Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Image Augmentation Techniques for Marine Wildlife Detection in Aerial Images

Bachelor thesis
at Hamburger Informatik Technologie-Center e.V.
Dr. Tayfun Alpay
Dr. Sven Magg
Department of Informatics
MIN-Faculty
Universität Hamburg

submitted by
Thang Le Phuoc
Course of study: B.Sc. Wirtschaftssinformatik
Matrikelnr.: 7315507
on
30.03.2023

Examiners: Dr. Tayfun Alpay
Dr. Sven Magg

Abstract

Marine wildlife conservation is of paramount importance due to the alarming decline in global marine species populations. Accurate and efficient identification of marine species is challenged by factors such as imbalanced datasets, high-resolution images, and various distortions. This thesis explores the effectiveness of data augmentation techniques, including geometric, color-space, cutout, and mixed augmentations, in improving marine wildlife image classification and detection tasks. Additionally, it investigates the potential of Generative Adversarial Networks (GANs) to generate images and enhance the classification of underrepresented species. The study's results demonstrate that the effectiveness of augmentation techniques is highly dependent on the dataset, task, and model employed, with certain augmentations proving beneficial for specific models. The research also highlights the trade-offs between box loss and classification loss in using color-space augmentations and reveals the limitations of vanilla GANs and Wasserstein GANs. By optimizing data augmentation policies and leveraging GAN-generated augmentations, this thesis contributes to the development of more accurate and efficient marine species identification systems, ultimately supporting marine conservation efforts.

Zusammenfassung

Der Schutz der Seetieren hat aufgrund des alarmierenden Rückgangs der weltweiten Meeresarten-Populationen höchste Priorität. Eine genaue und effiziente Identifizierung von Meeresarten wird durch Faktoren wie unausgeglichene Datensätze, hochauflösende Bilder und verschiedene Verzerrungen erschwert. Diese Arbeit untersucht die Wirksamkeit von Data-Augmentation-Techniken, einschließlich geometrischer, farbraumbezogener, Ausschnitt- und gemischter Augmentationen, bei der Verbesserung von Bildklassifikations- und Detektionsaufgaben für Seetieren. Darüber hinaus wird das Potenzial von Generative Adversarial Networks (GANs) untersucht, um Bilder zu erzeugen und die Klassifizierung unterrepräsentierter Arten zu verbessern. Die Ergebnisse der Studie zeigen, dass die Wirksamkeit von Augmentationstechniken stark von Datensatz, Aufgabe und verwendetem Modell abhängig ist, wobei bestimmte Augmentationen für spezifische Modelle vorteilhaft sind. Die Untersuchung hebt auch die Kompromisse zwischen Box-Verlust und Klassifikations-Verlust bei der Verwendung von farbraumbezogenen Augmentationen hervor und zeigt die Grenzen von Vanilla GANs und Wasserstein GANs auf. Durch die Optimierung von Data-Augmentation-Richtlinien und die Nutzung von GAN-generierten Augmentationen trägt diese Arbeit zur Entwicklung genauerer und effizienterer Identifikationssysteme für Meeresarten bei und unterstützt letztendlich den Schutz der Seetieren.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Marine wildlife conservation is an important issue that requires more attention due to the significant decline in the population [29]. One of the key challenges in marine conservation is the accurate identification of marine species, which is often time-consuming and error-prone. With the rise of technology and the increasing availability of marine wildlife imagery, computer vision techniques, such as object detection and classification with deep learning, have emerged as a potential solution for accurate and efficient species identification and monitoring.

However, machine learning techniques are highly dependent on the amount and quantity of the data used. In the case of marine wildlife imagery, obtaining high-quality data is often difficult due to the challenging conditions and often requires a lot of manual works. An imbalanced dataset may occur when the number of images in a particular species is limited compared to others. Additionally, the complexity of marine wildlife aerial images also lies in high resolution and various types of distortions such as perspective distortion and non-uniform illumination.

To address these challenges, data augmentation techniques could be used to artificially increase the size of a data-set by creating variations of existing images. The primary goal of image augmentation is to increase the diversity of the training data and prevent over-fitting. Finding the optimal augmentation policy is challenging because it heavily depends on the dataset, and there is no heuristic to determine it.

Therefore, this thesis aims to investigate the effectiveness of various types of augmentations, including geometric, color-space, cutout, and mixed, for marine wildlife imagery in both classification and detection tasks. Additionally, the thesis will investigate the potential use of Generative Adversarial Networks (GANs) to generate images and enhance the classification of underrepresented species.

1.2 Problem Statement

In the absence of a heuristic approach, the search for an optimal augmentation policy is highly empirical and dependent on the specific dataset. The primary objective of this study is to evaluate suitable augmentation techniques for the marine wildlife dataset by employing basic augmentation methods and exploring the effectiveness of GAN-generated images to improve the classification of underrepresented species. Additionally, this study aims to investigate whether an augmentation technique that enhances the performance of one task might necessarily be beneficial for another, and whether the effectiveness of the augmentation policy is model-agnostic. To sum up, this study seeks to address the following questions:

- How is the effectiveness of common augmentation (geometric, color-space, cutout) for the maritime wildlife aerial image data-set in detection and classification tasks?
- Is the augmentation effectiveness only dependent by the dataset and agnostic to the model used and the tasks?
- If and how low-data classes may benefit from the GAN-synthetic method?

1.3 Thesis Layout

The thesis has six chapters.

- Chapter 1 presents the motivations for exploring augmentation techniques, the problem statement, and the overall structure of the thesis.
- Chapter 2 serves as a source of relevant background information for the thesis, including an overview of related works, a detailed description of the marine wildlife dataset, and an introduction to important concepts in Generative Adversarial Network (GAN).
- Chapter 3 of the thesis covers the fundamental concepts of basic augmentation techniques, providing a detailed description of the basic augmentations and a systematic approach to tuning their hyperparameters.
- Chapter 4 outlines the experimental approach, including the evaluation metrics and models employed, as well as the training pipelines. The training pipelines are divided into two categories: the experiment pipeline for basic augmentations and the GAN-generated pipeline, where the specific architectures of GANs will be discussed.
- Chapter 5 presents the outcomes of the experiments and subsequent analysis.
- Chapter 6 summarized the findings from the experiments, outlines the limitations of the thesis, and proposes potential avenues for future research.

Chapter 2

Background

This chapter presents an overview of state-of-the-art image augmentation techniques such as data warping, oversampling, and undersampling. Furthermore, it offers a detailed description of our marine wildlife dataset. Finally, the chapter includes a short theoretical introduction to Generative Adversarial Networks (GANs), encompassing regular GAN, Wasserstein GAN (WGAN), and Wasserstein GAN with Gradient Penalty (WGAN-GP).

2.1 Related work

A colored image is composed of three channels: red, green, and blue, each of which has a value range from 0 to 255. By subtracting one value of a pixel by 1, the image can be considered as changed and be read by a model as new data. That means there are unlimited image augmentation techniques. The challenge now is how to select efficient augmentation methods. This question is an active topic in Computer Vision as the field of machine learning is continuously exploring new ideas and challenging existing techniques.

In this section, we aim to provide an overview of the current state-of-the-art augmentation techniques. To enhance readability and comprehensibility, we have grouped these techniques into two main categories: data warping, oversampling and undersampling.

2.1.1 Data warping

One of the earliest and most commonly used image augmentation techniques is data warping. Data warping simply means transforming existing images to create new ones. This includes augmentations such as geometric transformations (e.g. flipping, rotation, and translation), color transformations (e.g. blocking one or two of the red, green, and blue channels), and random crop (cropping patches from each image). These techniques are relatively simple to implement and have been shown to be effective in improving the performance of machine learning models on many data-sets [34]. However, there are several more advanced data warping techniques

used in augmentation beyond basic ones such as AutoAugment, RandAugment, Cutout, Random Erasing, Hide and Seek and Grid Mask.

AutoAugment, Fast AutoAugment, RandAugment

AutoAugment is a relatively recent image augmentation technique that uses reinforcement learning to find the optimal set of augmentations to apply to an image [11]. This augmentation technique tackle the questions "if there are many augmentations, which ones should be implemented and with what maginitude?". The authors made a space for searching where there are different sets of rules, each with a way of changing an image, like flipping or changing colors. To find the best combination of these rules, a reinforcement learning approach is implemented to navigate this search space. The ideal augmentation policy is determined by the probabilities and magnitudes of the functions. While AutoAugment can produce effective policies, it is computationally expensive and requires significant resources, such as 1500 GPU hours to explore a good policy for the ImageNet dataset [26].

In an effort to address this limitation, Lim et al. proposed Fast AutoAugment [26], which reduces the search phase and allows for adjustment based on different model sizes. This results in a more efficient and less computationally intensive method with improved regularization strength, requiring less than one-third of the time required by AutoAugment.

A year after the creation of AutoAugment, Cubuk et al. introduced RandAugment [12], which addressed two of AutoAugment's disadvantages. The separate search phase of AutoAugment incurred increased computational cost and weak regularization strength for varying model sizes. RandAugment is a similar technique, but it gives up reinforcement learning completely. Instead, RandAugment uses random sampling. This allows RandAugment to be applied directly to the target task and be adjusted according to different model sizes. RandAugment is extremely fast and efficient. It is still being used as common practice today.

Cutout, Random Erasing, Hide and Seek, GridDropout

Recently, inspired by the effectiveness of the dropout layers in convolutional networks [37], image augmentation techniques such as Cutout, Random Erasing, and Hide And Seek are extensively researched. Cutout, Random Erasing, and Hide And Seek are image augmentation techniques that involve removing parts of images to create new ones. Cutout involves blacking out square regions of an image [14], while random erasing covers certain parts of an image with noise, and the size and number of parts are randomly chosen [48]. Hide and seek involves hiding and then revealing parts of an image, essentially making the classifier focus on different parts of the original image [36]. A more systematic drop-out strategy is Gridmask. GridMask masks out rectangular regions of an image in a grid pattern, forcing the model to learn to recognize objects and features in different positions within the image [9]. Similar to weight regularization in dropout layers, these techniques encourage the model to learn from the surrounding context of an image instead

of focusing solely on specific features. By randomly removing parts of an image, the model is encouraged to be more robust and better equipped to handle missing information. This is particularly useful when dealing with real-world scenarios where images may not always be complete or clear, as increasing the diversity of the training set can lead to more accurate and reliable models.

Adversarial attacks

Another approach to make the dataset more robust is to attack its weakness, such as Adversarial training. Adversarial attacks are techniques used to manipulate input data in such a way that it is misclassified by the model. So adversarial training works by introducing carefully crafted adversarial attacks into the training data of the model, which causes mislabelling by the model. The model will be punished and then learns to recognize these adversarial examples and adjust its parameters accordingly, in order to improve its performance on both clean and adversarial data [35].

Neural style transfer

The most recent and novel approach is augmentation using neural style transfer [17]. This is a type of deep learning-based augmentation that changes the look of an image by applying a certain style. To use neural style transfer for image augmentation, we first need to train a style transfer model on a set of reference images with different styles. During training, the model learns how to extract the style of an image and apply it to a different image while preserving the content. However, selecting the appropriate style for image generation can be a challenging task, often requiring a significant amount of computational resources. As a result, achieving high-quality and realistic images can be quite difficult.

2.1.2 Oversampling and Undersampling

Apart from data warping, another group of image augmentation techniques involves oversampling and undersampling. The common of this group is not to transform an existing image, but to create new images based on existing ones. While oversampling refers to the creation of new images from existing ones by adding noise or altering specific features, undersampling involves removing parts of images to create new ones. These techniques can be used to balance the number of samples in different classes of a data-set, which can be useful in preventing class imbalance problems. This group of data augmentations includes mixing images, feature space augmentation, and using AI networks e.g GAN-based augmentation.

Mixing Images

We can also combine different images in the same class to do augmentation. Mixing images is a type of data augmentation that combines two or more images to create a new image [39]. There are different methods for mixing images, including simple

blending, weighted blending, and alpha blending. Simple blending involves overlaying two images on top of each other using a basic blending function. Weighted blending assigns different weights to each image based on their importance or relevance to the task at hand. Alpha blending is a more sophisticated technique that involves creating a transparent mask that blends two images based on their pixel intensities.

This new image is then used as additional training data. This helps to increase the diversity of the data and can lead to better performance of the model being trained. By mixing images after image processing, the error rate on the CIFAR-10 data-set was reduced from 5.4% to 3.8% [25]. However, the problem with Mixing images is its randomness. It combines pixel values from different images without taking into account the important features, which can result in the inclusion of unwanted noise from the original images.

Augmentations in latent space

All the discussed data augmentation techniques mentioned above are used on the original images before they are processed. But by using neural networks, we can reduce the size of images and augment directly the representation in latent-space. Autoencoders have two parts, an encoder that simplifies the image, and a decoder that rebuilds the image back to its original size. By simplifying the image, the autoencoder can find the important parts and relationships between the pixels, which can then be used to make changes to the image and improve its accuracy. This information can then be used to enhance the features in the latent space, making it a useful tool for data augmentation. For examples, DeVries and Taylor used k-nearest neighbors to generate new images in the latent space and saw positive results on five different types of data sets [42].

GAN-based Augmentations

Latent space augmentation can sometimes result in features with too much noise. Generative Adversarial Networks (GAN), introduced by Ian Goodfellow in 2014 [18], is another exciting strategy for data augmentation. GAN-generated methods can help in augmentation by generating synthetic data that can be used to supplement the original training data, especially in cases where the training data is limited or imbalanced. The GAN model can learn the underlying distribution of the training data and generate new data samples that are similar to the real data but have different features that can help the model generalize better.

GAN-based data augmentation involves two parts: a generator and a discriminator. The generator is trained to create images from input that the discriminator can't identify. The discriminator then tries to recognize the labels of the generated images. Eventually, the generator gets better and creates images that keep the original label even after being augmented. A vanilla GAN is refered to the GAN architecture which uses multilayer perceptron networks in the generator and discriminator networks. GAN-based data augmentation is an ongoing area of re-

search, with various modifications made to the GAN framework such as DCGANs [32], which utilize convolutional neural networks as the architecture, Wasserstein GAN (WGAN) [3] which uses Wasserstein distance as loss function or CycleGANs [49], which use two GANs working in tandem to match two domains to each other without paired training data.

Chou's work in 2019 demonstrated that using a simple GAN structure for data augmentation improved the performance of sorting the defective beans task, outperforming HCADIS and YOLOv3 [10]. Similarly, in 2021, Dimoiu et al. used conditional GANs to enhance the performance of aerial image segmentation in a real context of a rural zone in Romania, with their proposed scheme outperforming other GAN implementations [15]. In 2019, Zhang and colleagues used a Wasserstein GAN to handle unequal image data in detecting weld defects [47]. Both of their experiments on augmented images and real world defect images achieve satisfying accuracy, which substantiates the possibility that the proposed approach is promising for weld defect detection.

GANs have the potential to be very powerful, but GANs are computationally expensive and require a significant amount of resources, such as high-end GPUs and large amounts of data. This makes it difficult for researchers with limited resources to train and generate high-quality images. Addtionally, GANs may not always be effective as an augmentation tool because GAN-generated data may not cover the entire distribution of the original data. This can result in overfitting to the limited variation present in the GAN-generated images, which can lead to poor generalization performance, particularly when there are differences in the distribution between the training and test sets. These factors can make it challenging to use GANs effectively for data augmentation purposes.

Downsampling

In contrast to generating new data, downsampling presents an alternative approach for addressing the issue of an unbalanced dataset. Downsampling involves randomly deleting samples from the majority class with sufficient samples, in order to achieve an equal number of samples across all classes, or reducing the image resolution to expedite processing time. In the former case, this approach helps balance the number of samples across the classes, thereby mitigating biases towards dominant classes. In the latter, downsampling can prove useful in scenarios where the original image resolution is excessively high, leading to computationally intensive processing times. Reducing the image size enables significant processing time reduction, while retaining the essential image features. Although this method is easy to implement and helps with the computational feasibility, there is a high risk that the deleted samples contain key informative data.

Despite significant progress in the field of image augmentations, there remain several limitations that must be addressed. One such limitation is the need for a dataset-specific augmentation policy. It can be challenging to develop a strategy that is effective for all types of images, given the inherent variations and invariances that may exist within each dataset. For instance, in the case of aerial image

datasets, issues such as center biases and non-uniform illuminations must be carefully considered and understood to develop effective augmentation strategies.

2.2 Dataset

The dataset used in this thesis is a subset of a larger dataset obtained from a company in Hamburg. The dataset is a collection of aerial images captured by manned aerial vehicles. The images were obtained at different times of the day and year and under varying weather conditions, resulting in a diverse set of environmental and lighting conditions, e.g sometimes the water is blue, sometimes gray.

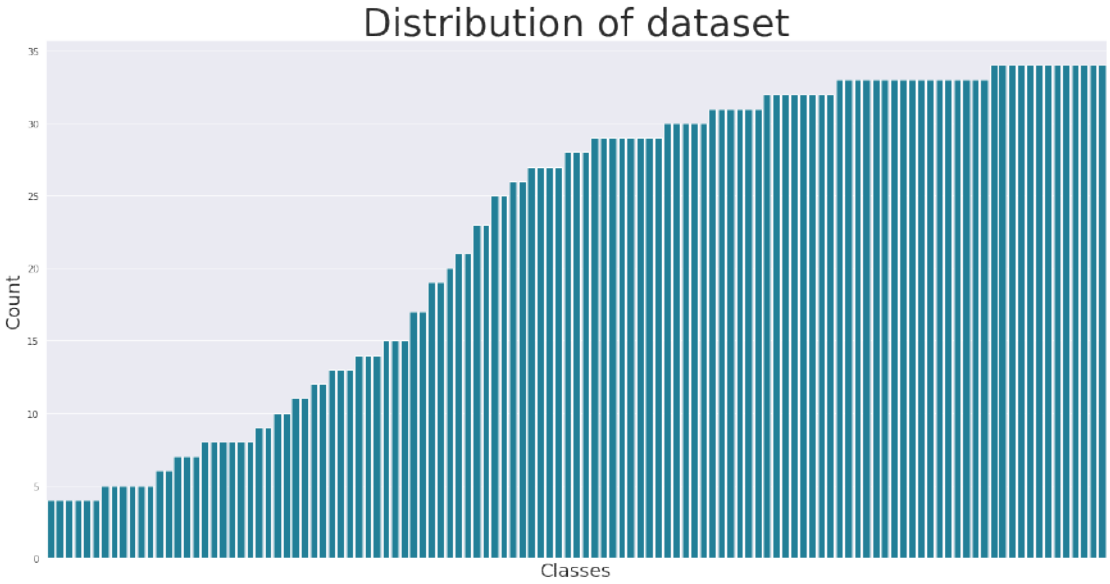


Figure 2.1: Distribution of the dataset

The dataset comprises more than 2,700 images with a resolution of 500 x 500 pixels. Ornithologist annotated the images to identify the marine wildlife and anthropological artifacts present in each image. The objects of interest include different species of marine mammals, sea turtles, and various bird species. Illustrated in Figure 2.1, the dataset consists of 118 classes and is divided into four primary types: anthropological artifacts, birds, mammals, and miscellaneous. The numerical statistics of each type is displayed in Table 2.1. Figure 2.2 shows some examples from each big types. A full description of the dataset is shown in Figure A in the appendix.

In order to train YOLOv5 on the detection dataset, the images are required to be accompanied by labels and masks of the objects. However, since YOLOv5 can only accept bounding boxes, the masks were converted into coordinate and size formats that conform to the YOLOv5 format (x, y, height, width). Figure 2.3 shows a sample training batch.

To improve practicality and reduce training time, the image resolution for the experiment was to 256x256 pixels. This dataset was then used to train and evaluate

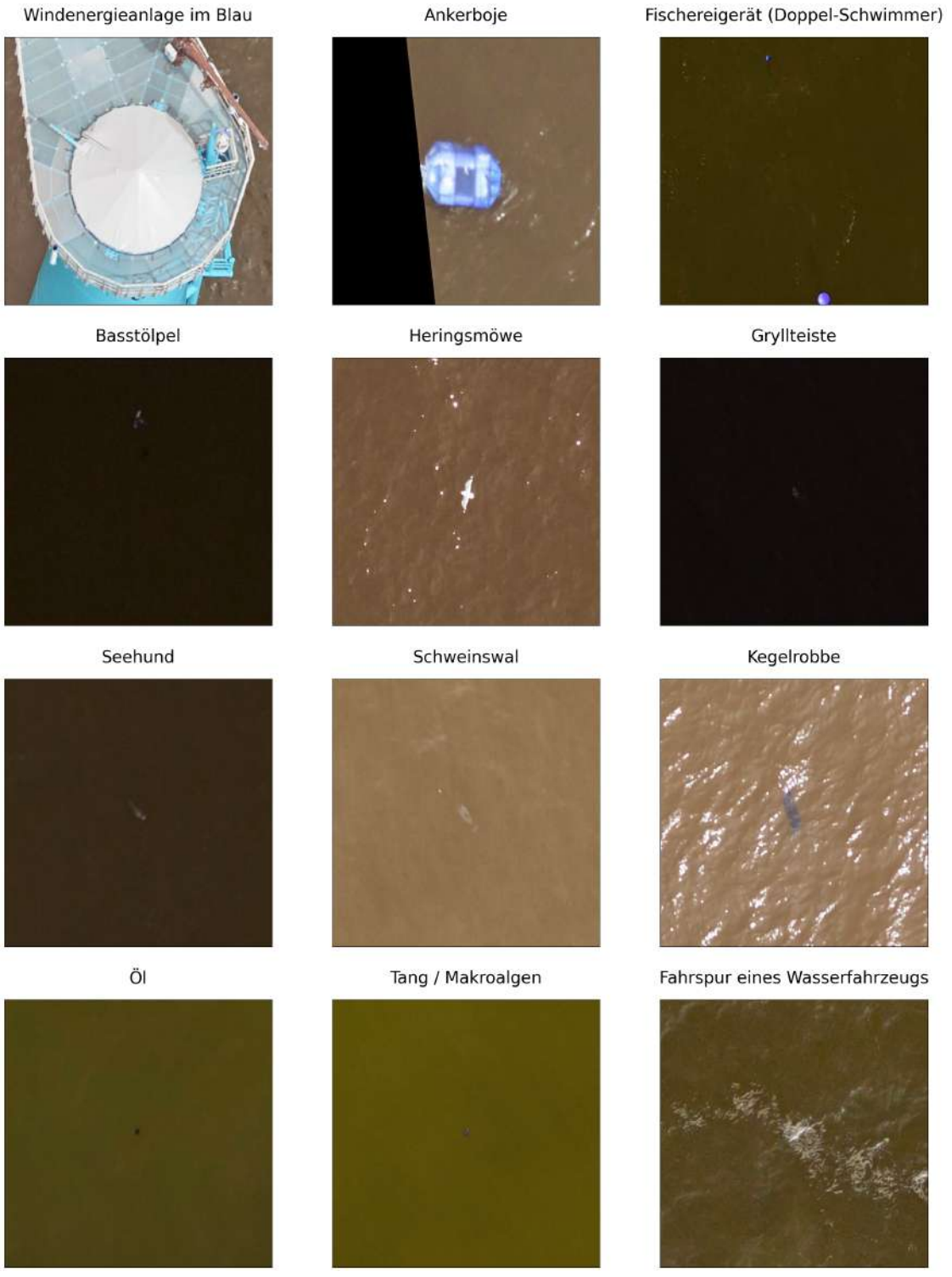


Figure 2.2: Examples of the data classes in the dataset. The top row displays the anthropological artifacts, followed by birds in the second row, mammals in the third row, and miscellaneous in the last row.

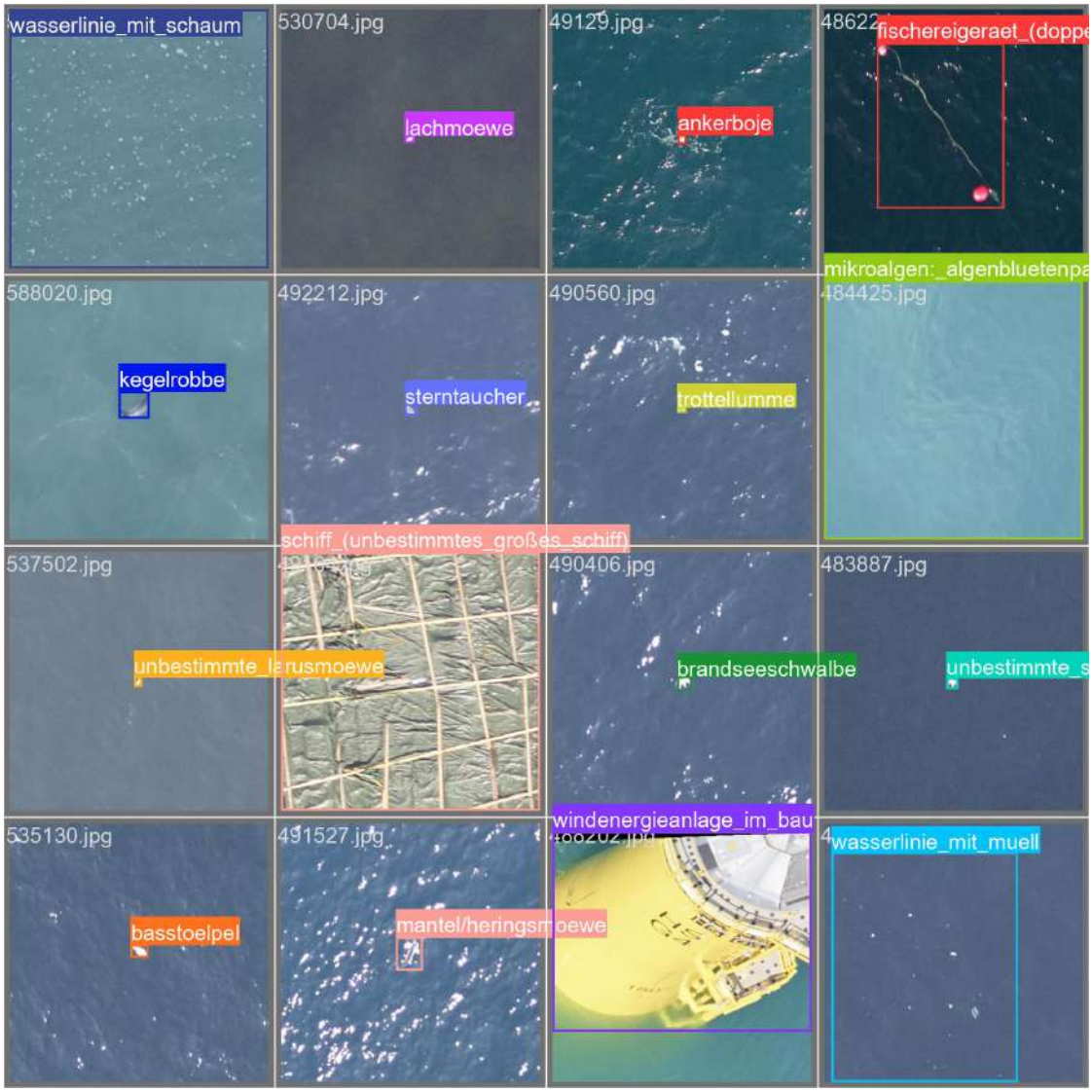


Figure 2.3: Examples of a training batch in YOLOv5s with our dataset

Yolov5s for object detection and EfficientNet version B0 for classification of marine wildlife in aerial images.

