University of West Bohemia in Pilsen

Faculty of Applied Sciences

The Department of Cybernetics

Diploma thesis

Pilsen 2023 Bc. Lukáš Kölbl

Prohlášení

Předkládám tímto k posouzení a obhajobě Diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni. Prohlašuji, že jsem Diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

Statement

I hereby submit for assessment and defense the Diploma thesis prepared at the end of my studies at the Faculty of Applied Sciences of the University of West Bohemia in Pilsen. I declare that I prepared the Diploma thesis independently and exclusively using professional literature and sources, the complete list is included.

In Pilsen May 2023 Lukáš Kölbl

Poděkování

Velmi děkuji svému vedoucímu Ing. Gruber Ivan, Ph.D. et Ph.D. za jeho neustálou podporu, angažovanost a cennou zpětnou vazbu. Jeho vedení a rady mě provedly všemi fázemi psaní mé práce.

Acknowledgments

I'm extremely grateful to my supervisor Ing. Gruber Ivan, Ph.D. et Ph.D. for his continuous support, engagement and valuable feedback. His guidance and advice carried me through all the stages of writing my thesis.

Abstract

In recent years, the use of automatic surveillance devices called camera traps has become widespread. These devices are used to collect information needed to monitor different ecosystems. With the acquisition of a large amount of data, there is a time burden on researchers and the need for new ways to process this data. This diploma thesis is dealing with the use of artificial intelligence and computer vision as a tool for effective processing of images from camera traps. In this thesis, we participate in the competition, from which an image dataset of animals occurring in a national park in Central Africa is available. To obtain classified images, we use convolutional neural network models and the Vision Transformer. The best results are achieved using an ensemble of Vision Transformer models and other image processing methods.

KeyWords: classification, computer vision, camera traps, convolutional neural network, Vision Transformer

Abstrakt

V posledních letech se rozšířilo používání automatických sledovacích zařízení nazývaných fotopasti. Tyto zařízení slouží ke sběru informací potřebných ke sledování různých ekosystémů. Se získáním velkého množství dat dochází k časovému zatížení výzkumníků a k potřebě nových způsobů jak tyto data zpracovávat. V této diplomové práci se budeme zabývat použitím umělé inteligence a počítačového vidění jako nástroje k efektivnímu zpracování obrazů z fotopastí. V rámci této práce jsme se zúčastnili soutěže, z které bude k dispozici dataset obrazů zvířat vyskytujících se v národním parku ve střední Africe. V této práci využijeme modelů konvolučních neuronových sítí a Vision tranformeru pro získání klasifikovaných obrazů. Nejlepší výsledky byly dosaženy použitím ensemblu Vision tranformer modelů a dalších metod zpracování obrazu.

Klíčová slova: klasifikace, počítačové vidění, fotopasti, konvoluční neuronové sítě, Vision Transformer

Contents

Chapter 1

Introduction

We rely on our planet's ecosystem every day. Whether it is healthy forests that provide accommodation for various types of animals, they also help store carbon that would otherwise be released into the atmosphere [1]. Oceans provide food and livelihood for millions of people around the planet. Animals and insects pollinate crops intended for consumption [2]. Natural habitats serve as a means of income for low-resource communities. With the progressive action of humans, the global climate is changing. There are huge effects and changes to wildlife. By protecting natural habitats, we can prevent the spread of diseases between humans and animals and prevent the extinction of populations of various animals [3]. Good tools and methods are needed to observe and understand these problems.

The health of these ecosystems depends on a diverse and complex network of plants and animals known as biodiversity. Today, we face a biodiversity crisis. Tracking species in the wild is a common component of conservation efforts to monitor and maintain biodiversity. Camera traps are one of the best resources we have for researching wildlife populations [4].

This thesis will focus on the use of machine learning and computer vision to make the best use of the camera traps just mentioned. The goal will be to investigate and validate different computer vision methods and procedures for the classification of images provided by camera traps. This data is used to find out how animal populations change over time or what effect human presence has on wildlife. Processing and dividing data from a large number of camera traps is tedious and time-consuming. Even an expert can take tens of seconds to process an unknown image. However, this time could be used more constructively by an expert on the given problem. It is also known that monotonous activity leads to errors, even in a relatively simple problem. Classification of unknown images is very monotonous. Classification can be simple for one image and, conversely, very difficult for another. This is the part for which it is appropriate to use computer vision and artificial intelligence, which can quickly process a large number of images. With the right application, it is then possible to process these images in real time.

As the performance of computers increases, so does the possibility of applying artificial intelligence with greater complexity and for more difficult and diverse tasks. Especially with the growing popularity of neural networks and their variants, which are proving to be a suitable tool for solving problems in the field of computer vision. In recent years, a lot of competitions and various institutions have been created that try to encourage the combination of ecology and artificial intelligence [5].

In this thesis, we are participating in one such competition, where we will try to look at the given problem from the point of view of computer vision. The most important parts of code produced during this thesis are available on GitHub[1], the link is in the footnotes.

1.1 Task description

Creating a model that will be suitable for this competition can be divided into three parts. The first is that the model should be able to generalize well with the possibility of applying it to new data that researchers have not seen yet, thereby ensuring usefulness for other projects. The second criterion is simple application and use so that even researchers that are not experts in computer science are able to use this model and apply it to their problem. The third condition, the selected methods for designing the model should be technically motivated, so that the best possible results are achieved and users know what behavior and what results they can expect when using it.

To ensure all these conditions, we will try to implement different types of models and methods for their optimization. Then we will compare all results with each other, gradually removing methods and procedures that will not work for the given task and leaving the models that will show the best performance. Including competition winning models or models used for classification in the last couple of years.

1.2 Limitations

Even if there is a lot of hype around artificial intelligence, it is not capable of everything and even though it has intelligence in its name, it is not capable of its own thinking. These models can learn patterns from training data. However, they do not act based on ordered rules. This can result, and often does, in a bias occurring in a given model based on the data seen. Later, when the model tries to generalize on unseen data, it might discriminate against specific groups or it might be able to misrecognize when the background changes. For the classification of images on camera traps, the environment around a given static trap may be learned and the animal may be misclassified. This would be problematic when trying to apply this model to other locations, not in the training data.

Trying to avoid this problem of generalization leads to another limitation. A common way to get rid of the bad generalization problem is to get a lot of data. Using large amounts of data generally leads to better results on unseen data. This can lead to problems using a small amount of data on a given project, it is necessary to have a good set containing all the labels that we will try to classify. For this particular competition, a fairly balanced set of images is provided, but as we will see, the problem of poor generalization will still be evident in this task. If the set of images did not contain enough of a specific species, it would be very

[1]`https://github.com/LKolblFAV/Diploma-thesis`

difficult to get correct predictions for that animal. In this competition, we will also be limited by the camera traps used, which often have different resolutions and overall image quality. There is also a problem with the placement of traps and the number of captured animals on the given cameras. The second problem is also providing sequences of images where one trap captures the same animal several times in a row and thus, we get data that is not completely 100% new.

1.3 Goals

This work aims to create a useful model for a selected competition. An examination of the methods and processes used for various classification problems and whether these methods are suitable for use on images obtained from camera traps. We will also be interested in what is the best way to deal with the given data and find out if it is possible to get a classifier that will be less susceptible to changing data from a new location where even a different kind of camera trap will be placed.

This project does not have millions of labeled images as other projects did, so we will be interested in how pre-trained models will behave compared to models that will be trained only on our data.

1.4 Outline

The thesis is divided into five main chapters. In this way, it should be easier for the reader to choose which part to focus on based on their interest or understanding of the problem.

After the introductory chapter, which serves to introduce the basic topic of the work and what the reader should expect from this work. We have a second background chapter, which will mainly deal with the setting of this project in a wider context. How does the given competition work, and what are the main goals and motivations of this competition? Next, we will deal with how our input data is created and where the given project takes place overall. Here we will also learn something about how artificial intelligence works in this particular area and what other projects are trying to connect these areas and use machine learning, especially computer vision.

Then follows the third chapter the theory, where we will deal with the theoretical part hidden behind this project and what we will try to use to get the results. Some parts will only be a quick overview, while in others we will try to go more in-depth and describe how the given topics work. Although this section will cover machine learning concepts, it will by no means be an entire course. If you are familiar with the theory of neural networks and Vision Transformers, you can proceed to the next section.

In the fourth chapter, we will deal with all the practical activities in this thesis. It will be explained and shown how the practical part of the competition works and what specific methods and models were used along with their results or at least their influence on a specific task. We will also take a closer look at the data provided and what problems or benefits that data has for our goal. The implementation process for several methods will be thoroughly described, along with the rationale for using them in this specific fashion. To help you decide whether a method or model is successful, a variety of findings will be presented both graphically and textually. At the end of this section, there will be an evaluation of the practical part at the end of the section.

In the last chapter, we will conclude and summarize the entire thesis. We will describe the results and shortcomings of the entire work. It will also be indicated how the competition could be moved forward and how the use of camera traps could be improved.

Chapter 2

Background

In this chapter, we will deal with an overview of the entire competition. The competition was created based on the need to process a large amount of data from camera traps from various places. Through the competition, the organization cooperates with the national park and the institution of anthropology to give space to people to participate in similar projects and draw attention to the possibility of combining and using computer vision and ecology.

2.1 Competition

This is a competition called Conser-vision Practice Area: Image Classification hosted by DrivenData and is designed to classify species that appear in camera trap images. Wildlife classification is an important step in sorting images, quantifying sightings, and quickly determining whether individual species are present in an image. DrivenData has been collaborating with the Wild Chimpanzee Foundation and the Max Planck Institute for Evolutionary Anthropology on Project Zamba (Lingala for "forest") since 2017. This is a multi-year effort to develop machine learning tools for camera trap videos and dramatically accelerate their progress. The speed at which these videos can be processed and used. This team of conservation researchers has made a set of camera trap data available for the community to study and practice [6].

2.2 Taï National Park

All images that are used as training come from Taï National Park in Côte d'Ivoire see Figure 2.1. It is a national park that contains one of the last primary rainforests in West Africa. This area has been registered as a World Heritage Site, which indicates that the area is under the legal protection of an international convention administered by UNESCO. These sites receive this registration if they have cultural, historical, scientific or other significance. This national park has an area of about 3,300 km^2 and is located 100km from the border of Liberia between two rivers.

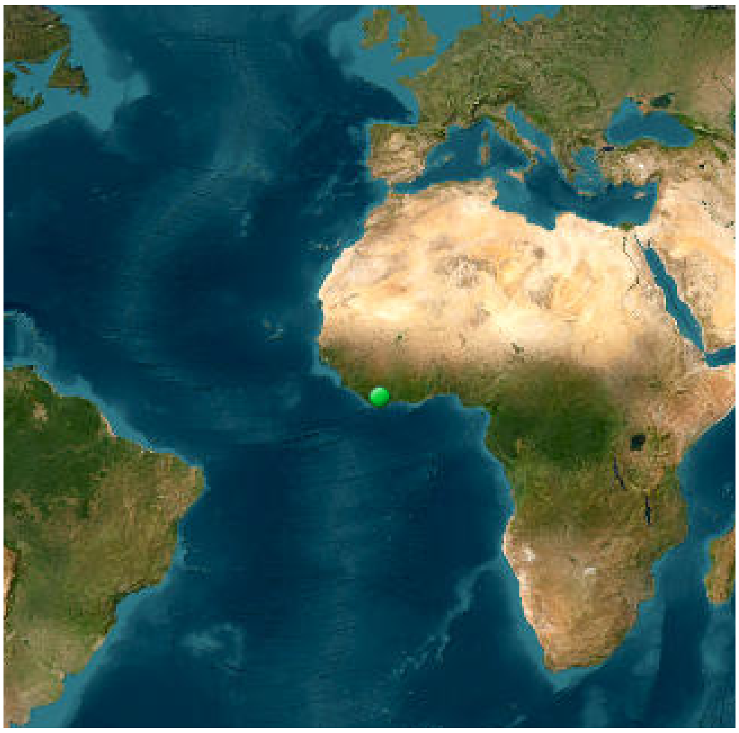


Figure 2.1: Taï National Park location

One of the main goals why the scientific community is interested in this rainforest, in addition to conservation itself, is the occurrence of primates, especially chimpanzees, who are critically endangered. Many other animals that are currently endangered also live in this rainforest. It is also a natural reservoir of the Ebola virus, which the World Health Organization is interested in, for example UNESCO [7].

