HIERARCHICAL IMAGE CLASSIFICATION WITH SELF-SUPERVISED VISION TRANSFORMER FEATURES

A Thesis Submitted to the
Graduate School Izmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

In Computer Engineering

by
Caner KARAGÜLER

March 2022
İZMİR

ACKNOWLEDGEMENTS

First of all, I would like to give my special thanks to my supervisor Asst. Prof. Mustafa Özuysal for all kinds of support and guidance to make this study happen. Also, I am very grateful to him for sharing his time and knowledge with me.

I also would like to thank my friends and colleagues for not leaving me alone and motivating me whenever I needed to.

I am infinitely grateful to my family for their lifetime support including this stressful phase.

ABSTRACT

HIERARCHICAL IMAGE CLASSIFICATION WITH SELF-SUPERVISED VISION TRANSFORMER FEATURES

There are lots of works about image classification and most of them are based on convolutional neural networks (CNN). In image classification, some classes are more difficult to distinguish than others because of non-even visual separability. These difficult classes require domain-specific classifiers but traditional convolutional neural networks are trained as flat N-way classifiers. These flat classifiers can not leverage the hierarchical information of the classes well. To solve this issue, researchers proposed new techniques that embeds class-hierarchy into the convolutional neural networks and most of these techniques exceed existing convolutional neural networks' success rates on large-scale datasets like ImageNet.

In this work, we questioned if a hierarchical image classification with self-supervised vision transformer features can exceed hierarchical convolutional neural networks. During this work, we used a hierarchical ETHEC dataset and extract attention features with the help of vision transformers. Using these attention features, we implemented 3 different hierarchical classification approaches and compared the results with CNN alternative of our approaches.

ÖZET

ÖZDENETİMLİ GÖRÜ DÖNÜŞTÜRÜCÜ ÖZNİTELİKLERİ İLE HİYERARŞİK İMGE SINIFLANDIRMASI

Görüntü sınıflandırma ile ilgili pek çok çalışma bulunuyor ve bunların çoğu evrişimli sinir ağları (CNN) temel alınarak gerçekleştirilmiştir. Görüntü sınıflandırmada, eşit olmayan görsel ayrılabilirlik nedeniyle bazı sınıfları diğerlerinden ayırt etmek daha zordur. Bu zor sınıfların ayrılabilmesi için, ilgili alana özgü sınıflandırıcılar gerekmektedir, ancak geleneksel evrişimli sinir ağları, düz N-yollu sınıflandırıcıları olarak eğitildiği için sınıflar arasındaki hiyerarşik bilgiden yeteri kadar yararlanamazlar. Bu sorunu çözmek için araştırmacılar, sınıf hiyerarşisini evrişimli sinir ağlarına dahil eden yeni teknikler keşfettiler ve bu tekniklerin çoğu, ImageNet gibi büyük ölçekli veri kümelerinde mevcut evrişimli sinir ağlarının başarı oranlarını geçmektedir.

Bu çalışmada, özdenetimli görü dönüştürücü özniteliklerini kullanan bir hiyerarşik imge sınıflandırıcının hiyerarşik evrişimli sinir ağlarını geçip geçemeyeceğini sorguladık. Bu çalışma sırasında hiyerarşik bir ETHEC veri seti kullandık ve görüntü transformatörleri yardımıyla dikkat öznitelikleri çıkardık. Bu dikkat özelliklerini kullanarak 3 farklı hiyerarşik sınıflandırma yaklaşımı uyguladık ve sonuçları yaklaşımlarımızın CNN alternatifi ile karşılaştırdık.

To my family...

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

In the real world, there are lots of classification problems that we can fit into a hierarchical domain. For these domains, using a flat classifier means that you are eliminating the power of hierarchy information. Instead of a flat classifier, using a hierarchical classifier probably performs better in these kinds of domains. [1] So, the first question is "Why do we not use hierarchical classifiers a lot if they can outperform flat classifiers in hierarchical domains?" To be able to answer this question, we first need to understand the difference between a hierarchical classifier and a flat classifier.

The flat classifier is simple. For a domain that can also fit into a hierarchy, you can easily create and deploy a single model with some hyperparameter optimization. And if you have enough data, the results can also be satisfactory. On the other hand, hierarchical classifiers are more complex to design and to implement. You need to create multiple classifiers for each level or even for each node. Optimizing all these classifiers is a difficult problem. If you can properly deploy a hierarchical model, it will likely outperform the flat classifiers for the same domain [1]. But in the real-world, accuracy is not the only parameter to determine which approach is better. For a flat classifier, you do not need to have as much computational power as the hierarchical classifier requires. And also making a prediction in a flat classifier is generally much faster than using a hierarchical classifier as it needs to compute more. So we can answer the question like " If the accuracy difference between flat classifier and hierarchical classifier is acceptable, then using flat classifier will be a better choice rather than hierarchical classifier due to high computational cost requirement, optimization difficulty, time consumption and etc. "

So, the main problem with hierarchical classifiers is actually based on their implementations. If we can reduce the computational cost, time, and amount of optimization parameters, it will be a better alternative to flat classifiers. In this work, we inspected and analyzed different hierarchical CNN approaches. All these approaches have the same problems such as the high amount of data requirement, training and optimization difficulties, time requirements and etc.

In this work, we are focused on outperforming two different hierarchical CNNs which are Per Level Classifier [2] and Masked Per Level Classifier [2] by using the ETHEC [3] dataset. Both models have a single CNN for each Level. What we tried to do is replace these CNNs with K-NNs that are trained with self-supervised vision transformer features [4]. In CNN there is a probability that shows the classification results. Similarly in K-NN, there are distances and similarities that represent how close is the test item to the train items. As you can see, in both CNN and K-NN we can obtain a metric that allows us to make predictions. K-NN have a great advantage on CNN such as:

- Requires less data
- No training requirement
- Works better in smaller datasets
- Easy to deploy
- Only two parameters for optimization
- Easy to add new classes
- Lower computational cost

All these advantages are solving the main problems of hierarchical CNNs but there is a problem with our approach. Replacing CNNs with K-NNs is not enough to make a hierarchical classifier. We need to first convert the similarity metric into a probability to be able to make a prediction. To do this we add an SVM [5] as the last phase of the hierarchical classifier.

Also, instead of using the dataset directly, we feed it into a Vision Transformer [4] to obtain attention. With this attention, we are expecting to increase the overall accuracy of the proposed approaches as we eliminate most of the outlier features in the images.

We can say with the proposed methods in this work we are expecting to reduce the training times, computational cost, optimization difficulties and increase the overall accuracy for hierarchical classifiers.

CHAPTER 2

THEORETICAL BACKGROUND

In this section, 'Classification by Machine Learning', 'Hierarchical Classification' and 'Vision Transformers' concepts are mentioned.

2.1 Classification by Machine Learning

Machine learning focuses on creating intelligent systems that learn like a human and improves their performance based on the amount and structure of the data consumed.

Currently, we can see lots of machine learning applications in banks, online shopping systems, security systems, social media, and so on. Machine learning provides us with efficient and reliable solutions to the problems of different industries. [6]

A standard classification algorithm can be divided into three different steps that are:

Decision Process: The main purpose of a classification algorithm is to make a prediction. The model will predict an output (ex: a resulting class in classification problem) based on input data (labeled or unlabeled) with the help of learned features in the training phase.

Error Function: Error functions are used in the prediction step of the model. The main objective of the error function is determining the difference between the real result and models prediction. This function can also be used for benchmarking or evaluating the model's accuracy. There are many different error functions such as cross-entropy, focal, Huber, etc. [7]

Model Optimization Process: For training a model we need to minimize the error rate between model prediction and real results. To be able to do this, during the training phase we need to update the weights until the model reaches the threshold of the accuracy [8] .

Machine learning algorithms have been classified in many ways in the literature. Mainly, it is divided into supervised and unsupervised learning. However, with the

progress and development of technology and algorithms, there are additions to these two main points. In addition to these, the methods can be listed as semi-supervised, reinforcement, transudative reasoning, online, and active learning. According to Brownlee, learning styles were examined under four headings such as supervised, unsupervised, semi-supervised, and reinforcement learning [9].

2.1.1 Supervised Learning

It is a learning method in which the effect and interaction of the inputs on the outputs are observed under the supervision of a supervisor of the training set consisting of inputs and outputs. The basic element in supervised learning is the existence of a training set consisting of previous observations and this training set is taught and introduced to the system by a supervisor. And because of this learning, it is possible to make the necessary estimation for a sample that has never been introduced before. In this learning system, all training data is labeled and the model knows which input data is related to which label so that the relationship between input set and output labels are learned by the model. The aim of supervised learning is to minimize the error rate between the prediction and the actual result for each input with the function created from the learning set. If the difference is more than the predetermined error value, the system continues training. When the difference reaches the desired range, the training is completed and ends. After the model reaches the desired level, a new observation that was not in the training set before is processed by the model, and an attempt is made to make the closest prediction to the truth. Supervised learning is generally used in classification and regression-based estimation problems. The algorithms such as Decision trees [10], SVM [5], K-NN [11], ANN [12], Genetic Algorithms [13] can be described as supervised learning methods.

2.1.5 Machine Learning Steps

There are certain basic steps to be followed in the face of a problem that is desired to be solved by making use of machine learning. It is important to implement these steps to solve the existing problem successfully and at the desired time. The steps to be followed in this process are as follows [14]:

- Defining the Problem
- Data Analysis
- Data Preparation
- Selecting the Model
- Evaluation of the Model
- Using the Model

2.1.5.1 Defining the Problem

The first step to be taken in the problems to be solved with machine learning is to define the problem in the best way and to clearly state what the goal is to be achieved with the solution. Because a project or problem that is not well defined or whose purpose is not clear will create a severe problem for the researcher who will work on it, in reaching a conclusion about how to draw a path. This will put the researcher in a difficult position to reach the desired result.

At this step, along with the purpose of the problem, the success criterion and the current situation should be well defined. Because it is important to know which criteria and output will be taken into account in order to correctly interpret the outputs obtained as a result of machine learning.

2.1.5.2 Data Analysis

The second step in the application of machine learning is to obtain data suitable for the problem. Data for learning can be provided in 2 ways. First of all, the desired data about the problem can be accessed from existing data warehouses and databases. Secondly, the researcher's own questionnaire, debate, certain measurements, etc. related to the problem. data groups that can be obtained through studies.

The researcher examines the data obtained at this stage in general terms. Issues such as format, quantity, and several data are reviewed. In addition, it should be checked whether the main feature to be used in the model is met by that data set. .g; If age is an important field in the study, but the data collected does not reflect the entire age range, that data set can be changed, or the desired age distribution should be reached with new additions.

2.1.5.3 Data Preparation

After obtaining the necessary data for machine learning, the data should be made available to machine learning algorithms. In this sense, the obtained data needs to be pre-processed. These are data cleaning, data integration and transformation, and data reduction. Which of these operations will be used depends on the data set.

During the data cleaning phase, two basic operations are performed. First, each data set may contain data that contain inconsistent or erroneous information. The cleaning process of this data, called noise, is done at this stage. Secondly, it is to convert the data called missing value into a meaningful expression or to remove it from the data set. Sometimes different values cannot be measured or overlooked in the data set. These data, called missing values, can be removed from the data set if they do not affect the study. If the meaning it adds to the data set is high, the mean value of that feature can be assigned to those missing values or the most recurring value can be assigned, or the missing values can be added to the data set with different methods such as using regression.

In the data integration and transformation phase, it is ensured that the expressions in the same data set need to belong to the same environment. Indicators expressing the same situation in the data set can be displayed more than once and in different ways. Since this may cause the result to change by perceiving each expression differently during the learning phase of the machine, it is ensured that it is converted into a uniform expression.

The process of reducing the number of data or variables in a way that does not change the results to process the data more easily and quickly in the learning process is called data reduction. There are lots of different methods that can be used for reducing the data.

After the data preprocessing phase, feature engineering studies are carried out according to the structure of the problem and the data. Normalization and Data Scaling is one of the prominent processes in feature engineering studies.

Normalization: Numerical variables in the data set to change at different intervals. Since this numerical difference will increase the effect of large range values on the result, data normalization is performed. Thus, it is aimed to achieve the best results

by normalizing the variables. For this, different normalization methods such as Z-transform, minimum-maximum, 0-1 range are used [15].

Data Scaling: It is defined as dividing the entire data by a fixed number. It is to speed upslope expansion by keeping each of the input values in the same range. In scaling, a new range is created by dividing the input values by the range of that input variable.

2.1.5.4 Selecting the Model

It is the process of determining the algorithm or technique to be applied in the data set by deciding which method to achieve the best result by trial and error. It is the stage where the most suitable models and algorithms that will best analyze the connection between the variables in the data set are determined. At this stage, more than one suitable model is determined, and the best result is tried to be achieved by applying it to the data set. For this reason, the more models and algorithm techniques are determined and applied, the higher the percentage of yield from the data set.

2.1.5.5 Evaluation of the Model

It is the stage where the quality and impact of the result obtained by measuring how much of the project or business objectives determined at the beginning are met, is evaluated before the model becomes widespread. In addition, clear decisions should be made about the extent to which the points especially emphasized in the problem are considered and whether the results will be beneficial or not.

At this stage, different models and algorithms applied are compared. The results obtained are compared and verified with the results of other studies. In addition, in the evaluation process of the model, it is necessary to evaluate whether the successfully predicted algorithm can be generalized within itself or not. There are different evaluation methods such as k-fold cross-validation [16]. With this method, the data is split into k different parts equally. One at a time of k part is used for testing and k-1 for training. As a result, k error rates are obtained, and the average of the errors is taken to calculate the entire forecast error.

2.1.5.6 Using the Model

After the model that will solve the problem in the hands of the researcher is evaluated and determined, that model is the main model for the problem. Now, that model can be used very easily, and desired results can be obtained in subjects like the existing problem or in research. Thus, that determined model will become widespread and will be the main source of problem solutions. The success rate of the model will vary in proportion to the rate of benefiting from the model.

2.1.6 Algorithms For Classification

2.1.6.1 K-Nearest Neighbor

The k-nearest neighbor [11], method is a classification method that determines the class in which the observations will take place and the nearest neighbor according to the k-value. It is one of the supervised machine learning algorithms that classify based on the distance between observations or objects. It is used in many fields such as pattern recognition [17], computer vision [18], data mining [19], statistics [20], cognitive psychology [21], medicine [22], and bioinformatics [23].