Type	Number of classes	Numer of samples
Anthropological Artifact	18	331
Bird	74	1729
Mammals	7	189
Miscellaneous	18	423

Table 2.1: Description of the dataset according to types

Prior knowledge of datasets

Prior knowledge about the dataset is important for developing effective machine learning models. In the context of our marine wildlife detection and classification task, we have observed that our dataset contains center bias, meaning that the objects of interest are often located in the center of the image. This is important to keep in mind when designing our object detection models, as the model may have a tendency to focus on the center of the image and may struggle with detecting objects that are located near the edges of the image.

Another challenge is that the object of interest, i.e., the marine wildlife, is usually very small when compared to the background of the image. This makes it difficult for the model to identify the object of interest and distinguish it from the background. It is also important to note that most of the images in our dataset contain only one object, i.e., a single bird. However, there are also images where a flock of birds can be seen. This introduces another layer of complexity in object detection, as the model needs to be able to identify and differentiate between individual objects within a group. Luckily, one advantage of our dataset is that it is not affected by occlusion from clouds or bad weather or motion blurs, allowing for clearer images of the birds.

Understanding these characteristics of the dataset can inform our selection and application of data augmentation techniques to improve the robustness of our models. For example, we may choose to apply CenterCrop or Perspective to address the center bias in our dataset, or use sharpen or CLAHE to enhance the visibility of small objects in the images. But while doing that, we have to be careful to see if the performance in pictures of a flock of bird decreases. By taking these factors into consideration, we can better prepare our models for the challenges of our specific dataset and improve their overall performance.

Underpresented classes (edge cases)

It is crucial to pay attention to the underrepresented classes (edge cases) in our dataset since they can have a significant impact on the performance of the machine learning models. In this thesis, we will compare the performance of our models before and after augmenting the dataset with GAN-generated images. We need to ensure that our evaluation includes extreme cases, where some classes have a small number of samples, such as less than 10 samples, to account for edge cases in our experiments. By doing so, we can determine if the GAN-generated images have a dramatic effect on the performance of the models, especially on underrepresented classes. Table 2.2 shows some examples of these edge classes.

Dataset stratification

When splitting our dataset, we must use stratified sampling instead of random sampling to ensure that the proportion of each class in the training and testing sets is approximately the same as that in the entire dataset. This is because our dataset is highly imbalanced. The intuition behind this relates to the bias of most

Name	Number of samples	Type
Grylllteiste	4	Bird
Schnatterente	4	Bird
unbestimmte Larusmöwe	4	Bird
Schmarotzer/Spatel/Falkenraubmöwe	5	Bird
Wasserlinie mit Großalgen	5	Miscellaneous
Feldlerche	5	Bird
Schmarotzerraubmöwe	5	Bird
Grosser Brachvogel	5	Bird
unbestimmte Raubmöwe	7	Birds
Turmfalke	7	Bird
Trauerseeschwalbe	7	Bird
unbestimmter Schwan	8	Bird

Table 2.2: Some underpresented classes (edge classes) that will be used as input for our GAN model

classification algorithms. They tend to weight each instance equally which means overrepresented classes get too much weight (e.g. optimizing F-measure, Accuracy or a complementary form of error) [31]. By performing dataset stratification, we split the dataset 80% into training and minimum 20% into validation sets so they guarantees the representation of all classes in both subsets. Failure to ensure such stratification may result in the training and validation sets having different class distributions, leading to biased evaluations of the model's performance.

2.3 Generative Adversarial Network (GAN)

In this section, a brief overview of Generative Adversarial Network (GAN) is presented. The fundamental concept of the minmax game in GAN is introduced along with a discussion of different GAN architectures, such as the deep conditional network used in DCGAN or the multilayer-perceptrons employed in Vanilla GAN. Additionally, this section provides an introduction to Wasserstein GAN and Wasserstein GAN with Gradient Punishment, which use the Wasserstein distance as a loss function instead of the traditional binary cross-entropy loss.

2.3.1 Core ideas

Basic GAN

Generative Adversarial Network (GAN) are a type of deep learning algorithm that can be used to generate new data that mimics the characteristics of a given

dataset. GANs consist of two neural networks, a generator network and a discriminator network, that are trained together in a two-player minimax game. Figure 2.4 illustrates an example of a GAN structure.

During training, the generator and discriminator models work together. The generator generates a batch of samples, which are then combined with real examples from the domain and presented to the discriminator to be classified as either real or fake. The discriminator is updated to improve its ability to distinguish between real and fake samples in the next round. Importantly, the generator is updated based on how successful it was at fooling the discriminator with its generated samples. In this case, zero-sum means that when the discriminator successfully identifies real and fake samples, it is rewarded or no change is needed to the model parameters, whereas the generator is penalized with large updates to model parameters. Alternately, when the generator fools the discriminator, it is rewarded, or no change is needed to the model parameters, but the discriminator is penalized and its model parameters are updated.

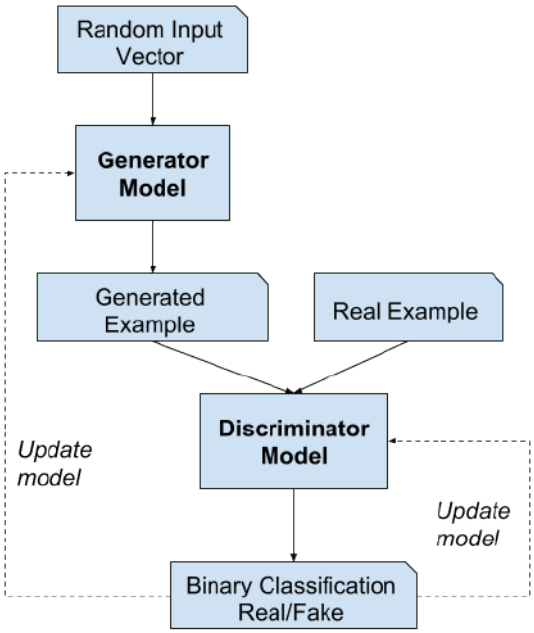


Figure 2.4: Example of the Generative Adversarial Network Model Architecture [8]

Conditional Generative Adversarial Networks (cGAN)

A conditional Generative Adversarial Network (cGAN) is a type of GAN that includes additional information in the input. Instead of just using noise as input for the generator, a conditional GAN appends a one-hot vector that indicates the class of the input. Similarly, for the discriminator, one-hot vectors are appended to the picture to predict the class as well. This allows the generator to generate samples conditioned on specific classes, resulting in more realistic and targeted samples. Figure 2.5 illustrates the example of a conditional GAN.

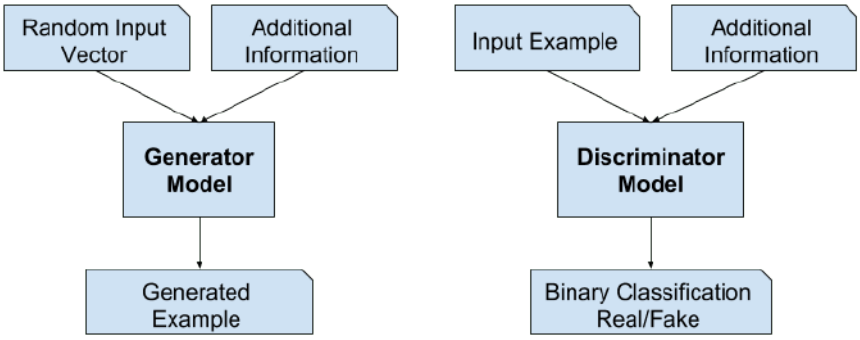


Figure 2.5: Example of the conditional Generative Adversarial Network Model Architecture [8]

2.3.2 GAN Implementations

The Vanilla GAN architecture, which solely consists of multi-layer perceptrons, often yields low-resolution images. In contrast, DCGAN addresses this issue by utilizing convolutional neural networks (CNNs) for both the generator and discriminator. Furthermore, the Vanilla GAN suffers from mode collapse, where the generator produces a limited range of samples that fail to represent the entire distribution of the training data. To overcome this limitation, Wasserstein distance is used as the loss function instead of the traditional cross-entropy loss (Kullback-Leibler divergence). In this section, we will provide an overview of these improvements.

DCGAN

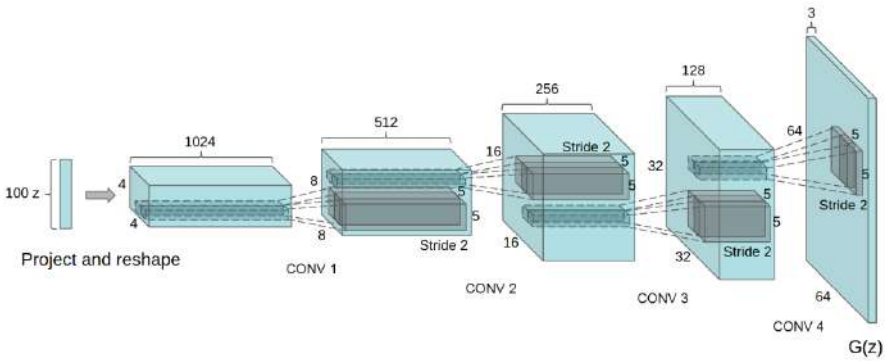


Figure 2.6: Architecture of DCGAN Generator [32]

DCGAN [32] is a GAN architecture that uses convolutional neural networks (CNNs) as both the generator and discriminator networks to generate high-quality images. It follows a set of guidelines, such as using strided convolutions instead of pooling layers, batch normalization in both generator and discriminator, and ReLU activation in the generator with Tanh for the output, and LeakyReLU activation

in the discriminator for all layers. Figure 2.6 shows the architecture of DCGAN generator.

WGAN

Wasserstein GAN (WGAN) is a type of GAN that uses Wasserstein distance instead of traditional loss functions like cross-entropy. Wasserstein distance, also known as earth mover's distance, is a measure of distance between two probability distributions. It measures the amount of "work" or "distance" required to transform one distribution into the other. In other words, it quantifies the dissimilarity between two distributions. This not only mitigates the problem of mode collapse, where the generator produces a limited range of samples, but also establishes a reliable stopping point for training when the distance falls below a certain threshold.

Figure 2.7 provides a demonstration of the vanishing gradient problem in regular GANs and the effectiveness of Wasserstein distance in addressing this issue [3]. It reveals that the gradient of the GAN discriminator becomes unstable and either explodes or diminishes, which negatively affects the learning process of the generator. In contrast, the WGAN critic is trained using Wasserstein distance as the loss function, leading to a smoother and more stable gradient.

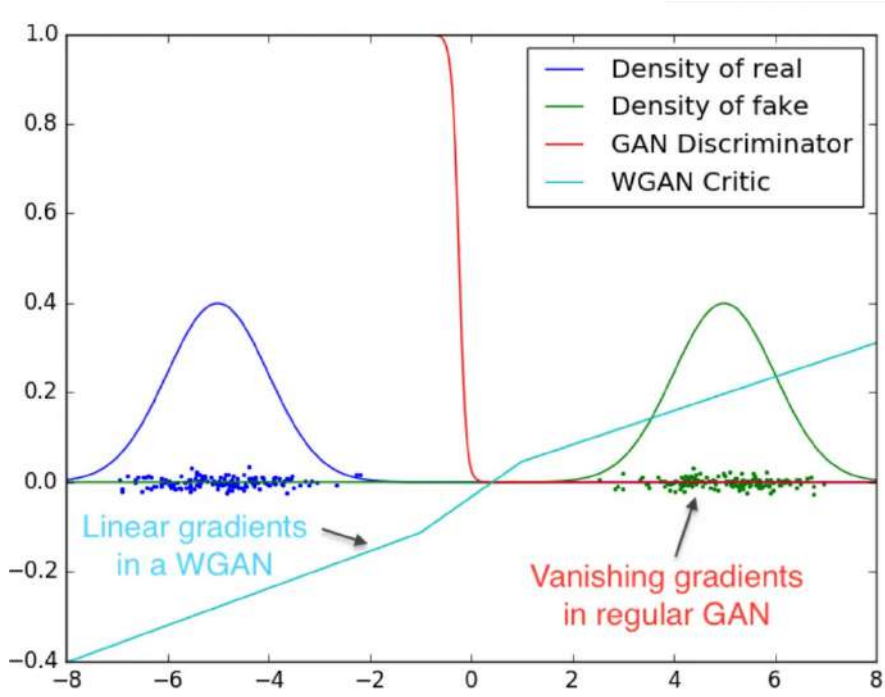


Figure 2.7: Illustration of Wasserstein Distance[3]

WGAN-GP

To enforce the Lipschitz constraint (the gradients do not explode during training), weight clipping is used in WGAN. However, one of the major problems with weight

clipping is that it can lead to vanishing gradients, which can slow down the training process and result in poor performance. Figure 2.8 illustrates this hyperparamter-sensitive problem.

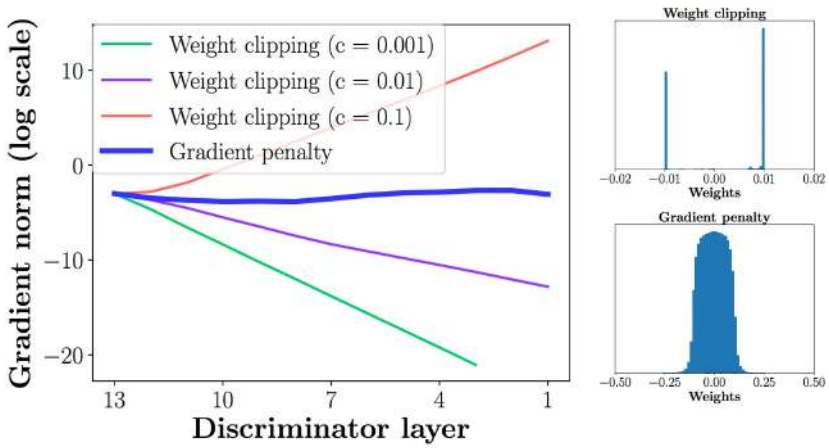


Figure 2.8: Illustration of the sensitivity problem of gradient to weight clipping in regular WGAN [19]

Wasserstein GAN with Gradient Penalty (WGAN-GP) is a modification to the original WGAN proposed to address the issue of gradient explosions or vanishing gradients during the training of WGANs. The WGAN-GP method replaces the weight clipping constraint used in WGAN with a gradient penalty term that enforces the Lipschitz continuity of the critic function. Specifically, WGAN-GP penalizes the model if the gradient norm moves away from its target norm value 1.

Chapter 3

Fundamentals of basic augmentation techniques

3.1 Augmentation Techniques Backgrounds

This section introduces four groups of commonly used image augmentation techniques in computer vision: geometric, color-space, cutout and mixed. Each technique is demonstrated using three images of birds, including Genus Fulmarus (Eissturmvogel), Common Goldeneye (Schnellente), and Great Cormorant (Kormoran). These three images are selected to demonstrate the effects in scenarios where the object of interest is clear (Kormoran), the object shape is small (Eissturmvogel), and when working with a group of birds (Schnellente).

3.1.1 Geometric augmentation

Geometric augmentations are a type of image augmentation technique used in computer vision that modify the geometric properties of an image, such as its size, position, orientation, and perspective. In this section, we will focus on five common geometric augmentations that are particularly useful for aerial images: Cropout, Flipping, Motion Blur, Perspective and Rotation.

CenterCrop

CenterCrop transformation is a type of image augmentation technique that involves cropping an image from its center to generate a smaller sub-image. This transformation can be used to remove irrelevant areas from the image and focus on the most important features. The CenterCrop transformation can also help to standardize the size of images used for training machine learning models. The CenterCrop transformation can be advantageous for aerial images, as it can eliminate irrelevant background areas, enabling the focus on essential features. This is particularly useful for our dataset, which contains numerous images with center biases.

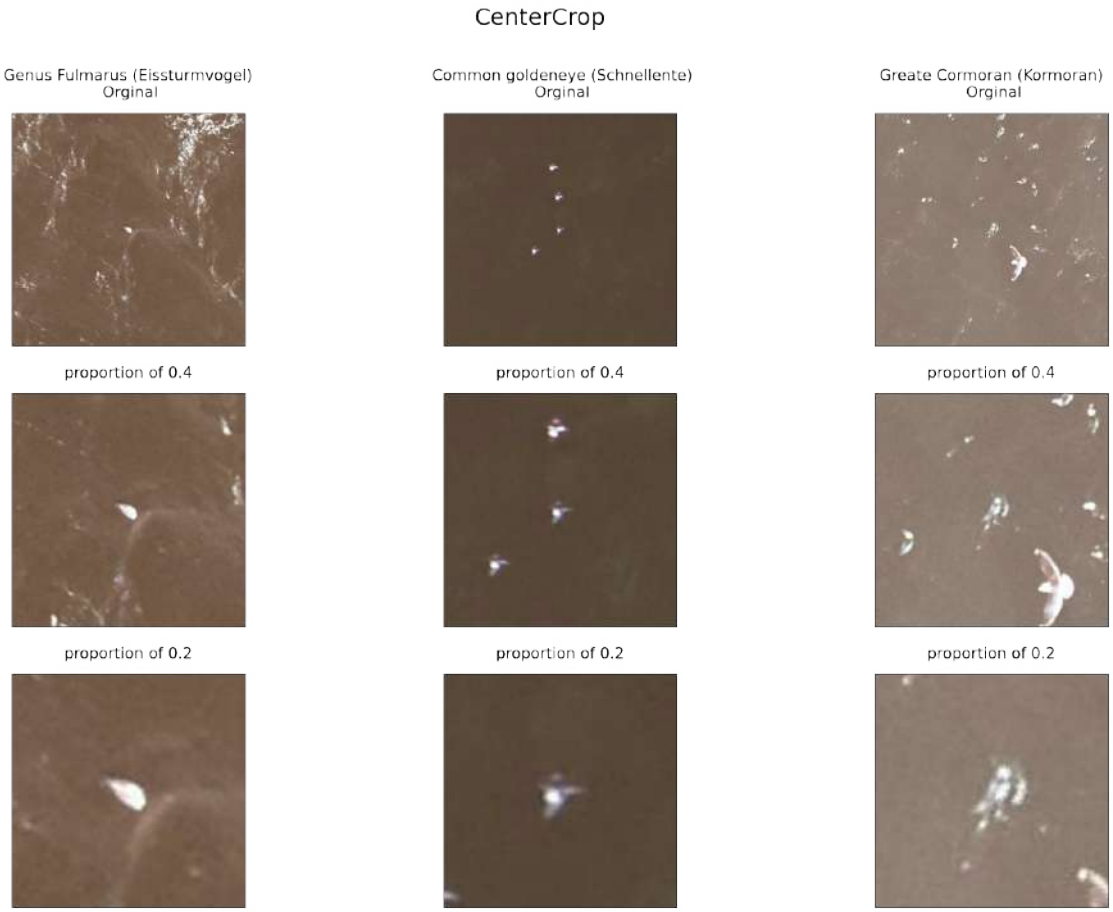


Figure 3.1: Example of CenterCrop with proportion of 0.2 and 0.4

However, it is crucial to note that in some pictures where there is no center bias, such as the Kormoran image in Figure 3.1 or images showing a flock of birds, this transformation may remove important information from the image. Therefore, it is necessary to evaluate each image carefully and determine whether the CenterCrop transformation is suitable, as its usage may depend on the image's characteristics.

In our experiment, we will apply CenterCrop transformation to our aerial images, using proportions of 0.2 and 0.4, as shown in Figure 3.1. This means that the center of the image will be cropped by one-fifth and two-fifths of its width and height. Then the image will be resized to the original size.

Flipping

Flipping transformation is an image augmentation technique that flips an image horizontally or vertically. It involves reflecting the image along a vertical or horizontal axis, which results in a mirror image of the original image.

The advantage of flipping is that it is a simple transformation that does not risk any loss of information. However, it is limited in terms of variety because it can only create mirrored versions of existing examples, which may not be sufficient

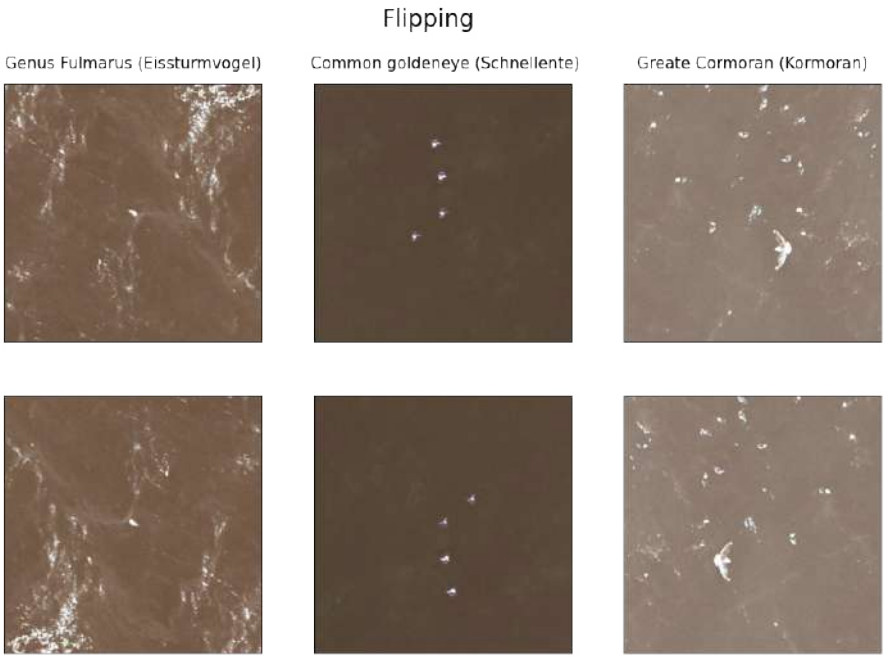


Figure 3.2: Examples of Flipping

for capturing the full range of variations in the dataset. Figure 3.2 illustrates the efffect of the transformations.

Motion Blur

Motion Blur transformation is an image augmentation technique that simulates Motion Blur in an image. The Motion Blur transformation works by applying a blur filter to the image, which simulates the effect of an object in motion. This is accomplished by convolving the image with a kernel function that creates the blur effect. Figure 3.3 illustrated the blur transformation. The size and direction of the kernel function can be varied to control the strength and direction of the Motion Blur effect. The blur limit in Motion Blur augmentation refers to the maximum amount of blur that can be applied to an image. This limit can be defined in terms of the size and direction of the kernel function used to apply the blur effect.

One advantage of Motion Blur transformation is that it can help to generate a more diverse set of training images for machine learning models. By varying the size and direction of the kernel function, it is possible to generate a large number of images with different types and strengths of Motion Blur. This can help improve the accuracy and generalization of the trained model.

While Motion Blur transformation can be useful for generating a diverse set of training images, it does have some significant drawbacks, particularly for object detection in aerial images of marine wildlife. A significant disadvantage of this technique is that it can reduce the image's sharpness and detail, making it more challenging to detect or recognize objects in the image. In the case of aerial images of marine wildlife, where shape and detail are essential for accurate detection,

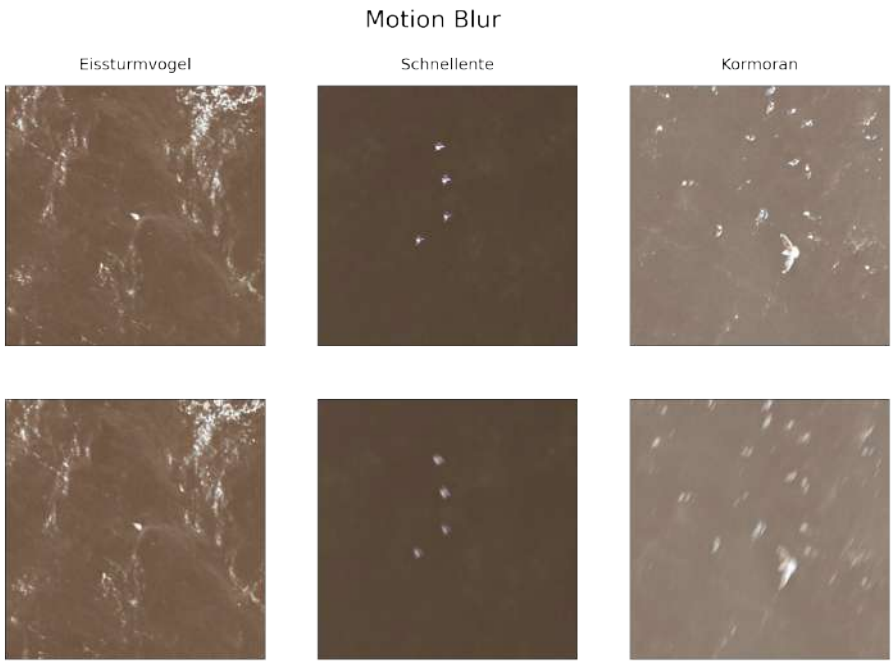


Figure 3.3: Examples of Motion Blur with blur litmit of 30

Motion Blur transformation can negatively impact the performance of the machine learning model.

Perspective

Perspective transformation is an image augmentation technique that involves applying a geometric transformation to an image in order to alter its perspective. To perform perspective transformation, the algorithm first defines a set of four source points and a set of four destination points that define the corners of the original and transformed image, respectively. Then, it computes a transformation matrix that maps the source points to the destination points using techniques such as homography or projective geometry. An example of perspective transformation is the transformation of a squared image of a square into a trapezoid shape, then resize it to their original size. This transformation can create the illusion of a 3D perspective in a 2D image.

