Imperial College London

Imperial College London

Department of Computing

Generative-Based Data Augmentation Methods for Fine-Grained Visual Classification

Author:
Siyun Hu

Supervisor:
Dr. Pancham Shukla

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing of Imperial College London

April 1, 2024

Abstract

Machine learning is a rapidly expanding area that enables the computer to learn from data, thus improving human lives. Deep learning, a popular branch of machine learning, has shown people its ability to solve many machine learning tasks. Providing sufficient and reliable data is a key to the success of deep learning as the training of deep learning models requires a vast amount of data. However, data is typically scarce for some machine learning tasks. A set of techniques named "data augmentation" has been developed to combat the scarcity of data by artificially creating new samples from existing data.

In this thesis, we tackle the problem of not having an adequate amount of data for a sub-field of classification task named fined-grained visual classification(FGVC). We used a new dataset which has not been previously studied in the data augmentation field. We mainly investigate how generative methods, a branch of data augmentation technique, would assist this classification task.

We begin our work by learning the fine-grained classification task and exploring a wide range of data augmentation techniques. A more detailed explanation is provided for generative methods since they are our primary focus. Subsequently, we proposed a new generative method based on VAE/GAN framework that generates new realistic instances by learning the distribution of real data, successfully improving the classification accuracy. Finally, the last part of the thesis will illustrate experiment and evaluation of different generative models and some other popular data augmentation techniques on our new dataset.

Acknowledgements

First, I want to thank my supervisor Dr. Pancham Shukla, and external collators Dr. Walambe Rahee and Dr. Ketan Kotecha. They arranged meetings with me out of their busy schedules and provided me with valuable guidance for this project. They helped me to solve the difficulties I encountered during this project. I wish the best for them.

I also want to thank Dr. Robert Craven, my second marker, who carefully read my background report and offered suggestions for how to improve it. I wish the best for him.

I would also like to thank Wenjia Wang, a friend and classmate of mine. She always supported me when I struggled and felt low.

I would like to thank my cats: Snow, Django and Brewer. They are my most incredible friends who have accompanied the most of my life during college. They have provided me with emotional support and unconditional love. They have witnessed every important moment of mine in recent years, and I hope they will be with me always.

Also, I want to thank my parents, who provided me with the resources to finish my undergraduate and master's degree. Without their support, I could never be who I am today, and this thesis would not be presented. I hope they live a happy and long life.

Finally, I want to give my greatest thank to my loved boyfriend, Jianzhong (Max) You. He gave me the motivation to devote myself to my project. His hard-working and excellent academic performance encourages me. Also, his accompany is indispensable to me. He taught me to persevere in difficulties and gave me encouragement when I felt less motivated. Best wishes to our two.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivations

Machine learning has become an inseparable part of our daily life. A huge range of applications have already applied machine learning methods, and lots of research is conducted in this field as well. A very popular subset of Machine learning named Deep learning which aims to simulate human's brain activity has been extremely popular. Drive-less cars, personalized recommendation systems, machine translation, question-answering robots and many other applications are not a dream anymore because of machine learning and deep learning. Deep learning has also been heavily applied in computer vision tasks, where computers are trained to learn and understand the visual world. Image classification is one of the domains of computer vision. In this domain, images are labelled by a trained network, namely a classifier.

While it is generally recognised that a more complex deep learning model usually leads to a better outcome, there are many inherent problems. Overfitting is a very common issue for complex networks, where the network is trained to fit the training data overly good by memorising data. In most cases, the distributions of training data and testing data are different, so overfitting is likely to lead to a reduction in the model's true performance. Dropout[37] and batch normalisation[45] are two simple yet powerful regularization methods to reduce overfitting. Many other methods have been developed to improve the generalisability of the model as well. However, to address the root of overfitting, we need enormous data to make training and test data distributions similar. Manually collecting enormous amounts of data is extremely costly and impractical. The focus of this project, data augmentation, is another helpful method which can address the leading cause of the overfitting problem. Data augmentation refers to the techniques which artificially inflate the dataset by data warping or oversampling[47]. The class imbalance problem can also be alleviated through this technique

by augmenting the minority class. In addition, data augmentation can improve the robustness of models. To summarise, data augmentation is an indispensable tool in the field of machine learning.

Fine-grained visual classification(FGVC) is a subfield of image classification. FGVC is about distinguishing images of different sub-categories within the same category[60], for instance, the species of cats and the type of cars. FGVC is a challenging task due to the three main reasons[21]:

- **High intra-class variances**: Images containing objects with the same category can be very different from each other due to the problem of a variety of viewpoints, lighting and poses.
- **Low inter-class variances**: Images containing objects from different categories can be very similar to each other since those objects are all belonging to the same family. The differences between the characteristics of those objects are usually minor.
- **Limited dataset**: It requires domain knowledge to annotate a fine-grained dataset which is very costly, making the dataset scarce. Also, class imbalance problem occurs in this task very frequently.

Using the data augmentation technique in FGVC is very challenging yet useful. Suppose some data augmentation method can extract and generate the discriminative parts of objects while ensuring the variety of the augmented images. In that case, it is believed that this method can improve the classification results as it somewhat solves the three challenges mentioned above.

Many existing data augmentation techniques for FGVC focus on extracting discriminative regions and performing basic image transformations on those areas[21, 8, 15, 30, 23] while using generative models for this task is not very popular. Also, generative methods are challenging in themself. First, there are many probabilistic theories behind intuition, so learning probability is necessary. Second, it is a challenging task to design, implement and train those models to generate images of acceptable quality. Therefore, we want to focus on employing generative-based data augmentation methods for FGVC.

Figure 1.1: Classification Task

(a) Chinchilla

(b) British Short Hair

Figure 1.2: Fine-grained classification for cats

1.2 Objectives

This project aims to study state-of-the-art data augmentation techniques for classification tasks. Also, we try to apply augmentation methods to a new dataset. Thirdly, we want to design our own or improve the existing generative approach for our chosen task and dataset. Finally, we want to evaluate common generative models and compare them with other data augmentation methods for our selected dataset.

1.3 Contributions

Here is a summary of the findings and contributions to this project.

- We survey commonly used data augmentation techniques on a new dataset.

- We proposed a generative method based on VAE/GAN framework for the chosen dataset to produce high-quality images.
- We showed that using a pre-trained classifier in the training of our model can make our GAN produce images which help improve classification accuracy.
- We showed that using spectral normalisation can make our GAN produce images which helps improve classification accuracy.
- We successfully increased the baseline classification accuracy with our proposed method.

Chapter 2

Background and Literature Review

This chapter will start by giving a short summary of how deep learning evolves for fine-grained visual classification. Then an introduction of a wide range of modern data augmentation techniques along with how they help the classification task will be given. Although some approaches are not appropriate for our dataset, a brief explanation of those methods will nonetheless be provided in order to provide the groundwork for the data augmentation field. We will go into more detail on generative approaches since they are the most challenging augmentation methods and also the primary focus of our study.

2.1 Fine-Grained Visual Classification

Various methods have been designed for solving FGVC tasks. Deep Convolutional Neural Networks—(CNN) are the basic models for classification tasks. However, they only achieve moderate results on FGVC tasks because they lack the ability to learn the subtle differences of objects[21]. Many methods have been developed to overcome the insufficiency of basic CNNs. As the name suggested, Part R-CNN learns the object's parts and geometric constraints and predicts from a pose-normalized representation[59]. However, methods like Part R-CNN incurs additional labelling, which is costly, so studies on methods that only require image-level annotations are conducted. Bilinear pooling[31], computing the outer product of feature maps from two streams CNNs to capture the discriminative parts of the image, has been developed since and then widely used in FGVC tasks. Recurrent Attention CNN[11] was then proposed, which solves FGVC tasks by recursively learning the region-based representation of feature and the representative region attention. Yang et al. presented a self-supervision mechanism to locate the discriminative regions accurately[54]. An attention convolutional binary neural tree architecture is proposed by Zhuang et al.[25] to address the high intra-class variances and low inter-class variances problem.

2.2 Preliminary

2.2.1 CAM

CAM[62] stands for class activation maps, a type of saliency map. The CAM of a category represents the discriminative regions of the input image that a classifier uses for the classification tasks. To get CAM, the network should have a series of convolutional layers followed by one fully connected layer. Global average pooling is performed on the last convolutional layer. Then in the output layer of a classifier, a weighted sum with coefficient $w_1, w_2...w_k$ of the feature maps will be calculated to get the final prediction, and those weights will be reused to generate the CAM. The CAM for a particular location is then the weighted sum of the last feature maps values before global pooling at this location using the same weights $w_1, w_2...w_k$. This technique has been applied in some data augmentation models such as F-CGAN[12]. In our original hypothesis, CAM would be useful for our chosen task, but in the later chapter, examples will be provided to illustrate the limitation of CAM.

2.2.2 Self-Attention Mechanism

Self-attention computes a weighted sum of the feature representation at different positions of input to get the response at a position in the same input, allowing networks to learn global dependencies without considering the distance between those positions[52].

2.3 Modern Data Augmentation Methods

2.3.1 Basic Image Manipulations

This section illustrates augmentation methods based on common manipulations which directly act on images, including geometric transformations, colour space transformations, cropping and mixing. This kind of data augmentation is usually easy to implement and is highly effective, making it the most popular data augmentation technique. However, the usefulness of this type of method is not guaranteed and is data-dependent[47]. The safety of those methods is about whether them are label-preserving or not. If applying an operation to an image does not change the label of this image, we say this operation is safe[47]. Attention should be paid when applying those methods to ensure the safety. We will evaluate those methods on our dataset.

Geometric Transformations

Geometric transformations refer to the operations which transfer an image's orientation and positions. [35] has shown that geometric transformations only worsen the outcome of the classifier for the COVID-19 dataset, while [46] shows that some combinations of geometric transformations improve the performance. As these augmentation techniques don't change the discriminative properties of the images from our chosen dataset, they will be used and assessed in our research. **Flipping** is one of the geometric transformations which refers to the operation which flips the image in a horizontal or a vertical direction. Research has shown that flipping an image is useful on some datasets like CIFAR-10[46]. In the contrast, one should not use this technique on text recognition datasets since it won't be label-preserving anymore. **Rotation** is the procedure of changing the angle of an image. This method can improve the robustness of the network because it simulates images taken by cameras in different positions. However, there is a risk of information loss as the corners of the original image might be cropped. This operation is also not always label-preserving. **Translation** shifts the image to avoid positional bias of the data. By having the object in different positions, the network is forced to forget the position information of the object but to learn more discriminative features.

Color Space Transformations

There are three channels for most of the image data, and each channel is responsible for an individual RGB (Red, green, blue) value. Colour space transformations manipulate those RGB values to reduce the influences of lighting biases; for instance, one simple fix to a highly dark image is to increase the pixel values by a constant value. This augmentation method has lots of freedom, but careful consideration is required when applying them. Transforming colour pictures into greyscale can fasten the training process, but [6] shows a performance decrease by this operation as colour space transformations could lead to information loss. Also, this type of transformation is not always safe. This strategy is not safe for our dataset, changing the colour of one element could result in label changes as well.

Cropping

Cropping removes the undesired area from an image. **Central cropping** is commonplace in deep learning when input images have different dimensions. **Random cropping** helps with the generalisability of the models because it forces the model to learn the situations when the objects of interest are not fully visible in the images. Cropping can not be applied in our project since all objects are located in the centre of the images.

Random Erasing

Occlusion problem frequently occurs in computer vision tasks. Zhong et al. developed a random erasing data augmentation method to combat the occlusion problem[61]. The idea is to select a random patch on the image and fill this patch with either 0s or 255s, forcing the network to learn from the whole image rather than a region of it.

Mixing Methods

The mixing technique mixes two or more images into a single image. Those methods enforce the network to learn the spatial structure of the object. The models trained with this technique are proven to outperform those without it for certain dataset[36].

The fundamental concept behind **MixUp**[58] and **CutMix**[56] is to create artificial images with labels using linear transformations. The primary difference between MixUp and CutMix is that the former applies the transformation to the image's pixels, whereas the latter removes portions of one image and pastes them into another. Experiments demonstrate that MixUp and CutMix can help big networks avoid memorising fixed augmentations and reduce generalisation errors for the datasets selected[58, 56].

AugMix[17] combines mixing methods and basic image manipulations and uses a consistency loss and slots to the training pipelines. In AugMix, firstly, several images will be generated by performing randomly basic manipulations on the original images, and then the augmented image is a weighted sum of those images. Then in one training epoch of the classifier, two different augmented images will be produced by AugMix for one input image. In addition to the original classification loss of the original image, Jensen-Shannon Divergence Consistency Loss among the posterior of the input image and its two augmented images is added to ensure consistency. Compared to the traditional mixing methods, AugMix successfully lowers absolute corruption error by around 16% on the CIFAR dataset.

MixUp and CutMix mix labels proportionally to the operated area, which turns out to be inaccurate since the actual contribution of the label of each input image depends on the size of the discriminative area on the cropped patch rather than the size of the patch or pixels. To overcome this insufficiency, **SnapMix**[23] is developed, which utilises CAM to calculate Semantic Percent Map (SPM). SPM will then be used to calculate the weighting of labels of the original images. Another novelty is that SnapMix no longer mixes images at symmetric locations but at any locations to increase the diversity of generated images. This mixing

technique outperforms CutMix and MixUp on the CUB dataset.

2.3.2 Feature Space Augmentation

All the above methods work on the image space directly. In this section, techniques applied to the feature space will be discussed. Feature space refers to the lower-dimensional representations in deep layers of a network. [10] opens up an exciting area regarding applying transformations to the feature representations of data. Experiments show that extrapolating in the feature space beats input space data extrapolation for the MNIST dataset. **Synthetic Minority Oversampling Technique(SMOTE)**[4] is a method developed to combat the imbalanced class problem by augmenting the feature space of the minority class. SMOTE inflates the minority class by joining the k nearest neighbours in the feature space, and it performs better than simple oversampling for certain dataset.

2.3.3 Weakly Supervised Data Augmentation

The weakly supervised data augmentation network(WS-DAN)[21] is designed for FGVC. As mentioned above, extracting discriminative features is the key to the FGVC task. WS-DAN applied weak supervision to localise those informative regions by attention maps. One attention map stands for one discriminative part of the object. After locating the discriminative parts, two data augmentation methods are applied to a randomly selected informative location. The first method is attention cropping which crops and zooms the area in the raw images corresponding to the selected attention map, aiming to enlarge the discriminative part to help the classifier learns. The other is attention dropping, which does the opposite of attention cropping, forcing the classifier to learn additional discriminative information. WS-DAN outperforms many state-of-art methods.