K-nearest neighbor algorithm makes classification with the help of distance or proximity calculation. In summary, the basis of this classification algorithm is the idea that "objects that are close to each other in the sample space probably belong to the same category". The purpose of the algorithm is to assign individuals or objects to predetermined classes or groups in the most accurate way, by making use of the properties of these objects. The method also provides a classification of a new observation. With the help of the learning data set, the observation to be classified is classified in the same data set with the k closest observations and the most similar ones.

K-nearest neighbor method has many advantages such as giving clear and effective results [24], ignoring missing observations in continuous variables [25], having the option to evaluate missing observations in the categorical variable, the answer variable being categorical, continuous, or a combination of the two, and having few assumptions because it is a non-parametric method also, the disadvantages of the algorithm are that the number of k, which gives the number of nearest neighbors, is required, it is affected

by the selected distance measure, and there is no certainty about which distance measure to use [26].

The k-nearest neighbor algorithm is used to classify observations according to their similarity to other phenomena [27]. It was developed as a way of recognizing data patterns without exact matching to learned patterns. Similar observations are close to each other (adjacent) and dissimilar observations are far apart. Therefore, the distance between two observations is a criterion that determines the dissimilarity. The distances of a new observation from the observations in the model are calculated. This observation is assigned to the most repetitive/similar category. The algorithm steps are:

- The distance of the new observation to all the observations in the data set is calculated,
- These distance values found are listed,
- k observations with the smallest distance are selected,
- The category with the most repetitions (majority voting) in k observations becomes the class value

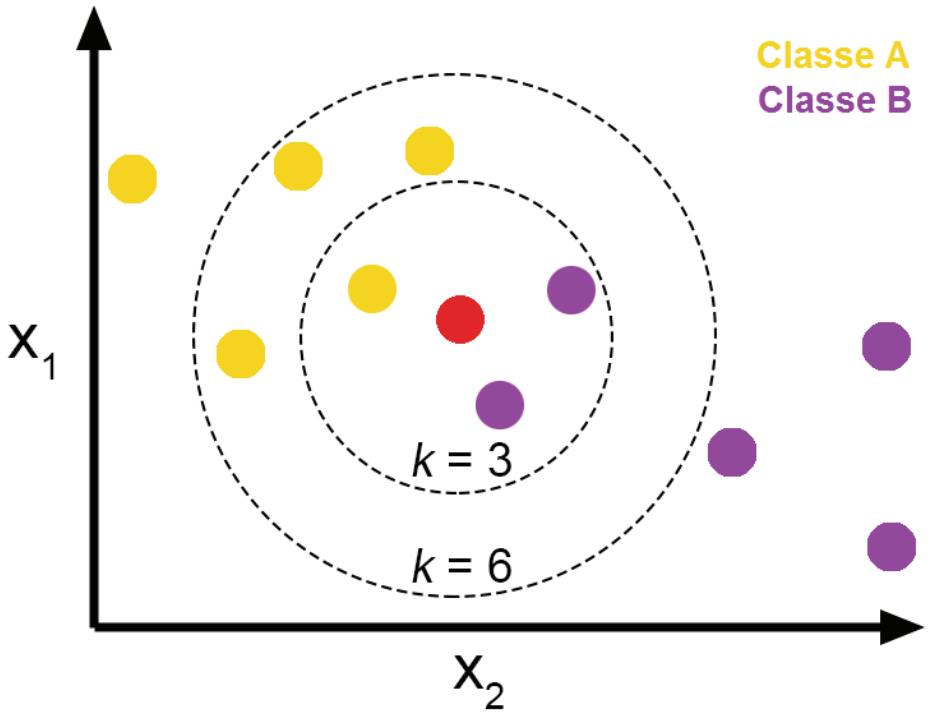

Figure 1 K-NN with different k values [28]

2.1.6.2 Support Vector Machine

SVM is one of the supervised learning algorithms. The main principle is finding the optimum hyperplane in multi-dimensional space for classification [29] or regression [30] tasks. There can be multiple hyperplanes so for selecting the optimum hyperplane we need to select the maximum margin area. There can be multiple hyperplanes so for selecting the optimum hyperplane we need to select the maximum margin area.

There are two types of margins that are soft and hard margins. In the soft margin, there can be some points in the margin area which can be called outliers contrary to that the hard margin works if only the points can be separated linearly. To be able to select the optimum margin, we can use the C parameter. If the C is going high, the margin area becomes smaller. Also If the model overfits decreasing C can be a solution.

In some big and complicated datasets using a low number of dimensions in SVM cannot be enough for good classification results. To be able to solve that, points are multiplied with a function called Kernel Function [31].

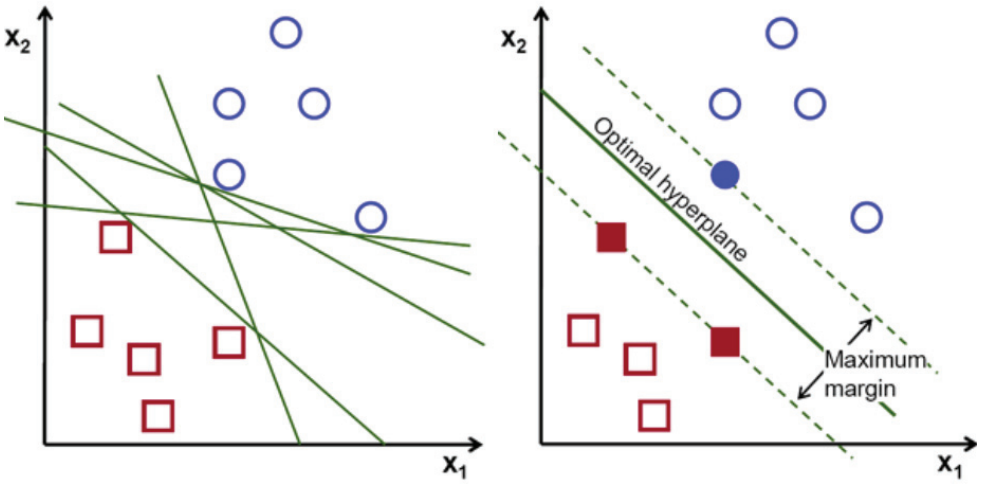

Figure 2 Optimal hyperplane [32]

2.1.6.3 Artificial Neural Networks

Artificial neural networks (ANN [12]) try to create a new system by imitating the structure of biological nerve cells in the human brain and creating learning and decision-making mechanisms with these systems. Artificial neural networks, like the human brain, have a learning mechanism and a decision-making mechanism based on the information they learn.

The task of an artificial neural network is to create an output set by training itself from the given input set. For this process, artificial neural networks are trained with a training set of the related problems and become able to solve the expectations related to the related problems [33] .

Learning occurs in artificial neural networks. They may make similar decisions in similar situations. They can even give information about unseen examples after learning. They can recognize and classify images.

Although artificial neural networks are extremely useful, they also have many disadvantages. The use of the trial-and-error method while determining the network structure complicates the training process and cannot guarantee that the network structure found is the most correct solution. There is no end of education rule. Although it is considered sufficient for the training to be completed, it does not guarantee that the result is the best result. [34]

2.1.6.4 Deep Learning

Deep learning can be described as an evolved version of artificial neural networks [35]. Since the number of neurons is low in the first artificial neural network models, only simple learning takes place. In order to solve complex problems with artificial neural networks, the number of layers must be increased. Thus, the decision mechanism is also deepened

Because the computational methods in deep learning require a lot of mathematical processing power, the CPU architecture is insufficient. Recently, graphics processing unit (GPU) systems have been used for deep learning. Thanks to the GPU architecture, very large data can be used for training [36]. Training with big data has a positive effect on the results. Therefore, the power of GPU systems directly affects the result. Recently, many big technology companies have been working on deep learning, realizing live and working projects in this field, and investing heavily in this field [37].

Deep learning, when trained, learns its distinguishing features by looking at many inputs given. In the feature learning stage, which consists of layers, the lower-level layers have less distinctiveness, while the upper-level layers formed by the merger of the lower levels have more distinctiveness. Because the layers at the lower level form an infrastructure for the layers at the upper level.

Deep learning is a complex artificial neural network that contains multiple hidden layers [38]. In deep learning, each input layer is trained on different data depending on the previous output layers. As the network gets deeper, more complex features can be learned. Because the features of previous layers are transferred to the next layer and these features are combined. The peculiarity of deep networks is that they learn hidden undiscovered structures in datasets. The greater number of hidden layers in the network, the deeper the network becomes.

In deep learning, a hierarchy can be created by passing low-level features to higher levels to obtain high-level features. In this hierarchy, there can be multiple levels that need to be trained individually. Deep learning can learn from different types of property representations. For example, for an image, properties such as density vector per pixel, edge clusters, special shapes can be said to represent.

2.1.6.5 Transfer Learning

The use of pre-trained models as initial parameters for a different task is called transfer learning [39]. This method is frequently used in some machine learning problems. With the applied transfer learning method, the designers had the opportunity to both save time and obtain high accuracy rates. Complex models are difficult to design. With the proposed transfer learning, it is possible to achieve higher performance with a smaller dataset.

There are application differences between classical machine learning and transfer learning. In classical machine learning, all problems are tried to be solved by starting the parameters randomly. Transfer learning, on the other hand, uses pre-trained models used in the solution of different problems as a starting parameter for the solution of the desired problems and provides solutions with faster and higher performance.

Transfer learning is divided into three main groups inductive, transductive, and unsupervised. While making this distinction, it is considered whether the task and data to be solved are labeled. The type of learning in which the source and target problems are different from each other is inductive learning. In this type of learning, the applications from which the data are obtained are not important. Pre-trained networks need to transfer their parameters to solve different problems. The target and source tasks of the designers may differ from each other. Along with retraining using pre-trained networks in CNN, a

weight update method with the fine-tuning method should be developed. Solving existing problems with deep learning requires a lot of data. So that the number of data should be large to eliminate the overfitting problem. With the transfer learning method, transfer learning is used instead of training the network with random initial values. With this method, training of CNN structures is provided effectively with fewer data.

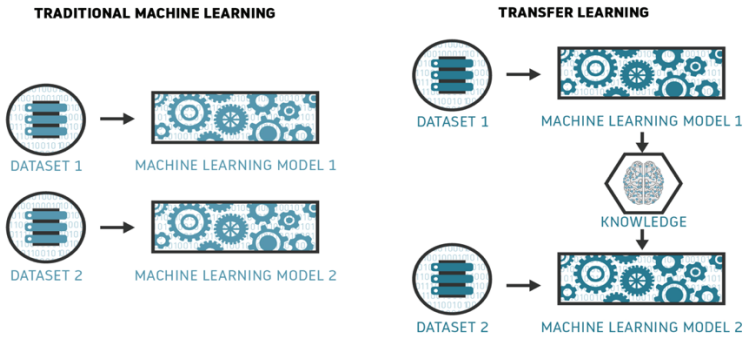

Figure 3 Traditional Machine Learning vs Transfer Learning [40]

2.2 Hierarchical Classification

Most of the research in machine learning is based on standard flat classification which is binary or multiclass classification problems. In real-world actually, most of the problems are actually hierarchical classification problems. In these problems, we can fit classes into a tree or DAC to obtain a hierarchy.

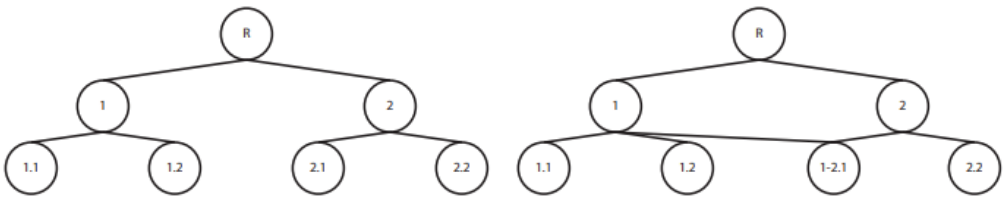

Figure 4 Tree structure (left) and DAG structure (right) [41]

There is no such thing as a standard way to generate a hierarchical classification model. In most cases, these models are domain-specific so it is hard to evaluate the same classifier in different domains. However, the most common hierarchical classification strategies are One-Against-One and the One-Against-All strategies.

We can differ hierarchical classification in multiple criteria's that are:

- **Type of structure:** There are two types of structures that are a tree or DAG. In figure 6 both of these structure types are shown. The main difference between tree and DAG is that DAG nodes can have more than one parent but the tree structure can not.
- **Depth of hierarchy:** There are 2 different methods to implement a hierarchical classification. The first method is always classifying the leaf node and the other one is stopping classification in any level of hierarchy
- **Structure exploring:** Current research generally focused on top-down approaches but there are lots of different approaches such as flat-classification, local classification, global classification and etc.

Flat Classification Approach: This is a simple approach for hierarchical classification problems. In this approach, the model predicts only leaf node classes so that this approach ignores the class hierarchy.

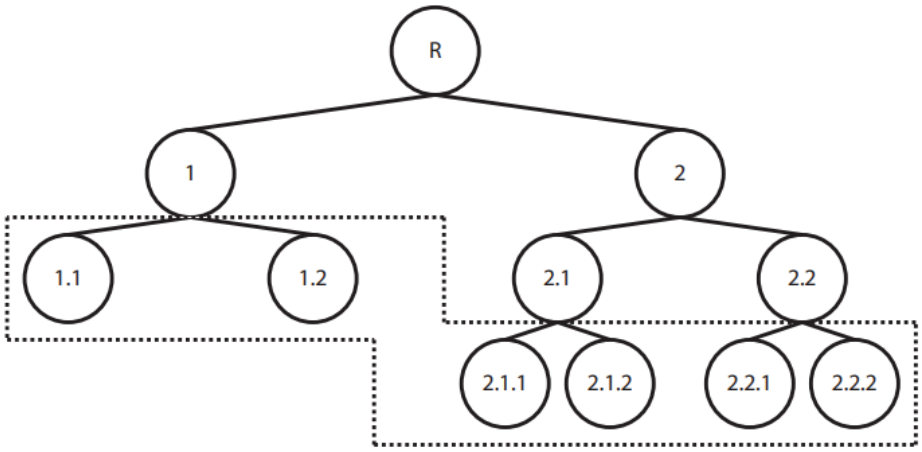

Figure 5 Flat, multi-class classification approach [41]

Local Classifier Per Node Approach: This approach is one of the popular approaches in hierarchical classification approaches. In this approach, for each node on the hierarchy, there is a binary classifier that needs to be trained individually.