Hyperparameter of Perspective includes random scale factor. The scale factor is used to determine the new points' distance from the original image corners during the transformation, which is calculated by sampling from a normal distribution with a mean of zero and a standard deviation set by the scale factor. A larger scale factor leads to a more pronounced transformation. In the experiment, the scale factor will be tuned, the effect of Perspective transformation is shown in Figure 3.4.

It should be noted that in the case of aerial images, perspective transformation may not be as effective. Aerial images are typically captured from a top-down perspective, where the camera is fixed in a position and oriented straight downward.

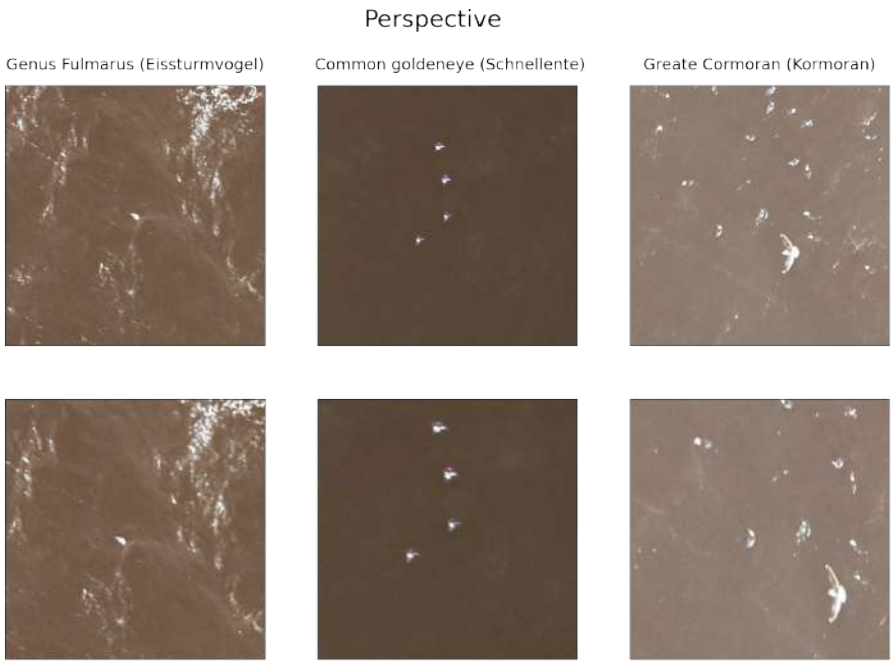


Figure 3.4: Example of Perspective with random scale between 0.1 and 0.2

As a result, the scene in the image is already in a consistent perspective, and applying a perspective transformation may not yield significant benefits.

Rotation

Rotation transformation is an image augmentation technique that involves rotating an image by a certain angle. The rotation can be performed in either a clockwise or counter-clockwise direction.

One advantage of rotation transformation is that it can increase the diversity of the training data without risking any information loss. This is particular useful for our dataset because marine wildlife aerial images often have objects in different orientations and angles. By applying rotation transformations, the model can learn to recognize the same object in various angles, which improves the model's robustness and accuracy in real-world scenarios. Additionally, rotation transformation is a simple and computationally efficient technique that can generate a wide variety of training examples. In this thesis, we apply a rotation transformation to our images within a range of -90 to 90 degrees. The impact of this transformation is demonstrated in Figure 3.5.

3.1.2 Color-space augmentation

Color-space augmentation is an image augmentation technique that involves altering the color channels of an image to create new training images for machine learning models. This technique can be used to modify the color distribution of images, introduce color variations, and aid models in adapting to various lighting

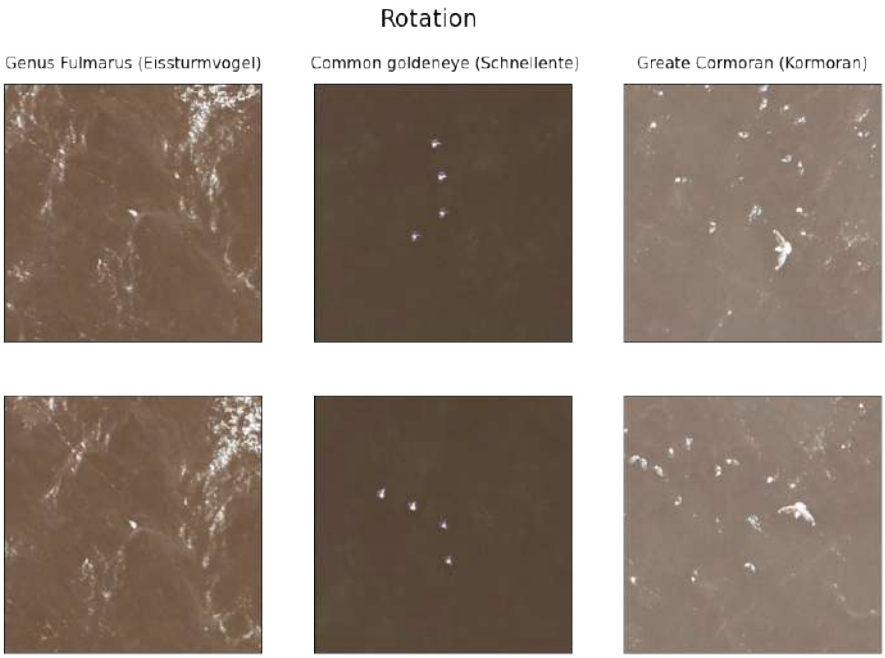


Figure 3.5: Examples of Rotation

conditions. In this section, we will present Contrast Limited Adaptive Histogram Equalization (CLAHE), Sharpen, Channel Shuffling, Color Jitter, Solarize, ToGray, ToSepia, and Gaussian Noise Injection as examples of color-space augmentation.

CLAHE

Contrast Limited Adaptive Histogram Equalization (CLAHE) is an image enhancement technique that improves the contrast and brightness of an image. This technique is widely used in image processing and computer vision tasks to improve the visual quality of images and make it easier to detect objects and features, example of CLAHE is shown in Figure 3.6.

CLAHE works by dividing the image into small rectangular regions called tiles and then equalizing the histogram of each tile separately. This approach ensures that the brightness and contrast of each tile are optimized independently, which can be beneficial in cases where the brightness or contrast varies significantly across the image.

One advantage of CLAHE is that it is an adaptive technique, meaning that it can adapt to the local characteristics of the image. This allows it to provide better results than traditional histogram equalization, which applies the same transformation to the entire image. CLAHE also allows for better visualization of fine details and edges in images, which can be especially useful in medical imaging applications or aerial images of marine wildlife where subtle changes in texture or contrast can be important for detecting features or anomalies.

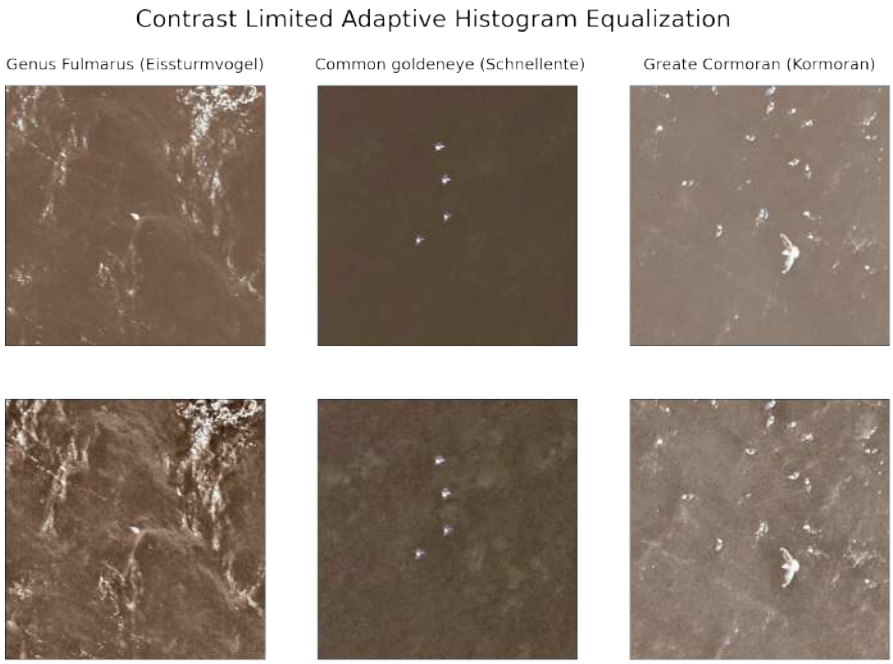


Figure 3.6: Examples of CLAHE with contrast limit between 1 and 4 and title grid with size of 8

Sharpen

Sharpen augmentation is a technique used in image processing and computer vision that enhances the edges and fine details in an image. The technique involves applying a filter to the image that increases the contrast between adjacent pixels, making the edges more distinct and prominent.

Sharpen augmentation can be particularly advantageous in the context of aerial images for a number of reasons. Aerial images of the marine environment, for instance, often contain low-contrast regions such as large bodies of water or sky that can reduce the sharpness and clarity of the image. Sharpen augmentation can help to increase the contrast between adjacent pixels, making edges and fine details more visible and distinct. This can improve the visual quality of the image and make it easier to detect and recognize objects, including marine wildlife, in the image.

The alpha parameter in sharpen augmentation is a factor that controls the strength of the filter. A higher value of alpha will result in a stronger sharpening effect, while a lower value will result in a weaker effect. The lightness parameter in sharpen augmentation is a factor that controls the amount of lightness or brightness in the image. A higher value of lightness will increase the brightness of the image, while a lower value will decrease it. Examples of augmented images are represented in Figure 3.7

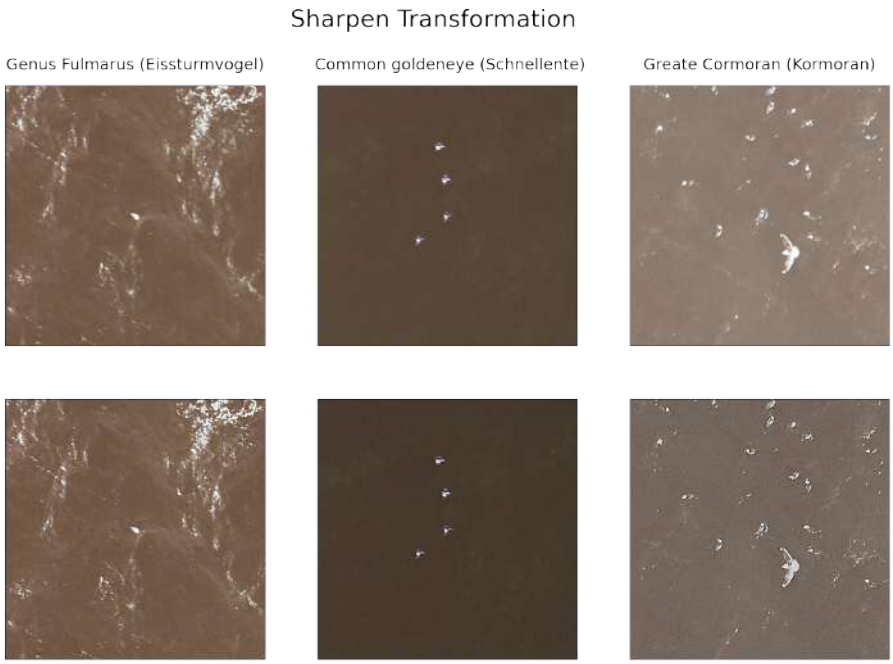


Figure 3.7: Examples of Sharpen with alpha value between 0.2 and 0.5, lightness value between 0.5 and 1.0

Channel Shuffling

Channel Shuffling augmentation is an image transformation technique that can be used to generate a diverse set of training images for machine learning models. In channel shuffles, the color channels of an image are randomly permuted, which can introduce a significant amount of variation to the image data.

The advantage of Channel shuffling is its simplicity in both understanding and implementation. However, the variation of the transformation is limited to the permutation of the channels, which is only six. This limitation restricts the ability to enhance the robustness of the data. Moreover, one color channel may be more important than others. For example, if blue is an important feature to recognize the ocean, shuffling the channels could pose an obstacle for the model. Figure 3.8 show the effect of Channel Shuffling on the image.

Color Jitter

Color Jitter is an image augmentation technique that introduces random color variation to an image by perturbing the color channels of the image. The goal of this technique is to increase the diversity of the training data-set for machine learning models, which can help improve the model's ability to generalize to new, unseen data.

Color Jitter works by randomly perturbing the hue, saturation, brightness, and contrast of an image. These perturbations are generated by adding small amounts of noise or by scaling the color values in channels. By applying these perturbations,

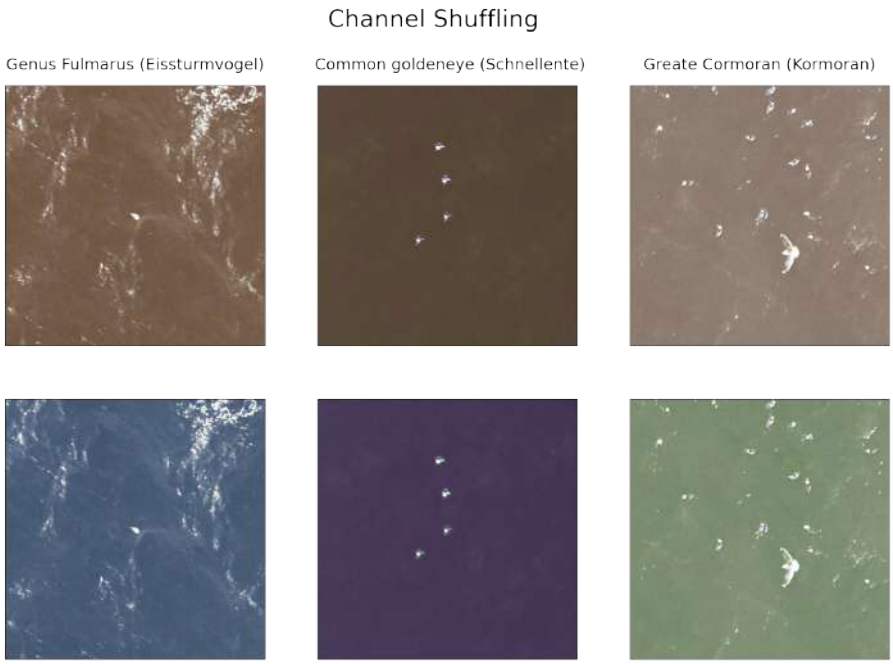


Figure 3.8: Examples of Channel Shuffling

Color Jitter creates new images that are visually distinct from the original image, which can help improve the model's ability to detect and recognize objects with varying color properties.

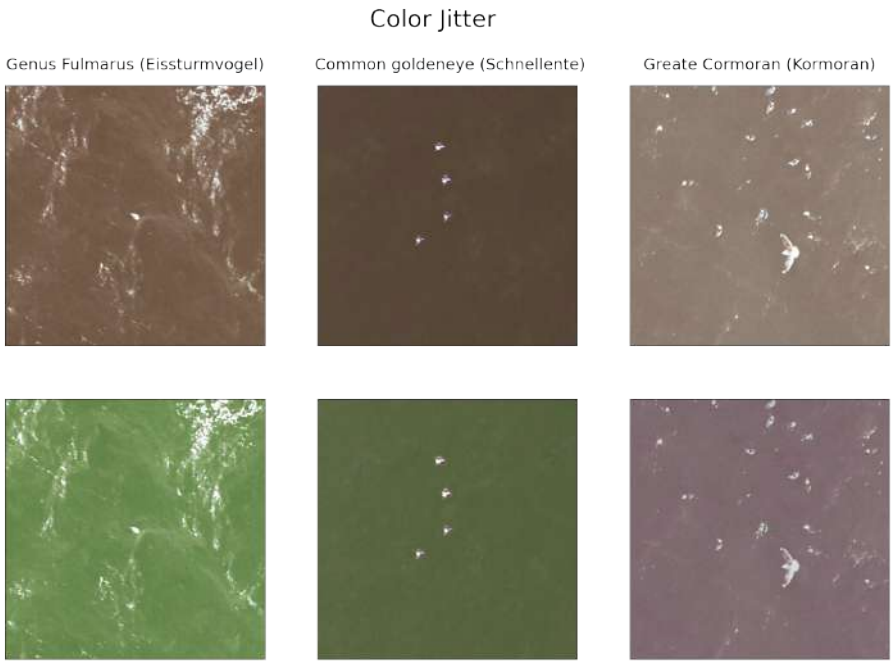


Figure 3.9: Examples of Color Jitter with the random changes in brightness, contrast, saturation, hue between 0 and 0.2

Similar but more sophisicated than A.1ChannelShuffles]Channel Shuffling, Color Jitter can be advantageous for aerial images because it introduces random color variation to the images. For example, when detecting marine wildlife in aerial images, variations in the lighting and coloring of the water and surrounding environment can make it challenging to detect and differentiate the animals. By using Color Jitter to generate a diverse set of training images with random color perturbations, the machine learning model can learn to detect marine wildlife in a wider range of visual conditions. Additionally, standardizing the color distribution across images in the data-set can help to reduce the impact of color biases in the data-set, which can lead to improved accuracy and performance of the machine learning model, especially in classes with small samples. Figure 3.9 illustrates the effect of the transformations.

Solarize

Solarize can be advantageous for aerial images, as it introduces a striking visual effect by inverting the colors of an image that exceed a certain threshold.

In the context of aerial images of marine wildlife, solarize can be useful to create a diverse set of training images for machine learning models. By inverting the colors of the images, solarize can create visually distinct images that can improve the model's ability to detect and recognize objects with varying color properties. Similar to Color Jitter 3.1.2, Solarize helps to reduce the impact of color biases in the dataset. For example, if the majority of images of a particular marine species are predominantly of a particular color, the machine learning model may learn to associate that color with that class, leading to reduced accuracy and performance when detecting that species in images with different colors. By using solarize to invert the colors of these images, the model can be trained to recognize the species regardless of the color of the image.

ToGray

One advantage of converting images into grayscale is that it can help to reduce the impact of color biases in the dataset. Color biases can occur when images of a particular class are predominantly of a particular color. By converting the images into grayscale, the model can be trained to recognize objects of a particular class based on their shapes and textures, rather than their color. However, the obvious disadvantage of converting images into grayscale is that it can result in the loss of important color information. Color Patterns may still be important for distinguishing different types of marine wildlife, and for identifying specific features of the environment that are important for object recognition. Figure 3.11 illustrates our examples in grayscale.

ToSepia

Sepia is a type of image colorization technique that adds a brownish or yellowish tint to an image, giving it a vintage or aged look. It is often used in artistic or

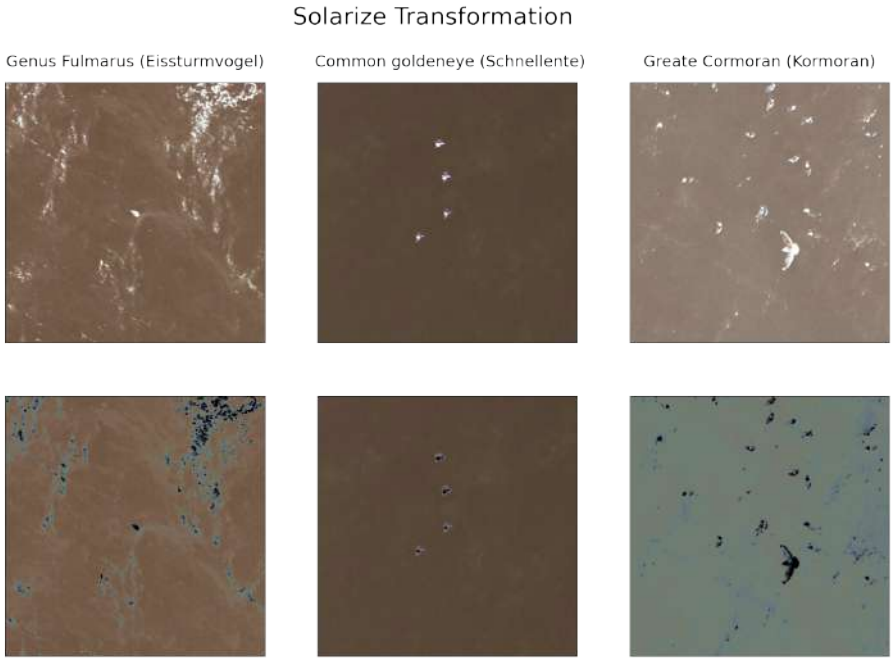


Figure 3.10: Examples of Solarize with threshold of 128

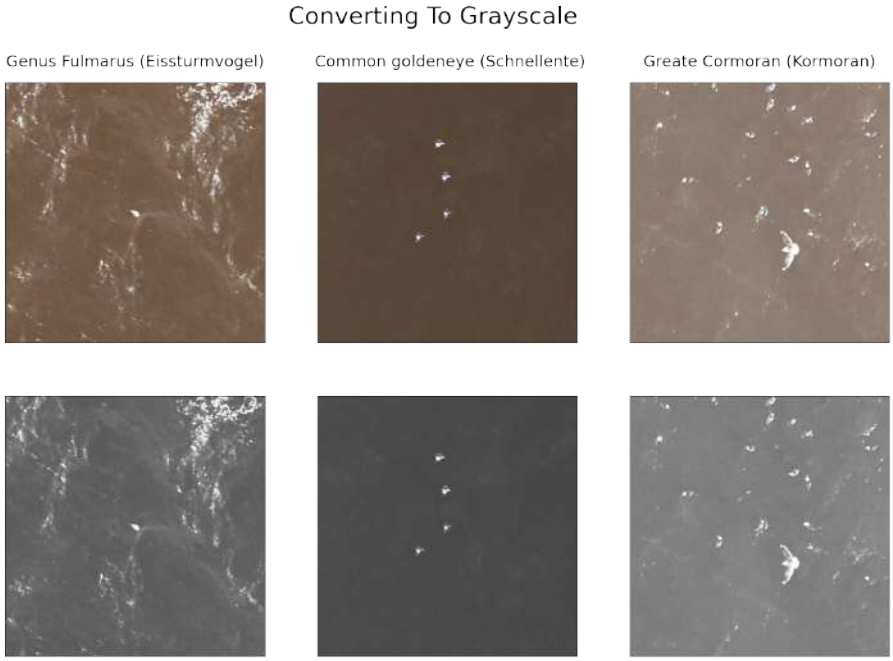


Figure 3.11: Examples of ToGray

design applications to give images a nostalgic or classic feel.

While converting an image to grayscale 3.1.2 involves discarding color information and representing the image in shades of gray, applying a sepia tint preserves the original color information and modifies it to create a vintage or aged effect. This makes the sepia technique a good option for applications such as edge detec-

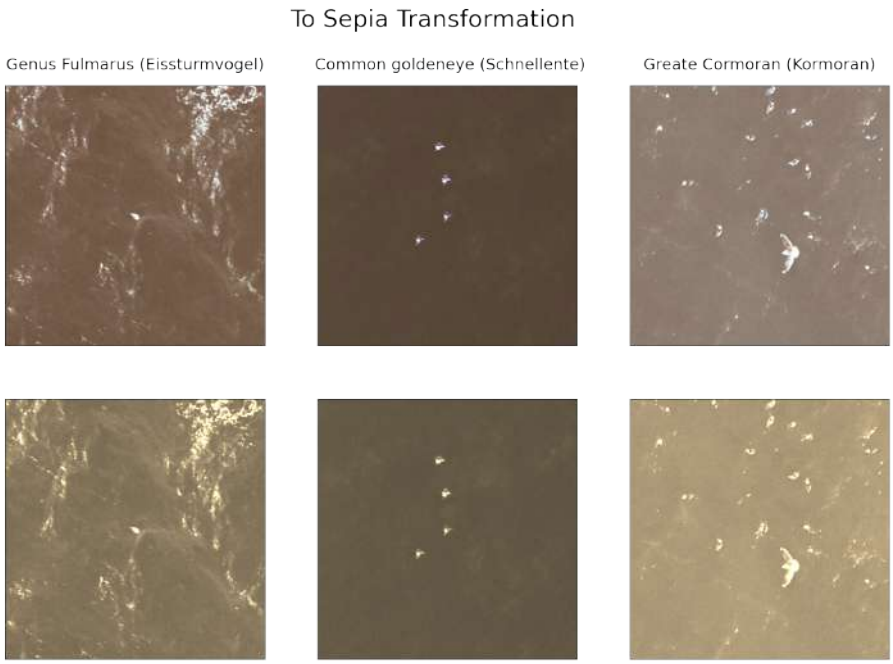


Figure 3.12: Examples of ToSepia

tion, where the shape and structure of objects in the image are more important than their color, without the risk of significant loss of color information.

Gaussian Noise

Gaussian noise injection is an image augmentation technique that adds random noise to the image by sampling from a normal distribution. In this technique, a random value is added to each pixel of the image, and the value of the added noise is determined by the standard deviation of the normal distribution, as illustrated in Figure 3.13.

The advantage of Gaussian noise injection is that it can create a diverse set of training images with varying levels of noise. This can help the machine learning model to better recognize objects in images with varying levels of noise, which can occur due to factors such as lighting, camera settings, or weather conditions. By training the model on a diverse set of noisy images, it can learn to better generalize to new, unseen data with varying levels of noise.

However, in the context of aerial images of marine wildlife, Gaussian noise injection may not be the most effective image augmentation technique. Aerial images of marine environments are often captured from a high altitude and are subject to atmospheric distortions, which can result in noise that is different from the type of noise added by Gaussian noise injection. Gaussian noise injection can negatively impact the visual quality of the image, which can make it more challenging to detect and recognize objects in the image.