2.3 Camera traps

To study nature, especially animals in their natural habitat, which is useful for ecologists to assess their population, it is necessary to use reliable devices. So-called camera traps are used for this purpose see Figure 2.2. These cameras are used to capture wildlife on film. These cameras are nothing special. They are mainly everyday cameras, but they are often equipped with infrared sensors that take a picture when movement is detected. These cameras can also detect temperature changes. The advantage of these cameras is to provide data and observe the natural world without a human presence. Although these cameras themselves are not complicated at all, getting these images can be. Often these cameras have to be placed in places that are difficult to visit, these cameras also have to be moved over time because animals can be alerted to the presence of these cameras and then they avoid those places. Due to the adverse effects of the environment, which is not suitable for devices such as cameras, it is not uncommon for the cameras to break. Another problem is that the cameras react to any movement and, therefore, up to one-third of images do not contain any animals at all. For the best results from these devices, it is necessary to carefully select the most efficient and productive place to place the camera [4].

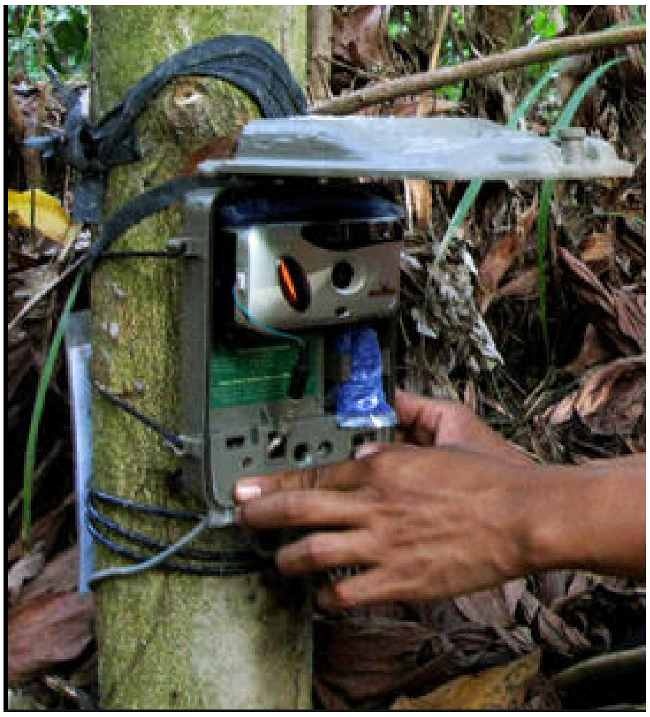


Figure 2.2: Camera trap

Camera traps hold great potential for ecologists, and the rapid introduction of new camera trap projects has exploded the amount of available data. Without a massive increase in the time spent on examining images, researchers must use tools from another field, namely computer vision.

2.4 Computer vision

Since artificial intelligence is a complex field, it is necessary to divide it into individual parts, each dealing with one aspect, which is why computer vision was created. An area of artificial intelligence known as computer vision enables computers and systems to extract useful information from digital photos, videos, and other visual inputs and to understand high level information in digital images and its 3D representation. If AI gives computers the ability to think, computer vision gives them the ability to see, observe, and comprehend. At first, it may seem like a simple problem, because we apply the ability to see in everyday life. Unfortunately, computer vision is a complex field, and a lot of problems arise with capturing visual information. For example, loss of information due to perspective projection or brightness depends on multiple factors such as light source, material reflectivity, camera pose or its orientation. These were examples related to capturing information. Then we have the field of digital information processing, which has its own pitfalls [8].

2.5 Related work

In this section, we review previous research in a similar field. In study [9] they trained machine learning models that used convolutional neural networks with the ResNet-18 architecture and had more than 3 million images available that contained wildlife species captured on images from camera traps. These images were specifically from five states in the United States. They have shown and achieved great accuracy for a large amount of data 98% top-1 accuracy. Finally, trained on smaller datasets, where 82% top-1 accuracy and 94% top-1 accuracy were achieved. Cross-validation by sequence was used.

This [10] paper describes the camera trap data management platform that is being developed for French institutions. Where artificial intelligence tools deal with detection and classification. They use MDv4 for filtering and cropping. And train EfficientNetB3 for species classification. Achieving 95% accuracy on the validation set.

Schneider at al. [11] validated the classification performance on 47 thousand images from Canada. For classification, they use DenseNet201, which achieves an accuracy of 95.6 % on the validation data, which is located at the same location. Ensemble will then be used to obtain better performance. However, noticing a large drop in accuracy when moving to a new location, namely 71 % accuracy from the ensemble. They also point out a change in performance during class imbalance. Finding that a minimum of 1000 images from each class is needed to obtain a mean accuracy of 95 %.

All these articles have a common problem and that is the need for as many images as possible from the given cameras. Furthermore, it can be seen that the models tend to perform poorly when testing on data that is taken from a different location than the original training set.

Whytock at al. [12] trains a species classifier on 347 thousand images from different regions of central Africa. They have a split training and test set based on location and also have a test set containing new locations. Making use of ResNet-50 for training. Achieving results of around 77 % on top-1 accuracy. They also show results when aggregating labels, which were often misclassified. Lastly, showing the calculated species richness and activity patterns for four focal species. Practically pointing out that machine learning-generated results are as good as expert-generated results. This could mean that for many applications it is not necessary to perform labeling of every single image.

All these projects try to achieve the same results that is the best possible classification from camera traps. Let's look at the last project, which is closely related to our competition since our competition is derived from that project.

Zamba Cloud is a project by DrivenData which is also a host of our competition. They made a tool built in Python that uses machine learning and computer vision to automatically detect and classify animals that appear in videos from camera traps. In this project, you can identify unknown species, filter empty images or create custom models that then search for species in specific habitats. Their main focus is on 23 species that are often seen in central Africa. Five models and their ensemble is used for training. The model for African species achieves a top-1 accuracy of 82 % [5].

The first three projects are rather an example of what problems occur in this field and what results could be achieved under the right conditions. While the last two projects are more closely related to our competition and apply methods to similar data.

Chapter 3

Theory

In this chapter, we will focus on the theory that needs to be understood in order to use the mentioned methods correctly. First, we will focus on machine learning in general. Subsequently, we will focus on the topic of classification and how neural networks can be applied for this purpose. Finally, there will be an analysis of the specific methods and models that are used in this thesis and how the data can be handled and edited to our benefit.

3.1 Machine learning

Machine learning is a branch of artificial intelligence, which is further broadly defined as the ability of a machine to imitate intelligence and human behavior. Artificial intelligence are systems used to perform simple to complex tasks that they try to solve in a similar way, as if an expert on the problem would solve the given task. Machine learning usually starts with data. These can be numbers, images, text or any other data that is obtained and modified specifically so that it contains sufficient information to solve the given problem. This data is then used as training data and serves as information for the given machine-learning model. In general, the more data, the better the solution. Part of the data is held out from training and is used as a validation set, which tests how accurate the machine learning model is when it is shown new data. In general, the machine learning function of a system can be descriptive, that is, it tries to describe what happened from the data, or a predictive system tries to predict what will happen from the data, or prescriptive, the system tries to use the data to recommend next steps. Machine learning generally falls into two categories. The first category is supervised machine learning. In this case, the model is trained using a labeled data set. This way the model is allowed to gradually improve and learn with each additional piece of information. The second method is unsupervised machine learning, where the model does not receive information about the data from the so-called teacher, but tries to identify patterns in unlabeled data by itself [13].

3.2 Classification

The world is increasingly data-driven. With the help of data, we try to understand society, predict economic decisions and, based on them, make decisions about our next steps. Classification is one of the most commonly used supervised machine learning processes. Data classification is the process of organizing data into different categories. Thanks to classification, we can easily retrieve, sort, and store data for later use. In this case, classification means a request from the computer program to specify which of the k categories the given input belongs to. This usually means creating a function $f : \mathbb{R}^n \to \{1, ..., k\}$. When $y = f(\mathbf{x})$, then the model assigns the input described by the vector $\mathbf{x}$ to the category identified by the numerical code y [14]. This thesis uses labeled images and the classification is performed exclusively using a supervised algorithm. Example of classified image, see Figure 3.1.

Figure 3.1: Classification - Leopard

3.3 ResNet

ResNet [15] or Deep Residual Learning for Image Recognition is a deep convolutional neural network.

Deep neural networks are more difficult to train. In this paper, a method of training a neural network is shown which serves to facilitate the training of deeper networks than what was previously possible. All models that had good results in competitions such as ImageNet [16] or other visual recognition tasks used the deepest possible networks, so it is not surprising that the depth of the network is an important parameter to obtain the best possible results. In this study, they address the question of whether better network training means increasing network depth. To increase the depth of the network, it is necessary to remove a very well-known problem, namely vanishing/exploding gradients. This problem causes the network to not converge with the addition of more layers.

When training a deep neural network using gradient and back-propagation, we get partial derivatives when passing through the neural network. When using the chain rule, deeper layers go through continuous matrix multiplication to calculate their derivatives. With an n-layer network, there are n derivatives multiplied together. If the derivatives are large, then there is an exponential growth of the gradient by gradual propagation through the model until it explodes. This is called an exploding gradient. If the derivatives are small, then there is an exponential decrease during the passage through the network and the complete disappearance of the gradient occurs. This is called the vanishing gradient.

So far, this problem has been solved by normalizing the initialization and continuous normalization of layers, thus achieving convergence for networks with a depth of tens of layers using SGD and back-propagation.

Deep networks also have the problem of degradation. Gradually, the accuracy of the network becomes saturated and subsequently degrades rapidly. In this study, authors propose a solution using a deep residual learning framework. Instead of hoping that several stacked layers will exactly fit the desired underlying mapping, they leave these layers to fit residual mapping. If the underlying mapping $H(x)$ is the desired, we let the nonlinear layers fit mapping $F(x) := H(x) - x$. The original mapping is recast into $F(x) + x$. This formulation is implemented using so-called skip connections, see Figure 3.2.

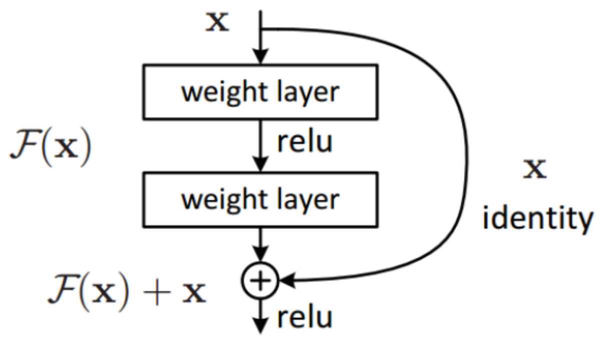

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.2: Skip connection

In this case, the skip connection performs identity mapping, and their outputs are added to the outputs of the stacked layer.

The reformulation using residual learning is inspired by the degradation problem. If the deeper model is constructed such that the added layers are identity mappings, then the deeper model should not have a training error larger than its shallower version. It follows from the degradation problem that solvers have a problem with the approximation of identity mappings by multiple nonlinear layers. From this reformulation, it is possible to bring the weights to zero to approach identity mapping, which can only happen if the identity mappings are optimal. However, they are not optimal in general, but this reformulation still helps to precondition the problem. If the optimal function is closer to the identity mapping than the zero mapping, then it is easier for the solver to find perturbations than to learn a completely new function [15].