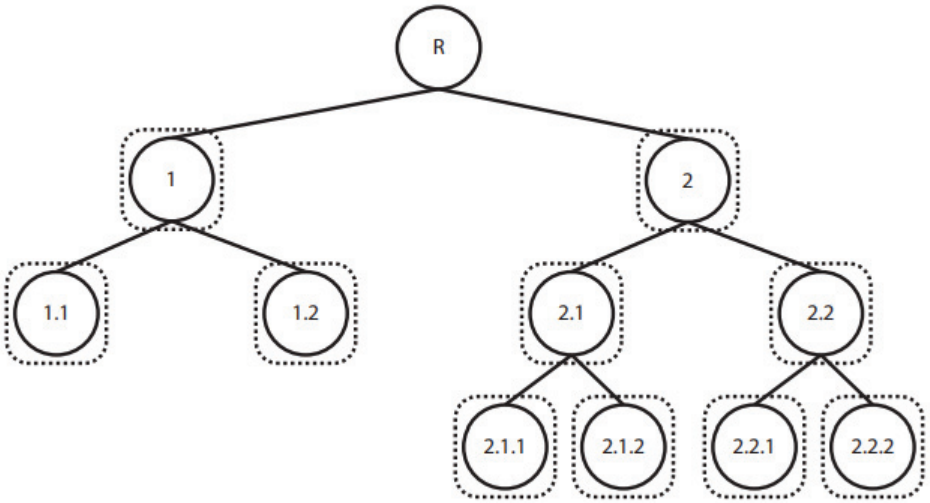

Figure 6 Local classifier per node approach circles are classes and rectangles are binary classifiers. [41]

Local Classifier Per Parent Node Approach: In this approach, for each parent node in the hierarchy, there is a multi-class classifier that predicts the child nodes.

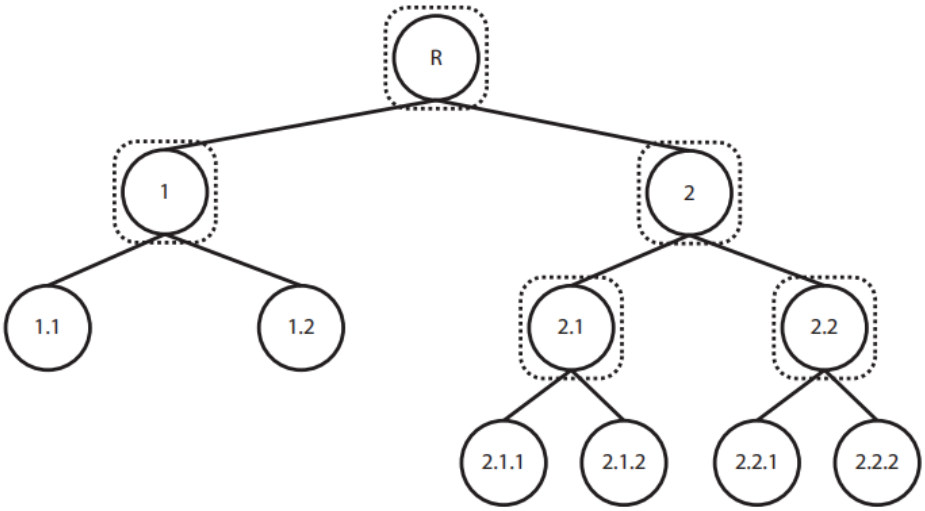

Figure 7 Local classifier per parent node approach. Circles are classes and squares are multi-class classifiers [41]

Local Classifier Per Level Approach : In this approach for each level in the hierarchy, there is a single multi-class classifier.

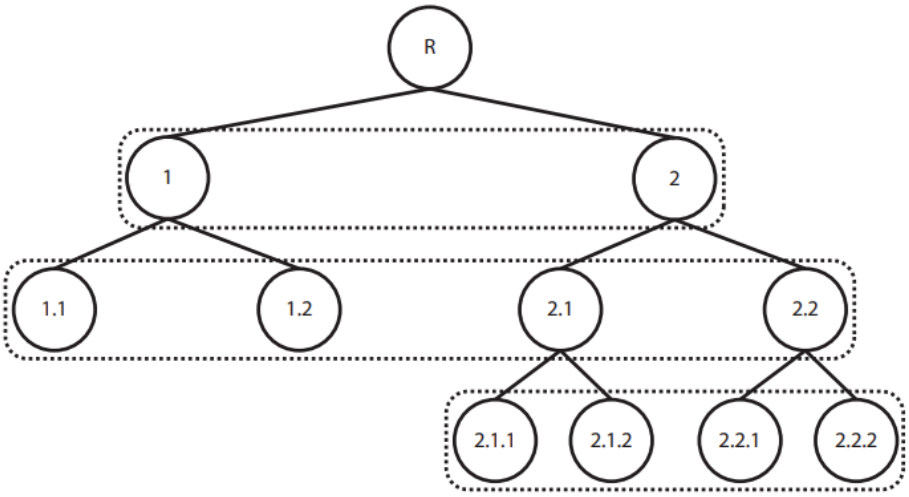

Figure 8 Local classifier per level approach circles are classes rectangles are multi-class classifier [41]

Global Classifier Approach : Although there are different local approaches that work well for hierarchical classification problems, there is another approach that uses a single classifier for the entire model. There are some major advantages for using this model such as the size of models described before are much bigger than this approach model.

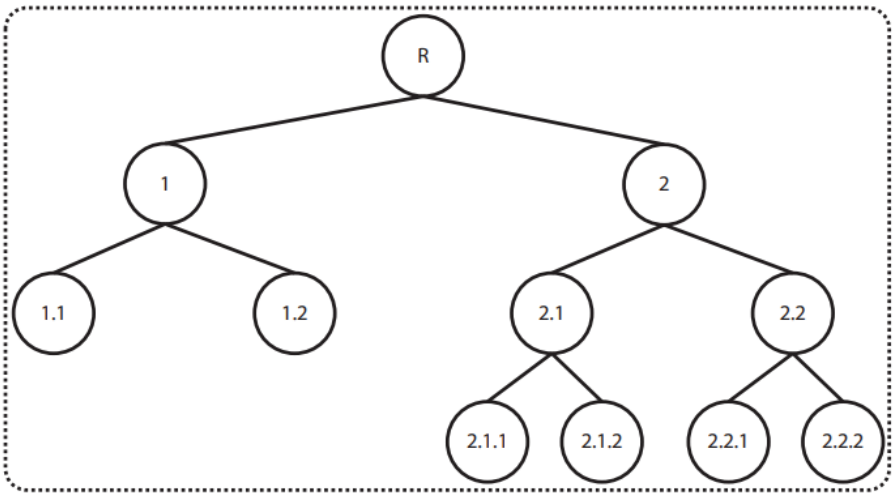

Figure 9 Global classifier approach [41]

2.3 Vision Transformers

Vision transformers [4] have become popular in recent years and emerged as an alternative to CNN. As CNN's are currently state of the art in computer vision for image classification tasks, vision transformers outperform CNN in terms of computational cost and accuracy.

Transformer architecture is mostly used in NLP [42] and Vision tasks [43]. The attention feature is generally used with CNNs or used to substitute certain aspects of CNN while keeping their entire composition intact. There is no certain rule about the dependency on CNNs. There is some research that applies vision transformers directly to images for classification tasks. Vision Transformers have achieved highly competitive performance in image classification [44], image segmentation [45], and object detection [46].

Vision transformers accept input images as image patches like word series in NLP. CNN's are using pixels as input but vision transformers split images into some special visual tokens. T first split images into patches and then embeds each of them. The resulting positional embedding becomes an input to the transformer encoder.

In ViT there is a layer called the self-attention [47] layer. This layer makes it possible to embed information globally across the image. The model also learns how to reconstruct the image during the training phase. The encoder includes some different structured layers that are:

- Multi-Head Self Attention Layer: This layer concatenates attention outputs to the related dimensions.
- Multi-Layer Perceptron Layer: This layer contains two-layer with Gaussian Error Linear Unit
- Layer Norm: This layer is added to each block and it does not include any new dependencies. With this layer training time and performance will be improved.

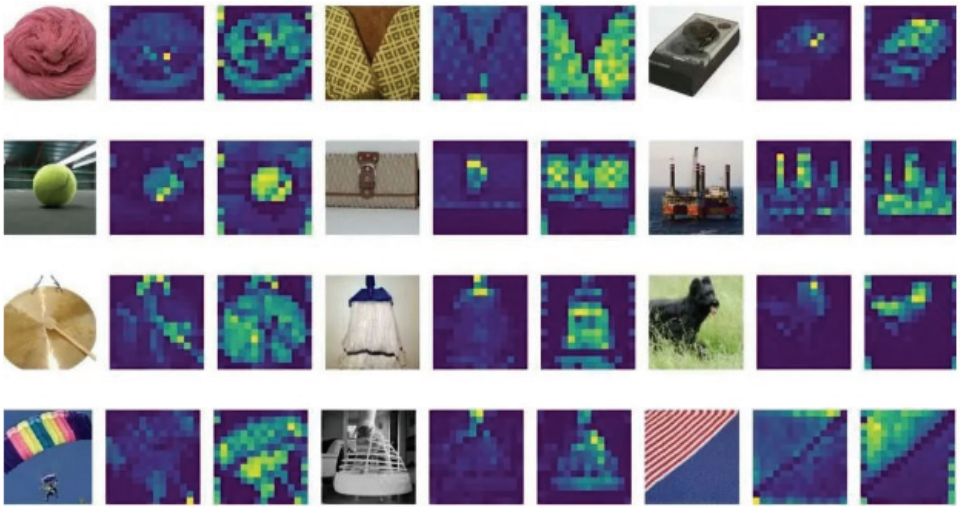

Figure 10 Images with attentions. [47]

Self-attention [47] is one of the most important blocks of vision transformers. It helps the network to learn hierarchies and alignments present inside the input dataset. The workflow of a vision transformer is like:

- Generate fixed-sized patches from images
- Flatten the patches and obtain a vector
- Create linear embeddings from patch vectors
- Include positional embeddings
- Give resulting sequence to transformer encoder
- Pre-Train the ViT model
- Fine-Tuning

There is no work done before that combines the vision transformers, similarity and hierarchy information for image classification. But there are some similar works that covers some of the technologies that we used in our work.

Decision-making is important in some critical areas such as medical and military. Behnaz Gheflati and Hassan Rivaz [44] proposed a method for classifying breast cancer using Vision transformers. As there are not much data in their domain, they applied the transfer learning method to the pre-trained ViT and compared the results with the CNN approaches. As we can see from their approach, ViT architecture achieved comparable performance with other CNN approaches.

Jun Wang et al. [48] also proposed an architecture that integrates ViT into fine-grained visual categorization problems. They designed a module that selects the important

tokens based on the self-attention scores from each transformer layer and obtains local, low and mid-level information.

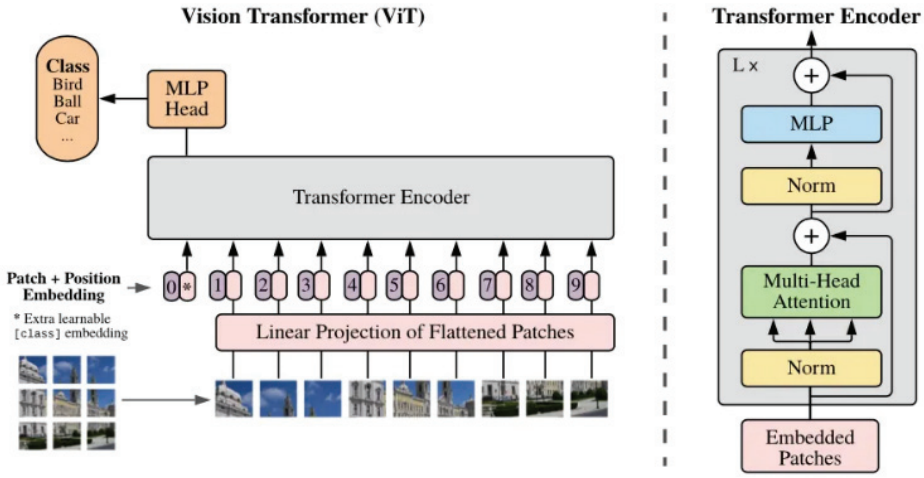

Figure 11 ViT architecture [49]

CHAPTER 3

EXPERIMENTS

3.1 Used Technologies

All the technologies that are used during this study are listed below:

- **Python:** Python is an interpretive language, meaning it can be run without the need for compilation, unlike languages such as C and C++. In Python you do not need to code as much as C or C++ because most of the data structures and tools are already defined. With the help of python we can create programs faster.
- **PyTorch:** PyTorch is the Python library [50] for building deep learning models that leverage the power of graphics processing units. It is to build deep neural networks in tensor computations and tape-based auto grade systems with powerful GPU acceleration support. One of the main reasons behind the success of PyTorch is that it is purely Pythonic and can build neural network models without any problems. PyTorch, which uses graphics processing units in projects, is extremely popular today with the flexibility and speed it provides due to its structure.
- **NumPy:** NumPy is a library that is used in most of the ML tasks because of easy multidimensional array operations.
- **DINO:** This is a project that is developed by the Facebook AI team, to train Vision Transformers (ViT) with no supervision [47].
- **Ubuntu:** Ubuntu is an open source and free operating system developed based on the Linux kernel.
- **Hardware:**
    - MSI VGA GEFORCE RTX 2080TI
    - MSI MEG X570 ACE AM4
    - AMD RYZEN 9 3950X
    - CORSAIR 32GB (2x16GB) DDR4 4000MHz RAM
    - CORSAIR 960 GB MP510 NVMe SSD

3.2 Dataset Description

The real-world dataset named “ETH Entomological Collection (ETHEC)” [3] is handled during this study. The dataset contains images of Lepidoptera specimens with their taxonomy tree. There is an imbalance in the number of images per class and also in the taxonomical trees. The authors of the dataset wanted to represent real-world scenarios and make the dataset mode realistic compared to ImageNet which has balanced classes. This imbalanced makes the image classification task more challenging and realistic.