SWS-DAN[55] was built upon WS-DAN. The authors Z. Yang et al. believe that randomly selecting one attention map like WS-DAN is less reliable, so they proposed to use a saliency map like CAM instead of attention maps. SWS-DAN selects k informative parts based on CAM instead of choosing one region and augments the area in the raw image corresponding to the minimum box area containing those regions. The augmented images often contain noise, which is another issue with the original attention-dropping technique. A mask is created to remove some of the non-discriminative regions to solve this problem. In addition, SCutMix, a similar mixing method, was used to augment the data. Experiments show that SWS-DAN is superior to WS-DAN.

2.3.4 Meta-learning

Meta-learning is a relatively new concept in deep learning, referring to the idea of optimizing neural networks with neural networks. The downside of this domain is that it has not been heavily tested and can be costly to implement or train. Since we aim to conduct data augmentation to increase classification accuracy, we can apply this concept to our research. We could let our generative model learn how to augment images for the purpose of improving classification accuracy.

Neural augmentation[41] is an example of meta-learning. In neural augmentation, there are two networks: the augmentation network and the classification network. Both networks are trained together. The augmentation network takes two images from the same category as input and returns an augmented image. Then the augmented image, along with the original images, are passed into the classification network. There are two losses: the classification loss at the end of the classification network, and the loss for the augmentation network which measures the similarity between the augmented image and the two input raw images. The total loss is a weighted sum of the two losses, and augmentation is learned along with the classification. The result shows that neural augmentation beats CNN without augmentation. The interesting part is that many augmented images have no visual meaning to humans but still capture the objects' characteristics in the input images, which are meaningful to the computer.

Auto augmentation[7] is a more effective method than neural computation. This technique uses the idea of reinforcement learning, forming a discrete search problem to learn the best set of augmentation policies for a particular task. The policy is a set of geometric transformations. M Geng et al. believe Auto augmentation only leads to a sub-optimal solution. Hence, they change the discrete search space into a continuous one with Augmented Random Search[13].

2.4 Deep Generative Models Background

Deep generative models are invented by combining deep neural networks and generative models. Those models aim to estimate the distributions of sample data and generate new instances that are similar to the original data. This type of technique is the primary focus of our research because it is less typically employed for classification tasks.

2.4.1 VAE Background

Variational autoencoder(VAE) is a network architecture invented by Diederik P. et al. [28]. Since VAE serves as the basis for many additional models, a thorough explanation will be provided.

In VAE, input is encoded into **latent variables** by the **encoder** and then input is reconstructed by those variables through the **decoder**.

Latent Variables

Latent variables refer to the variables that can not be observed but inferred through mathematical models. We denote latent variables as z. A latent variable model assumes that the observable variables result from latent variables. In other words, we may reconstruct the original information by its latent variable through some models.

Encoder/Decoder

The encoder E is a network that wants to find the latent variables from a given input ($E(x) = z$). At the same time, the decoder D takes latent variables as input, aiming to reconstruct the original input ($D(z) = x$). VAE is a latent variable model consisting of an encoder and a decoder.

Intuition

The intuition behind VAE is that we want to fit a network model $p_\theta(x)$ to the distribution $p_data(x)$ of a given dataset $(x_n)_{n=1}^N$.

Divergence

For a set of probability distributions $\mathcal{P}$ on a random variable $\mathcal{X}$, the divergence is a function $\mathcal{D}[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \leftarrow \mathbb{R}$ such that for all P, Q $\in \mathcal{P}$, $\mathcal{D}[P||Q] \geq 0$ and $\mathcal{D}[P||Q] = 0$ if and only if $P = Q$. In simple speaking, divergence measures how close the given two distributions are. Divergence loss is an important component of the loss functions of some generative models.

A divergence named Kullback-Leibler(KL) divergence $KL[\cdot||\cdot]$ is used to measure the closeness of the model distribution $p_\theta(x)$ and the true data distribution $p_{data}(x)$. The equation for KL divergence is:

$$\int p(x) log \frac{p(x)}{q(x)} dx, \ p, q \in \mathcal{P} \tag{2.1}$$

We want to minimize such divergence. Hence, the set of optimal parameters for this network following the intuition is:

$$\theta^* = arg\,min KL[p_{data}(x)||p_\theta(x)] \tag{2.2}$$

Objective Function

Expanding equation 2.2 by the definition of KL and ignoring the constant terms with respect to θ, we have

$$\theta^* = arg\,max \mathbb{E}_{p_{data}(x)}[\log p_\theta(x)] \tag{2.3}$$

Using latent variable model, we get

$$log p_\theta(x) = \int_x p_\theta(x|z)p(z)dz \tag{2.4}$$

where z is the latent variable assumed to sample from normal distribution $N(z; 0, I)$ for simplicity. However, the integral over all the configurations of z is infeasible to be calculated.

Jensen's Inequality

Jensen's inequality states that for a convex function $f : \mathbb{R} \to \mathbb{R}$, $\mathbb{E}_{p(x)}[f(x)] \geq f(\mathbb{E}_{p(x)}[x])$ for any distribution $p(x)$.

Variational Lower bound

A new distribution $q(z)$ such that $q(z) > 0$ is introduced. By Jensen's inequality, we obtained

$$\begin{aligned} \int p_\theta(x|z)p(z)dz &= log \int q(z)\frac{p_\theta(x|z)p(z)}{q(z)}dz \\ &\geq \int q(z) log \frac{p_\theta(x|z)p(z)}{q(z)}dz \\ &= \mathbb{E}_{q(z)}[log p_\theta(x|z)] - KL[q(z)||p(z)] \end{aligned} \tag{2.5}$$

which is our variational lower bound.

Objective after Using Variational Lower Bound

The choice of $q(x)$ is critical as it is related to the tightness of the lower bound which motivates VAE approach. When $q(x)$ is more similar to $p_\theta(z|x)$, the lower bound is more tight. In VAE, $q(x)$ is defined as $q_\phi(z|x)$, where $q_\phi(z|x) = N(z; \mu_\phi(x), diag(\sigma^2_\mu(x)))$, $\mu_\phi(x)$ $\sigma^2_\mu(x)$ are the output of a nerual network $NN_\phi(x)$. We want $q_\phi(z|x)$ to be close as possible

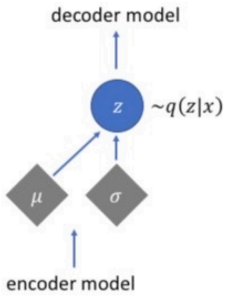


Figure 2.1: After Monte Carlo Estimation

to $p_\theta(z|x)$ since we want a tiger lower bound. The final objective of VAE architecture is then derived:

$$\phi^*, \theta^* = arg\,max\mathbb{E}_{p_{data(x)}}[\mathbb{E}_{q_{\phi(z|x)}}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p(z)]] := L(\phi, \theta). \tag{2.6}$$

Analytic KL between Factorised Gaussians

Since $q_\phi(z|x)$ and $p(z)$ are Gaussian distributions, the later term in 2.6 has an analytic form

$$KL[q_\phi(z|x)||p(z)]] = \frac{1}{2}(||\mu_\phi(x)||_2^2 + ||\sigma_\phi(x)||_2^2 - 2 < \log \mu_\phi(x), 1 > -d) \tag{2.7}$$

The gradient of this analytic form can be easily computed.

Monte Carlo Estimation

Equation 2.6 is still intractable because the calculation of $\mathbb{E}_{q_\phi[\cdot]}$ needs evaluation of all possible z. Using Monte Carlo estimation, we can approximate $\mathbb{E}_{q_{\phi(z|x)}}[\log p_\theta(x|z)]$ as $\log p_\theta(x|z), z \sim q_\phi(z|x)$.

Reparameterisation Trick

In the training of a network, the gradient of a loss function must be calculated if gradient optimisation method is applied. While the gradient of $L(\phi, \theta)$ with respect to θ can be easily

computed after applying the Monte Carlo Estimation, the gradient with respect to ϕ

$$\nabla_\phi L(\phi, \theta) \approx \nabla_\phi \mathbb{E}_{q_{\phi(z|x)}}[\log p_\theta(x|z)] - \nabla_\phi KL[q_\phi(z|x)||p(z)]] \tag{2.8}$$

is still intractable due to the first term. By introducing a new parameter $\epsilon : z = \mu_\phi + \sigma_\phi \odot \epsilon, \epsilon \sim N(\epsilon; 0, I)$, letting $\pi(\epsilon) := N(\epsilon; 0, I)$ and $T_\phi(x, \epsilon) := \mu_\phi + \sigma_\phi \odot \epsilon$, now the gradient of the first term is

$$\mathbb{E}_\pi(\epsilon)[\nabla_\phi z \nabla_z \log p_\theta(x|z)|_{z=T_\phi(x,\epsilon)}] \tag{2.9}$$

Apply Monte Carlo estimation again, we get

$$\mathbb{E}_{\pi(\epsilon)}[\nabla_\phi z \nabla_z \log p_\theta(x|z)|_{z=T_\phi(x,\epsilon)}] \approx \nabla_\phi z \nabla_z \log p_\theta(x|z), \epsilon \sim \pi(\epsilon) \tag{2.10}$$

which is tractable.

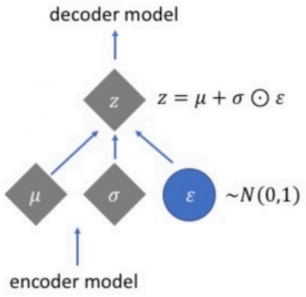


Figure 2.2: After Reparameterisation Trick

However, research shows that the original VAE objective leads to underfitting, so β is introduced:

$$L(\phi, \theta) \approx \mathbb{E}_{q_{\phi(z|x)}}[\log p_\theta(x|z)] - \beta KL[q_\phi(z|x)||p(z)]] \tag{2.11}$$

where β is a coefficient introduced to control the weighting between the reconstruction loss and the KL-loss. The greater the β is, the closer the prior and posterior are getting, meaning the model is better at generating new images from randomly sampled latent vectors. In contrast, the smaller the β is, the better the model reconstructs images from latent variables sampled from real images.

The common drawback of VAE-based methods is that the generated images are usually blurry due to the normal Gaussian distribution assumption of the prior $p(z)$[5].

2.4.2 VAE Variants

CVAE

Vanilla VAE models latent variables directly, but generating a new instance based on category information is needed for many cases. Conditional VAE[48] was thereby invented. There is only a minor difference between VAE and CVAE. CVAE aims to approximate $p_\theta(x|y)$, instead of $p_\theta(x)$, where y is the condition we want to infer about. The objective function of CVAE can be derived following the same steps as the one of VAE, which is

$$\phi^*, \theta^* = arg\,max\mathbb{E}_{p_{data}(x,y)}[\mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|z,y)] - KL[q_\phi(z|x,y)||p(z)]] := L(\phi,\theta). \quad (2.12)$$

2.4.3 GAN Background

While Generative Adversarial Networks(GANs)[14] performs a similar role to VAE, their underlying theories are dissimilar. As many further models are built upon GAN, a detailed explanation will be provided.

Intuition

The idea behind GAN is to make two artificial neural networks, namely **generator** G, which provides a distribution $p_\theta(x)$ to estimate the data distribution $p_{data}(x)$ and **discriminator** D, to play against each other. The discriminator is a binary classifier that determines if the input image is real or fake and serves as a teacher to the generator. The purpose of the discriminator is to discriminate the fake images from the real ones as much as possible. On the other hand, the generator aims to generate synthetic images that can fool the discriminator. Same as VAE, GANs are also latent variables model where $x \sim p_\theta(x) \approx x = G(z), z \sim p(z)$.

Objective

For simplicity, common GANs use a label of ones to indicate real images, while a zeros label represents fake images generated by the generator.

Let θ be the parameters of the generative model and ϕ be the parameters of the discriminator,

then the optimal value of GAN architecture is then defined as:

$$\theta^*, \phi^* = arg \min_{\theta} \max_{\phi} \mathbb{E}_{p_{data(x)}}[\log D_\phi(x)] + \mathbb{E}_{p_\theta(x)}[\log(1 - D_\phi(x))] \tag{2.13}$$

In practice, $\mathbb{E}_{p_\theta(x)}[\log(1 - D_\phi(x))] \approx \log(1 - D_\phi(x)), x \sim p_\theta(x)$ by Monte Carlo estimation.

The Training of GANs

The training of GANs is like a two-player game. We fix one component, train the next, exchange, and repeat. It is widely accepted that GANs are very difficult to train due to their adversarial nature. In the early stage of training, the discriminator usually is too strong since the distribution of real and fake images is very different at this time, so $D_\phi(x) \approx 0$ for $x \sim p_\theta(x)$. Therefore when updating the generator, the gradient $\Delta Loss_{D(x)} \to -\infty$ and the generator could learn nothing. This problem is called **vanishing gradient problem**. Reducing the power of the discriminator seems like a solution to this problem. However, if the generator is too powerful, then **exploding gradient** might occur, leading to another problem called **mode collapse**. Mode collapse occurs when the generator produces a single high-quality image that can fool the discriminator and the generator only keeps making the same result again and again. In other words, GAN would fail to produce diverse images. Alternative loss functions have been designed to alleviate this problem. WGAN[1] used Wasserstein distance to improve the stability of training and mitigate the mode collapse problem.

2.4.4 GAN Variants

DCGAN

DCGAN[42] utilised CNN architecture in GAN. The novelty in DCGAN is that, first, the network has fully convolutional layers. Second, batch normalisation is used for both the discriminator and the generator to stabilise the training. Third, LeakyRuLU is used for all layers of the discriminator, and ReLU activation is utilised for all layers of the generator, with the exception of the last layer. Experiments show that DCGAN is capable of learning good representations of images, but mode collapse sometimes happens when the network is trained for a long time. The most common GANs were built upon DCGAN.