By implementing these procedures and experiments on ImageNet and CIFAR-10 [17], the final ResNet neural network is created. They were able to increase the depth of the neural network and show improved results for 18, 36, 50, 101 and 152 layers. By using the ensemble, this network won first place in the ILSVRC 2015 classification competition.

3.4 Vision Transformer

Vision Transformer [18] is a model that tries to apply the Transformers: self-attention-based architecture, which is commonly used for natural language processing, as a model for image recognition. In this area, the dominant way is to pre-train on a large corpus and then fine-tune it on a smaller dataset. Using Transformers, it is possible to train models of huge sizes, which have more than 100 billion parameters. Even with larger models, there are no signs of reaching performance saturation.

There have been several similar attempts to combine self-attention with CNN-like architectures in the past, but they were not as successful as classic ResNet-like architectures on large projects related to image recognition. In this study, authors apply Transformers directly to images with as few modifications as possible. The image is divided into patches and a sequence of linear embeddings from these patches is fed as input to the Transformer. Image patches are subsequently handled in the same way as tokens (words) in NLP. However, the model fitted in this way has a problem with translation equivariance and locality, due to which it has problems generalizing if it does not have a sufficient amount of training data. An improvement occurs if the model is trained on a sufficiently large dataset (14M-300M images). With such a pre-trained model, it shows good results when subsequently applied to tasks with fewer data points.

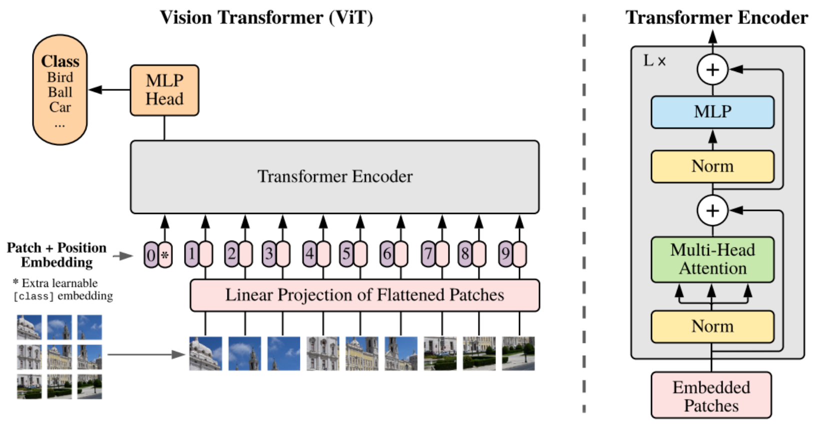


Figure 3.3: Vision Transformer

On Figure 3.3 we can see the model overview. The standard transformer receives as input a one-dimensional sequence of token embeddings. To handle two-dimensional images, you need to reshape image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened two-dimensional patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the original image and C is the number of channels. (P, P)

is the resolution of each image patch, thereby obtaining the final number of patches, $N = HW/P^2$, which are subsequently used as input for Transformer. Transformers use a latent vector through all their patches and so it is necessary to flatten the patches and map to the same dimension as the latent vector using a trainable linear projection.

Vision transformers have a much smaller image-specific inductive bias than CNNs. For convolution neural networks, locality and two-dimensional neighborhood structure and translational equivariance are present throughout the model. In Vision Transformer, only MLP layers are local and translationally equivariant, while self-attention layers are global. Except for some moments during the cutting of images into patches and fine-tuning. Position embeddings do not carry any positional information at the beginning of initialization, and it is therefore necessary for the model to learn all two-dimensional positions in patches and their spatial relation from the very beginning.

Experiments were performed on ResNet and Vision Transformer and their combinations. They focus on datasets of different sizes and the computational cost of pre-training, where the Vision Transformer is less demanding to pre-train. Furthermore, the Vision Transformer is compared with past state-of-the-art results, where it performs the same or even better on most benchmarks [18].

3.5 Augmentation

Augmentation is the process of generating new transformed versions of images from a given image dataset to obtain diversity. The image is represented by a two-dimensional array of numbers. These numbers represent the pixel values in the image. By changing these values we can generate new augmented images. These images are very similar to the original ones from which they were created. However, they contain additional information that can be used to improve generalization in a machine-learning task. In general, augmentation is suitable for improving the performance of detection, classification or segmentation. For the best results of computer vision tasks, it is advisable to have extensive datasets that contain all visual aspects of the targets that we want to classify. However, it is very difficult for a specific dataset to meet these requirements. The data is often manually collected and manually annotated, so it is very time-consuming and often impossible to produce such a dataset. As an example, if we want to capture an image with all possible lighting conditions, it is likely that, despite all efforts, there will always be some information missing that will limit the model in its best possible performance. Using augmentation, we are able to create data that can fill in this missing information. We are therefore able to save time when collecting data, but also improve the overall performance of an existing dataset, thus preventing overfitting.

Overfitting is a common problem in computer vision. Generally, this is a condition where great results are obtained on the training set, but there is drastic deterioration when applied to the test data. In principle, the model learns underlying patterns with high accuracy on seen data but misrecognizes similar patterns on unseen data. This phenomenon generally occurs with a small dataset that is not sufficient for the task at hand. As a result, the model does not have available all possible patterns that it could learn. Augmentation is one of the techniques used to improve the overfitting problem. For computer vision tasks, there is a large selection of augmentations that can be used. The basic ones include position manipulation: such as rotation, scaling and flipping. Color manipulation: such as brightness, contrast, saturation and hue [19].

Chapter 4

Practical tests

In this chapter, attention will be paid mainly to the practical parts of this work. First, the competition will be introduced. Then we look at the baseline and gradually get to the modifications and procedures implemented to improve the neural network and its classification.

4.1 Practical introduction to competition

I participated in a competition as part of this thesis. This is a competition called Conser-vision Practice Area: Image Classification hosted by DrivenData[1]. The aim of this competition is classification. Specifically, the classification of wildlife species that appear in camera traps. Available at our disposal are images with additional attributes of each image. These will be later used for setting up testing and training sets. Training and testing images are in *.jpg* format each in its own file. Attributes are then provided in *.csv* file. These files contain the following fields:

- id (string, unique identifier) - a unique identifier for each image
- filepath (string, feature) - image path including its split directory
- site (string, feature) - the site in which the image was taken

Sites in the training and testing datasets do not overlap. This fact will be later used for reduction in overfitting and splitting the main data set into training and validation datasets.

The last piece of information that is provided are labels for training images. Testing images do not have these labels available. And file containing submission format for the competition.

4.1.1 Dataset

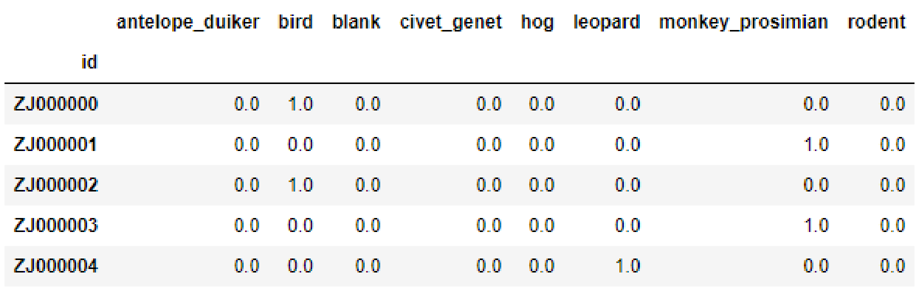
After reviewing the data and attributes in the files mentioned earlier, we will examine the images themselves and their properties. There are eight possible labels for every image, which means our classification is divided into eight classes.

[1] `https://www.drivendata.org/`

If the image does not contain any animals, it is labeled as blank. If not, one of the seven labels designating a species group is present. Every single image contains a maximum of one group. Those are:

1. antelope_duiker
2. bird
3. civet_genet
4. hog
5. leopard
6. monkey_prosimian
7. rodent

First, let's take a look at train_labels.csv file. File has a single row per image, first row contains ids of all the species in alphabetical order. Every other row then contains image name starting *ZJ000000* and ending *ZJ016487* followed by a binary validation. One column for each of the possible predicted classes that indicates whether the image is of that class or not. Each row will add up to 1, since the classes are mutually exclusive.



	antelope_duiker	bird	blank	civet_genet	hog	leopard	monkey_prosimian	rodent
id								
ZJ000000	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
ZJ000001	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
ZJ000002	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
ZJ000003	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
ZJ000004	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

Figure 4.1: Labels

The train_features.csv file again contains one row per image. Each row contains id, filepath, and site. The exactly same format is used in test_features.csv, image names start with *ZJ016488* and ends with *ZJ020951*.

The train features and test features files, which include all of the training and testing images, is the next part we examine. Train feature file contains all 16487 training images in .jpg format. We will review specifics in the next section. The same goes for the test feature file which contains 4463 images.

4.2 Data

In this section we will look in more detail at image properties. Here is an example of a single image in our dataset. It is an image 4.2 named - ZJ016048.

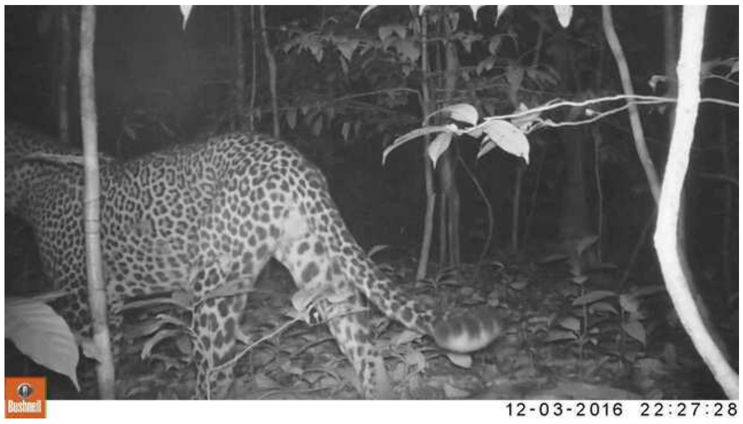


Figure 4.2: Leopard image

Information that we have is the id, site - S0085, filepath - train _features /ZJ000048.jpg and we know it contains leopard - [0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0].

Let's explore how the various species are distributed throughout the training set, first in terms of total counts and then in terms of percentages.

Count and percentage of animals		
	Count	**Percentage**
Monkey_prosimian	2492	0.151%
Antelope_duiker	2474	0.150%
Civet_genet	2423	0.147%
Leopard	2254	0.137%
Blank	2213	0.134%
Rodent	2013	0.122%
Bird	1641	0.010%
Hog	978	0.059%

Table 4.1: Table of animal distribution

As you can see, the distribution 4.1 is not completely even. Although it is better than found in nature, the photos are selected to form a more uniform distribution. The most common species group are monkeys and the least common group are hogs. The number of all training images is 16488 with 148 different used sites.