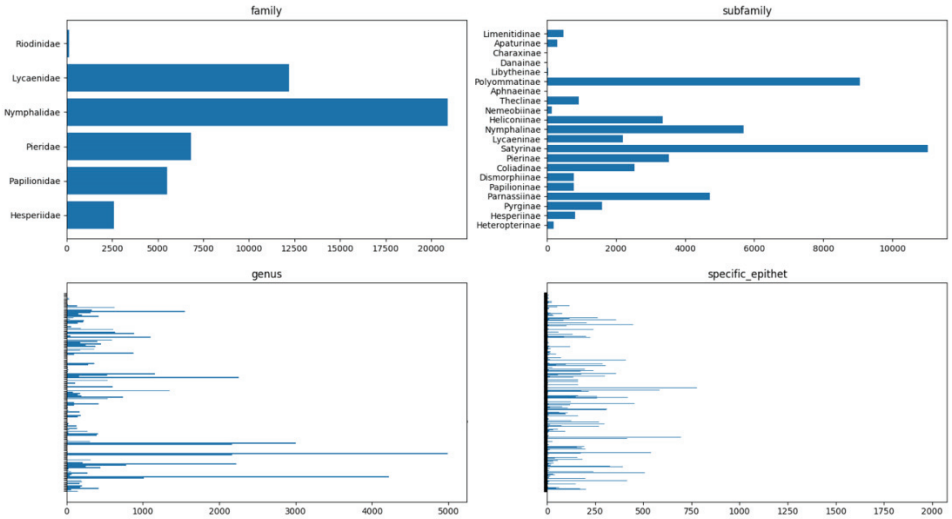

Figure 12 Image distribution over labels and classes. [2]

The dataset contains which contains 41.350 images and 712 classes have 4 different levels of hierarchy that are :

- Level 1 : 6 Family class
- Level 2 : 21 Sub-Family class
- Level 3 : 135 Genus class
- Level 4 : 550 Species class

Figure 13 Example images with their levels. [2]

The images in the dataset are collected from ETH Entomological Collection. During the digitization process, all the texts, marks, barcodes and etc. are removed to eliminate unwanted information. The images are center cropped with dimension 448 x 448 and the authors provide detailed metadata information for 4 levels of the hierarchy.


```
"f8bac77d-2d46-4fe2-bcc1-2c781585ed49": {
    "token": "f8bac77d-2d46-4fe2-bcc1-2c781585ed49",
    "image_path": "2017_01_30R",
    "image_name": "ETHZ_ENT01_2017_01_30_000064.JPG",
    "family": "Hesperiidae",
    "subfamily": "Heteropterinae",
    "genus": "Carterocephalus",
    "specific_epithet": "palaemon",
    "subspecific_epithet": "",
    "infraspecific_epithet": "",
    "author": "(Pallas, 1771)",
    "country": "Switzerland",
    "primary_division": "Bern",
    "dec_lat": 46.9479774284,
    "dec_long": 7.447437875,
    "barcode": "ETHZ-ENT0003493"
},
"e61eb251-94c6-4e51-a560-a382df0fe415": {
    "token": "e61eb251-94c6-4e51-a560-a382df0fe415",
    "image_path": "2017_01_30R",
    "image_name": "ETHZ_ENT01_2017_01_30_000065.JPG",
    "family": "Hesperiidae",
    "subfamily": "Heteropterinae",
    "genus": "Carterocephalus",
    "specific_epithet": "palaemon",
    "subspecific_epithet": "",
    "infraspecific_epithet": "",
    "author": "(Pallas, 1771)",
    "country": "Switzerland",
    "primary_division": "Zurich",
    "dec_lat": 47.43205,
    "dec_long": 8.5634,
    "barcode": "ETHZ-ENT0003494"
},
```


Figure 14 Example metadata information

3.3 Data Preprocessing

The dataset is divided into multiple subdirectories that contain unordered data. So the first step was organizing these datasets. To do that I first create a python script to read metadata and copy the images into the related train or validation folder. As the metadata is already split into train and validation, we did not need to split the metadata.

Figure 15 ETHEC dataset folder unorganized

The organized dataset contains 36.283 images for the train folder and 5067 images for the validation folder. The dataset contains unequal image distribution. After splitting the dataset into train and validation sets, some of the classes have only 2 or 3 images for training.

For each level of the hierarchy, I prepared different datasets without any augmentation to be able to make some experiments. As the training and validation images are the same (obtained from metadata file), the difference between these datasets is the output classes. The prepared datasets are:

- **ETHEC-Fam**: Level-1 of the hierarchy. There are 6 classes that are family.
- **ETHEC-Sub:** Level-2 of the hierarchy. There are 21 classes that are sub-family.

- **ETHEC-Gen:** Level-3 of the hierarchy. There are 135 classes that are genus.
- **ETHEC-Spec:** Level-4 of the hierarchy. There are 550 classes that are species

Table 1 Prepared datasets for experiments. “Level” represents the level of hierarchy. “Num. Classes" represent the total number of classes in that level of hierarchy. “Train images” and “Valid images” represent the number of images for training and validating. “Min Image” represents the number of minimum images in a single class of the related level of hierarchy. “Max Image” represents the number of maximum images in a single class of the related level of hierarchy. As you can see, there are lots of differences in the number of images between classes even they belong to the same level of hierarchy.

No	Dataset Name	Level	Num. Classes	Train Images	Valid Images	Min. Image	Max. Image
1	ETHEC-Fam	L1	6	36.283	5.067	113	16.063
2	ETHEC-Sub	L2	21	36.283	5.067	2	8.716
3	ETHEC-Gen	L3	135	36.283	5.067	1	3.933
4	ETHEC-Spec	L4	550	36.283	5.067	1	489

3.4 Obtaining Attention

In this study, instead of using image pixels for classification, we intend to use visual attention. To be able to do this, we used the DINO repository which provides us

with a self-supervised vision transformer. There are different ViT architectures that DINO [47] provides us.

Table 2 DINO ViT model parameters. [47]

model	blocks	dim	heads	#tokens	#params	im/s
ResNet-50	–	2048	–	–	23M	1237
ViT-S/16	12	384	6	197	21M	1007
ViT-S/8	12	384	6	785	21M	180
ViT-B/16	12	768	12	197	85M	312
ViT-B/8	12	768	12	785	85M	63

We used the ViT-S/16 model for our experiments. The model implements a multi-head attention technique and you can see the 6 different self-attention masks and heatmaps for the same image in Figure 20 and Figure 21.

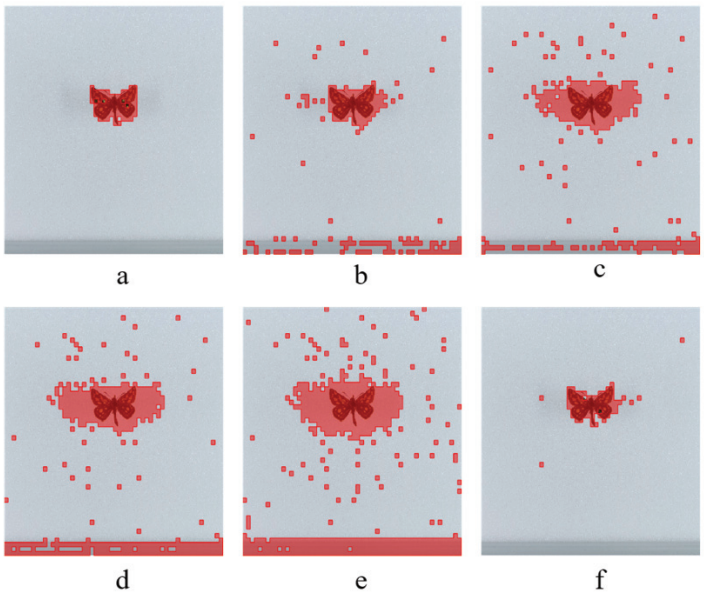

Figure 16 Self-attention mask of a butterfly for 6 different heads. Image a represents mask for head 1, image b represents mask for head 2, image c represents mask for head 3, image d represents mask for head 4, image e represents mask for head 5, and image f represents mask for head 6

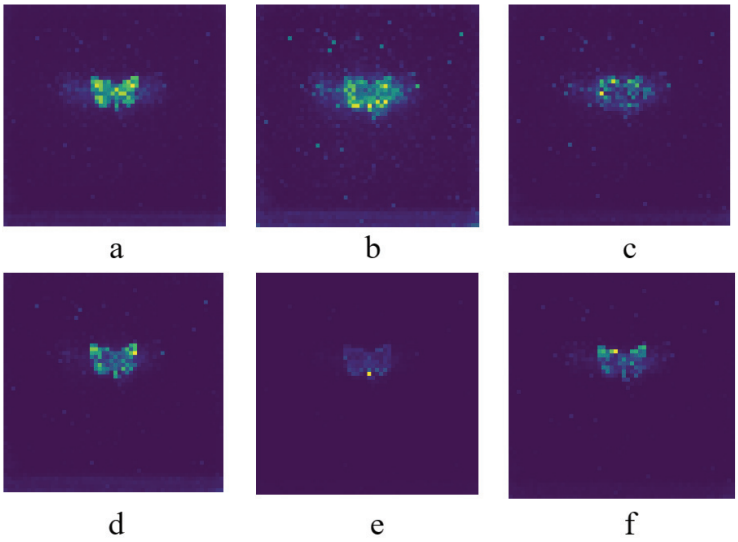

Figure 17 Self-attention heatmap of a butterfly for 6 different heads. Image a represents heatmap for head 1, image b represents heatmap for head 2, image c represents heatmap for head 3, image d represents heatmap for head 4, image e represents heatmap for head 5, and image f represents heatmap for head 6

3.4 Experiments

All the experimental results are discussed in this section. To be able to compare our results, we selected a reference paper that uses the same dataset and implements hierarchical classification on CNN-based approaches. They implemented 3 different approaches called Per-Level (PLC), Marginalization classifier (MC) and Masked Per-level classifier (M-PLC) [2]. Each approach implements different hierarchical classification techniques.

3.4.1 K-Nearest Neighbor Experiment With ViT Features

In this experiment, we just developed a simple K-NN that accepts ViT features as an input. With that, we aimed to question if the ViT features of different hierarchy levels of the ETHEC dataset can be classified successfully by a K-NN. With the help of similarity metric, we can calculate top k neighbors.

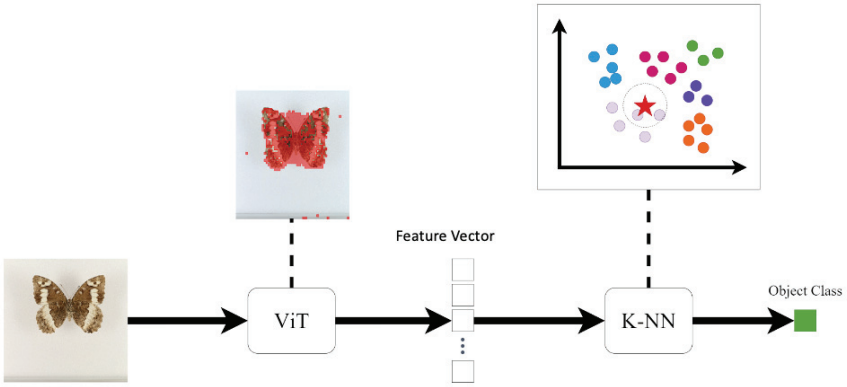

Figure 18 Experiment-1 diagram. An image first feed into the ViT and then obtained Feature Vector feed into the K-NN to obtain image class.

ETHEC-Fam:

ETHEC-Fam is the first dataset that represents the first level of the hierarchy. Classes of this level contain a higher number of images rather than the other levels of hierarchy. Table 3 shows that the K-NN with ViT features worked well with the ETEC-Fam dataset.

Table 3 ETHEC-Fam results on K-NN with ViT features.

Experiment No	K Value	Top 1	Top 5
1	5	%99.62	%99.92
2	10	%99.52	%99.96
3	15	%99.42	%99.96
4	20	%99.38	%99.96

ETHEC-Sub:

ETHEC-Sub is the second dataset that represents the second level of the hierarchy. Table 4 shows that the K-NN with ViT features worked well with the ETEC-Sub dataset.

Table 4 ETHEC-Sub results on K-NN with ViT features.

Experiment No	K Value	Top 1	Top 5
1	5	%98.79	%99.78
2	10	%98.75	%99.84
3	15	%98.53	%99.86
4	20	%98.36	%99.86

ETHEC-Gen:

ETHEC-Gen is the third dataset that represents the third level of the hierarchy. Table 5 shows that the K-NN with ViT features worked well with the ETEC-Gen dataset.

Table 5 ETHEC-Gen results on K-NN with ViT features.

Experiment No	K Value	Top 1	Top 5
1	5	%94.61	%98.20
2	10	%94.07	%98.79
3	15	%93.54	%98.91
4	20	%92.75	%98.85

ETHEC-Spec:

ETHEC-Spec is the last dataset that represents the last level of the hierarchy. As expected, Table 6 shows that the results of the K-NN with ViT features are not high as upper-level datasets in the hierarchy.

Table 6 ETHEC-Spec results on K-NN with ViT features.

Experiment No	K Value	Top 1	Top 5
1	5	%82.92	%92.73
2	10	%81.72	%94.43
3	15	%80.24	%94.80
4	20	%79.00	%94.78

Table 7 Comparison of datasets wrt. Scores

Dataset	K Value	Top 1	Top 5
ETHEC-Fam	10	%99.52	%99.96
ETHEC-Sub	15	%98.53	%99.86
ETHEC-Gen	15	%93.54	%98.91
ETHEC-Spec	15	%80.24	%94.80

Table 7 shows us that K-NN performance decreased as we went through lower levels. This kind of effect is actually predictable. In some classes (especially in lower levels of hierarchy), the amount of data is really low and this data problem decreases the overall accuracy of the model.