CGAN

Similar to the relation between VAE and CVAE, Conditional GAN[33] tries to estimate $p_\theta(x|y)$ instead of $p_\theta(x)$, where y is the condition. For generating images, we need to supply

both y and z to the generator. The input to the discriminator is now x (the real images and the fake images) and y, while its output is still real or fake. The modified objective is

$$\theta^*, \phi^* = arg \min_{\theta} \max_{\phi} \mathbb{E}_{p_{data(x,y)}}[\log D_\phi(x, y)] + \mathbb{E}_{p_\theta(x|y)}[\log(1 - D_\phi(x, y))] \tag{2.14}$$

ACGAN

ACGAN[38] employs additional information y, which is assumed to be independent of the latent variable z, resulting in a higher quality of generated images and a more stable training process than the vanilla GAN. Now the generated image is $x = G(y, z)$, $y \sim p_y$ while $x = G(z)$ in GAN. The difference between ACGAN and CGAN is that the discriminator in ACGAN uses only x as input and outputs real or fake along with the class information. An additional loss is introduced:

$$L_c = \mathbb{E}_{p_{data(x,y)}}[\log p(y|x)] + \mathbb{E}_{p_\theta(x|y)}[\log p(y|x))] \tag{2.15}$$

The discriminator of ACGAN aims to maximize equation 2.15 + 2.13 while the generator is trained to maximize equation 2.15 - 2.13

ECGAN

Like ACGAN, ECGAN[49] also classifies, but the difference is that ECGAN has an external classifier. The intuition behind this is that the tasks of discrimination and classification are not similar, so it is better to use one network for each task. Also, ECGAN was designed to use GAN to supplement classification, while ACGAN employs a classifier for a better quality of generated images. The objective function for the discriminator and the generator is the same as the vanilla GAN. In addition to it, a new loss for the external classifier is introduced:

$$L_c(x, y, z) = CE(C(x), y) + \lambda CE(C(G(z)), argmax(C(G(z))) > t) \tag{2.16}$$

λ is called the adversarial weight and is used to make the classification loss on the generated images less significant since the fake images only exist to help the classification task. Also, **confidence-based pseudo labeling**[24](The second part of the equation 2.16) is used to avoid using low-quality fake images to train the classifier during the early stage of training.

Self-Attention GAN

Tradition DCGAN is good at generating texture yet lacks the ability to capture geometric structure in the images. This insufficiency is caused by the nature of the convolution since convolution only operates pixels locally. Therefore, long-range dependencies are not

learnt. Self-Attention GAN(SAGAN) was then proposed, being the first GAN to employ self-attention to manage to handle this downside of Convolutional GANs[57]. Two techniques to stabilize the training of GANs are also used in SAGAN, namely spectral normaliz sation[34] and an imbalanced learning rate[18]. One point worth mentioning is that, unlike the original idea of applying spectral normalisation to only the discriminator, Han et al. believe that the generator can also take advantage of this normalisation technique. Incorporated with self-attention, SAGAN can generate more globally coherent images with fine details.

F-CGAN

F-CGAN[12] was developed based on CGAN with a fine-grained feature preserver and a multi-task classification model. The most crucial novelty in F-CGAN is the feature preserver, which uses a fine-tuned pre-trained classification network to extract the feature maps. A saliency map is also utilised to remove redundant information from the feature maps. Then the feature maps are inputted to the generator to enable it better learn and generate the discriminative information. In addition, a multi-task classifier is used. Unlike CGAN, where the discriminator outputs the label condition and discrimination result together, the multi-task classifier has two output layers, one for each task.

2.4.5 Combinations of VAE and GAN

CVAE-GAN

CVAE-GAN[2] is the combination of CAVE and GAN that supports a great range of applications. The idea is to use the encoder network from VAE to get a latent representation of the input and then use the generator from GAN to reconstruct the image from the latent vector. Also, a classifier is used along with the discriminator. The intuition behind this is to combine the strength of VAE and GAN, as explained in chapter two. Novel objectives for the generator are introduced:

$$L_{GD} = \frac{1}{2}||\mathbb{E}_{x\sim P_r}[f_D(x)] - \mathbb{E}_{z\sim P_z}[f_D(G(z))]||_2^2 \tag{2.17}$$

$$L_{GC} = \sum_c \frac{1}{2}||\mathbb{E}_{x\sim P_r}[f_C(x)] - \mathbb{E}_{z\sim P_z}[f_C(G(z,c))]||_2^2 \tag{2.18}$$

$$L_G = \frac{1}{2}(||x - x^{'}||_2^2 + ||f_D(x) - f_D(x^{'})||_2^2 + ||f_C(x) - f_C(x^{'})||_2^2) \tag{2.19}$$

$f_D(x)$ and $f_C(x)$ are the feature maps of the discriminator and classifier respectively. Also, like VAE, KL-divergence is used to reduce the distance between the prior $P(z)$ and $q_\phi(z|x,c)$. The final objective is $L = L_D + L_C + \lambda_1 L_K L + \lambda_2 L_G + \lambda_3 L_{DG} + \lambda_4 L_{DC}$, where $\lambda_1 \sim \lambda_4$ are hyper-parameters. CAVE-GAN framework is proven to be more stable in the training stage than the traditional GAN. Also, compared to WGAN, there is no need to clip the parameters. Compared to valina VAE, CVAEGAN generates better images. However, since CVAE-GAN is still based on CVAE framework, so the quality of the generated image is not satisfying enough.

VCGAN

VCGAN[20] was developed to overcome the drawback of CVAE and CGAN. In CVAE, the generated images are usually blurry, and in CGAN, condition information c is simply assumed to be independent of the hidden latent information z. VCGAN used the manifold hypothesis to assume the latent space has local aggregated properties on conditions. Therefore images can be generated from latent information $q_\phi(z|c)$, instead of $q_\phi(z|c,x)$. The objective function is:

$$- KL(q_\phi(z|c,\varphi)||p_\theta(z))) + \mathbb{E}_{q_\phi(z|c,\varphi)} \log D(g_\theta(z)|x,c) \tag{2.20}$$

Where φ is a Gaussian distributed noise introduced to compensate the disturbance caused by the input x, such that $z \sim q_\phi(z|c,\varphi)$. Experiments show VCGAN outperforms DCGAN, ACGAN, WGAN and CVAE-GAN regarding inception score and FID.

2.5 Summary

This chapter covered a wide range of data augmentation techniques, including meta-learning, weakly supervised data augmentation, deep generative approaches, and traditional image manipulation. Since standard image manipulation is relatively simple to use, generative approaches will be the core area of our research. Additionally, because our objects are in the centre of the images, weakly supervised data augmentation is inappropriate for our dataset. The idea of meta-learning will be taken into consideration when designing our model.

Chapter 3

Dataset Description

3.1 Overview

Our research is conducted on a new dataset named FeathersV1[3] that has rarely been investigated for the data augmentation field. FeathersV1 is the earliest public dataset regarding birds' feathers. Each image in the dataset contains one or several overlapped feathers belonging to the same bird specie. The dataset is introduced with the aim of distinguishing a bird's species based on a single feather, raising interest in a new FGVC task. Classifying feathers is a challenging problem. First, The structures of feathers across different species have no apparent differences (Low inter-class variances). They all have the same four components: calamus, rachis, barbs, and afterfeather, and feathers of two different species can be very similar (See figure 3.2 (a), (b)). Second, there can be significant intraspecific variances (High intra-class variances). Even within the same specie, the shape and colour of a bird's feathers might vary greatly (See figure 3.2 (b), (c)). Also, the image quality varies as they are gathered from different sources[3]. FeathersV1's authors claim that this dataset raises the chance of identifying birds without domain knowledge, making it useful for aviation ornithology. Birdstrikes can have severely damaging consequences. One possible remedy for it is to identify those species which typically cause incidents. Currently, bird classification requires expertise and laboratory assistance, making the task pricey, so visual bird recognition is still being used in many airports[3].

Figure 3.1 gives an visulization of our dataset.

Figure 3.1: Visulization of feather dataset

(a) Example Feather of Alpine Swift

(b) Example Feather of Accipiter Nisus

(c) Example Feather of Accipiter Nisus

Figure 3.2: Examples Feathers

3.2 Data Composition and Data Selection

There are 28,272 photos of feathers from 595 different bird species in FeatherV1. The dataset is organized into three subsets: all species, top-100 species and top-50 species. As implied

by the name, the all-species subset includes all the images from the dataset. The top-100 subset is made up of 14,941 images from 100 bird species, and the top-50 subset consists of 10,314 images from 50 classes. Due to the limited computing power and model capacity, we will only use the top-50 set in this project. Another reason behind it is that this set achieves the lowest accuracy among the three (figure 3.3), giving a larger improvement space. The images in the top-50 subset are not equally distributed. The most observed class has 498 images, while the least one has 94 images. The authors have further split the dataset into training and testing subsets, and we will use the same splitting strategy for consistency.

3.3 Data Preprocessing

Our data preprocessing is simple due to the nature of our project. No data augmentation is necessary at this stage. The only preprocess we conducted was to rotate images to ensure they were vertical and resize them to the size of 240×40. In addition, we perform one-hot labelling to change the labels of the data.

Subset	Model	Sparse Categorical Crossentropy	Sparse Top 1 Categorical Accuracy	Sparse Top 5 Categorical Accuracy
Top-50	DenseNet121	1,4597	0,6394	0,8871
	DenseNet169	1,3592	0,6684	0,9186
	DenseNet201	1,9363	0,5700	0,8740
Top-100	DenseNet121	0,6771	0,7989	0,9709
	DenseNet169	0,7131	0,7979	0,9695
	DenseNet201	1,0256	0,7266	0,9491
All	DenseNet121	0,8549	0,7642	0,9482
	DenseNet169	1,0586	0,7181	0,9360
	DenseNet201	0,7689	0,7978	0,9586

Figure 3.3: Baseline accuracy [3]

Chapter 4

Proposed Methodology

In this chapter, we will cover our proposed methodology. The model structure, and the techniques used in this project will be discussed at the beginning. The objective function and the training settings are further explained in the last part of the chapter. The implementation for this project can be found on this github repository.

4.1 Network Architecture

Our model uses a similar structure to that of CVAEGAN[2] and other models[53, 16] that explore the potential of combining VAE and GAN. We employ this kind of architecture because this combination has outperformed CVAE and CGAN in terms of discriminability, diversity and realism according to [2, 53]. Our initial model selection experiment also verified this, CVAE produced very blurry feather images while the mode collapse problem was observed in CGAN. Our model consists of four sub-networks, namely the encoder, the classifier, the discriminator and the decoder, and all four parts are trained simultaneously. Figure 4.1 gives a clear idea about how those parts are connected.

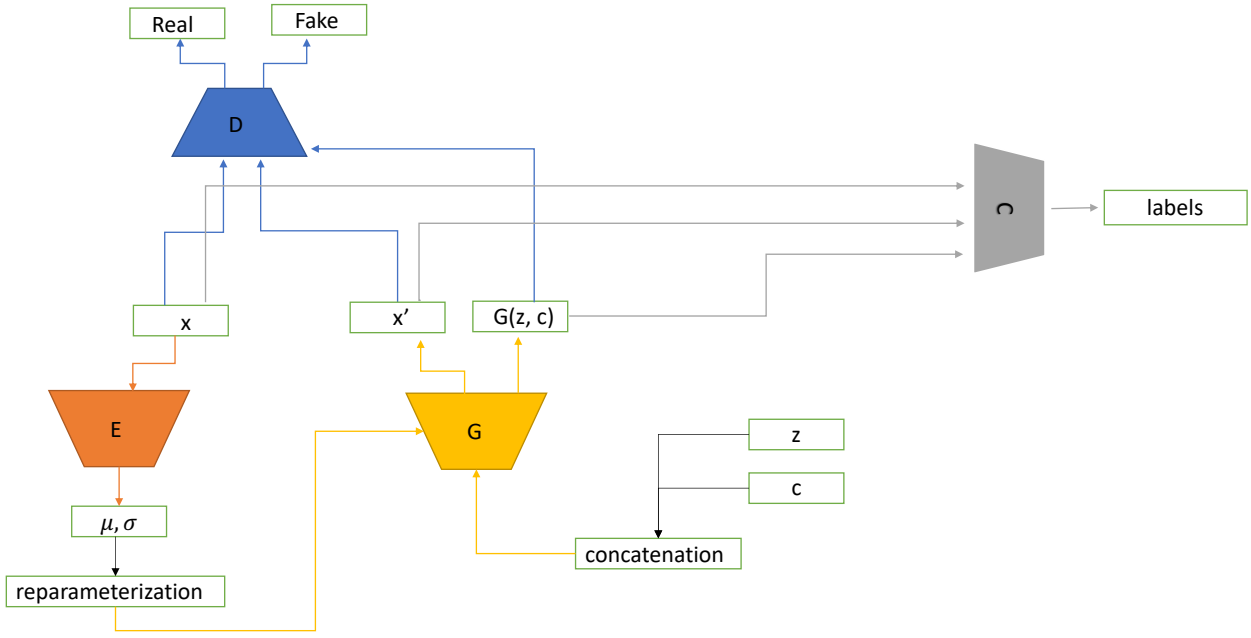


Figure 4.1: The structure of our model

The size of the latent variable z we chose was 150×1, since during our preliminary research, we found that this dimension is enough for our chosen dataset. For simplicity, we assume the prior $P(z)$ is normal distribution i.e. $z \sim N(z; 0, I)$. Also, our model uses fully convolutional layers, which could make the training more stable, as suggested in DCGAN[42].

4.1.1 Encoder

In our model, we assume latent information is independent of class information. Although [20] argued that this independent assumption is naive, we still think it is better appropriate for our chosen dataset because, as explained in the dataset chapter, the feathers of the same specie can vary significantly due to some species' independent reasons[3]. There is no optimum technique to model latent information; in our opinion, it depends on the dataset. Therefore, our encoder tries to learn a distribution $P(z|x)$, independent of the class information, which maps the input x to a hidden representation z. Instead of designing an encoder by ourselves, we use a pre-trained GoogleNet[50], inspired by CVAEGAN[2]. GoogleNet's output layer has been altered to become two parallel, fully connected layers, each of which produces output with the size of 150×1, which is the same as our chosen latent dimension. Two outputs are, respectively, the mean and variance of the corresponding latent variables approximated from the input image. The modification is needed due to the reparameterisation trick introduced

in the background chapter.

4.1.2 Decoder

The decoder aims to convert the hidden information to feather images. Because of the independent assumption introduced above, our decoder network starts from a fully connected layer which takes the concatenation of latent variables and condition information as input. The following is a series of decoder blocks, where each decoder block consists of an upsampling layer, a convolutional layer, a batch normalisation layer then a ReLU activation layer. The output size of the decoder is $3\times$ 240 $\times$ 40, the same as the input image dimension. Despite the fact that many decoder networks use a deconvolutional layer rather than an upsampling layer followed by a convolutional layer, our preliminary research suggested that this would produce unrealistic grid-like images.

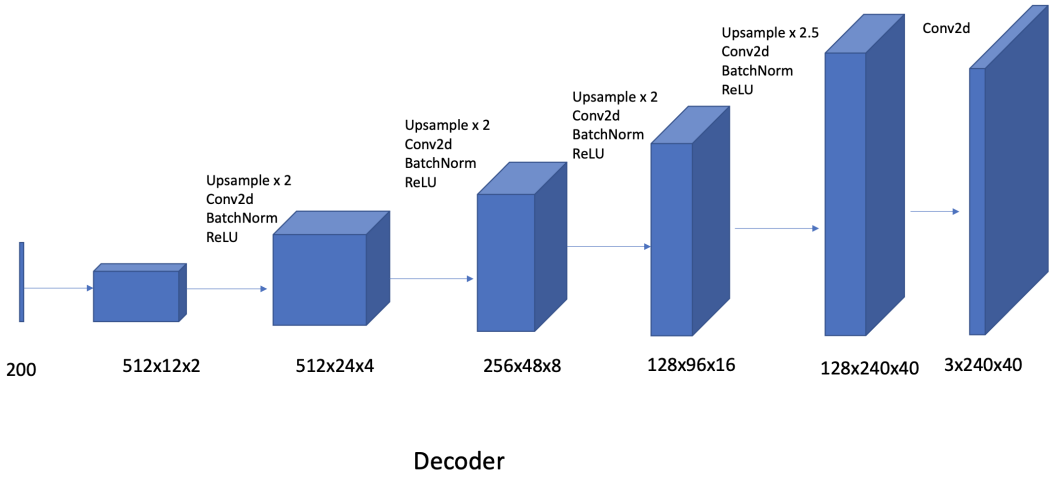


Figure 4.2: The structure of our decoder

4.1.3 Classifier

We incorporated a pre-trained denseNet as our classifier in our model. The choice of using an external classifier was based on EC-GAN[49] and F-CGAN[12]. Also, it can mitigate the vanishing gradient problem since now our generator loss not solely depends on the discriminator, so our generator can still learn from the classification loss on the generated images even when the discriminator is too powerful. In addition, using a single network for both discrimination tasks and fine-grained categorization tasks raises potential issues because, in our opinion, the two tasks are very distinct from one another. Additionally, in the spirit of Neural Augmentation[41], we want our model to learn how to create fictitious images so that

we may use those images to raise classification accuracy, which gives us the motivation to use denseNet - our fine-grained classification task model, as our external classifier. In contrast to conventional GAN-based approaches[38, 29], where the external classifier is trained from the stretch, we chose to utilize knowledge transfer by using a pre-trained classifier. According to our hypothesis, since fine-grained categorization is already challenging, using a pre-trained classifier can better guide the generator without providing excessive erroneous information at the beginning of the training process.

4.1.4 Discriminator

The discriminator aims to distinguish between real and fake images. In our discriminator, there is a series of conv2d layers with spectral normalisation, followed by the LeakyReLU activation function with a negative slope of 0.2. The output layer is a conv2d layer with Sigmoid activation to discriminate. Initially, batch normalisation was utilised as it is a prevalent normalisation technique in GAN-based models. However, by monitoring the training process, we found that the discriminator with batch normalisation consistently beats the generator, leading to convergence failure. We adopt spectral normalisation to combat this problem, and it turns out this technique successfully accelerates our model training and gives better convergence.

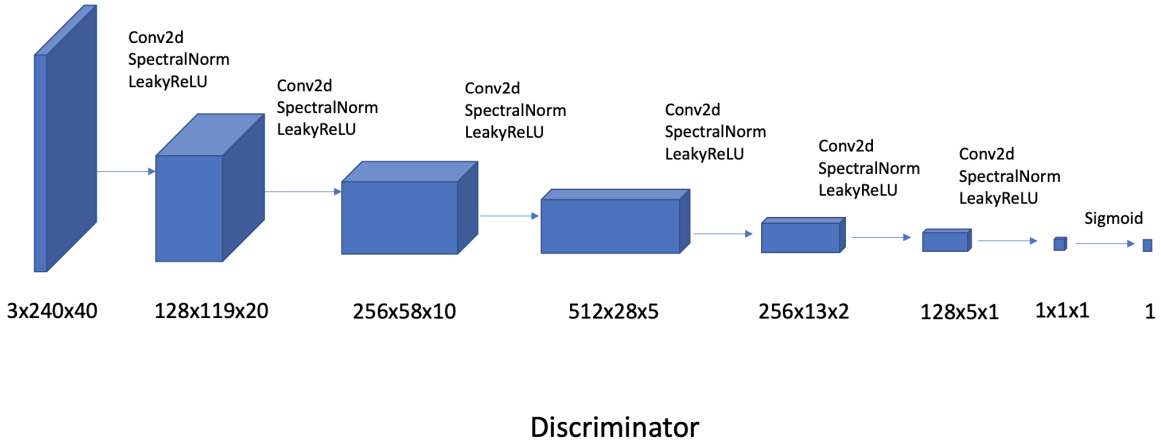


Figure 4.3: The structure of our discriminator

Spectral Normalisation

The training of GAN is a very challenging process due to the adversarial nature of the network. The result shows adding a restriction to the discriminator can stabilize the training. Spectral normalisation[34] is a weight normalisation method applied for discriminator, which

prevents gradient exploding or vanishing. Research[32] has shown that if the activation is (Leaky)ReLU, then

$$||graident||_{Frobenius} \leq \sqrt{number\ of\ layers} \cdot ||input|| \tag{4.1}$$

when using spectral normalisation.

Therefore the gradient of our discriminator is strictly upper bounded, so the exploding gradient problem is mitigated.

4.2 Loss Function

Each component of our model has its own objective, so there are four loss functions in total. Although we have a classifier, this network doesn't compete with other networks. Meanwhile, our encoder, decoder and discriminator are competing with each other, making our training a three-player game.

4.2.1 Encoder

For encoder E, the loss function L_E is:

$$KL[q_\phi(z|x)||p(z)]] - \mathbb{E}_{x,c\sim P_r}[logD(G(E(x),c))] - \lambda\mathbb{E}_{x,c\sim P_r}[||G(E(x),c) - x||_2] \tag{4.2}$$

The first component of equation 4.2 is the same KL divergence loss introduced in VAE, enforcing that the distribution we get from the encoder is close to the prior $P(z)$. The intuition behind the second loss function is that we want our encoder to get a better latent representation $E(x)$, so that the discriminator would think that the reconstructed images $x^{'} = G(E(x), c)$ produced by the decoder G are real. We use a label of 1 to indicate that the discriminator believes that the data are actual, while a label of zeros means fake. The final part is the reconstruction loss from the original VAE loss. This pixel-wise L_2 loss ensures our latent representation captures the characteristics and structures of the input. λ controls the weighting of the reconstruction loss, and we used a value of 0.01 for it. The higher the value, the more blurry the images will be, while the lower the value, the more likely the mode collapse would occur.

4.2.2 Decoder

The loss L_D of decoder is:

$$-\mathbb{E}_{x,c\sim P_r}[logD(G(E(x),c))] - \mathbb{E}_{z\sim P(z),c\sim Pr}[logD(G(z,c))] + \lambda\mathbb{E}_{x,c\sim P_r}[||G(E(x),c) - x||_2] \\ -\mathbb{E}_{x,c\sim P_r}[logP(c|G(E(x),c))] - \mathbb{E}_{z\sim P(z),c\sim P_r}[logP(c|G(z,c))] \tag{4.3}$$

The first component is similar to the second one in equation 4.2; the only difference is that here we assume the encoder has produced an excellent representation, and we want our decoder to generate an image from this representation to fool the discriminator. The second part motivates the decoder to generate realistic images from randomly sampled latent variables and class information. We directly incorporate the reconstruction loss into our loss function, which is the third part. It enforces the decoder to reconstruct the image as close as possible to the original image from given hidden information. Additionally, this pixel-wise L_2 loss serves as a remedy to the mode collapse problem because it makes the decoder generate diverse images (producing a single image only for a class will have a large reconstruction loss). The λ we used is 0.01. We made this decision to balance the reconstruction loss and other components in the generator's loss function since we used MSE loss to compute this reconstruction loss, so it would be relatively large before being balanced. The latter two components are the classification loss on reconstructed and generated images, which will aid the decoder in maintaining the fine-grained features by learning the information the classifier used to predict.

4.2.3 Classifier

The loss L_C of the classifier is:

$$-\mathbb{E}_{x,c\sim P_r}[logP(c|x))] \tag{4.4}$$

L_C simply trains the classifier to recognize the classes of the real images. It is not related to other sub-networks in the model since the ultimate goal of our model is to boost classification accuracy, so it must be left unchanged.

4.2.4 Discriminator

The loss L_d of the classifier is:

$$-\mathbb{E}_{x,c\sim P_r}[logD(x))] - \mathbb{E}_{x,c\sim P_r}[1 - logD(G(E(x),c))] - \mathbb{E}_{z\sim P(z),c\sim P_r}[1 - logD(G(z,c))] \tag{4.5}$$

The goal for the discriminator is to differentiate those synthetic images from the real, so we want our discriminator to indicate that the dataset's data are real while the reconstructed

images, as well as the images produced from randomly selected z and c, are fake.

4.3 Training Pipeline

Algorithm 1 The training pipeline of our proposed method

Require: initialize networks E, G, D, C
  $accuracy\_fake \leftarrow 0$
  while $accuracy\_fake \leq required\_accuracy$ **do**
    Sample $x_r, c_r \sim P_r$
    $accuracy\_real \leftarrow accuracy(C(x_r), c_r)$
    while $accuracy\_real \geq 0.75$ **do**
      $\mu, \sigma \leftarrow E(x_r)$
      $z_r \leftarrow reparametrize(\mu, \sigma)$
      Sample $z \sim P(z)$
      $L_d \leftarrow -log(D(x_r)) - log(1 - D(G(z, c_r))) - log(1 - D(G(z_r, c_r)))$
      $\theta_d \overset{+}{\leftarrow} -\nabla_{\theta_d}(L_d)$
      $L_g \leftarrow -log(P(c_r|G(z_r, c_r))) - log(P(c_r|G(z, c_r))) - log(D(G(z, c_r))) - log(D(G(z_r, c_r))) + \lambda \times ||x_r - G(z_r, c_r)||_2$
      $\theta_g \overset{+}{\leftarrow} -\nabla_{\theta_g}(L_g)$
      $L_e \leftarrow kl\_loss(\mu, \sigma) - log(D(G(z_r, c_r))) + \lambda \times ||x_r - G(z_r, c_r)||_2$
      $\theta_e \overset{+}{\leftarrow} -\nabla_{\theta_e}(L_e)$
    end while
    $L_c \leftarrow -log(P(c_r|x_r))$
    $\theta_c \overset{+}{\leftarrow} -\nabla_{\theta_c}(L_c)$
  end while

To guarantee that our generator is taught by a capable teacher, we only trained the discriminator, generator, and encoder once our classifier had attained a reasonable accuracy. One thing to note is that since the discriminator and generator learn from one another while the classifier does not, we cannot apply the same strategy to the discriminator.

4.4 Training Setup

All of our code is implemented in Pytorch[39], an open-source library for machine learning. We experiment with our code on Colab pro+, where the GPUs are assigned at the discretion of Colab's algorithm.

Parameter Initialization

As suggested in DCGAN[42], we employ 0.0002 as our learning rate for the discriminator. For the encoder, we also use the same learning rate. We employ an imbalanced learning rate for our decoder. A learning rate of 0.0004 is utilised for our decoder since we found that sometimes the generator becomes less potent than the discriminator if they use the same value, which might result in convergence failure. Our classifier uses 0.001 as our learning rate since it is the suggested value for fine-tuning pre-trained DenseNet on Pytorch's official website. Regarding the optimiser, we use a variant of the SGD optimizer[43] named Adam[27] for discriminator, generator and encoder with betas = [0.5, 0.999]. The reason we chose Adam is that it converges faster with fewer parameters to tune. SGD optimiser with momentum 0.9 is applied for our classifier since it is the default optimiser on DenseNet's official web page. We apply weight initiation for our decoder and encoder because GANs are sensitive to initialisation. During our research, we found that two different initialisation settings can have completely different results: in one case, the decoder can generate meaningful images, while in the other case, only noise images can be generated.

Training Validation Split

We performed a further train/valid split on the original training dataset given in the dataset. The package we used is sklearn[40]. We selected 70% of our training set to train our model and used the remanding 30% for validation. Usually, there is no need to validate in GAN's training. However, since our ultimate goal is to apply GAN for the classification task, we still need a validation set to measure how well the generated images from our model can boost the classification accuracy.

Bandwidth

Our batch size is 64, the maximum value that won't cause CUDA out-of-memory issues.

Chapter 5

Experiment and Evaluation

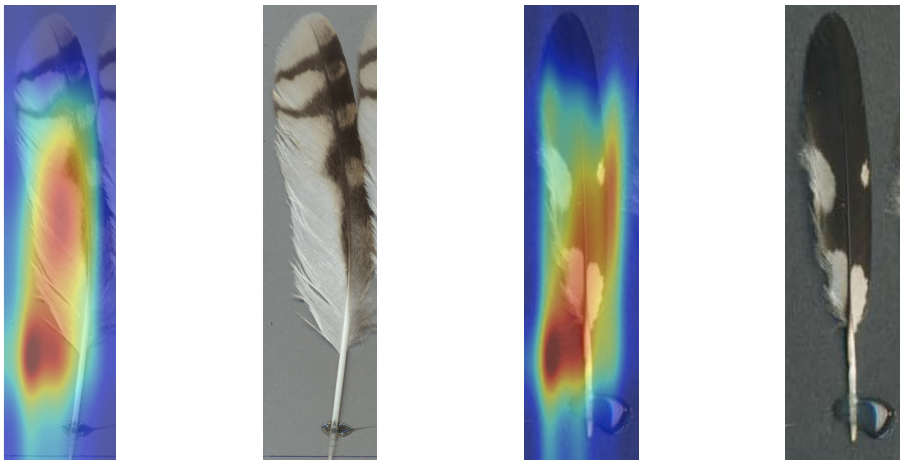
Experiment settings, as well as the results made in our research, will be presented in this chapter. We will compare and evaluate modern data augmentation techniques, including traditional data augmentation methods, general deep learning augmentation techniques and our proposed method.

5.1 Discussions

5.1.1 CAM

Our investigation began by determining how useful CAM was for our dataset since our initial guess was that CAM could be a great tool to preserve fine-grained details. The way we measure CAM is by visual inspection because there is no fine-grained ground truth.

We used an advanced CAM called LayerCAM[26] which was extracted from a trained DenseNet[22] with 0.9062 validation accuracy. However, although the accuracy is high enough, our experiment shows that CAM still can not accurately capture critical information, meaning that the model mostly learned from the spuriously correlated features. For instance, Figure 5.1 (a) (b) shows that the most significant characteristic, the horizontal stripes of the feature of Asio flammeus, is not highlighted, whereas the white region that is highlighted is typical of other feature classes. Also Figure 5.1 (c) (d) shows that unwanted background is highlighted. Therefore, we decided not to use CAM in our model to avoid confusion caused by non-discriminative features.



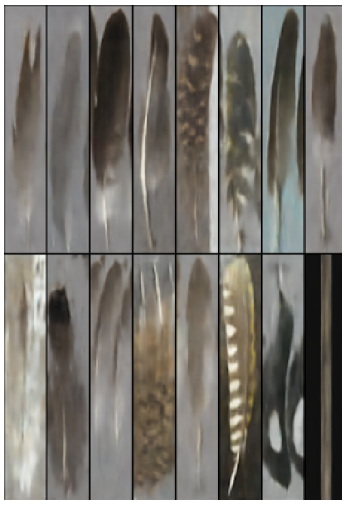
(a) Example of Asio flammeus Feather (b) CAM visualisation of Asio flammeus (c) Example of Great spotted woodpecker Feather (d) CAM visualisation of Great spotted woodpecker

Figure 5.1: CAM visualisation result

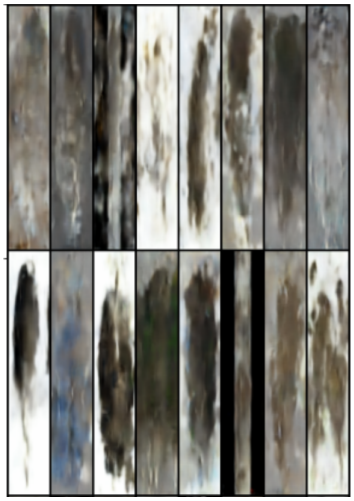
5.1.2 Failure Models

We also wished to evaluate the CVAE, CGAN, and CVAEGAN generating models. The models have been implemented, but we were unable to produce images that qualified as satisfactory.

CVAE failed because no matter how many training epochs we run, the KL- loss does not go down, even after increasing the weighting of the KL-loss(β) to five times the reconstruction loss. The reconstructed images look great, but the model failed to generate new images, as shown in figure 5.2. For CGAN, only noisy images are produced.



(a) Images reconstructed by CVAE trained for 3420 epochs



(b) Images generated by CVAE trained for 3420 epochs

Figure 5.2: Result of CVAE trained for 3420 epochs

We implemented CVAE-GAN exactly according to the original paper[2]. However, this model is extremely hard to train due to the computation of the loss function. We spent over two days only training this model for about 500 epochs. Also, as described in the paper introducing this method, 2M iterations were trained. Although tiny improvement can still be observed for each epoch, considering our limited computational resources, we only are able to provide partially trained CVAE-GAN for our dataset.

Figure 5.3: Images generated by CVAE-GAN trained by 500 epochs

5.2 Evaluation Metrics

Two evaluation metrics which we believe suit our task will be introduced in this section.

5.2.1 Fréchet Inception Distance (FID)

FID[19] is a current standard metric for evaluating the quality of the resulting images of generative models. The idea behind FID is to compare the distribution between the generated and the real images used to train the generative models. To get the FID score, first, the deepest feature layer of Inception V3[51] will be used to calculate the mean μ and standard deviation Σ for the real and generated images separately. Then we apply the FID equation to get the FID score:

$$||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{(1/2)}) \tag{5.1}$$

where r stands for real images, g is generated images and Tr is trace. One benefit of this measure is that FID evaluates how close the created image dataset is to the ground truth images dataset rather than just how realistic the generated image is, making it more reasonable. However, the drawback of FID is that it can not reflect the existence of mode collapse; hence we will apply manual inception to examine if it is the case. One thing to be noted is that the FID score can only be applied to generative models.

The implementation of FID is based on repository [44].

5.2.2 Classification Accuracy

Classification accuracy is our primary measurement metric because our research is regarding applying data augmentation for the classification task. The accuracy of both the validation set and test set will be given.

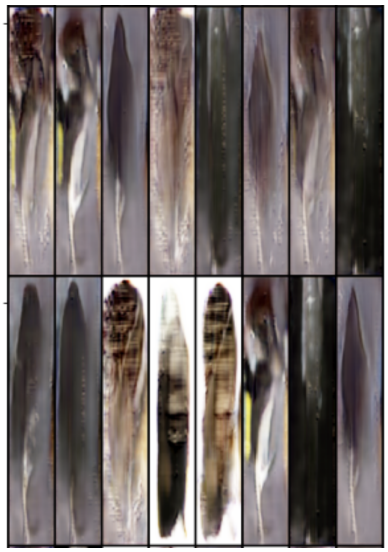
5.3 Experiment Setting + Visualisation

In this part, experiment settings for various methodologies will be covered. Especially, there is no early stopping criterion for generative methods, so we will manually inspect each method to determine which training epoch is optimum. In addition, for all generative models, batch size 64 will be used. To further ensure fairness, we maintained the generator's structure and the number of parameters constant across all approaches. In addition, the learning rate and optimiser are the same as the ones to train our proposed method. Since there are two

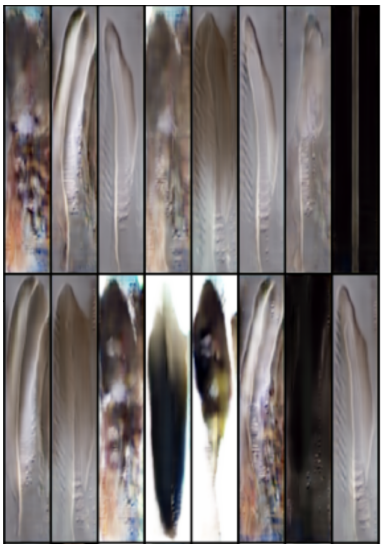
classifiers and each one serves an entirely different purpose in our research, to avoid confusion, we refer to the classifier used to train the conditional generative models as the "training classifier" and the classifier used to assess how our faked images affect the classification accuracy for our dataset as the "evaluation classifier".

5.3.1 Traditional GAN + Pseudo Labelling

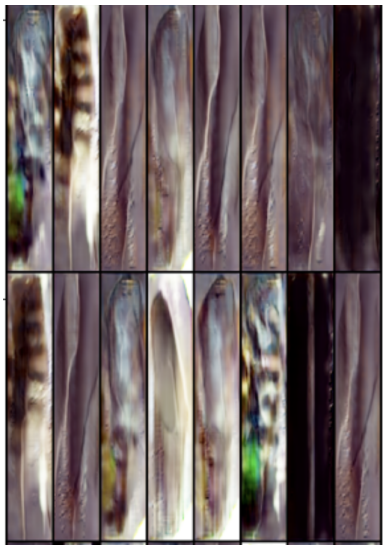
We implemented traditional GAN by DCGAN structure. One problem is that this type of GAN simply generates images from randomly sampled latent variables, regardless of what the classes are. In order to incorporate class conditions, we applied a confidence-based pseudo labelling scheme which was introduced in EC-GAN. Figure 5.4 gives an visualisation of the generated examples. We can see that 200 epochs achieved the best result, so we will evaluate the traditional GAN model trained by this number of epochs.



(a) 150 epochs



(b) 200 epochs

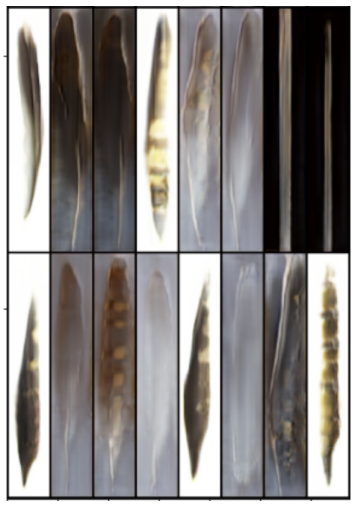


(c) 250 epochs

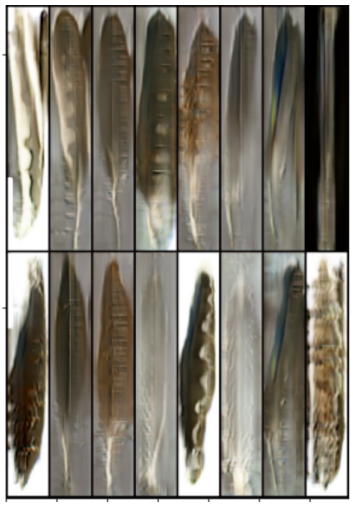
Figure 5.4: Examples of the generated images by traditional GAN trained by 150, 200 and 250 epochs respectively

5.3.2 Self-Attention GAN + Pseudo Labelling

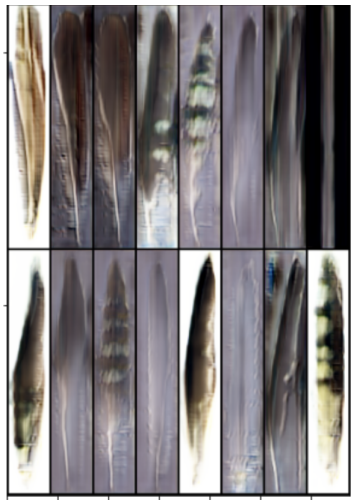
Similar as traditional GAN, confidence-based pseudo labeling scheme is utilised here.



(a) 80 epochs



(b) 86 epochs



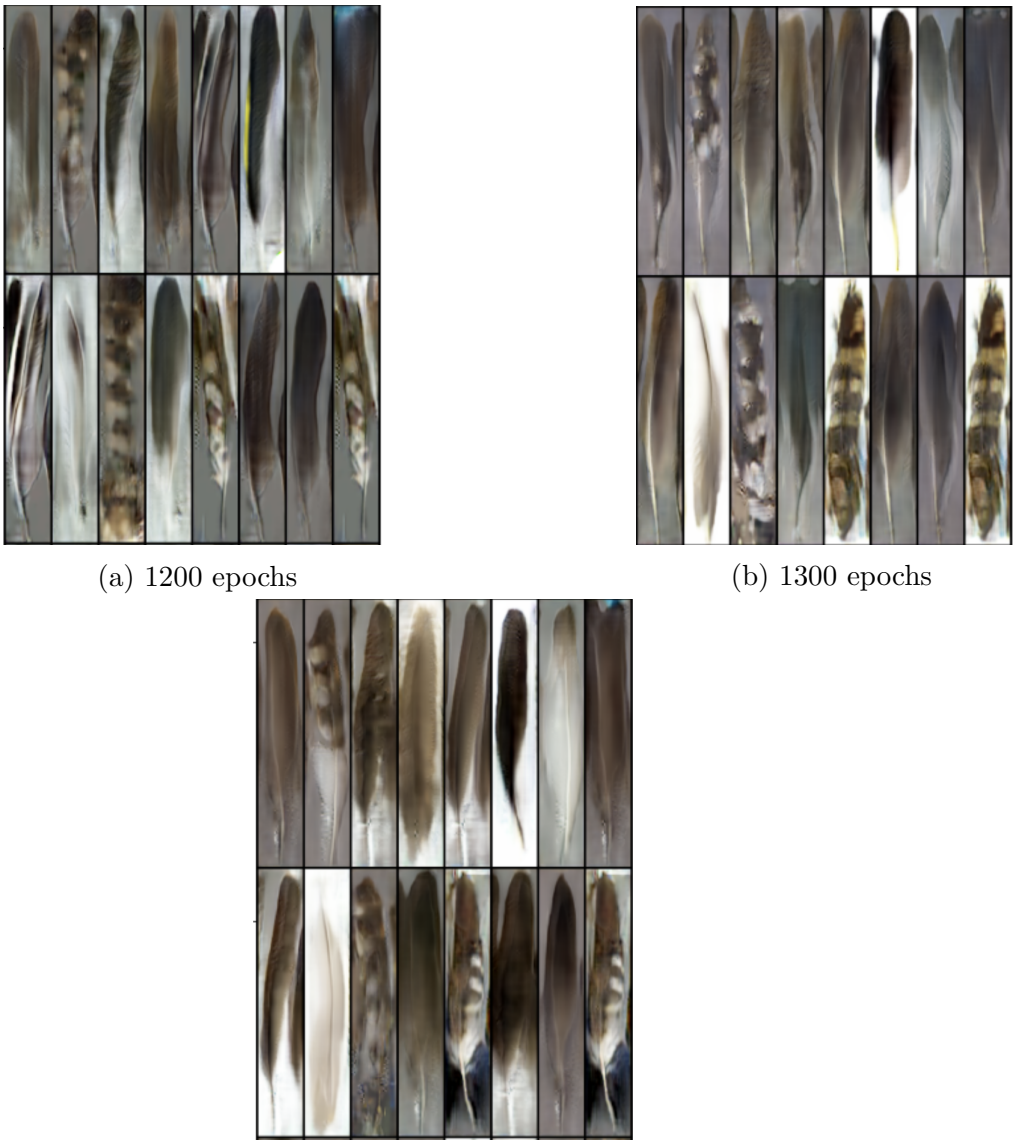
(c) 90 epochs

Figure 5.5: Examples of the generated images by Self-Attention GAN trained by 80, 86 and 90 epochs respectively

We decided to stop at 86 epochs since the self-attention GAN trained by a smaller epoch can not welly generate the structure of the feathers. In contrast, the feathers generated by this model trained for a longer period lose their actual colour.

5.3.3 EC-GAN

EC-GAN is a variation of CGAN, so no additional labelling scheme is required. To choose the optimum stopping epoch, we used classification accuracy on the fake images in addition to the manual inspection. The training classifier used was DenseNet pre-trained on our feather dataset with training accuracy.



(a) 1200 epochs

(b) 1300 epochs

(c) 1400 epochs

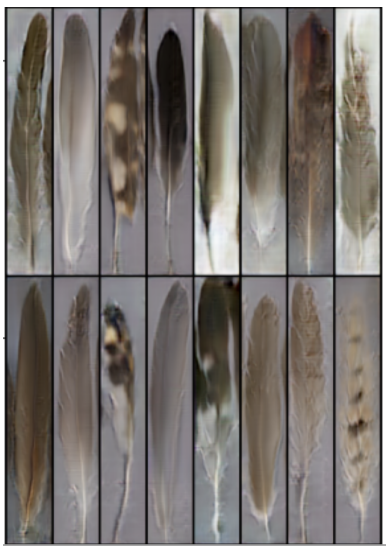
Figure 5.6: Examples of the generated images by EC-GAN trained by 1200, 1300 and 1400 epochs respectively

Epoch 1400 achieves the best classification accuracy with the value of 0.8248, while epoch 1200 and 1300 has only achieved 0.6814 and 0.7656, respectively. More extended training could lead to higher classification accuracy, but a strong model collapse was observed at this stage, so we decided to stop at 1400.

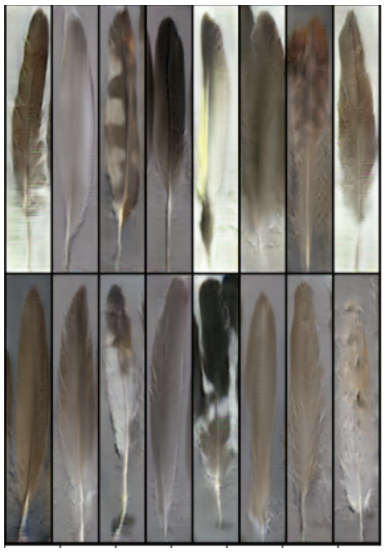
5.3.4 Our Model

We chose the optimum stopping epoch of our proposed model in a similar way to EC-GAN. With epoch 200, our model achieved a classification accuracy of 0.9559. The values of 0.9648

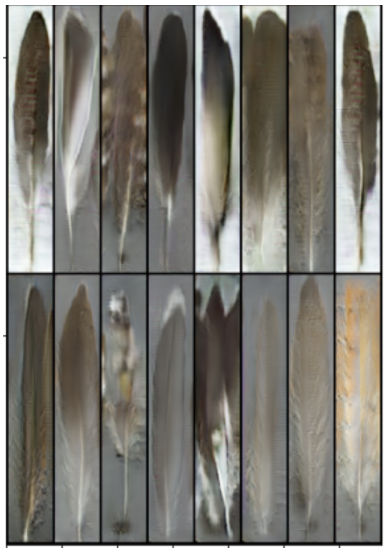
and 0.9748 were obtained for epoch 300 and 400 respectively.



(a) 200 epochs



(b) 300 epochs



(c) 400 epochs

Figure 5.7: Examples of the generated images by our model trained by 200, 300 and 400 epochs respectively

Although epoch 400 achieved the highest classification accuracy, we decided to stop at epoch 300. From figure 5.7, we can observe that our model at epoch 300 achieved the best result visually.

5.3.5 Traditional Data Augmentation

We applied vertical/horizontal flipping and contrast changing to augment our dataset since those operations do not modify the labels of our data.

We also evaluated the colour change augmentation method, which we believed was not safe on our dataset.

5.3.6 AugMix

We experimented AugMix (based on paper[17]) on our dataset as well. We augmented 30% of real images by this technique.

5.4 Accuracy Comparison

We doubled our training set by using a combination of generated images and real images to train an evaluation classifier named DenseNet121[22], which was pre-trained on ImageNet[9]. We chose such a classifier because we wanted to accelerate our evaluation process. One thing to mention is that we failed to reproduce the baseline accuracy from figure 3.3. We have tried the same classifier and trained it from stretch, but we only got accuracy higher than the baseline, therefore we will use our own baseline. The classifier's overall loss L_c consists of two parts: the loss from identifying real images L_r and the loss from classifying synthetic ones L_g. The formula of the loss function is:

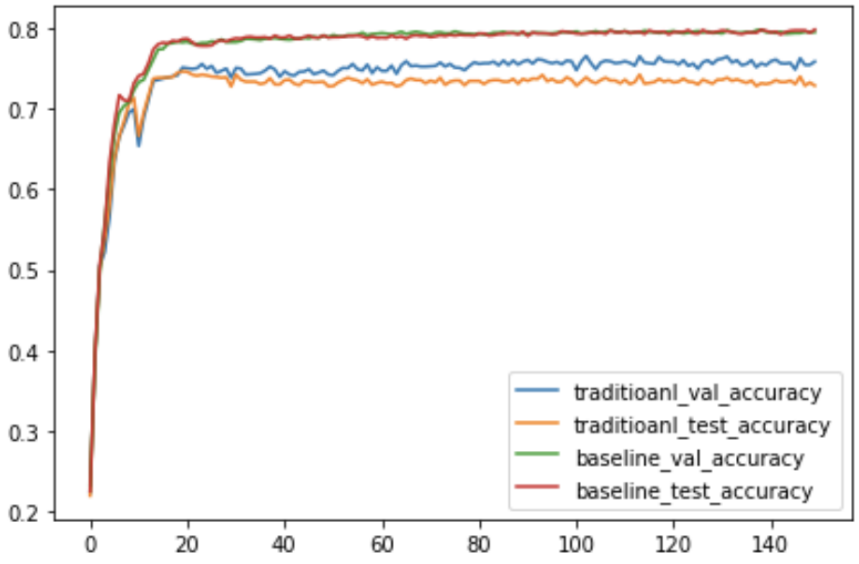
$$L_c = L_r + \lambda L_g \tag{5.2}$$

where λ is the adversarial weight. For our accuracy comparison experiment, we will use $\lambda = 0.3$.

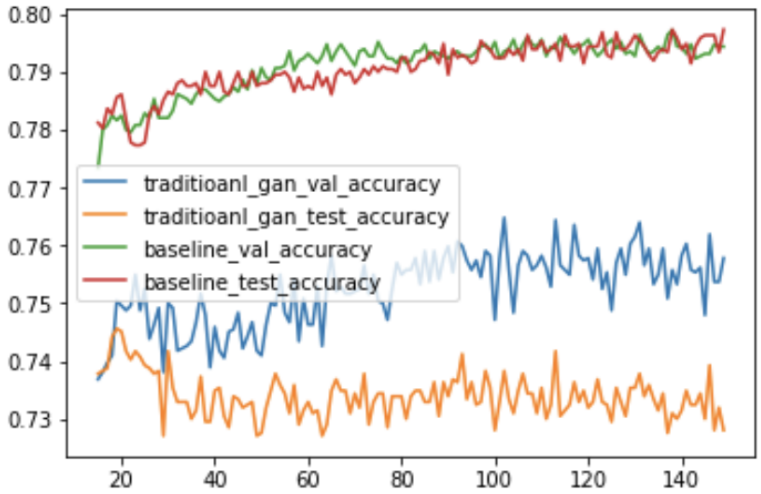
For generative models without category information, confidence-pseudo labelling with a confidence ratio of 0.8 is used.

The baseline accuracy is obtained by training our evaluation classifier on real images solely.

5.4.1 Traditional GAN



(a) Accuracy from epochs 0 to 150

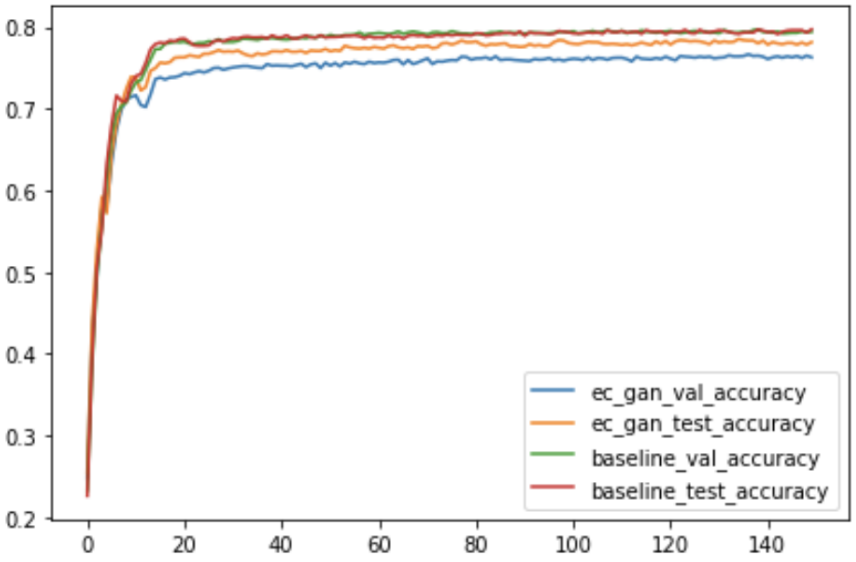


(b) Accuracy from epochs 15 to 150

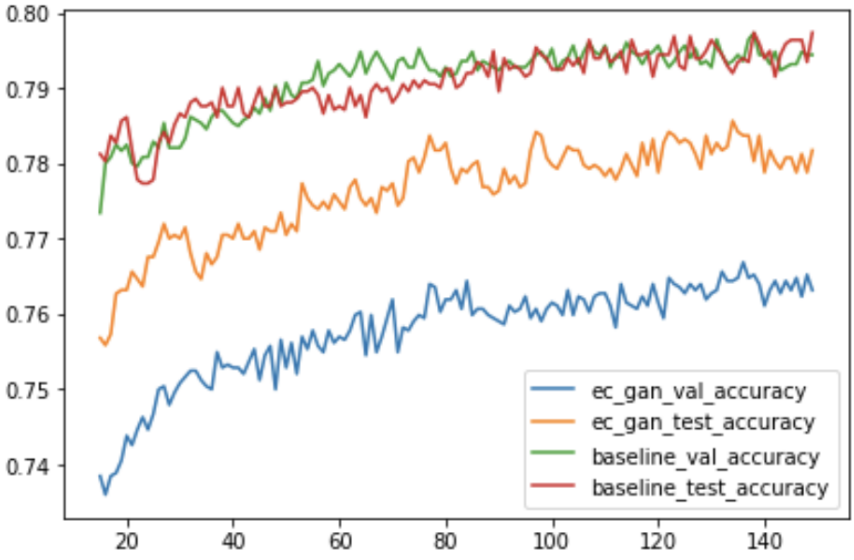
Figure 5.8: Validation and testing accuracy for traditional GAN and our baseline

The result shows that the fake images generated by traditional GAN can not help to train a better classification model. In fact, the accuracy worsens. The gap between the validation accuracy and the test accuracy for using a combination of images generated by traditional GAN and real images is larger than the gap between that for our baseline, indicating our traditional GAN does not sufficiently capture the actual data distribution and the generated images are not reliable.

5.4.2 EC-GAN



(a) Accuracy from epochs 0 to 150

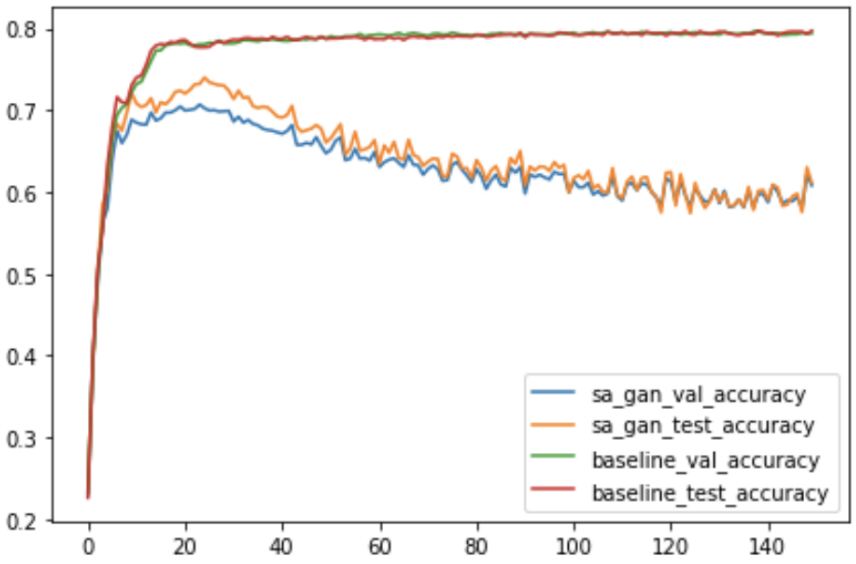


(b) Accuracy from epochs 15 to 150

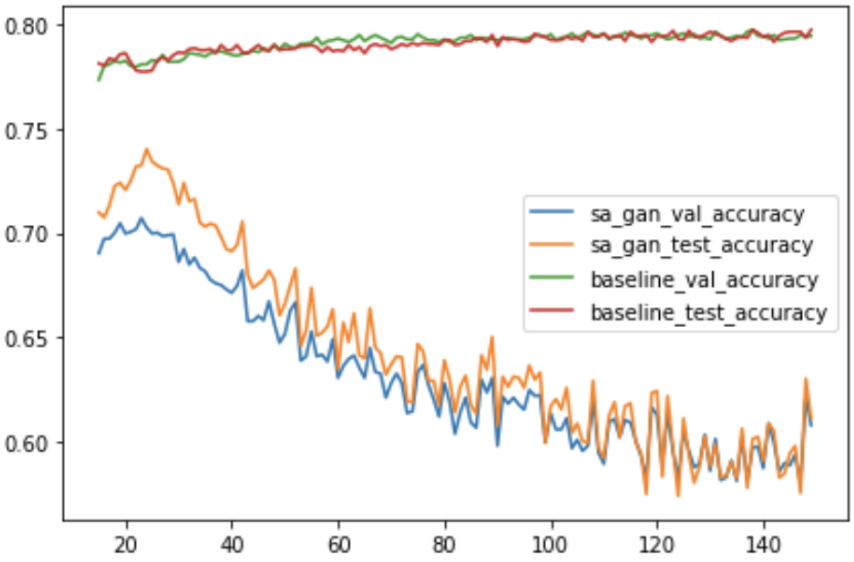
Figure 5.9: Validation and testing accuracy for EC-GAN and our baseline

Using the images generated by EC-GAN along with the real images can not help with the classification accuracy neither. However, the test accuracy is higher than the validation accuracy.

5.4.3 SA-GAN



(a) Accuracy from epochs 0 to 150



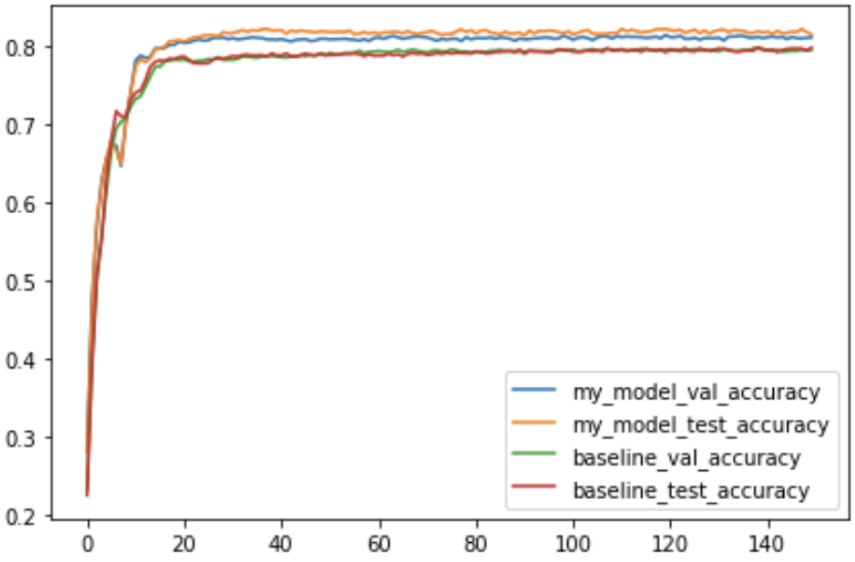
(b) Accuracy from epochs 15 to 150

Figure 5.10: Validation and testing accuracy for SA-GAN and our baseline

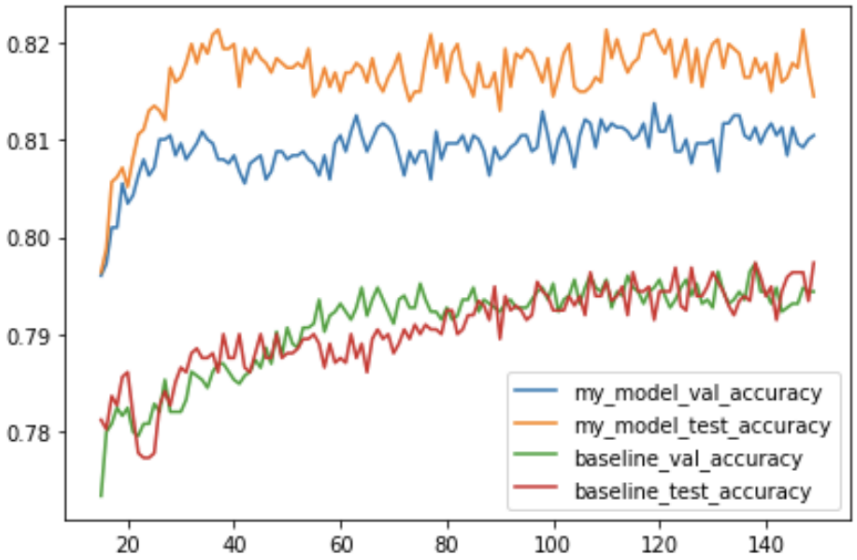
Surprisingly, self-attention GAN obtained the worst result, which contradicts our original guess. Theoretically, self-attention GAN should be able to produce more realistic images than traditional GAN. We infer why that the self-attention mechanism is ineffective because, first, every object in our sample image is centred. In addition, by visualizing CAM (see Section 5.1.1), we found that the fine-grained details are hard to be extracted; thereby, this

mechanism might attend many noises rather than beneficial information.

5.4.4 Our Model



(a) Accuracy from epochs 0 to 150



(b) Accuracy from epochs 15 to 150

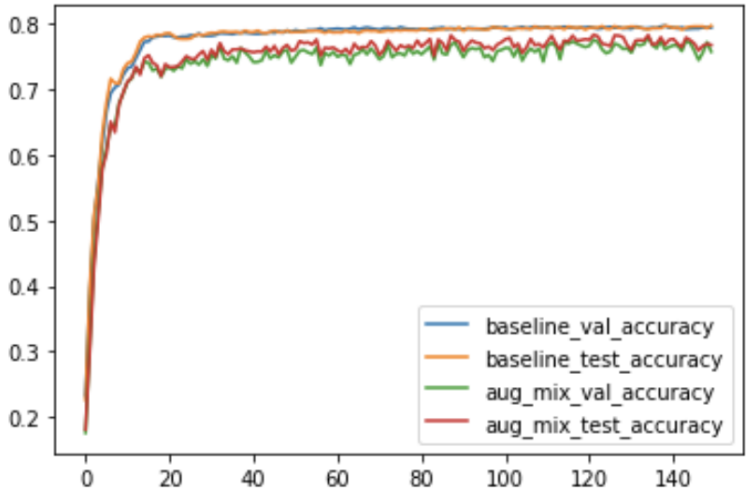
Figure 5.11: Validation and testing accuracy for our model and our baseline

	Traditional GAN	EC-GAN	SA-GAN	My model	Baseline
Maximum test accuracy achieved	0.7456	0.7856	0.7402	**0.8213**	0.7974

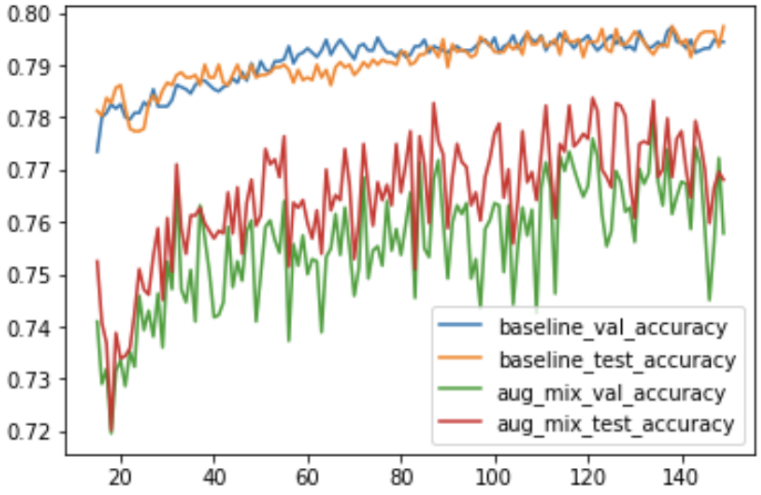
Table 5.1: Maximum test accuracy comparison between generative models

The result shows that our proposed model successfully boosted the classification accuracy by 2.39%. In addition, Our model is the only generative model evaluated capable of generating synthesized images that can be utilised for training a better feather classification model. EC-GAN is the second best model, and one thing in common between our model and EC-GAN is that there is an external classifier.

5.4.5 AugMix



(a) Accuracy from epochs 0 to 150

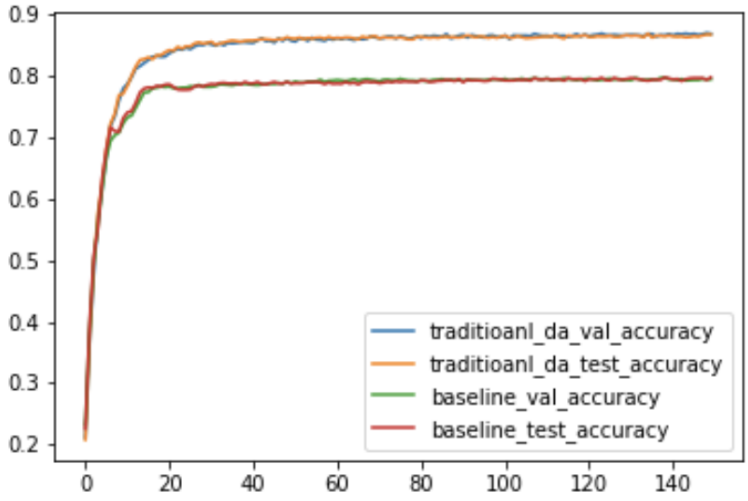


(b) Accuracy from epochs 15 to 150

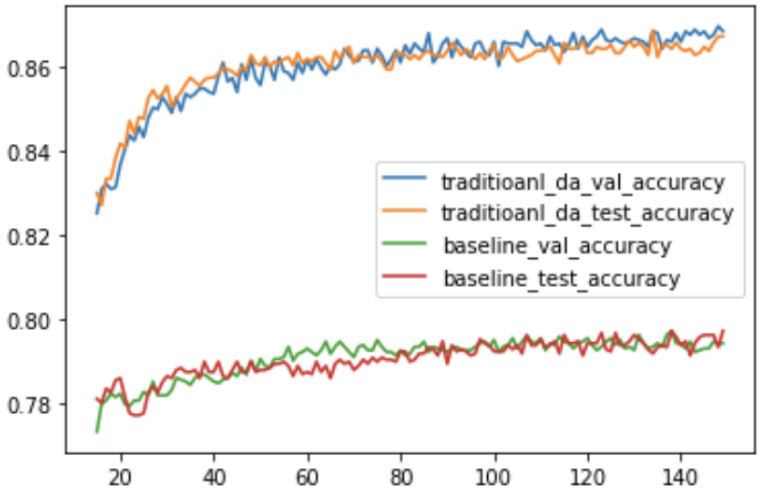
Figure 5.12: Validation and testing accuracy for AugMix and our baseline

AugMix does not improve classification accuracy, in fact, this data augmentation technique worsens the accuracy and makes the training of our evaluation classifier unstable. We believe the reason behind it is that this method is not label preserving: the feathers in augmented images are likely to belong to other categories rather than their original classes.

5.4.6 Traditional Data Augmentation



(a) Accuracy from epochs 0 to 150



(b) Accuracy from epochs 15 to 150

Figure 5.13: Validation and testing accuracy for safety image manipulations and our baseline

The experiment illustrates that traditional data augmentation is the most effective augmentation methods we tested. Since we selected safety operations to guarantee the reliability of the augmented images, the outcome is what we anticipated. Additionally, standard image operations ensure that augmented photos are of high quality, in contrast to generative models that produce images whose quality cannot be guaranteed.

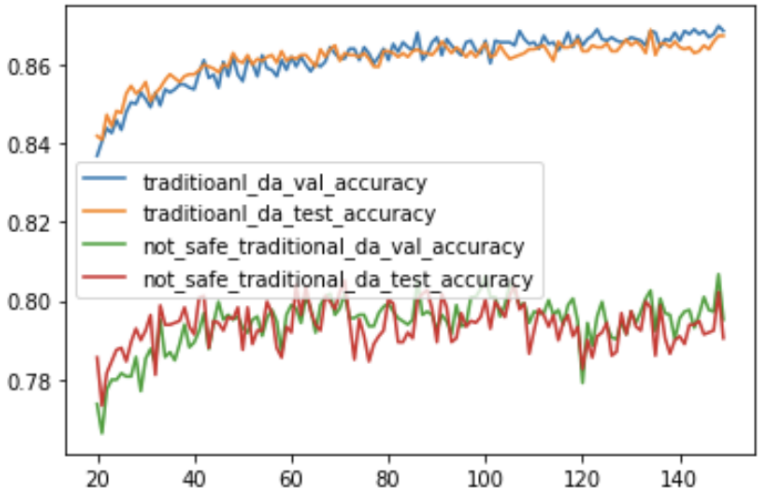


Figure 5.14: Validation and testing accuracy for unsafe/safe image manipulations

We can observe that unsafe image manipulations can not reach the result of using safe image manipulations. Also, the training of the evaluation classifier tends to be less stable. Therefore, one should carefully choose image operations according to the selected dataset to get the best classification accuracy improvement.

5.4.7 Traditional Data Augmentation + Our Model

We evaluated how the performance of our evaluation classifier would alter if we used both the safe standard image operations data augmentation methodology and generative methods, as one does not preclude the other.

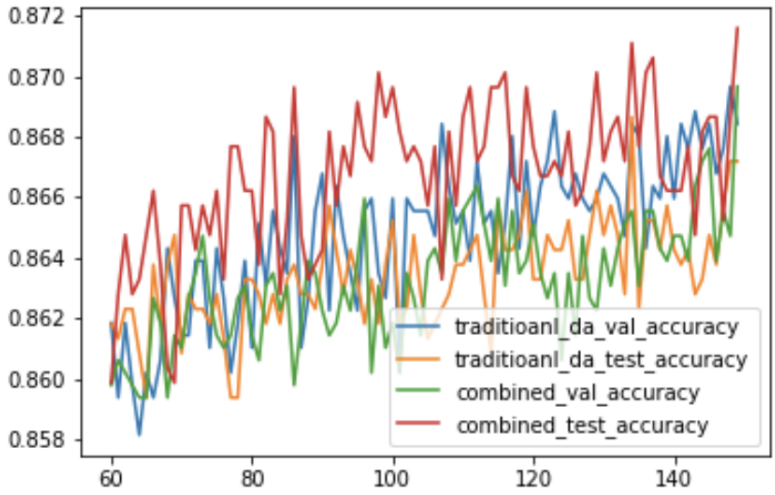


Figure 5.15: Validation and testing accuracy from epochs 60 to 150 for using a combination traditional data augmentation and our model

The performance was further improved by adding generative approaches to the traditional data augmentation techniques. Therefore, generative data augmentation techniques are still worth investigating since adding them on top of existing data augmentation techniques may boost accuracy even further, although traditional data augmentation techniques are the most efficient and successful approaches in general.

5.4.8 Summary

	Maximum Test Accuracy
Generative Models	
Traditional GAN	0.7456
EC-GAN	0.7856
SA-GAN	0.7402
Our Model	**0.8213**
Others	
Color changing augmentation(not safe)	0.8052
Traditional Data Augmentation	0.8687
Traditional Data Augmentation + Our Model	**0.8716**
AugMix	0.7837
Baseline	
No data augmentation	0.7974

Table 5.2: Summary of test accuracy for different data augmentation methods

Table 5.2 shows a summary of the maximum test accuracy achieved for different data augmentation methods. Among all the generative models we analysed, the model we have proposed is the most effective for augmenting the dataset we have chosen which exceeds the baseline accuracy by 2.39%. Self-attention GAN behaved the worst with an accuracy drop by 5.72%. When augmenting images by basic image manipulations, we need to choose label-persevering operations. Not all image operations can be applied. Although safe traditional data augmentation outperforms all generative models, including our model, applying our model on top of it can still enhance the classifier's performance (accuracy further improved by 0.29%).

5.5 FID Score Comparison

FID score evaluation only applies to generative models. The lower the FID score, the closer the distribution of our generated images is to real data distribution, meaning the better quality.

	FID Score
Traditional GAN	1.0227
EC-GAN	0.9566
SA-GAN	1.1311
Our model	**0.8204**

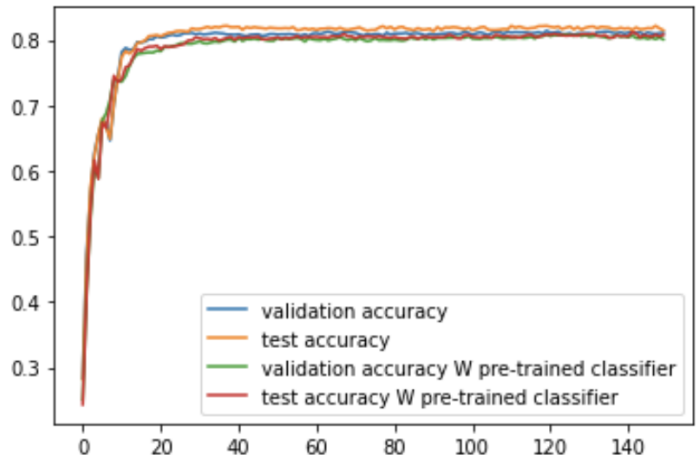
Table 5.3: FID score for generative models

There is a direct link between the FID score and the accuracy of the models we evaluated. Generative models with lower FID scores usually produce synthetic images, which can better train the evaluation classifier than those with higher scores.

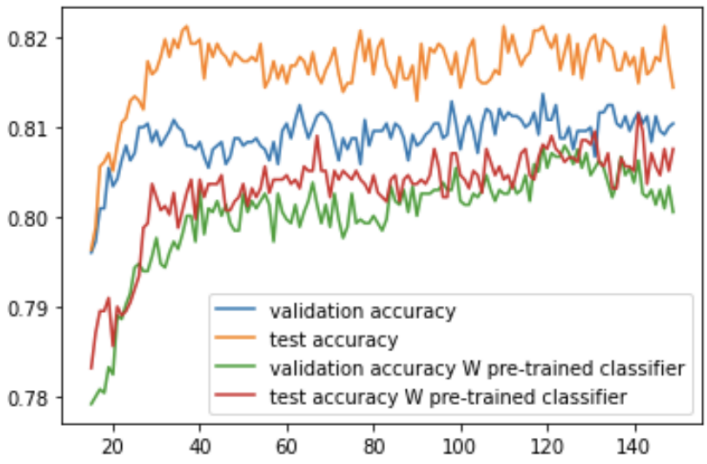
5.6 Ablation Study

5.6.1 Pre-trained Training Classifier

We now look into whether employing a pre-trained training classifier aided conditional generative model training. We used the same settings to train our model twice; however, one employs a pre-trained classifier, and the other does not. We stopped training at 300 epochs for both cases and then used the generated images along with the real images to train our evaluation classifier.



(a) Accuracy from epochs 0 to 150



(b) Accuracy from epochs 15 to 150

Figure 5.16: Validation and testing accuracy for our model trained with/without a pre-trained classifier

	FID
Our model without a pre-trained classifier	0.8860
Our model with a pre-trained classifier	**0.8203**

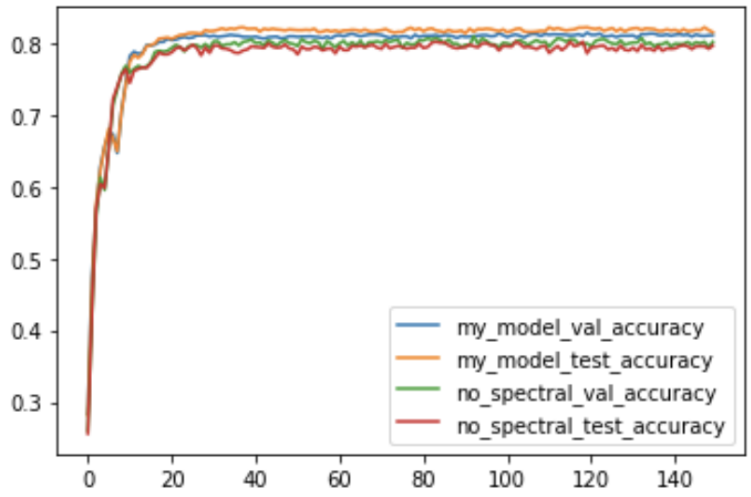
Table 5.4: FID score for generated images by our model with/without a pre-trained training classifier

The overall trend of accuracy curves is similar for the two cases (See Figure 5.16 (a)). However, we can see that employing a pre-trained training classifier aids in the learning of generating fake images by our generative model, hence enhancing classification accuracy (See Figure 5.16 (b)). Also, table 5.4 shows that the quality of the generated images is better

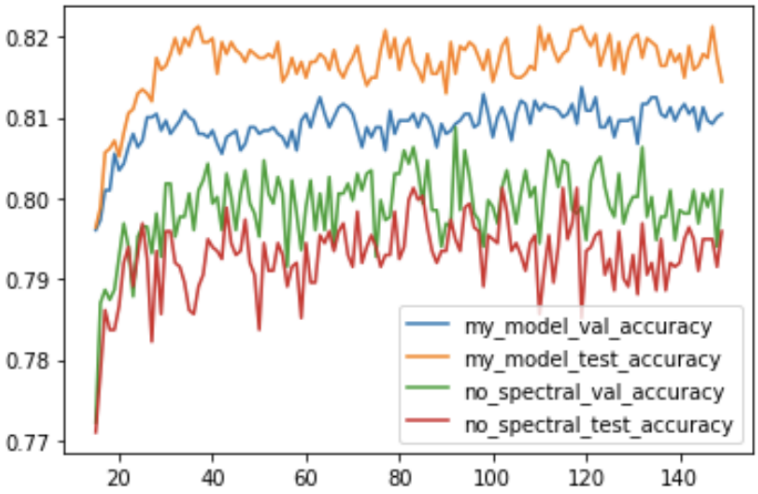
using a pre-trained training classifier. Even though the improvement is slight, it confirms our initial hypothesis when designing our model: having a good training classifier early on can better teach our generator to learn from the distinct features and be less affected by false information.

5.6.2 Spectral normalisation

We also investigated how spectral normalisation impacts the performance of our model. We trained our model without spectral normalisation for 300 epochs as well.



(a) Accuracy from epochs 0 to 150



(b) Accuracy from epochs 15 to 150

Figure 5.17: Validation and testing accuracy for our model trained with/without spectral normalisation

	FID
Our model without spectral normalisation	0.9236
Our model with spectral normalisation	**0.8203**

Table 5.5: FID score for generated images by our model with/without spectral normalisation

From figure 5.17, we can see that without spectral normalisation, the performance of our model degraded. Although the accuracy improvement is small compared to the use of a pre-trained training classifier, there is a greater enhancement in terms of the FID scores.

5.6.3 Summary

	Maximum Test Accuracy
Our model without spectral normalisation	0.8012
Our model without a pre-trained classifier	0.8116
Our model	**0.8213**
Baseline	0.7974

Table 5.6: Summary of test accuracy for ablation study

	FID
Our model without spectral normalisation	0.9236
Our model without a pre-trained classifier	0.8860
Our model	**0.8203**

Table 5.7: Summary of the FID scores for ablation study

Both spectral normalisation and a pre-trained training classifier boosted the performance of our model. The removal of spectral normalisation has the greatest impact on our model, which leads to an accuracy decrease of 2.01%. Using a pre-trained classifier can further increase the performance of our generative model (accuracy improved by 0.97%). Spectral normalisation contributes the most in terms of the FID scores.

Chapter 6

Conclusion and Future Work

Conclusions drawn from the research are discussed in this chapter, along with some potential work for additional research.

6.1 Conclusion

Data augmentation is an important area in machine learning. Although basic image manipulations are mainly used to augment data, researchers have realised the power of generative methods for classification tasks. More experiments about this area have been done in recent years.

In our study, we proposed a generative model which has been proved able to assist the fine-grained classification task on our chosen dataset, whereas standard generative methods cannot. We developed our model using VAE/GAN framework, and we found that incorporating a pre-trained classifier can better lead our generator to learn from the most fine-grained details. This pre-trained classifier is an external classifier and we chose use the same network same as the evaluation classifier for accuracy measurement in the inspirit of meta-learning. Also, using spectral normalisation instead of batch normalisation on our discriminator enables our generator to produce more realistic images. In addition to the accuracy measurement, our model produces the most realistic images compared to other generative models. Also, there was a direct link between the FID score and classification accuracy for our chosen dataset. The better the FID score, the better the generative model can help the classifier to classify feathers.

For our chosen dataset, traditional data augmentation is still the best way since it is easy to implement and can achieve a notable result. However, generative models are worth investi-

gating since a good one can boost the accuracy additionally.

6.2 Future Work

There are still many things left to do in the future. First, we still believe that CAM and self-attention mechanisms can be applied in this area, but we just do not have the resources to find suitable ones. Also, those models we evaluated have the potential to have better performance if we have enough time to fine-tune them well. Thirdly, there is still ample space to improve the performance of the proposed model. Our model architecture could be further enhanced since we did not spend much time designing or experimenting with the layers of our network due to time limitations. Finally, we plan to use similarity measurement, a more reasonable loss for reconstruction, instead of pixel-wise error loss in our loss functions, but we currently do not have the resources since computing similarity is costly.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: Fine-grained image generation through asymmetric training, 2017.

[3] Alina Belko, Konstantin Dobratulin, and Andrey Kuznetsov. Feathers dataset for fine-grained visual categorization, 2020.

[4] Lusa L. Blagus, R. Smote for high-dimensional class-imbalanced data. bmc bioinformatics 14, 106. 2013.

[5] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation, 2017.

[6] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014.

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

[8] Rahul Dalal and Teng-Sheng Moh. Fine-grained object detection using transfer learning and data augmentation. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '18, page 893–896. IEEE Press, 2018.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[10] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space, 2017.

[11] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition, 2017.

[12] Yan Fu, Xutao Li, and Yunming Ye. A multi-task learning model with adversarial data augmentation for classification of fine-grained images. *Neurocomputing*, 377:122–129, 2020.

[13] Mingyang Geng, Kele Xu, Bo Ding, Huaimin Wang, and Lei Zhang. Learning data augmentation policies using augmented random search, 2018.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[15] Wenming Guo, Yifei Wang, and Fang Han. Attention-guided cutmix data augmentation network for fine-grained bird recognition. In *2021 2nd International Conference on Artificial Intelligence and Information Systems*, ICAIIS 2021, New York, NY, USA, 2021. Association for Computing Machinery.

[16] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample, 2020.

[17] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty, 2019.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017.

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] Mingqi Hu, Deyu Zhou, and Yulan He. Variational conditional gan for fine-grained controllable image generation, Oct 2019.

[21] Tao Hu and Honggang Qi. See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification. *CoRR*, abs/1901.09891, 2019.

[22] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[23] Shaoli Huang, Xinchao Wang, and Dacheng Tao. Snapmix: Semantically proportional mixing for augmenting fine-grained data, 2020.

[24] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, 2013.

[25] Ruyi Ji, Longyin Wen, Libo Zhang, Dawei Du, Yanjun Wu, Chen Zhao, Xianglong Liu, and Feiyue Huang. Attention convolutional binary neural tree for fine-grained visual categorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[26] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing*, 30:5875–5888, 2021.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[29] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets, 2017.

[30] Hao Li, Xiaopeng Zhang, Hongkai Xiong, and Qi Tian. Attribute mix: Semantic data augmentation for fine grained recognition, 2020.

[31] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnns for fine-grained visual recognition, 2015.

[32] Zinan Lin, Vyas Sekar, and Giulia Fanti. Why spectral normalization stabilizes gans: Analysis and improvements, 2020.

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[34] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[35] Qunfeng Tang David Smith John-Paul Grenier Catherine Batte Bradley Spieler William Donald Leslie1 Carlo Menon Richard Ribbon Fletcher Newton Howard Rabab Ward William Parker Mohamed Elgendi, Muhammad Umer Nasir and Savvas Nicolao. The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective. 2021.

[36] Humza Naveed. Survey: Image mixing and deleting for data augmentation. *CoRR*, abs/2106.07085, 2021.

[37] Alex K Ilya S Ruslan S. Nitish S, Geoffrey H. Dropout: a simple way to prevent neural networks from overfitting. 2014.

[38] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2016.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[41] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[43] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 2007.

[44] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.2.1.

[45] Christan S Sergey I. Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015.

[46] Jia Shijie, Wang Ping, Jia Peiyi, and Hu Siping. Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese Automation Congress (CAC)*, pages 4165–4170, 2017.

[47] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning - journal of big data, Jul 2019.

[48] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[49] Baikai Sui, Tao Jiang, Zhen Zhang, and Xinliang Pan. Ecgan: An improved conditional generative adversarial network with edge detection to augment limited training data for the classification of remote sensing images with high spatial resolution. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:1311–1325, 2021.

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[53] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning, 2019.

[54] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification, 2018.

[55] Zhen Yang, Zhipeng Wang, Lingkun Luo, Hongping Gan, and Tao Zhang. Sws-dan: Subtler ws-dan for fine-grained image classification. *J. Vis. Comun. Image Represent.*, 79(C), aug 2021.

[56] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019.

[57] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018.

[58] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[59] Ning Zhang, Jeff Donahue, Ross B. Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. *CoRR*, abs/1407.3867, 2014.

[60] Tian Zhang, Dongliang Chang, Zhanyu Ma, and Jun Guo. Progressive co-attention network for fine-grained visual classification. *CoRR*, abs/2101.08527, 2021.

[61] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017.

[62] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015.