Now let's look at examples of all eight classes:

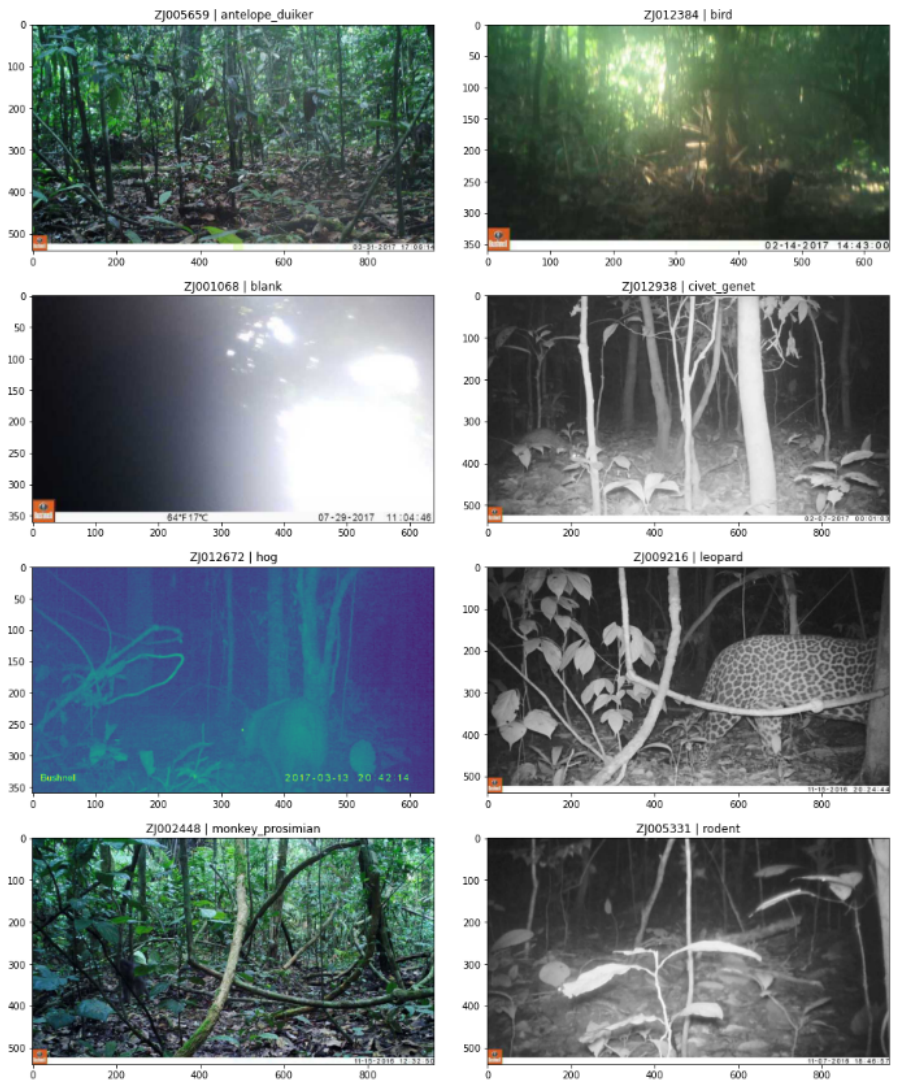


Figure 4.3: All classes example

As you can see, our depictions 4.3 of each species differ significantly from one another. Among other variants, the animals could be further or closer to the camera, in the sun or the shadows, or facing the camera. The hue of the images, the weather at the time they were taken, and the type of camera are also different. One of the main struggles will be the ability to teach the network to generalize across different sites and environments.

In the Fig. 4.4 you can see the number of images corresponding to a specific camera trap.

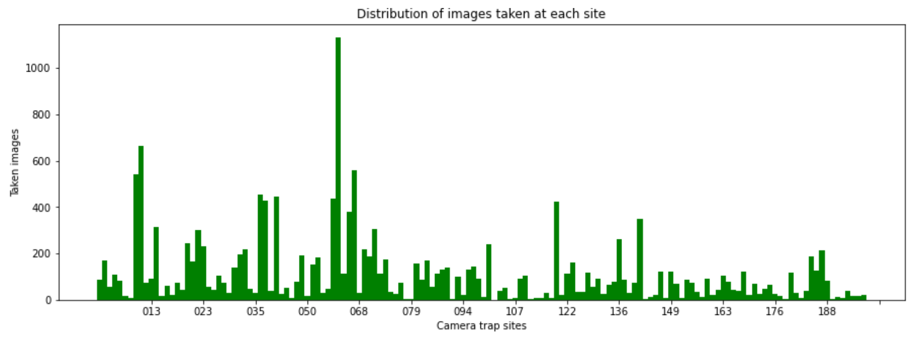


Figure 4.4: Images per site

Again, the distribution of images taken by each camera trap is not even. For example, we have 1132 images from camera trap number 60 and only 2 images from camera trap number 78.

We can also say that various cameras would record various numbers of photos of each species. It is more likely that parts of the rain forest will be inhabited by different species or that some animals wont live next to each other.

As already mentioned, the images are very different from each other. There are 2197 grayscale images; these images don't have 3 channels, and 14291 RGB images. Also, the brightness across the images varies, in the next picture you can see a count of the average brightness in the whole dataset. RGB images are converted to grayscale and then the mean brightness value is calculated.

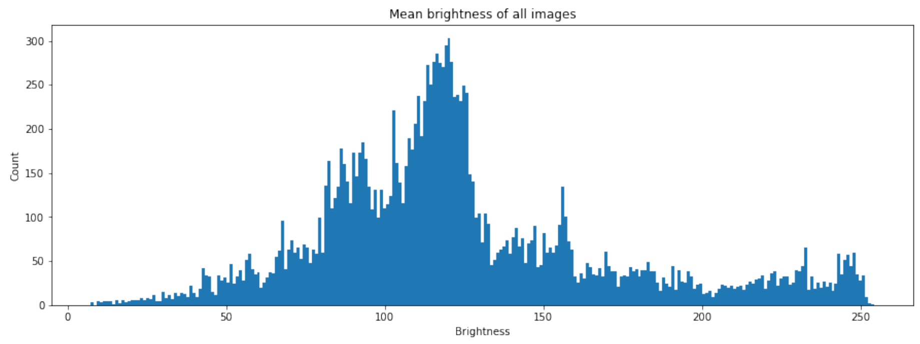


Figure 4.5: Mean brightness across all images

As you can see in the Fig. 4.5 dataset contains a diverse set of pictures from really bright to really dark.

The last two properties that we will focus on are the dimensions of the images and subsequently the covariance matrix of the occurrence of animals. Dimensions of the images are written in the Table 4.2:

Size of images and aspect ratio		
	Count	**Pixels**
Heights	7490	360
	6345	540
	970	335
	864	240
	458	515
	293	120
	67	215
	1	95
Widths	8460	640
	6803	960
	931	360
	294	160
	Count	**Ratio**
Aspect ratio	13835	1.777
	970	1.910
	864	1.500
	458	1.864
	293	1.333
	67	1.674
	1	1.684

Table 4.2: Table of image sizes and ratios

In the Table 4.2 you can see that images differ in their sizes and aspect ratios. We must take this fact into account when handling these images.

Lastly we have the covariance matrix of animal occurrence:

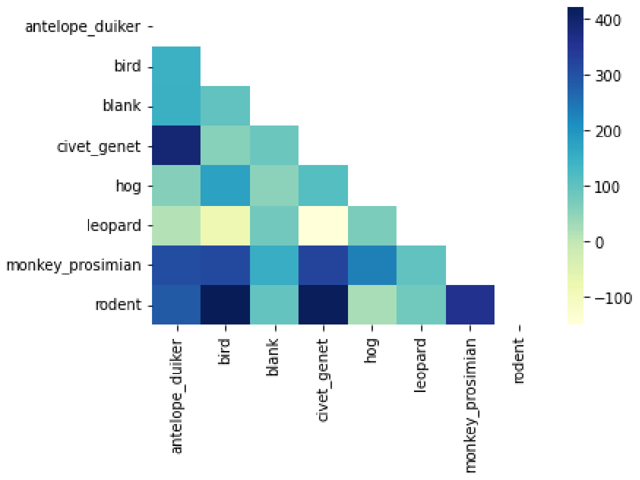


Figure 4.6: Covariance matrix of animal occurence

In the Fig. 4.6 we can see that the coappearance is not equal between species pairs. This corresponds with our earlier estimate. For example, we can see that leopards and civets don't share their space very often. This is expected as leopards are natural predators that prey on civets in the wild.

4.2.1 Data split

We will try to use all the data available to us. The training set should contain a larger part of the data than the validation set. It is possible to use for example 75:25, but this section will describe why we use a different split ratio. A common practice is to use the so-called K-fold [20]. The data will be divided into k-folds in our particular case into five folds. This is a part of cross-validation, where we try to select the best possible data for our training. First, we are going to use stratify k-folds; this will result in five training sets and 5 validation sets with an 80:20 ratio. Now these splits help us a little bit with overfitting; we did not use any additional information except our data distribution. We can further improve our data split by using a stratified group k-fold [20]. This technique will add additional grouping condition to our split. This grouping condition is going to be made according to our additional attributes that our images have and that is the site at which the photograph were taken at. This particular split, which has completely different sites in the training set than in the validation set, will help us with overfitting, now our network must learn to generalize in previously unknown locations. This method will also help our model to make better predictions for the test set, which also contains different sites compared to the training and validation sets.

4.2.2 Data augmentation

This is a technique used for improving the performance of deep neural networks. To achieve good results and avoid overfitting, deep neural networks need a lot of training data. But getting enough training samples is frequently very challenging. This is where augmentation comes into play. By augmenting already labeled images with non-label changing transformations, we can achieve a much larger dataset. We can also get more samples with different distortions present in a few images [21].

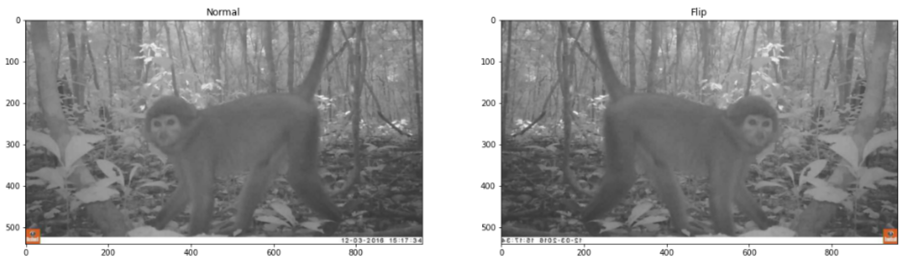


Figure 4.7: Augmentation example horizontal flip

These augmentated images 4.7 are not fully-fledged images as they do not contain new data. Now let's look at the specific augmentation used. First of all, resize to 224x224 is used to provide uniform input; this also corresponds with input used in original ResNet [15] paper; then we use normalize, that is changing mean and std with parameters also corresponding with original paper.

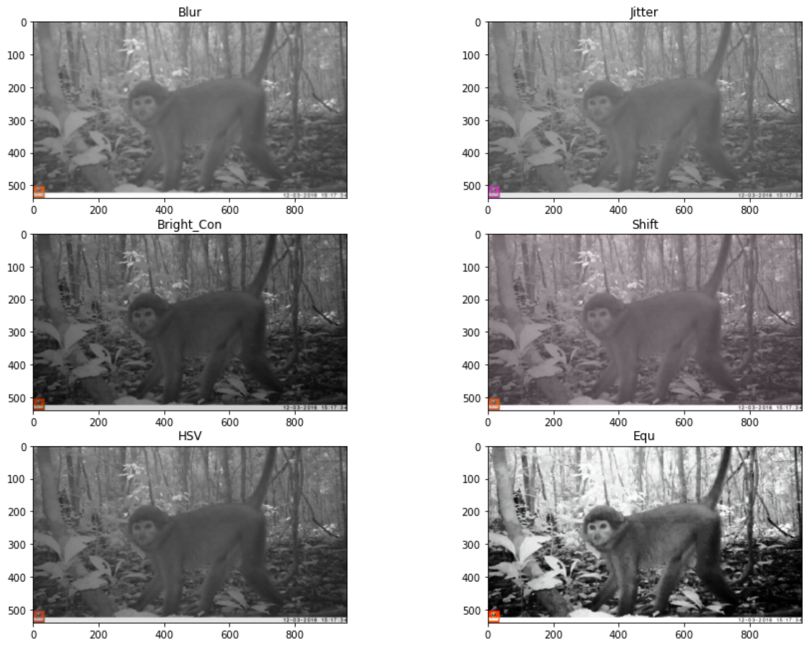


Figure 4.8: Used augmentations

We use a combination of:

Horizontal flip 4.7 - flips the image around the y-axis with 0.5 probability. Gaussian blur 4.8 - blurs the image using a Gaussian filter with random kernel size (3 to 7) with 0.2 probability. Although it may seem that the image is not too blurry for the human eye, for the neural network, all the values in the image change. Next is color jitter and random brightness-contrast 4.8 together with 0.5 probability. Both have an equal chance of being used and they both serve the same purpose, changing brightness and contrast values, color jitter having more options. We use either hue-saturation or RGB shift 4.8. These two have a 0.2 probability of being applied. The first one is changing values of hue and saturation in the image, and the RGB shift is randomly shifting values in each channel of the image. The last one is equalize, which equalizes the image histogram; this is a pretty big change; the probability of it being applied is only 0.05.