3.4.2 Per-Level Classifier Experiments

In this experiment, we implemented a similar architecture to Per-Level Classifier called VA-PLC and VM-PLC. The main principle for this architecture is that for every level of hierarchy there is a classifier, and the entire model predicts L different labels in which L is the number of levels in the hierarchy.

Basically, in our approach, there is a K Nearest Neighbor classifier for each level that is trained with vision transformer features. We used this K-NN classifier to obtain cosine similarities for each level. Then we concatenated these similarity vectors into a single vector and feed to an SVM as an input for the final decision.

For this approach, the first step was obtaining vision transformer features of the entire dataset. As we have four different levels in the hierarchy, we prepared four different oriented feature set from the same vision transformer for each level. To do this we feed the ETHEC-Fam, ETHEC-Sub, ETHEC-Gen, and ETHEC-Spec datasets into a vision transformer and saved the obtained feature arrays like the diagram below.

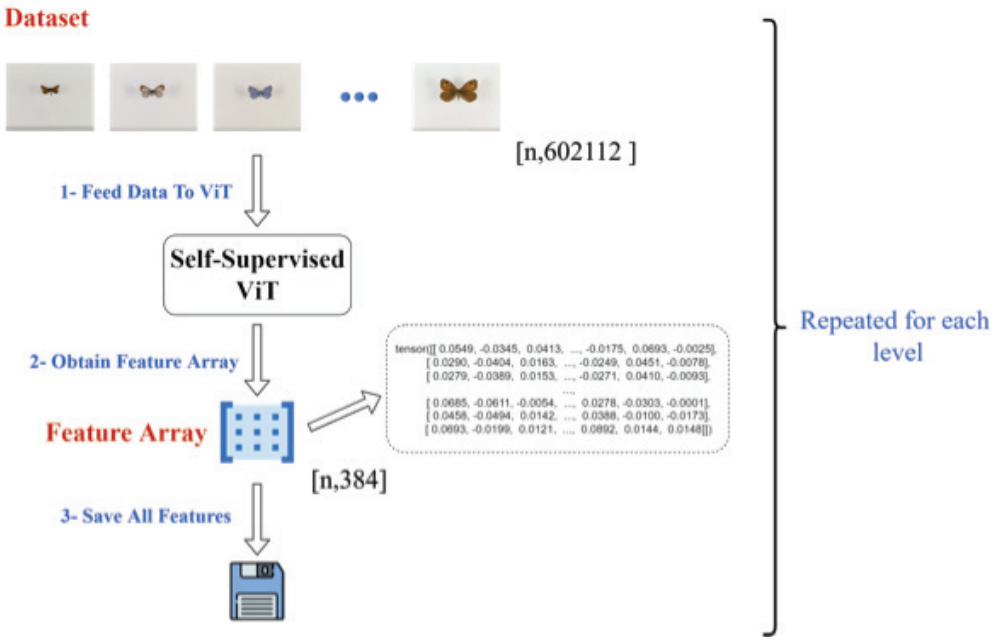

Figure 19 Obtaining ViT features for each level of the hierarchy

As the datasets for each level of the hierarchy are organized in a different way, the obtained feature arrays are not sorted. As we want to obtain the similarity value of a

single image in the dataset for each level, unsorted arrays make calculations difficult and the search operation consumes lots of time.

Figure 20 Unsorted ViT Feature arrays for each level

To be able to sort all of these features, we created the feature sorter module that sort the Level 2, Level 3, and Level 4 feature array with respect to the orientation of the Level 1 feature array (Figure 21).

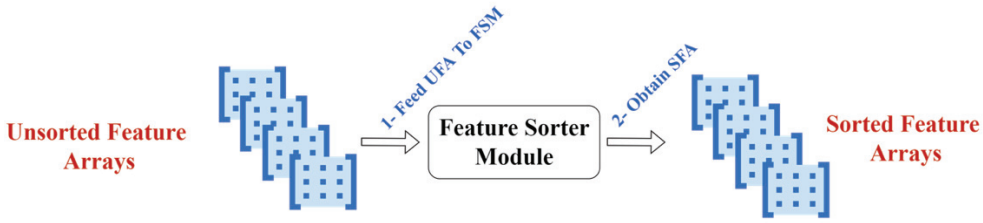

Figure 21 Feature sorter module

After sorting all the feature arrays for each level into a single orientation, we can easily apply matrix operations as all the indexes of these feature arrays are belongs to the same image in the dataset (Figure 22).

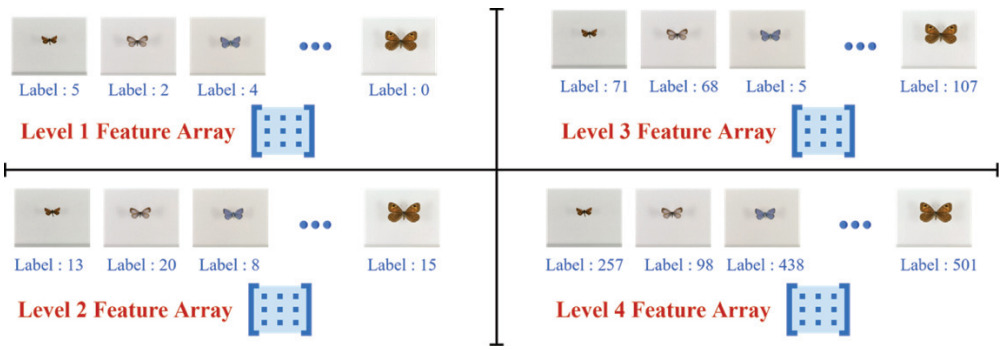

Figure 22 Sorted ViT feature arrays for each level

After obtaining sorted feature arrays, the next step was obtaining the similarity vectors. The term similarity vector is that, for all items in the validation dataset, we obtain a similarity vector that holds the cosine similarity value of the item in the validation dataset to all items in the training dataset. In cosine similarity, the returned values are located between 0 and 1. If two feature vectors are completely different the function returns 0 otherwise if two vectors are identical then it returns 1 [29] .

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

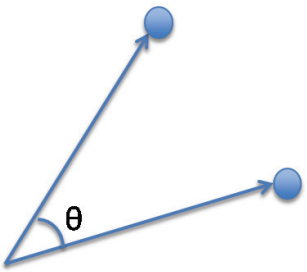

Figure 23 Cosine similarity formula

To be able to obtain this similarity vector, we prepared the Feature To Similarity Module (Figure 24).

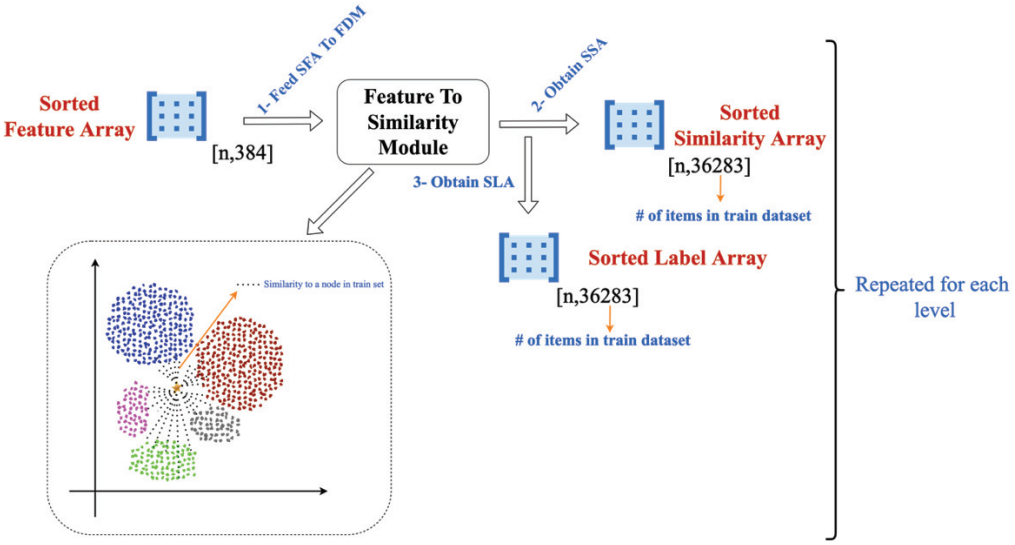

Figure 24 Feature to Similarity module

The Feature To Similarity Module calculates the similarity array based on a K-NN classifier's similarity metrics for each level in the dataset. The module outputs 2 different arrays that are Sorted Similarity Array (which holds the similarity values) and

Sorted Candidate Array (which holds the label of each similarity value in the Sorted Similarity Array). These arrays are saved into a file for future steps to be able to save some time.

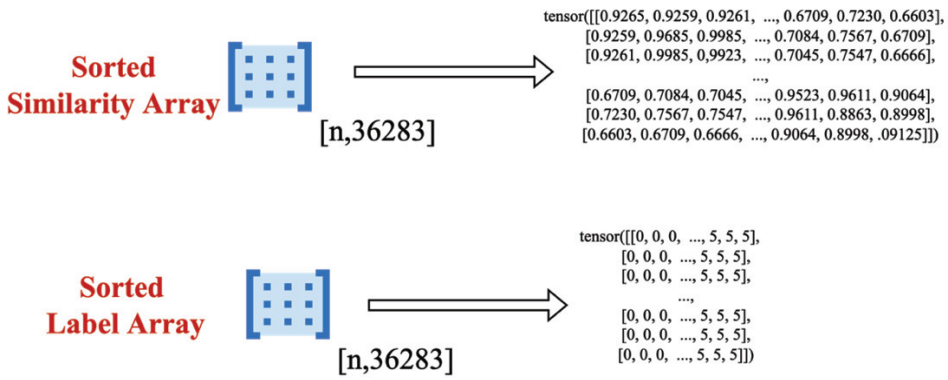

Figure 25 Sorted Similarity And Label Array for Level 1 dataset

As you can see from the figure above there are multiple similarity values for a single label. What we want to obtain is a single similarity value for each label. To be able to eliminate the other similarity values we prepared 2 different approaches that are:

- Average Similarity Per Label
- Maximum Similarity Per Label

The main difference between these approaches is that in the Average Similarity Per Label approach we calculated the average of all similarities that belongs to the same label but in the Maximum Similarity Per Label approach, we just select the maximum similarity value for each label.

3.4.2.1 Average Similarity Per Label

For this approach, we prepared a module called Average Similarity Per Label Calculator. This module basically gets the Sorted Similarity Array and Sorted Label Array as an input and outputs the Average Similarity Array for each level in the hierarchy (Figure 26).

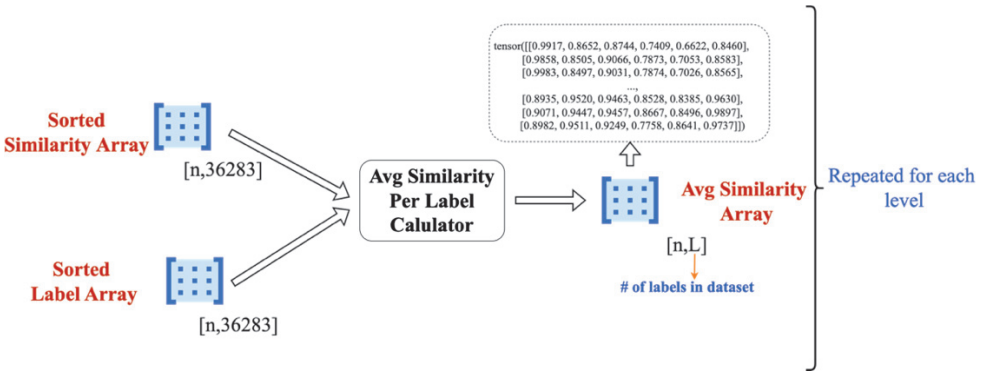

Figure 26 Average Similarity Per Label Calculator module

To be able to obtain average similarities, we grouped the similarity values for each label and calculated their average values. Then we created an Average Similarity Array that holds a single similarity value for each label. In this approach, there are some similarity values that can be classified as outliers so to be able to eliminate them we used K biggest similarity values for each label (Figure 27).

Figure 27 Average Similarity Per Label Calculator module in detail

In the previous experiment, we trained K-NNs that represent different levels of hierarchy. For each K-NN we selected the suitable K value with the help of Table 7.

Table 8 Trained K-NNs for each level of hierarchy

Dataset	K Value	K-NN
ETHEC-Fam	10	L1 K-NN
ETHEC-Sub	15	L2 K-NN
ETHEC-Gen	15	L3 K-NN
ETHEC-Spec	15	L4 K-NN

After obtaining the Average Similarity Arrays for each level of the hierarchy, we need to concatenate these similarity arrays into a single one to be able to obtain a single feature array that holds all the information for all levels in the hierarchy. To be able to concatenate these arrays, we created the ASA Concatenation Module (Figure 28).

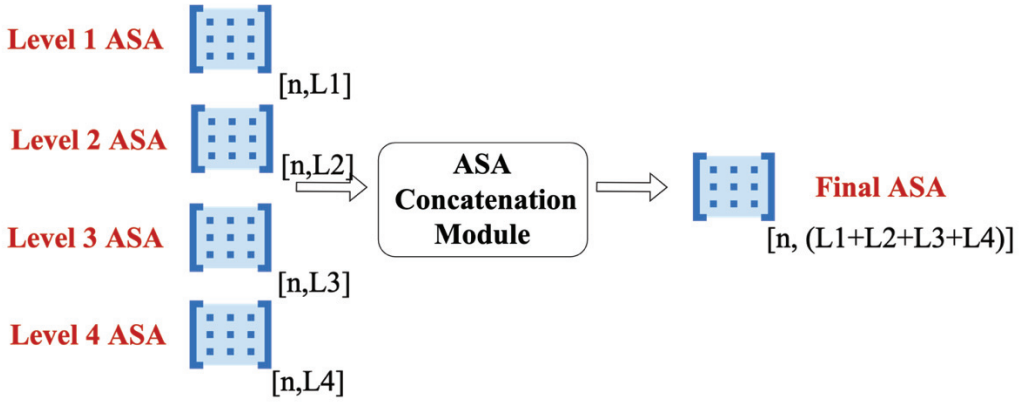

Figure 28 Concatenating Average Similarity Arrays for each level in to a Final Average Similarity Array

The ASA Concatenation Module basically accepts all Average Similarity Arrays for each level and merges them in the order of Level 1 to Level 4. The final array holds the Similarity information for each label in the entire hierarchy which has a length of 713 (Figure 29).

