Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Semantic Synthesis of Diabetic Retinal Fundus Images

Author:
Joon-Ho Son

Supervisor:
Dr Benjamin Hou

Second Marker:
Dr Amir Alansary

June, 2021

Abstract

Retinal fundus imaging is a modality that's quick and easy to obtain, however its usefulness is limited by the availability of human graders. With advancements in deep learning, data-driven models may facilitate the process of diagnosis, alleviating this bottleneck. Despite this, training robust machine learning models requires manually annotated labels en-masse, which are costly to obtain, and remains to be one of the primary obstacles in producing generalisable models. This challenge can be viewed in two aspects: the scarcity of manually annotated ground truths such as lesion segmentation maps, and class imbalance in available datasets. Both of these can be seen in datasets designed for classification and segmentation tasks related to diabetic retinopathy (DR).

In this project, we propose a novel two-step process for generating photo-realistic fundus images conditioned on synthetic "ground truth" semantic labels. First, synthetic semantic labels for retina structures and lesions are generated from random noise, conditioned on DR severity. These are then subsequently used to condition an image-to-image translation model to create the final retinal fundus image. We compare the performance of these sophisticated generative models against a simple data augmentation method.

We go on to demonstrate its potential to improve performance on downstream tasks in classifying and segmenting diabetic retinal fundus images.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisor **Dr Benjamin Hou** for his unwavering support and guidance throughout this project. I would also like to thank my second marker **Dr Amir Alansary** for his valuable feedback during the early stages.

I reserve special thanks for my mother, whose countless sacrifices made a better life possible for her sons.

Contents

Chapter 1

Introduction

1.1 Motivation

Diabetic retinopathy (DR) is an eye disease that commonly arises as a complication of diabetes. It is estimated that by the year 2045, 693 million people worldwide will be diabetic. Of these, nearly all of those with type 1 and two-thirds of those with type 2 will be suffering some degree of retinopathy within 20 years of receiving their diagnosis [1]. Despite DR being preventable with early detection and intervention, this is made difficult by the fact that the early stages of DR show no symptoms and may only be identified by screening. However, the scalability of mass screening is severely limited by the availability of medical professionals, causing DR to remain the leading cause of blindness in the UK [2]. Currently, diagnosis requires manual inspection of retinal fundus images for the presence of abnormalities. This is a time-consuming and error-prone process for ophthalmologists who, even when available, have been shown to be inconsistent [3]. For this reason, automated and semi-automated techniques for DR diagnosis have been a popular topic of research, going back as far as 1984 [4]. As more sophisticated and powerful methods are developed, we come ever closer to accessible screening for *all* susceptible individuals.

The presence of DR is characterised by the formation of lesions on a patient's retina, the type and quantity of which indicate the severity of the disease. Grading (i.e. classifying) the progression of diabetic retinopathy that a patient exhibits allows for the most effective and timely treatment to be given. Moreover, the ability to identify the precise locations of these lesions and distinguish them from the healthy parts of a retina ensures that the most appropriate and interpretable diagnosis possible can be provided, as well as having uses in aiding surgical procedures. These two classification and semantic segmentation tasks are the two major research interests at the intersection of machine learning and diabetic retinopathy treatment.

Developments in deep learning over the past decade have sparked a number of breakthroughs in the field of medical imaging [5]. Deep neural networks provide state-of-the-art models across a variety of domains and continue to show even greater potential. These innovations are fuelled by greater quantities of data, allowing models to generalise to unseen examples. It is this insatiable hunger for data that presents one of the primary obstacles in automated DR diagnosis, as large-scale, annotated datasets are scarce.

An easy way to improve data diversity is by applying classical data augmentation techniques such as reflections, crops, rotations, and colour perturbations. However, these methods produce images that are highly correlated, and ultimately limited in their diversity. A more advanced method of image simulation has been to hand-craft complex mathematical models representing the anatomy of the eye, from which images can be sampled. More recently, with the rise of data-driven techniques, we have seen a paradigm shift away from this top-down approach to a bottom-up approach of learning the data distribution *directly* from the data itself. This has been made possible by the introduction of Generative Adversarial Networks (GANs). The aim of this project is to leverage these generative models, as well as other data augmentation techniques, to produce realistic synthetic training data in large volumes.

Achieving large-scale data generation with arbitrary labels would yield significant improvements in the ability of neural networks to both semantically segment retinal fundus images, as well as assign image-level grades. Ultimately, with enough high-quality synthetic data, deep learning models are poised to *surpass* human ability in diagnosing DR from fundus images. This project represents a step towards the goal of fully-automated retinal screening for the detection of diabetic retinopathy, by presenting methods to combat the scarcity of data.

1.2 Objectives and Challenges

The aim of this project is to investigate whether it is feasible to enhance the performance of DR diagnosis models by training on synthetic data. This can be summarised by the following objectives:

1. Establish and compare methods for the generation of synthetic retinal fundus images that are:
   (a) conditioned on DR severity; and
   (b) paired with segmentation maps.
2. Investigate how training on synthetic data can affect the performance of:
   (a) lesion segmentation models; and
   (b) severity classification models.

While there is an established body of work on using generative models to create natural-looking images, their use in generating synthetic training data – and in particular semantic labels – remains nascent. The absence of a large body of literature introduces more unknowns into determining the success of this project.

Somewhat ironically, the scarcity of existing data makes it difficult to train effective models to generate further, synthetic data. Motivated by this, we will examine both a learning-based method, as well as a "naive", heuristic-based method.

Lastly, large, unstable models and limited access to hardware mean that we must be extremely economical with the experiments we choose to run, making tuning difficult.

1.3 Contributions

This work's main contributions are:

Exploratory Data Analysis
We begin by providing a survey and analysis of datasets for training such generative models in Chapter 4. The goal of this is to expose any biases present in the data, which may inform future work. As an additional contribution, we provide new manual optic disc annotations for several datasets.

Semantic Synthesis of Diabetic Retinal Fundus Images
In Chapter 5, we present novel methods to generate synthetic semantic labels, conditioned on DR severity, thereby providing downstream tasks with both segmentation maps and DR severity information. We go on to translate these semantic labels into realistic retinal fundus images in Chapter 6, creating pairs of semantic labels and fundus images. We provide insights on how existing techniques must be adapted for this domain, building on the existing literature.

Evaluation of Synthetic Data on Downstream Tasks

Finally, in Chapter 7 we compare and contrast different approaches to training on synthetic data for various downstream tasks in order to evaluate the viability of generated images as a means of data augmentation.

1.4 Publication

A preliminary version of this work was accepted as a short paper to MIDL 2021. Its findings, and impact on this version of the project are discussed briefly in Section 7.1.

Chapter 2

Background

In this chapter, we provide the necessary preliminary concepts to understand the problem domain and current state-of-the art in image generation. We assume that the reader has prior knowledge of the basic concepts around artificial neural networks and their training.

2.1 Diabetic Retinopathy

Diabetic retinopathy is damage to the blood vessels of the retina caused by high blood sugar levels. The level of damage, and hence the stage of DR, can be determined by identifying and categorising different types of lesions on the patient's retina. A description of the appearance of major lesion types is given as follows [6]:

Microaneurysms (MA)
Small red round dots caused by weakness in the vessel's walls.

Haemorrhages (HE)
Larger spots on the retina.

Hard exudates (EX)
Bright yellow spots caused by the leakage of plasma.

Soft exudates (SE)
White spots caused by swelling of nerve fibre. Also known as "cotton wool spots".

Intraretinal microvascular abnormalities (IRMA)
Abnormally shaped blood vessels. Usually a precursor to neovascularization.

Neovascularization (NV)
Abnormal growth of new blood vessels. Similar in appearance to IRMA, but tend to be finer and show signs of blood leakage.

Examples of each type of lesion, and the optic disc, are labelled on a colour fundus photograph in Figure 2.1. Note that, while other imaging techniques such as fluorescein angiography or optical coherence tomography (OCT) are possible, we are interested in only colour fundus photographs. They are key in scaling DR screening, being low-cost, easy to obtain, and offering the option of a remote ophthalmologist examination via a telemedicine platform [7]. At the intersection of machine learning and DR detection there are a number of distinct, but related, research areas:

Image-Level Grading
To identify a DR severity grade for a fundus photograph. This can be according to the the international disease scale (summarised in Table 2.1), or a more coarse scale which combines several stages e.g. healthy, referred, diseased. For example, see [8]. This task is difficult due to inter- and intra-grader variation, which causes datasets to be noisy.

Pixel-Level Segmentation
To perform semantic segmentation of the retinal structure (blood vessels, optic disc) and/or lesions. For example, see [9].

Figure 2.1: A retinal fundus image annotated with the optic disc as well as six types of lesions: hard exudates (EX), haemorrhages (HE), microaneurysms (MA), soft exudates (SE), neovascularization (NV), and intraretinal microvascular abnormalities (IRMA).

Computer-Assisted Diagnosis

To generate attention maps for suspicious areas, to be reviewed by a ophthalmologist. For example, see [10].

Image Synthesis

To generate synthetic retinal fundus photographs. These could be conditioned on, for example, vascular structure or DR grade. This task is the subject of this project, and related work is discussed in greater detail in later sections.

Severity	Characteristics
No apparent retinopathy	No abnormalities
Mild NPDR	Microaneurysms only
Moderate NPDR	More than just microaneurysms but less than severe NPDR
Severe NPDR	Any of the following and no signs of PDR: • $>$ 20 intraretinal haemorrhages in all 4 quadrants • Venous beading in $\geq$ 2 quadrants • Intraretinal microvascular abnormalities in $\geq$ 1 quadrant
PDR	One or both of: • Neovascularisation • Vitreous/preretinal haemorrhage

Table 2.1: The International Clinical Diabetic Retinopathy Disease Scale [11].
NPDR = non-proliferative diabetic retinopathy; PDR = proliferative diabetic retinopathy.

2.2 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a machine learning framework introduced by Goodfellow et al. in 2014 [12] consisting of two neural networks in adversarial competition. The generative model G creates candidate images that look as "real" as possible, while the discriminative model D attempts to distinguish between real and synthetic images. Training continues until the discriminator can no longer tell which inputs are fabricated by G and which are real. In this sense, G and D are trained *adversarially*. From this, we can derive the adversarial loss function by taking the cross-entropy of the real and generated samples:

$$\mathcal{L}_{GAN} = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z))] \tag{2.1}$$

This yields the corresponding objective function:

$$\min_G \max_D \mathcal{L}_{GAN} \tag{2.2}$$

where $D(x)$ is the discriminator's estimate that $x \sim p_{data}$ is real; $G(z)$ is the generator's output when given random noise $z \sim p_z$; and $D(G(z))$ is the discriminator's estimate that a fake instance is real. The discriminator will attempt to maximise this loss, whilst the generator attempts to minimise it. In practice, this happens in alternating phases, keeping the generator fixed when updating the discriminator, and vice versa. This means that the discriminator and generator can have differing loss functions instead of strictly adhering to the joint definition in Equation (2.1).

The versatility of GANs has enabled them to be applied to a plethora of problems including semantic segmentation and text generation, but their first and primary use is image synthesis. The vanilla GAN design is far from perfect, however, and as such there has been an explosion of GAN variants since their initial introduction, making incremental improvements over each other. In the following sections, we examine a selection of extensions and variants that are relevant to this project.

Note that, while I have attempted to disentangle developments in GAN research over time by introducing these concepts in a coherent order, research does not always progress in a straight line, and new techniques do not always incorporate all the ideas of previous techniques. Refer to Figure 2.2 for a holistic overview of how the discussed concepts relate to each other.

2.3 Improving GAN Training

Training GANs is known to be unstable, difficult to make converge, and prone to issues like mode collapse. There have been a number of attempts to address these problems.

2.3.1 Modified Minimax Loss

In their original paper, Goodfellow et al. describe an issue with the original loss formulation where G fails to gain any traction in the early stages of training, and therefore never even begins to converge. This is due to D being able to easily reject to the output of G, which causes the gradients to be extremely small. To mitigate this, the authors propose to maximise $\log D(G(z))$ instead of minimising $\log(1 - D(G(z))$ as in Equation (2.1) i.e. the generator loss (denoted by superscript G) becomes

$$\mathcal{L}^G_{MM} = -D(G(z)) \tag{2.3}$$

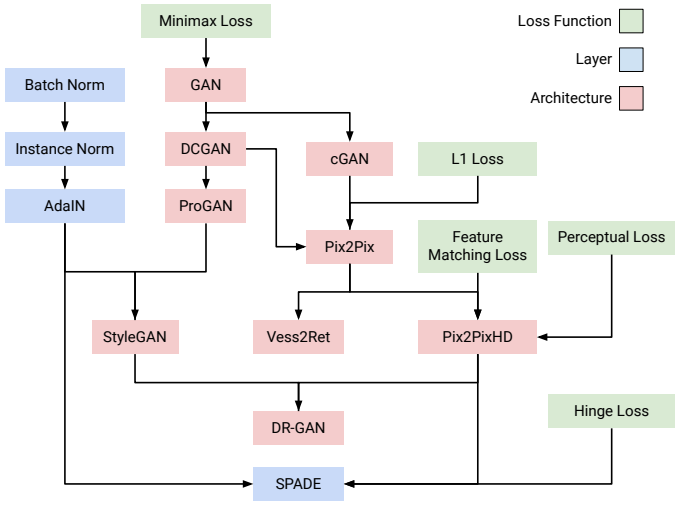


Figure 2.2: Concept map of discussed techniques and how they relate to one another.

However, even this loss function is prone to vanishing gradients, and as a result other types of adversarial losses such as Wasserstein and Hinge loss have also been proposed. These alternatives aim to provide consistently strong gradients throughout training.

2.3.2 Feature Matching Loss

Even with the above modification, GANs can still fail to converge (i.e. the generator and discriminator fail to reach a Nash equilibrium). This is because when the one network gets closer to the equilibrium, it can cause the other to move further from the equilibrium. This oscillation can increase over time, causing divergence. To address this, in 2016 Salimans et al. proposed a modification to the loss function called the feature matching loss [13]. In this scheme, instead of asking the generator to directly match the output of the discriminator, we ask it only to match some statistic of the real data. These statistics are retrieved from an intermediate layer, i, of the discriminator, denoted by $D^{(i)}$:

$$\mathcal{L}_{FM}^{G} = \left\| \mathbb{E}_x[D^{(i)}(x)] - \mathbb{E}_z[D^{(i)}(G(z))] \right\| \tag{2.4}$$

where $\|\cdot\|$ is some norm, usually L_1 or L_2.

2.3.3 Perceptual Loss

Perceptual loss[1], closely related to feature matching loss[2], refers to the idea of defining a loss function in terms of a separate network F in which image features are encoded [17]. We take the loss as the distance between the two feature representations.

$$\mathcal{L}_{P}^{G} = \left\| F^{(i)}(x) - F^{(i)}(G(z)) \right\| \tag{2.5}$$

It has been shown that, in convolutional neural networks, perceptual losses measure the similarity of images in a more robust manner compared to per-pixel losses [15]. A popular choice for network F, called the perceptual network, is VGG.

[1]This has also been called the "content representation loss" [14] and "feature reconstruction loss [15] in the literature.

[2]The distinction between feature matching and perceptual losses is not entirely clear, with some publications even using terms interchangeably [16]. Here, we will use "feature matching loss" to refer to Equation (2.4) and "perceptual loss" to refer to Equation (2.5).

2.3.4 Batch Normalisation

During training, the distribution of the inputs into any given layer will constantly be changing as the parameters of preceding layers change – a phenomenon known as "internal covariate shift". This means that the current layer must keep readjusting to ever-changing distributions. With deeper neural networks, this effect is amplified as changes in scale propagate down the layers. Therefore, in order to keep training stable it is necessary to use lower learning rates, causing training to be slow overall. To address this, Ioffe and Szegedy proposed a method called batch normalisation[3]. When applied to a particular layer, the input of said layer is re-centred (to zero mean) and re-scaled (to unit standard deviation) across the batch and spatial dimensions for each channel [18]. This allows for the use of larger learning rates since we do not run the risk of exploding gradients, which in turn enables faster training overall.

This normalising step means that the layer's activations will be unaffected by changes in scale in previous layers. However, the authors point out that by normalising a layer we may change what the layer is capable of representing. To restore the "representational power" of the layer, we now *denormalise* the normalised activations by performing an affine transformation using learned parameters. Note how this transformation is still independent of the distribution of activations in previous layers.

Formally, consider a batch of feature maps $X \in \mathbb{R}^{N \times C \times H \times W}$. We can compute the mean and standard deviation for each channel c as

$$\mu_c(X) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{nchw} \tag{2.6}$$

$$\sigma_c(X) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (X_{nchw} - \mu_c(X))^2} \tag{2.7}$$

Then the batch transform for each element is given by

$$\mathrm{BN}_{\gamma,\beta}(X_{nchw}) = \gamma_c \left(\frac{X_{nchw} - \mu_c(X)}{\sigma_c(X)} \right) + \beta_c \tag{2.8}$$

where $\gamma, \beta \in \mathbb{R}^C$ are the per-channel transformation parameters.

2.4 Image Synthesis

This section establishes variations on GANs that establish the current state-of-the-art in image synthesis.

2.4.1 Hierarchical GANs

The original paper detailing GANs by Goodfellow et al. uses the MNIST dataset as a demonstration of the capability of GANs. These images are very low-resolution, at just 28×28 pixels, due to the fact that the original approach did not scale well to high-resolution images.

In classical computer vision, the concept of a Laplacian pyramid on which filters can operate on multiple different scales is well established [21]. Extending this idea to GANs, one of the first attempts at improving high-resolution image synthesis was LAPGAN, introduced

[3] While batch normalisation has been shown to be empirically effective, it has since been argued that the "internal covariate shift" explanation does not hold water [19, 20].

by Denton et al. in 2015 [22], where a separate generative CNN is trained at each scale. Since then, variants on this key insight of "stacking" models that operate on different scales has been explored thoroughly, whether it is stacking entire GANs [23], stacking only discriminators [24], or stacking only generators [25].

In spite of these advances, high-resolution image synthesis has remained difficult due to the fundamental fact that the increased detail makes it easier for the discriminator to distinguish real from generated images, amplifying the vanishing gradient problem that has historically plagued GAN training.

2.4.2 DCGAN

The difficulty of scaling up a GAN with CNNs to generate high-resolution images led to LAPGAN's alternative approach of leveraging multiple separate CNNs. Deep Convolutional GANs (DCGAN) are a class of GAN architectures proposed by Radford, Metz, and Chintala in 2015 [26]. In their paper, the authors outline a set of constraints that should be adhered to in order to create GANs which can produce higher-resolution images, and whose training is more stable.

- Use strided convolutions in the discriminator and fractional-strided convolutions in the generator instead of pooling layers.
- Use batch normalisation in the generator and discriminator.
- Remove fully-connected layers.
- Use ReLU activation for all layers in the generator, except the output which should use tanh.
- Use leaky ReLU activation for all layers in the discriminator.

2.4.3 Progressive Growing GAN

Building on these ideas, progressive growing GANs (ProGAN) were introduced by Karras et al. in 2017 [27]. The fundamental idea is that instead of training all layers at once, we incrementally *grow* both the discriminator and generator during training, starting from low-resolution images up to high-resolution images. With each additional layer that is faded in, finer details at higher resolutions are able to be added. This allows for the stable training of generator models that are capable of producing high-quality images.

2.4.4 Instance Normalisation

Instance normalisation is similar in concept to batch normalisation, but we just normalise the inputs across the only the spatial dimensions for each sample *and* channel. Although batch normalisation was introduced with the intention of improving the rate of training, in 2016 Ulyanov, Vedaldi, and Lempitsky found that just by replacing batch normalisation with instance normalisation, the *visual quality* of certain image synthesis networks could be significantly improved [28].

Consider, as in Section 2.3.4, a batch of feature maps $X \in \mathbb{R}^{N \times C \times H \times W}$. We can compute the mean μ_{nc} and standard deviation σ_{nc} for each sample and channel as

$$\mu_{nc}(X) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{nchw} \tag{2.9}$$

$$\sigma_{nc} = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (X_{nchw} - \mu_{nc}(X))^2} \tag{2.10}$$

Note the differences between this and the definitions used in batch normalisation (Equation (2.6) and (2.7)). Then the instance normalisation transform for each element is given by

$$\mathrm{IN}_{\gamma,\beta}(X) = \gamma_c \left(\frac{X_{nchw} - \mu_{nc}(X)}{\sigma_{nc}(X)} \right) + \beta_c \tag{2.11}$$

where $\gamma, \beta \in \mathbb{R}^C$ are the per-channel transformation parameters.

2.4.5 AdaIN

Adaptive Instance Normalisation (AdaIN) is a conditional extension to instance normalisation introduced in 2017 by Huang and Belongie [29], meaning that it uses external information to learn the denormalisation parameters γ, β. In contrast, standard instance normalisation and batch normalisation are unconditional normalisation layers. AdaIN takes a content input X and an adaptive style input Y, and aligns the mean and variance of X with those of Y. Unlike batch normalisation or instance normalisation, its parameters are not learned but instead adaptively computed from Y.

$$\mathrm{AdaIN}(X, Y) = \sigma_{nc}(Y) \left(\frac{X_{nchw} - \mu_{nc}(X)}{\sigma_{nc}(X)} \right) + \mu_{nc}(Y) \tag{2.12}$$

Intuitively, we can understand this operation as applying the style of Y to X. Despite being first developed for style transfer, AdaIN was later adapted for a variety of other computer vision tasks.

2.4.6 StyleGAN

Introduced in 2018 by Karras, Laine, and Aila, StyleGAN combines both AdaIN and progressive growing to produce high-resolution photo-realistic images. Up until now, we have been providing a linear, feed-forward neural network generator a latent code $z \in \mathcal{Z}$ at the input layer. Instead, StyleGAN introduces a generator architecture in which a vector in the intermediate latent space $w \in \mathcal{W}$ is supplied to AdaIN layers after each convolutional layer. Figure 2.3 shows this new architecture and the role of the mapping network. The intermediate latent space $\mathcal{W}$ is learned by a mapping network, and the point in the network at which provide w is provided is correlated with how coarse or fine the transferred features are.

2.5 Conditional Image Generation

Having discussed the progression of *unconditional* high-resolution image generation, we now turn to work done in the field of *conditional* image generation. We start by looking at how GANs can be guided to generate images of a certain class, and then at the specific task of image-to-image synthesis, wherein an input image (such as a semantic label map) is transformed to an output image (such as a retinal fundus image).

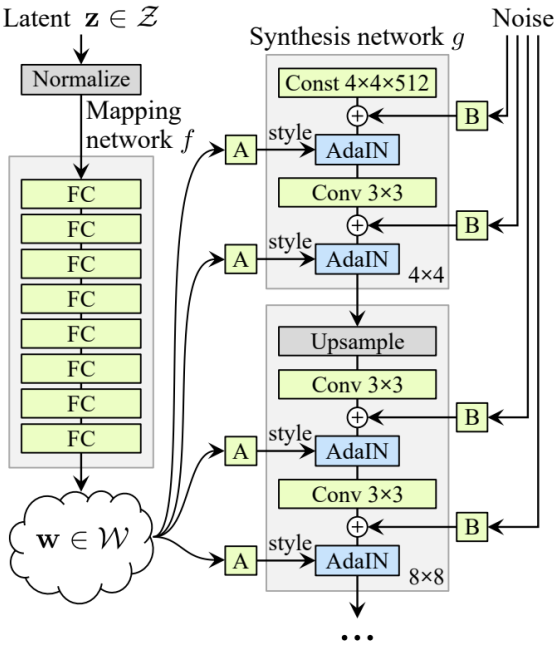


Figure 2.3: Diagram of the StyleGAN generator architecture. Taken from [30].

2.5.1 Conditional GAN

One of the earliest extensions to the original design, a conditional GAN (cGAN) extends the vanilla GAN by conditioning on additional information y, allowing us to direct the generation process [31]. Consequently, the loss function previously defined in Equation (2.1) now becomes

$$\mathcal{L}_{cGAN} = \mathbb{E}_{x,y}[\log D(x|y)] + \mathbb{E}_{y,z}[\log(1 - D(G(z|y))] \tag{2.13}$$

Today, there are four well-understood methods of conditioning the generator:

- concatenation of a vector representing the label;
- adding an auxiliary classifier;
- using projection; and
- conditional batch normalisation.

2.5.2 Pix2Pix

In 2016, Isola et al. introduced pix2pix, a conditional GAN for general purpose image-to-image translation (as opposed to class-conditional image generation), capable of producing photo-realistic images from a semantic label map [32]. First, inspired by [33], the authors added a global reconstruction loss penalty to the basic adversarial cGAN objective (Equation (2.13)), implemented using the L_1 norm.

$$\mathcal{L}_{L_1} = \mathbb{E}_{x,y,z} \left\| x - G(z|y) \right\|_1 \tag{2.14}$$

This captures the overall structure of the image, and therefore serves to prevent the generator from creating synthetic images that deviate too far from the real image. Together, this yields the overall objective function

$$\mathcal{L}_{pix2pix} = \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G) \tag{2.15}$$

where λ is a hyperparameter that balances the contribution of the two terms.

Next, they iterate on the DCGAN generator and discriminator architectures. For the generator, skip-connections are added to the standard encoder-decoder architecture, resembling that of a U-Net [34], used to directly share information across the network. Turning to the discriminator, the authors identified that since the L_1 loss from above already enforces correctness at the larger scale, the discriminator need only penalise at smaller scales. For this reason, they opted for a patch-based CNN discriminator which calculates loss for $N \times N$ patches, where N is smaller than the image size.

2.5.3 Pix2PixHD

While promising, pix2pix was only designed to generate images at resolutions up to 512×512. Attempting to apply pix2pix to higher resolution images resulted in unstable training, and blurry details in the generated images. To address these issues, in 2017 Wang et al. released pix2pixHD [35] as an extension to pix2pix.

Building on the idea of a hierarchical GAN structure, a multi-scale generator is adopted to work on images of different resolutions. In their paper, the authors choose to use two sub-networks with one acting as a "global generator" and the other a "local enhancer".

Similarly, multi-scale discriminators operating at three different scales are employed. This works slightly differently to the multi-scale generators, since instead of having a nested sub-network, three separate networks are used.

The adversarial loss defined in Equation (2.15) is improved by adding a feature matching loss. Together, we get the overall objective:

$$\min_G \left(\left(\max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k) \right) + \lambda_{FM} \sum_{k=1,2,3} \mathcal{L}_{FM}(G, D_k) \right) \tag{2.16}$$

The discriminator's objective is maximised over each of the discriminators, and that D_k does not maximimise $\mathcal{L}_{FM}$, serving only as a feature extractor. They also found that adding perceptual loss using a pre-trained VGG network slightly improved results, but was not critical. With the perceptual loss, we have the objective:

$$\begin{aligned} \min_G \Bigg(& \left(\max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k) \right) \\ & + \lambda_{FM} \sum_{k=1,2,3} \mathcal{L}_{FM}(G, D_k) \\ & + \lambda_P \min_{D_k} \sum_{k=1,2,3} \mathcal{L}_P(G, D_k) \Bigg) \end{aligned} \tag{2.17}$$

To enhance object edges, instance boundary maps are used.

2.6 Metrics

We introduce some more uncommon metrics that the may be unfamiliar to the reader.

2.6.1 Fréchet Inception Distance

The Fréchet Inception Distance (FID) [36] is a metric for assessing the "quality" and diversity of generated images by comparing the distribution of the images used to train the

generator with the distribution of generated images. Specifically, feature vectors from a pre-trained Inception-v3 network [37] are used to build the distribution $\mathcal{N}(\mu, \Sigma)$ on the generated images, and the distribution $\mathcal{N}(\mu_w, \Sigma_w)$ on real images. Then, we calculate the FID score as the Wasserstein metric between these two distributions:

$$\text{FID} = |\mu - \mu_w|^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w))^{\frac{1}{2}} \tag{2.18}$$

In essence, a smaller FID indicates a smaller distance between the two distributions. FID is not without its issues [38], however it remains the most popular GAN evaluation metric as of today.

In this work, we use Maximilian Seitzer's PyTorch implementation of the FID score[4], with feature vectors extracted from the final average pooling layer of the Inception network.

2.6.2 Cohen's Quadratic Weighted Kappa

Cohen's kappa, κ, is a metric which measures the degree of agreement between two raters who each classify N items into C mutually exclusive categories. $\kappa = 1$ indicates complete agreement, $\kappa = 0$ indicates random agreement, and $\kappa < 0$ indicates worse than random agreement. It is calculated by

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{2.19}$$

where p_o is the observed probability of agreement, and p_e is the probability of random agreement. We can also generalise this statistic to the weighted kappa, which is useful when some disagreements are more important than others. In our case, we use quadratic weighting, calculated as

$$\kappa = 1 - \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} x_{ij}}{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} m_{ij}} \tag{2.20}$$

Cohen's quadratic weighted kappa is used as the primary metric by both Kaggle DR grading competitions.

[4] https://github.com/mseitzer/pytorch-fid

Chapter 3

Related Work

We now set the research context in which to understand this project's contributions.

3.1 Retinal Image Synthesis

Early work surrounding the generation of synthetic retinal fundus images involved patch-based techniques and complex mathematical models representing the anatomy of the eye [39, 40]. Different algorithms had to be proposed for different components of the eye, relying heavily on domain knowledge.

3.1.1 Vess2Ret

More recently, purely data-driven approaches have been shown to be highly effective. In 2017, Costa et al. published their foundational work utilising GANs for retinal image synthesis, dubbed vess2ret [41]. The proposed generator network operated by performing image-to-image translation from a vessel network to a retinal fundus image. Like pix2pix, they add a global L_1 loss term, using Equation (2.15) as their loss function. In the absence of a large dataset containing manual annotations, vessel masks were inferred from the output of a segmentation model. The network architectures also resemble those of pix2pix, using a U-Net architecture for the generator and patch-based CNN for the discriminator.

However, this approach still suffers some major drawbacks. The fact that the generator relies on a vessel segmentation mask as input is unideal since the user will have to either (a) manually obtain their own vessel segmentation; (b) use a pre-existing vessel segmentation, of which there is a limited pool; or (c) infer a vessel segmentation (as the authors did) which may be unreliable. Moreover, since a vessel segmentation maps to exactly one synthetic retina, data diversity is naturally limited. The authors note that a poorly inferred vessel network, as may be produced by a segmentation model, fails to produce a usable fundus image. The output images have a resolution of 512×512 which, while large relative to the images often used in computer vision publications at the time, is small compared to those produced by modern fundus photography.

The same authors extend this existing architecture in a later work by removing the dependency on a pre-existing vessel segmentation as input [42]. Instead, an adversarial autoencoder is employed to generate vessel segmentation masks, which can be fed into the previously described image-to-image model. This requires the user to simply supply a sample from a multivariate normal distribution, which means that a theoretically unbounded number of different vessel segmentations can be generated (although this does not speak for diversity). While this overcomes the first of the issues with the previous work by obviating the need for the user to obtain a segmentation mask, the generated vasculatures are prone to displaying abnormalities, and the issue of resolution remains.

3.1.2 Tub-GAN

Zhao et al. [43] incrementally built upon this work in 2018 by introducing Tub-GAN. It resembles its predecessor in that the same loss function is used, and the generator architec-

ture is also based on U-Net, except using tanh in the output layer instead of the sigmoid function. They depart from vess2ret, however, in their discriminator design. Instead of using a patch-based CNN, a DCGAN is used, consisting of Convolution-BatchNorm-LeakyReLU blocks. The notable improvements are the fact that manual annotations are used for training, and that the generator takes a noise code which allows for stochastic variation in the outputs from a single vessel segmentation. However, in the absence of a vessel generator, it reinstates a dependency on a pre-existing vessel network annotation.

While representing an important step towards the generation of synthetic fundus images, both Tub-GAN and vess2ret lack the ability to control the generation of pathological instances. However, in the same publication, the authors also present a variant of Tub-GAN incorporating style transfer called Tub-sGAN which presents a possible solution to this limitation. In the Tub-sGAN framework, a target style is provided alongside the input segmentation. The output image is expected to posses the style of the target image, while retaining the structure dictated by the segmentation. This is much more useful since we now have the ability to transfer the texture of pathological examples onto the existing vasculature, which should include its lesions.

Nonetheless, a few shortcomings are noted. First, the generated images still exhibit failure cases when they are not anatomically correct, particularly with respect to the optic disc and macula. Furthermore, while the style-transfer appears to work well for global texture, it performs poorly in synthesising fine local details. This is problematic since many retinal lesions are extremely small. Although the authors report increased segmentation performance of vasculatures, we are interested in the more challenging task of lesion segmentation, whose ground-truths are not supplied by this network.

3.1.3 Retinal Pathological Descriptor

In 2019, Niu et al. [44] proposed an interesting "symptom transfer" network which allowed for lesion manipulation by exploiting the neuronal activations of an existing image-level DR grading network. This allows them to arbitrarily augment the position and quantity of lesions in generated images.

While interesting in concept, there is no ability to directly condition the generation on DR severity or lesion segmentations. Therefore, we consider this work orthogonal to our objectives.

3.1.4 DR-GAN

More recently, in 2020[1] Zhou et al. introduced DR-GAN [45]: a network that ensembles many of the discussed techniques and applies them to retina generation. DR-GAN draws heavily from recent developments in image-to-image translation by incorporating feature matching loss; multi-scale generators and discriminators; instance boundary maps; spatial and channel attention; and adaptive instance normalisation. By leveraging these methods, DR-GAN overcomes many of the limitation of prior work, taking into account image-level grade information, pixel-wise lesion labels, and optic disc instance masks.

The authors acknowledge that, due to the extremely small size of lesions, synthesis of fundus images must be done at a high resolution to be useful. To deal with this, they employ multi-scale generators and discriminators similar to those used in pix2pixHD. In order to preserve very fine features, they use a spatial and channel attention module (SCA) in the synthesis blocks inspired by [46], which in turn is inspired by [47]. Recalling the

[1]The 2020 work is an extension of preliminary work published in 2019. We consider the extended work only, since it supersedes the preliminary work.

techniques used in StyleGAN, each of the synthesis blocks also contains an AdaIN layer which gives the ability to manipulate the grade of the generated images using what the authors call "adaptive grading vectors". The latent grading space is first learnt using the ResNet-50 network. From this, a set of normal distributions representing the grading space are computed. During synthesis, the appropriate distribution is sampled from and injected into the AdaIN component of synthesis blocks. In this way, the "style" of the sampled vector is transferred to the generated image. They extend the pix2pixHD loss function by adding a classification loss $\mathcal{L}_C$, specifically focal loss [48], to combat class imbalance. They use VGG-19 as their perceptual network.

$$\begin{aligned}\min_G \Bigg(\Bigg(\max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k) \Bigg) \\ + \lambda_{FM} \sum_{k=1,2,3} \mathcal{L}_{FM}(G, D_k) \\ + \lambda_P \min_{D_k} \sum_{k=1,2,3} \mathcal{L}_P(G, D_k) \\ + \lambda_C \min_{D_k} \sum_{k=1,2,3} \mathcal{L}_C(G, D_k) \Bigg)\end{aligned} \tag{3.1}$$

DR-GAN represents the state-of-the-art at the time of writing, incorporating some of the most recent techniques in computer vision. The authors are able to demonstrate quantitatively the improvements in image quality over previous methods, and briefly show how synthetic data can improve classification performance. However, the effect of synthetic data on segmentation performance is not discussed. The model is still held back by limitations in the training data, and hence still exhibits a number of failure cases. DR-GAN still relies upon inferred structural and lesion segmentation masks to bootstrap its training due to the lack of manually annotated data.

One concern that is unclear from the paper is how these adaptive grading vectors interact with the lesions of the input mask. For instance, if a lesion mask exhibiting proliferative DR is given to the network, and during synthesis adaptive grading vectors corresponding to a healthy retina are provided, what will the resulting output resemble? And will the lesion mask still be useful if the lesions have been manipulated? Without the source code, these questions remain unanswered.

3.2 Semantic Label Generation

Meanwhile, literature on the task of semantic label generation is relatively sparse. Wang and Gupta [49] first introduced this factorisation into "structure" and "texture" in 2016 when attempting to first generate normal maps which encode the geometry of the scene, and from this generating a realistic 2D image. Their "Structure-GAN" generates small $72 \times 72 \times 3$ images, and the generator is deeper than the discriminator. They also apply joint fine-tuning of both networks together, after training each one independently.

Later, in 2019, Ghelfi et al. [50] introduced a variant of the GAN where the last layer is simply replaced by a softmax layer. The authors describe this as a "more principled" approach to generating semantic images, as opposed to treating them like standard RGB images, as in Wang and Gupta's earlier work. However, beyond that the papers insights into the challenges unique to this domain are limited.

More recently, in 2020, Volokitin, Konukoglu, and Van Gool [51] went on to apply a similar technique to demonstrate how the generation of natural images could be separated into

the distinct steps of layout prediction and texturing. As opposed to improving segmentation performance, they were instead motivated by how this decomposition could improve the plausibility of multi-object scenes. To generate semantic labels they used a DCGAN architecture, and a pix2pix model for image texturing.

Concurrently, Azadi et al. [52] introduced a method in which the two components are trained independently at first, then fine-tuned by training in an end-to-end fashion, resembling the aforementioned Structure-GAN by Wang and Gupta. This approach was dubbed the "Semantic Bottleneck GAN", or SB-GAN. The authors note how the task of learning the interactions between structural elements to generate discrete labels is "extremely challenging". However, the focus of their work was generating high-fidelity images, and did not study the effect on downstream tasks. Moreover, the end-to-end fine-tuning was done by introducing an additional, unconditional discriminator, trained to distinguish real RGB images and those generated from synthetic layouts. This is likely to weaken the conditioning between the semantic labels and the output image, potentially making this approach inappropriate for our use case since we would like to use the semantic labels as segmentation maps.

Moreover, none of these approaches attempt class-conditioning of semantic labels, nor do they evaluate the effect of synthetic data on downstream tasks. A further complication is that the semantic labels of interest to us are far sparser than those used in the datasets of previous work.

3.3 Training on Synthetic Data

There already exists a history of synthetic data already being used to train machine learning models; however this work has largely been to do with using 3D scenes to train 2D computer vision algorithms ([53] provides a good overview). Regardless, this has built a case that synthetic data could result in a real-world improvement. As early as 2016, Gaidon et al. [54] showed that pre-training on synthetic data and subsequent fine-tuning on real data could improve the performance of computer vision algorithms.

Even closer to our mixed-data approach, Shrivastava et al. [55] showed that training on a mixture of real and synthetic data improved performance of pose estimation models, compared to training on only real data.

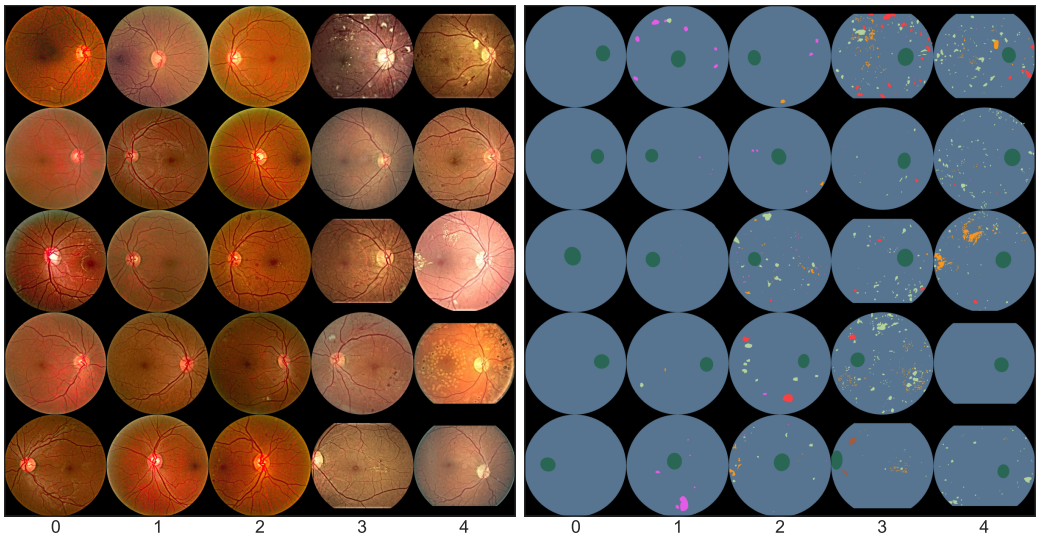
Chapter 4

Data

In this chapter, we survey the available datasets and examine how their characteristics may influence our models. For our work to be of any practical use, it is imperative that we collect data that is as diverse as possible, representing a wide range of imaging conditions and patient backgrounds. While large datasets for image-level graded images are available, the lesion segmentation task suffers from a severe lack of pixel-wise segmented data, and even fewer provide both grading and semantic label information.

In the field of data science, fully understanding the properties of your data domain is just as imperative as model development. Exploratory data analysis provides the necessary context and insights to choose the appropriate modelling techniques and put forth an informed interpretation of one's results.

4.1 FGADR



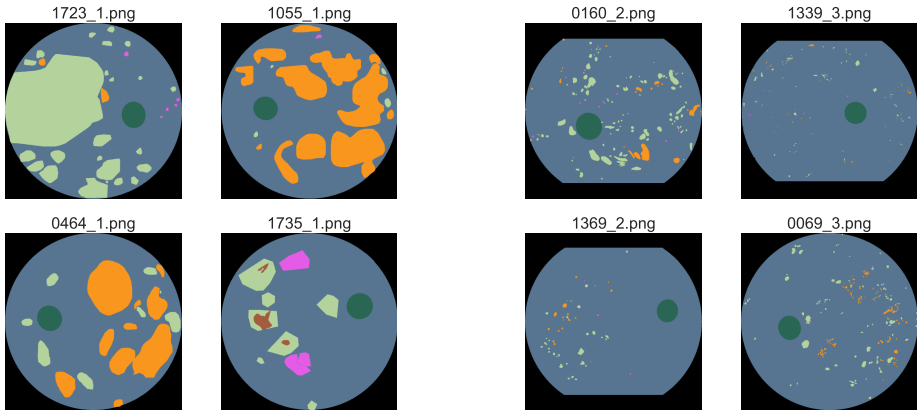
(a) Retinal fundus images. (b) Corresponding semantic labels.

Figure 4.1: A sample of images from the FGADR dataset and accompanying manual semantic annotations. Columns from left to right correspond to DR grades 0 through 4.

The Fine-Grained Annotated Diabetic Retinopathy (FGADR) dataset [56] was released in 2020 alongside DR-GAN, and consists of two subsets. The first, called the *Seg-set*, contains 1842 images with both image-level annotations for DR severity, and pixel-level annotations for microaneurysms, haemorrhages, hard exudates, soft exudates, intraretinal microvascular abnormalities, and neovascularization. Optic disc masks are not provided, so manual annotations were collected for this project (provided in Appendix A.2). The second, called the *Grade-set*, contains 1000 images with only image-level DR grades. It is useful for its high annotation confidence, however the *Grade-set* is still pending approval by the researcher's legal department at the time of writing, and therefore is regrettably not available for use in this project. All images in both sets have a resolution of 1280×1280.

This is the only pixel-wise annotated dataset that includes labels for neovascularization and intraretinal microvascular abnormalities, which is of particular significant since according to Table 2.1, NV and IRMA are key to differentiating stage 3 from stage 4.

Figure 4.1 shows a sample of images and associated annotations from the FGADR dataset. Visually, the relationship between the prevalence of lesions and DR grade is clear. Note that overlap of semantic labels is possible with the images. We resolve these arbitrarily, based on the index of the label (according to Table 4.4). The photographs themselves are noisier than those from the other datasets, which can likely be attributed to the illumination conditions at the time of imaging.

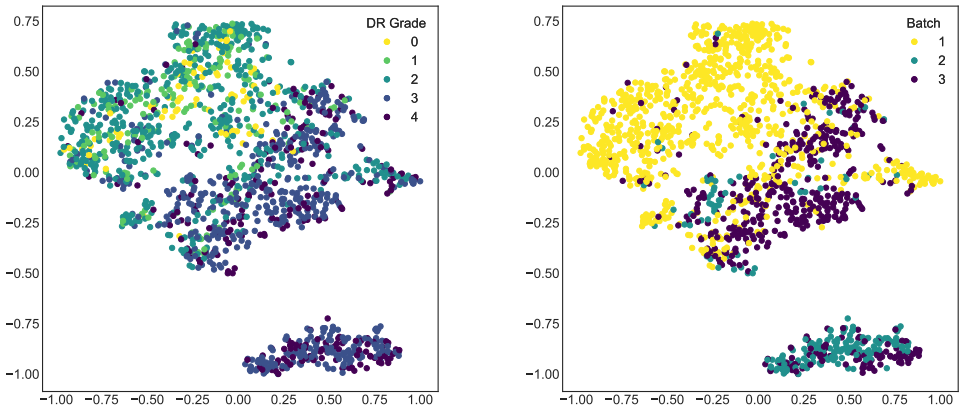


(a) Sample of images from batch 1. (b) Sample of images from batches 2 and 3.

Figure 4.2: Comparison of annotator bias between batches.

Interestingly, a visual examination of the annotations reveals a number of samples which are incongruously annotated compared to the other samples, likely due to annotator bias. These coarsely annotated files all share a file suffix, following the pattern `XXXX_1.png`, a selection of these are shown in Figure 4.2. While not entirely clear from the dataset's descriptor paper, this may indicate the grader, hospital, or something else. This suffix value can be one of `1`, `2`, or `3`, and we'll refer to this number as the "batch". Apart from the style of annotation, another difference is that this group of files do not have the top and bottom section of the retina cropped, while the other groups have a mix. One way of removing this bias would be to exclude all 1000 images which contain this suffix. However, this is not viable for three reasons: first, this batch makes up over half of the dataset so we would be drastically limiting the amount of available data; secondly, not all of these images suffer from coarse annotations; and finally, this batch represents the entirety of images with grades 0, 1, and 2, removing them would heavily imbalance our data. Instead, we manually review each of the images in this batch and list those that are determined to be "too coarse" (by a non-expert and subjective measure). We call this the "FGADR exclusion list", and can be found in Appendix A.1.

To deepen our understanding the relationships present in the data, we can apply a dimensionality reduction technique such as t-SNE to create a visual representation which may reveal underlying patterns. We use the ResNet-50 network as a feature extractor by training it to predict the DR grade of semantic labels from the FGADR dataset. Note that, while pre-trained networks (on datasets like ImageNet) are commonly used for feature extraction, we decided to train from scratch since our data domain bears little resemblance to those used to train general-purpose networks. Once trained, we can then extract the



(a) Coloured by DR severity grade.

(b) Coloured by batch number.

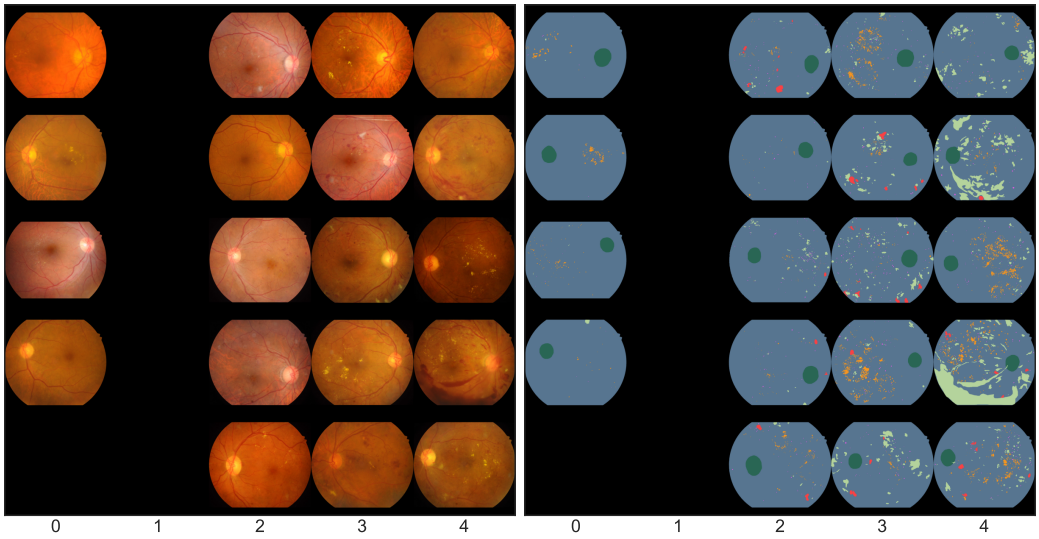
Figure 4.3: t-SNE applied to feature vectors extracted from ResNet-50 trained on FGADR labels.

learned feature vectors from the layer immediately prior to the final fully-connected layer. During training, the network was able achieve to 0.86 accuracy on the training set, and 0.53 accuracy on the validation set, which is an indication that there is at least some predictive power in the images. This relationship is confirmed visually by examining the result of running t-SNE on these feature vectors, shown in Figure 4.3a. There is a distinct separation between images which have a high DR severity and low DR severity, even if the clusters are not particularly well-defined.

However, given the discrepancy between image batches we identified earlier, it's possible that the features being calculated are more to do with the batch. To explore this possibility, we plot the same t-SNE components, but this time colour the points by the batch number, depicted in Figure 4.3b. There are two factors at play here: difference in features between batches, and grade distribution. For instance, as mentioned earlier, all images with grade 0, 1, and 2 belong to batch 1, and this is reflected in the dimensionality reduction. The isolated cluster in the lower-right corner is an example of the latter factor, and consists of images which have the top and bottom sections cropped. While interesting, we expect that there is sufficient diversity within the dataset, and the final training dataset will further combine multiple other datasets, and hence does not pose an obstacle.

4.2 IDRiD

The Indian Diabetic Retinopathy Image Dataset (IDRiD) [57] was initially released in 2018 as part of the "Diabetic Retinopathy: Segmentation and Grading Challenge" held at ISBI-2018. Similarly to the FGADR dataset, the IDRiD dataset also consists of multiple subsets. The first subset, aimed at training segmentation models, contains 81 retinal fundus images with varying levels of DR, each of which are accompanied by pixel-level annotations for the optic disc, microaneurysms, soft exudates, hard exudates, and haemorrhages. Unlike FGADR, annotations for neovascularization, intraretinal microvascular abnormalities, or image-level DR grades for these images are not provided. The second subset, aimed at training classification models, consists of 516 images annotated with DR severity grades only. The images in the second subset are exclusive with those in the first subset, and thus



(a) Retina fundus images. (b) Corresponding semantic labels.

Figure 4.4: A sample of images from the IDRiD dataset and accompanying manual semantic annotations. Columns from left to right correspond to *inferred* DR grades 0 through 4.

none of the images in this dataset contain both image-level and pixel-level annotations, as in the FGADR dataset. Each image has a resolution of 4288×2848.

To obtain inferred DR grades on this dataset for weakly-supervised learning, we train a model to predict the DR grade from the retinal fundus images. This method is given in greater detail in Section 4.8. The distribution of the resulting grades has relatively few grade 0s since all images exhibit some form of DR. There are no grade 1 images, which can likely be attributed tsto bias in the grade inference network.

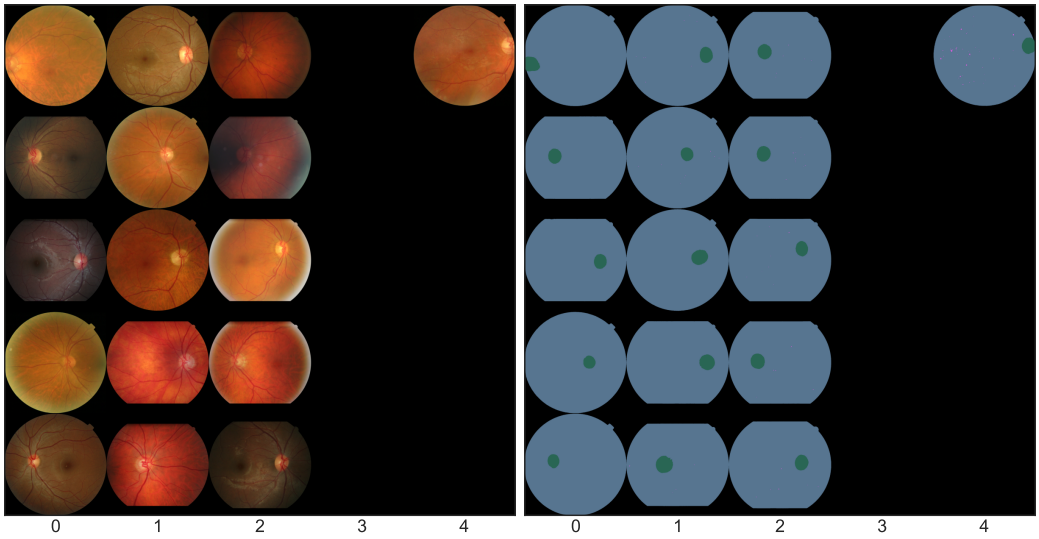
4.3 E-Ophtha

The e-ophtha[1] [58] dataset was released in 2013 and contains pixel-level annotations for exudates (it is not specified whether these are hard or soft exudates) and microaneurysms. The images are provided as two subsets: e-ophtha-EX and e-ophtha-MA, each of which also contains a number of "healthy" images which do not exhibit the respective lesion (but may have other lesions). In total, 26 images are annotated with exudates only, 128 are annotated with microaneurysms only, 21 are annotated with both, and 269 images are not annotated with any kind of lesion. Image resolution varies from 1440×960 to 2544×1696. A sample of images is shown in Figure 4.5; notice how the density of annotations here is much less than those in the FGADR and IDRiD datasets. This can be largely attributed to the fact that fewer lesion types are annotated, as well as the annotations simply being more fine-grained. For this dataset, instead of manually annotating optics discs, we infer them using a model as described in Section 4.9.

4.4 EyePACS

The EyePACS dataset was first released in 2015 for the Kaggle Diabetic Retinopathy Detection challenge [59]. It is the largest dataset of retinal fundus images for the purpose of grading the severity of diabetic retinopathy, consisting of 88,702 images graded from 0 to

[1] Sometimes (mis-)spelled "e-optha" in the dataset files.

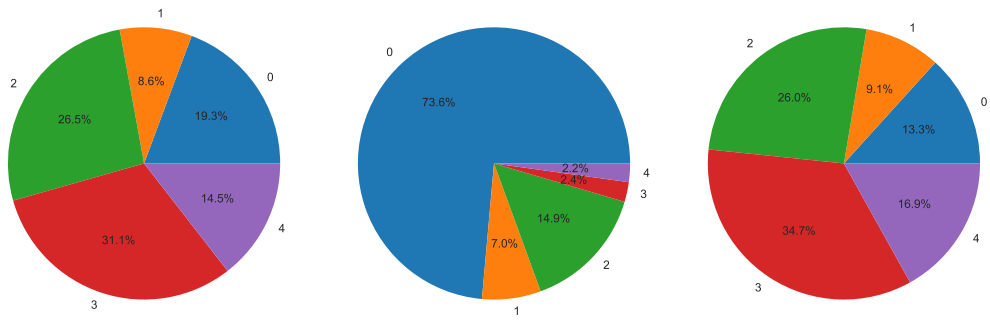


(a) Retina fundus images. (b) Corresponding semantic labels.

Figure 4.5: A sample of images from the e-ophtha dataset and accompanying manual semantic annotations. Columns from left to right correspond to *inferred* DR grades 0 through 4.

4. Pixel-level annotations are not provided. Despite the large volume of data, the quality of images is highly variable, particularly compared to the other datasets we are using, which appear to be more closely curated. Moreover, the labels are highly imbalanced, with 73.5% of images being grade 0, and just 2.0% being grade 4. For the purpose of the original competition, public/private training and testing sets are provided, but we disregard these and instead create our own subsets.

4.5 Data Split



(a) Segmentation dataset. (b) EyePACS dataset. (c) Grading dataset.

Figure 4.6: Distribution of DR grades in various datasets.

We combine the FGADR, IDRiD, and e-ophtha datasets to train our semantic label generation models. In order to prevent data leakage, we randomly separate out (and fix) training, validation, and test sets, according to Table 4.1. We train all generative models on the training set, compare models using the validation set, and use the test set for the final evaluation.

For the EyePACS training set we simply take a training and test split, according to Table 4.2.

Subset	Proportion	# Images			
		FGADR	IDRiD	e-ophtha	Total
Training	0.80	1194	61	352	1607
Validation	0.10	151	13	37	201
Test	0.10	149	7	45	201
Total	1.00	1494	81	434	2009

Table 4.1: Data split for segmentation data.

Subset	Proportion	# Images
Training	0.80	70,961
Test	0.20	17,741
Total	1.00	88,702

Table 4.2: Data split for the EyePACS dataset.

We also create a small grading set by combining samples from FGADR *Seg-set* and the IDRiD grading set, shown in Table 4.3. We use the same FGADR training set from the segmentation split to avoid data leakage. This is not a concern for the IDRiD dataset since the IDRiD segmentation and grading datasets are mutually exclusive.

Subset	Proportion	# Images		
		FGADR	IDRiD	Total
Training	0.80	1194	412	1606
Validation	0.10	151	52	203
Test	0.10	149	52	201
Total	1.00	1494	516	2010

Table 4.3: Data split for grading data.

4.6 Other Datasets

Other datasets for diabetic retinopathy related tasks are available, but were inappropriate for our use case. Specific reasons are given below:

DiaRetDB0

Released in 2006 as part of the ImageRet[2] project. DiaRetDB0 does not have pixel-level annotations, only image-level annotations labelling the presence of lesions. DiaRetDB1 and DiaRetDB1 v2 were also released by the ImageRet project.

DiaRetDB1

Contains 89 images at a resolution of 1500×1152, 84 of which show signs of DR and five which are healthy. Images are accompanied by pixel-wise labels for hard exudates, soft exudates, microaneurysms, and "red small dots" – which we interpret as haemorrhages. Labels are not provided for neovascularization or intraretinal microvascular abnormalities, nor are image-level DR grades. Interestingly, each image was annotated independently by four different experts, and a confidence level

[2] https://www.it.lut.fi/project/imageret

was provided for each marking. Unfortunately, after running some experiments, we determined that the annotations are too coarse to be used as segmentation maps. However, we still collected manual optic disc annotations for these images, also provided in Appendix A.2.

DiaRetDB1 v2
Contains the same data as DiaRetDB1 but in a more flexible data format.

DRIVE
Contains 40 images with pixel-level segmentations for vessels only.

CHASE_DB1
Contains 28 images with pixel-level segmentations for vessels only.

STARE
Contains 400 images with pixel-level segmentations for vessels only.

DRISHTI-GS
Contains 101 images with pixel-level segmentations for the optic disc only.

Messidor
Contains 1200 images graded by their risk of diabetic macular oedema, instead of diabetic retinopathy.

4.7 Data Pre-Processing

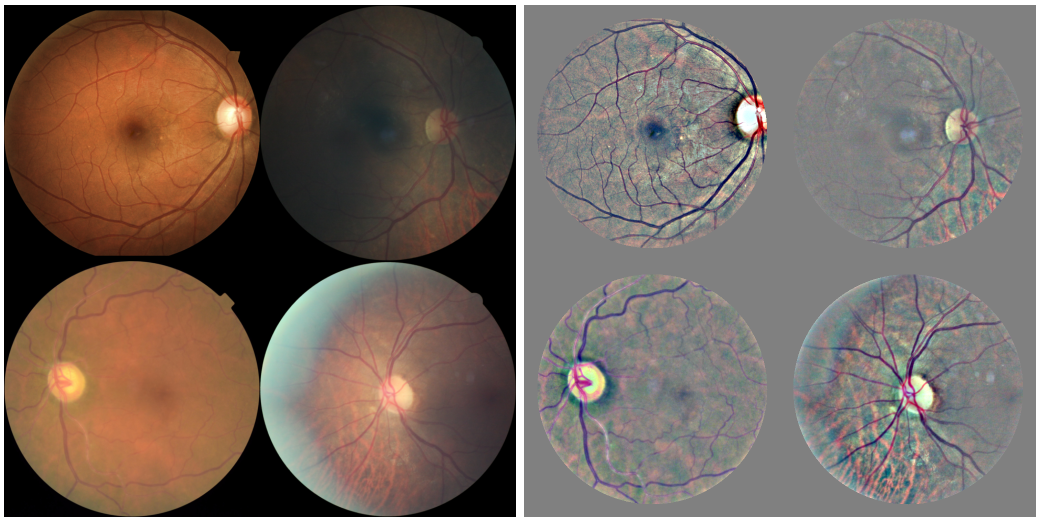


Figure 4.7: Before (left) and after (right) illumination correction.

To ensure image size and retina positioning is uniform across all images, we perform the following steps:

1. Extract a mask for the retinal fundus boundary by performing a binary threshold.
2. Draw a bounding box around this boundary.
3. Crop around the bounding box so that the retina is now centred and fills the image.
4. Pad to square.
5. Resize to 1280×1280.

To further correct for illumination differences between retina photographs, we borrow image processing techniques from Benjamin Graham [60], winner of the 2015 Kaggle Diabetic Retinopathy Challenge[3]:

6. Subtract the local average.
7. Clip the radius to remove boundary effects.

Each pixel-annotated dataset provides binary masks for each lesion class. The segmentation masks for each class are combined into a single-channel image, where the discrete value of each pixel encodes its label. That is, each retina semantic label with height H and width W is represented as an image $M \in \mathbb{L}^{H \times W}$ where $\mathbb{L} = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, ensuring that every pixel belongs to exactly one class. For clarity, the semantic labels have been mapped to RGB colours according to Table 4.4.

Label	Value	RGB Colour
Retina	0	(87,117,144)
Optic Disc	1	(41,102,84)
Microaneurysms	2	(228,92,229)
Haemorrhages	3	(180,211,156)
Hard Exudates	4	(248,150,30)
Soft Exudates	5	(249,65,68)
Neovascularization	6	(250,208,44)
Intraretinal Microvascular Abnormalities	7	(164,92,64)
Background	8	(0,0,0)

Table 4.4: Semantic label values and colour mappings.

4.8 Grade Inference

To infer grades, a ResNet-101 network was trained for 20 epochs on the EyePACS dataset, with images pre-processed as detailed in the previous section. We use this over the smaller grading dataset since the EyePACS dataset contains a much greater volume of data with more diversity. Images were resized to 512×512 before being passed through the network. Training was done on on one Nvidia GeForce GTX 1080 GPU over 48 hours. The data was split according to Table 4.2, and the model was trained with the Adam optimiser, $\alpha = 0.001$ and batch size 8.

To accelerate training on such a large dataset, the images were first pre-processed and saved to disk in the HDF5 format, allowing for more efficient data retrieval. Doing this allowed data to be loaded up to 12.54 times faster (at the cost of storage space). However, training was largely bottlenecked by GPU processing speed instead of disk I/O, so this did not translate to real-world training improvement.

For data augmentation, we randomly rotate between 0 and 360 degrees, randomly translate by between (-10%, +10%) vertically and horizontally, and randomly shear parallel to the x axis in the range (-0.2, +0.2).

We also experimented with test-time augmentation, but did not find that this improved the model's performance, and incurred a significant performance penalty. The evaluation metrics on the test set are reported in Table 4.5.

The trained model can be found in Appendix A.3.

[3] https://www.kaggle.com/c/diabetic-retinopathy-detection

Configuration	Accuracy	Precision	Recall	F_1	κ
ResNet-101	0.8208	0.6574	0.3836	0.4845	0.6309

Table 4.5: Performance of the grade inference model on the EyePACS test set.

4.9 Optic Disc Inference

To infer optic discs, we train a basic U-Net architecture on the pre-processed retinal fundus images to predict the optic discs provided by the IDRiD dataset and manual annotations collected on the training subset of the FGADR and DiaRetDB1 datasets. For the FGADR dataset, we include the coarsely annotated images since they have no bearing on the optic disc annotations. As data augmentation, we apply custom transformations during training to vertically and horizontally flip the images and masks jointly. Images were resized to 512×512.

We train using the Adam optimiser with a $\alpha = 0.001$ and batch size 8 for 50 epochs on one Nvidia Titan X Pascal GPU for 1 hour and 21 minutes. The evaluation metrics on the test set are reported in Table 4.6.

Configuration	Precision	Recall	F_1
U-Net	0.9484	0.9657	0.9561

Table 4.6: Performance of the optic disc inference model on the semantic label test set.

After inference, there were 11 spurious annotations which had to be corrected manually using the GIMP[4] image editor.

The trained model can be found in Appendix A.2.

[4] https://www.gimp.org

Chapter 5

Semantic Label Generation

The main objective of this work is concerned with predicting plausible layouts of retina structures and lesions. We first approach this by using a GAN to generate semantic labels, trained on manual annotations. By using a sophisticated generative model, we hope to be able to capture subtle relationships and interactions between the various different retinal structures, which would not be possible with classical data augmentation techniques. Then we study how a heuristic-based method that samples from existing data in a "naive" way compares with the generative models.

In this section, we discuss the unique challenges of this task, and go on to compare three different approaches.

5.1 Initial Experiments

During the initial exploration phase for this task, we began by applying implementations of the most modern GAN designs, such as ContraGAN and StyleGAN. To do this, we had to modify the generator and discriminator architectures to suit our problem domain, instead of creating RGB images. Recall our semantic *images* are represented as $M \in \mathbb{L}^{H \times W}$, where $\mathbb{L} = \{0, ..., 9\}$. To extract the one-hot encoded (w.r.t each pixel) semantic *label* representation, we extract the appropriate labels from this image, to get the desired input $S \in \{0, 1\}^{C \times H \times W}$. The final layer of the generator becomes a softmax layer, which allows us to interpret each pixel of the output feature map $P \in \mathbb{R}^{C \times H \times W}$ as the probability of that pixel belonging to each class; that is:

$$\forall h \in H,\ \forall w \in W \quad \sum_{c}^{C} P_{chw} = 1 \text{ and } P_{chw} \geq 0 \tag{5.1}$$

To recover the semantic image representation, we simply take the arg max of this probability map over the channels.

$$M = \arg\max_{c} P \tag{5.2}$$

This relationship between these representations is depicted graphically in Figure 5.1.

However, we quickly discovered that these architectures – primarily concerned with creating high-fidelity, dense, natural, 3-channel images – collapsed almost immediately, before generating any useful outputs. In particular, we found that GAN architectures which utilised a ResNet-based discriminator were particularly problematic, even when the capacity of the generator, in terms of the number of parameters, was much greater. Interestingly, this was not reported by any of the (admittedly limited) existing literature, which largely use architectures based on conventional CNNs.

We theorise that this is because the sparse nature of the semantic labels, coupled with the large number of channels, made it a challenge for the generator to create plausible outputs, while the discriminator learnt very quickly to distinguish real and fake images. Hence, it was established that we would need to create a bespoke architecture starting from first principles, based on an understanding of what made the task challenging, and how these issues could be mitigated.

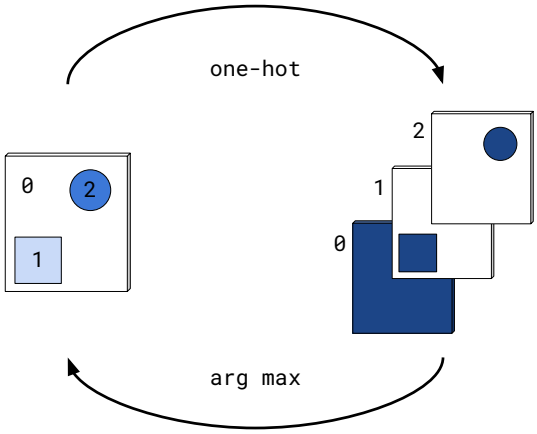


Figure 5.1: Converting between the integer-encoded semantic image and one-hot encoded semantic label representation.

5.2 ACGAN

Motivated by this, we designed a DCGAN-based generator and discriminator architecture to generate class-conditioned semantic labels, using auxiliary discriminator losses (ACGAN). In this conditioning strategy, we add a classification layer to the end of the discriminator which learns to predict the class of the input image [61]. As the generator attempts to minimise this auxiliary loss, the quality of conditioning will increase.

The generator creates images at a resolution of 256×256; we found that attempting to use this architecture to generate images at higher resolutions than this greatly increased instability, since discriminability increases with resolution. To condition the generator, we take the Hadamard product between the latent noise vector z and the embedded representation of the image class. Generated images are upsampled to 512×512 using bilinear interpolation (and then thresholded to valid values) as a post-processing step.

Where this design departs from conventional ACGANs, as presented in their original form, is the asymmetry between the generator capacity (2,672,630 parameters) and discriminator capacity (259,070 parameters). Limiting the discriminator in terms of both the number of parameters and other training techniques, detailed in the following sections, were key in allowing for stable training.

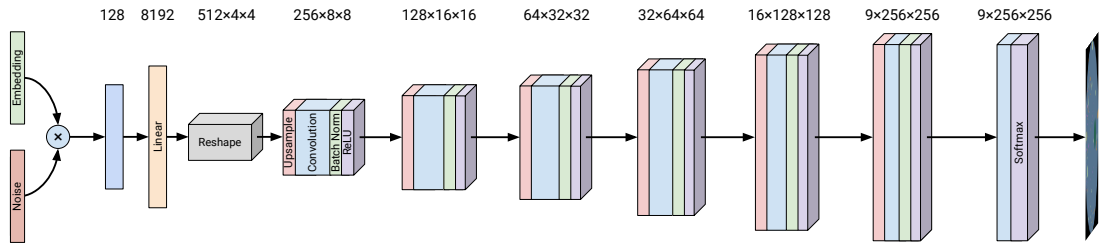
5.2.1 Optimiser

We use the Adam optimiser with $\alpha_G = 0.0005$ and $\alpha_D = 0.0001$ with $\beta_1 = 0.5$, $\beta_2 = 0.999$. Two generator training iterations are performed for each discriminator training iteration. This imbalanced learning allowed the generator to keep up with the discriminator, and was critical to prevent training from collapsing.

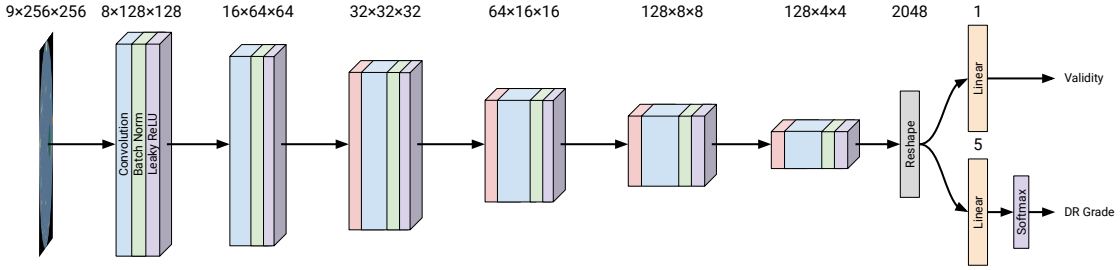
5.2.2 One-Sided Label Smoothing

Label smoothing is a technique where the target values of 1 and 0 are replaced by "smoothed" labels such as $\alpha = 0.9$ for the positive value and $\beta = 0.1$ for the negative value. This yields the new optimal discriminator

$$D(x) = \frac{\alpha p_{data}(x) + \beta p_{model}(x)}{p_{data}(x) + p_{model}(x)} \tag{5.3}$$



(a) ACGAN generator architecture.



(b) ACGAN discriminator architecture.

Figure 5.2: ACGAN network architectures.

However, when $\beta \neq 0$, $p_{data} \approx 0$, and p_{model} is large, erroneous samples from p_{model} have no incentive to move nearer to the data. Hence, we smooth only the positive labels to $\alpha = 0.9$, and leave the negative examples as 0 [13]. This has the effect of penalising the discriminator for being overconfident.

5.2.3 Loss Functions

For the discriminator, we use the Hinge loss [62]. Inspired by its use in SVMs, it aims to find a separating hyperplane between real and fake samples. The generator attempts to decrease the margin, whereas the discriminator attempts to increase it. This allows for strong gradients, even with a confident discriminator.

$$\mathcal{L}_{adv}^{D} = -\mathbb{E}_x[\min(0, -1 + D(x)] - \mathbb{E}_z[\min(0, -1 - D(G(z)))] \tag{5.4}$$

To implement the ACGAN, we also use auxiliary losses.

$$\mathcal{L}_{aux}^{D} = \mathbb{E}[\log P(y|x)] + \mathbb{E}[\log P(y|G(z|y)] \tag{5.5}$$

For the generator, we use the comparatively simple Wasserstein loss.

$$\mathcal{L}^{G} = -\mathbb{E}_z[D(G(z))] \tag{5.6}$$

5.2.4 Adaptive Discriminator Augmentation

Adaptive Discriminator Augmentation (ADA) [63] is a technique developed to prevent the discriminator from overfitting when training a GAN on limited data. Various transformations are applied to all images seen by the discriminator (real and fake) with probability

p, where p is determined by the degree of overfitting exhibited by the discriminator. A simple heuristic is used to determine how much the discriminator is overfitting:

$$r_t = \mathbb{E}[\text{sign}(D_{train})] \tag{5.7}$$

On each training iteration, p is incremented or decremented by a fixed value depending on the value of r_t.

We found that ADA was crucial to prevent discriminator overfitting. Typically, GANs are trained with many thousands of images, which provides enough diversity to prevent overconfidence in the discriminator. However, since our training set is small, it is easy for the discriminator to overfit relatively early on in training. The transforms we use are probabilistic rotations, affine transforms, and Gaussian noise.

For the rotation, we apply a 90 degree rotation up to 4 times, as detailed in the original paper. When rotating, the background is filled by setting the background channel to 1 and all other channels to 0.

Listing 5.1: Python-like pseudocode for probabilistic rotation.

```
def probabilistic_rotate(x: Tensor, p: float) -> Tensor:
    """
    Rotates a tensor x of shape (C, H, W) with probability p.
    """
    if random() > p:
        return x

    # Randomly choose a number of rotations.
    num_rotations = random_choice([0, 1, 2, 3, 4])

    fill = get_fill_colour(x)

    # Rotate by multiples of 90 degrees.
    x = rotate(x, 90 * num_rotations, fill=fill)
    return x

def get_fill_colour(x: Tensor) -> List:
    # Only fill the background to 1s.
    n_channels, _, _ = x.shape
    fill = [1] + [0 for _ in range(n_channels - 1)]
    return fill
```

For the affine transformation, we perform a random scale, translation, and (fine-grained) rotation. By "random", we mean that the parameters of the transformation itself are also sampled randomly from the specified ranges, in addition to the probability p of the ADA transformation being applied in the first place.

Listing 5.2: Python-like pseudocode for probabilistic affine rotation.

```
def probabilistic_affine(x: Tensor, p: float) -> Tensor:
    """
    Performs a random affine transformation on tensor x of shape (C, H, W)
        with probability p.
    """
    if random() > p:
        return x
```

```
    fill = get_fill_colour(x)

    x = random_translate(x, (0.5, 0.5), fill=fill)
    x = random_scale(x, (0.8, 1.2), fill=fill)
    x = random_rotate(x, 360, fill=fill)
    return x
```

For the Gaussian noise, we also scale the standard deviation of the noise with the degree of overfitting, up to a specified maximum.

Listing 5.3: Python-like pseudocode for probabilistic Gaussian noise.

```
MEAN = 0
MAX_STD = 1.0

def probabilistic_noise(x: Tensor, p: float, mean: float, max_std: float)
    -> Tensor:
    """
    Adds noise to a tensor x of shape (C, H, W) with probability p.
    """
    if random() > p:
        return x

    # Scale the standard deviation of the noise based on the probability
    # i.e. degree of overfitting.
    std = p * MAX_STD
    noise = MEAN + normal_distribution(x.shape) * std
    return x + noise
```

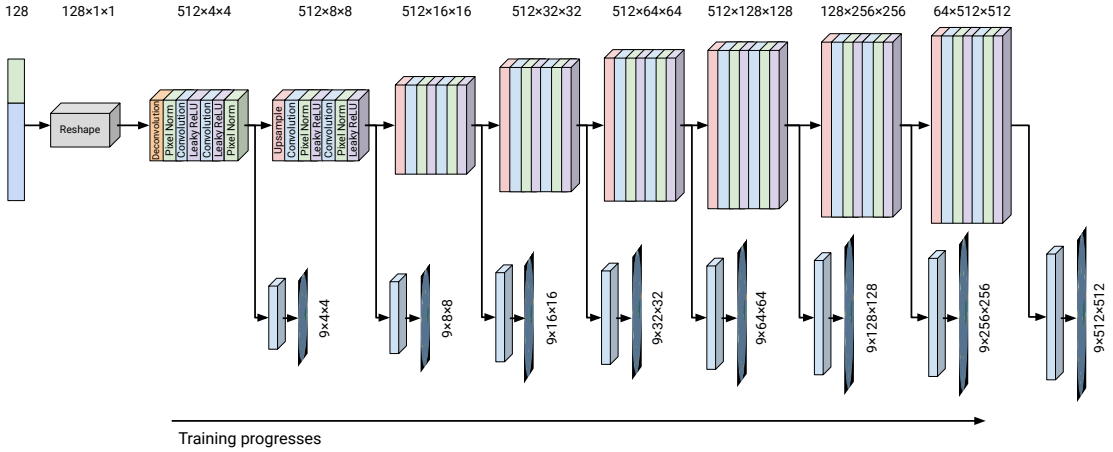
We set 0.6 as the target value for r_t, and p is capped at a maximum of 0.85, as the authors suggest that past this point the transforms will start to "leak" – meaning that they will begin to occur in the generated images. The value of p is adjusted by by 0.01 on each batch iteration.

5.2.5 Weight Initialisation

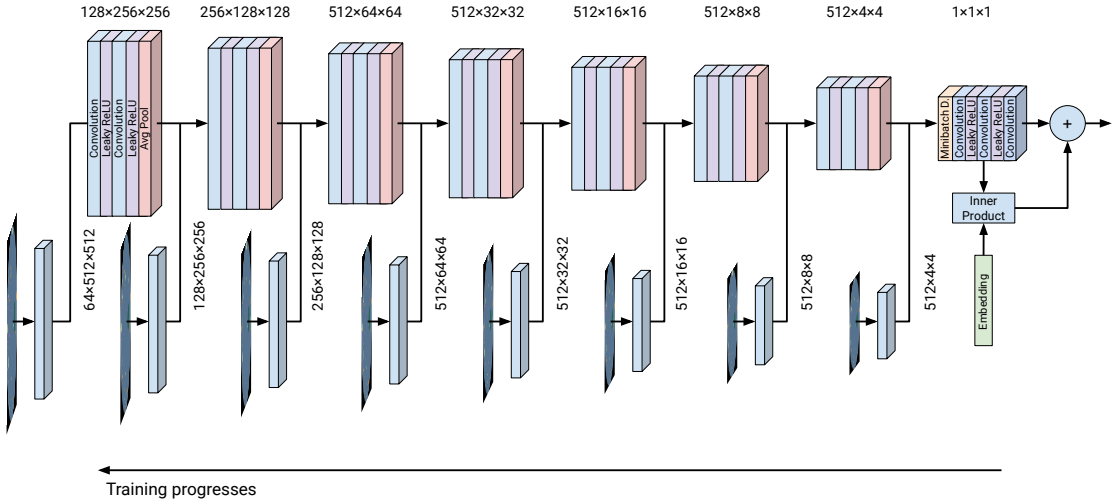
Weights are initialised using a normal distribution $\mathcal{N}(0, 0.02)$. We also experimented with orthogonal initialisation, but found it was more prone to collapse.

5.3 ProGAN

As discussed, much of the instability we experienced when attempting to train semantic label generation GANs was owing to the discriminator's confidence, and the generator being unable to "catch up". We also know that higher resolution images are more discriminable than lower resolution images. Hence, it's intuitive that by starting at lower resolutions and progressively growing the generator and discriminator – and therefore the resolution at which they operate – the networks will be appropriately balanced throughout training. Indeed, we found this to be the case, and that by allowing the generator to start with very small resolutions, the use of a ResNet-based discriminator was possible. The generator and discriminator capacities are also very balanced, particularly in comparison to the ACGAN, with the generator having 24,639,137 parameters and the discriminator 24,646,209.



(a) ProGAN generator architecture.



(b) ProGAN discriminator architecture.

Figure 5.3: ProGAN network architectures.

We use a ProGAN architecture to generate images starting from a resolution of 4×4 up to 512×512. The generator and discriminator architectures are depicted in Figure 5.3, with each additional block being capable of generating or discriminating images at a greater resolution. In order to support variable sized images during training, we use `toRGB` blocks in the generator, which are simply 1×1 convolutions that reduce the number of feature maps down to the nine required in the output. This allows us to create images at different resolutions while retaining the parameters learned in the prior hidden layers. Conversely, `fromRGB` blocks are used in the discriminator to increase the number of feature maps.

We retain many of the same techniques detailed above, including label smoothing, the same loss function, and ADA. However, we found that techniques that we relied upon with the fixed-resolution ACGAN, such as ADA, had a significantly weaker effect here. We also incorporate some of the improvements detailed in the original ProGAN paper such as equalised learning rates and pixelwise feature vector normalisation in the generator [27].

To condition the generator, we simply concatenate the one-hot encoding of the class with the latent noise vector z. We also experimented with using categorical batch normalisation as the conditioning strategy, but found that this caused mode collapse within each class.

5.3.1 Optimiser

We use the Adam optimiser with $\alpha_G = 0.0005$, $\alpha_D = 0.0001$, $\beta_1 = 0.0$, $\beta_2 = 0.99$ (same β values as in the original ProGAN paper). Unlike the ACGAN configuration, we perform just one generator training iteration for each discriminator training iteration.

5.3.2 Equalised Learning Rate

We use an equalised learning rate, where weights are initialised in the first instance using $\mathcal{N}(0, 1)$, and scaled dynamically during training. Specifically, we set

$$\hat{w}_i = \frac{w_i}{c} \tag{5.8}$$

where c is the per-layer normalisation constant from He initialisation [64],

$$c = \sqrt{\frac{2}{n_l}} \tag{5.9}$$

and n_l is the number of neurons in a layer.

5.3.3 Pixelwise Feature Vector Normalisation

We scale the feature vector of each pixel to unit length after each convolutional layer to prevent excessive signal magnitudes. For the feature vector at pixel (x, y), $a_{x,y}$, we obtain the normalised feature vector $b_{x,y}$ by

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{N}\sum_{j=0}^{N-1} a_{x,y}^{i} + \epsilon}} \tag{5.10}$$

where N is the number of feature maps and $\epsilon = 10^{-8}$.

5.3.4 Minibatch Standard Deviation

In this technique, we first compute the standard deviation over each feature and spatial location in a minibatch. We take the average of these standard deviations to yield a single value, which is then repeated and appended as an additional feature map.

5.3.5 Projection Discriminator

Introduced by Miyato and Koyama [65], in a projection discriminator, a projection between the feature maps and condition is added to the discriminator output. This is computed as

$$f(x, y) = y^{\intercal} V \phi(x) + \psi(\phi(x)) \tag{5.11}$$

where x is the feature vector, y is the class, and V is the class embedding matrix – making $y^{\intercal}V$ the corresponding embedding vector. The transformations ϕ and ψ are implemented as convolutional layers.

5.3.6 Training

For each fade-in phase, we calculate the alpha value to merge images from the new and old resolution. We spend 50% of each phase fading in the previous resolution using a residual block to "merge" the new and old resolutions. The length of each phase in the training regime is shown in Table 5.1.

Resolution	4^2	8^2	16^2	32^2	64^2	128^2	256^2	512^2
Epochs	20	40	60	80	100	120	140	160
Batch size	512	256	128	64	32	16	8	4

Table 5.1: ProGAN training phases.

5.4 Copy-Paste

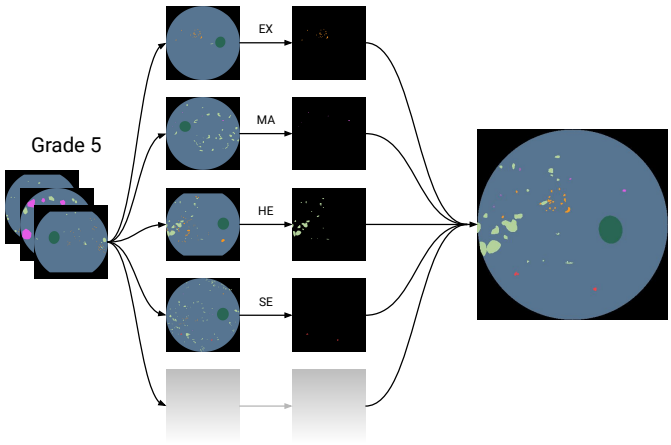


Figure 5.4: Illustration of copy-paste generation.

Ghiasi et al. [66] showed that simply copy-and-pasting objects onto a scene can improve the performance of instance segmentation algorithms on natural images. Inspired by this, we introduce a simple and efficient data augmentation technique that uses multi-label sampling to generate new data for semantic segmentation. We refer to this technique as "copy-paste". To generate copy-pasted data, semantics labels for each class are randomly sampled from existing data and concatenated in order to create a new semantic label of the same grade. This process is depicted in Figure 5.4.

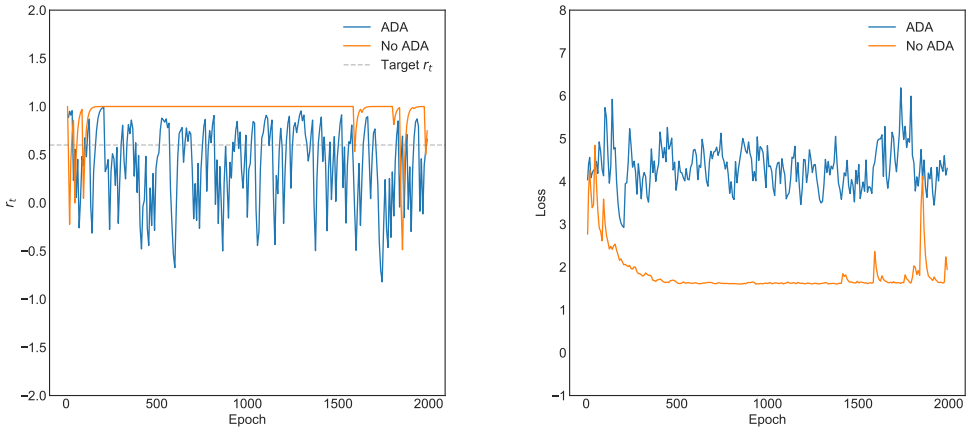
The advantages of this method are clear in that it requires very few compute resources and no prior training, while still being able to create a diverse range of samples. It is only possible because the layout of each image is uniform, and we are not limited by rigid structures such as vessels (which are filled in by the image-to-image translation network). In this sense, retinal fundus images are a very flexible target modality – the same method would be difficult to apply on, for example, the Cityscapes dataset.

Of course, this approach does not take into account any knowledge *a posteriori* about the retinal structure or relationship between lesions. Hence, images generated using this method can exhibit overlapping, or implausible semantic labels. Nevertheless, these same issues can arise with GANs; it may even be the case that certain lesions are simply too difficult or too small for the GANs (in their current form) to generate, whereas copy-paste has no such limitations.

As before, we take care to prevent data leakage by sampling new data points from only the fixed training set. We could also apply scale and position jittering to further increase diversity, but we leave this as a future investigation.

5.5 Experiments

For all ACGAN experiments, we set the batch size to 64 and trained for 2000 epochs. Early experiments included "slow" (e.g. 100 generator, 100 discriminator iterations per batch) and "fast" (e.g. 1 generator, 1 discriminator iteration per batch) training schemes, and exploring different variants on ADA, such as annealing the maximum p value over time, or tweaking the types of transformation. Rather predictably, any change that increased the probability of discriminator overfitting was likely to cause training to collapse. We also test a variant of the ACGAN where we condition the generator by concatenating the embedded class vector with the latent dimension, instead of taking the Hadamard product.



(a) Value of r_t during training.

(b) Discriminator loss during training.

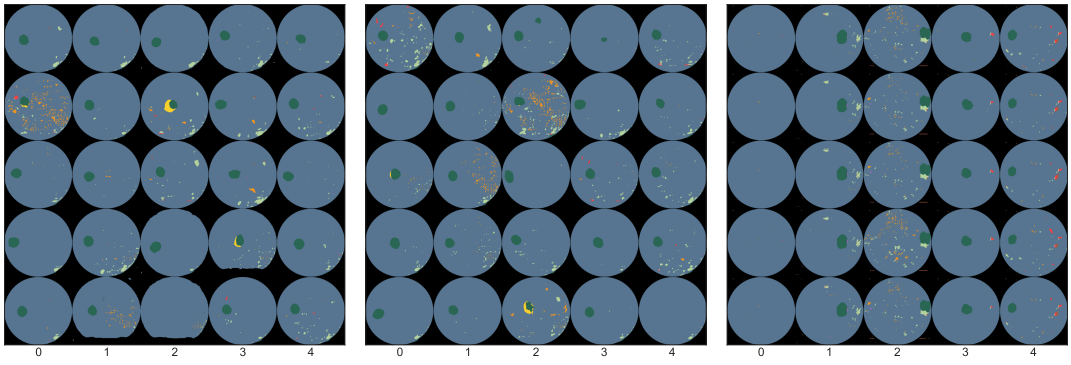
Figure 5.5: Comparison between training with and without ADA.

To investigate the importance of ADA to GAN stability, we trained the ACGAN with and without ADA and compared how training developed over time. From Figure 5.5, we can see that without ADA the adversarial discriminator loss quickly collapses. Note that the total discriminator loss does not go to zero due to the magnitude of the auxiliary losses. The overfitting heuristic r_t immediately rises to its maximum possible value of 1, and remains there for essentially the rest of training. The generator is unable to recover from this. Conversely, when we apply ADA, we are able to use the negative feedback from r_t to increase the probability of augmentations to reduce discriminator overfitting, and thereby keep training relatively stable.

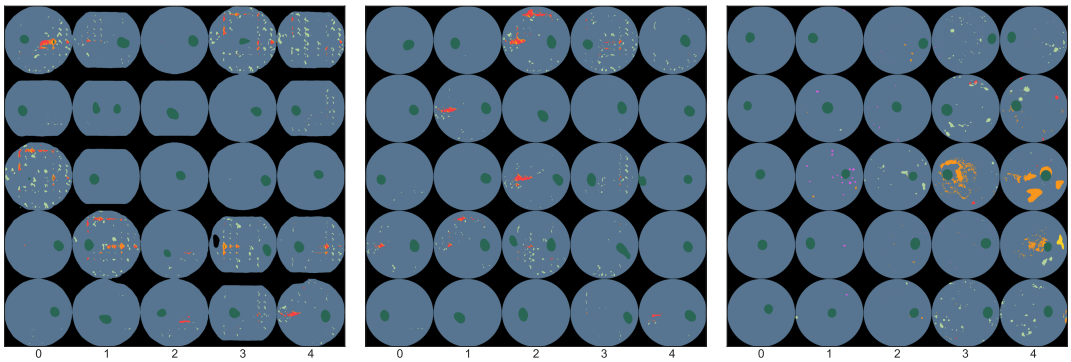
The processing and memory usage required by the ProGAN is significantly higher, which caused batch sizes to be smaller, and overall training times to be much longer. As a result, we used a batch size of 8 for the ProGAN. A side-effect of this was that r_t and p, which are updated every batch, were much noisier during training for the ProGAN than the ACGAN.

5.5.1 Samples

A sample of generated images is shown in Figure 5.6. From a brief visual examination, the GANs appear have successfully been able to capture the form of the semantic labels. The conditioning, however, has been less successful. Only the ACGAN with concatenation has been able to learn a definite conditioning, but it suffers mode collapse within each of the classes. We can also see a number of recurring modes in the ACGAN and ProGAN samples, which is indicative of low diversity. Meanwhile, the copy-pasted data shows a diverse range of samples, and is clearly well-conditioned. Despite this, there are some



(a) ACGAN (b) ACGAN + mask (c) ACGAN + mask, concat.

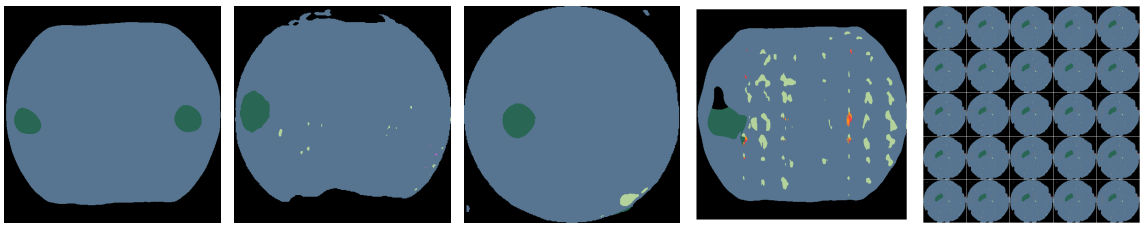


(d) ProGAN (e) ProGAN + mask (f) Copy-Paste

Figure 5.6: Sample of generated semantic labels for each configuration. Columns from left to right correspond to input DR grades 0 through 4.

instances where the optic disc has been pasted directly on top lesions, which is unrealistic. The "mask" variants have an additional post-processing step applied in which the fundus boundary is masked as a circle. This is in response to a failure mode discussed in the next section.

5.5.2 Common Failure Modes



(a) Multiple optic discs. (b) Malformed fundus. (c) Spurious pixels. (d) Holes. (e) Mode collapse.

Figure 5.7: Examples of common failure modes.

The models still display a number of failure modes which are yet to be addressed in the current iteration of this work.

Multiple Optic Discs

There is a tendency to generate multiple optic discs, which is not possible in the source data. Example shown in Figure 5.7a.

Malformed Fundus

Both GAN types encountered difficulty with generating geometric shapes, such as the circular fundus mask. One mitigation is to remove this responsibility from the generator and simply mask out the fundus as a post-processing step. Example shown in Figure 5.7b.

Spurious Pixels

If the GAN is over-trained, there is a point at which spurious pixels outside of the fundus boundary are generated. These kinds of artefacts are obviously unrealistic and not desirable. Example shown in Figure 5.7c.

Holes

Some images exhibit "tears" or gaps in the semantic labels. This is particularly prevalent with the ProGAN. Example shown in Figure 5.7d.

Low Diversity/Mode Collapse

As mentioned, the GANs can exhibit low diversity, even if this does not quite extend to total mode collapse. For instance, one can see from the ACGAN samples that the optic disc is generated in approximately the same spot in almost all samples. Without diversity, downstream models will be prone to overfitting to the synthetic data, decreasing performance. This happens when the discriminator is too strong, which implies that we may need to inhibit the discriminator even further. Example shown in Figure 5.7e.

5.5.3 Results

Configuration	FID↓	Accuracy↑	Precision↑	Recall↑	F_1↑	κ↑
Validation	17.118	0.709	0.700	0.741	0.710	0.739
ACGAN	64.388	0.191	0.249	0.187	0.115	-0.004
+ mask	44.674	0.195	0.194	0.195	0.154	0.001
+ mask, concatenation	80.743	0.037	0.216	0.036	0.027	0.051
ProGAN	51.970	0.206	0.194	0.205	0.142	0.020
+ mask	46.044	0.200	0.189	0.199	0.176	0.018
Copy-Paste	**21.717**	**0.691**	**0.688**	**0.691**	**0.686**	**0.807**

Table 5.2: Metrics assessing the image and conditioning quality for each generation method. Best results for each metric in bold (excluding the baseline).

To evaluate image quality, we compute the FID score between the coloured semantic labels of the training set and those of the target dataset. Even though the FID is designed for natural looking images (after all, the Inception network is trained on Image-Net), and the semantic image colours are chosen arbitrarily, it seems to function as a fairly good proxy for visual quality and diversity regardless. To evaluate the strength of DR grade conditioning, we trained a ResNet-50 network for 100 epochs on the 9-channel semantic labels of the FGADR subset of the real training data (since only these have annotations for DR severity labels). We use this network to perform inference on generated data, and retrieve the accuracy, precision, recall, F_1, and κ achieved. This model was trained on one Nvidia GeForce GTX 1080 for 100 epochs over 2 hours.

Quantitative results of these experiments are reported in Table 5.2. In addition to metrics on the synthetic datasets, we also include metrics on the real validation dataset as a baseline. The "mask" configurations refer to using a circle mask for the retinal fundus

boundary, and the "concatenation" configuration refers to concatenating the class vector instead of taking the Hadamard product.

From these, we find that copy-paste generated samples yield the best metrics across the board, by quite some margin – even yielding a greater κ than the validation set. Moreover, our fears about poor conditioning are confirmed, with each GAN-based method displaying classifying power which is effectively no better than random in their samples. The variants with masked retinal fundus boundaries exhibited a marked improvement in FID, which indicates that this is a simple, but promising, post-processing technique.

Of the GANs, the ACGAN with concatenation had the largest κ (while still being low compared to real data). However, it had a much higher FID, despite the individual generated images appearing similar to the others. This is because the FID score attempts to capture diversity as well as image quality, and these samples exhibit very little diversity owing to mode collapse during training.

In addition to the quality of images, training and inference time are also important factors for practicality. In Table 5.3 we report the mean training time of each model, as well as the average inference time for 1000 images.

Configuration	Time Per Epoch (s) ↓	Inference Time↓ (s)	
ACGAN	36.9	**13.08**	± 0.933
ProGAN	112.5	35.51	± 0.100
Copy-Paste	–	183.29	± 1.304

Table 5.3: Training and inference time for different semantic label generation methods. We report the mean of three runs and the standard error of the mean. Best results for each metric in bold.

These experiments were run on a machine with a Nvidia GeForce GTX 1080 Ti GPU, Intel i7-7700K 4.20GHz CPU, and 32 GB of memory. For inference, we use a batch size of 128 for the ACGAN and 8 for the ProGAN. The "time per epoch" figure for the ProGAN is given as an average across the entire duration of training, despite smaller resolutions taking significantly less time than higher resolutions.

Being able to generate images at twice the resolution, and as a much larger network overall, the ProGAN is over 3 times slower to train than the ACGAN. It also incurs a similar performance penalty during inference. This can be partly attributed to the fact that batch size is far more limited for the ProGAN. Meanwhile, data generation via copy-pasting does not require any form of training. While copy-paste appears performs poorly here in terms of inference time, it is worth bearing in mind that this implementation is far from optimised. For instance, no GPU acceleration is performed, nor is it multi-threaded. This leaves large room for improvement, and could feasibly be faster than the neural networks when properly optimised.

Chapter 6

Retinal Fundus Image Synthesis

To generate realistic retinal fundus image from the semantic labels, we use an image-to-image translation network.

6.1 SPADE

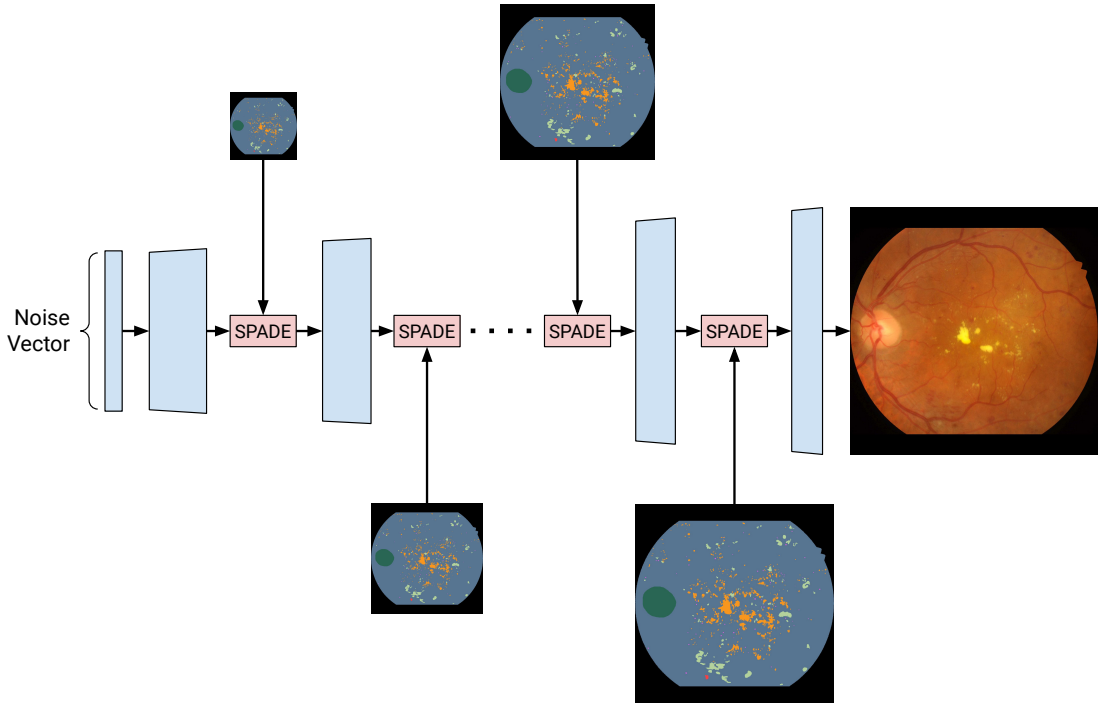


Figure 6.1: Illustration of how the SPADE generator is conditioned.

Spatially-Adaptive (De)normalisation (SPADE) is a conditional normalisation layer, like AdaIN[1], but specially crafted for the semantic image synthesis task [67]. In SPADE, the parameters γ and β are now tensors instead of vectors as in AdaIN or batch normalisation. Once again, consider $X \in \mathbb{R}^{N\times C\times H\times W}$ and also the segmentation mask $M \in \mathbb{L}^{H\times W}$ where $\mathbb{L}$ is a set of integers representing the labels present in the mask. Then the SPADE transform is given by

$$\text{SPADE}(X, M) = \gamma_{chw}(M)\left(\frac{X_{nchw} - \mu_c(X)}{\sigma_c(X)}\right) + \beta_{chw}(M) \tag{6.1}$$

where μ_c, σ_c are defined in the same way as in batch normalisation (Equation (2.6) and (2.7)), and $\gamma_{chw}, \beta_{chw}$ denote functions that convert the segmentation map M to the scalar denormalisation parameters. In practice, these functions are implemented as small convolutional networks.

[1]Interestingly, other conditional normalisation techniques such as AdaIN are actually special cases of SPADE.

Since the denormalisation parameters themselves encode enough information about the label map, we are able to discard the entire encoder part of the network in the SPADE generator. Instead, we pass the resized segmentation map directly to the SPADE layers in the generator network, as depicted in Figure 6.1. Note the similarity with StyleGAN in that they both inject information into conditional normalisation layers, whereas traditional GAN architectures feed information forward from the first layer.

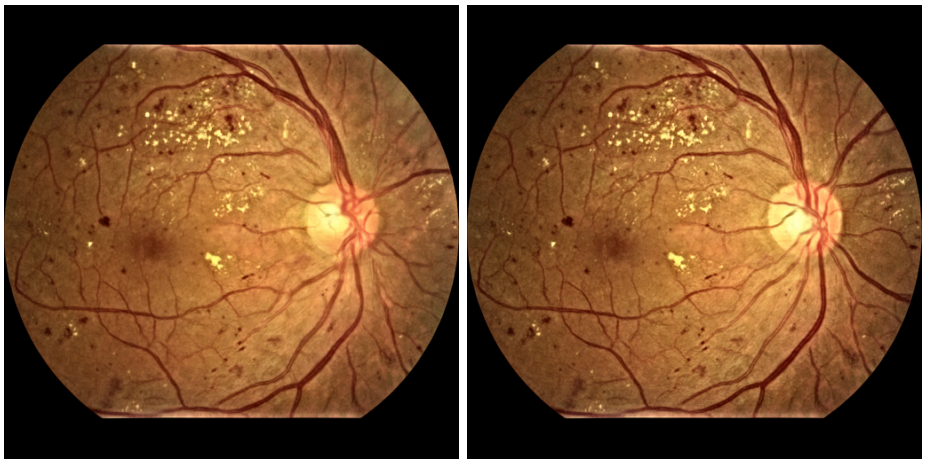
We use the same multi-scale discriminator used in pix2pixHD. The network is trained to translate semantic labels of resolution 512×512 to realistic retinal fundus images of the same size. Weights are initialised using Xavier initialisation, and we use Hinge loss as our loss function.

Additional information is provided to the model in the form of optic disc instance maps. Apart from these and the semantic labels, we do not provide any other form of conditioning, such as grading information.

6.2 Experiments

Experiments with this network were much more limited due to long training times and limited access to hardware.

We use the Adam optimiser with $\alpha_G = \alpha_D = 0.0002$, $\beta_1 = 0.0$, $\beta_2 = 0.9$, and a latent dimension of 256. The network was trained for 500 epochs on two GPUs over 182 hours.



(a) Without instance map. (b) With instance map.

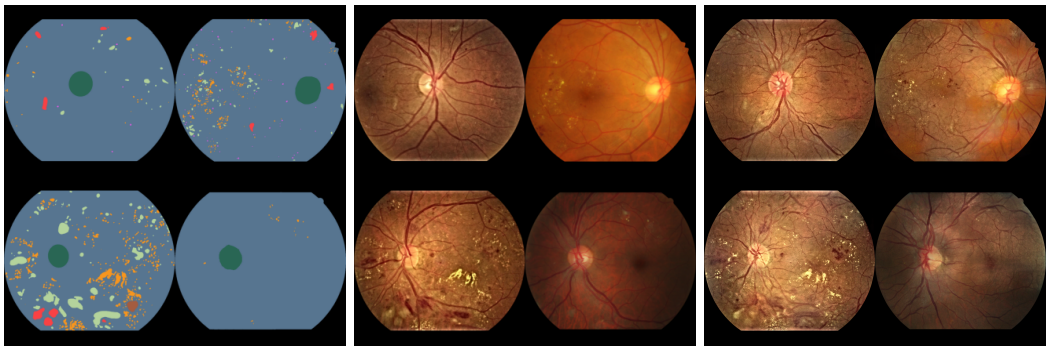
Figure 6.2: Effect of instance maps.

We found that the network was capable of producing realistic retina texture and structures, including blood vessels and the macula (the dark circle opposite the optic disc). This is impressive, considering that there is no guidance for retinal structures in the semantic labels. The network is also able to accurately place the lesions according to the semantic labels.

Layout prediction of the optic disc in particular was important for visually plausible retinas, as we found that without providing explicit conditioning in the form of instance maps the SPADE generator created images which exhibited optic discs with poorly defined boundaries. This can be seen in Figure 6.2, where without an instance map, the optic disc is blurry and the vessels display more inconsistencies.

6.2.1 Samples

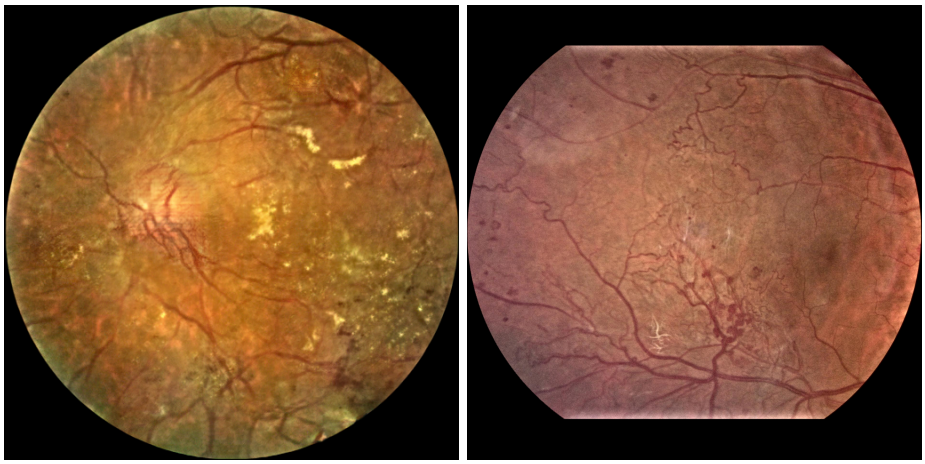
The resulting images show good diversity, with the ability to the generate different lighting conditions that were present in the source data. A sample of images are shown in Figure 6.3.



(a) Real semantic labels. (b) Real images. (c) SPADE output.

Figure 6.3: Comparison of real images and SPADE outputs for semantic labels from the validation set.

6.2.2 Common Failure Modes



(a) Generated image with a misplaced vascular root. (b) Real photograph with obscured optic disc.

Figure 6.4: Examples of common failure modes.

Misplaced Vascular Root

When the optic disc is not present in the semantic label, the network has difficulty placing the vascular "root". Example shown in Figure 6.4a. It is possible that the optic disc is obscured in a real retinal fundus photograph, as in Figure 6.4b, however there are only two examples of this in the training set, and the network is unable to infer that the vascular root should exist elsewhere.

Unrealistic Vascular Structure

Even when the optic disc is not occluded, the network sometimes generates unrealistic retinal vasculature, with non-contiguous regions.

Blurriness

In general, the images appear blurrier at smaller scales than the real photographs,

which are sharp. Again, this is particularly apparently when examining the blood vessels in detail.

6.2.3 Results

Similarly to before, we compute the FID between the target dataset and the real training set, with results shown in Table 6.1. The synthetic images are generated by providing SPADE with the real validation semantic labels as input to produce fake retinal fundus images as output. Again, we also provide the FID score on the real validation retinal fundus photographs as a baseline. We compare the original retinal fundus images, without applying illumination correction.

Configuration	FID↓
Validation	23.491
SPADE	48.438
+ instance maps	**46.603**

Table 6.1: FID scores between real images from the training set and the output of SPADE on the real validation semantic labels. Best results for each metric in bold (excluding the baseline).

This shows that the use of instance maps yield a small improvement in overall image quality. Unfortunately, since we are not able to obtain the code for DR-GAN or Tub-sGAN, and since the authors did not state the dimensionality used to compute the FID scores reported in their papers (which can alter the score by orders of magnitude), it's impossible to draw a quantitative comparison. Instead, we leave this as future work.

Chapter 7

Evaluation

In Chapter 5 and Chapter 6, we examined the quality of generation of each part of the system in isolation. Having having done this, we return to the overarching goal of this project: to observe the effect of *fully-synthetic* data (i.e. generated retinas from generated semantic labels) on downstream tasks. Hence, in the evaluation phase we aim to answer the following questions:

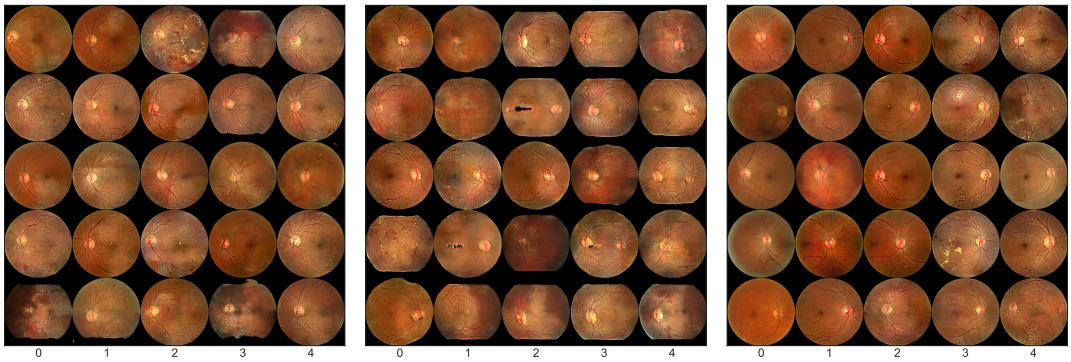
- How closely does synthetic data resemble the real data?
- What effect does synthetic data have on lesion segmentation performance?
- What effect does synthetic data have on grading performance?

7.1 Preliminary Results

An early version of this project was submitted as a short paper[1] to MIDL 2021. While using similar techniques, this work focused on generation of only the optic disc and hard exudate lesions, with the retina being added afterwards using a circle mask, in the same way as discussed above. These preliminary results were promising, and showed that inflating real data with synthetic data was successful in improving the performance of semantic segmentation for lesions.

Despite this, the results of the paper were limited in that restricting the problem domain in this way made the task far easier, and naturally allowed for more stable training. This was not only reflected in the generated image quality, in that the GAN was able to generate more plausible semantic labels, but also in the level of conditioning. Here, we extend those models to generate 9 classes (from 3), and provide a more complete evaluation.

7.2 Visual Quality



(a) ACGAN (b) ProGAN (c) Copy-Paste

Figure 7.1: Fully-synthetic retinal fundus images.

[1] What qualifies as a "short paper" can be found at https://2021.midl.io/call-for-papers.html.

We begin by qualitatively and quantitatively evaluating the overall visual quality of the fully-synthetic images. A sample of images, created using each semantic label generation method, are shown in Figure 7.1. For all experiments in this evaluation, we use the ACGAN and ProGAN generated samples without any fundus masking, unless otherwise specified. The semantic labels are translated into retinal fundus images using the same trained SPADE network for each method with instance maps.

Curiously, images that have few lesions resemble photographs taken from the e-ophtha dataset most closely. This is likely because most photographs of healthy retinas come from the e-ophtha dataset, and this bias is reflected in the output of the SPADE network, highlighting the need for greater diversity in the source data.

To quantify differences in diversity and quality, we begin by generating 3000 samples of fully-synthetic images in equal proportions across the five DR grades using each of the methods. We then compute the FID score between these and the retinal fundus photographs from the training set, with results shown in Table 7.1. We are particularly interested in how these compare in relation to real images; to this end, the FID between the real training set and real validation/test sets are also given.

Surprisingly, the images who had their retina boundaries masked post-generation performed worse than their unmodified counterparts – despite them performing better in the semantic label evaluation. The reason for this is not entirely clear, but we can put forward a couple of theories. One possible reason is that the FID score is simply not designed for these types of images, and thus gives spurious results that would be inconsistent with human perception. In particular, purely geometric shapes such as the circle that forms the retinal fundus boundary are rare in the real world, and could be thought of as "unnatural". Moreover, in the masked images these boundaries are only ever perfect circles, whereas in the source dataset, samples whose top and bottom sections have been cropped are common, which could cause distance between the data distributions.

Configuration		FID↓
Test	23.281	
Validation	23.491	
ACGAN	**43.042**	± 0.196
+ mask	50.642	± 0.229
ProGAN	48.882	± 0.305
+ mask	54.353	± 0.092
Copy-Paste	45.394	± 0.072

Table 7.1: FID scores between real images from the training set and the output of SPADE on the generated semantic labels. We report the mean of three runs and the standard error of the mean. Raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline).

7.3 Classification Performance

The intuition for the first part of our classification performance evaluation is that images that closely resemble their source class should be able to be labelled correctly by a classifier trained on real data. To quantify this, we use the standard precision, recall, and F_1 score metrics.

Next, we investigate two different approaches to using synthetic data *together* with real data to augment model performance. In the first, we train models on a mix of real and generated data. As part of this, we investigate the effects that varying the amount of synthetic data used, in terms of the proportion of the training data which is synthetic, has on the final model performance. Each experiment will use the entire set of real training data, and only the amount of synthetic data will be tuned. From the generated data, we sample the appropriate number of images according to the experiment, and concatenate this with the real training data. Table 7.2 shows how much synthetic data is used in each classification experiment – note how for these experiments, we use the smaller grading set instead of the EyePACS dataset, since EyePACS is extremely large and therefore slow to train on. We also considered an alternative approach where we simply take a small subset of the EyePACS dataset, but we found that the class imbalance and high amount of noise in the parent dataset (which would usually be offset by the large volume of data) caused extremely poor performance on the test set.

# Real	% Synthetic	# Synthetic	# Total
1606	25	535	2141
	50	1606	3212
	75	4818	6424

Table 7.2: Data amounts for classification experiments.

The second approach is related to transfer learning. In this particular version of the technique, we will pre-train a network on only synthetic data (7000 images) for 100 epochs, and then subsequently fine-tune on only real data for a further 100 epochs. We are interested in determining what the effect is on both how fast the model improves on real data, and if the final performance is changed. To contrast and compare the merits of both approaches, we collect performance metrics on the test set for each configuration, and compare these against a model trained on real data only.

Our classification network is a ResNet-50 trained to classify 512×512 illumination-corrected retinal fundus images as one of the five DR grades. Across all experiments, the network is trained for 100 epochs (where training loss plateaued) with learning rate $\alpha = 0.001$, and batch size 16, on a mix of single Nvidia Titan X Pascal and Nvidia GeForce GTX 1080 GPUs. Each configuration is trained from scratch, with the exception of the fine-tuning stage of the transfer learning experiment. We refer to models trained on real data only as "baseline" models.

7.3.1 Results

Configuration	Accuracy↑	Precision↑	Recall↑	F_1↑	κ↑
Validation	0.517	0.455	0.437	0.435	0.569
Test	0.509	0.442	0.441	0.430	0.539
ACGAN	0.202	0.196	0.201	0.164	−0.006
ProGAN	0.199	0.216	0.200	0.159	−0.002
Copy-Paste	**0.313**	**0.366**	**0.313**	**0.247**	**0.315**

Table 7.3: Performance of a classification model trained with only real data on real and synthetic datasets. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline).

The results of the real classifier on synthetic data are shown in Table 7.3. These figures are somewhat surprising, with all types of synthetic data unable to be well-classified by the ResNet. We expected that the copy-paste generated data would perform best out of the synthetic datasets, since its semantic labels were able to be accurately classified (Table 5.2), and this expectation is confirmed by the data. However, even if copy-paste generated data performed the best out of the synthetic datasets, it performs poorly relative to the real validation and test sets. Since the semantic labels of copy-pasted data were able to be classified well, it could be that performance here is being bottlenecked by the image-to-image translation portion of the generation process.

Configuration	% Synthetic	Accuracy↑	Precision↑	Recall↑	F_1↑	κ↑
Real	0	0.509	0.442	0.441	0.430	0.539
ACGAN	25	0.400	0.440	0.366	0.324	0.292
	50	0.428	0.467*	0.414	0.366	0.328
	75	0.386	0.338	0.302	0.253	0.281
ProGAN	25	0.519*	0.491*	0.506*	0.477*	0.631*
	50	0.410	0.470*	0.436	0.360	0.525
	75	0.570*	0.487*	0.409	0.395	0.489
Copy-Paste	25	0.502	0.462*	0.479*	0.448*	0.607*
	50	0.561*	**0.533***	**0.566***	**0.522***	**0.721***
	75	**0.599***	0.522*	0.536*	0.511*	0.703*

Table 7.4: Performance of a classification model trained with mixed data on the test set. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline), and improvements on the baseline are marked with *.

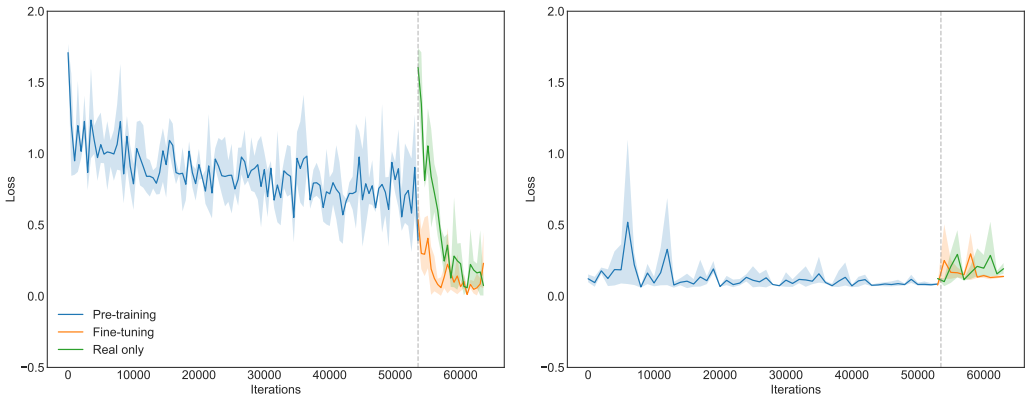
Next, we conduct experiments by training the classifier on a mix of real and synthetic data, with results shown in Table 7.4. This set of results are less surprising, with the ACGAN unanimously causing classification performance to drop. A small amount of synthetic data generated from the ProGAN appears to be beneficial, slightly boosting the classification metrics at 25% synthetic data, above which performance begins to fall. Meanwhile, training on a mix of real and copy-pasted data is able to significantly boost classification performance. What is most noteworthy here is how copy-pasted data can increase performance by a large margin when mixed with real data, yet is not able to be accurately classified by a model trained on only real data. Both the ACGAN and copy-paste configurations observe a peak at 50% synthetic data, with 25% and 75% performing worse. This suggests that increasing the amount of synthetic data will improve model performance to a critical point, after which the model with begin to overfit to unrealistic features in the synthetic data. The models used in these experiments were trained over between one and eleven hours each, depending on the size of the training set.

Configuration		Accuracy↑	Precision↑	Recall↑	F_1↑	κ↑
Real		0.509	0.442	0.441	0.430	0.539
Copy-Paste	Pre-Trained	**0.605***	**0.559***	**0.534***	**0.529***	**0.694***
	Fine-Tuned	0.566*	0.555*	0.525*	0.512*	0.660*

Table 7.5: Performance of a classification model on the test set before and after fine-tuning. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline), and improvements on the baseline are marked with *.

Finally, we look at how transfer learning affects the ability of the model to classify images correctly. We only use copy-pasted data for pre-training since it appears to be most effective for this task, and the results of these experiments are shown in Table 7.5. Here, "pre-trained" refers to the network once it has finished training on only synthetic data, and "fine-tuned" refers to the same network once it has finished further training on real data. The results are unexpected, with the synthetic-only model performing *better* on the test set than the fine-tuned network. A possible explanation for this is that the real dataset is small, and has noisy DR severity labels. Images generated by copy-pasting are the result of sampling semantic labels from multiple different images of the same grade, causing the classes to be more robust, and therefore less noisy. Both the pre-trained and fine-tuned networks perform better than training on only real data, which implies that pre-training increases the ability of the model to extract useful features. The pre-trained model is able to outperform the baseline and fine-tuned models, but its κ falls short of the copy-paste mixed-data training approach, despite having a slightly more favourable F_1 score. This is a consequence of the quadratic weighting used to calculate κ, but we maintain use of κ as the determining factor of "overall" performance.

It is also interesting to analyse the resulting loss curves of transfer learning, and to compare these against the losses of a network trained on only real data, as shown in Figure 7.2. The loss in the pre-training stage decreases slowly, which makes sense since the greater volume of synthetic data (which could be thought of as a type of data augmentation) has a regularising effect on training. Once the network enters the fine-tuning stage, the model is able to leverage its past experience, and begins training at a much lower loss than the one trained from scratch. Despite this "head-start", the baseline model's loss decreases quickly, and by the end of training is approximately the same magnitude as the fine-tuned network. The effect of transfer learning on the validation loss is more subtle, and there is no clear difference between the losses of the baseline and fine-tuned models.



(a) Training loss.

(b) Validation loss.

Figure 7.2: Loss curves for transfer learning. Solid lines are the means of three runs, and the shaded regions show the minimum and maximum values.

7.4 Segmentation Performance

Applying a similar principle to the segmentation task: if our model produces clear and well-conditional retinal fundus images with precise segmentation maps, a model should be able to segment our synthesised images about as well as it can segment real images. We then go on to investigate the same two approaches to training on synthetic data as

# Real	% Synthetic	# Synthetic	# Total
	25	536	2143
1607	50	1607	3214
	75	4821	6428

Table 7.6: Data amounts for segmentation experiments.

described in the previous section. For the mixed-data approach, we use use the proportions specified in Table 7.6.

A standard U-Net is used as our segmentation model architecture. Models were trained to segment all six lesion types (microaneurysms, haemorrhages, hard exudates, soft exudates, intraretinal microvascular abnormalities, and neovascularistion) at a resolution of 512×512. They do not segment other structures such as the optic disc or fundus boundary. All models were trained for 50 epochs with a learning rate of $\alpha = 0.0005$ and batch size of 4 on a mix of single Nvidia Titan X Pascal and Nvidia GeForce GTX 1080 GPUs. We focus on the *relative* performance as opposed to the absolute performance, and therefore do not spend time optimising the model hyperparameters.

7.4.1 Results

Configuration	Precision↑	Recall↑	F_1↑
Test	0.710	0.400	0.439
Validation	0.690	0.374	0.434
ACGAN	0.662	0.200	0.278
ProGAN	0.754	0.183	0.523
Copy-Paste	**0.808**	**0.462**	**0.527**

Table 7.7: Performance of a segmentation model trained with only real data on real and synthetic datasets. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline).

Table 7.7 shows the results of running the baseline segmentation model trained with real data on the synthetic datasets. We find that the model performs best across all metrics on the copy-pasted data, followed closely by the ProGAN. The overall performance (as determined by the F_1 score) for both of these models is greater than that of the test and validation sets. This is indicative of plausible generated semantic labels and well-conditioned image-to-image translation. Performance on the ACGAN generated data, however, is significantly poorer than for any other dataset.

Turning our attention to experiments conducted with mixed training data, Table 7.8 shows the performance of the segmentation model trained under varying amounts of synthetic data. The results are extremely promising, showing that segmentation performance can be boosted by training on synthetic data. We identify an interesting trend where increasing the amount of synthetic data simultaneously decreases precision while increasing recall. The overall effect of this is that the F_1 score improves, at least up to 75% synthetic data. We can think of precision as "how many selected items are relevant" and recall as "how many relevant items are selected"; hence, decreased precision and increased recall can be interpreted as the resulting model being more "liberal" with its selection. It's likely that,

Configuration	% Synthetic	Precision↑	Recall↑	F_1↑
Real	0	0.710	0.400	0.439
ACGAN	25	0.688	0.368	0.421
	50	0.608	0.353	0.406
	75	**0.723***	0.412*	0.473*
ProGAN	25	0.687	0.400	0.449*
	50	0.662	**0.482***	0.464*
	75	0.685	0.406*	0.461*
Copy-Paste	25	0.716*	0.297	0.364
	50	0.702	0.392	0.452*
	75	0.689	0.423*	**0.476***

Table 7.8: Performance of a segmentation model trained with mixed data on the test set. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline), and improvements on the baseline are marked with *.

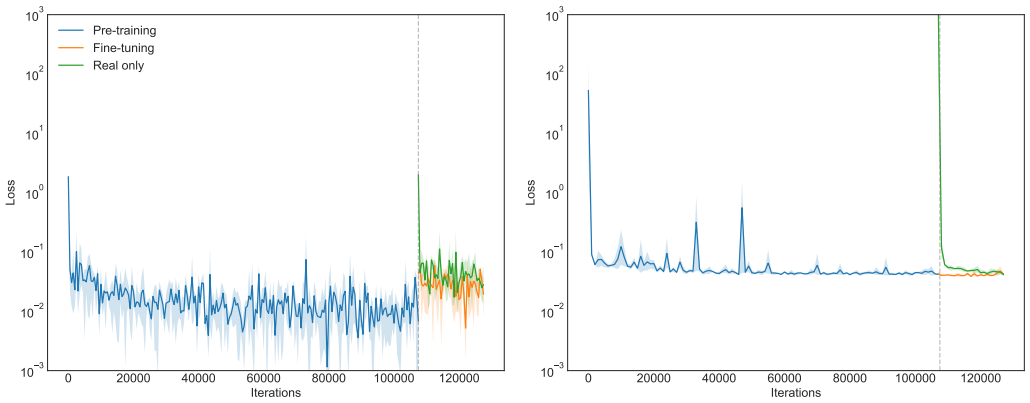
as with the classification experiments, this relationship reaches a point where the F_1 score begins to decrease, but further experiments are required to verify this.

Configuration		Precision↑	Recall↑	F_1↑
Real		0.710	0.400	0.439
Copy-Paste	Pre-Trained	0.646	**0.453***	**0.510***
	Fine-Tuned	**0.729***	0.420*	0.501*

Table 7.9: Performance of a segmentation model on the test set before and after fine-tuning. Mean of three runs, raw data can be found in Appendix A.4. Best results for each metric in bold (excluding the baseline), and improvements on the baseline are marked with *.

This trend continues with the transfer learning approach, where the pre-trained network exhibits a higher recall but lower precision than the baseline model, with a greater overall F_1 score. Conversely, continuing training on the real data has the inverse effect of increasing precision and decreasing recall. Again, we see parallels with the transfer learning approach applied to classification in that the F_1 score is better on the pre-trained model than the fine-tuned model, as if seeing real data causes performance to drop. Both the pre-trained and fine-tuned models are able to achieve a greater F_1 score than any of the baseline or mixed-data training approaches, suggesting that this is the optimal strategy to maximise segmentation performance.

Looking at the loss curves in Figure 7.3, we see a similar pattern to the classification transfer learning loss, although training loss actually increases when entering the fine-tuning stage, suggesting the model has overfit on the synthetic data.



(a) Training loss.

(b) Validation loss.

Figure 7.3: Loss curves for transfer learning, plotted on a logarithmic scale. Solid lines are the means of three runs, and the shaded regions show the minimum and maximum values.

Chapter 8

Conclusion

In this project, we have shown that semantic generation of retinal fundus images is viable as a method of data augmentation, and shows great potential in improving the performance downstream tasks. First, we provided a survey of available datasets and their properties. To expand our training data, we showed how weak supervision can be used to inflate the training set in low-data regimes. We then introduced a novel two-step process for the synthesis of realistic retinal fundus images with accompanying semantic labels, conditioned on DR severity. Finally, we showed that using synthetic data as a basis for mixed-data training or transfer learning can increase the performance of segmentation and classification models on real data.

8.1 Limitations

While it remains true that there is no better substitute for high-quality annotated data, we have shown that synthetic data and its application to medical imaging is a promising avenue of research. However, it should be noted that this project presents only an early proof-of-concept, and as such exhibits a number of limitations.

Despite the fact that the images used in these experiments were of relatively low resolution when compared to raw images captured by cameras, generation of high-resolution, natural, medical images has already been shown to be feasible [68]. However, it remains unclear how the same can be reliably done for the generation of sparse semantic labels. This work suggests that DCGAN-based architectures are ill-suited for this task, and that progressive growing is the key to yielding improved results.

While we were successful in enhancing model performance, there remains much room for improvement in terms of the quality of generated images. Having seen impressive results on natural looking images, it would be exciting to bring semantic label generation GANs up to parity. A synthetic data generator can be considered to rely on two major factors, either of which can become a bottleneck: the quality of input data; and the capacity of the model to learn the representation of the data distribution. There are multiple facets that determine the "quality" of training data, the most obvious being simply the volume. In our case, having less than 2000 training images, and further subdividing this set due to conditioning, makes it difficult to learn a general representation without overfitting. To test this, we could train the GANs on a large volume of copy-paste generated data, with the goal not necessarily being to generate anatomically correct retinal fundus images, but rather to see if increasing the amount of available data allows a GAN to better represent the images. Otherwise, if this is not the case then we can identify that the representational power of the generator or discriminator architecture simply is not great enough.

Moreover, the quality of annotations available to us presented an issue. Many datasets are missing the annotations we needed for these tasks, and other datasets simply have annotations which are too coarse. Not only is the accuracy and precision in the labels of the training data important, but also the diversity of data. Changes in imaging conditions, such as field-of-view, focal length, and lighting, can cause differences between the training and production data. Since generative models aim to create data points which are plausibly

sampled from the input distribution, this covariate shift may cause the models trained on synthetic data to not be robust to all field conditions, and therefore produce spurious results.

Finally, there are a number of caveats inherent to the use of synthetic data. The generative models presented in this work provide no guarantee on the validity or anatomically correctness of generated data. Hence, the model may be susceptible to producing anomalous results, and a downstream model which incorrectly learns these features will reflect this. One must also be aware that over-training on synthetic data can cause a model to overfit to unrealistic features in the synthetic data.

8.2 Future Work

In this work, we generated synthetic data that had "annotations" for both the segmentation and classification task. We then evaluated the two tasks separately. Yet, methods for joint classification and segmentation of the retinal fundus modality exist in the literature [69], so supplying these models with the synthetic samples could be an interesting study, and may improve performance by a greater margin than on each individual task separately.

When conditioning our models, did not incorporate retinal structures such as the vasculature or location of the macula. Instead, these details were generated by the SPADE network. However, these structures potentially contain important anatomical information; for example, the proximity of lesions to the macula may inform the severity of disease. Incorporating these conditions could further guide and inform the generation of semantic labels.

While thorough, the evaluation for this project was limited to ad-hoc metrics for measuring GAN performance, which may not be entirely appropriate, and so there are a number of improvements that could be made here. We have discussed how the FID is not ideal for quantifying the quality of retinal fundus images or semantic labels. It's possible that new metrics must be developed for these modalities – this could possibly be as simple as training the Inception network used to calculate the FID on retinal fundus images. Also, we were unable to secure any human experts for this project, meaning we cannot draw any concrete conclusions about the anatomical correctness of generated images. Further applications not explored here include use in training human experts to better identify retinal lesions, for which we would need highly plausible images.

A large part of this work emphasises the difficulty of training GANs in a stable manner. While it is generally accepted that GANs lead the state-of-the-art in terms of high-fidelity image generation, other advances in generative models such as VAEs and autoregressive models show potential, bringing into question whether GANs are necessarily the most suitable framework for this use-case. One considerable advantage that these techniques have is that training is much more stable, and going forward it is well-worth exploring whether equal or greater quality semantic labels can be synthesised by these models.

8.3 Ethical Issues

In this section, we discuss the potential implications of deploying an automated decision-making system in the context of medicine.

8.3.1 Machine Learning in Healthcare

Machine learning techniques and their application to healthcare have see massive popularity in the past decade, largely thanks to their versatility and ability to achieve groundbreaking results. The use of artificial intelligence in medicine goes back as far as the 1960s [70], but what sets machine learning apart is its ability to learn directly from large amounts of data, instead of relying on hard-coded rules.

As a result, the usefulness of a machine learning model is bounded by the quality of its training data. Hence, it is of massive importance that the source data is as free of bias as possible, as prejudice in the training data will translate to prejudice in the decisions that the model makes. This is not just a hypothetical, but a very real problem that has been demonstrated time and time again. For instance, in 2019 Obermeyer et al. [71] found evidence of gross racial bias in a commercial AI system used in the U.S healthcare system. A similar risk is present in our domain since skin tone affects fundus pigmentation, and the prevalence of ocular diseases has been shown to vary between ethnicities [72]. By deliberately sampling an unbalanced subset from the EyePACS dataset, Burlina et al. [73] was able to show a significant disparity based on skin-tone in the performance of a DR classification model, when trained on unbalanced data. In our case, data bias is likely since the little data we have is collated from local hospitals, decreasing the chance of a diverse and balanced dataset, and we cannot afford to selectively undersample our data. The consequence of this is that a synthetic data generator trained on biased data will naturally generate more data reflecting that bias, exacerbating the issue. Burlina et al. proposed an effective method of debiasing the source data by applying StyleGAN to generate fundus images which are then manipulated in the latent space to exhibit different skin tones. A similar idea could be interesting to explore as an extension to this project.

8.3.2 Data Privacy

Diabetic retinopathy is not a rare disease, and the precise locations from which the patient data was collected is not given in any of the used datasets. Applying the "motivated intruder" test [74], it is clear that individuals will not be directly or indirectly re-identifiable from the retina images. Since there is no identifying information accompanying the retinal fundus photographs in any of our datasets, and an individual cannot (practically) be identified from a retinal fundus image, we consider our data to be "truly anonymised", as defined under Retical 26 of the GDPR [75].

8.3.3 Copyright and Licensing

To ensure that the contributions of this project are as as freely available as possible, any software will be distributed under a permissive FOSS license. Proper attribution is desirable, making the (3-clause) BSD license the natural choice. Since BSD only covers source code, we license other work (images, documentation, etc.) under CC BY 4.0 which is similar to BSD in spirit. An additional complication is that copyright and licensing only covers works created by humans. However, since a degree of human creative input did go into creating the synthetic data generator, it's reasonable to put these under the same license. In terms of using third-party software, we must take care to retain the copyright notices of PyTorch[1] and any other software that uses a similar license. This project will not be applied commercially, so we are free to use works licensed under non-commercial licenses such as SPADE[2].

[1] https://github.com/pytorch/pytorch/blob/master/LICENSE
[2] https://github.com/NVlabs/SPADE/blob/master/LICENSE.md

Appendix A

Supplementary Materials

Codebase https://github.com/semantic-retina/semantic-retina-generation

Materials https://github.com/semantic-retina/supplementary-materials

A.1 FGADR Exclusion List

This is a list of files that we determined too coarse to be used as segmentation maps, and were therefore excluded from our training data. It can be found in CSV format in the supplementary materials repository under `data/fgadr_denylist.csv`.

A.2 Optic Disc Annotations

For this project, we manually annotated the optic discs for the FGADR and DiaRetDB1 datasets. This was done using the open-source Make Sense annotation tool[1], and are stored as JSON files under the `data` directory. Utilities for pre-processing datasets using these files are included in the main code repository. Inferred annotations for e-ophtha were collected as detailed in Section 4.9, and are stored as PNG files containing binary masks under the `data/eophtha_od` directory.

A.3 Models

For reproducibility, the supplementary materials repository contains all model weights used in this project, including the ACGAN, ProGAN, ResNet, and U-Net. These are contained under the `models` directory.

A.4 Raw Data

Raw data collected from experiments can be found as spreadsheet files under the `raw data` directory.

A.5 Mask Overlay Tool

Mask Overlay Tool https://github.com/semantic-retina/mask-overlay

The mask overlay tool is a small command-line program built to create labelled retinal fundus image diagrams, such as Figure 2.1.

[1] https://www.makesense.ai

References

[1] Mathur R, Bhaskaran K, Edwards E, Lee H, Chaturvedi N, Smeeth L, et al. "Population trends in the 10-year incidence and prevalence of diabetic retinopathy in the UK: a cohort study in the Clinical Practice Research Datalink 2004-2014". In: *BMJ Open* 7.2 (2017). ISSN: 2044-6055. DOI: 10.1136/bmjopen-2016-014444. URL: https://bmjopen.bmj.com/content/7/2/e014444.

[2] Liew G, Michaelides M, and Bunce C. "A comparison of the causes of blindness certifications in England and Wales in working age adults (16–64 years), 1999–2000 with 2009–2010". In: *BMJ Open* 4.2 (2014). ISSN: 2044-6055. DOI: 10.1136/bmjopen-2013-004015. eprint: https://bmjopen.bmj.com/content/4/2/e004015.full.pdf. URL: https://bmjopen.bmj.com/content/4/2/e004015.

[3] Krause J, Gulshan V, Rahimy E, Karth P, Widner K, Corrado GS, et al. "Grader variability and the importance of reference standards for evaluating machine learning models for diabetic retinopathy". In: *CoRR* abs/1710.01711 (2017). arXiv: 1710.01711. URL: http://arxiv.org/abs/1710.01711.

[4] Baudoin CE, Lay BJ, and Klein JC. "Automatic detection of microaneurysms in diabetic fluorescein angiography". In: *Revue d'epidemiologie et de sante publique* 32.3-4 (1984), pp. 254–261. ISSN: 0398-7620. URL: http://europepmc.org/abstract/MED/6522738.

[5] Lundervold AS and Lundervold A. "An overview of deep learning in medical imaging focusing on MRI". In: *Zeitschrift für Medizinische Physik* 29.2 (2019), pp. 102–127. ISSN: 0939-3889. DOI: https://doi.org/10.1016/j.zemedi.2018.11.002. URL: http://www.sciencedirect.com/science/article/pii/S0939388918301181.

[6] Taylor R and Batey D. *Handbook of retinal screening in diabetes: diagnosis and management*. John Wiley & Sons, 2012.

[7] Salz DA and Witkin AJ. "Imaging in diabetic retinopathy". In: *Middle East Afr J Ophthalmol* 22.2 (2015), pp. 145–150.

[8] Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, et al. "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs". In: *JAMA* (2016). URL: http://jamanetwork.com/journals/jama/fullarticle/2588763.

[9] Xiao Q, Zou J, Yang M, Gaudio A, Kitani K, Smailagic A, et al. "Improving Lesion Segmentation for Diabetic Retinopathy Using Adversarial Learning". In: *Image Analysis and Recognition* (2019), pp. 333–344. ISSN: 1611-3349. DOI: 10.1007/978-3-030-27272-2_29. URL: http://dx.doi.org/10.1007/978-3-030-27272-2_29.

[10] Wang Z, Yin Y, Shi J, Fang W, Li H, and Wang X. "Zoom-in-Net: Deep Mining Lesions for Diabetic Retinopathy Detection". In: *CoRR* abs/1706.04372 (2017). arXiv: 1706.04372. URL: http://arxiv.org/abs/1706.04372.

[11] Ophthalmoscopy D and Levels E. "International clinical diabetic retinopathy disease severity scale detailed table". In: *International Council of Ophthalmology* (2002).

[12] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Ghahramani Z, Welling M, Cortes C, Lawrence N, and Weinberger KQ. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[13] Salimans T, Goodfellow IJ, Zaremba W, Cheung V, Radford A, and Chen X. "Improved Techniques for Training GANs". In: *CoRR* abs/1606.03498 (2016). arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498.

[14] Gatys LA, Ecker AS, and Bethge M. "Image Style Transfer Using Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[15] Johnson J, Alahi A, and Li FF. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155.

[16] Chen Q and Koltun V. "Photographic Image Synthesis with Cascaded Refinement Networks". In: *CoRR* abs/1707.09405 (2017). arXiv: 1707.09405. URL: http://arxiv.org/abs/1707.09405.

[17] Dosovitskiy A and Brox T. "Generating Images with Perceptual Similarity Metrics based on Deep Networks". In: *CoRR* abs/1602.02644 (2016). arXiv: 1602.02644. URL: http://arxiv.org/abs/1602.02644.

[18] Ioffe S and Szegedy C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[19] Lipton ZC and Steinhardt J. *Troubling Trends in Machine Learning Scholarship*. 2018. arXiv: 1807.03341 [stat.ML].

[20] Santurkar S, Tsipras D, Ilyas A, and Madry A. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML].

[21] Burt P and Adelson E. "The Laplacian Pyramid as a Compact Image Code". In: *IEEE Transactions on Communications* 31.4 (1983), pp. 532–540. DOI: 10.1109/TCOM.1983.1095851.

[22] Denton EL, Chintala S, Szlam A, and Fergus R. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". In: *CoRR* abs/1506.05751 (2015). arXiv: 1506.05751. URL: http://arxiv.org/abs/1506.05751.

[23] Huang X, Li Y, Poursaeed O, Hopcroft JE, and Belongie SJ. "Stacked Generative Adversarial Networks". In: *CoRR* abs/1612.04357 (2016). arXiv: 1612.04357. URL: http://arxiv.org/abs/1612.04357.

[24] Wang TC, Liu MY, Zhu JY, Tao A, Kautz J, and Catanzaro B. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *CoRR* abs/1711.11585 (2017). arXiv: 1711.11585. URL: http://arxiv.org/abs/1711.11585.

[25] Ghosh A, Kulharia V, Namboodiri VP, Torr PHS, and Dokania PK. "Multi-Agent Diverse Generative Adversarial Networks". In: *CoRR* abs/1704.02906 (2017). arXiv: 1704.02906. URL: http://arxiv.org/abs/1704.02906.

[26] Radford A, Metz L, and Chintala S. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: (Nov. 2015).

[27] Karras T, Aila T, Laine S, and Lehtinen J. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *CoRR* abs/1710.10196 (2017). arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196.

[28] Ulyanov D, Vedaldi A, and Lempitsky VS. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *CoRR* abs/1607.08022 (2016). arXiv: 1607.08022. URL: http://arxiv.org/abs/1607.08022.

[29] Huang X and Belongie SJ. "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization". In: *CoRR* abs/1703.06868 (2017). arXiv: 1703.06868. URL: http://arxiv.org/abs/1703.06868.

[30] Karras T, Laine S, and Aila T. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[31] Mirza M and Osindero S. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[32] Isola P, Zhu JY, Zhou T, and Efros AA. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CoRR* abs/1611.07004 (2016). arXiv: 1611.07004. URL: http://arxiv.org/abs/1611.07004.

[33] Pathak D, Krähenbühl P, Donahue J, Darrell T, and Efros AA. "Context Encoders: Feature Learning by Inpainting". In: *CoRR* abs/1604.07379 (2016). arXiv: 1604.07379. URL: http://arxiv.org/abs/1604.07379.

[34] Ronneberger O, Fischer P, and Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[35] Wang TC, Liu MY, Zhu JY, Tao A, Kautz J, and Catanzaro B. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *CoRR* abs/1711.11585 (2017). arXiv: 1711.11585. URL: http://arxiv.org/abs/1711.11585.

[36] Heusel M, Ramsauer H, Unterthiner T, Nessler B, Klambauer G, and Hochreiter S. "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium". In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500.

[37] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, and Wojna Z. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[38] Kynkäänniemi T, Karras T, Laine S, Lehtinen J, and Aila T. *Improved Precision and Recall Metric for Assessing Generative Models*. 2019. arXiv: 1904.06991 [stat.ML].

[39] Bonaldi L, Menti E, Ballerini L, Ruggeri A, and Trucco E. "Automatic Generation of Synthetic Retinal Fundus Images: Vascular Network". In: *Procedia Computer Science* 90 (2016), pp. 54–60. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.07.010. URL: http://www.sciencedirect.com/science/article/pii/S1877050916311887.

[40] Fiorini S, Biasi MD, Ballerini L, Trucco E, and Ruggeri A. "Automatic Generation of Synthetic Retinal Fundus Images". In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Giachetti A. The Eurographics Association, 2014. ISBN: 978-3-905674-72-9. DOI: 10.2312/stag.20141238.

[41] Costa P, Galdran A, Meyer MI, Abràmoff MD, Niemeijer M, Mendonça AM, et al. "Towards Adversarial Retinal Image Synthesis". In: *CoRR* abs/1701.08974 (2017). arXiv: 1701.08974. URL: http://arxiv.org/abs/1701.08974.

[42] Costa P, Galdran A, Meyer MI, Niemeijer M, Abràmoff M, Mendonça AM, et al. "End-to-End Adversarial Retinal Image Synthesis". In: *IEEE Transactions on Medical Imaging* 37.3 (2018), pp. 781–791. DOI: 10.1109/TMI.2017.2759102.

[43] Zhao H, Li H, Maurer-Stroh S, and Cheng L. "Synthesizing retinal and neuronal images with generative adversarial nets". In: *Medical Image Analysis* 49 (2018), pp. 14–26. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2018.07.001. URL: http://www.sciencedirect.com/science/article/pii/S1361841518304596.

[44] Niu Y, Gu L, Lu F, Lv F, Wang Z, Sato I, et al. "Pathological Evidence Exploration in Deep Retinal Image Diagnosis". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 1093–1101. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33011093. URL: http://dx.doi.org/10.1609/aaai.v33i01.33011093.

[45] Zhou Y, Wang B, He X, Cui S, and Shao L. "DR-GAN: Conditional Generative Adversarial Network for Fine-Grained Lesion Synthesis on Diabetic Retinopathy Images". In: *IEEE Journal of Biomedical and Health Informatics* (2020), pp. 1–1. ISSN: 2168-2208. DOI: 10.1109/jbhi.2020.3045475. URL: http://dx.doi.org/10.1109/JBHI.2020.3045475.

[46] Sinha A and Dolz J. "Multi-scale guided attention for medical image segmentation". In: *CoRR* abs/1906.02849 (2019). arXiv: 1906.02849. URL: http://arxiv.org/abs/1906.02849.

[47] Fu J, Liu J, Tian H, Fang Z, and Lu H. "Dual Attention Network for Scene Segmentation". In: *CoRR* abs/1809.02983 (2018). arXiv: 1809.02983. URL: http://arxiv.org/abs/1809.02983.

[48] Lin TY, Goyal P, Girshick RB, He K, and Dollár P. "Focal Loss for Dense Object Detection". In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[49] Wang X and Gupta A. "Generative Image Modeling using Style and Structure Adversarial Networks". In: *CoRR* abs/1603.05631 (2016). arXiv: 1603.05631. URL: http://arxiv.org/abs/1603.05631.

[50] Ghelfi E, Galeone P, Simoni MD, and Mattia FD. "Adversarial Pixel-Level Generation of Semantic Images". In: *CoRR* abs/1906.12195 (2019). arXiv: 1906.12195. URL: http://arxiv.org/abs/1906.12195.

[51] Volokitin A, Konukoglu E, and Van Gool L. "Decomposing Image Generation Into Layout Prediction and Conditional Synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[52] Azadi S, Tschannen M, Tzeng E, Gelly S, Darrell T, and Lucic M. "Semantic Bottleneck Scene Generation". In: *CoRR* abs/1911.11357 (2019). arXiv: 1911.11357. URL: http://arxiv.org/abs/1911.11357.

[53] Gaidon A, Lopez A, and Perronnin F. "The Reasonable Effectiveness of Synthetic Visual Data". In: *International Journal of Computer Vision* 126.9 (Sept. 2018), pp. 899–901. ISSN: 1573-1405. DOI: 10.1007/s11263-018-1108-0. URL: https://doi.org/10.1007/s11263-018-1108-0.

[54] Gaidon A, Wang Q, Cabon Y, and Vig E. "Virtual Worlds as Proxy for Multi-Object Tracking Analysis". In: *CoRR* abs/1605.06457 (2016). arXiv: 1605.06457. URL: http://arxiv.org/abs/1605.06457.

[55] Shrivastava A, Pfister T, Tuzel O, Susskind J, Wang W, and Webb R. "Learning from Simulated and Unsupervised Images through Adversarial Training". In: *CoRR* abs/1612.07828 (2016). arXiv: 1612.07828. URL: http://arxiv.org/abs/1612.07828.

[56] Zhou Y, Wang B, Huang L, Cui S, and Shao L. "A Benchmark for Studying Diabetic Retinopathy: Segmentation, Grading, and Transferability". In: *IEEE Transactions on Medical Imaging* (2020), pp. 1–1. DOI: 10.1109/TMI.2020.3037771.

[57] Porwal P, Pachade S, Kamble R, Kokare M, Deshmukh G, Sahasrabuddhe V, et al. "Indian Diabetic Retinopathy Image Dataset (IDRiD): A Database for Diabetic Retinopathy Screening Research". In: *Data* 3.3 (July 2018), p. 25. ISSN: 2306-5729. DOI: 10.3390/data3030025. URL: http://www.mdpi.com/2306-5729/3/3/25.

[58] Decencière E, Cazuguel G, Zhang X, Thibault G, Klein JC, Meyer F, et al. "TeleOphta: Machine learning and image processing methods for teleophthalmology". In: *IRBM* 34.2 (2013), pp. 196–203. ISSN: 1959-0318. DOI: https://doi.org/10.1016/j.irbm.2013.01.010. URL: http://www.sciencedirect.com/science/article/pii/S1959031813000237.

[59] *Diabetic Retinopathy Detection*. https://www.kaggle.com/c/diabetic-retinopathy-detection/data. Accessed: 2021-05-12.

[60] Graham B. *Kaggle Diabetic Retinopathy Detection Competition Report*. https://www.kaggle.com/c/diabetic-retinopathy-detection/discussion/15801. Accessed: 2021-05-25.

[61] Odena A, Olah C, and Shlens J. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2017. arXiv: 1610.09585 [stat.ML].

[62] Lim JH and Ye JC. *Geometric GAN*. 2017. arXiv: 1705.02894 [stat.ML].

[63] Karras T, Aittala M, Hellsten J, Laine S, Lehtinen J, and Aila T. *Training Generative Adversarial Networks with Limited Data*. 2020. arXiv: 2006.06676 [cs.CV].

[64] He K, Zhang X, Ren S, and Sun J. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[65] Miyato T and Koyama M. *cGANs with Projection Discriminator*. 2018. arXiv: 1802.05637 [cs.LG].

[66] Ghiasi G, Cui Y, Srinivas A, Qian R, Lin TY, Cubuk ED, et al. *Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation*. 2020. arXiv: 2012.07177 [cs.CV].

[67] Park T, Liu MY, Wang TC, and Zhu JY. "Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *CoRR* abs/1903.07291 (2019). arXiv: 1903.07291. URL: http://arxiv.org/abs/1903.07291.

[68] Korkinof D, Heindl A, Rijken T, Harvey H, and Glocker B. "MammoGAN: High-Resolution Synthesis of Realistic Mammograms". In: *International Conference on Medical Imaging with Deep Learning – Extended Abstract Track*. London, United Kingdom, July 2019. URL: https://openreview.net/forum?id=SJeichaN5E.

[69] Girard F, Kavalec C, and Cheriet F. "Joint segmentation and classification of retinal arteries/veins from fundus images". In: *CoRR* abs/1903.01330 (2019). arXiv: 1903.01330. URL: http://arxiv.org/abs/1903.01330.

[70] Lindsay RK, Buchanan BG, Feigenbaum EA, and Lederberg J. "DENDRAL: A case study of the first expert system for scientific hypothesis formation". In: *Artificial Intelligence* 61.2 (1993), pp. 209–261. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(93)90068-M. URL: http://www.sciencedirect.com/science/article/pii/000437029390068M.

[71] Obermeyer Z, Powers B, Vogeli C, and Mullainathan S. "Dissecting racial bias in an algorithm used to manage the health of populations". In: *Science* 366.6464 (2019), pp. 447–453. ISSN: 0036-8075. DOI: 10.1126/science.aax2342. URL: https://science.sciencemag.org/content/366/6464/447.

[72] Bourne RR. "Ethnicity and ocular imaging". In: *Eye (Lond)* 25.3 (May 2011), pp. 297–300.

[73] Burlina P, Joshi N, Paul W, Pacheco KD, and Bressler NM. "Addressing Artificial Intelligence Bias in Retinal Disease Diagnostics". In: *arXiv preprint arXiv:2004.13515* (2020).

[74] *Anonymisation: managing data protection risk code of practice*. Information Comissioner's Office. URL: https://ico.org.uk/media/1061/anonymisation-code.pdf.

[75] *EU General Data Protection Regulation*. European Commission. URL: https://www.privacy-regulation.eu/en/recital-26-GDPR.htm.