All of these augmentations are made "on the fly", which means that we are changing them while training and we are not storing these changed images. All of these augmentations are probability-based, so only part of them is used on each image; these probabilities never reach one to ensure that the original image is being seen by a neural network. Validation images are only augmentated by resize [21].

There is a penalty if the network declares the ground truth class to be incorrect, and competition severely handicaps the network that confidently classifies in the incorrect class. It is frequently possible to significantly increase the generalization and learning rate of a multi-class neural network by employing soft targets, which are a weighted average of the hard targets and the uniform distribution over labels. Label smoothing has been used in many state-of-the-art models preventing the network from becoming over-confident [22]. Another change that belongs to the augmentation category is cropping the image. We want to keep the 224x224 resolution as the pre-trained model is trained on these inputs. Often the edges of images are not containing any significant information, so we will first resize to 256x256 pixels and then crop to 224x224 pixels resulting in a decrease in loss on testing dataset.

Models		
Architecture	**Applied methods**	**Competition loss**
ResNet50	No augmentation	1.9200
ResNet50	Augmentation	1.9261

Table 4.3: Table of basic augmentation results

In 4.3 the loss function for basic augmentations is the best result from the combination of augmentations mentioned above. We can see that there is no improvement for the best result, but there is a decrease in overfitting when training more epochs.

Cutout

Because the implemented weaker augmentations did not have such a fundamental effect on improving overfitting, stronger augmentations will be implemented, which will also result in label changes. The first augmentation added is a Cutout [23]. This is a regularization technique used mainly for convolutional neural networks. This process removes contiguous regions of the selected input image. This way, we augment the input dataset with versions of the image that have occluded parts. This procedure can be seen as an extension of dropout without changing the feature maps. In the article, they pointed out that the shape hyperparameter does not affect performance as much as the size of the removed regions. Cutout forces model to take full image context into consideration, instead of focusing solely on a few key features. In this way, we prepare the model so that it is ready to recognize images that have obstructed parts. Cutout is used as the completely last augmentation on input images. In this implementation, the occluded region is made by randomly choosing a pixel, and according to the required size, the corresponding rectangle is masked out. There is also a possibility to select multiple regions in a single image.

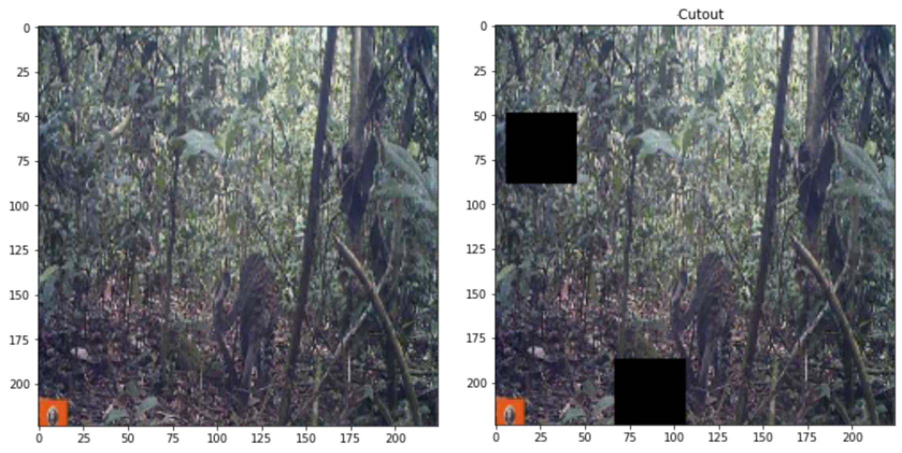


Figure 4.9: Cutout example

In the 4.9 we can see a specific random execution of the Cutout method for one or more removed regions. This augmentation does not affect the resulting label of the given image.

Mixup

The second augmentation method is called Mixup [24]. Mixup uses convex combinations of samples and labels to train neural networks. Mixup optimizes the neural network to favor straightforward linear behavior between training samples. Additionally, it decreases the memorization of corrupted labels and improves robustness to adversarial examples. By using Mixup we construct virtual training examples where parts of images are swapped with each other. Given two images and their ground truth labels: $(x_i, y_i), (x_j, y_j)$ a virtual training sample $(\hat{x}, \hat{y})$ is generated:

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j \tag{4.1}$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y_j \tag{4.2}$$

where $\lambda \approx Beta(\alpha = 0.2)$.

The hyperparameter α controls the extent of interpolation between feature-target pairs. In our specific implementation, Mixup is applied in pairs, where two λ values are randomly chosen for each pair, and a pair of images is augmented twice. The labels of these pairs are also mixed in the same way.

These 4.10 are the original images before the Mixup is applied. In the 4.11, we can see a specific random example of images after the mixup has been applied. The first pair on the left has λ_1 equal to 0.2145, and the second is equal to 0.3424. The label of each image is also changed according to λ values.

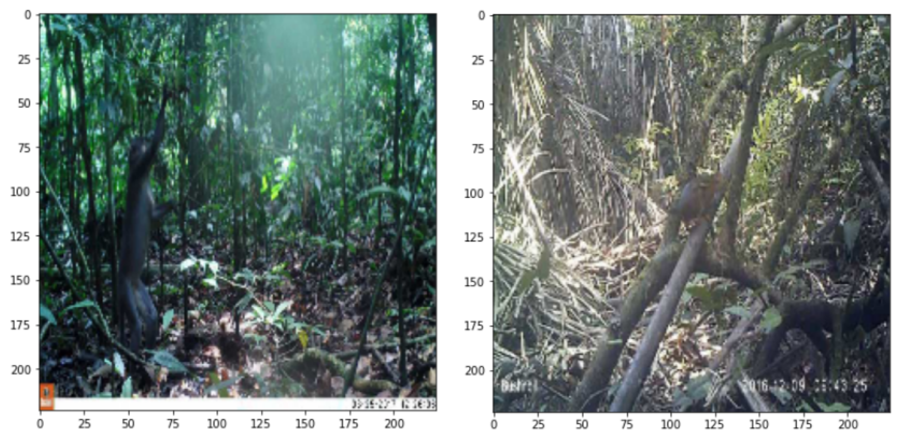


Figure 4.10: Original image before Mixup

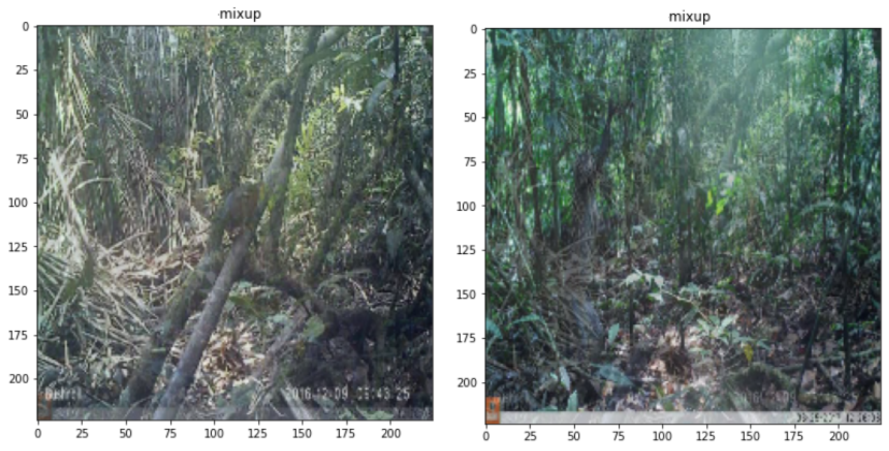
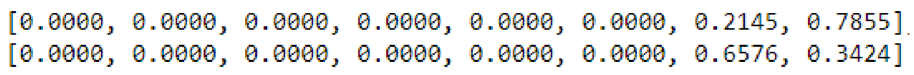


Figure 4.11: Mixup example



```
[0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.2145, 0.7855]
[0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6576, 0.3424]
```

Figure 4.12: Example of how labels are changed

As we can see in the 4.12, the first image is a monkey, and the second image is a rodent. Their labels are mixed according to λ values where first λ_1 is the ratio for the first image and second λ_2 is the ratio of the second image.

CutMix

The third augmentation method is called CutMix [25]. This method addresses an issue of having uninformative pixels in the training images during training while retaining the advantages of regional dropout. The additional patches make the model recognize the object from a partial view, which improves localization capability even more. This method shares similarities with [24] and [23]. The goal of this method is to generate a new training sample $(\hat{x}, \hat{y})$ by combining two training samples $(x_i, y_i), (x_j, y_j)$, where the combination is defined:

$$\hat{x} = \mathbf{M} \odot x_i + (1 - \mathbf{M}) \odot x_j \tag{4.3}$$
$$\hat{y} = \lambda y_i + (1 - \lambda) y_j \tag{4.4}$$

Where $\mathbf{M}$ denotes a binary mask indicating where to drop out and fill in. In this particular implementation, the dropout region is chosen in a similar way as in Cutout. This region is then filled with information from another image. Cutmix is an augmentation that changes the labels of training images. That is done by taking the ratio between the original image area and the ratio of the filled-in image's label.

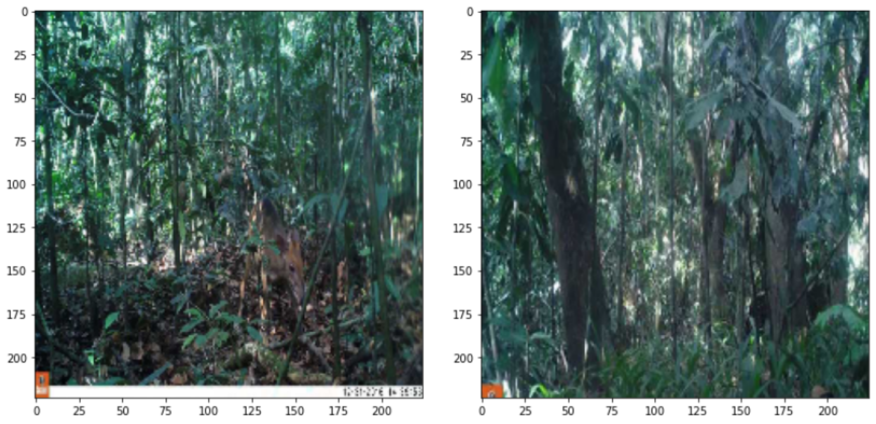


Figure 4.13: Original image before CutMix

These 4.13 are the reference images before CutMix is applied. In the left image, we can clearly see an antelope; in the right image, we can see a hidden leopard behind a tree.