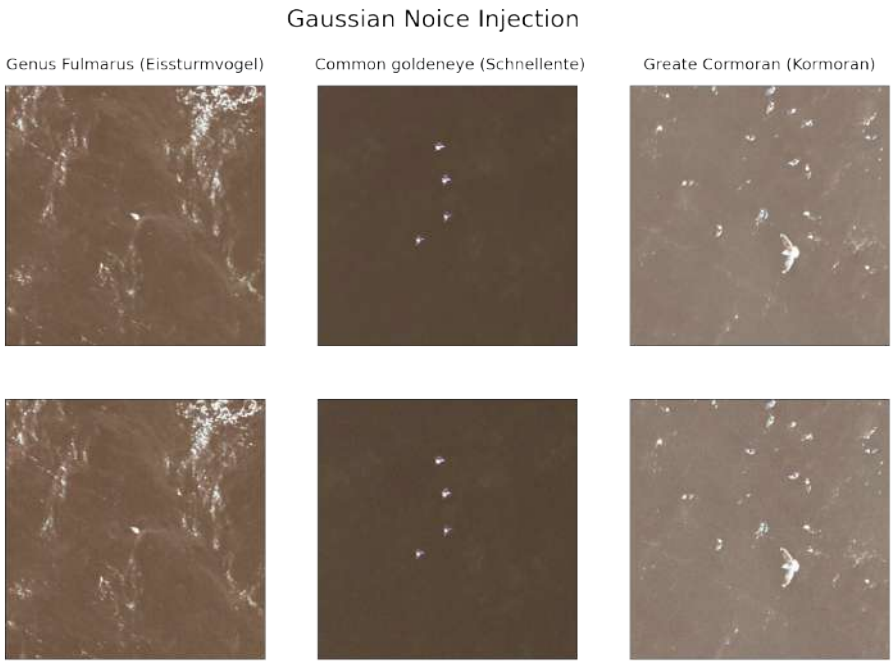


Figure 3.13: Examples of Gaussian Noise Injection with variance range for noise between 10 and 50

3.1.3 Cutout, GridDropout, Random Erasing

In this section, we will introduce three augmentation techniques: Cutout, GridDropout, and Random Erasing. These techniques are frequently used to enhance the model's ability to concentrate on the important information in the image and increase its resistance to occlusion or missing data.

Cutout/Coarse Dropout

Cutout is an image augmentation technique that involves randomly selecting a rectangular region of an image and setting the pixel values in that region to zero or a constant value. This creates a "hole" in the image, which can be used to encourage the model to focus on the remaining information in the image and improve its robustness to occlusion or missing data. The size, shape, and number of cutout regions can be varied to create a diverse set of training images and prevent overfitting.

Cutout augmentation can potentially be non label-preserving in aerial images because it involves removing parts of the image, which can also remove relevant objects or features that the model needs to learn, e.g a bird in the center of the image. This can cause inconsistencies between the original labels of the image and the labels assigned to the modified image, especially if the removed region contains the object of interest. For examples in Figure3.14, one common goldeneye got cut out, if there were just one bird in the picture, it would be considered non-label preserving.

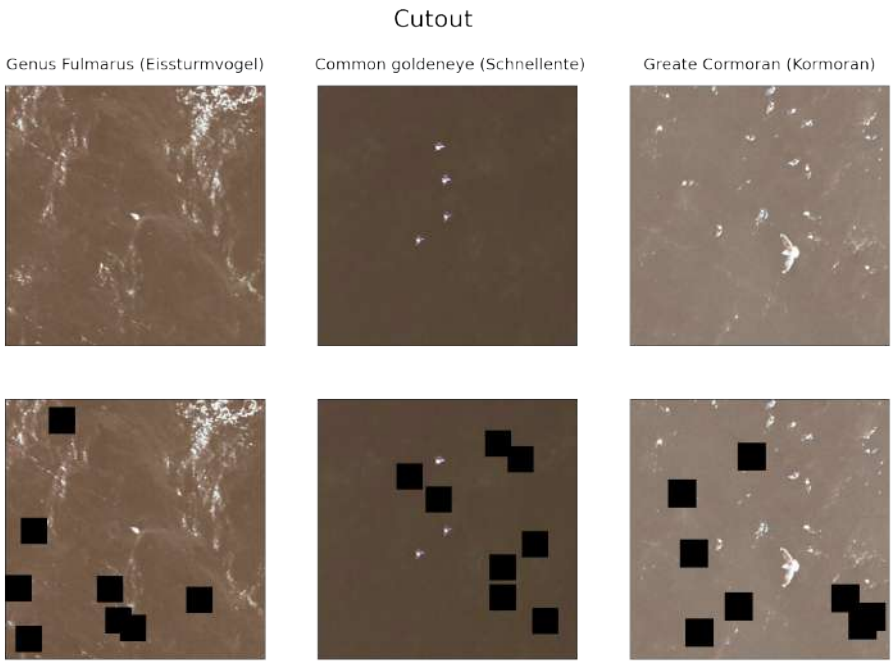


Figure 3.14: Examples of Cutout with 8 boxes, each size 1/10 height of the image

GridDropout

GridDropout is an image augmentation technique that is based on the Cutout method 3.1.3 by the same author [9]. It involves dividing the image into a grid of equal-sized cells and randomly dropping out entire cells instead of rectangular patches. This creates a checkerboard pattern of dropped cells that encourages the model to learn features that are invariant to the exact location of objects in the image. The advantage of using GridDropout over Cutout is that it allows for larger and more complex holes to be created while also being less likely to remove entire objects from the image. In the case of our aerial image dataset, the objects of interest are often small in size, and therefore applying cutout augmentation may not be label-preserving due to the risk of removing important object features, as illustrated in Figure3.15.

3.1.4 Mixed technique

This section will review shortly about the mixed approaches in image augmentation. Custom mixed augmentation and RandAugment will be introduced.

RandAugment

RandAugment is a random image augmentation technique that applies a sequence of randomly selected image transformations to the training images [12]. The specific transformations and the sequence of their application are randomly generated for each image using a set of hyperparameters, resulting in a diverse set of training

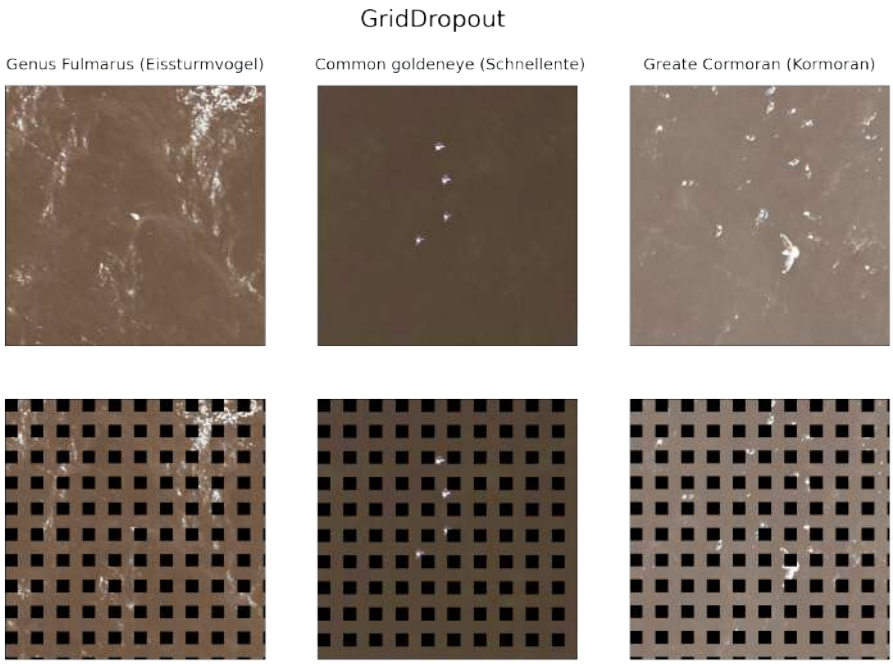


Figure 3.15: Examples of GridDropout with 100 boxes

images with a wide range of visual variations. This can improve the performance of machine learning models by helping them to better generalize to new, unseen data, but careful tuning of the hyperparameters is necessary to avoid overfitting or underfitting the model. RandAugment is seen as a strong and useful way to increase data for machine learning, and it is still being used today.

Algorithm 1 RandAugment

procedure RANDAUGMENT($image, n, m$)

$trans_list \leftarrow$ a list of possible image transformations

$trans_seq \leftarrow$ a random sequence of n transformations from $trans_list$

for $transform \in trans_seq$ **do**

$mag \leftarrow$ a random number between 0 and m

Apply the transformation $transform$ to the $image$ with magnitude mag

end for

return the augmented image

end procedure

This pseudo code 1 of RandAugment takes an input image, the number of transformations to apply (n), and the magnitude of the transformations (m) as parameters. The procedure first defines a list of possible image transformations and then generates a random sequence of n transformations from this list. For

each transformation in the sequence, the procedure applies the transformation to the input image with a randomly selected magnitude between 0 and m. Finally, the procedure returns the augmented image.

These specifics will depend on the application and dataset and may need to be adjusted accordingly. In our experiment, RandAugment takes transformation of Rotation, Flipping, Motion Blur, Sharpen, Center Crop 0.4, Gaussian Noise Injection, Channel Shuffling, Cutout, and Color Jittering. The samples of augmented images are illustrated in figure 3.16

3.2 Hyperparameter tuning

We have learned about various augmentation techniques in chapter 3.1. However, the possibilities of image manipulation are endless, which means that there are an unlimited number of augmentation methods to consider. And each augmentation technique may have hyperparameters that control the magnitude of the transformation, leading to even more possibilities for image manipulation. To tackle the challenge of selecting the appropriate augmentation techniques and hyperparameter values for a specific dataset, it is critical to adopt a systematic approach. This involves having prior knowledge of the dataset and a solid understanding of the augmentation techniques.

In this section, we will discuss the importance of choosing augmentation techniques as well as hyperparameter tuning in augmentation. Then, we will present a systematic approach for tuning these hyperparameters to optimize the image transformation pipeline for the specific dataset and model architecture. In chapter 5, we will represent the optimal hyperparameter values in a table format for each augmentation technique.

3.2.1 Choosing the right augmentation techniques

Unfortunately, there is no shortcut for selecting the appropriate image augmentation techniques as this process heavily depends on the characteristics of the dataset used to train the model. The optimal set of augmentations can vary significantly based on the distribution, size, and complexity of the dataset. While a similar policy may be referenced if the datasets are similar in size and object, in our case, the image size is relatively large (256x256 pixels) compared to other popular aerial image datasets, such as BigEarthNet (120x120 pixels) [38] or Airbus Wind Turbines Datasets (128x128 pixels) [1]. Additionally, the object of interest is small compared to the background, necessitating an empirical exploration of new augmentation policies.

To demonstrate the significance of exploring augmentation policies for different datasets, let us consider the example of CenterCrop. CenterCrop transformation can be meaningful in datasets with center biases and small objects. However, applying it to other datasets can remove important details and make training more challenging. The selection of appropriate image augmentation techniques should

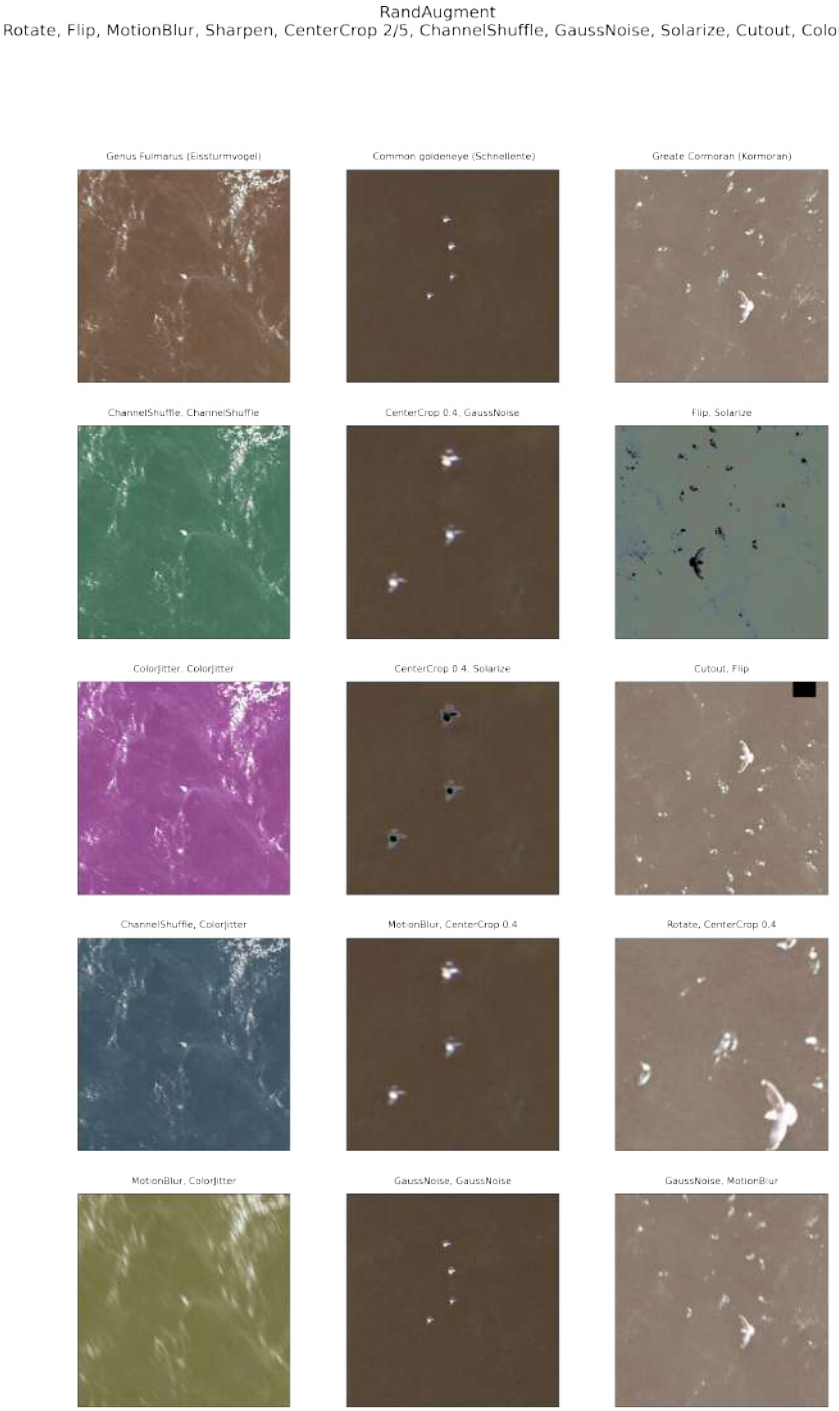


Figure 3.16: Examples of RandAugment

consider the characteristics of the dataset and the specific use case to improve model performance. Figure 3.17 demonstrates the impact of applying the CenterCrop transformation inappropriately in Airbus Wind Turbines Windmill Dataset [1].

Therefore, it is important to carefully consider the characteristics of the dataset

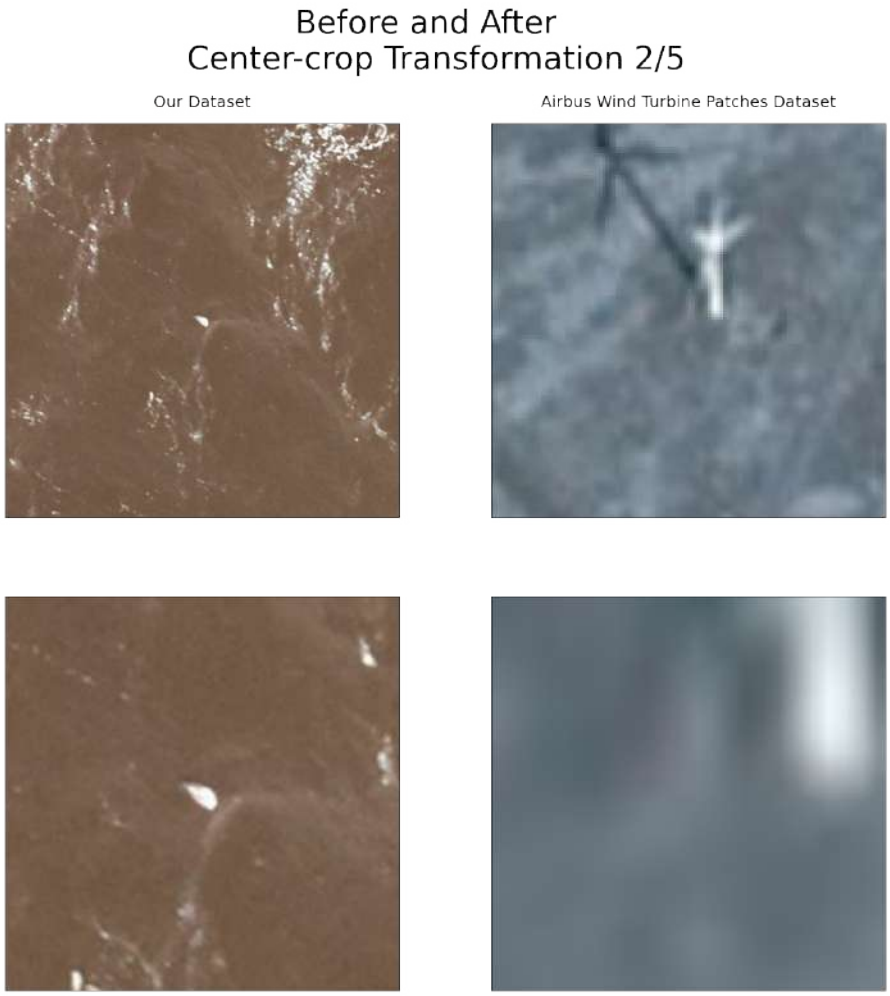


Figure 3.17: Illustration of the CenterCrop Transformation 2/5 in our data and one sample from Airbus Windturbines Dataset [1]. The image on the bottom right illustrates the windmill being barely recognizable after the transformation, highlighting the importance of appropriate selection of image augmentation techniques for the specific use case.

and the specific learning task when choosing the right augmentation technique. This can involve exploring a wide range of augmentations methods and evaluating the performance of the model on a validation set to determine the optimal combination of hyperparameters.

However, due to computational resource limitations, exploring all possible augmentation techniques is not feasible. If a transformation is expected to not enhance robustness to known datasets, the technique will have lower priority to be explored. In this context, a good example for our datasets is RandomSunFlare, as illustrated in Figure 3.18. RandomSunFlare is an image augmentation technique that adds a bright patch or flare to the image, simulating the effect of the sun or other light source shining on the camera lens. However in our case, as the camera is often mounted on a drone or other airborne device and positioned high above the

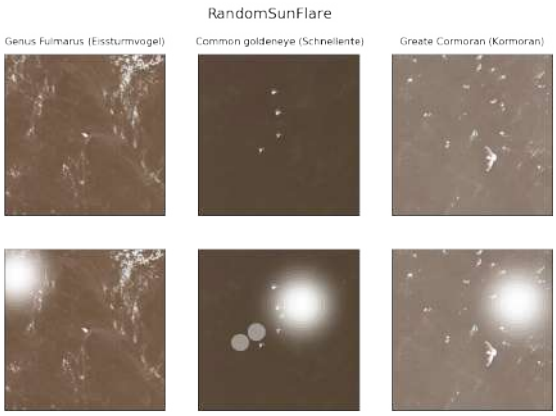


Figure 3.18: Illustration of the RandomSunflare Transformation. While we cannot be certain about its effects, we do not find the simulation of sunlight meaningful for our aerial image dataset. Therefore, this method is not a promising option and has a lower priority for exploration.

ground. Hence, the angle and position of the sun or light source relative to the camera are likely to be different from those in ground-level photography, making the addition of a sun flare unrealistic and potentially misleading. As a result, other augmentation techniques, such as rotation, translation, or color jitter, may be more promising for increasing the diversity and variability of the dataset.

3.2.2 Choosing the right hyperparmeters

After considering the appropriate augmentation techniques, we also have to pick the right parameter for the augmentation. In chapter 3.1, we learned about the various image augmentation techniques used in this work and how they have different sets of parameters that control different aspects of the transformation, such as magnitude, change limit, filter size, threshold, and more. Although every augmentation technique incorporates a certain degree of randomness, the range of the randomness is not learned during training, but rather set manually by the user at the beginning. Therefore, the choice of hyperparameters can significantly impact the quality and diversity of the augmented images, which in turn can affect the performance of the model.

To conserve computational resources, it is important to mention that not all hyperparameter values are meaningful or relevant to explore. For example, in the Solarize method, pixels exceeding a particular threshold are inverted using the formular:

$$\text{max value} - \text{pixel value} \quad \text{if} \quad \text{pixel value} \geq \text{threshold} \tag{3.1}$$

Exploring extremely small values, such as zero, is not informative as the resulting image may resemble an inverted picture, which does not provide the model with any new features, because this can be figured by the model itself by adapting the convolutional filter kernels [24].

On the other hand, it is also non-optimal to explore extreme hyperparameter values that are highly likely to introduce non-label-preserving transformations. For instance, when exploring contrast in the CLAHE method, exploring values of clipping limit greater than 100, may often result in too noisy images as in Figure 3.19, hence reduce the model performance. Therefore, it is essential to carefully select the hyperparameter values to explore to optimize the performance of the model while minimizing the computational cost.

In hyperparameter tuning algorithms, it is crucial to define a search space beforehand, which sets bounds for all hyperparameters and incorporates prior knowledge of them. However, using a uniform distribution to explore values within a range could be inefficient since it assumes that all values in the range are equally important. To solve this problem, a log-uniform distribution is used, which explores values uniformly across several orders of magnitude. For instance, when searching for the best value for a hyperparameter such as learning rate, exploring a range from 0.1 to 1e-5 using a uniform distribution may result in spending more time exploring many more values in the range of 0.1 and 0.2 than in the range of 1e-4 and 2e-4, because the interval between 0.1 and 0.2 is 1000 times bigger. This can be inefficient especially for hyperparameters like the learning rate that have a logarithmic relationship with the model's performance metrics. This approach also applies to augmentation hyperparameter search, for example, when tuning the clipping limit in CLAHE. Rather than exploring values from 4 to 8, which would require seven runs, exploring values such as 4, 8, 16, and 32 helps to pinpoint the optimal range.

3.2.3 Methods for tuning hyperparameters

In order to select the optimal hyperparameters for a particular image augmentation pipeline, various search methods can be used. There are many advanced techniques for hyperparamter tuning which is different in complexity and computational resource requirement. Two of the most common search methods are Grid Search and Random Search, which are both simple to implement and computationally feasible. In this section, we introduce and compare these search methods and discuss their advantages and disadvantages in hyperparameter tuning for image augmentation.

Grid Search

The ideas of grid search is fairly simple. We construct a grid of potential discrete hyperparameter values and train the model with every possible combination. We record the performance of the model for each set of hyperparameters and select the combination that yields the best performance. This can be accomplished using nested for-loops through predefined values. To illustrate this concept, we utilize an example with the CLAHE Transformation 3.1.2. As shown in Figure 3.20, we explore the clipping limit and filter size values each of 2, 4, 8, 16, and 32.

Grid Search is an exhausive algorithm that explores all possible combinations, allowing it to find the optimal point within the search space. However, the downside

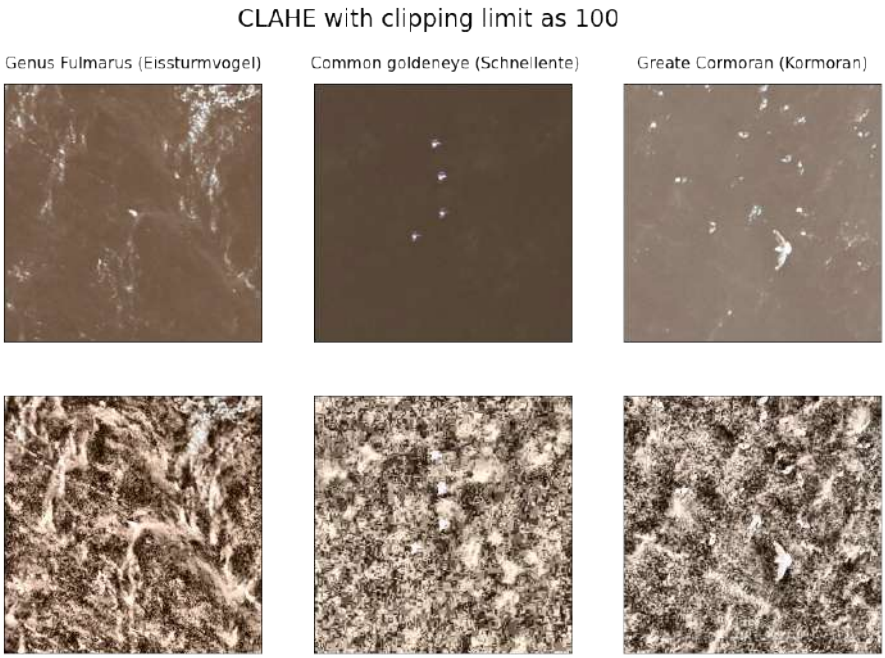


Figure 3.19: Illustration of the CLAHE Transformation with Clipping limit as 100. This demonstrates that when applying extreme hyperparameters, the resulting augmented images can be overly noisy and the objects in the images may become unrecognizable.

is that it can be slow. Checking every combination within the space requires a significant amount of time, which may not always be available or sometimes even impossible.

Random Search

Random Search is an alternative to Grid Search that randomly samples the search space instead of discretizing it into a Cartesian grid. Unlike Grid Search, Random Search does not have a fixed endpoint but requires a predetermined time budget that determines the number of trials. Random Search has been found to be particularly effective, especially when some hyperparameters have a much larger range of variation than others, which can result in a non-cubic search space [6]. As an example, we can consider the hyperparameter tuning of Color Jittering, which consists of four parameters: Brightness, Hue, Contrast, and Saturation. Each parameter has a range between 0.2 and 0.8.. Table 3.1 shows the results of 4 iterations.

Random search is a valuable option when dealing with several hyperparameters with a continuous values. By selecting a subset consisting of 5-50 randomly chosen points, depending on the computational resources available, we can obtain a set of hyperparameter values that are reasonably good. In our experiment, we only use 5-10 points. While this set may not necessarily be the best point, it can still provide a good model.

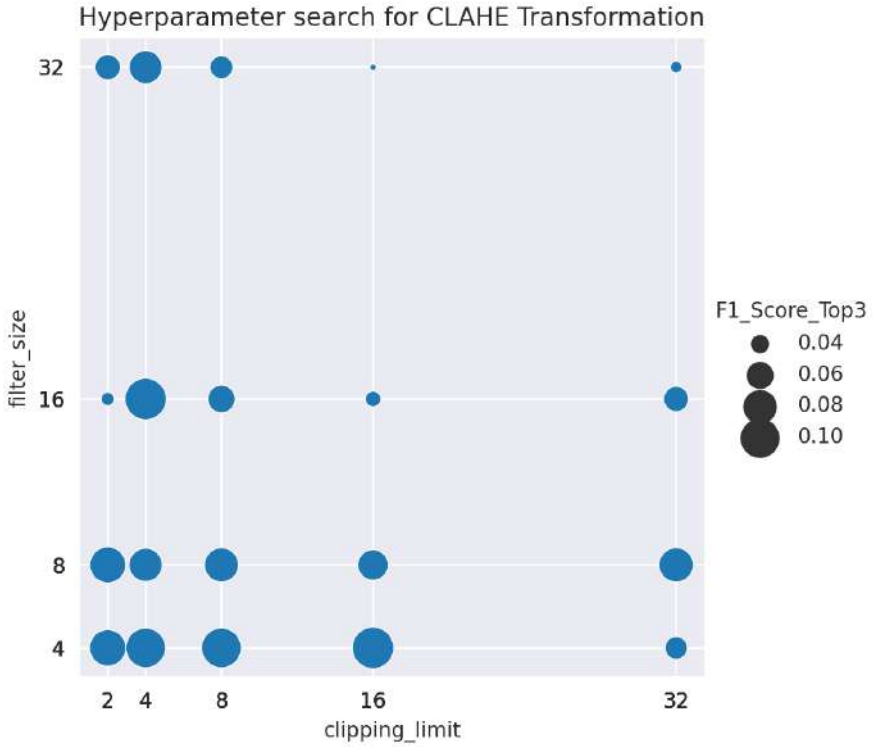


Figure 3.20: Ilustration of Grid Search in CLAHE. The size of each dot represents the F1 score for each variable combination. As illustrated, there are several obvious minima and some combinations that are relatively comparable in efficiency. The combination with the best F1 score, which is 0.119422, is selected. This combination consists of a clipping limit of 16 and a filter size of 4.