Figure 29 ASA Concatenation Module

After applying all these steps that are described above, we obtained a Final Average Similarity Array for each image in the dataset and as we already know their actual for each level, we trained 4 different SVMs that classify for all different labels in the hierarchy (Figure 30).

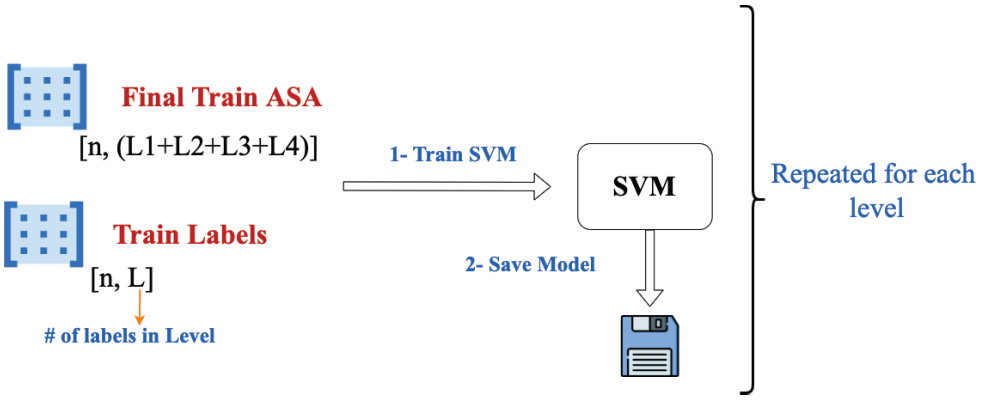

Figure 30 Training SVM for each level in the hierarchy

After training the SVMs, for each SVM we fed our Final Validation Average Similarity Array and obtained the L1, L2, L3 and L4 results.

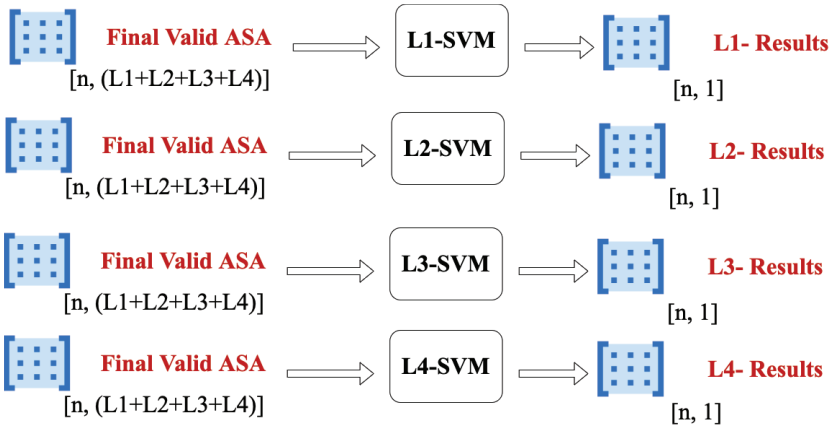

Figure 31 Validation of the VA-PLC approach

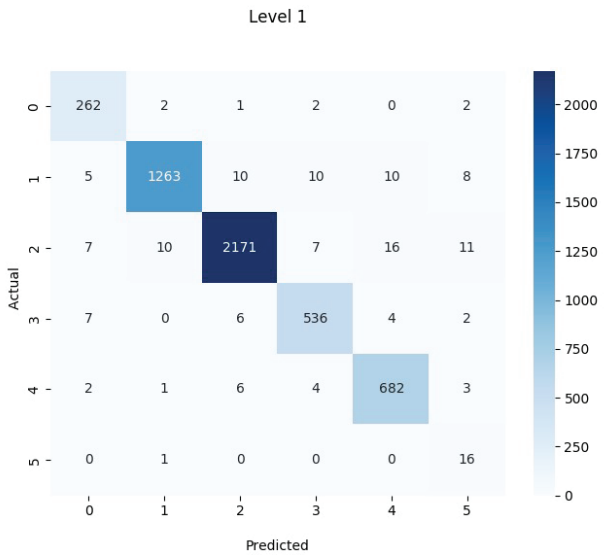

Figure 32 Confusion matrix of the VA-PLC approach level 1

Most of the binary classification problems can be evaluated with precision, recall and F1 score metrics but as our problem is a multiclass classification problem, we need to calculate the TP, FN, FP and TN values from the confusion matrix.

- TP: True positive, actual value and the predicted value is the same
- FN: False negative, sum of values of all rows except TP
- FP: False positive, sum of values of all columns except TP
- TN: True negative, sum of values of all columns and rows except values of the related class

For calculating the metrics, we used:

- Precision: tp / (tp + fp)
- Recall: tp / (tp + fn)
- F1 Score: 2 * (precision * recall) / (precision + recall)

Table 9 VA-PLC Accuracy, Precision, Recall and F1 results

Metric	Avarage	L1	L2	L3	L4
Accuracy	%90,43	%97,43	%96,59	%91,18	%76,54
Precision	%67,57	%85,03	%79,97	%64,63	%40,66
Recall	%69,44	%96,80	%90,80	%63,97	%43,70
F1	%69,44	%90,92	%83,12	%63,00	%40,74

Table 10 VA-PLC and PLC model results on different levels.

Model	m-F1	L1	L2	L3	L4
VA-PLC	%90,43	%97,43	%96,59	%91,18	%76,54
PLC	%90,84	%97,66	%96,61	%92,04	%77,04

As you can see from the Table 9 VA-PLC can not outperform the PLC model but obtained quite similar results. In this approach, even we select the best K value for averaging the top K similarity values, there are still some outlier similarity values that decrease the overall accuracy. Also as there are unequally distributed data, the precision and recall values are decreased a lot in level 3 and level 4.

3.4.2.1 Maximum Similarity Per Label

For this approach, we prepared a module called Maximum Similarity Per Label Calculator (Figure 33). This module basically gets the Sorted Similarity Array and Sorted Candidate Array as an input and outputs the Maximum Similarity Array for each level in the hierarchy.

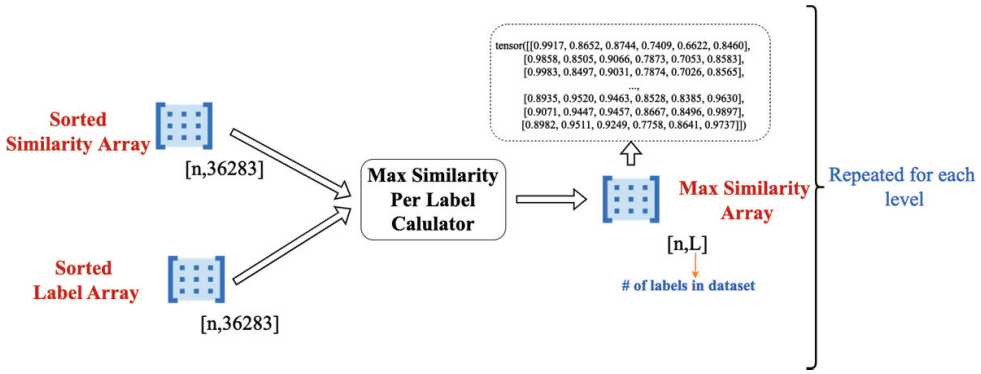

Figure 33 Maximum Similarity Per Label Calculator Module

To be able to obtain maximum similarities, we grouped the similarity values for each label and selected the maximum value from them. Then we created a Maximum Similarity Array that holds a single similarity value for each label (Figure 34).

Figure 34 Maximum Similarity Per Label Calculator Module in detail

After obtaining the Maximum Similarity Arrays for each level of the hierarchy, we need to merge these similarity arrays into a single one to be able to obtain a single feature array that holds all the information for all levels in the hierarchy. To be able to concatenate these arrays, we created the MSA Concatenation Module which is exactly the same module as ASA Concatenation Module (Figure 35).

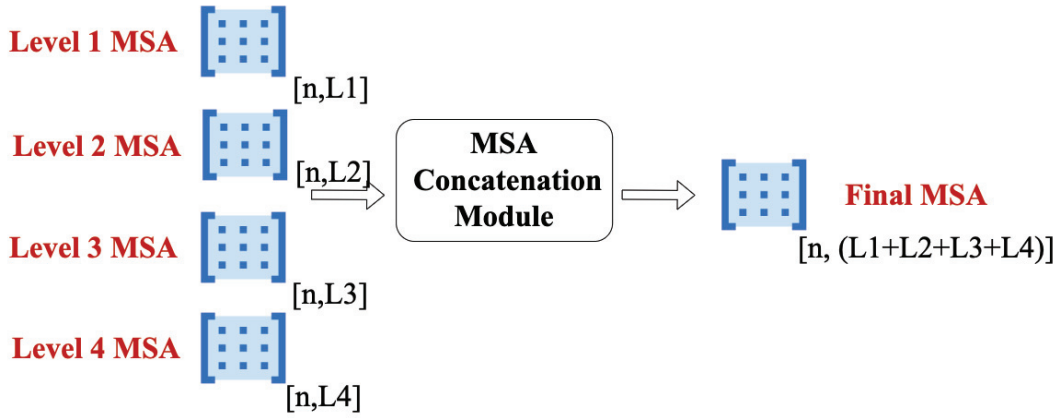

Figure 35 Concatenating Maximum Similarity Arrays for each level into a Final Average Similarity Array

The MSA Concatenation Module basically accepts all Maximum Similarity Arrays for each level and merges them in the order of Level 1 to Level 4. The final array holds the similarity information for each label in the entire hierarchy which has length of 713 (Figure 36).

Figure 36 MSA Concatenation Module

After applying all these steps that are described above, we obtained a Final Maximum Similarity Array for each image in the dataset and as we already know their actual results for each level, we trained 4 different SVMs that classifies for all different

labels in the hierarchy as we did for the Average Similarity Value Approach before (Figure 37).

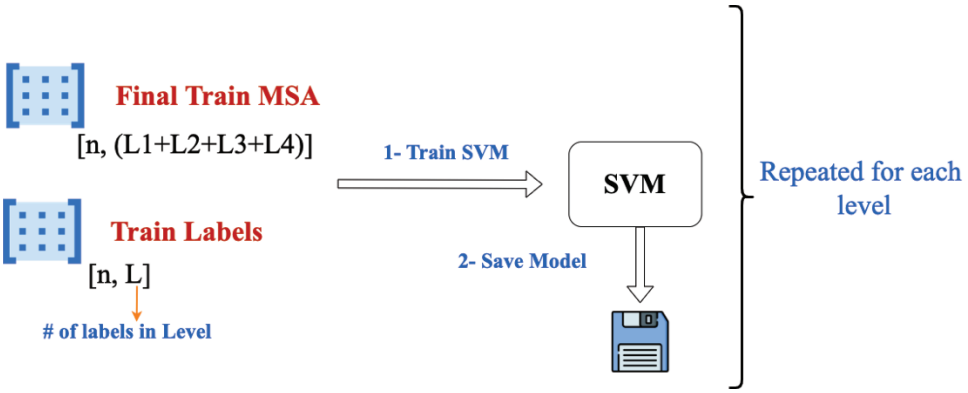

Figure 37 Training SVM for each level in the hierarchy

After training the SVMs, for each SVM we fed our Final Validation Maximum Similarity Array and obtained the L1, L2, L3, and L4 results (Figure 38).

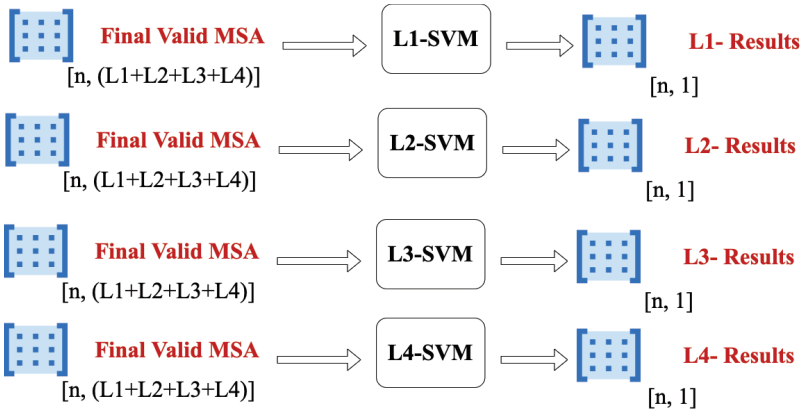

Figure 38 Validation of the VM-PLC approach

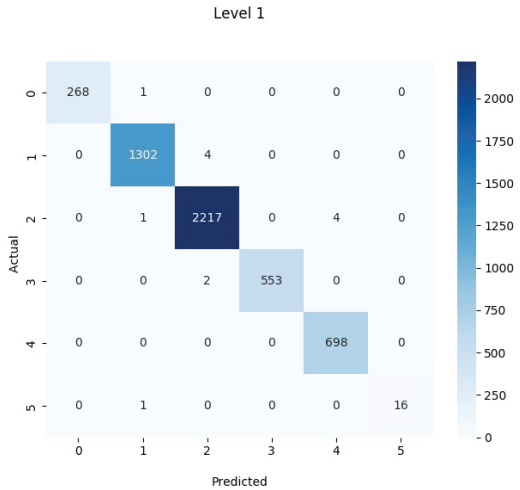

Figure 39 Confusion matrix of the VM-PLC approach level 1

Table 11 VM-PLC Accuracy, Precision, Recall and F1 results

Metric	Avarage	L1	L2	L3	L4
Accuracy	%92,84	%99,74	%98,79	%93,58	%79,25
Precision	%76,38	%99,82	%94,16	%69,98	%42,56
Recall	%75,62	%98,81	%92,70	%65,56	%45,44
F1	%75,40	%99,30	%93,37	%66,37	%42,57

Table 12 VA-PLC, VM-PLC and PLC model results on different levels.

Model	m-F1	L1	L2	L3	L4
VA-PLC	%90,43	%97,43	%96,59	%91,18	%76,54
VM-PLC	%92,84	%99,74	%98,79	%93,58	%79,25
PLC	%90,84	%97,66	%96,61	%92,04	%77,04

As you can see from Table 11 VM-PLC outperforms the PLC model with high results even in the 4th level. And also, there is an improvement on precision and recall compare to VA-PLC approach. The precision and recall are still bad in last two levels that shows us in some classes our model still predicts completely wrong.

There are some problems with this approach that are :

- This approach only exploits the number of levels in the hierarchy
- Not a safe approach. Classification in lower levels is not accurate as higher levels and as there is no sub-tree relation, the overall accuracy can be affected in a bad way.

3.4.3 Masked Per-Level Classifier Experiments

In this experiment, we implemented a similar architecture to Masked Per-Level Classifier which is called VM-MPLC. The main principle for this architecture is that for every level of hierarchy there is a classifier, and the entire model predicts L different

labels. The difference from previous approach is that implausible classes are masked so the model do not need to deal with classes in which their parent classes are not predicted in upper level of hierarchy.

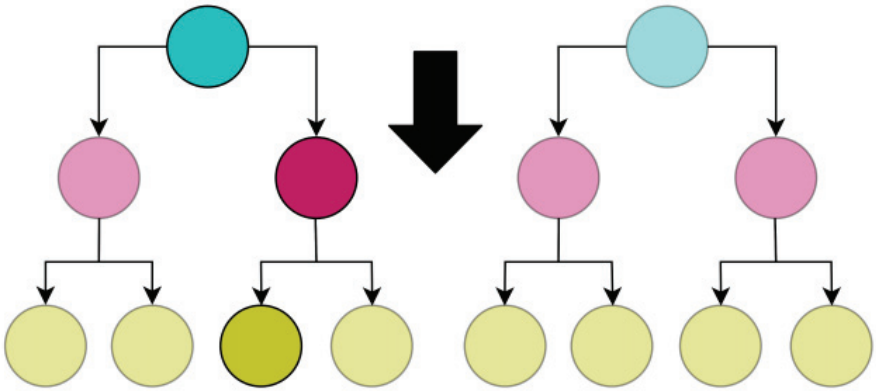

Figure 40 Example representation of masked classes in different levels of hierarchy [2]

Basically, in our approach, there is a K Nearest Neighbor classifier for each level that is trained with vision transformer features. We used this K-NN classifier to obtain similarities and predictions for each level. Then we eliminated the implausible classes and merged the masked similarity arrays into a single vector. Similar to previous approaches we trained 4 SVMs for each level and obtain final results.

For this approach, we used the sorted vision transformer feature arrays for each level that we obtained from the previous experiment. We updated the Feature to Similarity Module that we used in our previous experiment so that we can also obtain the K-NN prediction array as well as Sorted Similarity Array and Sorted Label Arrays (Figure 41).

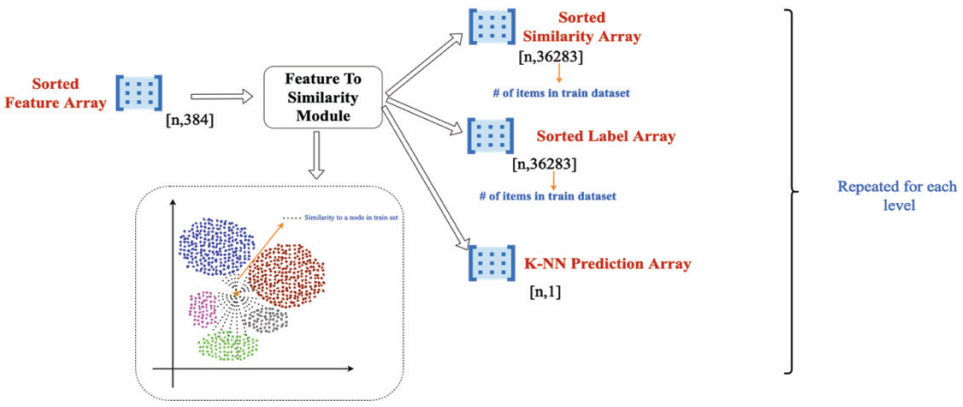

Figure 41 Feature to Similarity module

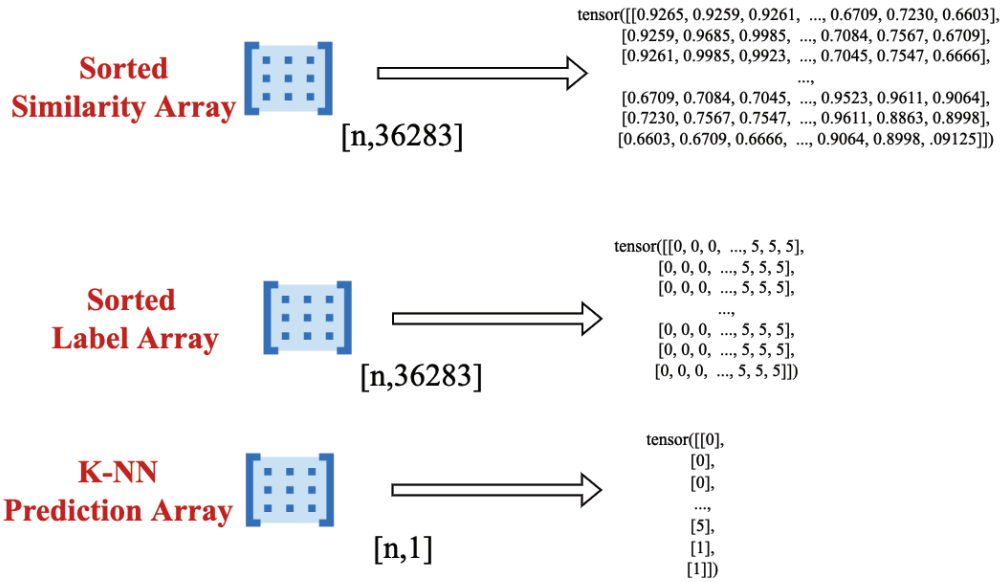

Figure 42 Feature To Similarity Module Output Arrays

In previous approaches, we do not mask any labels as the hierarchy goes deeper. In this approach, we will mask the implausible labels. To do this we prepared a module called Child Filter Module (Figure 43). Basically, this module gets the K-NN prediction array of the upper level and Sorted Similarity Array of the current level as an input and outputs the Filtered Sorted Similarity Array for each level of the hierarchy.

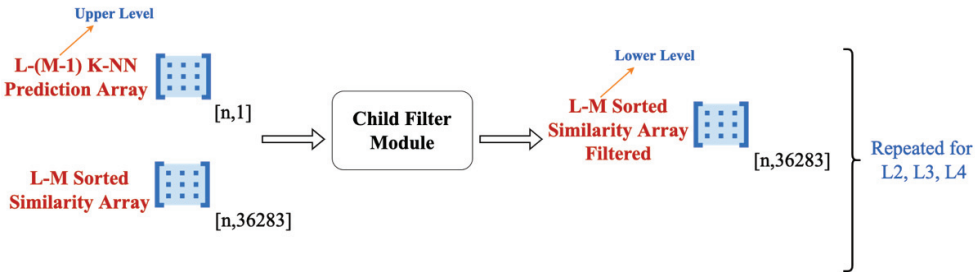

Figure 43 Child Filter Module

For this module, we prepared a map that holds the parent-child relationship. With this information, we can easily eliminate the nonchild elements in lower levels. To eliminate these nonchild elements we set their similarity value to 0. With this operation, the module outputs an array that contains only the child elements similarity values as their real values and the rest of them are 0 (Figure 44).

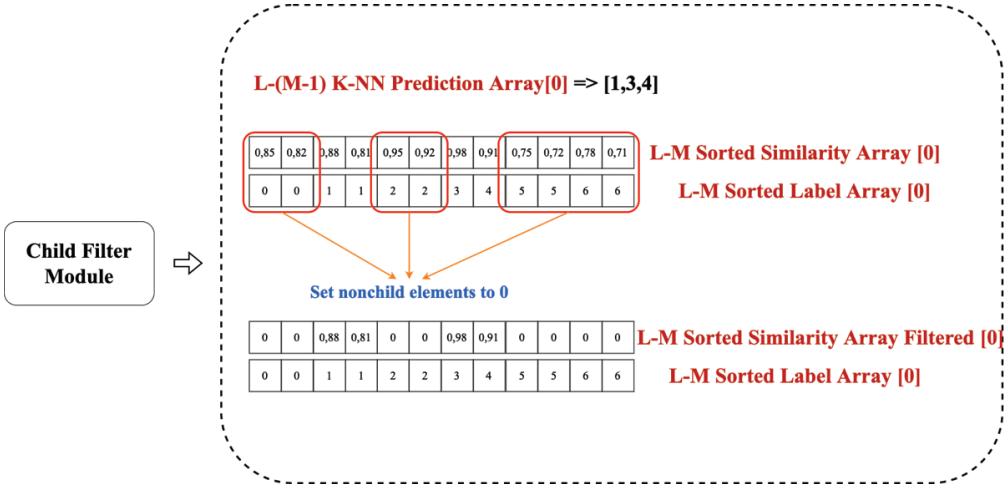

Figure 44 Child Filter Module in detail

After obtaining the Filtered Sorted Similarity Array, We used the Max Similarity Per Label Calculator (Figure 45) as described in the previous experiment. The only difference is that the maximum operation is only applied to child elements (Figure 46).

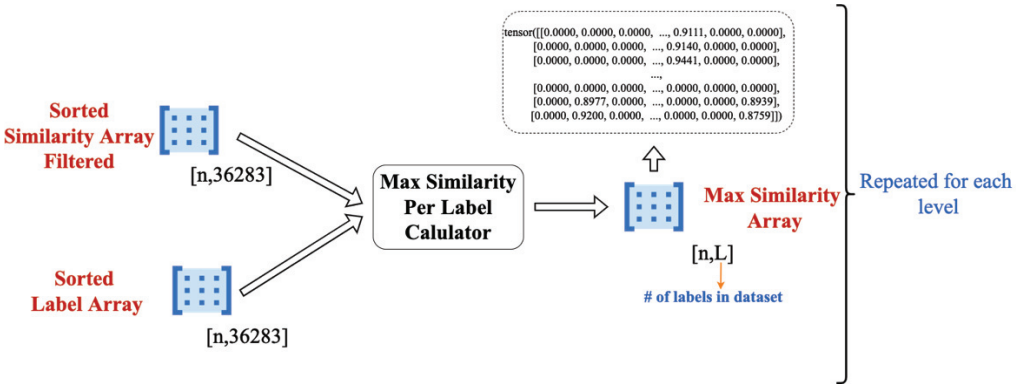

Figure 45 Max Similarity Per Label Calculator with nonchild elements

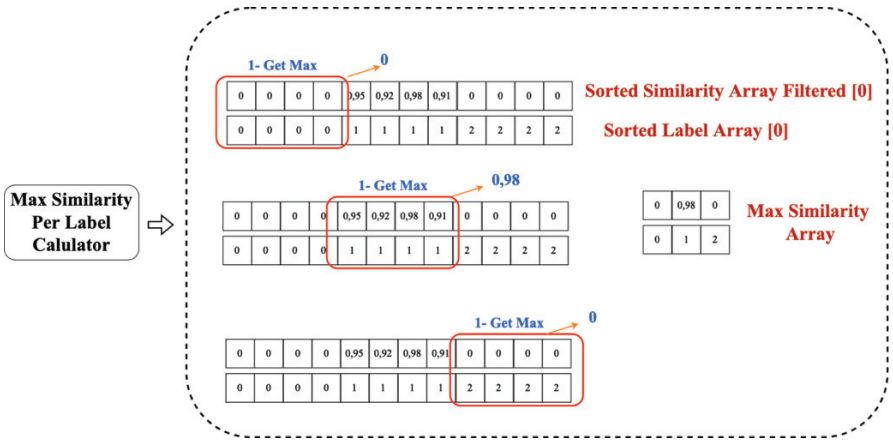

Figure 46 Max Similarity Per Label Calculator with nonchild elements in detail

The rest of this approach is the same as the VM-PLC approach. We used these Filtered Maximum Similarity Arrays to train SVMs for each level of the hierarchy and after that, we obtained the L1, L2, L3, and the L4 results.

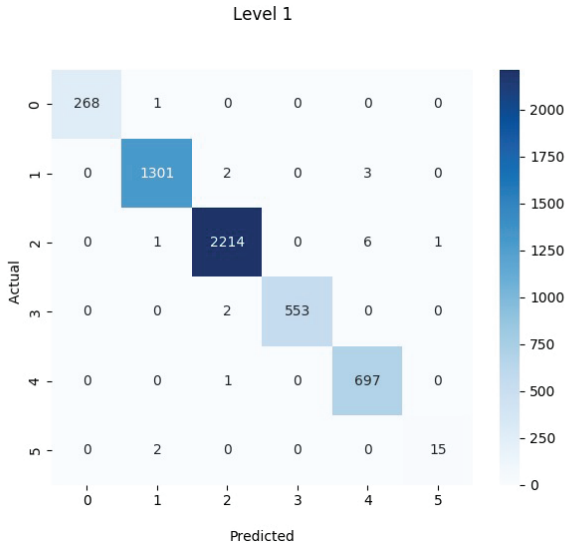

Figure 47 Confusion matrix of the VM-MPLC approach level 1

Table 13 VM-MPLC Accuracy, Precision, Recall and F1 results

Metric	Avarage	L1	L2	L3	L4
Accuracy	%92,96	%99,63	%98,81	%94,57	%78,85
Precision	%77,65	%99,66	%93,79	%74,87	%42,36
Recall	%76,60	%97,77	%93,27	%70,37	%45,00
F1	%76,39	%98,20	%93,51	%71,60	%42,27

Table 14 VM-MPLC results

Model	m-F1	L1	L2	L3	L4
VA-PLC	%90,43	%97,43	%96,59	%91,18	%76,54
VM-PLC	%92,84	%99,74	%98,79	%93,58	%79,25
VM-MPLC	**%92,96**	**%99,63**	**%98,81**	**%94,57**	**%78,85**
M-PLC	%91,73	%98,28	%97,01	%92,33	%79,30

3.4.5 Experiment Summary

We have implemented 3 different experiments and compared our results with the reference paper. In experiments section I described the first two implementations that are PLC and M-PLC but also there is another implementation called MC which results the best score in the reference paper.

