ed. The MIT Press, Cambridge, MA, 2014.

[2] N/A: "Zero-Shot Classification", s Interneta, https://huggingface.co/tasks/zero-shot-classification, 04. rujan 2024.

[3] Potrimba, P.: “What is Zero Shot Learning in Computer Vision?”, s Interneta, https://blog.roboflow.com/zero-shot-learning-computer-vision/, 04. Rujan 2024.

[4] Ding, H., Ma, Y., Deoras, A., Wang, Y.: "ZERO-SHOT RECOMMENDER SYSTEMS", arXiv preprint, 2021. Dostupno s Interneta: https://arxiv.org/abs/2105.08318, 12. kolovoz 2024.

[5] Wang, Z., Shen, R., Stadie, B.: "Solving Robotics Problems in Zero-Shot with Vision-Language Models", arXiv preprint, 2024. Dostupno s Interneta: https://arxiv.org/abs/2407.19094 , 04. Rujan 2024.

[6] Gull, S., Minhas, F.: "AMP0: Species-Specific Prediction of Anti-microbial Peptides Using Zero and Few Shot Learning," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 19, no. 1, pp. 275-283, Jan-Feb 2022. doi: 10.1109/TCBB.2020.2999399. Epub 2022 Feb 3. PMID: 32750857

[7] Rizinski, M., Jankov, A., Sankaradas, V. i dr.: "Company classification using zero-shot learning", arXiv preprint, 2023. Dostupno s Interneta: https://ar5iv.labs.arxiv.org/html/2305.01028v2 , 04. Rujan 202

[8] Liu, T., Zhang, S., Qin, Y., Tao, X.: "A Text Prompt-Based Approach for Zero-Shot Corner Case Object Detection in Autonomous Driving," in Proc. 2023 IEEE 26th Int. Conf. on Intelligent Transportation Systems (ITSC), Bilbao, Spain, 2023, pp. 3241-3246. doi: 10.1109/ITSC57777.2023.10422170.

[9] N/A: "Know about Zero-Shot, One-Shot, and Few-Shot Learning", s Interneta, https://www.analyticsvidhya.com/blog/2022/12/know-about-zero-shot-one-shot-and-few-shot-learning/, 12. kolovoz 2024.

[10] Tan, C., Xu, X., Shen, F.: " A Survey Of zero shot detection: Methods and applications", Cognitive Robotics, vol. 1, str. 159-167, 2021. Dostupno s Interneta: https://www.sciencedirect.com/science/article/pii/S2667241321000124, 24. Kolovoz 2024.

[11] Le Cacheux, Y., Le Borgne, H., Crucianu, M.: "Zero-Shot Learning with Deep Neural Networks", s Interneta, https://www.catalyzex.com/paper/zero-shot-learning-with-deep-neural-networks, 12. kolovoz 2024.

[12] Nisha, A.: "Zero-shot learning, Explained", s Interneta, https://www.kdnuggets.com/2022/12/zeroshot-learning-explained.html, 04. rujan 2024.

[13] N/A: "Accuracy, precision, and recall in multi-class classification", s Interneta, https://www.evidentlyai.com/classification-metrics/multi-class-metrics , 04. Rujan 2024.

[14] Britannica, The Editors of Encyclopaedia. "Euclidean distance". Encyclopedia Britannica, 28 Jun. 2024, https://www.britannica.com/science/Euclidean-distance. Accessed 31 August 2024.

[15] N/A: "Hamming Distance", s Interneta, https://en.wikipedia.org/wiki/Hamming_distance , 12. kolovoz 2024.

[16] Peterson, L. E.: "K-nearest neighbor," Scholarpedia, vol. 4, no. 2, pp. 1883, 2009. Dostupno na: https://scholarpedia.org/article/K-nearest_neighbor, 12. kolovoz 2024.

[17] Shi, W., Michael, J., Gururangan, S. I dr.: "kNN-Prompt: Nearest Neighbor Zero-Shot Inference", s Interneta, https://aclanthology.org/2022.emnlp-main.214.pdf, 04. rujan 2024.

[18] Lampert, C. H., Nickisch, H., Harmeling, S.: "Attribute-based classification for zero-shot learning", s Interneta, https://hannes.nickisch.org/papers/articles/lampert13attributes.pdf, 13. kolovoz 2024.

[19] Abdi, H., Valentin, D., Edelman, B.: "Neural Networks," s Interneta, dostupno na: https://books.google.hr/books?hl=en&lr=&id=ZEID3w9STOUC&oi=fnd&pg=PP7&dq=neural+networks&ots=sAJxEZVCZN&sig=hqIRJ36tAXG82GQ0KqZmn4_yM8o&redir_esc=y#v=onepage&q=neural%20networks&f=false, 13. kolovoz 2024.

[20] Han, S. H., Kim, K. W., Kim, S., Youn, Y. C.: "Artificial Neural Network: Understanding the Basic Concepts without Mathematics," Dementia and Neurocognitive Disorders, vol. 17, no. 3, pp. 83-89, Sep. 2018. doi: 10.12779/dnd.2018.17.3.83. Epub 2018 Dec 13. PMID: 30906397; PMCID: PMC6428006.

[21] Shmilovici, A.: "Support Vector Machines," u: Maimon, O., Rokach, L. (ur.) Data Mining and Knowledge Discovery Handbook, Springer, Boston, MA, 2005. Dostupno s Interneta: https://doi.org/10.1007/0-387-25465-X_12.

[22] Huang, S., Cai, N., Pacheco, P. P., Narrandes, S., Wang, Y., Xu, W.: "Applications of Support Vector Machine (SVM) Learning in Cancer Genomics," Cancer Genomics & Proteomics, vol. 15, no. 1, pp. 41-51, Jan-Feb 2018. doi: 10.21873/cgp.20063. PMID: 29275361; PMCID: PMC5822181.

[23] N/A: "Zero-shot Learning - IBM", s Interneta, https://www.ibm.com/topics/zero-shot-learning, 16. kolovoz 2024.

[24] Kodirov, E., Xiang, T., Gong, S.: "Unsupervised Domain Adaptation in Zero-Shot Learning", s Interneta, https://openaccess.thecvf.com/content_iccv_2015/papers/Kodirov_Unsupervised_Domain_Adaptation_ICCV_2015_paper.pdf, 16. kolovoz 2024.

[25] Romera-Paredes, B., Torr, P.H.S.: "An Embarrassingly Simple Approach to Zero-Shot Learning ", Visual Attributes, Advances in Computer Vision and Pattern Recognition, str. 11-30, 2017.

[26] Sahin, A.: "Few-shot and Zero-shot Learning," s Interneta, dostupno na: https://medium.com/@sahin.samia/few-shot-and-zero-shot-learning-teaching-ai-with-minimal-data-801603ed40f8, 26. kolovoz 2024.

[27] Kundu, R.: "Few-shot Learning Guide," s Interneta, dostupno na: https://blog.paperspace.com/few-shot-learning /, 20. kolovoz 2024.

[28] N/A: "Comprehensive Guide to Zero-shot vs Few-shot Techniques in NLP", s Interneta, https://ubiai.tools/comprehensive-guide-of-zero-shot-vs-few-shot-techniques-in-nlp/ , 20. kolovoz 2024.

[29] Mayo, M.: "Tips for Choosing the Right Machine Learning Model," s Interneta, dostupno na: https://machinelearningmastery.com/tips-for-choosing-the-right-machine-learning-model-for-your-data/, 20. kolovoz 2024.

[30] Bharadwaj, S.: "Embarrassingly Simple Zero-shot Learning," GitHub, dostupno na: https://github.com/sbharadwajj/embarrassingly-simple-zero-shot-learning, 26. kolovoz 2024.

[31] Xian, Y. i dr.: "Zero-Shot Learning — A Comperhensive Evaluation of the Good, the Bad and the Ugly,", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), str. 3077-3086., Honolulu, HI, USA, 2017.

[32] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, et al., "Caltech-UCSD Birds 200", 2010.

[33] A. Farhadi, I. Endres, D. Hoiem and D. Forsyth, "Describing objects by their attributes", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 1778-1785, 2009.

[34] G. Patterson and J. Hays, "Sun attribute database: Discovering annotating and recognizing scene attributes", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 2751-2758, 2012.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 248-255, 2009.

[36] Paul, S.: "Popular ZSL Algorithms," s Interneta, dostupno na: https://github.com/mvp18/Popular-ZSL-Algorithms/tree/master/ESZSL, 26. kolovoz 2024.

FIGURES

FIGURE 2.1: *ZERO-SHOT LEARNING OVERVIEW, [9]* 4
FIGURE 2.2: *PHASES OF A BASIC ZERO-SHOT LEARNING MODEL, [11]* 5
FIGURE 2.3: PERCENTILE-WISE RFPP SCORES, [6] 10
FIGURE 3.1*: PHASES OF A BASIC ZERO-SHOT LEARNING MODEL [15]* 11
FIGURE 3.2 PSEUDOCODE FOR K-FOLD CROSS-VALIDATION 12
FIGURE 3.3: *DIRECT ATTRIBUTE PREDICTION (DAP), [18]* 14
FIGURE 3.4 *: INDIRECT ATTRIBUTE PREDICTION (IAP), [18]* 15
FIGURE 3.5 *STEP FUNCTION (LEFT) AND SIGMOID FUNCTION (RIGHT), [20]* 16
FIGURE 3.6 *BIOLOGICAL NEURAL NETWORK (TOP) AND MULTI-LAYER PERCEPTION IN ARTIFICIAL NEURAL NETWORK (BOTTOM), [20]* 18
FIGURE 3.7 *LINEAR SVM MODEL. TWO CLASSES (READ VERSUS BLUE) WERE CLASSIFIED, [22]* 20
FIGURE 3.8 *KERNEL FUNCTION, [22]* 20
FIGURE 3.9 *GENERAL OVERVIEW OF A FEW-SHOT LEARNING FRAMEWORK, [27]* 23
FIGURE 4.1 *METHOD FOR HYPERPARAMETER CALCULATION, [30]* 28
FIGURE 4.2 *SUMMARY OF ESZSL MODEL, [25]* 29

SUMMARY

This thesis analyses a cutting innovative technique in machine learning. Zero-Shot Learning enables models to recognize and classify data from classes they have never encountered before. This method is particularly valuable in situations where acquiring labelled data for every possible class is impractical or impossible. Zero-shot learning has broad applications across various industries, including healthcare, e-commerce, and autonomous driving, rapidly gaining importance. Because of recent advancement there are a number of methods, approaches, and techniques used to achieve this type of learning. This thesis offers a comprehensive explanation of the fundamental concepts of zero-shot learning, explores the primary methods and applications, and demonstrates the performance and behaviour of a basic zero-shot learning model on different datasets.

Keywords: machine learning, zero-shot learning, ZSL, few-shot learning, FSL, neural networks, Support Vector Machines, transfer learning

SAŽETAK

Ovaj rad analizira inovativnu tehniku strojnog učenja, učenje bez pokušaja. Učenje bez pokušaja omogućuje modelima prepoznavanje i klasificiranje podataka iz klasa s kojima se nikada prije nisu susreli. Ova metoda je posebno vrijedna u situacijama kada je prikupljanje označenih podataka za svaku moguću klasu nepraktično ili nemoguće. Zero-shot učenje ima široku primjenu u raznim industrijama, uključujući zdravstvo, e-trgovinu i autonomnu vožnju, brzo dobivajući na važnosti. Zbog nedavnog napretka postoji niz metoda, pristupa i tehnika koje se koriste za postizanje ove vrste učenja. Ovaj rad daje objašnjenje temeljnih koncepata zero-shot učenja, istražuje primarne metode i primjene te demonstrira izvedbu i ponašanje osnovnog zero-shot modela učenja na različitim skupovima podataka.

Ključne riječi: strojno učenje, učenje bez pokušaja, ZSL, učenje s malim brojem pokušaja, FSL, neuronske mreže, Support Vector Machines, učenje s prijenosom znanja