Other tuning techniques

In addition to grid search and random search, there are other hyperparameter tuning techniques such as Bayesian optimization, genetic algorithms, and gradient-based optimization.

- Bayesian optimization [16]: This advanced hyperparameter tuning approach uses Bayesian inference to construct a probabilistic model of the objective function. The objective is to create a model that accurately approximates the performance of the model for any given set of hyperparameters. The Bayesian optimization then selects the next set of hyperparameters to evaluate using this model, aiming to maximize the model's performance while minimizing the number of evaluations needed. The main advantage of Bayesian optimization is that it can quickly identify promising regions of the hyperparameter space, allowing for efficient exploration of a large number of potential hyperparameter configurations. However, it is computationally expensive and requires expertise in selecting the prior distribution and the acquisition function.

Table 3.1: Hyperparameter exploration with Random Search in Color Jittering. The maximum values of brightness, hue, contrast, and saturation are uniformly selected within the range of 0.2 and 0.8. After eight runs, the chosen maximum values are: 0.8 for brightness, 0.42 for hue, 0.54 for contrast, and 0.65 for saturation.

Index	Brightness	Hue	Contrast	Saturation	F1 Score
1	0.57	0.45	0.56	0.78	0.0125
2	0.74	0.73	0.6	0.4	0.0250
3	0.41	0.28	0.55	0.47	0.0388
4	0.55	0.2	0.23	0.64	0.0418
5	0.67	0.26	0.65	0.27	0.0507
6	0.77	0.77	0.24	0.36	0.0518
7	0.48	0.55	0.34	0.64	0.0519
8	0.8	0.42	0.54	0.65	**0.0585**

- Genetic algorithms [30] [4]: This technique is inspired by natural selection and uses a population of candidate solutions that evolve over time through crossover and mutation to find the optimal set of hyperparameters. The main advantage of genetic algorithms is their ability to search a large space of potential hyperparameters quickly and efficiently. But this approach is very computationally expensive, and it tends to find multiple local optima rather than one global optimum.

- Gradient-based optimization [5]: This approach involves computing the gradients of the objective function with respect to the hyperparameters and then using these gradients to iteratively update the hyperparameters until the optimal values are found. Gradient-based optimization algorithms require the objective function to be differentiable with respect to the hyperparameters, which is sometimes infeasible.

While these advanced techniques are known to be more optimal in finding the optimal combination of hyperparameters, they tend to be computationally expensive and require a high level of expertise to design. As a result, in this thesis, we will focus solely on grid search and random search as they are simpler and more straightforward to implement.

Chapter 4

Approach

This chapter presents our experimental approach, which includes the evaluation metrics and models for both classification and detection tasks. We will also discuss the specific GAN-generated architectures used and provide an overview of the overall pipeline for the experiments.

4.1 Metrics

In this session, we will discuss the evaluation metrics used in this thesis and their significance. The evaluation metrics are categorized into classification metrics and detection metrics, which are specific to their respective tasks. The classification metrics include accuracy, recall, precision, and F1-score, while the detection metrics include Intersection over Union (IoU), mean average precision (mAP), precision, class loss, object loss, and classification loss. We will provide a brief explanation and overview of each metric.

4.1.1 Classfication Metrics

In classification tasks, evaluation metrics are used to measure the accuracy of a model in predicting the correct class label for an input instance. Some of the most important metrics in classification tasks used in this thesis are accuracy, precision, recall, F1-score and their top-N metrics:

Accuracy, precision, recall and F1-score

- Accuracy is a commonly used metric that measures the overall correctness of the predictions made by a model, expressed as the percentage of correctly classified samples over the total number of samples.

- Precision, also known as positive predictive value, measures the proportion of true positives that are correctly identified by the model over the total number of positive predictions. This metric is important in cases where false positives

can have significant consequences, such as in the case of misidentifying an important class.

- Recall, also known as sensitivity or true positive rate, measures the proportion of true positives that are correctly identified by the model over the total number of actual positives. This metric is particularly important in cases where false negatives can have severe consequences, such as in the detection of rare species.

- The F1 score is a weighted average of precision and recall, and provides a more balanced measure of the model's performance. It is calculated as the harmonic mean of precision and recall. It is calculated as:

$$\frac{2 \cdot (precision \cdot recall)}{(precision + recall)} \tag{4.1}$$

While precision and recall are both important metrics for different reasons - precision measures the model's ability to correctly identify true positives, while recall measures the model's ability to identify all positive instances - F1 score is often considered the most relevant metric for evaluating a machine learning model. By providing a balanced measure of the model's precision and recall, F1-score offers a more comprehensive evaluation of a model's performance. This is especially useful for imbalanced datasets, such as ours, where precision and recall alone may not accurately represent the model's performance. Therefore, in our thesis, we will use F1-score, beside accuracy, as the primary metric to compare significant outcomes.

Top N Metrics

When evaluating a classification model with a high number of classes, such as in our case with more than 100 classes, top-N metrics such as top-N accuracy or top-N F1 score can be useful. Top-N metrics are a variation of the traditional metrics that consider a prediction to be correct if the true class label is among the N highest-ranked predicted class labels. This is particularly useful when a model can provide a ranked list of predictions, as it allows for partial credit to be given for correct predictions that may not be the top-ranked prediction. The specific value of N can be chosen based on the problem domain and use case, typically a small value such as 1 or 5. In our case, we will use top-1 and top-3 metrics.

To illustrate the concepts of top-N metrics, let's use a sample of class "herring gull" or "Silbermöwe", as illustrated in Figure 4.1a). For this given test example, EfficientNet produces a ranked list of predicted labels, along with their associated probabilities as follows:

- Herring Gull (Silbermöwe): 60%
- Common Gull (Sturmmöwe): 20%
- Buoy or mooring ring (Boje oder Bojenkette): 10%

(a) Herring Gull (b) Common Gull

Figure 4.1: Illustration of (a) Herring Gull (Silbermöwe) and (b) Common Gull (Sturmmöwe) to illustrate similar-looking classes

- Mallard Duck (Stockente): 10%

If the prediction in this example was Common Gull, it would be considered a false prediction in top-1 accuracy, but true in top-3 accuracy. This example also highlights the importance of using top-N metrics, as our dataset contains over 100 classes, and some bird classes, such as Herring Gull and Common Gull in Figure 4.1, have very similar appearances. Thus, we can still give credit to the models when their predictions are close to the actual labels.

4.1.2 Detection Metrics

In detection tasks, it is crucial to evaluate the accuracy of a model in both localizing and classifying objects in an image. Therefore, besides the metrics used in classification tasks such as precision or recall, there are metrics specifically designed to measure the accuracy of predicted localization. This section will introduce some of the important statistics, including Intersection over Union (IoU), Mean Average Precision (mAP), and various loss metrics such as class loss, object loss, and classification loss used during training.

Intersection over Union (IoU)

IoU is a measure of the overlap between predicted bounding boxes and ground-truth bounding boxes. IoU is calculated as the ratio of the area of intersection between the predicted and ground-truth bounding boxes to the area of their union. It ranges from 0 to 1, where 1 indicates perfect overlap and 0 indicates no overlap. IoU is commonly used as a threshold for determining true positive or false positive detections, with 0.5 being a common threshold. Figure 4.2 illustrates the intersection over union calculations.

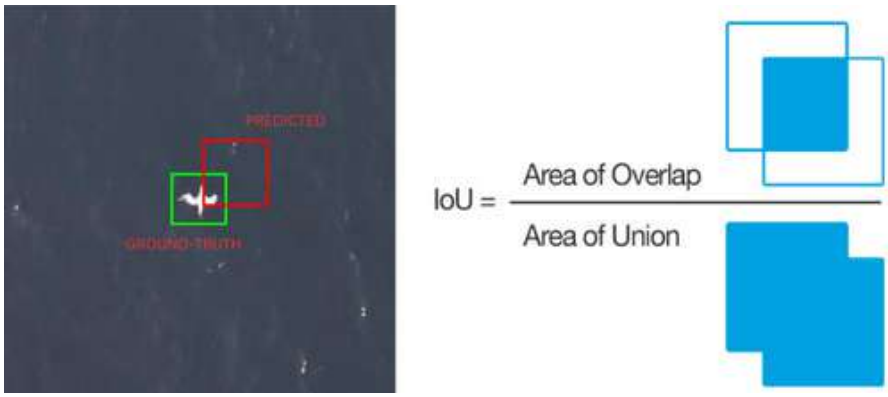


Figure 4.2: Illustration of the Intersection over Union calculation [33]

Mean Average Precision (mAP)

The average precision (AP) is a measure of the accuracy of an object detection model in detecting objects of a particular class. AP is calculated by computing the precision and recall for a range of IoU thresholds, and then calculating the area under the precision-recall curve (PR curve).

The following steps describe how to calculate the AP at a certain IoU:

- For each object class, calculate the precision and recall.
- Plot the precision-recall curve by plotting the precision values against the corresponding recall values at each IoU threshold.
- Calculate the area under the precision-recall curve. The resulting value is the AP for that object class.

AP is a useful metric for measuring the accuracy of an object detection model for a specific class, but its calculation is dependent on the IoU threshold, which can be arbitrary and varies for different tasks. To address this ambiguity, mean average precision (mAP) was introduced as a comprehensive metric that evaluates the model's accuracy across multiple object classes and IoU thresholds.

Calculating mAP is a simple process that builds upon the calculation of AP. Firstly, a set of thresholds is considered for AP calculation. In this thesis, we use thresholds ranging from 0.5 to 0.95 with a step size of 0.05 (mAP@[0.5:0.95]). AP is then calculated across the set of IoU thresholds for each object class, and the average of all AP values is taken to obtain the mAP. mAP@[0.5:0.95] is a widely used metric for evaluating object detection models because it provides a comprehensive evaluation of the model's accuracy across a reasonable range of IoU levels.

Loss Metrics

During training we use box loss, classification loss, and object loss.

- Box loss measures the discrepancy between predicted and ground-truth bounding box coordinates. It encourages the model to better localize objects in an image.
- Classification loss measures the discrepancy between predicted and ground-truth class probabilities. It encourages the model to correctly classify objects in an image.
- Object loss is a combination of box loss and classification loss, and is used to evaluate the overall accuracy of the model in localizing and classifying objects in an image.

4.2 Evaluation Models

In this section, we provide a brief overview of two popular deep learning models that we use to evaluate our augmentation methods: YOLOv5s for object detection and EfficientNet version B0 for image classification. The evaluation is based on F1-score for classification task and mAP@[0.5:0.95] for detection task. Understanding these models is essential to accurately interpret the results of our evaluation.

4.2.1 EfficientNet

There are numerous classification models available, such as YOLOv8 Classification, ResNet, EfficientNet, and Vision Transformer. Despite their good performance, as demonstrated in benchmarks like ImageNet [2], we have opted for EfficientNet due to its ease of modification and implementation using the Timm library [45].

EfficientNet [40] is a family of neural network models that achieve state-of-the-art performance on various computer vision tasks. It is a convolutional neural network architecture designed to achieve high accuracy while keeping the number of parameters low. This is achieved through a technique called "compound scaling" where the network's depth, width, and resolution are scaled in a balanced way, but independently based on a fixed scaling coefficient. This approach is different from traditional neural network scaling, where the dimensions are increased simultaneously by a fixed factor, which can lead to overfitting, computational inefficiency, and decreased accuracy.

This approach allows EfficientNet to achieve better performance than previous models, such as ResNet and Inception, while requiring fewer parameters. This makes EfficientNet an attractive choice for our classification task, as we are dealing with a large dataset with more than 100 classes and limited computational resources. Another advantage of using EfficientNet is that it has been pre-trained on a large-scale dataset such as ImageNet, which can help improve the performance of our model on our specific dataset. Figure 4.3 provides a more comprehensive understanding of the performance of EfficientNet between different models [40].

EfficientNet models come in several sizes, from EfficientNet B0 (the smallest) to EfficientNet B7 (the largest). The larger models have more layers and more param-

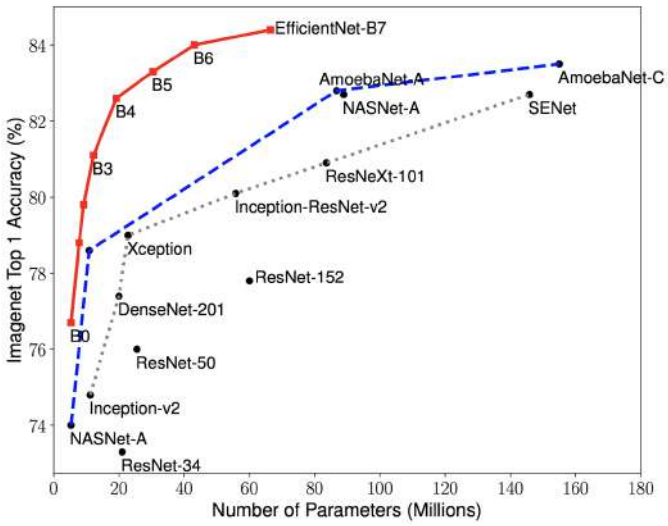


Figure 4.3: Model Size vs. Accuracy Comparison. EfficientNet-B0 is the baseline network developed by AutoML MNAS, while Efficient-B1 to B7 are obtained by scaling up the baseline network. In particular, EfficientNet-B7 achieves new state-of-the-art 84.4% top-1 / 97.1% top-5 accuracy, while being 8.4x smaller than the best existing CNN. [40]

eters, and are capable of achieving higher accuracy, but require more computational resources to train and run. This thesis uses pretrained EfficientNet with version B0. Although the performance may be affected by using version B0, it aligns with the goal of our thesis to compare various augmentations, making it acceptable. By using version B0, which has fewer parameters, we can train the model faster with different configurations, resulting in quicker analysis. Figure 4.4 shows the architecture of our baseline network EfficientNet-B0.

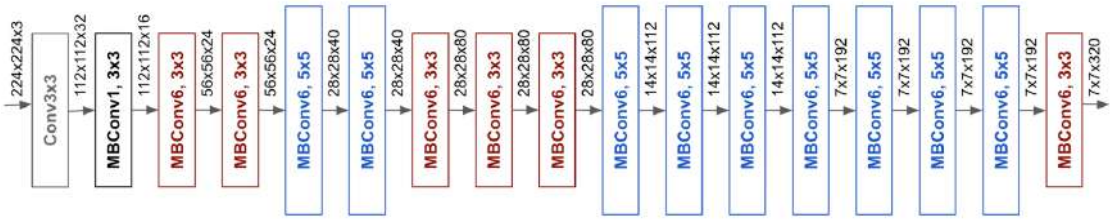


Figure 4.4: The architecture for our baseline network EfficientNet-B0 [41]

4.2.2 YOLOv5 (You Only Look Once version 5)

YOLOv5 (You Only Look Once version 5) [23] is a state-of-the-art object detection model developed by Ultralytics. It is inspired by the success of previous versions of YOLO [7] and introduces several improvements, including a new backbone architecture and a more efficient training process (mixed precision training [28] and gradient accumulation [20]).

Figure 4.5 shows the high-level architecture for an object detector like YOLOv5 which is comprised of three components: backbone, neck, and head [21].

- The model backbone: a pre-trained network used for feature extraction from the images, which reduces the spatial resolution while increasing the feature (channel) resolution.
- The model neck: responsible for extracting feature pyramids, which enables the model to effectively detect objects of varying sizes and scales, leading to better generalization performance.
- The model head: responsible for performing the final operations, which involve applying anchor boxes to feature maps and producing the final output. This output includes information such as object classes, objectness scores, and bounding box coordinates.

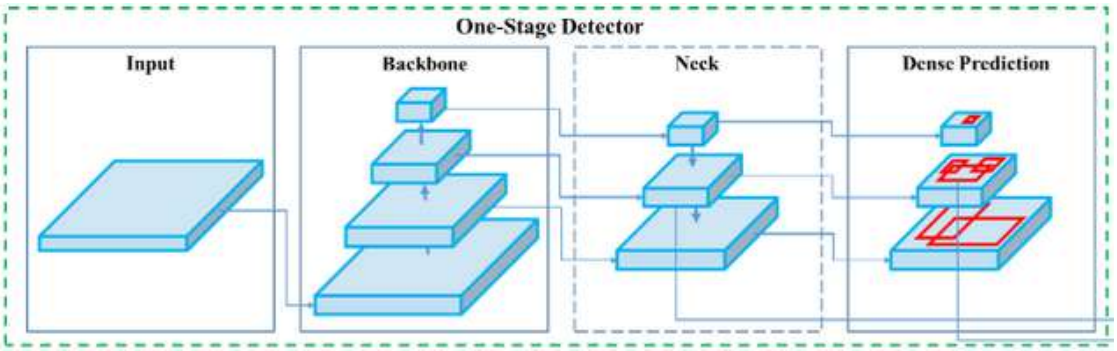


Figure 4.5: Single-Stage Detector Architecture[7]

The network architecture of YOLOv5 is shown in Figure 4.6, where the backbone architecture is based on CSPNet (Cross Stage Partial Network) [44]. CSPNet is a novel CNN design that divides a standard convolution layer into two parallel streams, reducing computation costs and enhancing the model's ability to capture complex features. The Neck of YOLOv5 incorporates Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet). SPP enables object recognition of different sizes without image resizing or cropping by dividing the image into a fixed number of regions and pooling features from each region. PANet utilizes a bottom-up pathway to extract features from the input image and a top-down pathway to generate a feature map for object detection, effectively aggregating multi-scale features to capture objects at different scales and resolutions. The head of YOLOv5 is composed from three convolution layers that predicts the location of the bounding boxes (x, y, height, width), the scores and the objects classes.

There are different sizes available in YOLOv5, ranging from YOLOv5n with 1.9 million trainable parameters to YOLOv5x6 with 140.7 million trainable parameters. As the model size increases, it becomes more capable of achieving higher mAP [23]. However, to ensure our computational resources are not exceeded, we will use YOLOv5s with 7.2 million trainable parameters.

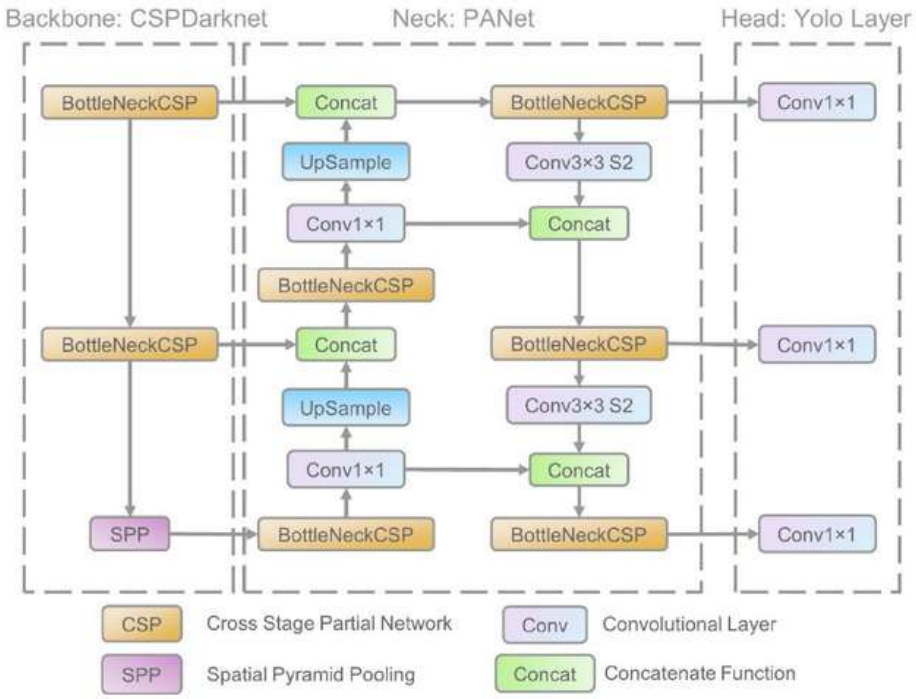


Figure 4.6: Network architecture for YOLOv5e [46]

4.3 Training Pipeline

This section presents a detailed description of the experimental set-up and training pipeline used in our thesis.

The experiments were performed on a computer with an AMD Ryzen Threadripper 2970WX 24-Core Processor and 188 Gb RAM. The deep learning models Yolov5s and GANs were trained on an NVIDIA GeForce RTX 3090 graphics card, while the EfficientNet model was trained on an NVIDIA GeForce RTX 2080 Ti. The software environment included Python 3.10.8, Pytorch, and the Albumentations library for data augmentation.

4.3.1 Pipeline for basic augmentations

Figure 4.7 illustrates the experiment pipeline for non-GAN approaches. For the basic augmentations, including geometric, color-space, cutout, and mixed augmentations, we apply online augmentation during training for the classification task. This involves directly applying the transformations to the images during training and then feeding them to the EfficientNet version B0. However, for the detection task using YOLOv5s format, we perform the augmentations offline on the dataset before feeding it into the model.

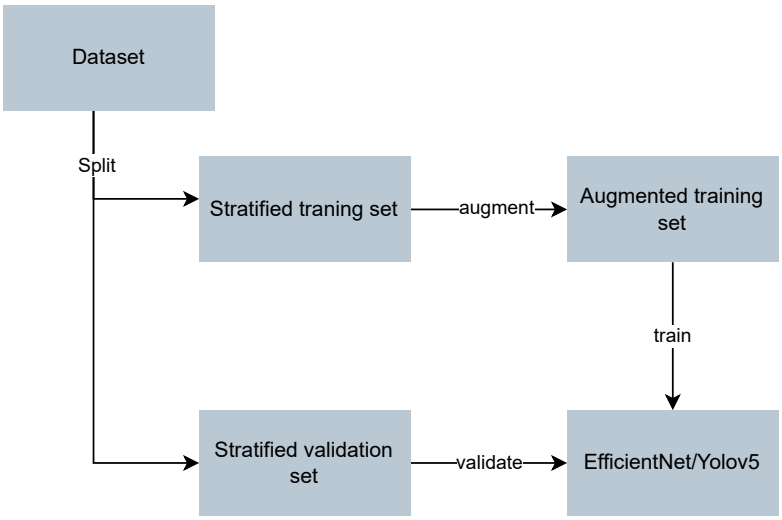


Figure 4.7: Experiment process for basic augmentations (Color-space, geometrical, cutout and mixed augmentations)

4.3.2 Pipeline for GAN-generated augmentation

Figure 4.8 shows the pipeline for GAN-generated methods. We focus on improving the edge cases in our dataset. To achieve this, we only train the GAN on classes that have less than 10 samples, which we consider as edge cases. Once the GAN is trained, we use it to generate half the amount of available images for the edge cases and add them to the training set. We then train this extended training set without any additional augmentation and compare the results to the baseline. It is also important to note that we pay special attention to the underrepresented classes in our dataset. This is because we want to ensure that our models perform well on all classes, not just the most frequently occurring ones. By comparing the results of our GAN-generated methods on the edge cases to the baseline, we can see if the model's performance has been significantly improved for these underrepresented classes.

The performance of GANs heavily relies on the specific architecture of the networks used. Both vanilla GAN and Wasserstein GAN have their own unique architectures that can significantly impact their performance. In the following sections, we will delve into the details of the architectures used for vanilla GAN and Wasserstein GAN.

Vanilla GAN architecture

The architecture of our vanilla GAN, depicted in Figure 4.9, uses a concatenated input of the one-hot coded label and noise for the generator. The generator is composed of four dense layers, each followed by LeakyReLU activation layer with a slop of 0.2. Starting with 512 neurons, the output has a size of 196608 at the end, which is then reshaped into (3, 256, 256) after going through tanh() function for normalization to the range of -1 and 1. For the discriminator, the input is

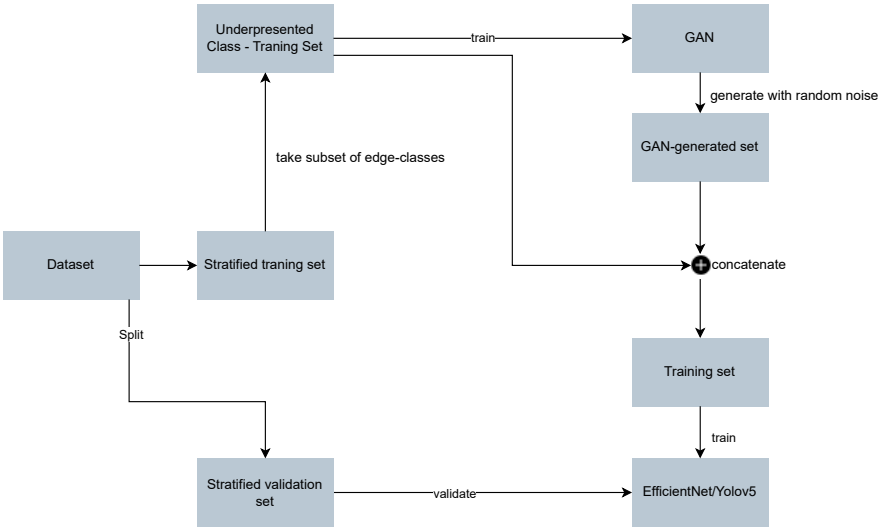


Figure 4.8: Experiment process for GAN-generated approach

flattened and concatenated with the embedded label. The discriminator also has four dense layers, each followed by LeakyReLU activation layer with a slop of 0.2 and Dropout with a probability of 0.3. The output after the sigmoid layer is a single value between 0 and 1, predicting whether the image is real or fake.