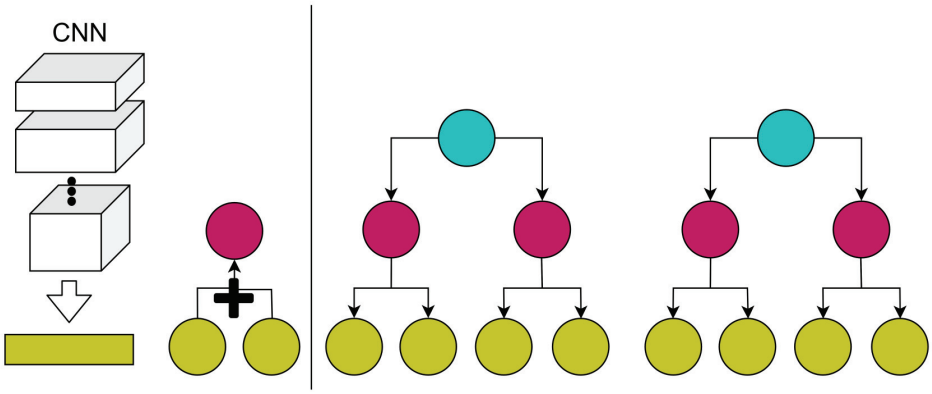

Figure 48 Marginalization approach [2]

This approach basically makes the prediction with a probability distribution over the leaf nodes and the non-leaf node probability is determined by the marginalization method. We did not implement this approach in our work currently but this will be future work for us.

V-CNN is an additional experiment that aims to check that if training a deep learning model with ViT features increases the accuracy.

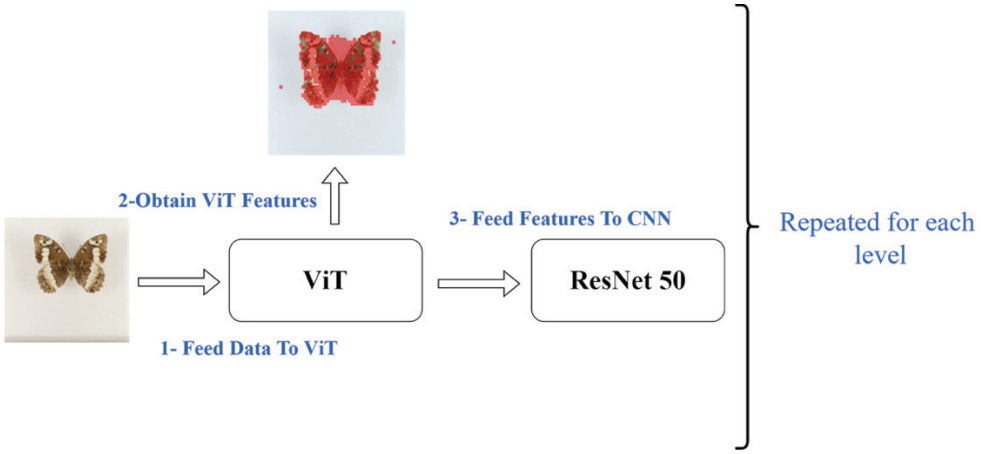

Figure 49 V-CNN approach

As you can see from the results this approach not worked well. We used a pretrained ResNet 50 model and trained 4 different model for each level with different training parameters (Table 16).

Table 15 V-CNN Accuracy and training time results

Dataset	Epoch	Validation Acc	Training Time (min)
ETHEC-Fam	30	%98.36	45
ETHEC-Sub	30	%83.69	45
ETHEC-Gen	50	%52.21	100
ETHEC-Spec	100	%32.6	170

As you can see from the Table 16, hierarchical image classification with self-supervised Vision Transformer features outperforms traditional hierarchical CNN.

Table 16 Experiment Results

Model	m-F1	L1	L2	L3	L4
VA-PLC	%90,43	%97,43	%96,59	%91,18	%76,54
VM-PLC	%92,84	%99,74	%98,79	%93,58	%79,25
VM-MPLC	**%92,96**	**%99,63**	**%98,81**	**%94,57**	**%78,85**
V-KNN	%92,95	%99,52	%98,53	%93,54	%80,24
V-CNN	%66,71	%98,36	%83,69	%52,21	%32,6
PLC	%90,84	%97,66	%96,61	%92,04	%77,04
M-PLC	%91,73	%98,28	%97,01	%92,33	%79,30
MC	**%92,23**	**%98,87**	**%97,58**	**%92,73**	**%79,72**

Instead of the performance advantages, K-NN approaches that we discussed, There are some other advantages of K-NN that helps us to develop a hierarchical classification system that are:

- **No Training Required:** K-NN just stores the training dataset. There is no training or deriving discriminative functions. It learns while making a prediction. For a hierarchy, as we need lots of small models, K-NN makes us save lots of time.
- **New Data Integration:** As K-NN does not require any training, we can easily add or extract data without any major problem.
- **Easy To Implement:** There are only two hyperparameters that we can change that is K value and similarity formula. For a hierarchy, as we need lots of small models, we do not need to spend time for fine-tunning as much as CNN fine-tunning.
- Also there are some disadvantages that are :
- **Small datasets:** K-NN operates well in smaller datasets like ETHEC but in larger datasets as the computation cost increases the performance may decrease.
- **Test and Prediction Times:** As K-NN stores the entire training data, when the data increases the test or prediction time also increases linearly. On the other hand with the CNN approach, as the number of trainable parameters such as weights is not dependent on the training dataset size, the prediction times are always the same. This might be mitigated by using Approximate Nearest Neighbor techniques [51] .

CHAPTER 4

CONCLUSION AND FUTURE WORK

In this work, we proposed 3 different methods to exceed hierarchical CNNs. The main problem with hierarchical CNNs is that they require lots of data, computational cost, training time and etc. Also, they are hard to optimize and deploy. We used K-NNs that are trained with self-supervised vision transformers to obtain similarities and for the classification, we trained an SVM for each level. We compared our approaches with the CNN approaches from our reference paper with respect to accuracy and 2 of our approaches that are VM-PLC and VM-MPLC outperformed the best CNN-based approach of the reference paper which is MC.

As we used a K-NN to obtain similarities, there is no need for training. We just train a simple SVM for each level of the hierarchy. Also, there is no need for training an SVM for each level of the hierarchy if the problem is just covering only one level of the hierarchy. So with our approach we solved the problems that are:

- High amount of data requirement
- High training times
- Optimization problems
- Not extendable hierarchy

In the future work, we are planning to improve our approach and implement some different hierarchical classification techniques such as local classifier per parent node, local classifier per node, and so on. As we are not using CNNs, number of local classifiers will be not hard to train and deploy. In literature, local classifiers seem more accurate than the global ones so with these implementations we are planning to obtain higher accuracies with bigger datasets.

REFERENCES

[1] Y. Gavish, J. O'Connell, C. J.Marsh, C. Tarantino, P. Blonda, V. Tomaselli and W. E.Kunin, "Comparing the performance of flat and hierarchical Habitat/Land-Cover classification models in a NATURA 2000 site," 2018.

[2] A. Dhall, A. Makarova, O. Ganea, D. Pavllo, M. Greeff and A. Krause, "Hierarchical Image Classification using Entailment Cone Embeddings," 2020.

[3] Dhall and Ankit, "ETH Entomological Collection (ETHEC) Dataset," 2019.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2020.

[5] T. Evgeniou and M. Pontil, "SUPPORT VECTOR MACHINES: THEORY AND APPLICATIONS," 2001.

[6] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," 2021.

[7] H. Yessou, G. Sumbul and B. Demir, "A COMPARATIVE STUDY OF DEEP LEARNING LOSS FUNCTIONS FOR MULTI-LABEL REMOTE SENSING IMAGE CLASSIFICATION," 2020.

[8] S. Sun, Z. Cao, H. Zhu and J. Zhao, "A Survey of Optimization Methods from a Machine Learning Perspective," 2019.

[9] J. Brownlee, "14 Different Types of Learning in Machine Learning," 2019. [Online]. Available: https://machinelearningmastery.com/types-of-learning-in-machine-learning/. [Accessed 2022].

[10] L. Rokach and O. Maimon, "Desicion Trees," 2005.

[11] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers," 2007.

[12] E. Grossi and M. Buscema, "Introduction to artificial neural networks," 2008.

[13] S. Katoch, S. S. Chauhan and V. Kumar, "A review on genetic algorithm: past, present, and future," 2020.

[14] M. Mayo, "Frameworks for Approaching the Machine Learning Process," 2018. [Online]. Available: https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html. [Accessed 2022].

[15] S. Parveez, "Machine Learning Standardization (Z-Score Normalization) with Mathematics," 2020. [Online]. Available: https://towardsai.net/p/machine-learning/machine-learning-standardization-z-score-normalization-with-mathematics. [Accessed 2022].

[16] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," pp. 28-30, 2020.

[17] S. Hotta, S. Kiyasu and S. Miyahara, "Pattern recognition using average patterns of categorical k-nearest neighbors," 2004.

[18] G. Amato and F. Falchi, "KNN based image classification relying on local feature similarity," 2010.

[19] D.A.Adeniyi, Z.Wei and Y.Yongquan, "Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method," 2016.

[20] H. Saadatfar, S. Khosravi, J. H. Joloudari and A. Mosavi, "A New K-Nearest Neighbors Classifier for Big Data Based on Efficient Data Pruning," 2020.

[21] A. K. Ghosh, T. Rahman and M. M. H. Shuvo, "Mental Stress Recognition using K-Nearest Neighbor (KNN) Classifier on EEG Signals," 2015.

[22] C. Li, S. Zhang, H. Zhang, L. Pang, K. Lam, C. Hui and S. Zhang, "Using the K-Nearest Neighbor Algorithm for the Classification of Lymph Node Metastasis in Gastric Cancer," 2012.

[23] Z. Yao and W. L. Ruzzo, "A Regression-based K nearest neighbor algorithm for gene function prediction from heterogeneous data," 2006.

[24] N. Ali, D. Neagu and P. Trundle, "Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets," 2019.

[25] A. Choudhury and M. R. Kosorok, "Missing Data Imputation for Classification Problems," 2020.

[26] Y. Zhongguo, L. Hongqi, L. Qiang and Z. Liping, "A case based method to predict optimal k value for k-NN algorithm," 2017.

[27] V. B. S. Prasath, H. A. A. Alfeilat, A. B. A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat and H. S. E. Salmane, "Effects of Distance Measure Choice on KNN Classifier Performance - A Review," 2019.

[28] I. José, "KNN (K-Nearest Neighbors)," 2018. [Online]. Available: https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d. [Accessed 2022].

[29] D. Srivastava and L. Bhambhu, "Data classification using support vector machine," 2010.

[30] D. Basak, S. Pal and D. C. Patranabis, "Support Vector Regression," 2007.

[31] A. Patle and D. S. Chouhan, " SVM kernel functions for classification," 2013.

[32] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed 2022].

[33] B. Knyazev, M. Drozdzal, G. W. Taylor and A. Romero-Soriano, "Parameter Prediction for Unseen Deep Architectures," 2021.

[34] J. Brownlee, "A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks," 2018. [Online]. Available: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/. [Accessed 2022].

[35] C. Janiesch, P. Zschech and K. Heinrich, "Machine learning and deep learning," 2020.

[36] A. Kayid and Y. Khaled, "Performance of CPUs/GPUs for Deep Learning workloads," 2018.

[37] C. Brewster, "14 Companies Using Machine Learning," 2021. [Online]. Available: https://trio.dev/blog/companies-using-machine-learning. [Accessed 2022].

[38] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," 2021.

[39] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He, "A Comprehensive Survey on Transfer Learning," 2019.

[40] D. Martinez, " Is Transfer Learning the final step for enabling AI in Aviation?," 2020. [Online]. Available: https://datascience.aero/transfer-learning-aviation/. [Accessed 2022].

[41] C. N. S. Jr. and A. A. Freitas, "A Survey of Hierarchical Classification Across Different Application Domains," 2011.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," 2017.

[43] Y. Bazi, L. Bashmal, M. M. A. Rahhal, R. A. Dayil and N. A. Ajlan, "Vision Transformers for Remote Sensing Image Classification," 2021.

[44] B. Gheflati and H. Rivaz, "VISION TRANSFORMERS FOR CLASSIFICATION OF BREAST ULTRASOUND IMAGES," 2022.

[45] A. Sagar, "ViTBIS: Vision Transformer for Biomedical Image Segmentation," 2022.

[46] H. Song, D. Sun, S. Chun, V. Jampani, D. Han, B. Heo, W. Kim and M.-H. Yang, "ViDT: An Efficient and Effective Fully Transformer-based Object Detector," 2021.

[47] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers," 2021.

[48] J. Wang, X. Yu and Y. Gao, "Feature Fusion Vision Transformer for Fine-Grained Visual Categorization," 2021.

[49] P. J. P., "Vision Xformers: Efficient Attention for Image Classification," 2021.

[50] Wikipedia, "Pytorch," 2022. [Online]. Available: https://en.wikipedia.org/wiki/PyTorch. [Accessed 2022].

[51] M. Muja and D. G. Lowe, "FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATION," 2009.

[52] S. Makki, R. Haque, Y. Taher, Z. Assaghir, M.-S. Hacid and H. Zeineddine, "A Cost-Sensitive Cosine Similarity K-Nearest Neighbor for Credit Card Fraud Detection," 2019.

[53] Y. Hoshen and L. Wolf, "Unsupervised Correlation Analysis," 2018.

[54] J. F. Hair, W. C. Black and B. J. Babin, Multivariate Data Analysis A Global Perspective, 2010.

[55] J. Brownlee, "Difference Between Classification and Regression in Machine Learning," Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/. [Accessed 2022].

[56] M.-C. Su, D.-Y. Huang, C.-H. Chou and C.-C. Hsieh, "A reinforcement-learning approach to robot navigation," 2004.

[57] K. Shao, Z. Tang, Y. Zhu, N. Li and D. Zhao, "A Survey of Deep Reinforcement Learning in Video Games," 2019.