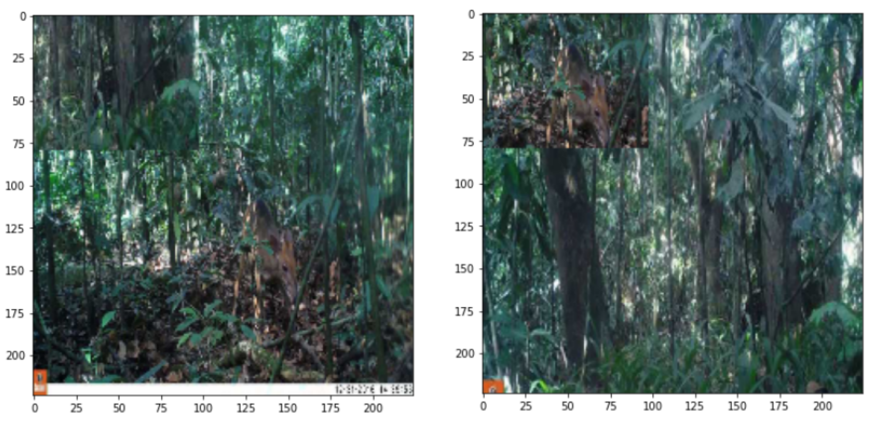


Figure 4.14: CutMix example

In the 4.14, we can see two images with Cutmix being applied. The labels are also changed according to the area changed. This 4.15 is an example of how labels are changed with these 4.14 images. The labels are changed for both images, with the changed area being about 15 %.

```
[0.8438, 0.0000, 0.0000, 0.0000, 0.0000, 0.1562, 0.0000, 0.0000]
[0.1562, 0.0000, 0.0000, 0.0000, 0.0000, 0.8438, 0.0000, 0.0000]
```

Figure 4.15: Example of how labels are changed

Cutmix performs better than the other two methods, as demonstrated in the original paper. However, we'll combine all three approaches to achieve our goals. There is always room for improvement regarding the choice of the hyperparameters of used augmentations.

4.3 Neural network architecture

In this section, we will focus on exploring and selecting architectures that are suitable for this competition. First up we have ResNet50 [15]. This network was also used to obtain 1st place on the ILSVRC 2015 classification task. The second network is ResNeXt [26], as the name suggests, it is a follow-up to the work of ResNet. ResNeXt iterates a building block that aggregates a set of transformations with the same topology. Compared to ResNet, a new dimension called cardinality arises here, which is the size of the set of transformations, which is a large factor of the neural network, along with the depth and width dimensions. SE-ResNeXt50 [27] is the third selected architecture. Again, this is a modification of the previous architecture, in this case it is the addition of a Squeeze-and-Excitation block, that adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels . EfficientNet [28] is another candidate for this task. This architecture adds a new way in which the model is scaled, resulting in better accuracy and effectiveness. This is achieved by balancing depth, width, and resolution using a compound coefficient . The last architecture we will focus on will be the Vision Transformer

[18]. Transformer architecture, which is mainly used in the field of natural language, is applied to sequences of image patches and shows very good results on image classification tasks.

4.4 Baseline

This section will contain the implementation procedure and a description of the baseline that will be used for classification. It will also be mentioned how the data is used and first results from experiments [29].

We must first split the image set into training and validation sets. To guarantee that the relative frequencies of each class in the training and validation sets are comparable, we will stratify by the target labels [20]. Validation set will contain 25% of the data and training set will contain the rest. As the baseline pre-trained ResNet50 is chosen, this is a pretty common convolution neural network used as a starting point for different types of tasks such as classification or detection [15]. This architecture will create our backbone. The 2048-dimension embedding produced by the pre-trained model is connected to one additional dense layer. In ResNet this last dense layer would be connecting the 2048-dimension embedding to a 1000-dimension output used in ImageNet. In our case, we will connect 2048-dimension embedding to a 100-dimension linear layer, which is connected via ReLU and Dropout to an 8-dimension output. These last layers are the new ”head” of our model this let us convert the image embeddings generated by the pre-trained ”backbone” into the 8-dimensional output. The ReLU introduces a non-linearity into the head and the dropout is a regularization component helping with overfitting. Cross-entropy will be used as a loss function which is commonly used in multi-class classification. Stochastic gradient descent will be used as our optimizer. This optimizer works with learning rate and momentum parameters. For now, the learning rate will be set to 0.001 and the momentum to 0.9. The number of epochs is initially set to only one, but in later stages, we will mainly work with fifty epochs.

Even before we start making big changes, either to the data or to the way of training or changing parameters, it is advisable to first look at one of our goals, and whether it is appropriate to use a pre-trained model right from the start, as it is written in the previous paragraph, or whether it is better to try to train the model from scratch only on our data.

The advantage of using the pre-trained model is its speed of training and getting the first results. Training on our data takes much more time and we risk not having enough data for the network to be able to learn all the necessary patterns that can occur in unseen images. After the first few trained models see Table 4.4, it is clear that to obtain better results, but also to save time, we will make use of the pre-trained models. To save time and get the opportunity to train more models, we will use external servers. It would be demanding to obtain parameters again and again for each new architecture.

Now that we are able to train the model, it is important to track parameters that will help us determine, how well a given neural network setup performs. Accuracy and loss are introduced both for training data and validation data. These values along with the number of epochs will be stored. For storing purposes Wandb [30] is going to be used which is a central dashboard to keep track of your hyperparameters, system metrics, and predictions so you can compare

Models		
Architecture	**Applied methods**	**Accuracy on validation set**
ResNet50	Pre-trained	46%
ResNet50	Not pre-trained	14%

Table 4.4: Pre-trained vs trained just on seen data

models live, and share your findings.

According to these values, models with the lowest loss and the highest accuracy will be stored and used to create submissions on testing data. The accuracy and loss used for competiton is calculated on the validation set from predictions made by the neural network.

One parameter for assessing classification models is accuracy. Accuracy is the fraction of predictions our model got right. We can define it:

$$Accuracy = \frac{Number\_of\_correct\_predictions}{Number\_of\_all\_predictions} \tag{4.5}$$

The accuracy 4.5 is calculated over the entire batch and then it is added to the following batches after a constant number of batches, the average accuracy is validated. In this way, we are able to track the progression of accuracy and learning on the training set. Accuracy on validation set is validated in the same manner.

Log loss [20] , 4.6 was chosen as the performance metric to judge the progress of neural network training. Primarily because it is similar to the performance metric employed in the competition in which we are taking part. Let the true labels for a set of samples be encoded as a 1-of-K binary indicator matrix **Y**, $y_{i,k} = 1$ if sample i has label k taken from a set of **K** labels. Let **P** be a matrix of probability estimates, with $p_{i,k} = Pr(y_{i,k} = 1)$ then log loss is :

$$L_{log}(Y, P) = -logPr(Y|P) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k} log p_{i,k} \tag{4.6}$$

Now we want to look at the predictions made by the neural network and also participate in the competition. Predictions will be made in the same way as for the validation set, but this time the test set will be used. We will provide the likelihood for each of the eight potential classes for each image in test_features in order to compete in this competition [6].

Submission format contains nine columns, with id and with a column for each of the eight possible classes containing the probability that the image is of that class. These values range between 0.0 and 1.0, with each row adding up to 1. We will now take a closer look at the performance metric used by the competition. They decided to use a loss metric instead of an accuracy one and use a similar log loss function to ours. We will want to get as low values as possible. Log loss defined for the competition:

$$loss = -\frac{1}{N} \cdot \sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log p_{ij} \tag{4.7}$$

Where **N** is the number of observations, **M** is the number of classes, y_{ij} is a binary variable indicating if classification for observation i was correct, and p_{ij} was the user-predicted probability that label j applies to observation i.

The results are sent in .csv format and the corresponding loss value is given to us.

4.5 Improving baseline

In this section, numerous adjustments, experiments, and other methodologies will be mentioned. Our goal will be to obtain the smallest possible competition loss and generally avoid overfitting. The majority of these procedures will be written in the order in which they were tested or used.

Transferring training from the CPU to the GPU is among the first adjustments to be made when training the network. Despite the fact that training on the GPU significantly accelerates performance, using external servers with more powerful computers becomes necessary as training times and epochs increase. For these purposes MetaCentrum [31] will be used. In the beginning phases only part of the data and a low number of epochs were used. Now the logical step is to increase the number of epochs and use the whole training set. With these changes, the loss worsened dramatically, overfitting showed itself to be the issue, which we will attempt to minimize.

We will focus on experiments that could help in the fight against overfitting. The first is a change in the used loss function for training and validation. Focal loss [32] is selected for this purpose. This loss function is a direct modification of the Cross-entropy loss function. Focal loss is mainly intended for problems with unbalanced classes during training. This problem is frequently solved by adding a weighting factor α, which is commonly used as the inverse class frequency.

$$CE(p_t) = -\alpha_t log(p_t) \tag{4.8}$$

We also use α as a hyperparameter, even though we don't suffer that much from class imbalance 4.1. This may slightly improve our neural network performance. The second part, which is more important to us, is reshaping the loss function to down-weight the simple cases and adequately focus on the hard negatives. This is done by introducing a modulating factor $(1-p_t)^\gamma$ with a tunable γ focusing parameter. When $\gamma = 0$, Focal loss is equal to Cross-entropy loss, with an increasing value of γ, we can adjust the rate at which easy examples are down-weighted. Focal loss is defined as:

$$CL(p_t) = -(1-p_t)^\gamma log(p_t) \tag{4.9}$$

We will use the α version:

$$CL(p_t) = -\alpha_t(1-p_t)^\gamma log(p_t) \tag{4.10}$$

With this new loss function, we get two new hyperparameters, which we need to determine. For now, we'll set them to the values that achieved the best results in the paper [32].

Over time, it became clear that it is possible to reduce the size of the "head" of the neural network. This means a transition to 2048-dimension embedding connected directly to 8-dimensional output with the removal of Dropout and ReLU.

4.6 Sweep

As mentioned in the previous sections, we have a large number of hyper parameters that need to be determined and optimized as much as possible. For this purpose, a sweep [30] will be used, which automatically searches for hyperparameters and explores the space of possible models. Sweep works on the principle of selecting various parameters and then testing them during training. We have a variety of choices for searching, but it is recommended to start by running several searches with parameters that were chosen at random. From the best results, you can then choose the hyperparameters that are the most appropriate. Due to their quantity and duration, it is not advisable to start the search by searching all possible parameter combinations. It is essential to choose a parameter that we will aim to enhance during the search in order for it to operate properly. Either accuracy or loss function is suitable for this, we will try to maximize accuracy on unseen data.

4.6.1 Sweeping main parameters

We will try to concentrate first on the most important variables. This means the parameters that have the greatest influence on training and thus are able to influence our result the most. The first parameter to be changed is the batch size which is not needed to be fine-tuned as we will try to mimic the size used for training our particular baseline. Now to the parameters, we will change and fine-tune. Learning rate is one of the most important hyperparameters for training. With changing learning rate, we have an opportunity to try different optimizers. We will try SGD, Adam [33] and AdamW. The values from the previously mentioned Focal Loss function are among the last two parameters that we will attempt to optimize, those are focusing and weighting parameters.

Using a random sweep, over three hundred models with different parameters were trained. Now it is necessary to go through all the models and find the ones that met our condition of maximizing accuracy. During the search, we will also possibly pay attention to the loss function; if it were lower than usual for a specific model, it is undoubtedly advisable to examine this model more closely. When using a sweep, we have the advantage that it also provides us with the importance and correlation of the parameters we are looking for according to the selected condition. As expected, the learning rate has the largest contribution to the accuracy improvement followed by alpha and gamma. The three different optimizers have a smaller share in improving the accuracy. Adam and AdamW have a negative correlation, which means that when they are used, the results deteriorate compared to SGD 4.5.