Wasserstein GAN architecture

The architecture used in our experiments is illustrated in Figure 4.10, which is based on a DCGAN structure [32]. In the generator, the input noise is concatenated with the labels that are embedded into a one-hot vector, and fed into a ConvTranspose2D block (also known as a 2D Deconvolution block), followed by batch normalization and ReLU activation. We start with 8192 features and halve the depth while doubling the height and width for every subsequent block until we obtain a 256x256 image with 3 channels after 6 blocks, for a model with 256 features. Finally, we use the Tanh() layer to normalize the output within the range of -1 to 1.

The discriminator architecture consists of a dense layer with 65536 units after one-hot encoding for the label input. The output is then reshaped into (1, 256, 256) and concatenated with the image to predict. This concatenated input is given to a block of 4x4 Conv2D layer, followed by an instance normalization layer (except for the first layer) and a LeakyReLU activation layer with a slop of 0.2. At each subsequent Conv2D layer, the height and width are reduced by half, but the depth is doubled. Finally, at the last layer, the input of (8192, 4, 4) is passed through a convolutional layer with a filter size of 4x4 to produce a single-valued output, which gives the label of the class.

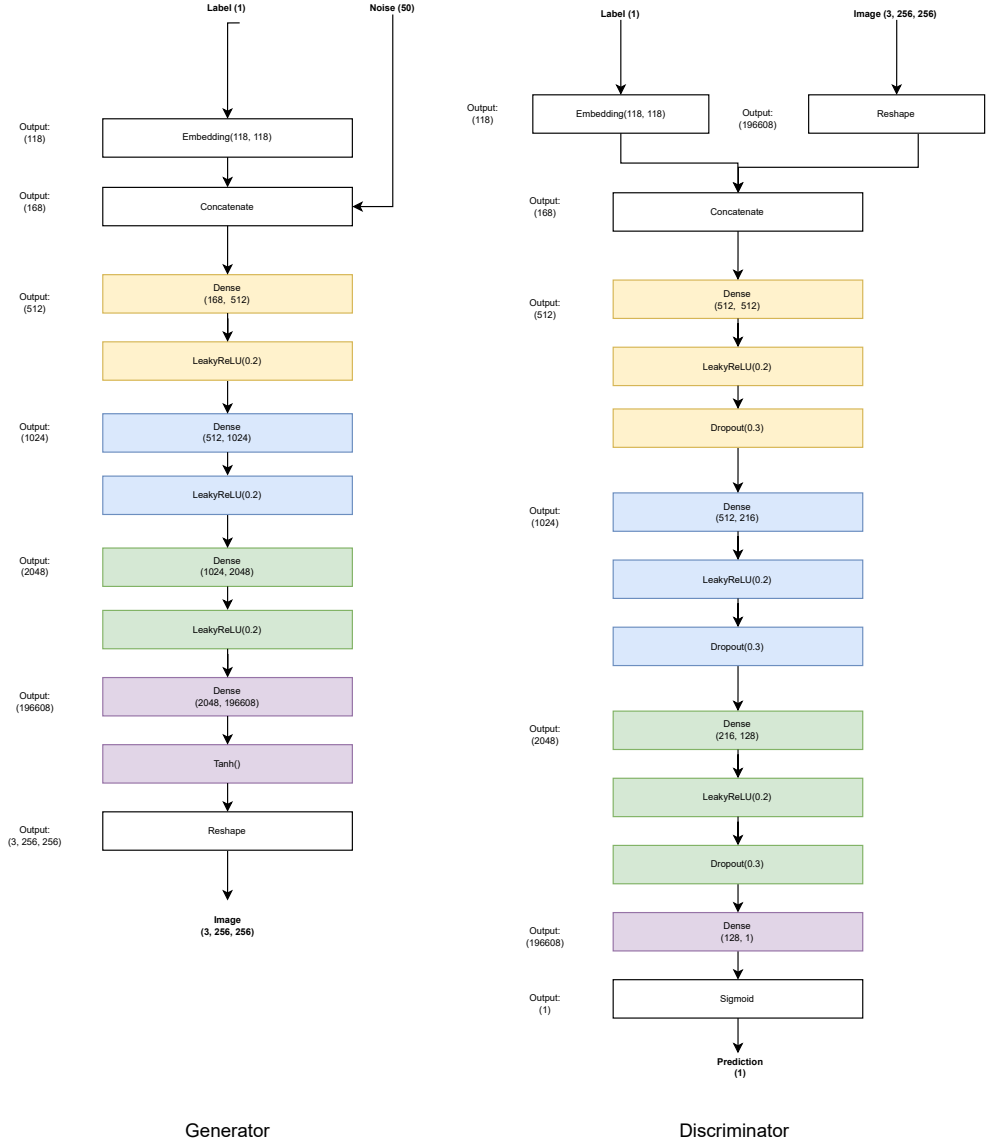


Figure 4.9: he architecture of our generator and discriminator in Vanilla GAN

4.3.3 WGAN-GP architecture

Our WGAN-GP architecture is inspired by implementation of Jalola [22]. For WGAN-GP, we use a different architecture than WGAN by incorporating Upsampling Residual Blocks instead of ConvTranspose2D layers for the Generator and Downsampling Residual Blocks for the Discriminator. The Generator architecture is depicted in Figure 4.11. The Upsampling Residual Block contains an UpSampleConv layer after the input, which is used as a shortcut to the output. This is followed by BatchNorm, ReLu layers, and Conv2D layer, which are repeated once more before being concatenated with the shortcut to produce the output. The Upsampling Residual Block has the same effect on the input size as the ConvTranspose2D, halving the depth while doubling the height and width. In

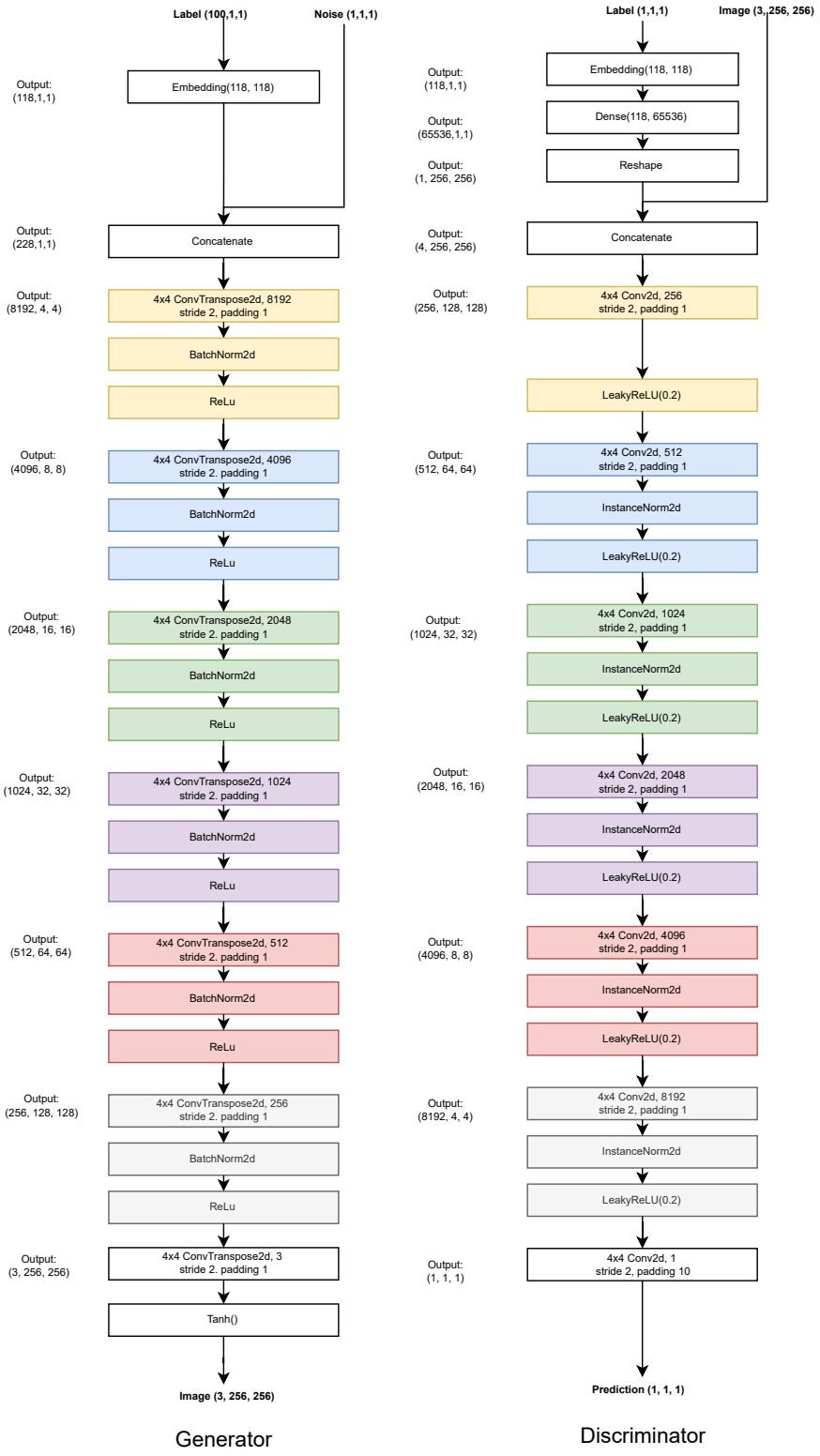


Figure 4.10: The architecture of our generator and discriminator in WGAN

the generator, the embedded Label and noise are concatenated and provided as input to a Conv2d layer, which is followed by four Upsampling Residual Blocks before being given to the another chain of BatchNorm, Conv2D and ReLU layers. Finally, the output is normalized using the tanh() layer.

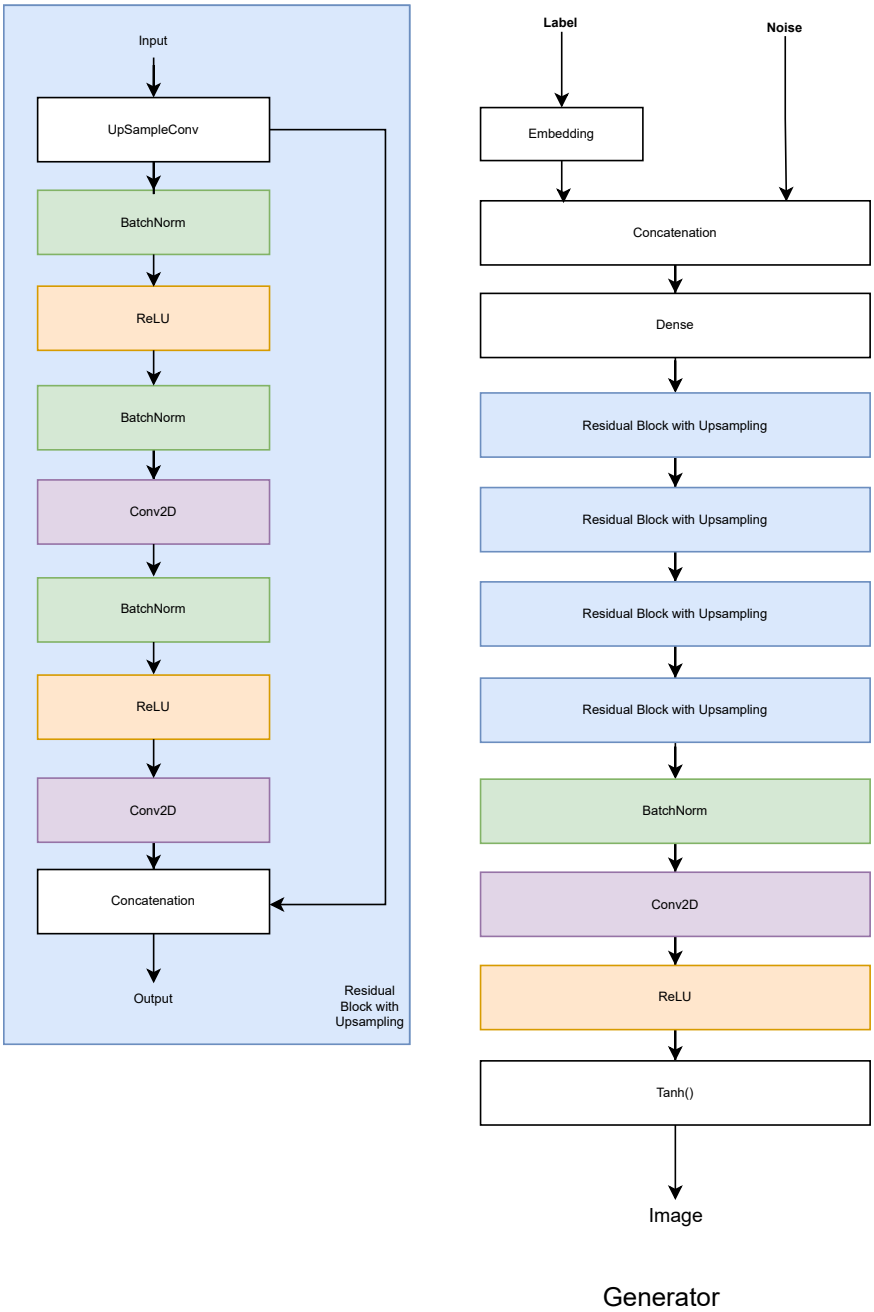


Figure 4.11: The architecture of our generator in WGAN-GP

Figure 4.12 illustrates the architecture of the discriminator in WGAN-GP. The Residual Block of the discriminator differs from that of the generator in that it uses DownSampleConv instead of UpSampleConv right after the input. The image is

concatenated with the embedded label and passed through a dense layer to reduce the number of neurons. This is then followed by four blocks of Residual Block with Downsampling before being given to a dense layer with an output size of one, which produces the final prediction.

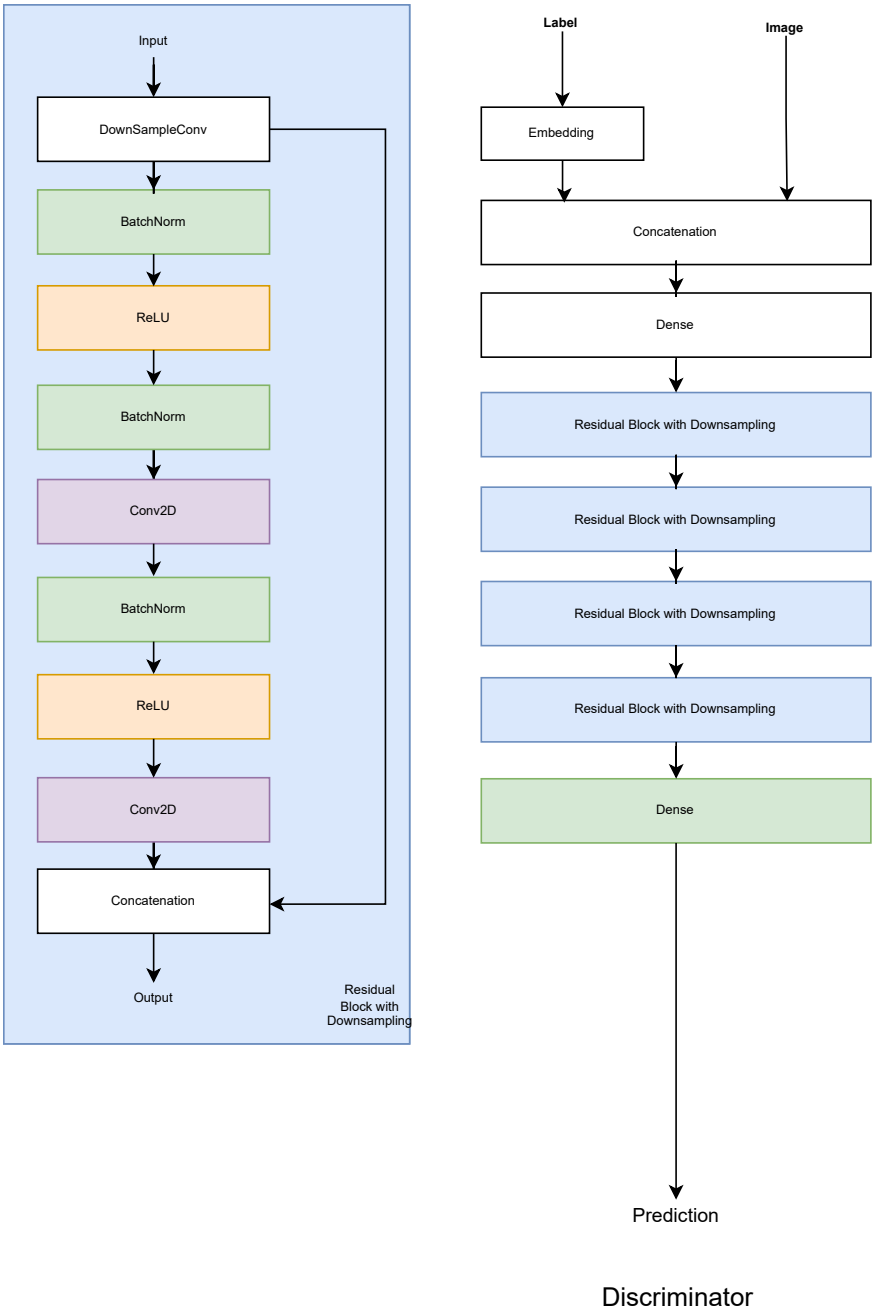


Figure 4.12: The architecture of our discriminator in WGAN-GP

Chapter 5

Results and Analysis

This chapter is dedicated to presenting the results of our experiments. We will begin by showcasing the outcomes of the hyperparameter tuning process for the evaluation models and the augmentation techniques. Then we will present the results of both the classification and detection tasks for the basic augmentations. Finally, we will dive into the results of using GAN to enhance the performance of edge cases. To improve the visual representation, a smoothing average window of 0.7 was applied to all graphics.

5.1 Baseline and Hyperparameter tuning

In this section, we present the baseline for the YOLOv5s and EfficientNet for the given task, along with the optimal hyperparameters for the augmentation techniques. The baseline experiments were conducted without any data augmentation and involved training the models both with and without pre-trained weights. The evaluation metrics used to measure the performance of the models include F1 score Top-3 and mAP@[0.5:0.95].

5.1.1 Pretrain weights and learning rate

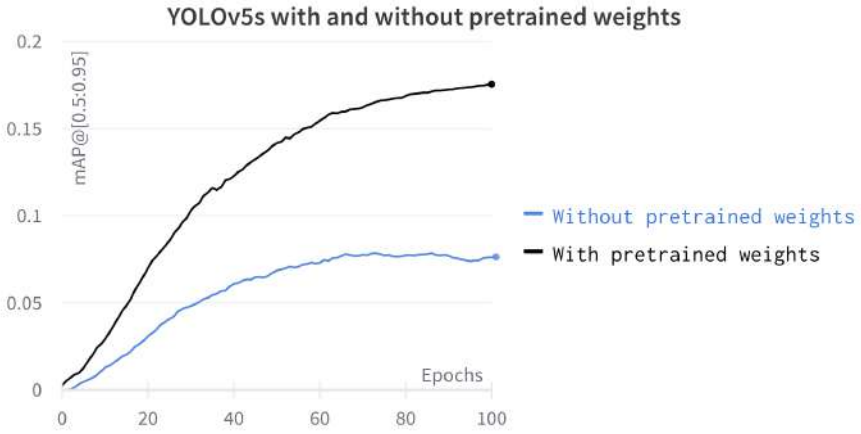
We perform learning rate hyperparameter tuning for our evaluation models and consider using either random weights or pretrained weights. Selecting the best possible evaluation model is crucial as it can greatly affect the evaluation of our augmentation techniques. An underperforming model may produce incorrect predictions, leading to inaccurate results and unreliable conclusions.

YOLOv5s

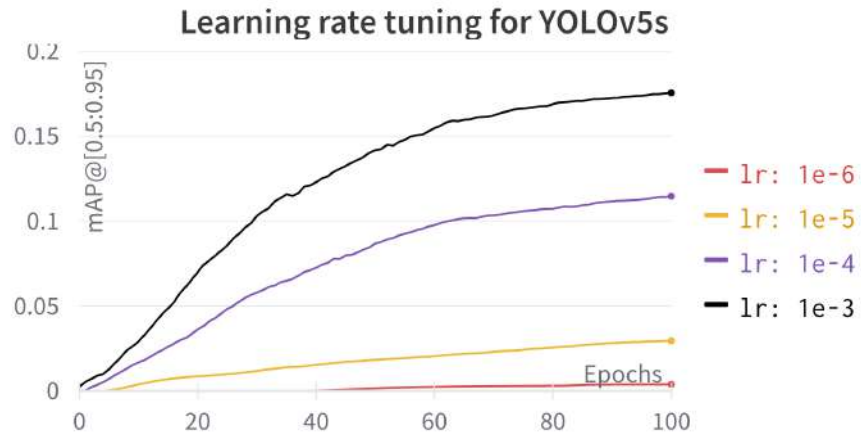
Two YOLOv5 models were trained without augmentation for 100 epochs: one with pretrained weights from the COCO dataset, and one without pretrained weights. Results in Figure 5.1a) demonstrate that the pretrained model outperforms the non-pretrained model. This suggests that the general features learned during pre-

training, such as edges, corners, and textures, are still useful for detection in our dataset even though the COCO is not specific to aerial imagery [27].

To further optimize performance, a hyperparameter tuning experiment was conducted to determine the optimal learning rate. The results, shown in Figure 5.1b), indicate that the best learning rate was 0.01, which achieved an F1-Score top-3 of 0.1773. To summarize, we have determined that YOLOv5s with pretrained weights and a learning rate of 0.01 is the baseline for the deteciton task.



(a) With and without pretrained weights



(b) Learning rate tuning

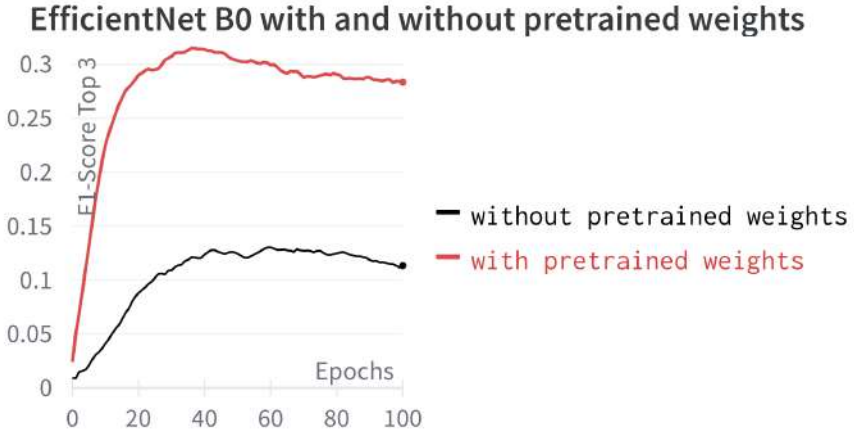
Figure 5.1: Illustration of (a) YOLOv5s with and without pretrained weights and (b) on different learning rate

EfficientNet

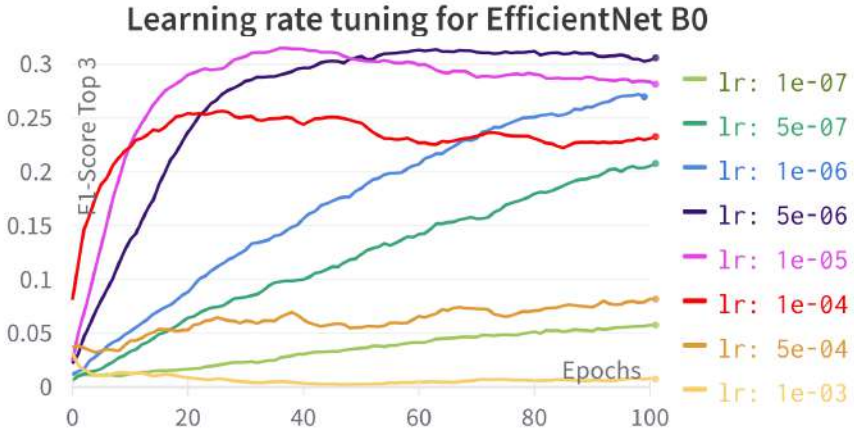
The EfficientNet was trained for 200 epochs using both random-initialized weights and pretrained weights from the Imagenet dataset, without any augmentation. The

results, as illustrated in figure 5.2a), demonstrate that the pretrained model outperformed the model without pretrained weights, suggesting that learning transferability exists even when the datasets lack significant common features.

After conducting hyperparameter tuning within a logarithmic search space between 10^{-4} and 10^{-8}, we determined that the optimal learning rate for EfficientNet version B0 with pretrained weights was 10^{-5}, as demonstrated in figure 5.2b). This yielded an F1-Score top-3 of 0.3434. Thus, we established the baseline for the classification task as using EfficientNet version B0 with pretrained weights and a learning rate of 10^{-5}.



(a) With and without pretrained weights



(b) Learning rate tuning

Figure 5.2: Illustration of EfficientNet version B0 (a) with and without pretrained weights and (b) on different learning rates

5.1.2 Augmentation Hyperparameter Tuning

The augmentation techniques that required tuning included CLAHE, Color Jittering, Griddropout, Cutout, Sharpen, Motion Blur, Perspective, and RandAugment. We conducted the tuning process using grid search and random grid methods. Table 5.1 presents a summary of the optimal hyperparameters obtained from the tuning process, and a more detailed description of the hyperparameter search can be found in the Appendix.

Table 5.1: Hyperparameter tuning results for augmentations

Method	**Hyperparameter**	**Optimal values**	**Search Space**
CLAHE	Clipping Limit	16	[2, 4, 8, 16, 32]
	Filter Size	4	[2, 4, 8, 16, 32]
Color Jittering	Brightness limit	0.8	Randomly between 0.2 and 0.8
	Hue	0.42	Randomly between 0.2 and 0.8
	Contrast limit	0.54	Randomly between 0.2 and 0.8
	Saturation limit	0.65	Randomly between 0.2 and 0.8
GridDropout	Hole Ratio	0.5	[0.2, 0.3, 0.4, 0.5, 0.6]
Sharpen	Alpha range	(0.6, 0.8)	0.2-interval between 0.2 and 0.8 with window sliding step of 0.1
	Lightness range	(0.6, 1.0)	0.2-interval between 0.2 and 0.8 with window sliding step of 0.1
Motion blur	Blur Limit	21	[11, 21, 31, 41, 51, 61, 71]
Perspective	Scale limit	0.2	[0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
RandAug	Number of combinations	2	[2, 3, 4]

5.2 Results of basic augmentation techniques

In this section, we present the outcomes of applying fundamental augmentations, which include Geometric, Color-Space, Cutout, and RandAugment techniques. The section is divided into two parts, with the first one showcasing the results of the detection task using YOLOv5s, and the second one presenting the findings of the classification task using EfficientNet B0.

5.2.1 Detection Task

We trained the pretrained YOLOv5 model using a learning rate of 0.01 for a total of 100 epochs. Table 5.2 provides a summary of the resulting performance metrics, including the average mean precision at 0.5 IoU threshold (mAP@0.5), average mean precision at IoU threshold between 0.5 and 0.95 (mAP@[0.5:0.95]), best precision, and best recall. Following the overall results, we proceed to analyze the outcomes of each group of augmentations, namely Geometric Augmentation, Color-Space Augmentation, Cutout, and RandAugment.

Overall

To summarize, most of the geometric augmentations negatively impacted the model's performance, with the exception of CenterCrop. CenterCrop achieved the highest mAP@0.5 and mAP@0.5:0.95, respectively with values of 0.3436 and 0.5214, which is nearly two times better than the baseline performance. On the other hand, color-space augmentations generally improved the model's performance, with the best augmentation technique being ToSepia, which scores 24% in mAP@[0.5:0.95], a notable 6.29% improvement over the baseline. Cutout and GridDropout had a notable negative impact on the model's performance, resulting in a decrease of 2% and 7.5%, respectively. However, the worst-performing augmentation technique was RandAugment, which produced a staggering 10% decrease in the mAP.

Geometric Augmentation

The mAP@[0.5:0.95] metric for the geometric augmentation technique over 100 epochs is visually represented in Figure 5.3. As demonstrated in the results, CenterCrop augmentation technique outperforms all other augmentation techniques. This observation confirms our initial assumption that an augmentation technique that ensures center invariance can considerably enhance the model's performance in terms of both bounding box detection and class prediction, as depicted in Figure 5.4.

Although it is reasonable to assume that motion blur would not improve a model's performance due to the added noise, the fact that the Rotation, Flipping, and Perspective augmentations decreased the overall performance of the model is counter-intuitive. These transformations are generally expected to increase the robustness of the training set, but they may actually have an adverse effect on the model's ability to learn.

One possible explanation for this unexpected outcome is that YOLOv5 is a bounding box detection model. A modification in the geometric shape of the object can result in a more varied squared shape of the bounding box, which can slow down the training convergence. This assumption is supported by the results shown in figure 5.4, where we observe that although the classification loss did not substantially increase with the augmentations (except for Perspective), the box

Table 5.2: Results of augmentations on YOLOv5s at fine-tuned learning rate 10^{-3}

Method	mAP@[0.5:0.95]	mAP@0.5	Precision	Recall
Baseline	0.1773	0.2879	0.9298	0.6391
CenterCrop 0.2	**0.3436**	**0.5214**	0.7719	0.6865
Flipping	0.1584	0.2772	**0.9576**	**0.6931**
Motion Blur	0.1517	0.2256	0.4693	0.2177
Perspective	0.1244	0.1999	0.2695	0.2256
Rotation	0.1427	0.2303	0.3325	0.2665
CLAHE	0.2152	0.3369	0.7346	0.4768
ChannelShuffle	0.235	0.3619	0.7651	0.575
GaussNoise	0.2077	0.3236	0.7858	0.525
Solarize	0.1612	0.2683	0.3872	0.273
Sharpen	0.2402	0.3604	0.7413	0.4545
ToGray	0.2394	**0.363**	**0.7741**	**0.5432**
ToSepia	**0.246**	0.3757	0.7404	0.5437
Cutout	**0.1503**	**0.2481**	0.4137	0.2427
GridDropout	0.1023	0.1535	0.3643	0.1549
RandAugment	0.07167	0.1347	0.5115	0.1305

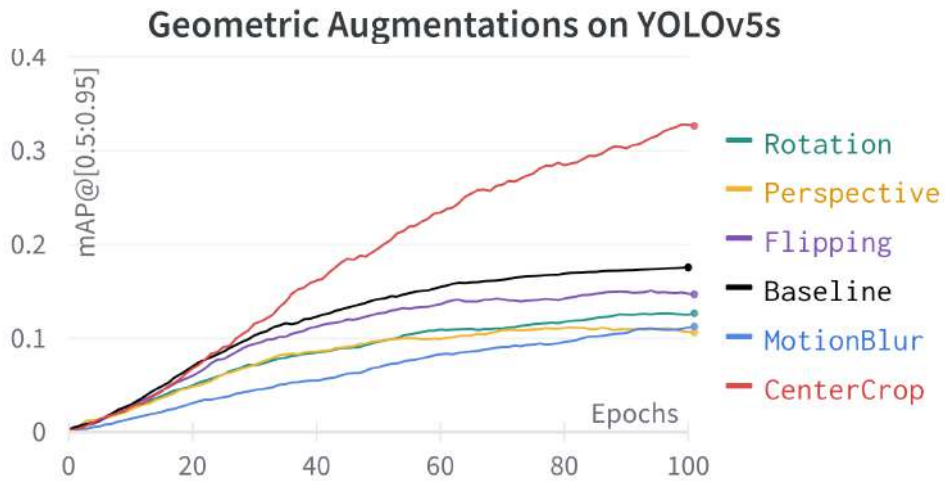
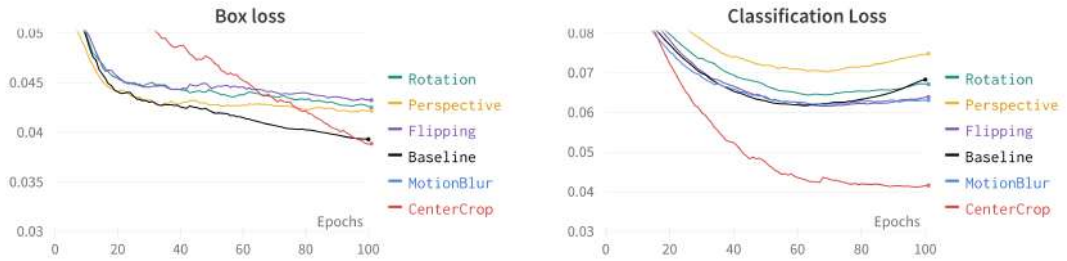


Figure 5.3: Mean average precision in threshold interval between 0.5 and 0.95 of YOLOv5s with geometric augmentations

loss of Flipping, Perspective, and Rotation decreased much slower than the baseline. Hence, the architecture of the YOLOv5 model needs to be considered while selecting augmentation techniques for improving the model's performance.



(a) Box loss (b) Classification loss

Figure 5.4: Illustration of (a) box loss and (b) classification loss in geometric augmentation group

Color-Space Augmentation

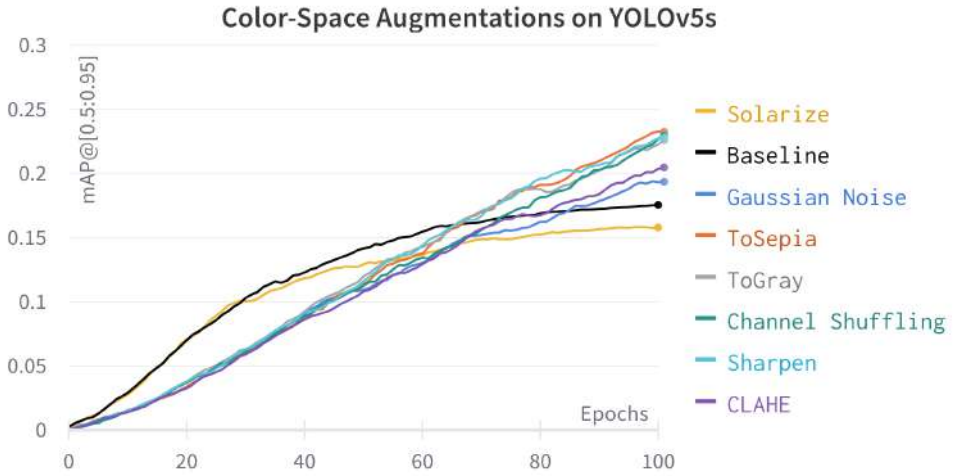
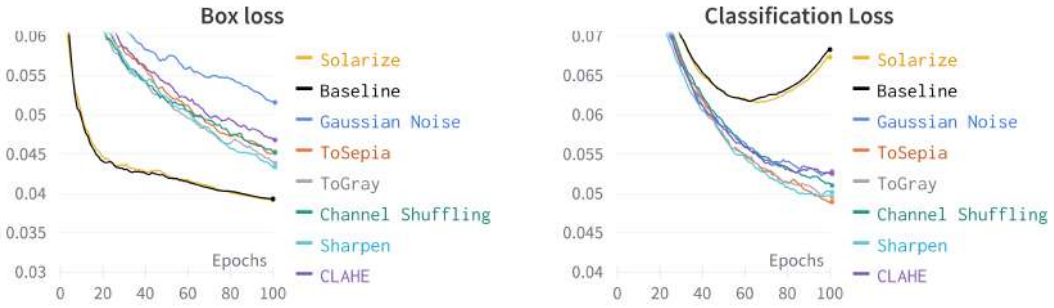


Figure 5.5: Mean average precision of YOLOv5s with Color-Space Augmentations

Figure 5.5 provides a visualization of the mAP@[0.5:0.95] metric for color-space augmentations. In general, all color-space augmentation techniques significantly improve the performance of YOLOv5.

The plot in Figure 5.6 illustrates the trade-off between classification loss and box-loss in color-space augmentation. The extra sharpness and contrast introduced by this augmentation can significantly improve the model's ability to predict the correct class label. However, this comes at a cost of decreased accuracy in detecting the bounding box. This indicates that the model with color-augmentation could also learn features of the background to aid in object classification.

While CLAHE and Sharpen work by enhancing the contrast and brightness of images, making it easier for the model to detect and classify objects, the fact that ToSepia and ToGray augmentations also improve performance is noteworthy. Interestingly, ToSepia outperforms all other color-space augmentations, indicating that



(a) Box loss

(b) Classification loss

Figure 5.6: Illustration of (a) box loss and (b) classification loss in Color-Space Augmentations

reducing color information can enhance the model's detection ability. By reducing the variances in the three channels and consolidating them into one dimension, the model can focus more on the object's shape rather than color, resulting in improved performance.

Cutout and RandAugment

Figure 5.7 shows that the performance of the model is reduced by both Cutout and GridDropout. Furthermore, RandAugment causes the most significant reduction in performance, with the largest margin observed among all augmentation techniques.

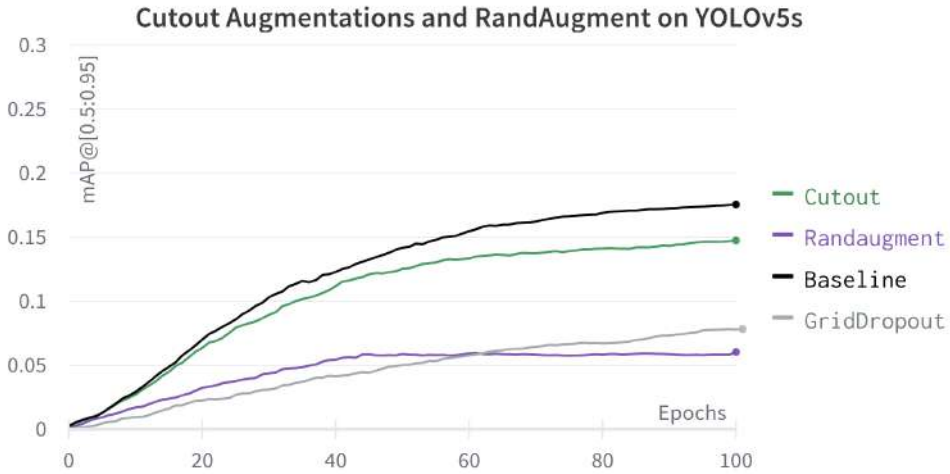
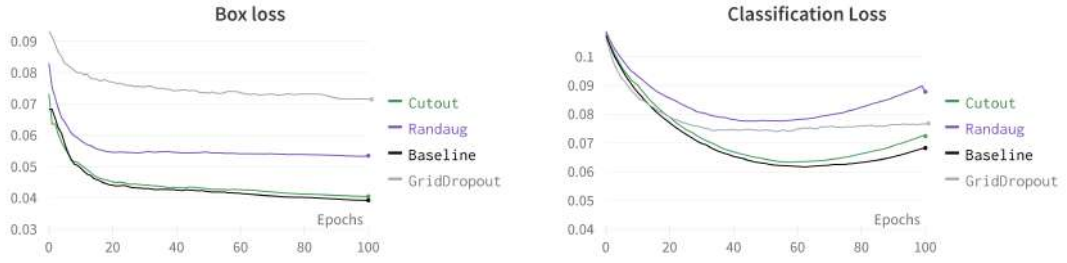


Figure 5.7: Mean average precision of YOLOv5s with Cutout Augmentations and RandAugment

The reasons for the decrease in performance caused by Cutout and Dropout can be relatively intuitive, as they are non-label-preserving augmentation techniques

that may remove crucial information required for the model to detect and classify objects accurately. Figure 5.8 demonstrates that Cutout produces a significant increase in the box loss, even more so than GridDropout, which actually removes more information. Meanwhile, RandAugment performs poorly in both classification and box losses, introducing too many inconsistencies into the model and hindering its ability to identify crucial features.



(a) Box loss

(b) Classification loss

Figure 5.8: Illustration of (a) box loss and (b) classification loss in Cutout augmentations and RandAugment

5.2.2 Classification Task

Table 5.3 presented Top-3 and Top-1 metrics obtained from the classification outcomes, which are categorized into four sections, namely baseline, geometric transformation, color-space augmentation, and cutout-related augmentation. The training process was carried out over 200 epochs and implemented a learning rate of 0.0001. The highest metric in each group is emphasized in bold. Following the overall results, we proceed to analyze the outcomes of each group of augmentations, namely Geometric Augmentation, Color-Space Augmentation, Cutout, and RandAugment.

Overall

In summary, most of the augmentation techniques did not lead to an improvement in the performance of EfficientNet. However, Rotation, Flipping, Perspective, and Gaussian Noise augmentations did show positive impacts on the model's performance. In particular, Rotation outperformed all other augmentations in terms of metrics and improved the baseline F1-Score Top-3 by almost 10%. While Gaussian Noise was the only augmentation in the color-space group that had a positive impact on the performance, the difference was relatively small, with an increase of 0.0064 in F1-Score Top-3. Finally, regularization techniques such as cutout and dropout worsened the model's performance.

Table 5.3: Result of augmentation on EfficinetNet B0 at fine-tuned learning rate 10^{-5}

	Top-3 Metrics			Top-1 Metrics		
Method	**F1-score Top-3**	**Accuracy Top-3**	**Precision Top-3**	**F1-score Top-1**	**Accuracy Top-1**	**Precision Top-1**
Baseline	0.3434	0.4977	0.3231	0.2784	0.3339	0.2264
Rotation	**0.4423**	**0.5876**	**0.432**	**0.3364**	**0.4586**	**0.2585**
Flipping	0.4208	0.5826	0.4104	0.3296	0.4351	0.2563
Perspective	0.3577	0.5209	0.3339	0.286	0.3414	0.218
CenterCrop 0.2	0.3252	0.5111	0.3028	0.2829	0.3284	0.2204
CenterCrop 0.4	0.316	0.5205	0.3023	0.2807	0.3333	0.209
MotionBlur	0.3347	0.5209	0.3233	0.3054	0.3486	0.2353
CLAHE	0.3027	0.4654	0.2932	0.2569	0.3093	0.1971
Sharpen	0.3393	0.5095	0.3168	**0.306**	0.3525	0.2486
ChannelShuffle	0.314	0.4682	0.2945	0.2646	0.3188	0.2011
Solarize	0.2354	0.4165	0.2173	0.2251	0.2191	0.1666
Gaussian Noise	**0.3498**	0.5154	**0.3289**	0.2868	**0.3546**	0.2168
MotionBlur	0.3347	**0.5209**	0.3233	0.3054	0.3486	**0.2353**
ToGray	0.2941	0.4637	0.2787	0.2638	0.3011	0.211
ToSepia	0.329	0.4708	0.3183	0.2695	0.3436	0.2111
Cutout	0.2769	0.4891	0.2167	0.3387	0.3199	0.3396
GridDropout	0.2827	0.4894	0.2167	0.333	0.3154	0.3454
RandAugment	0.2929	0.5243	0.3789	0.3598	0.3731	0.2174

Geometric Augmentations

Figure 5.9 depicts the classification performance of EfficientNet B0 when using geometric augmentations. Interestingly, contrary to the findings in the detection modeland our initial hypothesis, CenterCrop augmentation does not seem to improve the model's classification performance. This observation implies that the model may require contextual information from the background to accurately predict the class. For example, in the case of a flock of birds, it may indicate a specific bird class that is commonly seen in a group, such as Graugans. Additionally, the geographical background may appear more frequently in certain bird classes, such as Brandgans, which could also assist in accurate classification.

Alternatively, it is possible that EfficientNet is already designed to learn features at different scales and resolutions, and by reducing the information, the model may lose some crucial details for a specific class. Therefore, CenterCrop augmentation may not be as effective in improving the model's classification performance.

However, Rotation, Flipping, and Perspective augmentations demonstrate their effectiveness in helping the model learn features in different geometric positions.

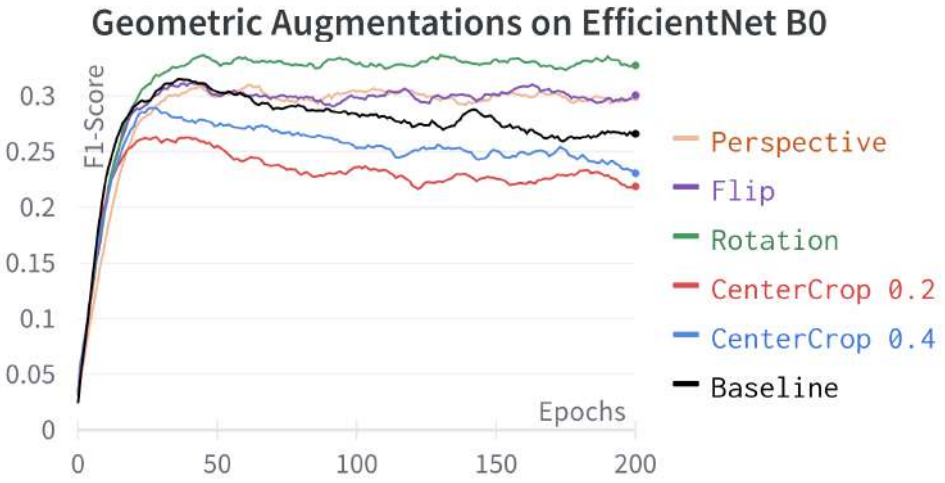


Figure 5.9: Performance of Efficientnet B0 with Geometric Augmentations with fined-tuned learning rate 10^{-5}.

With rotation the the F1-score improved substantially by 0.1.

Color-Space Augmentations

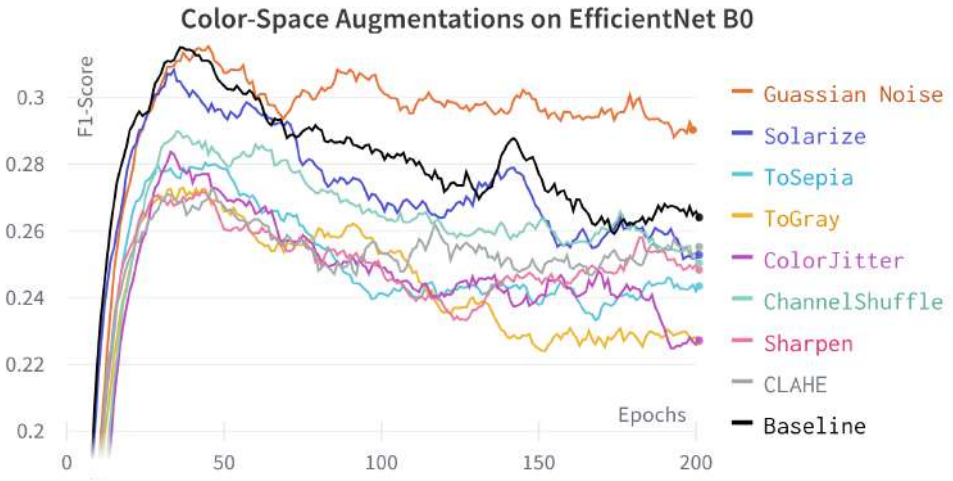


Figure 5.10: Performance of Efficientnet B0 with Color-space Augmentations with fined-tuned learning rate 10^{-5}.

Figure 5.10 displays the outcomes of the impact of color-space transformation on the performance of EfficientNets. The overall effect of this type of transformation is either negligible or insignificant with the exception of Gaussian Noise, which has a minimal positive effect. A possible explanation for this observation is that the architecture of EfficientNet is inherently designed to be resilient to color variations.

The model employs a feature extractor network that learns to extract meaningful features from the input image, which is then coupled with a scaling method to ensure proper normalization of the input data. This design choice enhances the model's invariance to changes in color-space, rendering color-space augmentation less effective in improving the model's performance.

Cutout Augmentations and RandAugment

The results presented in Figure 5.11 demonstrate the impact of the cutout transformations and RandAugment on the performance of EfficientNets. It can be observed that the cutout transformation causes a decrease in the network's performance due to the loss of information. Similarly, incorporating RandAugment in the training pipeline results in a decrease in the network's performance. This could be due to the extreme nature of the augmentation techniques used in RandAugment, which can potentially slow down the convergence of the network during training.

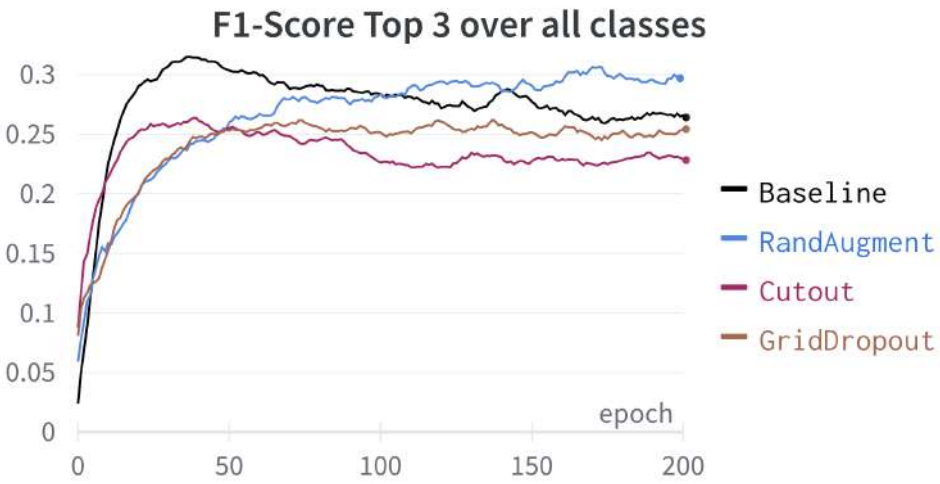


Figure 5.11: Performance of Efficientnet B0 with Cutout, GridDropout and RandAugment with fined-tuned learning rate 10^{-5}.

5.3 GAN-generated augmentations

In this section, we will examine the outcomes of the GAN-generated methods, with a particular focus on evaluating the image quality, diversity, and effectiveness of the generated images on edge cases. The primary objective of this analysis is to highlight the common issues that can arise when using GANs as an augmentation technique and to compare the performance of the three architectures with one another.

Because of their complex architecture and the extensive number of parameters that must be adjusted, Generative Adversarial Networks (GANs) often necessitate

a significant amount of time to train. Additionally, the training process can be computationally intensive, requiring high-end hardware and substantial computational resources. Table 5.4 provides an overview of the number of trainable parameters, training time, and epochs required for each GAN employed in this thesis.

Model	Number of trainable parameters	Epochs	Training Time
Vanilla GAN	405574756	10000	6h 58m 24s
WGAN	743760999	5000	1d 5h 31m
WGAN-GP	7645523	3000	10h 4m 50s

Table 5.4: Comparison of GAN models

Image quality

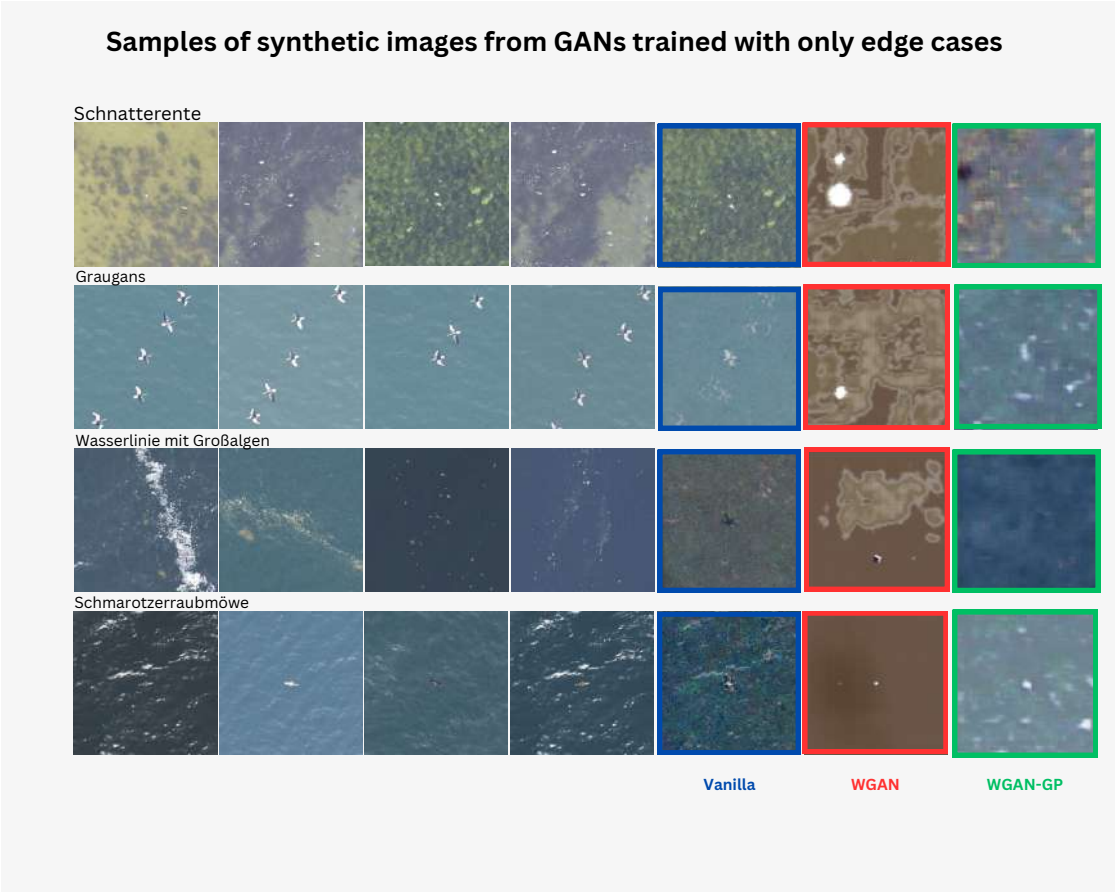


Figure 5.12: Samples of generated images from the conditional GANs

The samples of generated images from three different types of GANs are displayed in Figure 5.12. Each row represents an example of an edge class, such as Schnatterente, Graugans, Wasserlinie mit Großalgen, and Schmarotzerraubmöwe.

The first four columns show the input images used to train the different GAN models.

The images generated from Vanilla GAN appear to be the most realistic compared to other types of GANs. However, it is also observed that Vanilla GAN is prone to getting trapped in a local optima, resulting in generated images that closely resemble a specific real image, as seen in the case of the Schnatterente class.

As illustrated with small seeming random patches in WGAN, when trained on a small dataset, WGAN did not have enough information to learn the underlying distribution of the data and therefore generate unrecognizable images. This is in contrast to vanilla GAN, which may perform better on smaller datasets but may suffer from mode collapse.

Wasserstein GAN with gradient penalty has been demonstrated to produce generated samples that are more similar to the distribution of the training data in comparison to Vanilla WGAN. Although the generated images still exhibit a deficiency in high-resolution details, they share some similarities with the objects in the dataset. For instance, the class of Schnatterente features white dots and yellow patches, while the class of Wasserlinie mit Großalgen displays a similar shade of blue as shown in 5.12. This observation highlights the issue of hypersensitivity to hyperparameters in WGAN. In the absence of hyperparameter tuning for the model, it failed to converge rapidly.

Diversity

Figure 5.13 displays different samples generated by 3 types of GANs with varying noise inputs across 3 classes. It is evident that Vanilla GAN suffers from the mode collapse problem, as it only produces one repeated image with slight variations in brightness and hue. On the other hand, WGAN and WGAN with GP successfully generate diverse samples.

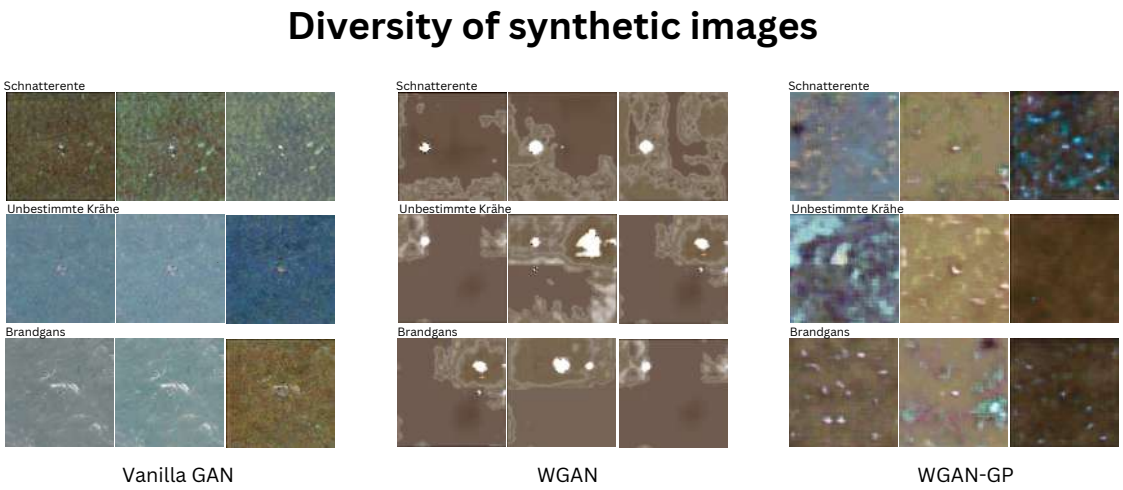
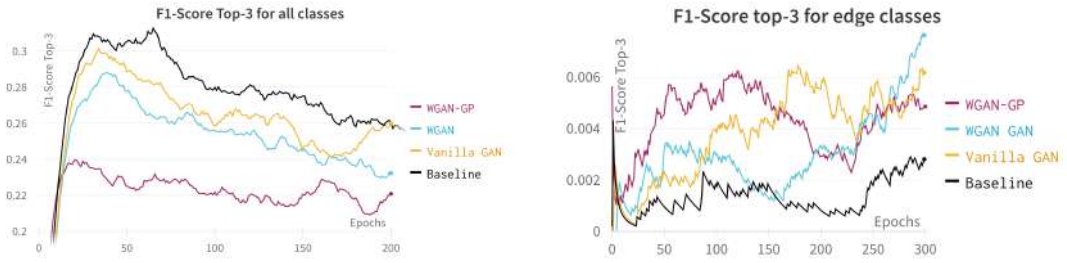


Figure 5.13: Variety of generated pictures in three GANs

Performance



(a) F1-Score Top-3 for all classes

(b) F1-Score Top-3 for only edge classes

Figure 5.14: F1-Score (a) for all classes and (b) for edge classes

Figure 5.14(a) depicts the F1-Score of EfficientNet when trained with additional GAN-generated data, while Figure 5.14(b) displays the F1-Score for edge cases. Overall, the overall classification performance decreases with the use of GAN-generated data. One possible explanation is that the GAN-generated images may contain noise or inconsistencies that are absent in the original dataset, thereby confusing the model and making it more difficult to generalize to new examples. Additionally, GAN-generated images may not cover the entire distribution of the original dataset, leading to overfitting and poor generalization performance.

WGAN with GP causes the largest decrease in F1-Score Top-3, with a decrease of nearly 5%. On the other hand, using data from Vanilla GAN leads to the smallest decrease in performance, with a decrease of only 1.8%. For the edge classes, both WGAN and Vanilla GAN show an improvement in F1-Score Top-3. WGAN results in a larger improvement of 0.013, while Vanilla GAN yields a smaller improvement of 0.0014. However, it is important to consider the overall decrease in performance when using GAN-generated data. The primary reason for the improvement in edge cases may be the introduction of inconsistencies and new features, which the network then learns to recognize. However, this improvement may not outweigh the decrease in performance seen in the rest of the dataset. In the case of WGAN with GP, the overall decrease in performance is 0.0355, which may indicate that the network is overfitting to the GAN-generated data. In the case of Vanilla GAN, the overall decrease in performance is 0.0205, which is still significant but less than that of WGAN with GP.

Chapter 6

Conclusion

This thesis investigated the effectiveness of various types of augmentations, including geometric, color-space, cutout, and mixed, for the marine wildlife image dataset in both classification and detection tasks. Specifically, we utilized EfficientNet B0 for classification and YOLOv5s for detection.

Furthermore, we explored the potential of GAN-generated methods to improve the classification of underrepresented classes (edge class) and implemented Vanilla GAN, Wasserstein GAN (WGAN), and Wasserstein GAN with Gradient Penalty (WGAN-GP) to achieve this goal.

In this chapter, we will discuss the main findings of the research, the limitations of the thesis, and potential avenues for future research.

6.1 Main findings

The results from our study indicate that the effectiveness of augmentation techniques is not only dependent on the dataset used, but also highly influenced by the task and model employed. As demonstrated in both the classification and detection tasks, certain augmentation policies proved effective for one model but failed to yield improvements in another. For instance, the CenterCrop technique significantly improved the mean average precision (mAP) score in YOLOv5s, but resulted in decreased performance in EfficientNet B0 for classification. Similarly, while rotation was the most effective policy for EfficientNet, it failed to yield any improvement in YOLOv5.

Geometric augmentations have been observed to have a negative impact on bounding box classification for the dataset under study. Specifically, Rotation, Flipping, and Perspective augmentations were found to decrease the model's performance by generating more varied square shapes of the object. On the other hand, these identical augmentations showcased a significant enhancement in the performance of pixel-based classification (EfficientNet B0), emphasizing once again the model-specific nature of augmentation policy decisions.

In terms of color-space augmentation, a trade-off has been observed when utilizing these techniques with YOLOv5, as they have been shown to improve classi-

fication performance while simultaneously decreasing the model's ability to detect bounding boxes. Despite this limitation, the overall increase in mean average precision (mAP) renders color-space augmentation a favorable method for augmenting YOLOv5. However, for classification tasks, most color-space augmentations appear to be ineffective.

While our study's examination of GAN-generated augmentations is limited, it can be generally stated that vanilla GAN should be avoided due to its tendency to experience mode collapse and the absence of a reliable stopping criterion. Using GAN-GP instead is more practical, as it requires less hyperparameter tuning compared to regular WGAN.

6.2 Limitation of the thesis

The hyperparameter tuning approach used in this work for the augmentation methods can be improved. The training time was limited to 8 iterations due to time and computational constraints, but a more efficient approach such as early stopping could allow for more exploration of the search space.

It is important to note that the hyperparameter tuning was not performed using the optimal learning rate of the classifier 10^{-5}, but rather a learning rate of 10^{-4}. While this may not be an issue if the optimal augmentation only depends on the dataset itself, the results show that each augmentation has a different impact on different models. Therefore, optimizing the hyperparameters for each model and using the optimal learning rate could potentially improve the performance of the augmentation methods.

The determination of a suitable configuration for RandAugment is also a challenging task, as it requires a heuristic approach to determine the number and magnitude of augmentation methods. In this study, only a subset of the best augmentations were employed, which may not be sufficient to fully explore the potential of RandAugment.

In addition, the selection process for GAN architectures (Vanilla GANs, WGANs, and WGAN-GP) in this thesis was not conducted in a systematic manner. An optimal approach is to begin with a well-established architecture such as WGAN-GP and then adjust its hyperparameters.

6.3 Future work

There is significant scope for further research on augmentation techniques for aerial images in general and for this dataset specifically. We suggest a more refined approach that addresses the limitations of our current work. A comprehensive hyperparameter tuning process would offer a promising augmentation policy for the specific dataset, enabling effective comparisons with other aerial image datasets such as BigEarthNet [38], Airbus Wind Turbines [1] for classification tasks, or xView [13] for detection tasks.

Furthermore, expanding the augmentation techniques beyond the ones studied in this thesis, incorporating additional techniques such as adversarial training, neural style transfer, undersampling, and image mixing as mentioned in the related work, could be a promising direction for future research. This approach could potentially provide valuable insights into which augmentation techniques yield the most significant performance improvements with the least computational costs.

While selecting an effective augmentation policy is important, it is equally important to understand why a particular augmentation method is effective. The subject of interpretability holds potential interest. While currently, the augmentation techniques are applied to the dataset, and the resulting performance is merely observed, it would be beneficial to explain the effectiveness of the augmentations by utilizing quantitative metrics such as sharpness and contrast or subjective measures such as human evaluation and eye-tracking.

Additionally, exploring the use of diffusion models, such as Stable Diffusion Models [43], for image generation, is another potential area for future work. Diffusion models model the diffusion process of noise in the image directly, unlike traditional generative models like GANs, which allows for more efficient training and better handling of high-dimensional data. This could potentially improve the effectiveness and efficiency of image generation tasks.

Appendix A

Appendix

Table A.1: Full detailed table of the dataset

Class	**Type**	**Total number of samples**	**Number of sample in testset**
Gryllteiste	bird	4	1
Offshore Windpark	anthro	4	1
Schnatterente	bird	4	1
Buchfink	bird	4	1
unbestimmte Larusmöwe	bird	4	1
Fischereigerät (Schwimmer)	anthro	4	1
Schmarotzer/Spatel/Falkenraubmöwe	bird	5	1
Brandgans	bird	5	1
Wasserlinie mit Großalgen	misc	5	1
unbestimmte Art	bird	5	1
Feldlerche	bird	5	1
Schmarotzerraubmöwe	bird	5	1
Grosser Brachvogel	bird	6	2
Öl	misc	6	2
unbestimmte Raubmöwe	bird	7	2
Turmfalke	bird	7	2
Trauerseeschwalbe	bird	7	2

Continued on next page

Table A.1: Full detailed table of the dataset (Continued)

Class	**Type**	**Total number of samples**	**Number of sample in testset**
Front	misc	8	2
unbestimmter Schwan	bird	8	2
Boot (unbestimmtes kleines Boot)	anthro	8	2
Tonne	anthro	8	2
Sperber	bird	8	2
Kiebitzregenpfeifer	bird	8	2
Skua	bird	9	2
Graugans	bird	9	2
Freizeitfahrzeug	anthro	10	2
unbestimmte Krähe	bird	10	2
Pfuhlschnepfe	bird	11	3
Frachter	anthro	11	3
Ankerboje	anthro	12	3
Fischereifahrzeug	anthro	12	3
Mantel-/Heringsmöwe	bird	13	3
unbestimmter kleiner Wal	mammal	13	3
Papageitaucher	bird	13	3
unbestimmte Limikole	bird	14	3
Versorgungsschiff	anthro	14	3
Gänsesäger	bird	14	3
Trübungsfahne	misc	15	3
Stellnetzfahne rot	anthro	15	3
Pfeifente	bird	15	3
Fisch: sichtbares Fischvorkommen	misc	17	4
Ringelgans	bird	17	4
Fahne zur markierung von Fischereifanggeräten	anthro	19	4

Continued on next page

Table A.1: Full detailed table of the dataset (Continued)

Class	**Type**	**Total number of samples**	**Number of sample in testset**
Mikroalgen (Phytoplankton): Algenwolke unter Wasser	misc	19	4
Rothalstaucher	bird	20	4
Fahrspur eines Wasserfahrzeugs	misc	21	5
Schiff (unbestimmtes großes Schiff)	anthro	21	5
Kegelrobbe	mammal	23	5
Eiderente	bird	23	5
Haubentaucher	bird	25	5
Tang / Makroalgen	misc	25	5
Mikroalgen: Algenblütenpatch an Wasseroberfläche	misc	26	6
Wasserlinie mit Schaum	misc	26	6
unbestimmter Singvogel	bird	27	6
Lachmöwe	bird	27	6
Wasserlinie mit Algenblüte	misc	27	6
Sturm/Silbermöwe	bird	27	6
unbestimmter Meeressäuger	mammal	28	6
Windenergieanlage	anthro	28	6
Höckerschwan	bird	28	6
Samtente	bird	29	6
Arbeitsschiff Windpark	anthro	29	6
unbestimmte Robbe (Kegelrobbe/Seehund)	mammal	29	6
unbestimmte Möwe	bird	29	6
Seehund	mammal	29	6
Ohrentaucher	bird	29	6
Trottellumme	bird	29	6
Trauerente	bird	29	6
Mantelmöwe	bird	30	6

Continued on next page

Table A.1: Full detailed table of the dataset (Continued)

Class	**Type**	**Total number of samples**	**Number of sample in testset**
Mittelsäger	bird	30	6
Prachttaucher	bird	30	6
Sonstiges	misc	30	6
Eisente	bird	30	6
Trottellumme/Tordalk	bird	31	7
Zwergmöwe	bird	31	7
Schweinswal	mammal	31	7
Dreizehenmöwe	bird	31	7
Silbermöwe	bird	31	7
unbestimmter Vogel	bird	31	7
Eissturmvogel/Möwe	bird	32	7
Marine Lebewesen	misc	32	7
Heringsmöwe	bird	32	7
Messboje	anthro	32	7
Basstölpel	bird	32	7
unbestimmte Kleinmöwe	bird	32	7
Boje oder Bojenkette	anthro	32	7
unbestimmter Alk	bird	32	7
Stockente	bird	33	7
unbestimmter Seetaucher	bird	33	7
Seeschwalbe / Kleinmöwe	bird	33	7
unbestimmte Großmöwe	bird	33	7
unbestimmte Ente	bird	33	7
Windenergieanlage im Bau	anthro	33	7
Müll	misc	33	7
Trauerente/Samtente	bird	33	7
Robbe / kleiner Wal	mammal	33	7
Sturmmöwe	bird	33	7

Continued on next page

Table A.1: Full detailed table of the dataset (Continued)

Class	**Type**	**Total number of samples**	**Number of sample in testset**
Wasserlinie mit Müll	misc	33	7
Langmuirstreifen	misc	33	7
Luftballon	misc	33	7
Qualle(n)	misc	33	7
Mantel/Heringsmöwe	bird	33	7
Tordalk	bird	33	7
Sterntaucher	bird	33	7
Eissturmvogel	bird	34	7
Goldregenpfeifer	bird	34	7
unbestimmte Sterna-Seeschwalbe	bird	34	7
Weisswangengans	bird	34	7
Blässhuhn	bird	34	7
Fischereigerät (Doppel-Schwimmer)	anthro	34	7
Kormoran	bird	34	7
Schellente	bird	34	7
unbestimmte Graue Gans	bird	34	7
Fluss-/Küstenseeschwalbe	bird	34	7
Bergente	bird	34	7
unbestimmte Seeschwalbe	bird	34	7
Brandseeschwalbe	bird	34	7

Table A.2: Results of hyperparmeter tuning for CLAHE.

Clipping Limit	Filter Size	F1-Score top-3
2	4	0.098
2	8	0.0965
2	16	0.0357
2	32	0.0607
4	4	0.1111
4	8	0.0862
4	16	0.1189
4	32	0.085
8	4	0.1113
8	8	0.0893
8	16	0.067
8	32	0.0543
16	4	**0.1194**
16	8	0.0765
16	16	0.0393
16	32	0.0282
32	4	0.0536
32	8	0.09
32	16	0.0601
32	32	0.0333

Table A.3: Results of hyperparmeter tuning for Color Jittering.

Brightness	Hue	Contrast	Saturation	F1-Score top-3
0.57	0.45	0.56	0.78	0.0125
0.74	0.73	0.6	0.4	0.0250
0.41	0.28	0.55	0.47	0.0388
0.55	0.2	0.23	0.64	0.0418
0.67	0.26	0.65	0.27	0.0507
0.77	0.77	0.24	0.36	0.0518
0.48	0.55	0.34	0.64	0.0519
0.8	0.42	0.54	0.65	**0.0585**

Table A.4: Results of hyperparmeter tuning for GridDropout.

Hole ratio	F1-Score top-3
0.6	0.1021
0.5	**0.1116**
0.2	0.08904
0.3	0.1013
0.4	0.0914

Table A.5: Results of hyperparmeter tuning for Sharpen.

Alpha range	Lightness range	F1-Score Top 3
(0.6, 0.8)	(0.6, 1.0)	**0.0908**
(0.4, 0.6)	(0.2, 0.6)	0.0965
(0.6, 0.8)	(0.4, 0.8)	0.0075
(0.2, 0.4)	(0.6, 1.0)	0.0774
(0.4, 0.6)	(0.4, 0.8)	0.0724
(0.2, 0.4)	(0.4, 0.8)	0.0878
(0.2, 0.4)	(0.4, 0.8)	0.0624
(0.6, 0.8)	(0.2, 0.6)	0.0568
(0.2, 0.4)	(0.2, 0.6)	0.0849
(0.4, 0.6)	(0.6, 1.0)	0.0477

Table A.6: Results of hyperparmeter tuning for Motion Blur.

Blur limit	F1-Score top-3
71	0.2689
61	0.2825
51	0.2707
41	0.2774
31	0.2831
21	**0.3048**
11	0.2872

Table A.7: Results of hyperparmeter tuning for Perspective.

Scale	F1-Score top-3
0.7	0.2544
0.6	0.2542
0.5	0.2704
0.4	0.3014
0.3	0.3091
0.2	**0.3126**

Table A.8: Classification result when learning rate is 10^{-4} *- a non-optimal classifier*

	Top-3 Metrics			**Top-1 Metrics**		
Method	**F1-score Top-3**	**Accuracy Top-3**	**Precision Top-3**	**F1-score Top-1**	**Accuracy Top-1**	**Precision Top-1**
Baseline	0.2784	0.4977	0.2264	0.3231	0.3434	0.3339
Rotation	**0.3364**	**0.5876**	**0.2585**	**0.432**	**0.4423**	**0.4586**
Flipping	0.3296	0.5826	0.2563	0.4104	0.4208	0.4351
Perspective	0.286	0.5209	0.218	0.3339	0.3577	0.3414
CenterCrop 0.2	0.2829	0.5111	0.2204	0.3028	0.3252	0.3284
CenterCrop 0.4	0.2807	0.5205	0.209	0.3023	0.316	0.3333
MotionBlur	0.3054	0.5209	0.2353	0.3233	0.3347	0.3486
CLAHE	0.2569	0.4654	0.1971	0.2932	0.3027	0.3093
Sharpen	**0.3060**	0.5095	**0.2486**	0.3168	0.3393	0.3525
ChannelShuffle	0.2646	0.4682	0.2011	0.2945	0.314	0.3188
Solarize	0.2251	0.4165	0.1666	0.2173	0.2354	0.2191
GaussNoise	0.2868	0.5154	0.2168	**0.3289**	**0.3498**	**0.3546**
Normalize	0.2778	**0.5123**	0.2152	0.3119	0.3315	0.3395
ToGray	0.2638	0.4637	0.211	0.2787	0.2941	0.3011
ToSepia	0.2695	0.4708	0.2111	0.3183	0.329	0.3436
Cutout	0.2769	0.4891	**0.2167**	**0.3199**	**0.3387**	0.3396
GridDropout	**0.2827**	**0.4894**	**0.2167**	0.3154	0.333	**0.3454**

Bibliography

[1] Airbus wind turbines patches dataset. `https://www.kaggle.com/datasets/airbusgeo/airbus-wind-turbines-patches`, 2021. Accessed: 2023-02-23.

[2] Papers with code - imagenet benchmark (image classification). `https://paperswithcode.com/sota/image-classification-on-imagenet`, 2021. Accessed: 2023-03-01.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[4] Nurshazlyn Mohd Aszemi and PDD Dominic. Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*, 10(6), 2019.

[5] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[8] Jason Brownlee. A gentle introduction to generative adversarial networks (gans). `https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/`, 2019. Accessed: 2023-03-06.

[9] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020.

[10] Yung-Chien Chou, Cheng-Ju Kuo, Tzu-Ting Chen, Gwo-Jiun Horng, Mao-Yuan Pai, Mu-En Wu, Yu-Chuan Lin, Min-Hsiung Hung, Wei-Tsung Su, Yi-Chung Chen, et al. Deep-learning-based defective bean inspection with gan-structured automated labeled data augmentation in coffee industry. *Applied Sciences*, 9(19):4166, 2019.

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[12] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[13] Defense Innovation Unit Experimental (DIUx) and the National Geospatial-Intelligence Agency (NGA). xview 2018 detection challenge dataset. `http://xviewdataset.org/`, 2018. Accessed: 2023-02-18.

[14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[15] Mihai Lorin Dimoiu, Dan Popescu, and Loretta Ichim. Improved conditional gan for aerial image segmentation. In *2021 IEEE AFRICON*, pages 1–6. IEEE, 2021.

[16] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[17] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[19] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[20] Joeri R Hermans, Gerasimos Spanakis, and Rico Möckel. Accumulated gradient normalization. In *Asian Conference on Machine Learning*, pages 439–454. PMLR, 2017.

[21] Cherifi Imane. Yolo v5 model architecture [explained]. `https://iq.opengenus.org/yolov5/`. Accessed: 2023-03-25.

[22] jalola. improved-wgan-pytorch. `https://ggithub.com/jalola/improved-wgan-pytorch`, 2022.

[23] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, tkianai, yxNONG, Piotr

Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[25] Daojun Liang, Feng Yang, Tian Zhang, and Peter Yang. Understanding mixup training methods. *IEEE access*, 6:58774–58783, 2018.

[26] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[28] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[29] Sarah E Nelms, Joanna Alfaro-Shigueto, John PY Arnould, Isabel C Avila, Susan Bengtson Nash, Elizabeth Campbell, Matt ID Carter, Timothy Collins, Rohan JC Currey, Camila Domit, et al. Marine mammal conservation: over the horizon. *Endangered Species Research*, 44:291–325, 2021.

[30] David Orive, G Sorrosal, Cruz Enrique Borges, Cristina Martin, and A Alonso-Vicario. Evolutionary algorithms for hyperparameter tuning on neural networks models. In *Proceedings of the 26th european modeling & simulation symposium. Burdeos, France*, pages 402–409, 2014.

[31] David MW Powers. What the f-measure doesn't measure: Features, flaws, fallacies and fixes. *arXiv preprint arXiv:1503.06410*, 2015.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[33] Deval Shah. Mean average precision (map) explained: Everything you need to know. `https://www.v7labs.com/blog/mean-average-precision`. Accessed: 2023-03-02.

[34] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[35] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[36] Krishna Kumar Singh, Hao Yu, Aron Sarmasi, Gautam Pradeep, and Yong Jae Lee. Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. *arXiv preprint arXiv:1811.02545*, 2018.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[38] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904. IEEE, 2019.

[39] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1262–1270. IEEE, 2019.

[40] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[41] Mingxing Tan and Quoc V Le. Efficientnet: Improving accuracy and efficiency through automl and model scaling. *arXiv preprint arXiv:1905.11946*, 2019.

[42] V Terrance and W Taylor Graham. Dataset augmentation in feature space. In *Proceedings of the international conference on machine learning (ICML), workshop track*, volume 3, 2017.

[43] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. *arXiv preprint arXiv:2302.07944*, 2023.

[44] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[45] Ross Wightman. timm: A lean, powerful, and modular PyTorch deep learning library of computer vision models. `https://pypi.org/project/timm`, 2020. Accessed: 2023-03-01.

[46] Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, and Yunfei Liu. A forest fire detection system based on ensemble learning. *Forests*, 12(2):217, 2021.

[47] Haodong Zhang, Zuzhi Chen, Chaoqun Zhang, Juntong Xi, and Xinyi Le. Weld defect detection based on deep learning method. In *2019 IEEE 15th international conference on automation science and engineering (CASE)*, pages 1574–1579. IEEE, 2019.

[48] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Erklärung der Urheberschaft

Hiermit versichere ich an Eides statt, dass ich die vorliegende Bachelor thesis im Studiengang B.Sc. Wirtschaftssinformatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg
30.03.2023

Ort, Datum

Unterschrift

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Bachelor thesis in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum Unterschrift