This 4.16 is what a graphical representation of the sweep might look like:

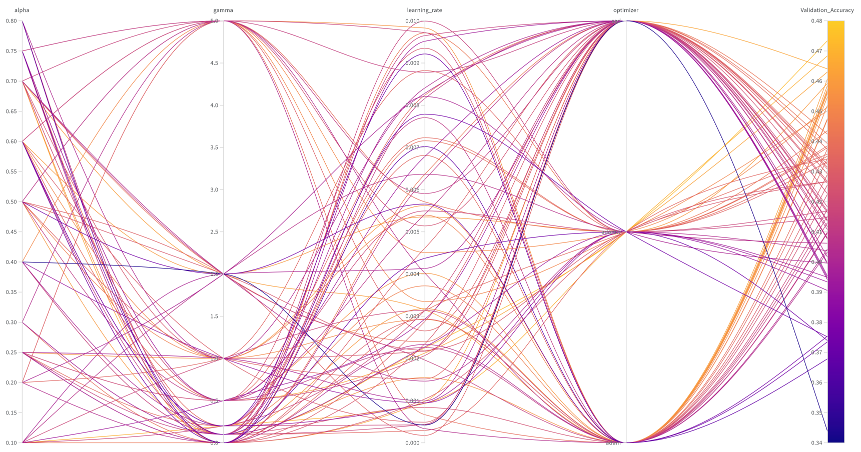


Figure 4.16: Sweep example

The best models are selected according to the facts mentioned; not a single one of them uses an Adam or AdamW optimizer. At the end, we selected six models 4.5 that we will try to submit and evaluate on the test set. These are:

Sweep models				
	Alpha	**Gamma**	**Learning rate**	**Optimizer**
1.	0.75	0.1	0.005006900569993443	SGD
2.	0.4	0.2	0.005357456544183196	SGD
3.	0.6	0.2	0.0034535583484160883	SGD
4.	0.6	5.0	0.007732227549238594	SGD
5.	0.1	0.2	0.0009314258069920744	SGD
6.	0.8	0.1	0.0022513333338251484	SGD

Table 4.5: Table of sweep models

Ultimately, model number four was the best-performing model on the test set. Now we have the model with the best performance and semi-optimized parameters. We will try to look at the variants with K-fold. We do not expect improvement in individual folds, but we will need them in the next part to see if any fold will perform better on the test set. Each model learns from a partially different data set; thus, individual models can better recognize specific characteristics and overall searched for animals. As expected, individual folds do not have a better result. Now we will try to use a combination of these models.

4.7 Ensemble

Another method to improve performance is the ensemble [34]. The ensemble is a way to achieve a better whole from individual parts. In this case, a combination of predictions from individual models. Neural networks generally have a high variance from the principle of stochastic learning. One of the ways to reduce this variance is to train several neural network models and combine their predictions. An ensemble not only reduces the variance of predictions, but can also lead to overall improved predictions that are better than any single model.

There are several ways to combine models. If we know how the given model predicts specific labels, we can adjust the weight of these predictions. In this way, we would obtain a weighted combination of models. For now, however, we will settle for the variant where we consider the models equal and weight the predictions by the mean value of all models.

The number of models used for the ensemble is often small, due to computational duration and also diminishing returns in performance by adding more models. The most commonly used quantities are three, five and ten.

When using the ensemble, it is possible to combine three main elements that can be improved in this way. First is the variation of the training data. In this case, each model is trained on different data. As already mentioned in the previous sections, in our case the ensemble will be composed of five models using k-fold cross-validation. Another element is the use of various models, the optimization problem that the network is trying to solve has multiple possibilities for the successful mapping of inputs to outputs. We will use this combination in the next chapters. How to combine the predictions of these models is the last element.

Although our combination now uses all the training data, unfortunately in our case there will be no improvement over the currently best performing model. Ensemble did not contribute to better results with the ResNet50 architecture used, but later it will be used again when the architecture is changed, in which case the performance will improve.

4.8 Post-processing

In this section, we will look at the predictions received by our model on the validation set and try to improve the results on the test set. In general, it is a procedure to adjust the final data obtained based on the knowledge of the given problem. For this purpose, we will use the confusion matrix and the distribution of which labels have the second and third-largest predicted values. From the submitted results, we know that the first split has a fairly good correlation between the loss function on the validation set and the loss function on the test set at this stage.

As we can see on 4.17, the neural network confuses animals of similar sizes, such as civets and rodents in general. The same can be said about hogs and leopards. And lastly, we can see an overabundance of empty image predictions. It is important to mention that only the information contained in the columns is relevant for us, since the row information about the correct predictions will not be available for the predictions on the test set. However, this information is only partial. First of all, it is an estimate of the top-1 classification, but we are evaluated with an overall loss function that penalizes us for all other predictions as well. Second, we assume that the results on the validation set will be of the same quality as on the test set. Therefore, we will also look at the top-2 distributions for predictions, where we will try to improve, for example, blank images.

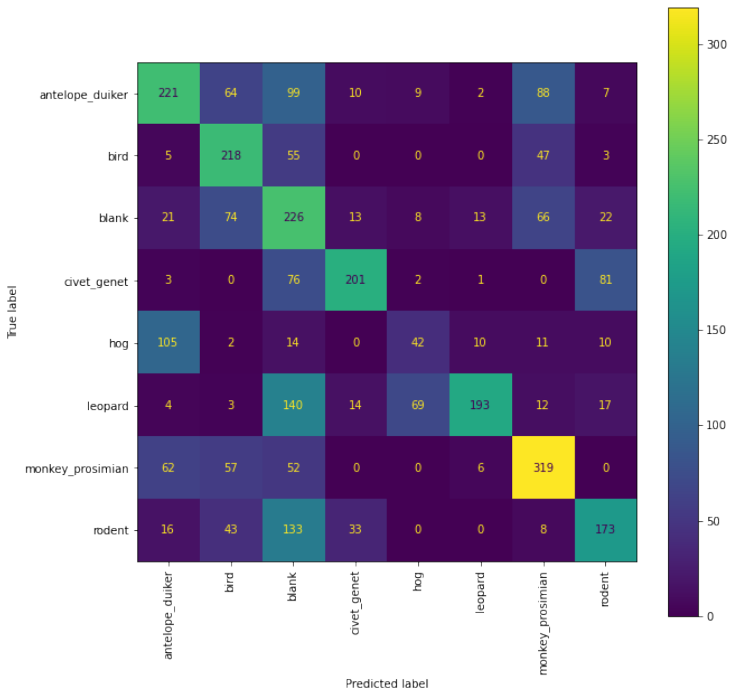


Figure 4.17: Confusion matrix of true and predicted labels

As you can see 4.18, the top-2 predictions are similarly divided as they were in the confusion matrix column. Nevertheless, we will try to get the best possible overview of the network's capabilities from all this information and adjust the data based on this fact for a better result. These predictions have a mean value of 0.21 and a mean deviation of 0.11. This means that if the network is very confident, it does not make sense to consider using these specific values, but if the network is not confident about the blank image, it gives us the possibility to further modify this data.

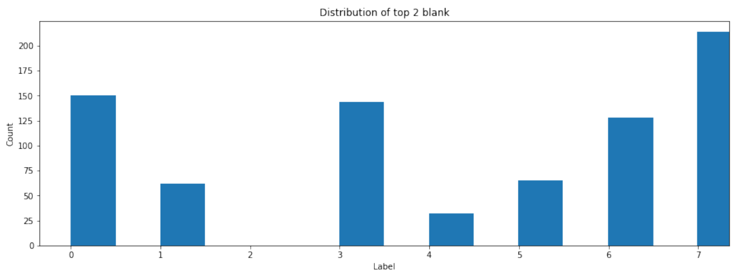


Figure 4.18: Distribution of top-2 blank images

As an example of how to use this data, if we look at 4.18 we can see that if the result for leopards and hogs is swapped when predicting a hog, the loss function must improve.

4.9 Vision Transformer

After inspecting the architectures in earlier section, now is the time to revisit these architectures. After trying different techniques and continuously improving results, we are going to see if there are ways to improve the baselines used. Let's take a look at the frequently used architectures: ResNeXt50, SE-ResNeXt50, EfficientNet and Vision Transformer. On all of these models, we'll start training with the best settings we've found so far on the current baseline. However, it is necessary to adjust some parts of the training process again, since each model is pre-trained on different data and on different augmentations.

Models	
Architecture	**Cross-entropy Loss**
ResNeXt50	1.693
SE-ResNeXt50	1.557
EfficientNet	1.688
ViT	1.298

Table 4.6: Results on different models

These 4.6 are the results of selected models. The values of the loss function are obtained from the predictions on the validation set. As we can see the first three models have similar results, with the SE-ResNeXt50 performing a little better. The best result was achieved by Vision Transformer.

After examining the results, the best improvement was found when using the Vision Transformer. Nevertheless, we return to element two of the ensemble method and try to combine predictions from different architecture types. Their combination has relatively good results, but the Vision Transformer itself remains the best performing model. Now we know that the Vision Transformer responds well to our task, so it is appropriate to explore different variants of this architecture, whether it is depth, the number of heads, patch size or embedding dimension. The selected Vision Transformer is of medium size and has a patch size of 16, being pre-trained on images of 224 pixels in size, as was the case with ResNet50.

So we settled on the Vision Transformer variant. Now it is necessary to iteratively go through some of the steps that we performed with the previous architecture. First, we'll look at the sweep again. We cannot expect the semi-optimal hyperparameters to be the same for our new architecture. So we will again search for learning rate, alpha, gamma, and optimizers. This time, however, we will choose a straight loss function as the criterion condition, since we found out in the previous experiment that it has a much greater influence on the overall success of the model.

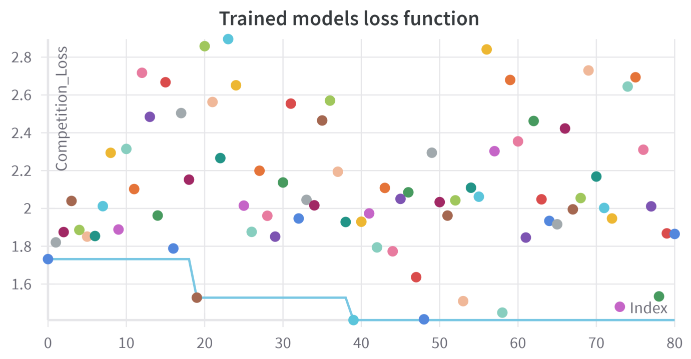


Figure 4.19: Sweep training results example

Again, it is fitting that learning rate and gamma have the largest share in reducing the loss function in this case. Compared to ResNet50, however, stochastic gradient descent shows the worst performance and thus a negative correlation with better results. As in the previous version, the six best models were selected, which must then be trained and compared with each other.

Sweep models				
	Alpha	**Gamma**	**Learning rate**	**Optimizer**
1.	0.5	2	0.003661243065019472	SGD
2.	0.2	1	0.005574236194484602	SGD
3.	0.7	2	0.0004827695883935068	SGD
4.	0.7	2	0.00003345107853573892	AdamW
5.	0.6	2	0.00007013683802495341	AdamW
6.	0.3	2	0.0002980929812770572	SGD

Table 4.7: Table of sweep models 2

From 4.7, the fourth model had the best performance. So we will continue with this model. Since the model has quite a large variance of results, this model will be iteratively trained several times to achieve the best results. As we can see, the winning optimizer in this case was AdamW.

So the next step is the ensemble. With the newly trained models, it is possible to try the ensemble again. We will look again at both variants, where we will first try the ensemble using k-fold cross-validation. And as a second option, we will try an ensemble of different architectures, which means a combination of k-fold ResNet50 and Vision Transformer, as well as a combination of the two best-performing models from both architectures.

The first variant, the Vision Transformer 5-fold ensemble, is created using the mean value of the results of all five models. Ensemble of ResNet50 and Vision Transformer is performed again using the average, but now of ten models, where five models are trained on each of the architectures. Another way is to combine the mean prediction values of the two best models from each of the five. All these ensembles have a significant effect on improving the results. However, the best scoring among these models was the ensemble of five models, all using the

Vision Transformer. An ensemble of these five models is tested last, with the best ResNet50 added as the last of the six or as a pair of 5-fold plus ResNet50. Unfortunately, neither of the last two experiments yielded better results on the test set.

4.9.1 Vision Transformer - post-processing

From the previous section we got a new model which is a combination of Vision Transformer models. Post-processing at this point is not as straightforward as it was in the last part when we focused on this task. There are several problems here. The first problem is that since it is a combination of models, we do not have results on the validation set, and thus it is not clear how to construct the confusion matrix. It is also unclear whether the confusion matrix would correlate between the test and validation sets due to the improvement of results compared to individual models on the test set. Nevertheless, we will try to bypass these issues and find some patterns that would help us improve predictions. We will first try to construct a confusion matrix not from the whole but from the concatenation of predictions from individual models and their respective true labels.

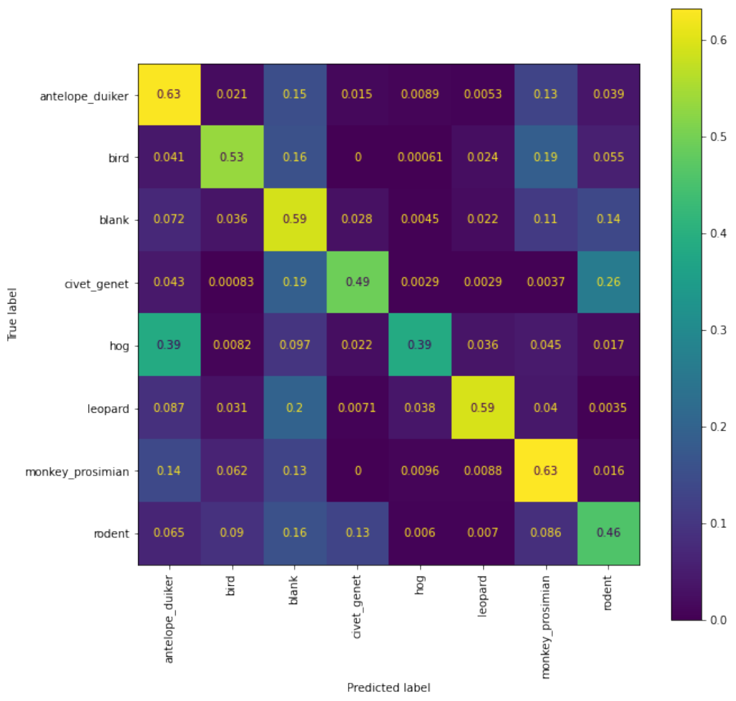


Figure 4.20: Confusion matrix for model ensemble

This 4.20 confusion matrix, therefore, consists of all the training data, which means five times more values than when we dealt with post-processing last time. This matrix is normalized in the column direction since it is the only useful information for us on the test set. We can see that, in this way, the models individually have a problem classifying part of the blank images. Also, civets and rodents are being mixed up. Lastly, when predicting antelopes, part of those predictions are incorrect and they should be rather hogs.

When dividing the matrix into two parts for very confident predictions and those where the models were not very sure, the results for confident predictions are much better than what can be seen in the confusion matrix. Therefore, we will mainly deal with the post-processing of data for uncertain predictions.

Here 4.21 we can see that if the model is not very confident, that means a prediction smaller than 0.6, the model makes mistakes much more often. It can be assumed that the error rate also increases with uncertainty. This, however, gives us room to improve, especially on these predictions. We see similar problems as with the previous confusion matrix. There is also increasing uncertainty in the predictions of antelopes and also in the predictions of monkeys.

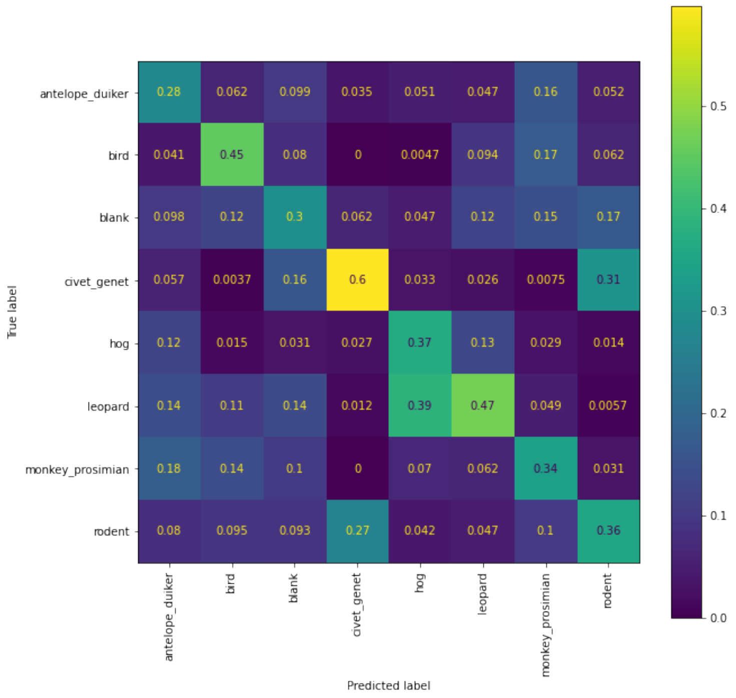


Figure 4.21: Confusion matrix for unsure predictions

4.10 Results

In this section, we will look at a summary of the progress of the entire competition and take a closer look at the specific results of individual experiments.

Models		
Architecture	**Applied methods**	**Competition loss**
ResNet50	Stratified K-fold , split - 0.75/0.25	3.5242
ResNet50	Stratified group K-fold, split - 0.8/0.2	1.9200
ResNet50	Label smoothing, augmentations	1.5826
ResNet50	Improving augmentations, Focal loss	1.5246
ResNet50	Sweep hyperparameters	1.4749
ViT	-	1.4657
ViT	Sweep hyperparameters	1.3125
ViT	Re-trained	1.2743
ViT	Post-processing	1.2156
ViT	Ensemble	1.1143
ViT	Post-processing	1.1087

Table 4.8: Competition Results

In the following table 4.8, the models and methods applied to them will be written, if the architecture is not changed or the same type of method as in the previous step is not changed, this means that the previous method is also applied even if it is not written for the given model. In particular, the models that influenced shifting and improving the results at the competition will be present. Re-trained in this case means that the model was repeatedly trained with the same parameters because the variance of the results was higher in order to continue with the best possible model.

From the final table, it is easy to see which procedures had the greatest influence on improving the results. Of course, there is the biggest room for improvement at the beginning, so you can see bigger jumps in the score achieved at the start. The implementation of the Stratified group K-fold, which improved the overall distribution of training and validation data, as well as augmentations with subsequent modifications of the loss function, had a great influence on the results. The last big improvement comes when the architecture itself is changed to Vision Transformer.

As you can see, some of the steps mentioned in the previous chapters are not listed in the table because they did not have a noticeable effect on the results at the given moment or did not provide improvement at all. The last row represents the last submitted model at the competition and thus our final model along with our final ranking. A graphic representation of the training process can be viewed in this report[2].

[2] `https://wandb.ai/scrower/resnet50_strat/reports/Results-Report--Vmlldzo0MjcwNzEx`

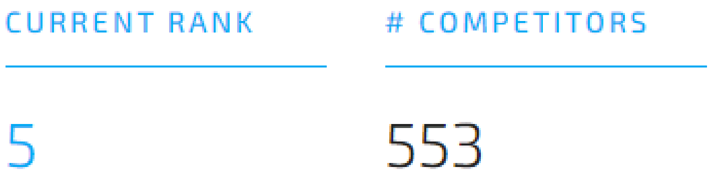

CURRENT RANK
5
COMPETITORS
553

Figure 4.22: Ranking in competition

At the time of writing this part and ending further participation and attempts to improve, our final place in the competition was fifth, out of a total of 553 competitors.

Chapter 5

Conclusions

This thesis shows that the application of computer vision methods to camera trap images is a useful tool for wildlife observation. For most projects that deal with monitoring animal populations and want to achieve improvements, it is certainly appropriate to include and implement computer vision methods and tools in purely ecological projects. The inclusion of these tools should save time spent working with the data and potentially allow researchers to obtain more information. The Vision Transformer shows great promise for use on the given dataset and, in this case, reaches the potential of outperforming otherwise well-established CNNs that are otherwise used for image classification. The most important changes going forward would be to improve the performance of small animals that are similar in size and to each other and also to reduce the dependence on the background of the image. In the competition, our best result achieved is 1.1087, as a score that comes from a loss function. Since this score includes the overall quality of the model's predictions, it is not entirely clear what the actual accuracy is on the test data set. For comparison, this result is worthy of fifth place out of about 500 competitors at the time of writing this.

Surprisingly, most of the architectures, except for the two finally used, did not perform well in this task. The advanced augmentations, which generally did not have a noticeable effect on improving the performance of the models, were equally unsatisfactory. Because the model can only be as good as its dataset. In order to improve the results, I would suggest that the competition provides a larger dataset, or if it is planned to use computer vision for this problem, the unification of the used camera traps and the selection of the places, where camera traps are installed. Using camera traps with computer vision in mind would certainly lead to improvement. Furthermore, possibly provide additional information regarding these images, such as the distance of these traps and their location or the hierarchy of the given animals that occur in the images. Relevant additional information can be used to improve results.

To further pursue this problem, I suggest trying the following experiments and other improvements. Try other custom models that might better fit this type of problem. Try connecting other branches of computer vision, such as object detection. Further division of the classification labels into smaller categories, for example, a specific type of rodent. Instead of using only one image, use a larger number of images from one event, where all images from the event would be classified as one label.

List of Tables

List of Figures

Bibliography

[1] A. Androff, “Trees are climate change, carbon storage heroes,” 2021. [Online]. Available: https://www.fs.usda.gov/features/trees-are-climate-change-carbon-storage-heroes

[2] “Pollination & human livelihoods.” [Online]. Available: https://www.fao.org/pollination/background/en/

[3] J. Tollefson, “Why deforestation and extinctions make pandemics more likely,” 2020. [Online]. Available: https://www.nature.com/articles/d41586-020-02341-1

[4] “Camera traps.” [Online]. Available: https://www.worldwildlife.org/initiatives/camera-traps

[5] “Zamba cloud.” [Online]. Available: https://www.zambacloud.com/

[6] “Conser-vision practice area: Image classification.” [Online]. Available: https://www.drivendata.org/competitions/87/competition-image-classification-wildlife-conservation/page/483/

[7] “Taï national park.” [Online]. Available: https://whc.unesco.org/en/list/195/

[8] “What is computer vision?” [Online]. Available: https://www.ibm.com/topics/computer-vision#:~:text=Resources-,What%20is%20computer%20vision%3F,recommendations%20based%20on%20that%20information.

[9] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, S. J. Sweeney, K. C. VerCauteren, N. P. Snow, J. M. Halseth, P. A. Di Salvo, J. S. Lewis, M. D. White *et al.*, “Machine learning to classify animal species in camera trap images: Applications in ecology,” *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 585–590, 2019.

[10] N. Rigoudy, A. Benyoub, A. Besnard, C. Birck, Y. Bollet, Y. Bunz, J. Chiffard Carriburu, G. Caussimont, E. Chetouane, P. Cornette *et al.*, “The deepfaune initiative: a collaborative effort towards the automatic identification of french fauna in camera-trap images,” *bioRxiv*, pp. 2022–03, 2022.

[11] S. Schneider, S. Greenberg, G. W. Taylor, and S. C. Kremer, “Three critical factors affecting automated image species recognition performance for camera traps,” *Ecology and evolution*, vol. 10, no. 7, pp. 3503–3517, 2020.

[12] R. C. Whytock, J. Świeżewski, J. A. Zwerts, T. Bara-Słupski, A. F. Koumba Pambo, M. Rogala, L. Bahaa-el din, K. Boekee, S. Brittain, A. W. Cardoso *et al.*, "Robust ecological analysis of camera trap data labelled by a machine learning model," *Methods in Ecology and Evolution*, vol. 12, no. 6, pp. 1080–1092, 2021.

[13] S. Brown, "Machine learning, explained," 2021. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 2009, pp. 248–255.

[17] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[19] H. Sajid, "What is image data augmentation?" 2022. [Online]. Available: https://www.picsellia.com/post/image-data-augmentation

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[22] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[23] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[24] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[25] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[28] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[29] M. Schlauch, "Introduction to image classification using camera trap images," 4 2022. [Online]. Available: https://drivendata.co/blog/conservision-benchmark/

[30] "Weights & biases – developer tools for ml." [Online]. Available: https://wandb.ai/site

[31] "Metacentrum (metavo) - virtuální organizace pro celou akademickou obec." [Online]. Available: https://metavo.metacentrum.cz/

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf