

A Comparative Analysis of Two Image Synthesis Networks Repurposed for Video Synthesis

**School of Computer Science & Applied Mathematics
University of the Witwatersrand**

**Kyle Weiher
1116087**

Supervised by Dr Richard Klein

September 19, 2021



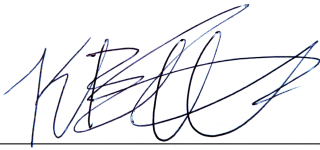
A dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science

Abstract

Modern, state-of-the-art video synthesis networks are large, complex, and difficult to train due to steep hardware requirements. Image synthesis networks are, however, far easier to train, and thus present an opportunity. We present a comparative analysis of two image synthesis networks when repurposed for video synthesis, namely the Cascaded Refinement Network (CRN), and pix2pixHD. We are primarily interested in the temporal consistency between output frames, however we analyse aspects such as image quality as well. The networks are not equal in neither features nor performance, with pix2pixHD superior in both regards, and thus we compare both base network configurations, and configurations where we port components from pix2pixHD to the CRN. Furthermore, we test video specific features, namely the use of prior frame information in the synthesis and training processes, and the use of optical flow estimation to warp prior frame content directly to the next frame. We find that while the CRN performed far worse than pix2pixHD in their base configurations, the CRN with pix2pixHD features closed the gap. Next, the addition of prior frame information provides the largest improvement to temporal consistency, leading to acceptably smooth synthetic video. Finally, optical flow estimation further improves the performance of pix2pixHD, while hindering the CRN.

Declaration

I, Kyle Weiher, hereby declare the contents of this thesis to be my own work. This dissertation is submitted for the degree of Master of Science in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

A handwritten signature in blue ink, appearing to read 'K. Weiher', is written over a horizontal line.

Signature

September 19, 2021

Date

Acknowledgements

I had a great deal of help while working on and completing this project, and I would like to say a few words to thank those responsible.

First, this research was supported in part by the National Research Foundation of South Africa (Grant number: 118075), as well as Centre for High Performance Computing (CHPC), South Africa, for providing computational resources to this research project under CSCII340. I am thankful for these contributions.

Without my supervisor, Dr. Richard Klein, I am not sure what I would have done. You always had help to give when I needed it most, you organised computational resources that became the backbone of this project, and your weekly meetings helped stabilise an otherwise difficult two years. Thank you dearly for all the effort that you put in to helping me with my project.

I would like to thank the School of Computer Science and Applied Mathematics, its staff and its students, for being welcoming and always willing to talk. I would like to thank everyone from the RAIL Lab for the many interesting conversations I had, and because of these conversations I found my research topic.

I would also like to thank my family and friends for supporting me when I was working seven days a week, and always believing that I could push forward and finish this project. Dhruv, knowing you were in the same boat as me, knowing that even though I felt like I had no idea what I was doing, I always knew that you too had no idea what you were doing, and that we would eventually figure it out and it would be alright in the end. Thalén, thank you for everything these last few years. You changed the way I think and the way I see the world, and you are always there for me when I need it most.

Finally, I would like to thank the authors and contributors of the open source projects and libraries that made this project possible, particularly:

- [PyTorch](#), the astonishingly powerful deep learning library that they give away for free...
- [The Python programming language](#), the glue that makes everything work.
- [PlotNeuralNet](#), the library that allows me to reliably create consistent neural network architecture diagrams in a $\LaTeX$.

Contents

Preface

Abstract	i
Declaration	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	vii

List of Figures	vii
------------------------	------------

List of Tables	ix
--------------------------	----

List of Tables	ix
-----------------------	-----------

1 Introduction	1
-----------------------	----------

2 Background	4
---------------------	----------

2.1 Introduction	4
2.2 Convolutional neural networks	4
2.3 Latent spaces	5
2.4 Encoder-decoder networks	5
2.5 Semantic segmentation	6
2.6 Generative networks	7
2.6.1 Non-conditional generators	7
2.6.2 Conditional generators	8
2.7 Generative Adversarial Networks (GANs)	9
2.8 Residual networks	11
2.8.1 Initial architecture	11
2.8.2 Architecture developments	12
2.9 Feature matching loss	12
2.10 Normalisation	13
2.10.1 Batch normalisation	13
2.10.2 Layer normalisation	14
2.10.3 Instance normalisation	14
2.10.4 Weight normalisation	14
2.10.5 Other normalisation methods	15
2.11 Optical flow	15
2.12 Conclusion	16

3	Related work	18
3.1	Introduction	18
3.2	Image synthesis	18
3.2.1	Cascaded Refinement Network (CRN)	19
3.2.2	pix2pixHD - A conditional GAN	21
3.3	Video synthesis	24
3.3.1	vid2vid - A conditional GAN for video synthesis	25
3.3.2	Other video networks	28
3.4	Conclusion	28
4	Research methodology	30
4.1	Introduction	30
4.2	Network variations	30
4.3	Research questions and hypotheses	31
4.4	Dataset	32
4.5	Experiment Methodology	32
4.5.1	Dataset preparation	32
4.5.2	Network recreation, architectures, and deviations	32
4.5.3	Training methodology	35
4.5.4	Evaluation metrics	36
4.6	Conclusion	38
5	Results	39
5.1	Introduction	39
5.2	Quantitative analysis	39
5.2.1	Comparison of the CRN and pix2pixHD base configurations [Research question 1]	41
5.2.2	Porting components from pix2pixHD to the CRN [Research questions 2 & 3]	42
5.2.3	Including prior frame information in synthesis and training processes of the networks [Research question 4]	43
5.2.4	Modifications of both networks to incorporate optical flow [Research question 5]	45
5.2.5	Networks with equal opportunity	46
5.3	Qualitative and general analysis	48
5.3.1	Effects of normalisation on synthesis output	48
5.3.2	The effect of the feature encoder	49
5.3.3	Perceptual loss issues	50
5.4	Conclusion	51
6	Conclusion	53
6.1	Discussion	53
6.2	Future work	54
A	Model components	56
A.1	Residual refinement module	56

References	57
Bibliography	57

List of Figures

2.1	Encoder-decoder Architecture - The layers in red denote the input and output of the network, with each block being the output of the respective operation it is labeled with. Each block is also labeled with the number of channels below it.	6
2.2	An image and its corresponding segmentation annotation - CityScapes dataset by Cordts <i>et al.</i> [2016]	7
2.3	A Gaussian distribution Q is fitted to samples drawn from a distribution P by minimising varying configurations of the JS divergence. Box A shows the ground truth distribution P . Box B shows the configuration where the JS divergence is set to mimic that of the KL divergence. Q attempts to encompass the outlier samples as well as the central mass, and as such includes a large amount of area that is not in P . In box C , Q covers a large amount of the samples provided, without including the erroneous area that was included in B . Box D demonstrates the configuration where the mode is prioritised, and as such a large amount of the samples are not covered by Q . This figure was taken from Huszár [2015].	10
2.4	A single residual block as proposed by He <i>et al.</i> [2016a]. The skip connection adds the input to the output of the normalisation layer. Following the addition, this is fed into a ReLU layer and the output is ready for use as the input to another network layer or residual block. The dimensions of the layers are kept consistent throughout the process.	12
2.5	A comparison of sparse and dense optical flow estimation, overlayed on a corresponding reference image. Colours are often used in optical flow visualisation to convey additional information, for example, in the sparse optical flow example, the different colours allow one to differentiate between different flow trails, whereas in dense flow the direction of the movement is shown through colour.	16
3.1	Style transfer as proposed by Gatys <i>et al.</i> [2016].	19
3.2	An example input and output from the CRN. Taken from Chen and Koltun [2017]	20
3.3	A CRN with two refinement modules. Semantic segmentation annotations are shown in colour for ease of viewing.	20
3.4	An example input and output from pix2pixHD, as well as a comparison to the output from the CRN. Taken from Wang <i>et al.</i> [2018b]	22

3.5	pix2pixHD global generator network architecture. <i>SConv</i> - Strided convolution; <i>Res</i> - Residual block; <i>TConv</i> - Transpose convolution. All convolution blocks, except <i>Final Conv</i> , are Conv-InstanceNorm-ReLU. Some strided convolution and transpose convolution blocks are excluded for the sake of simplicity.	23
3.6	vid2vid global generator network architecture. <i>SConv</i> - Strided convolution; <i>Res</i> - Residual block; <i>TConv</i> - Transpose convolution. All convolution blocks, except <i>Final Conv</i> , are Conv-InstanceNorm-ReLU. Some strided convolution and transpose convolution blocks are excluded for the sake of simplicity. . .	26
5.1	Training time in days for each network configuration.	40
5.2	A two-frame comparison of each of the CRN base configurations, indices 1, 2, and 3.	42
5.3	A two-frame comparison of the pix2pixHD base configuration, index 11. . .	42
5.4	A single output image from the CRN at indices 3, 4, 5, and 6. Notice how adding the components has a minor effect on image quality and content, until index 6 (bottom right), where the addition of the discriminator dramatically increases image complexity.	44
5.5	A two-frame comparison of the outputs of the CRN, without (top, index 6) and with (bottom, index 8) the use of prior frame information. Noticeable differences between the two pairs include the road lines and the face of the bus, which can be clearly seen in the difference images on the right. The images have been brightened to better show off the changes in temporal consistency	45
5.6	The tenth frame of the synthesised video from the CRN optical flow configuration (index 9), as well as the corresponding weight annotation. Lighter areas denote candidate image use, whereas dark areas denote warped image use. Note the distortion in the building sections and the repeated edge of the car.	46
5.7	The tenth frame of the synthesised video from the pix2pixHD optical flow configuration (index 14), as well as the corresponding weight annotation. Lighter areas denote candidate image use, whereas dark areas denote warped image use. Compared to the CRN, far less of the warped image is used as the warping is effectively done internally in the network.	47
5.8	A comparison of the outputs of the CRN (left, index 6) with pix2pixHD (right, index 11), showing how the CRN can match pix2pixHD in detail when both networks share the same components.	47
5.9	Different synthesis artefacts encountered in certain network on the same video frame. configurations	49
5.10	Example of a difference the feature encoder can make to the synthesised output. An easily noticed difference in nearly all configurations is the opportunity for different coloured cars.	50
5.11	The difference between the synthesised output of the CRN's original implementation and our implementation (index 1).	51

List of Tables

5.1	FVD scores for all network configurations.	40
5.2	FVD scores for base network comparison configurations.	41
5.3	FVD scores for porting components network comparison configurations. . .	43
5.4	FVD scores for prior frame network comparison configurations.	44
5.5	FVD scores for optical flow network comparison configurations.	45
5.6	FVD scores for equal opportunity network comparison configurations. . . .	47
5.7	FVD scores for network configurations relating to normalisation effects. . .	48
5.8	FVD scores for network configurations relating to feature encoder effects. . .	49
5.9	FVD scores for network configurations relating to perceptual loss issues. . .	50

Chapter 1

Introduction

Synthetic video has been used for many years, be it in animated movies or video games. The process of creating it is usually long and cumbersome, including creating the rendering engine, 3D assets, and scenes. It requires many hours, days, or possibly months before an initial version of the scene can be seen. While the outputs of these processes are often fantastic, there is a need for the ability to create photorealistic video using quicker means.

Image synthesis offers a possible solution to this problem, whereby a basic layout of the scene can be given to a network that, using this information, creates a photorealistic scene that matches the description of the input. The content in the output image would be based on the data that the network was trained on, and if one has a specific domain that they would like to create content for, such as particular architecture styles or certain room interior styles, the network can be trained to match these styles.

Video synthesis is the logical extension to image synthesis, where instead of synthesising single frames the network synthesises videos. Obvious uses of an extremely competent network would be as a replacement to the renderers used in video games, where the creator of the game would not have to worry about the difficulties of simulating realistic lighting, or creating high-quality assets since the network would handle these aspects naturally.

While current video synthesis networks are not capable enough to replace modern 3D graphics engines yet, great progress is being made. One of the main challenges of video synthesis is that of temporal consistency. While each frame could be a perfectly acceptable image, there is a need for the frames to follow on from each other so that when played together as a video there are smooth transitions between frames, with objects in the videos moving in a realistic manner depending on the motion of the camera.

To handle the problem of ensuring temporal consistency, video synthesis networks contain components that help ensure that the outputs are more consistent; however, this comes at the cost that the networks are becoming increasingly demanding to train, requiring more time to train and more VRAM. Video networks are often trained on GPU workstations costing millions of rands, making this inaccessible to most people not working for organisations with large deep learning budgets.

Our work aimed to investigate the possibilities of using image synthesis networks to synthesise temporally consistent video, and more specifically, a comparison of two networks when set to the task, namely the Cascaded Refinement Network (CRN) [Chen and Koltun 2017], and pix2pixHD [Wang *et al.* 2018b]. While both networks aim to achieve the same goal, synthesise photorealistic images at 1024x2048 resolution, they differ greatly in architecture, with pix2pixHD being proposed as a direct competitor to the CRN, having extra components and features for which the CRN does not have an equivalence.

These networks were recreated and tested in their base configurations; however, there was an opportunity to extract more insight from the workings of these networks by creating variants of these networks in an attempt to improve their ability to synthesise temporally consistent video. There are two classes of possible variations, one being the porting of the extra features from pix2pixHD to the CRN, and the other being the inclusion of video-specific features such as using prior frame information and optical flow estimation in the synthesis process. We used the vid2vid network [Wang *et al.* 2018a] as a guide on how to adapt the image synthesis networks to produce video, closely following the concepts and implementation used to guide our own implementations.

We implemented both networks in a generic manner so that they can share implementations for components to avoid unnecessary work, and to ensure similar configurations that allow comparison. While the base networks can be trained on more modest hardware, the more extensive modifications to the networks required more powerful hardware and longer training times. Thus, concessions for the sake of comparability, training time, and hardware availability were made, with the greatest attempt being made to ensure the resultant networks are on equal standing. This takes the form of lowering the network output resolution by half, as well as not using every feature from vid2vid in our network adaptations.

These networks and their variants were tested using both a temporal consistency metric for quantitative analysis, as well as our own inspection for qualitative analysis, focusing on the influences of the different component variations to the final results, the relationship between image quality and temporal consistency, and the differences in how the individual networks adapt to their new tasks. We took a specific interest in the CRN, since it lacked many features of pix2pixHD, and thus we also directed our focus to how the network adapts to the use of these new features. We provide the videos used for the testing of each network variation via links to YouTube videos, as well as the reference video they are tested against.

In our testing we found that pix2pixHD greatly outperforms the CRN in their base configurations; however, when the extra pix2pixHD features are ported to the CRN, this gap closes considerably. As can be expected, the addition of prior frame information to any network configuration led to the greatest temporal consistency improvement of all the component additions, and synthesised videos that sufficiently blend frames together to be considered smooth. Finally, the addition of optical flow estimation and its supporting network and training method modifications led to mixed results, with both the CRN and pix2pixHD directly reusing previous pixel data; however, the CRN could not correctly shift them to their new locations leading to severe degradation as the video progresses. pix2pixHD on the other hand better handled the required pixel movement and saw improvement, becoming the best performing network configuration.

Following this introductory chapter, chapter 2 contains the underlying concepts that are necessary for understanding our work, followed by chapter 3 which describes some prior work in the image synthesis domain, and details about the CRN, pix2pixHD, and vid2vid networks. Chapter 4 contains descriptions of the network configurations we tested, our research questions and hypotheses, and our methodology used to conduct our testing. In chapter 5, we present our results, including both quantitative and qualitative results as described above, and finally, chapter 6 concludes this work.

Chapter 2

Background

2.1 Introduction

Temporally consistent video synthesis requires complicated neural networks, touching on many different concepts. This chapter describes the topics that are used to construct and use these networks. Not all sections directly build on each other; however, they are all important for the networks that are presented and developed in this work.

We give a brief overview of convolutional neural networks, a vitally important network architecture underpinning modern computer vision networks, and then describe latent spaces, a crucial factor in how synthesis networks produce their outputs. Encoder-decoder networks are discussed, a network architecture that has seen much success for its power, simplicity and effectiveness in many tasks. We then describe semantic segmentation, a popular and well-studied problem, important for understanding concepts used by further topics. Generative networks are looked at in-depth as they are the basis for our work, with generative adversarial networks following, a generative network training framework that allows for more intricate training. We then give an overview of residual networks, a method of allowing networks to use all information from earlier in the network at later stages. Feature matching losses are then explained, where the activations of classification networks are used for training generative networks, accessing outputs in ways not possible with hand-crafted loss functions, followed by normalisation, a critical component in building synthesis networks, which has a significant impact on the ability of networks to train correctly and on the quality of the outputs. Finally, an overview of optical flow is given, a significant but complicated component in the attempt to achieve temporal consistency.

2.2 Convolutional neural networks

Convolutional Neural Networks (CNNs) are a type of network architecture that makes use of the convolution operation. Inspired by the visual processing method of data from an animal's eye as detailed in [Hubel and Wiesel \[1968\]](#), they have proven extremely successful for their

computational efficiency due to the use of smaller amounts of data for each output, and their ability to take into account spatial locality. While the concept dates back many years, with [LeCun et al. \[1995\]](#) detailing its usage within a machine learning context, they have dramatically increased in popularity with the rise of GPU computing allowing previously impossible-to-train networks to train in a reasonable time.

While they are often known for the processing of images, there are 1D (signal processing), 2D (image processing), and 3D (video processing) variants that allow them to be used in many settings. In the context of images, there are a variety of options when using CNNs as the standard operation can maintain image sizes or be used to downsample the input, as well as operations such as a the transpose convolution that can be used to increase the image size. Any convolution operation can also modify the number of channels in the output compared to the input. This allows convolution operations to be used throughout a network architecture, and as such have become a staple in computer vision network architectures.

2.3 Latent spaces

A latent space is a domain that contains some compressed representation data that is used by the network when synthesising the output. A flexible concept, the essence of which revolves around lossy compression. While not related to deep learning, principle component analysis (PCA) [[Pearson 1901](#)] contains a good example of a latent space. Once the principal components have been chosen, the domain of vectors that describe how to recreate the original data using the principle components is a latent space. These vectors, if the principal components are known, are inputs in a process where the output is the original data to some degree of accuracy.

In deep learning networks a latent space can be a bottleneck in an encoder-decoder network (section 2.4) where the essence of the input is compressed and used by the remaining section of the network to create the final output. In the context of a non-conditional generative network (section 2.6.1) the input to the network is often sampled randomly from a chosen domain, for example, a two-dimensional real number vector space, and during training, the network associates outputs to these inputs by creating a learned representation of the output data in the input space. This input domain is a latent space in the network as it is the seed that the network uses to synthesise the output.

In general, latent spaces are compressed representations of data at some point in a network, be it the input to a generative network, or the bottleneck section in the centre of an encoder-decoder network. An often applicable test is that of data in the region being able to be *hand-modified*, for example, modifying the input of the network, or changing values in the bottleneck section in an encoder-decoder network to try and achieve a desired feature in the output [[Bojanowski et al. 2017](#)].

2.4 Encoder-decoder networks

The encoder-decoder network architecture has proven to be an effective tool in creating networks that are flexible and perform well. The premise is a network that takes as input

some data that is in a similar domain to that of the output, processes and encodes that data into a central bottleneck section. It then decodes that bottleneck into the output. Figure 2.1 gives an example of a network following the encoder-decoder architecture.

Depending on the architecture, the bottleneck can be seen as a latent space. For example, in an auto-encoder network, where the network is trained to recreate its input, the bottleneck is usually constrained and one single layer. This layer can then be modified by hand to produce different results based on the data. On the other hand, pix2pixHD [Wang *et al.* 2018b] has a multi-layer bottleneck that is used as the primary processing section of the network and is used as such due to the low resolution of the data in this section allowing for an increase in both convolutional layers, as well as filters in these layers. This bottleneck does not support hand-modification of values, nor is it intended to be a latent space.

As expressed above, the output of the network is usually a similar domain to that of the input. By this, it is meant that if an image is the input of the network, something based on the image could be the output, for example in semantic segmentation (section 2.5) a colour image as the input to produce a segmentation annotation, or a segmentation annotation as the input to produce a colour image as in a conditional generative network (section 2.6.2).

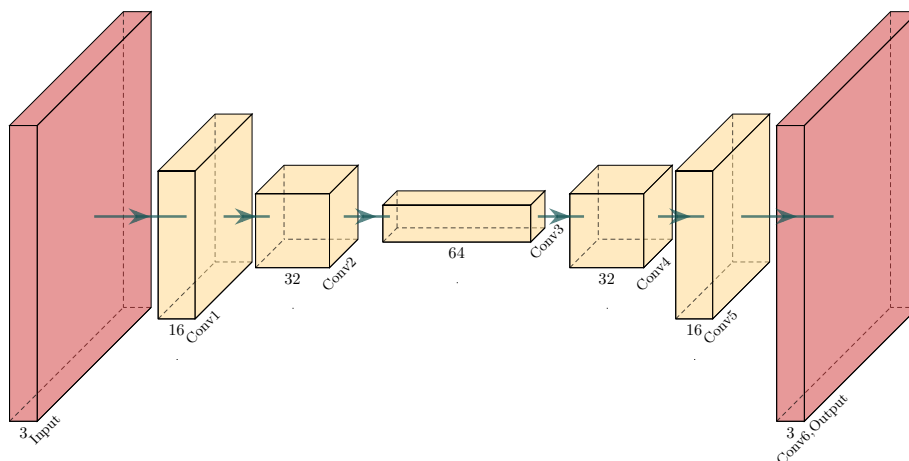


Figure 2.1: Encoder-decoder Architecture - The layers in red denote the input and output of the network, with each block being the output of the respective operation it is labeled with. Each block is also labeled with the number of channels below it.

2.5 Semantic segmentation

Semantic segmentation is the process of classifying every pixel in an image into a class. This classification is useful as a step in a larger computer vision pipeline, for example, a self-driving car might use segmented images as well as depth information from a separate camera to make decisions about how to proceed. The segmentation is useful as there is often a large amount of unnecessary detail in an image that is not required to make decisions, and would only serve as noise to the rest of the system.

Long *et al.* [2015] were the first to create a network that was trained end-to-end for the purpose of semantic segmentation. Previously, the computational power required to perform semantic segmentation in this manner was insurmountable; however, with the use of modern Graphics Processing Units (GPUs) to train machine learning networks, this has become feasible. Since then, the problem has been explored further with networks getting progressively better at labelling difficult scenes, and as such, it is considered reliable to use a segmentation network to synthesise training data as, among others, Wang *et al.* [2018a] have done for the training of their video synthesis network vid2vid.

The output of a semantic segmentation network is known as a segmentation annotation, an example of which can be seen in figure 2.2.

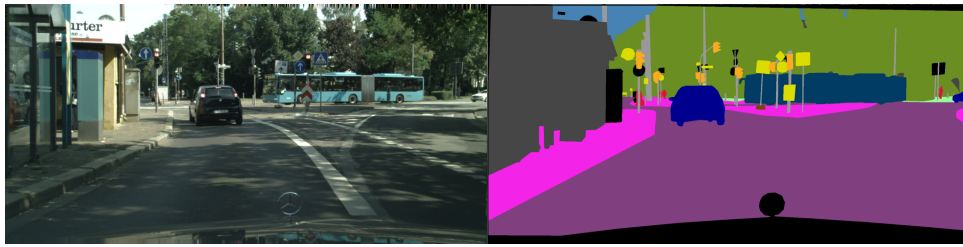


Figure 2.2: An image and its corresponding segmentation annotation - CityScapes dataset by Cordts *et al.* [2016]

2.6 Generative networks

Generative networks, or generators, are used to create data that can pass as if it were drawn from the data distribution that the network is trained on. This can be seen somewhat like the inverse of a classification network since the network takes a smaller amount of data as input than the amount that it produces as output.

The input to the network is often a sample drawn from a latent space, with synthesised images being created based on the input in an effect that can be likened to decompression. The distribution of possible inputs also affects the distribution of possible outputs, whereby constraining the possible values of the latent space causes the network to tend closer to the mode or commonly occurring characteristics that can be found within the dataset. If the input is a latent space, it can be *hand-modified*, or interpolated, to create novel results that are not in the dataset; however, these novelties are still only a combination of characteristics found in the dataset.

There are two types of generators: non-conditional and conditional. These networks can be trained in a variety of ways, with many modern high-resolution networks being trained adversarially, the process of which is explained further in section 2.7.

2.6.1 Non-conditional generators

Non-conditional generators take as input a vector of fixed size that is usually drawn from a predefined domain. This vector has no meaning outside the context of the generative network, and as the network learns meaning is associated with the values of this input latent

space. Drawing from the PCA example used above in section 2.3, the principal components would be the non-conditional generator, and the vector containing the proportion of the principal components to combine is analogous to the input to that network. Karras *et al.* [2020] propose a non-conditional generative network famous for synthesising realistic faces of people that do not exist on their website, where there is no user input, only a synthesised result from a random sample of the latent space of the network.

This learned mapping can be exploited however, for example, a network that synthesises pictures of faces might be able to do the following: find the mean of inputs that lead to the synthesis of people with sunglasses, and subtract from that the mean of the inputs that synthesise people without sunglasses. Adding the resulting vector to an input that synthesises a person that is not wearing sunglasses should, though is not always guaranteed to, add sunglasses to that image. Due to the complexity of generators as well as their specific network architectures and the domain of the input, the manner in which the input to the network relates to the output of the network is different for every network; however, relationships like the one explained above often present themselves. Thus, if *hand-modifying* inputs is something that one wants their network to support, it must be kept in mind when designing the network.

2.6.2 Conditional generators

Input to conditional generators contains information about the characteristics of the synthesised data. For example, the network created by Chen and Koltun [2017] takes a segmentation annotation as input and synthesises the corresponding scene that the annotation describes.

During training conditional generators can have a significant advantage over non-conditional generators in that there is often an opportunity to directly test the synthesised output against data relating to the input of the network, or the input itself, allowing for a more specific evaluation of how well the network is performing. For example, pix2pixHD [Wang *et al.* 2018b] takes as input a semantic segmentation annotation, which is nearly always accompanied by the ground truth image. This allows the synthesised image to be directly compared to the ground truth image, whereas a non-conditional generator would not have this opportunity.

This explicit relationship between the input and synthesised output means the input latent space has a better chance of learning more diverse and structured representations, as well as one's ability to direct the content that is mapped to the latent space. The inputs are also far easier to *hand-modify* since they can contain more explicit real-world meaning outside the context of that specific network.

Conditional generators are also useful as components in larger synthesis pipelines. For example, the network by Johnson *et al.* [2018] converts scene graphs into segmentation annotations, then using that segmentation annotation synthesises the resulting image to match the scene graph. A non-conditional generator does not accept input in such a structured manner, and thus will not allow preceding components in a pipeline.

2.7 Generative Adversarial Networks (GANs)

Introduced by [Goodfellow *et al.* \[2014\]](#), GANs are comprised of a generator that synthesises data and a discriminator that attempts to classify data as real or fake. In the context of images, the generator attempts to construct realistic images that can fool the discriminator. This forms a competition between the generator learning how to outsmart the discriminator, and the discriminator learning how to detect increasingly better-synthesised images. The standard definition of this minimax competition as presented by [Goodfellow *et al.* \[2014\]](#) is thus:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.1)$$

where G is the generator network, D is the discriminator network, x the training data, and z is an input from the latent distribution p_z . As described in section 2.6, z can be as simple as a random vector for a non-conditional generator, or a set of structured instructions for a conditional generator. The output for the discriminator in the case of equation (2.1) should be 1.0 for a real image, and 0.0 for a synthesised image. This equation is in effect binary cross-entropy, and a high loss is given when the discriminator correctly identifies both real and fake images, and a low loss is given when misidentifying them. The discriminator tries to maximise this value, preferring correct predictions, whereas the generator attempts to minimise this value, leading to incorrect predictions.

The discriminator acts as a powerful and continually adapting loss function that can learn to holistically examine many different aspects of the input image, including complicated details that might be impossible to program by hand. This can allow GAN generators to learn to synthesise results that fit into the training dataset without directly reusing elements in the dataset. Image synthesis is a difficult process due to there being a large number of factors that go into creating acceptable images. As GANs have been shown to succeed where prior architectures have battled there has been great interest in applying the architecture to many different image synthesis problems, although GANs have been used in other fields such as music synthesis, for example, the network proposed by [Engel *et al.* \[2019\]](#).

In practice, GANs are a variation of the training method for a generative network since the generator in a GAN is no different from any other generative network, and the discriminator is only used during training. This is further exemplified by the fact that one is not limited to using discriminators as the only loss function, as other conventional losses can be added as well. The discriminator can also be given extra data that it might use to make its decision, such as the input to the generator in the case of a conditional generative network, which for example can be seen in pix2pixHD [\[Wang *et al.* 2018b\]](#), where the semantic segmentation annotations used by the generator are also given to the discriminator to check the image for conformity.

The discriminator trains by being given samples of real and fake data and is then scored on how well it can tell them apart. The generator on the other hand is updated via the results of the discriminator scoring the data that is synthesised, and as such, does not have its output directly compared to the training data. In this regard, the training of the discriminator can

be classified as supervised learning; however, the training of the generator is unsupervised with respect to the dataset.

An important topic expressed by [Goodfellow *et al.* \[2014\]](#) is that of the GAN minimising the Jensen-Shannon (JS) divergence instead of the usual Kullback-Leibler (KL) divergence between the learned distribution and the distribution of the dataset. The KL divergence has a number of drawbacks, namely that it is not symmetric, and has the possibility to be infinite. The JS divergence is a weighted sum of two KL divergences, configured in such a way as to be symmetric and have no possibility of being infinite. The default weighting is an equal share of both KL components.

The KL divergence is likely to try to include all the data at the expense of poorly modelling intricacies of the data and can be likened to finding the mean of the data, which can be a poor representation if the distribution is not normal. On the other hand, finding the mode of the data may not be a good representation either, as it may be too constrained. The JS divergence strikes a good balance between covering the dataset distribution and learning specific representations [\[Huszár 2015\]](#). Figure 2.3 gives an example of the differences between minimising the two different divergences.

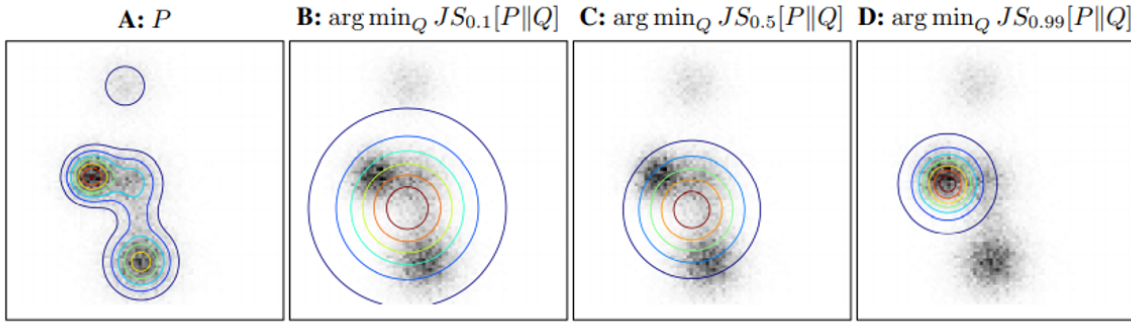


Figure 2.3: A Gaussian distribution Q is fitted to samples drawn from a distribution P by minimising varying configurations of the JS divergence. Box **A** shows the ground truth distribution P . Box **B** shows the configuration where the JS divergence is set to mimic that of the KL divergence. Q attempts to encompass the outlier samples as well as the central mass, and as such includes a large amount of area that is not in P . In box **C**, Q covers a large amount of the samples provided, without including the erroneous area that was included in **B**. Box **D** demonstrates the configuration where the mode is prioritised, and as such a large amount of the samples are not covered by Q . This figure was taken from [Huszár \[2015\]](#).

In training, [Goodfellow *et al.* \[2014\]](#) offered proofs that the competition in equation (2.1) has an equilibrium where the generator is synthesising samples that fit perfectly into the training dataset, and thus the discriminator cannot discern between a real and fake sample. In practice, however, there are a number of difficulties that can occur when training GANs, and their behaviour is often different to what was specified by [Goodfellow *et al.* \[2014\]](#). GANs are prone to the vanishing gradients problem, whereby the discriminator is able to discern between a real and fake sample with absolute surety, leading to no gradient flow from the discriminator to the generator during training, as well as mode collapse, where the generator will repeatedly synthesise the same sample regardless of the input. A number of

modifications to the training algorithm are proposed, such as changing cross-entropy loss in equation (2.1) to a different loss function, such as mean-squared error [Mao *et al.* 2017], or Wasserstein loss [Arjovsky *et al.* 2017], as well as including additional loss functions [Wang *et al.* 2018b; Dong *et al.* 2019; Wang *et al.* 2018a].

2.8 Residual networks

Proposed as a means of building longer networks without facing the vanishing gradients problem, residual networks make use of regular skip connections to carry earlier data in a network to later parts of the network, allowing multiple paths through the network for data to travel through. While skip connections predate residual networks, the way that these skip connections are used is key to the success of the architecture. Residual networks make use of residual blocks that could be the typical Conv-Norm-ReLU format; however, the key difference is the inclusion of a skip connection from before the data enters the Conv layer, to just before the final ReLU layer. The network is typically made up of a succession of these blocks, and with the skip connections existing throughout the entire network, a path can exist from the beginning of the network to the very end. Residual networks have had great success in both classification networks, such as the original network architecture by He *et al.* [2016a], to generative networks such as pix2pixHD [Wang *et al.* 2018b].

2.8.1 Initial architecture

He *et al.* [2016a] developed the original residual network architecture as a means of creating deeper classification networks and avoiding the vanishing gradients problem that comes with it. As a network grows in length, it may initially see a performance increase; however, after a point, the performance will begin to decrease due to vanishing gradients during training. This is an issue as deeper networks allow more complex concepts to be learned by the network.

The network proposed is made of a series of residual blocks of the form described in figure 2.4. As a classification network, the strategy is to reduce image size gradually, while increasing the number of channels. Every few residual blocks the image size is halved and the number of channels doubled. Since skip connections add prior activations to later activations element-wise, the dimensions of these activations need to match. After the image size is halved, an additional operation to regain compatibility for the element-wise addition is required. Thus, two different strategies are also tested to allow ill-fitting prior activations to be used in the skip connection. The proposed network is said to be easier to train for the same task than VGG-19 [Simonyan and Zisserman 2014], while having nearly twice as many convolutional layers.

During training, the network can learn to incorporate data from earlier parts of the network, creating training paths that may skip blocks depending on whether or not the information from that block is useful. This also leads to a levelling of contribution in the network, where the removal of a single residual block does not necessarily prevent the network from functioning.

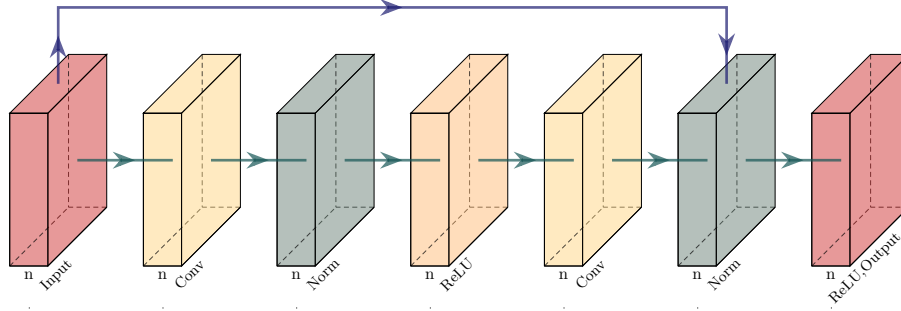


Figure 2.4: A single residual block as proposed by [He *et al.* \[2016a\]](#). The skip connection adds the input to the output of the normalisation layer. Following the addition, this is fed into a ReLU layer and the output is ready for use as the input to another network layer or residual block. The dimensions of the layers are kept consistent throughout the process.

2.8.2 Architecture developments

While the premise of the residual network has remained the same, further investigation has been done to refine the concept. [He *et al.* \[2016b\]](#) proposed using preactivated residual blocks, where the normalisation and ReLU layers are placed before the convolution layer instead of after it. This simple change provided a substantial reduction in test error on their 1001-layer ResNet.

[Xie *et al.* \[2017\]](#) propose a further modification to the ResNet architecture by making use of grouped convolutions to increase what they call the cardinality of the residual block. This is said to reduce complexity while maintaining the accuracy of the network. The proposed network, ResNext, with 101 layers is said to perform better than ResNet with 200 layers while having only 50% of the parameters.

The standard process for ResNets is to have a series of residual blocks, with skip connections only crossing one block. [Huang *et al.* \[2017\]](#) propose a network architecture comprised of a smaller number of large *dense* blocks, where the output of each layer inside the *dense* block is used as an input for all further layers inside that *dense* block. This input is not summed together in the manner of prior ResNets and is instead simply concatenated together. This means that, within a single *dense* block, the number of input channels in each subsequent layer increases, with a reduction happening between dense blocks to reduce these channels and prevent the number from exploding as the data progresses through the overall network. A convolutional and pooling layer is placed between each *dense* block, which performs this reduction. The network achieves state-of-the-art results while remaining computationally efficient.

2.9 Feature matching loss

The premise of feature matching loss is comparing the activations inside a network for two competing inputs, usually a reference image and a synthesised image. Based on work by [Gatys *et al.* \[2016\]](#), and formally introduced by [Johnson *et al.* \[2016\]](#) under the name perceptual loss, the premise of a feature matching loss is that a network that has been

trained to classify images contains many learned features about what defines the content in the images, and by comparing the activations of the classification network layers you get a more holistic evaluation of the synthesised image. In practice, perceptual losses are a subset of feature matching losses and use a pre-trained classification network as a basis.

[Chen and Koltun \[2017\]](#) presents a suitable equation for a feature matching loss, with $\Phi_l(\cdot)$ representing a selected layer activation from classification network Φ ,

$$\mathcal{L}_{I,L}(\theta) = \sum_l \lambda_l \|\Phi_l(I) - \Phi_l(g(L; \theta))\|_1. \quad (2.2)$$

Here, I is the reference image, L the input to the synthesis network, θ the set of synthesis network parameters, and $g(L; \theta)$ the output image from the synthesis network. Comparisons between the layer activations using the reference image I versus the synthetic image $g(L; \theta)$ are made and summed, resulting in a single loss value for the synthetic image compared to the reference image. This comparison is highly customizable, and in this case the comparison is the first norm of the difference between the two activations. λ_l is a predefined scaling factor for that particular layer, intended to weight the importance of that layer's activations as desired.

[Chen and Koltun \[2017\]](#) makes use of a perceptual loss, whereby a pre-trained VGG-19 [[Simonyan and Zisserman 2014](#)] network is used as the feature matching network. [Wang et al. \[2018b\]](#) makes use of a feature matching loss by using the discriminators for their proposed network, which begin untrained and train alongside the generator. This was developed as a means of increasing training stability due to GANs having issues with vanishing gradients when discriminators approach perfect confidence. This exploits that confidence and continues to feed the generator the information that the discriminators are using to correctly classify the synthesised images. This creates an ever-improving, tailored feature matching loss, leading to better and more stable training of the generator.

2.10 Normalisation

Normalisation has become a staple in training networks, allowing faster and more stable training. While there is no concrete explanation for why normalisation has the profound effect that it has, the current explanation is that it makes the loss function surface smoother and easier to optimise against [[Santurkar et al. 2018](#)].

2.10.1 Batch normalisation

Batch normalisation [[Ioffe and Szegedy 2015](#)] is perhaps the most well known and used normalisation method in computer vision. It normalises each neuron of the input in a minibatch with regards to the mean and standard deviation over that neuron in all samples in the minibatch.

There are some restrictions to the use of batch normalisation. Calculating the mean and variance does not work if the minibatch size is one, since the variance will be zero, as well

as the statistics being heavily dependent on the size of the minibatch, with an increasing noisiness as the minibatch size decreases. Batch normalisation cannot be used for inference, as the batch size is not guaranteed or may even be one, and as such a running record of the mean and variance are updated during training and used instead. Batch normalisation also includes a learnable gain and bias for each channel in the input that helps adapt the normalisation to the specific workload, as sometimes a mean of zero and a variance of one is not desired.

2.10.2 Layer normalisation

Layer normalisation [Lei Ba *et al.* 2016] was proposed as a means of attaining the benefits of batch normalisation in situations that demanded a minibatch size of one.

Layer normalisation is described as a transpose operation of batch normalisation, normalising each image by the statistics of that image, instead of by the statistics of the entire minibatch. This decouples it from the size of the minibatch, resulting in an identical operation regardless of minibatch size. This also means that the operation during inference is identical to the operation during training. Like batch normalisation, the normalisation process includes learnable parameters for adapting the normalisation to the specific task.

2.10.3 Instance normalisation

Instance normalisation [Ulyanov *et al.* 2016] is yet another modification of batch normalisation, with a specific focus on stylisation networks. Also known as contrast normalisation, it aimed to allow networks that use images as inputs to not be affected by varying contrast between different samples in the dataset.

The difference between instance normalisation and batch normalisation is that, like layer normalisation, it has no dependence on minibatch size and thus can be used on any minibatch size, as well as having identical training and inference operations. The difference between layer normalisation and instance normalisation however is that while layer normalisation uses the entire image, instance normalisation calculates statistics for each channel in each image in a minibatch separately. Included are learned parameters for scaling and shifting the resulting normalised activations in the same manner as batch and layer normalisation.

2.10.4 Weight normalisation

Weight normalisation [Salimans and Kingma 2016] works on a different premise than batch normalisation based methods, in that instead of normalising activations inside the network, it normalises the weights of the network itself.

Weight normalisation reparameterises the weight vector $\mathbf{w}$ in the network into two components, a direction vector $\mathbf{v}$ and a magnitude scalar g . During the forward pass of the network, the weight vector is recreated with

$$\mathbf{w} = \frac{g}{\|\mathbf{v}\|} \mathbf{v}$$

where $\|\mathbf{v}\|$ is the Euclidean norm of $\mathbf{v}$. The parameters $\mathbf{v}$ and g are updated during back-propagation, and $\|\mathbf{v}\|$ is calculated on every forward pass ensuring that $\|\mathbf{w}\| = g$.

Weight normalisation can be used instead of batch normalisation and does not have any requirements on the size of the minibatch, as well as being identical during training and inference. Other than the two parameters used instead of the weight vector, there are no additional learnable parameters.

According to [Salimans and Kingma \[2016\]](#), weight normalisation can offer a performance increase over using other types of normalisation when used on convolutional networks. Weight normalisation is also said to be nearly as effective as batch normalisation, without the additional noise introduced when minibatch sizes are small.

2.10.5 Other normalisation methods

Other forms of normalisation have been proposed, sometimes for specific use cases. Group normalisation by [Wu and He \[2018\]](#) is a generalisation of layer normalisation and instance normalisation. Group normalisation allows the grouping together of channels to be normalised, instead of only one channel in the case of instance normalisation, or all channels in the case of layer normalisation. Group normalisation, much like layer and instance normalisation, is presented as a way to reap the benefits of batch normalisation when minibatch size approaches single digits, without the instability that batch normalisation encounters.

Spectral normalisation, proposed by [Miyato *et al.* \[2018\]](#), is designed specifically for use inside the discriminators of GANs. Spectral normalisation is said to increase stability during training, something that GANs are known to have difficulties with, as well as being light, easy to incorporate into existing networks, and produce equal or better results than other normalisation techniques.

Spatially adaptive normalisation [[Park *et al.* 2019](#)], or SPADE normalisation, as a specialised normalisation alternative for image synthesis from semantic annotations. Combining the input and the normalisation process into one, a SPADE normalisation module is meant to prevent the detail from the input semantic annotation from being "washed out" when passing through normalisation layers. Instead of inputting the annotation at the beginning of the network, the annotation enters into the network at multiple points in the network. This is said to increase visual fidelity as well as the synthesised image's conformance to the input annotation.

2.11 Optical flow

Optical flow estimation is the process of determining how points in videos move between frames. A common use of optical flow estimation is in video encoding, allowing reuse of previous pixels to a predefined accuracy allowing large savings in storage space. There are two types of optical flow estimation, sparse optical flow estimation, where certain points or areas in a video are tracked, and dense optical flow estimation, where every single pixel in a video is matched between frames. Figure 2.5 shows an example of both sparse and dense optical flow estimation visualisations.



(a) Sparse optical flow over the course of many frames.[Bradski 2000]



(b) Dense optical flow for a single frame.[Wulff *et al.* 2012]

Figure 2.5: A comparison of sparse and dense optical flow estimation, overlaid on a corresponding reference image. Colours are often used in optical flow visualisation to convey additional information, for example, in the sparse optical flow example, the different colours allow one to differentiate between different flow trails, whereas in dense flow the direction of the movement is shown through colour.

Traditionally optical flow estimation has been calculated through iterative algorithms; however, in recent years a number of deep learning-based methods have emerged, specifically focusing on the problem of dense optical flow estimation. Dense optical flow estimation is understandably a more difficult problem, benefiting greatly from increased computing power in recent years. DeepFlow, [Weinzaepfel *et al.* 2013], began the process of integrating deep learning-based components into traditional methods, while Dosovitskiy *et al.* [2015] introduced FlowNet, the first end-to-end trained CNN based model, as well as a large synthetic dataset for learning dense optical flow estimation.

Ranjan and Black [2017] introduced the Spatial Pyramid Network with 96% fewer parameters than FlowNet while achieving near equivalent accuracy, and in the same year, FlowNet2 [Ilg *et al.* 2017] is 50% more accurate than FlowNet at the same inference speed, 8 fps, or as accurate as FlowNet at 140 fps, with multiple network configurations available depending on the needs of the user. Since then the field has expanded greatly, with networks consistently improving on the state of the art. In the survey performed by Hur and Roth [2020], the Volumetric Correspondence Network [Yang and Ramanan 2019] is 36.6% more accurate on MPI Sintel [Wulff *et al.* 2012] than FlowNet2; however, it only has 3.8% of the number of parameters.

2.12 Conclusion

This chapter focused on laying out the groundwork for understanding the image and video synthesis networks that follow in chapter 3. Convolutional neural networks form the basis of all networks used in this work, making use of convolution operations to incorporate spatial locality into the processing inside a network. Latent spaces are important as they help define how the input of the network maps to the output, offering ways to influence this process, which is seen in section 5.3.2. An overview of the encoder-decoder network architecture is given, involving compressing data into a central bottleneck and then decompressing it into the output, is powerful and flexible and used by many networks, including pix2pixHD

[Wang *et al.* 2018b], a network that we use in our experiments. An overview of semantic segmentation is given, the process of classifying each pixel in an image for easier use; however, these annotations also form the inputs to the networks that we are testing.

Generative networks are then described, networks that take less data as input than they produce as output. Non-conditional generative networks create outputs based on random input, whereas conditional generative networks synthesise outputs using, often by not always, human-interpretable inputs. This is important as all the networks that we recreate and use are conditional generative networks. The generative adversarial network architecture, an architecture that offers significant increases in output quality by having two networks compete to outsmart each other as they learn in tandem, plays an important role in our work as it is a major difference between the two networks we test. As such is was given a detailed overview.

Feature matching losses, a fundamental component in our work, use classification networks to extract features from two inputs, which are then compared and a loss calculated. Following this we give an overview of normalisation techniques, operations added inside networks that can dramatically improve training times and influence the quality of the network outputs, as can be seen in section 5.3.1. Finally, an overview of optical flow estimation is given, where points in the network are tracked over the course of multiple frames in a video, a useful concept in the exploration of temporal consistency and one subject of our experiments.

Chapter 3

Related work

3.1 Introduction

Our work builds on the work of others, and thus it is important to have a good understanding of this foundation and the surrounding work that attempts to achieve similar goals. Our work is focused on repurposing two image synthesis networks for video synthesis, thus these networks need to be understood to see how they differ and how they can be improved.

We begin in the image synthesis domain, describing some prior work as well as the base networks that we will be using as a foundation. We then cover video synthesis, describing the network upon which we will be drawing ideas in our attempt to improve the base image synthesis networks, as well as some alternate or more advanced networks for the sake of completeness.

3.2 Image synthesis

Image synthesis is the process of using generative networks to create images. With the rise of GANs, this has become an incredibly popular field due to the ability of the GAN architecture to adapt to difficult problems.

A popular topic has been that of neural style transfer [[Gatys *et al.* 2016](#)]. This involves combining a style image and a content image, such that the content image is transformed to have the style of the style image. Interestingly, the style transfer algorithm does not train a neural network to perform the operation. Instead, a pre-trained network, in this case the original VGG network [[Simonyan and Zisserman 2014](#)] is used to extract features for both images, then gradient descent is used to iteratively update a sampled noise vector into a combination of both images, using specifically calculated losses for content and style based on the extracted features. This differs from other generative networks in that it has all the information it needs in order to synthesise the output image, whereas usually a network is trained to add information based on the input to the network. The results of this process can be seen in figure [3.1](#).



Figure 3.1: Style transfer as proposed by Gatys *et al.* [2016].

A more *conventional* image synthesis problem is non-conditional image synthesis, synthesising images that can fit into the training dataset. Radford *et al.* [2015] proposed DCGAN, is a fully convolutional GAN that takes a noise sample as input and synthesises a 64x64 image. StyleGAN [Karras *et al.* 2019], is a non-conditional GAN that synthesises high-resolution images with state-of-the-art quality. Karras *et al.* [2020] then improve upon the results of StyleGAN with StyleGAN2, focusing on the issues that were identified in images synthesised by StyleGAN.

Some other examples of image synthesis include super-resolution, the process of synthesising an upscaled version of an input image, for example Haris *et al.* [2018], synthesising images from hand-drawn shapes, such as the network by Isola *et al.* [2017], as well as old photo restoration, the process of correcting damage and fadedness of old photos. Wan *et al.* [2020] makes use of both image inpainting to remove cracks in photos, as well as general colour restoration.

With the most relevance to our work, Chen and Koltun [2017] and Wang *et al.* [2018b] use semantic segmentation annotations as inputs to their networks to synthesise images that look realistic. This conditional image synthesis allows a diverse set of loss functions to be used due to the requirement for the output to follow the instructions of the inputs, leading to many methods of enhancing the training of the networks. These two networks are explained in detail in section 3.2.1 and section 3.2.2 respectively as they are the basis of our work.

3.2.1 Cascaded Refinement Network (CRN)

The CRN [Chen and Koltun 2017] is the first attempt at synthesising smooth and relatively detailed 2k (1024x2048) images from segmentation annotations. The network does not use a GAN architecture as it added instability to the training of the network and instead uses a perceptual loss. Example output from the CRN is shown in figure 3.2.

The network is a series of modules that operate at different scales of the synthesis process, each taking a copy of the network’s input data. The first module downsizes the input to 4x8 pixels and outputs at the same size. Subsequent modules operate at twice the resolution as that of the previous module, downsampling the input to the appropriate size, upsampling the previous module’s output, and concatenating the two together to form the input for that module. This continues until the output is at the desired size. The network does not

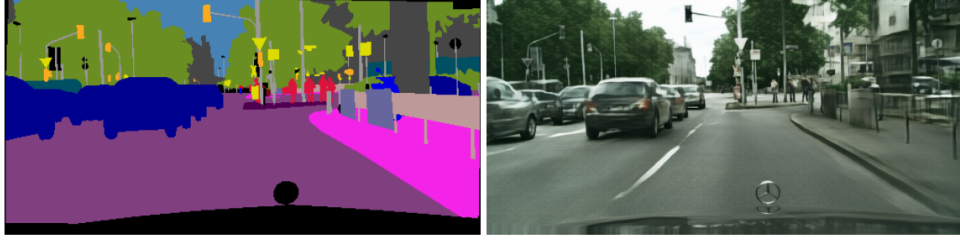


Figure 3.2: An example input and output from the CRN. Taken from [Chen and Koltun \[2017\]](#)

use transpose convolutions for upsampling and instead opts for bilinear interpolation for image/activation modification and nearest interpolation for segmentation annotations. The reasons given for such are that transpose convolution operations, due to the fact that they interpolate a smaller output into a larger one, can introduce repeating checkerboard patterns in the image in some cases, and this was sought to be avoided.

The stacking of modules that operate on an increasing resolution is the basis for the name of the network, in which a cascaded series of modules refine a coarse result from earlier modules into a more fine result in the latter layers. This method of building the network architecture allows one to choose the output resolution of the network, while each specific module always operates at the same resolution. This is something that cannot be easily accomplished with, for example, an encoder-decoder network where decreasing the input image size would effectively increase the percentage of the input image the convolutional layers in the network operate on, leading to inconsistent results. This effect can be seen clearly if pix2pixHD [\[Wang et al. 2018b\]](#) is trained using half-resolution images, where the quality of the synthesised images is dramatically decreased.

The refinement module

The modules that make up the CRN are known as refinement modules. Each module is made from two 3x3 Convolution-LayerNorm-LeakyReLU [\[Maas et al. 2013\]](#) blocks. After the final module, there is an additional 1x1 convolution that maps the high channel count output after the final leaky ReLU into 3 channels, the correct amount for a colour image.

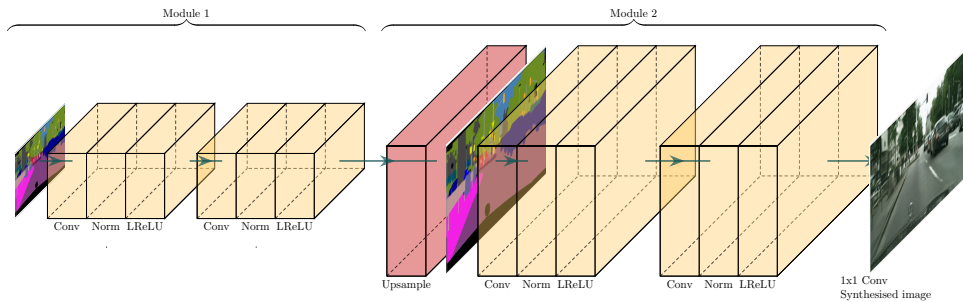


Figure 3.3: A CRN with two refinement modules. Semantic segmentation annotations are shown in colour for ease of viewing.

The filter count for the convolution layers inside the refinement modules for the 9 modules needed to reach the 2k resolution output size are as follows: *1024 for modules 0..4, 512 for modules 5 and 6, 128 for module 7, and 32 for module 8*. The high channel count is required

for the network to have the capacity to hold the required information to recreate acceptable samples of the dataset; however, this comes at a cost of memory usage. While the filter weight dimensions would remain constant throughout the network, the activations throughout the network increase in width and height as the image size increases, resulting in massive chunks of memory being required for each convolutional operation. This is why at certain points the filter count for the convolutions in the refinement modules decreases.

As mentioned prior, the input to the network takes the form of the semantic segmentation annotation that corresponds to the output image. This annotation is preprocessed by means of one-hot encoding, in which an annotation with n classes will create a tensor with n channels, with each channel containing ones only where the respective classes were in the annotation, with everywhere else being zero. While some networks will include a sample of noise in their input, the CRN does not include such and takes only the one-hot-encoded annotation as input. Each refinement module has a correctly scaled copy of this annotation, as well as the upsampled prior layers from the previous module if one exists.

The network begins training with enough modules to synthesise a 256x512 image. The majority of the training happens at this scale. After a certain number of epochs another refinement module is added, increasing the output resolution to 512x1024, and training resumes until a final refinement module is added, leading to an output resolution of 1024x2048, where training resumes for only a few epochs to fine-tune this layer since this layer only has 32 filters per convolution and does little more than clean up the upsampled input.

Diversity Loss

The CRN makes use of a perceptual loss that uses VGG-19 [Simonyan and Zisserman 2014] as the activation network. The activations from the VGG-19 network are compared in a fashion termed a diversity loss, a means of introducing diversity into the image synthesis of the network. The network synthesises images based only on the segmentation map, leading to the network choosing one style for each item class of the dataset and using that for all instances of that class. To combat this, the network can be set to synthesise multiple images, nine by default, and combine the areas of the classes of the images that score lowest, creating one meta-image. This is said to create competition between the different image outputs, leading to a diverse set of images synthesised; however, this is only performed during training.

The downside of this strategy is that there are now multiple candidate output images at inference time. This results in a situation where if a single image is required, a choice needs to be made on which of the nine output images to choose, with user preference being the only possible strategy. In this case, one can simply opt to not use the diversity functionality, output only one image, and the diversity loss cancels itself out and behaves like the standard perceptual loss.

3.2.2 pix2pixHD - A conditional GAN

Wang *et al.* [2018b] proposed pix2pixHD to improve upon the results of the CRN, thus the network follows the same general structure, such as using a one-hot encoded semantic segmentation annotation as input and synthesising a realistic image that matches the anno-

tation. The network architecture is vastly different in design to that of the CRN; however, the findings from designing the CRN are put to good use in the development of pix2pixHD. Figure 3.4 demonstrates the output from pix2pixHD.

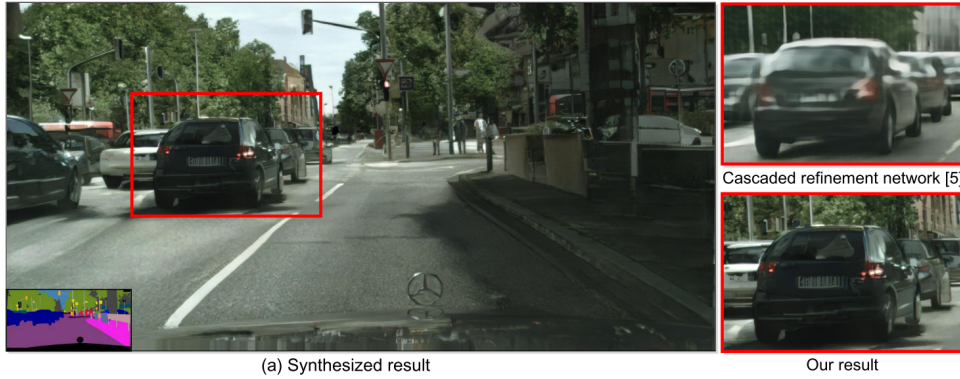


Figure 3.4: An example input and output from pix2pixHD, as well as a comparison to the output from the CRN. Taken from Wang *et al.* [2018b]

pix2pixHD uses a GAN architecture, and while it was said that GANs would be unstable in this domain, a number of measures are put in place to mitigate these issues, allowing the powerful advantages GANs have over conventional training methods to shine through. Unlike in the case of the CRN, pix2pixHD synthesises only one image and handles diversity in a different manner by means of what they call a feature encoder.

The network also takes additional information as input to further improve the output of the network. An extension of semantic segmentation is instance segmentation, where each instance of an object in the reference image is segmented separately, giving an annotation that can be used to differentiate one object from another, even if they overlap. Since there can be a varying amount of instances between one reference image and another, one-hot encoding the instance annotation would yield an inconsistent number of channels for each reference image.

Thus using the instance annotation in its raw form is not possible, and is thus used as an input for the network in the form of an edge annotation, showing the edges of different instances. This is trivial to compute, and takes the form of a sparse single-channel image regardless of the number of instances, simply containing ones where there are edges of instances. The instance annotation is also used to process the output of the feature encoder.

Generator architecture

The generator is split into two components, a global generator, and a local enhancer. The global generator is responsible for synthesising an image of size 512x1024 from the input of the same size; however, this network cannot handle resolutions higher than this. The local enhancer, using the full-size input, takes the final features of the global generator combined with some initial processing of the full resolution input, and synthesises the output in the desired 2k resolution. This allows a two-step training process that increases stability during training, where the easier-to-train global generator is trained on its own, then the local

enhancer is added to increase the resolution to the final image resolution and training is completed.

The global generator is based on the architecture proposed by [Johnson *et al.* \[2016\]](#), an encoder-decoder architecture where the input is downsampled using strided convolutions into a residual core, and multiple residual blocks process the data to extract the high-level features and convert the sparse semantic annotation input into a dense block of data, which is then upsampled using transpose convolutions and processed with a single 3x3 convolution into the intermediate colour image output. The network is composed of Convolution-InstanceNorm-ReLU sets, and is shown in figure 3.5. Differing from the CRN, pix2pixHD uses instance normalisation instead of layer normalisation, following the architecture by [Johnson *et al.* \[2016\]](#).

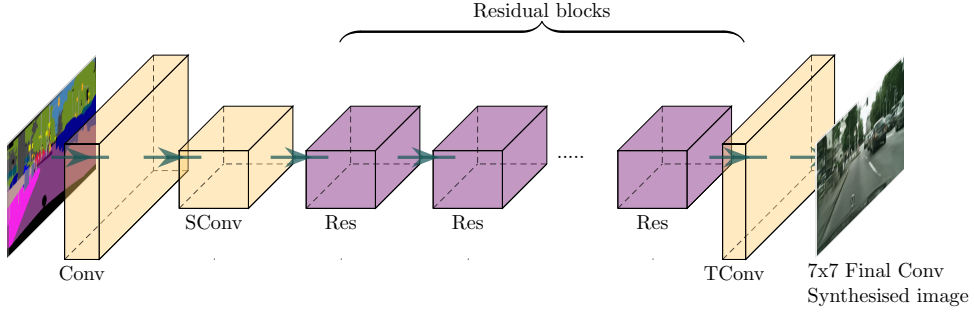


Figure 3.5: pix2pixHD global generator network architecture. *SConv* - Strided convolution; *Res* - Residual block; *TConv* - Transpose convolution. All convolution blocks, except *Final Conv*, are Conv-InstanceNorm-ReLU. Some strided convolution and transpose convolution blocks are excluded for the sake of simplicity.

The local enhancer follows a similar design, this time only downsampling the input once to match the resolution of the global generator. Then, the output from the final transpose convolution of the global generator is added to the downsampled input. The data then proceeds through a smaller set of residual blocks than that of the global generator, is upsampled once by a transpose convolution, and processed with a single 3x3 convolution into the final output. One can add multiple local enhancers to continually increase the resolution; however, the effectiveness of this was never elaborated upon.

When training the network, the global generator is initially trained on its own, using the intermediate colour image output. After a predefined set of epochs, the local enhancer is added to the overall network, and the intermediate output is no longer created, as the output of the final transpose convolution is fed into the local enhancer for further processing over the course of another set of epochs.

Discriminator architecture and additional loss metrics

The discriminator used to train pix2pixHD is based on the Patch-GAN architecture by [Isola *et al.* \[2017\]](#), where instead of the discriminator classifying the entire image as real or fake, it outputs a grid of classifications based on patches of the input image. This allows the discriminator to classify individual sections of the image as real or fake, instead of simply the entire image. Instead of calculating the loss specified in equation (2.1) using the

logarithmic terms, least-squares error is used instead [Mao *et al.* 2017]. This allows over-confident classifications to be penalised as well, sometimes preventing the classifications from locking to the extrema which would lead to entirely vanished gradients.

Two feature matching losses are used as a means of stabilising training and enhancing the input. The first makes use of the activations of the discriminator. While typically a really confident discriminator will lead to vanishing gradients, the feature matching loss will still allow the discriminator’s classifying information to propagate back to the generator, preventing training from stalling, as well as increasing synthesised image fidelity. The second feature matching loss takes the form of the same perceptual loss used by the CRN, using VGG-19 as the classification network.

Instead of using a single discriminator, it was decided that multiple discriminators with identical architecture are used, only the input for each is at a different scale. This multi-scale architecture allows the discriminators to individually focus on features at different scales, leading to more consistency across the entire image as well as fine detail in individual sections.

To improve discriminator performance, the segmentation and edge annotations are concatenated with the input image to give the discriminator more information as to what characteristics the image should have.

Feature encoder

A key aspect of the image synthesis process for pix2pixHD is the feature encoder. This is a separate network that is trained end-to-end with the generator, taking as input the reference image for the training sample, learning to encode the characteristics of the reference image into settings for the generator to use. This results in a per-pixel output that, while it might have the intended effect, cannot be used in inference given the need for a reference image, something only available during training. Instead, Wang *et al.* [2018b] chose to group the output pixels together using the instance annotation. For each instance, the pixels covered by that instance are averaged together to form a single setting for that instance. While not as fine-grained as a per-pixel setting, this opens up a strategy for not requiring the reference image during inference.

Once training has been completed, the trained feature encoder is run over all samples in the training dataset, and the feature encodings for all samples are collected. Then, for each semantic object class in the dataset, k-means clustering [MacQueen and others 1967] is used to create a set of feature encodings that are saved and can be used at inference time. Using an instance annotation, for each instance, a saved setting is drawn for the semantic class of the instance and used to populate the area that the instance covers, leading to a compatible feature encoding input that does not rely on the reference image. The feature encodings can also be used to replace parts of the image with different styles.

3.3 Video synthesis

Video synthesis is an extension of image synthesis, where a network needs to be able to synthesise a collection of images that can be grouped together into a video. There are a

number of different styles of video synthesis. One such style, future-frame-synthesis, uses a set of input frames to attempt to synthesis the frames that would continue from the input [Zhou *et al.* 2020]. Another method is that of using an image synthesis network such as pix2pixHD, which uses segmentation annotations as input, to synthesise one frame at a time, with modifications to the network architecture to accommodate the new task. This type of network is seen in vid2vid [Wang *et al.* 2018a], and is further explained in section 3.3.1.

With the extra time dimension present, a number of added difficulties present themselves. The largest of which is temporal consistency, where frames not only need to have high individual quality but be consistent between each other. A red car in a single frame may have high accuracy when compared to the reference image; however, if that car is synthesised to be blue in the following frame then the results are considered unsatisfactory. Another issue is that high quality labelled video data is not as common as image data, due to the requirement for a far higher number of images to be labelled to create sufficient data. This necessitates creative solutions such as creating synthetic data using systems such as AirSim [Shah *et al.* 2017], or using pre-trained segmentation networks to synthesise data labels for previously unlabeled video.

3.3.1 vid2vid - A conditional GAN for video synthesis

vid2vid, proposed by Wang *et al.* [2018a], converts pix2pixHD into a video synthesis network. It maintains most of the architecture and design decisions behind pix2pixHD, adding extra elements to facilitate temporally consistent video synthesis.

vid2vid synthesises one frame at a time, using prior frames as an additional input to guide the next frame. The largest addition to the network is using optical flow estimation to reuse prior frame content where possible, while synthesising content where required, leading to high levels of temporal consistency between frames.

The feature encoder still plays a primary role in guiding synthesis. A change is made to how the feature encoder is used during inference, where, instead of using k-means clustering, it was decided that fitting a Gaussian mixture model to the set of extracted encodings would allow one to sample a far greater range of encodings for use, while still using aggregate data.

Generator architecture

The network architecture is much the same as pix2pixHD; however, elements have been added and modified to adapt the network to video synthesis. The global generator has in effect been duplicated and run in parallel, with a join in the middle to share information. The extra input and output are responsible for additional video related data.

Using figure 3.6 as a guide, the generator can be divided into four sections, with an additional combination phase at the end not shown in the diagram. The top two sections are similar in purpose to the global generator for pix2pixHD, and the bottom two sections are for additional video-specific processing.

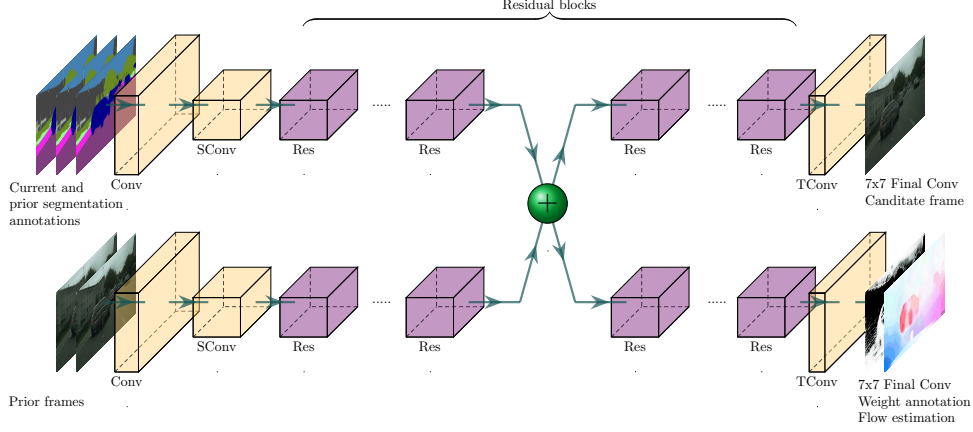


Figure 3.6: vid2vid global generator network architecture. *SConv* - Strided convolution; *Res* - Residual block; *TConv* - Transpose convolution. All convolution blocks, except *Final Conv*, are Conv-InstanceNorm-ReLU. Some strided convolution and transpose convolution blocks are excluded for the sake of simplicity.

The top-left section is simply modified to additionally accept the prior frame segmentation annotations, in addition to the already present feature encodings and edge annotation. The top-right section does not change, and simply outputs a synthesised candidate frame.

The bottom two sections have the same architecture as the top two, only with different input and output dimensions. The bottom-left section accepts and processes the prior synthesised frames concatenated together, and the bottom-right section outputs two new types of outputs: an optical flow estimation used to warp the last prior frame to conform to the segmentation annotation of the current frame, as well as a single-channel weight annotation with values between zero and one.

There are now three distinct outputs of this network: a newly synthesised candidate frame, a warped prior candidate frame, and a weight annotation. The weight annotation is used to linearly combine the candidate frames with the following equation:

$$I(S; P; w) = (1 - w)S + wP, \quad (3.1)$$

where S and P are the newly synthesised and warped prior candidate frames respectively, w is the weight annotation, and I the combined synthesised image.

A key aspect in the function of the complete generator is the join in the middle. The individual inputs are processed separately, then the activations of both sections are summed together. These new summed-together activations are used as the inputs to both remaining network sections, meaning that both sections have access to information from both inputs. These output sections are then able to specialise on synthesising their specific outputs, while using the combined information of both the semantic and prior frame inputs in the way that suits them best.

The purpose of the duplicated network architecture is to reuse as much content as possible from the prior synthesised images, allowing for higher temporal consistency. On the other

hand, the problem of occlusion is mentioned, where objects move to reveal areas of the image that have not previously been visible, meaning there is no prior content to reuse for these regions. This is where newly synthesised content should be added from the synthesised candidate image. The reuse of content is further directed using an additional foreground-background prior, created using the segmentation annotation, to designate certain semantic objects as background or foreground. The prior is used to more easily select content to be reused from prior frames since background areas are less likely to dramatically change. This is said to greatly improve temporal consistency.

The concept of a global generator and a local enhancer have been kept. The network will now have two local enhancers per resolution scale, one for the synthesised candidate image, and one for the optical flow estimation and weight annotation. The two local enhancers do not combine activations in the way that the global generators do, with each local enhancer having inputs and outputs that match the global generator half they are enhancing. While the global generator for pix2pixHD operated at 512×1024 , the vid2vid global generator operates at 256×512 , as well as having fewer stages of downsampling and upsampling compared to the pix2pixHD global generator. Thus two sets of local enhancers are needed to synthesise 1024×2048 images.

Discriminator architecture and additional loss functions

The discriminators used by vid2vid are much the same as the ones used by pix2pixHD. Two sets of discriminators are used, one set for image quality, and another for temporal consistency. The image quality discriminators operate in the same manner as the discriminators in pix2pixHD, taking in the segmentation annotations, edge annotations, and the image to test, and performing the same role.

The temporal consistency discriminators however take as input the image to be tested, a number of prior images, as well as the reference optical flow between those frames. This allows the temporal discriminator to match whether or not the synthesised images are consistent among frames, and encourages the reuse of information from prior frames, both from the prior frames as input as well as the warped prior frame. These discriminators are both multi-scale, using different image sizes, as well as multi-frame-rate, where in one examination the discriminator may skip frames to see further into the past or see all frames closer to the present. This is said to improve long term consistency.

Like pix2pixHD, a number of additional loss functions are included to assist in training. The discriminator feature matching loss that is used by pix2pixHD are retained, with the addition of feature matching losses for both the image and temporal discriminators. The VGG feature matching loss is also retained and shown to provide improved image quality.

A number of new loss functions are added as well, focusing on optical flow and its effects on the network. Reference optical flow is synthesised using FlowNet2 [Ilg *et al.* 2017]. An end-point-error loss function is added to guide the network to correctly estimate optical flow, by comparing the synthesised optical flow against reference optical flow using Euclidean distance. A warping error is also added, where the previous reference frame is warped using the newly-synthesised optical flow and compared against the current reference frame.

3.3.2 Other video networks

Since vid2vid was proposed, other networks that either improve on vid2vid or seek to solve different problems have been proposed. These networks were not focused on for our work since they diverged too far from their image synthesis counterparts; however, they are still worth mentioning.

“Few-shot vid2vid” was proposed by Wang *et al.* [2019] as a direct enhancement to vid2vid, where the content of the output video would be given by the usual semantic annotations; however, the style would be determined by a sample image given as an input during inference. The network is demonstrated to be able to take any source style image that matches the theme of the input data, and synthesise video content that the input style image could be said to be taken from.

“World-Consistent Video-to-Video Synthesis” by Mallya *et al.* [2020] is another enhancement of vid2vid, where the ultimate goal is to ensure permanent long term consistency. An example of this would be if the synthesised video is the view from a car driving once around a city block, that when the car arrives at a point it has previously travelled to, the content in the two frames is identical. This is achieved in the network by using additional depth data as input, as well as storing synthesised pixels as points in a growing point cloud that can be sampled from when data should be reused. This results in something very close to a 3D graphics renderer in quality due to the extreme content reuse leading to near-perfect temporal consistency as well as the network’s high image quality when synthesising new data.

Finally, there is “Flow-navigated warping GAN” by Dong *et al.* [2019]. This network follows a similar concept to “Few-shot vid2vid” in how the network is to be used. The network is intended to be used for virtual garment try-on, where an image of a person, the garment they wish to be seen trying on, and multiple frames of pose information are used as input. The output video is of the person wearing the garment and performing the routine described by the pose information. Optical flow is made use of in the same manner as that of the vid2vid series of networks, as well as specific architecture elements to facilitate the natural combination of the three different inputs into one consistent video.

3.4 Conclusion

In this chapter, we described the base network architectures that we will be building variations of, as well as the video networks that we are using to guide the process of adapting the image synthesis networks to perform video synthesis.

The cascaded refinement network, built using a series of refinement modules that progressively increase the output size of the network, lacks many of the enhancements that allow pix2pixHD to create higher quality images. The enhancements include feature encodings, which allow the user greater control over the composition of the network output, as well as a GAN architecture that allows the network to learn in an ever-evolving manner adapted specifically to the task at hand.

However, these enhancements were not added to specifically facilitate video synthesis, and so their usefulness when the network is tasked with temporally consistent video synthesis is not guaranteed. Finally, we describe video synthesis networks, which have specific components that are included solely for the task of video synthesis, leading them to differ quite substantially in composition and complexity from image synthesis networks.

Chapter 4

Research methodology

4.1 Introduction

This chapter describes our experimental methodology, research questions and hypotheses.

We first explain how we modified base image synthesis networks, the CRN and pix2pixHD, both in terms of porting features from one network to the other, as well as adding new functionality to both networks to improve their temporal consistency. An overview of the dataset used in this experiment is given, followed by the research questions and hypotheses, outlining the focus of our work. Finally, a detailed experimental methodology that includes dataset preparation, details about the networks we used, network training processes, and evaluation metrics is given.

4.2 Network variations

Our experiments are comprised of comparisons between the CRN and pix2pixHD, two image synthesis networks, when they are repurposed to synthesise video. The networks were tested in their original state, as well as after modifications are made to the networks in an attempt to enhance their ability to synthesise temporally consistent video.

pix2pixHD contains components that the CRN does not have an equivalent version of. Components such as the feature encoder, discriminators (with all the systems that allow for stable training), and using edge annotations for inputs are all components that are technically separate from the generator itself, and as such can be easily ported to the CRN. The inclusion of residual blocks in the network gives pix2pixHD an additional edge over the CRN. This opened up a greater opportunity for comparison between the CRN and pix2pixHD, where not only the base networks architectures and training methods can be compared, but also modified and upgraded variants of the networks.

Neither the CRN nor pix2pixHD support the use of prior frames as inputs, nor optical flow estimation as part of its image synthesis process. Variants of both networks containing support for these features are tested as well.

4.3 Research questions and hypotheses

The following research questions are the focus of this work, and appropriate hypotheses are included.

1. **Without any modifications, will the CRN or pix2pixHD synthesise more temporally consistent video, and by what margin will the difference be?**

Hypothesis:

pix2pixHD will synthesise video that is more temporally consistent than the CRN due to having a feature encoder that can be used to influence the output. This will give pix2pixHD a moderate advantage over the CRN.

2. **Can porting components from pix2pixHD to the CRN improve its ability to synthesise temporally consistent videos, and if so, to what extent?**

Hypothesis:

The CRN will benefit from the inclusion of these features; however, none of these components are specifically created to address temporal consistency, and thus their greatest effect will be in image quality. Regardless, some increase in temporal consistency is to be expected.

3. **Chen and Koltun [2017] mentioned that GAN architectures add instability to the training process, and thus chose not to use them in the training of the CRN. Is there a way to include the use of discriminators to enhance the training process without negatively affecting training stability?**

Hypothesis:

The discriminators and their supporting supplementary loss functions that pix2pixHD uses to train are specifically designed to address the stability issues that GAN architectures can introduce. The use of these discriminators in the training of the CRN should be possible, resulting in a dramatic increase in image quality.

4. **Does the inclusion of prior frames as inputs to the networks improve temporal consistency in video synthesis, and if so, by how much?**

Hypothesis:

The inclusion of prior frames will improve temporal consistency by a significant amount, due to the network having access to information of what was synthesised for the previous frames.

5. **Wang et al. [2018a] included the synthesis and use of optical flow in the vid2vid network architecture to make use of prior frames more effectively. If the CRN and pix2pixHD are modified to include features from vid2vid in order to make use of optical flow, would the temporal consistency of video synthesis increase?**

Hypothesis:

The inclusion of optical flow in the CRN and pix2pixHD may not result in the same improvements to temporal consistency they made in vid2vid, due to fine-tuning and network-specific adjustments possibly being required to successfully extract the possible benefit from optical flow. That said, some improvements may be witnessed if the features work as they are supposed to without modification.

4.4 Dataset

The dataset that we used in this work is the CityScapes dataset [Cordts *et al.* 2016], and is primarily used for training semantic segmentation networks. It contains 2975 training, 500 validation, and 1525 testing video sequences comprised of 30 frames each. The twentieth frame of each sequence is selected to be part of the primary image dataset, which contains hand-labelled segmentation and instance annotations for the training and validation sets.

In order to use all thirty frames of each video sequence, the provided reference frames must all have labels. This is achieved by using semantic segmentation and instance segmentation networks to process the data and synthesise the information to an acceptable degree of accuracy. The network implementations used for this in our work are DeepLabV3Plus [Chen *et al.* 2018] for semantic segmentation and Detectron2 [Wu *et al.* 2019] for instance segmentation.

4.5 Experiment Methodology

The methodology used to compare the CRN to pix2pixHD, and comparisons of variants of the networks, is performed as follows.

4.5.1 Dataset preparation

As described in section 4.4, the CityScapes dataset includes both a labelled image dataset and unlabelled sequence dataset. The labelled image dataset was used in the given form to train network variations that do not have a reliance on prior frames. Networks that use prior frames required the use of the sequence dataset. The sequence dataset has exactly 30 times as many images and was found to present difficulties in label synthesis, storage, and transfer between testing and training computers. Thus it was decided that a subset of the sequence dataset should be used. The dataset is separated into folders containing the respective data from the city that the data was captured in. A total of 808 out of 2975 sequences were kept by selecting a subset of these cities.

4.5.2 Network recreation, architectures, and deviations

While source code for both the CRN and pix2pixHD are available, they are only used as a reference to obtain aspects of the networks that are not referenced explicitly in their respective papers. All aspects of both networks, as well as the entire supporting training and evaluation data creation framework, were written from scratch in PyTorch [Paszke *et al.* 2019]. This allows reliably running both the CRN and pix2pixHD with and without

specific components dependent on the network configuration chosen, by sharing the exact components between the two networks. For example, adding a discriminator or feature encoder to the CRN involves simply using the exact pix2pixHD component with no changes required. The source code for our networks can be found [here](#). Table 5.1 contains a full list of configurations used allowing easy comparison of network architecture configurations.

Image synthesis output size

Both networks are intended to be used to synthesise images at size 1024x2048. While in the base configurations both networks can be trained to specification at this resolution, the inclusion of additional components such as support for optical flow estimation led this output size to be unreachable due to video memory requirements and, when viewed from the perspective of optimising for temporal consistency, unnecessary.

The CRN uses two refinement module additions during training to increase the output size of the synthesised image from 256x512 to 1024x2048. Section 3.2.1 specifies that the final refinement module only uses 32 convolutional filters in its convolution operations. This indicates that this module is responsible for upsampling the prior output rather than contributing new features. If this refinement module is left out, only a small amount of image content and quality is lost; however, the video memory savings are considerable.

pix2pixHD is comprised of a global generator and a local enhancer. The global generator synthesises at size 512x1024 and the local enhancer at size 1024x2048. Unlike the CRN, inputs of any size are able to be processed by the network; however, it was found that lowering the input size for the network resulted in a complete loss of quality in the resultant image due to the large kernel size (7x7) used within the processing bottleneck of the network, where the resolution of the data is already quite small. Thus, the only way to output at the desired 512x1024 image size is to simply remove the local enhancer portion of the network. While there is the possibility for a decrease in overall image quality from removing the extra processing that the local enhancer performs, the difficulty of synthesising high-quality images only increases with image size, thus by focusing only on the output of the global generator we have effectively reduced the difficulty of the problem, thus in our opinion offsetting the lost processing power of the local enhancer. This removal reduces the memory requirements of the network significantly.

Component sharing

Some minor simplifications were made in an effort to keep consistency. For example, both the CRN and pix2pixHD use a VGG perceptual loss. While the CRN uses this as its primary and only loss in the base configuration, pix2pixHD includes the VGG loss for additional quality in a supporting role. Thus, while the weightings for the individual layers in the feature matching process differ slightly between the two implementations, it was decided that the configuration for the CRN would be used for both networks.

Component modification

Some elements of the networks were modified when clear benefit could be seen or to alleviate certain noticeable issues. The first of which is that of using weight normalisation instead of

layer normalisation in the refinement modules of the CRN, to alleviate water-drop artefacts (this is further described in section 5.3.1).

Another modification was that of replacing VGG-19 with MobileNet-V2 [Sandler *et al.* 2018], since both are image classification networks; however, MobileNet-V2 is more accurate and is designed to run on mobile devices. Thus, the possibility for better feature extraction and lower memory usage was attractive, and as such MobileNet-V2 was incorporated as an alternative to using VGG-19 for the perceptual loss. This resulted in improvements to memory usage, training duration, and output image quality.

Prior frame usage

When a network is configured to use prior frame information (synthesised images and prior segmentation annotations), the information for the current frame is added to a list that is used as input for the synthesis of the next frame. A total of two prior frames were used, following the decision by vid2vid to do the same.

Since the number of input channels in a network is inflexible once the network has been instantiated, placeholder prior frames need to be used. In this case, the information is in the form of zero-filled tensors that are updated with actual prior frames once they become available. Without adjustment, using zeros for the prior image frames would be akin to telling the network that the previous frames were pure black, which is an extremum, given that image data is in the range of $[0.0, 1.0]$. The network will then need to learn to handle this extremum, while still making good use of images with a range of data above zero. To alleviate this large disparity, it was decided that the prior frames be shifted down to $[-0.5, 0.5]$. This results in the placeholder zeros being closer to the mean of the data, without any detriment to the , thus the network needs to do less to adapt to the use of both real data and zeros.

When a network configuration with a discriminator uses prior frames as input but does not use optical flow, the inputs to the discriminator were altered to include the prior frame information as well as the current frame information, allowing the discriminator to access both image quality and temporal consistency. When using optical flow; however, an additional discriminator is added specifically for temporal consistency and as such the image discriminator was not adjusted in this manner.

Adding residual blocks to the CRN

In an effort to enhance the ability of the CRN to accommodate using components such as discriminators and the demands of optical flow, while at the same time maintaining the memory requirements of the network, a configuration option for the CRN was added to replace the Convolution-Normalisation-LReLU blocks within the refinement modules with residual-based blocks taken from ResNext [Xie *et al.* 2017]. This had the effect of reducing the parameters in the network from ± 90 million to ± 60 million, while increasing the number of convolution operations from two to seven, with the addition of skip-connections to propagate information unobstructed through the network. Further architectural details on this change can be found in appendix A.1.

Optical flow usage

The inclusion of optical flow estimation requires dramatic adjustment to the network architecture. The concept of twin networks was brought over from vid2vid, where two separate inputs and outputs are used, prior frames are the second input, and the optical flow estimation and weight annotation the second outputs.

The twin networks architecture for pix2pixHD closely follows the architecture of the vid2vid global generator, where the network is duplicated and joined in the middle, leading to the second halves each having access to all the information from the first halves while specialising on their respective outputs. This also leads to double the number of parameters in the network, increasing both its memory and processing power at the expense of computation time and video memory usage.

Adding a twin network architecture to the CRN is more involved. Since there are multiple refinement modules, and each take in the inputs, it was decided that for each refinement module there would be a corresponding *twin* refinement module. Like with pix2pixHD, the original modules of the network take as input the semantic information, while the new twin modules take as input the prior images. Both however take in the summed outputs of the two prior modules, creating a network that looks like a chain, as well as opening up multiple paths for data. The final refinement modules serve the same purpose as the final halves of vid2vid, where one synthesises the candidate image, and the other synthesised the optical flow estimation and weight annotation.

The usage of optical flow estimation in these network configurations closely followed the method used by vid2vid, where FlowNet2 is used to synthesise reference optical flow, the optical flow estimation is used to warp the prior synthesised image, and the weight annotation is used to merge the synthesised candidate image with the warped prior image. The networks rely on the same discriminators, where the image discriminator evaluates the synthesised image against the segmentation annotation, and the temporal discriminator evaluates the synthesised image against the current and prior optical flow data, as well as the prior synthesised images. The optical flow portion of the network was trained using both end-point-error and the warping loss. One aspect that is not included is the foreground-background prior, as this would have changed too much of the network.

4.5.3 Training methodology

While there are some similarities in the training process of the CRN and pix2pixHD, there are however differences in procedure and training length. One consistent aspect for both networks however is that both the CRN and pix2pixHD are trained on a minibatch size of 1, presumably due to memory limitations of increasing the minibatch size.

CRN

To get to synthesising images of size 512x1024, the CRN requires two sets of training. Using the single image CityScapes dataset involves training for 200 epochs at 256x512, adding a refinement module, then training for another 20 epochs according to the official source code for the CRN. As this is meant to be followed by the addition of another refinement module

and a further 6 epochs of training, we decided that as we are not performing the final set of training we would increase the number of epochs in the second set of training from 20 to 50, both for additional fine-tuning and to allow more training time for the more complicated network configurations. The learning rate for the CRN was set to 0.0001, according to the official instructions.

In certain network configurations that include components from pix2pixHD, an additional aspect that is ported is the learning rate-scheduling that pix2pixHD performs. In these runs, the learning rate was fixed for the first half of each set, then decreased to zero over the course of the second half.

pix2pixHD

pix2pixHD would normally be trained in two sets, first the global generator, and second the local enhancer. Since we are not using the local enhancer in this experiment there is only one set of training for 200 epochs. The learning rate was set to 0.0002 according to the official instructions, which decays linearly to zero over the course of the last 100 epochs.

Considerations when training network variations

When training using prior frames a sequence length of 6 was used per sequence in the dataset, randomly sampled from the total 30 frames per sequence. When considering 6 images synthesised per training sequence, with 808 sequences, in order to approximately maintain the number of images processed when training a network using prior frames it was decided to reduce the number of epochs to 60%. Thus the CRN with prior frames was trained for 120 epochs, then 30 after adding the additional refinement module, and pix2pixHD with prior frames was trained for 120 epochs.

Training hardware

These networks were trained using the [CSIR CHPC Lengau cluster](#). The training sessions are submitted with the following hardware configuration request:

- *Nodes*: 1
- *CPU*: Intel® Xeon® 2.6 GHz (4 cores requested)
- *Memory*: Maximum used 79.60GB
- *GPU*: NVIDIA® Tesla® V100 16GB

4.5.4 Evaluation metrics

The evaluation of image and video synthesis can be challenging. Since a network may produce many different outputs that are all acceptable given a specific input, direct comparisons between synthesised and reference images are usually insufficient in measuring the abilities of a network.

[Chen and Koltun \[2017\]](#) makes extensive use of human preference scores to test the CRN, where humans are given pairs of images and are required to rate their preference of the

two. This is often a series of time-limited tests ranging from people only getting brief glimpses of images, to getting up to eight seconds. Amazon Mechanical Turk is used to scale the experiments to include a larger amount of people than would be possible to organise otherwise; however, this service is not free.

Wang *et al.* [2018b] conduct multiple tests using different metrics. One method is to use a semantic segmentation network to segment the synthesised images, using the premise that a segmentation network can accommodate for variations in colour and orientation of objects in images. The resulting annotations are then compared to the hand-crafted annotations provided with the dataset, using both pixel accuracy and mean intersection-over-union. Human preference scoring is also performed in the same manner as that of the testing used for the CRN.

Wang *et al.* [2018a], as well as making use of human preference scoring, make use of a modified form of Fréchet Inception Distance (FID) [Heusel *et al.* 2017]. FID uses a pre-trained image recognition CNN, in this case, the Inception-v3 network [Szegedy *et al.* 2016], to extract and compare features of image pairs in a manner that aligns more closely with how a human would compare the images. FID was developed as a means of comparing images, and thus modifications were made in order to support comparing videos. The Inception-v3 network was thus replaced with video recognition networks, allowing videos to be compared. Synthesised videos are compared directly to the corresponding reference videos, thus giving an indication as to how different the two sets of videos are from each other.

Fréchet Video Distance (FVD)

In an effort to create a formal and standardised method of evaluating video synthesis networks, Unterthiner *et al.* [2018] proposed the Fréchet Video Distance (FVD). This works in a similar manner as the modified FID used by Wang *et al.* [2018a], and through extensive testing, it is shown to align closely with human preference testing, where a network that scores worse in FVD will reliably be scored worse by humans. A video compared to itself will have an FVD of zero, as FVD is a distance from perfect reproduction. Thus, FVD was used in this work as the standard evaluation metric for comparing video samples as it analyses temporal consistency against a reference video.

Unterthiner *et al.* [2018] included a [reference implementation](#) of FVD with their paper. This reference implementation, while working, is written using an old version of the TensorFlow [Abadi *et al.* 2015] deep learning library (TensorFlow 1) and is not as user friendly as we would have liked it to be. Thus, this was ported to TensorFlow 2 to allow for easier integration with our workflow, and at the same time benefiting from all the improvements in the new version of the library.

FVD requires videos that are a square of size 244x244. The method automatically down-samples given inputs to the correct size; however, the videos produced by networks are rectangular, and thus the source videos should not be used directly. Instead, three square videos are sampled from the given input images: a far-left square, a central square, and a far-right square. These three squares are tested individually, and then the mean of the three is calculated.

FVD processes 16 frame videos, in batches of 16. To get a comprehensive result, it was decided that 8 batches should be used, representing a total of 128 videos. The mean of the 8 batches are then calculated and represent the final score of the network. Videos are all sampled from the same point in the 30 frame sequences, and networks that use prior frames synthesise two extra frames starting two frames early, to allow the network to fill their lists of prior frame information with data instead of the zeros they are initialised with.

Qualitative analysis

While FVD provides a convenient single value as a score for the network, it does not surface any of the intricacies of the individual networks, nor does it explain any of the details that it used to influence its decision. Thus, the videos were also analysed by hand.

4.6 Conclusion

In this chapter, we described the processes and details that were needed in order to carry out our experiments. The primary focus of our work is on temporal consistency, which does not directly align with image quality, and so network alterations that improve image synthesis quality may not improve results, or worse, offer more area for deviations between frames and thus harm temporal consistency.

We thus planned our experiments to involve a large amount of network variations in order to determine how and to what extent modifying the networks affect their abilities. These variations include porting components from pix2pixHD to the CRN, modifications to already existing components to increase their effectiveness, the inclusion of prior frames as inputs to give the networks a context of how to synthesise the following frame, modifying the CRN to use residual network concepts to improve its ability to adapt to the extra components that are added to it, and including the use of optical flow estimation in the synthesis process by using concepts from vid2vid to further adapt the networks.

The training methodology of the base networks was adapted to be consistent across network variations as best as possible, since adapting training methodology to each network variation was out of the scope of this work. In terms of evaluation metrics, Fréchet Video Distance is used to measure the temporal consistency of the networks and their variations, while qualitative analysis is performed by hand to expand on the difference between the network variations.

Chapter 5

Results

5.1 Introduction

In this chapter we discuss the results of our experiments, pivoting around a central table with all network configurations, FVD scores, synthesised video links, and a unique index for each configuration that is used to refer back to the table within the results discussions. It is highly recommended to watch these videos while progressing through the results.

The chapter is divided into two main sections, quantitative analysis, focusing on using FVD scores to answer our research questions (section 4.3), and qualitative analysis, focusing on additional analysis of the results in ways that were not covered in quantitative analysis, such as the effects of normalisation, the effects of the feature encoder, and issues with the perceptual loss.

5.2 Quantitative analysis

Table 5.1 shows the FVD score for each network configuration, as well as an index that can be used to refer to the individual configuration, and a link to the testing video hosted on YouTube. This testing video has been processed by YouTube to 480p (480x854), a standard-definition resolution used on the site, and as such minor changes may occur. Corresponding reference video is available [here](#). Sections addressing research questions as listed in section 4.3 are labeled appropriately.

Training time

Figure 5.1 shows the training time, in days, for the various network configurations using the hardware described in section 4.5.3. The total time to train all network configurations for the CRN is 36.657 days, and the total time for pix2pixHD is 19.120 days, leading to a total time of 55.777 days. This inhibits our ability to train each network configuration multiple times due to this extreme training time requirement even with us being able to train two network configurations at once effectively cutting the time required in half. Another aspect to factor

Table 5.1: FVD scores for all network configurations.

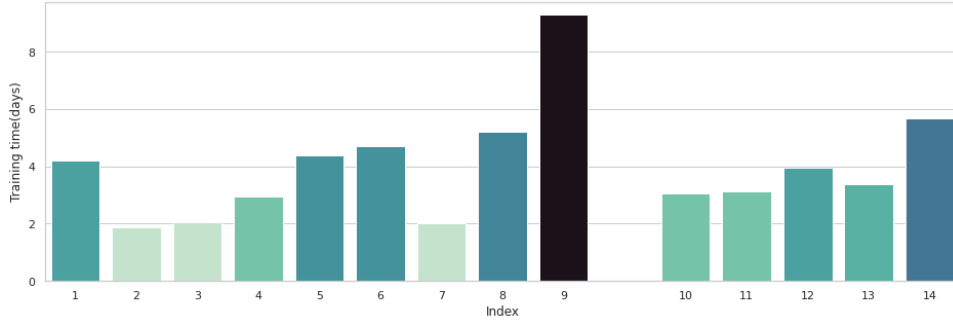
Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
1	1885.094	CRN	9 ¹	Layer	VGG	-	-	-	-	-	-	-	Link
2	1609.033	CRN	1	Weight	VGG	-	-	-	-	-	-	-	Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
4	1386.111	CRN	1	Weight	MNet	Edge	F.E.	-	-	-	-	-	Link
5	1344.705	CRN	1	Weight	MNet	Edge	F.E.	Res.	-	-	-	-	Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
7	1110.927	CRN	1	Weight	MNet	-	-	-	-	Adj.	-	-	Link
8	946.836	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	Adj.	-	-	Link
9	1007.056	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	Adj.	Twin	Flow	Link
10	1388.469	pix2pixHD	1	Inst.	-	Edge	F.E.	Res. ²	Disc.	-	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link
12	1032.801	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Ext.	-	-	Link
13	956.091	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	-	-	Link
14	909.387	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	Twin	Flow	Link

FVD-Fréchet video distance; **Net.**-Network; **O.I.**-Number of output images; **Norm**-Normalisation; **Percept**-Perceptual loss; **Edge**-Edge annotations; **F.E.**-Feature encoder; **Res.**-Residual blocks; **Disc.**-Discriminators; **Prior.**-Prior frames as input; **Twin**-Twin networks mode; **Flow**-Optical flow; **Layer**-Layer normalisation; **Weight**-Weight normalisation; **Inst.**-Instance normalisation; **VGG**-VGG-19 backend; **MNet**-MobileNetV2 backend; **Adj.**-Centred placeholder prior frames; **Ext.**-Extrema placeholder prior frames

¹ Due to VRAM constraints, when training at 512x1024 (epochs 201-250) the number of output images was reduced to 3.

² pix2pixHD uses residual blocks by default.

in that is not represented in figure 5.1 is the maximum running time of twelve hours per job on the cluster, leading to a requirement to restart jobs continually, opening up opportunities for lost hours between jobs depending on the hour the job finishes.

**Figure 5.1:** Training time in days for each network configuration.

While each configuration change affects the network training time, a key insight is the difference between *image network with prior frames*, and *video network*. The addition of the twin network configuration and optical flow, in network configurations 9 and 14, dramatically increase the training time required. While the details of these configurations are covered in section 5.2.4, table 5.1 shows that these configurations do not dramatically affect the FVD scores, while increasing training time by a large amount.

Feature encoder network configurations

Network configurations that include a feature encoder introduce a level of uncertainty into the scoring of the network. The process of sampling a set of saved feature encodings (section 3.2.2) leads to a large degree of variance in image quality, where some configurations are dramatically better than others. Given 10 possibilities for each object class and 20

classes, choosing feature encodings randomly will select from 10^{20} possible configurations. If we assume that each class' settings are mutually exclusive in effect from the settings for other classes, the number of configurations to search decreases to 200; however, this cannot easily be assumed. Regardless of the number of possible configurations, searching is still prohibitive as searching has a time requirement of nearly one hour from feature encoding selection to FVD score.

Thus a far quicker method was chosen to select the feature encodings for a network: randomly sample 50 sets of feature encodings and use them to synthesise images. The set of feature encodings that produce images that are closest to the reference image and have a good image composition are chosen. This results in a dramatically shorter selection process while giving each network a good chance to show its merits. This relies on the assumption that the set of feature encodings that give acceptable results is large enough to have an element appear in the 50 random samples; however, we found no issues using this method.

5.2.1 Comparison of the CRN and pix2pixHD base configurations [Research question 1]

Table 5.2: FVD scores for base network comparison configurations.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
1	1885.094	CRN	9 ¹	Layer	VGG	-	-	-	-	-	-	-	Link
2	1609.033	CRN	1	Weight	VGG	-	-	-	-	-	-	-	Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link

Refer to table 5.1 for table notes.

Table 5.2 is available for easier comparisons of network configurations details and FVD scores. For the CRN, we tested three different configurations that we consider base configurations, one following the original design (index 1), one using weight normalisation (index 2), and one using weight normalisation and MobileNet-V2 for the perceptual loss network (index 3). By making these deviations from the original we hoped to improve on the results with simple component substitutions. We found that layer normalisation introduced artefacts into the synthesised images, thus we replaced it with weight normalisation, and we saw an opportunity to improve the power of the perceptual loss for free by replacing the VGG-19 network with MobileNet-V2, a smaller, more efficient, and higher accuracy network.

Replacing layer normalisation had the result of improving the FVD score of the CRN by 246.061, most probably due to the large reduction in the flickering dark areas and increase in clarity. Replacing both layer normalisation and VGG-19 improves the score by 366.850 compared to the unmodified configuration, likely due to an increase in colour and sharpness as well as further reduction in flickering dark areas. These significant improvements show that the CRN has room for easy improvement. Following the success of replacing VGG-19 with MobileNet-V2, we decided that all other network configurations will use MobileNet-V2 for their perceptual loss backends. Figure 5.2 shows two output frames for each of the three CRN base configurations, with noticeable artefacts and dark areas in the first configuration.

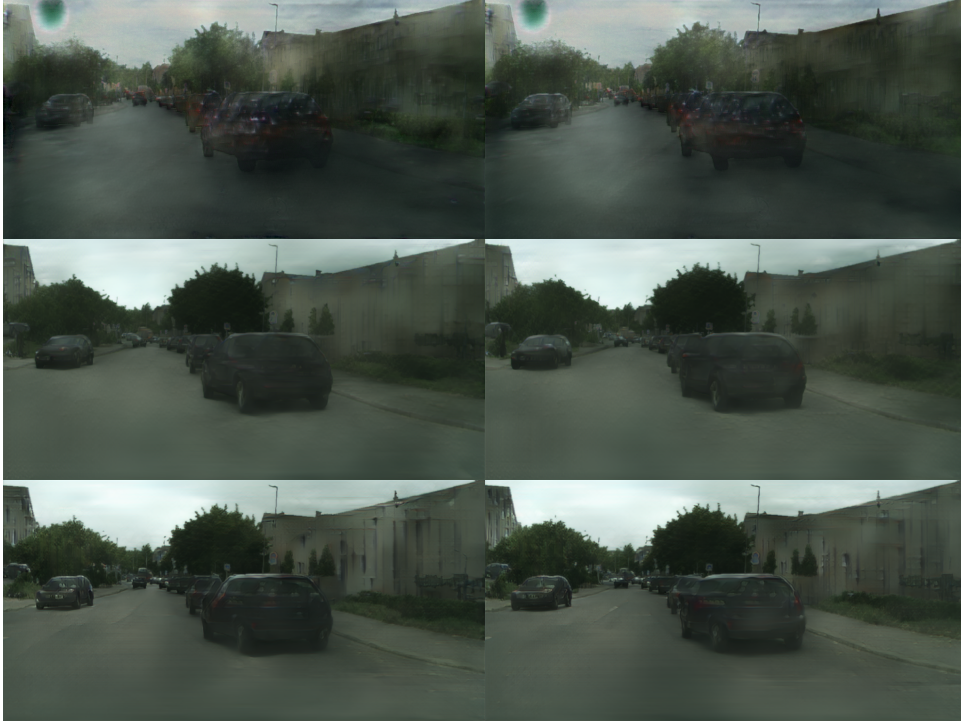


Figure 5.2: A two-frame comparison of each of the CRN base configurations, indices 1, 2, and 3.

The base configuration for pix2pixHD (index 11) has an FVD score of 1304.637, a score 183.607 lower than the updated CRN base configuration (index 3), and 580.457 lower than the original CRN configuration (index 1). Figure 5.3 shows two output frames from the network, with the most noticeable changes between the frames being the details on the buildings.

While the difference between index 3 and index 11 implies a noticeable difference in temporal consistency, it implies that even with all the extra components that pix2pixHD has that the CRN does not, pix2pixHD does not vastly outrank the CRN in terms of temporal consistency when minor adjustments are made to the CRN.



Figure 5.3: A two-frame comparison of the pix2pixHD base configuration, index 11.

5.2.2 Porting components from pix2pixHD to the CRN [Research questions 2 & 3]

Table 5.3 is available for easier comparisons of network configurations details and FVD scores. Section 4.5.2 describes the reasoning behind porting components from pix2pixHD

Table 5.3: FVD scores for porting components network comparison configurations.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
4	1386.111	CRN	1	Weight	MNet	Edge	F.E.	-	-	-	-	-	Link
5	1344.705	CRN	1	Weight	MNet	Edge	F.E.	Res.	-	-	-	-	Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link

Refer to table 5.1 for table notes.

to the CRN, as well as detailing the conversion of the CRN to use residual blocks. These changes can be seen as index 4, the addition of the edge annotations and feature encodings as inputs to the CRN, index 5, where the residual conversion took place, and index 6, where discriminators were added along with their supporting features such as the discriminator feature matching loss.

The addition of the edge annotations and feature encodings (index 4) sees the largest change in FVD of 102.133 compared to index 3. This could be attributed to the ability to differentiate between overlapping objects by using the edge annotations, as well as more consistency due to the fact that when feature encodings are used, the same encodings are used for all frames leading to repeated colours and styles across frames. It should be noted that the network, without feature encodings, tends to learn modes for the different classes in the images leading to little variation, and so the effect of the feature encodings is minor.

The conversion of the CRN to use residual blocks (index 5) improves the FVD score by a further 41.406. This, according to [Untertiner *et al.* \[2018\]](#) is a small enough change that is not usually human-perceptible. On inspection, you can see this is the case, with the two sample videos being similar consistency wise, with a change in colour being the main difference, which can be attributed to different feature encodings being used.

Finally, adding the discriminator and supporting discriminator feature matching loss (index 6) improved FVD by 26.871. This addition increased the complexity of the objects in the frames, making more pleasant still images, however with increasing complexity comes more opportunity for change between frames, which may be the reason for this smaller-than-expected improvement. A comparison of the image quality of the different CRN network configurations can be seen in figure 5.4, and a demonstration of the increased complexity in index 6 causing more differences between frames can be seen in the top two frames of figure 5.5.

5.2.3 Including prior frame information in synthesis and training processes of the networks [Research question 4]

Table 5.4 is available for easier comparisons of network configurations details and FVD scores. The addition of prior frame information to the synthesis and training processes of the networks yielded the greatest improvements to the FVD scores for both networks. Comparing the CRN base configuration index 3 with index 7, the difference between the two being the addition of prior frame inputs, an FVD improvement of 377.317 can be seen.



Figure 5.4: A single output image from the CRN at indices 3, 4, 5, and 6. Notice how adding the components has a minor effect on image quality and content, until index 6 (bottom right), where the addition of the discriminator dramatically increases image complexity.

Table 5.4: FVD scores for prior frame network comparison configurations.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
7	1110.927	CRN	1	Weight	MNet	-	-	-	-	Adj.	-	-	Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
8	946.836	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	Adj.	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link
13	956.091	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	-	-	Link

Refer to table 5.1 for table notes.

Similarly, comparing index 6, where all pix2pixHD components were ported to the CRN, to index 8, an improvement of 370.993 FVD score can be seen.

While the temporal consistency of the CRN base configuration improved, it is interesting to note that no temporal consistency training loss was included, implying that the increased temporal consistency is a byproduct of normal learning. It may be the case that the network learned that by using prior information, it could more reliably use the prior information instead of synthesising completely new information every frame, only adjusting where necessary. This outcome is similar to that of future-frame prediction [Zhou *et al.* 2020], where content reuse is encouraged, however it is important to note here that the CRN is learning to perform this reuse without suggestion. A byproduct of this content reuse is that the network could use more resources to create better quality objects in the images. Cars and windows are more defined and consistent, and in general, there is a slight increase in the fidelity of most object classes.

Comparing index 6 to index 8 (see figure 5.5), content is once again reused and enhanced, leading to better fidelity of objects in the scene. However a difference this time is that the discriminators for index 8 are adjusted to consider prior frame information as well as the current information when making a decision between real and synthesised frames. Thus, there is a direct incentive to increase temporal consistency as it is now something that it is

being measured on. Section 5.2.2 describes how the addition of the discriminators, and thus additional detail in the synthesised images may lead to more differences between one frame and another; however, with the ability to use prior frame information this additional detail generally stays consistent between frames.

pix2pixHD saw similar improvements to FVD scores when adding prior frame information. Comparing index 11 to index 13, an improvement of 348.546 FVD score is seen. pix2pixHD reacted similarly to how the CRN outputs changed with prior frame information; however, an additional effect can be observed, a general softening of features. This may both not have an effect on the FVD score as it may simultaneously improve temporal consistency by having less possible detail changes; however, it would compare worse with the reference video as it is further from the true content the frame should possess.



Figure 5.5: A two-frame comparison of the outputs of the CRN, without (top, index 6) and with (bottom, index 8) the use of prior frame information. Noticeable differences between the two pairs include the road lines and the face of the bus, which can be clearly seen in the difference images on the right. The images have been brightened to better show off the changes in temporal consistency

5.2.4 Modifications of both networks to incorporate optical flow [Research question 5]

Table 5.5: FVD scores for optical flow network comparison configurations.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
8	946.836	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	Adj.	-	-	Link
9	1007.056	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	Adj.	Twin	Flow	Link
13	956.091	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	-	-	Link
14	909.387	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	Twin	Flow	Link

Refer to table 5.1 for table notes.

Table 5.5 is available for easier comparisons of network configurations details and FVD scores. Indices 9 and 14 are configurations where both networks were converted to include optical flow in the synthesis process. To recap, this process involves the network synthesising an optical flow estimation between the prior frame and the next, a candidate synthetic image, as well as a weight annotation. The estimation is used to warp the prior frame to hopefully match the structure of the next, and the weight annotation is used to combine the candidate and warped images together.

CRN configuration index 9 achieves a 1007.056 FVD score, 60.22 higher than index 8, which had a score of 946.836. This regression can be attributed directly to the network being unable to learn to correctly estimate optical flow, as well as coming to rely on its use. The reuse of pixels from previous frames introduces the possibility of a great increase in both detail preservation between frames, as well as temporal consistency; however, this requires that all pieces of the system work together. If the optical flow estimation made by the network is incorrect, pixels from previous frames are misplaced, and over the course of a few frames, they can drift far away from their correct positions. In the case of the CRN, it learned to universally designate buildings as reusable with little correction from the synthesised candidate image, and thus issues in the relocation of pixels were propagated continuously. This led to dramatic degradation of detail preservation over the course of the 16 frames that form a testing video. This can be seen clearly in figure 5.6. Regardless, the network achieves a good FVD score compared to many other configurations, due to the fact that even though the detail preservation suffers greatly over time, the change in content is smooth, and some object classes and most object outlines have more reliance on the candidate image, offering simultaneous reuse of prior content and corrections from the synthesised candidate image.



Figure 5.6: The tenth frame of the synthesised video from the CRN optical flow configuration (index 9), as well as the corresponding weight annotation. Lighter areas denote candidate image use, whereas dark areas denote warped image use. Note the distortion in the building sections and the repeated edge of the car.

pix2pixHD configuration index 14 achieves 909.387 FVD score, the best score of any configuration, and 46.704 lower than index 13. This configuration behaves differently from how the CRN made use of optical flow, in that instead of explicitly reusing content via the warped previous frame, it simply uses what it learned by having to synthesise optical flow to internally reuse the prior frame content in the candidate image. The weight annotation then specifies that the majority of the content in the final synthesised image should come from the candidate image, which now reliably matches the previous frame. This shows that the network has a deeper understanding of the task and a better capacity to perform the prior frame warping inside the network, instead of explicitly using the synthesised optical flow estimation to warp the prior frame. Despite this, there is still deterioration over time; however, it is not as severe as the deterioration observed in the CRN. An example frame and weight annotation can be seen in figure 5.7.

5.2.5 Networks with equal opportunity

Table 5.6 is available for easier comparisons of network configurations details and FVD scores. After porting all components from pix2pixHD to the CRN, we have the comparison



Figure 5.7: The tenth frame of the synthesised video from the pix2pixHD optical flow configuration (index 14), as well as the corresponding weight annotation. Lighter areas denote candidate image use, whereas dark areas denote warped image use. Compared to the CRN, far less of the warped image is used as the warping is effectively done internally in the network.

Table 5.6: FVD scores for equal opportunity network comparison configurations.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link

Refer to table 5.1 for table notes.

of index 6 and index 11, with pix2pixHD only 13.192 FVD score ahead of the CRN. This falls within the 50 point range where Unterthiner *et al.* [2018] classified one network’s level of temporal consistency as indistinguishable from another network’s to the human eye. The outputs of both networks are similar, with details added by the discriminator training visible and changing between frames. Both networks appear to have, when given the same opportunity, learned a similar set of rules about how to synthesise images, even with their vastly different generator architectures.

Interestingly, when prior frame information is included in the image synthesis and training processes, both networks score roughly the same as well. Comparing index 8 to index 13, we can see the CRN even comes out on top, scoring 946.836, 9.255 FVD score ahead of pix2pixHD’s score of 956.091. This is once again imperceptible; however, when observing the videos, the CRN has taken the extra details provided by the discriminators and kept them under control, allowing fluid motion between frames, whereas pix2pixHD has applied a general smoothing and removal of sharp details instead. Both of these strategies work to increase temporal consistency, just in different manners.



Figure 5.8: A comparison of the outputs of the CRN (left, index 6) with pix2pixHD (right, index 11), showing how the CRN can match pix2pixHD in detail when both networks share the same components.

5.3 Qualitative and general analysis

This section contains analysis that does not focus on comparing FVD scores; however, similar results tables are included for each concept as an easy way of referencing the network configurations that are addressed.

5.3.1 Effects of normalisation on synthesis output

Table 5.7: FVD scores for network configurations relating to normalisation effects.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
1	1885.094	CRN	9 ¹	Layer	VGG	-	-	-	-	-	-	-	Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
12	1032.801	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Ext.	-	-	Link
13	956.091	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	Adj.	-	-	Link

Refer to table 5.1 for table notes.

Table 5.7 is available for easier comparisons of network configurations details. Section 4.5.2 mentions the use of weight normalisation instead of layer normalisation, as a means of alleviating artefacts present when the CRN synthesises images. Figure 5.9a shows one such manifestation of this artefact, which is present in an evolving form from the beginning of training. Initially, the artefact starts out as a swirl of colours spanning the entire image, and over the course of the first epoch of training gathers itself into a single spot. This spot continues to become smaller as training continues, however with diminishing returns, eventually not changing in any perceptible manner by the end of training. Once normalisation was found to be the issue, a number of other normalisation methods were tested. We found that any normalisation methods related to batch normalisation, such as instance, layer, and group normalisation were shown to present this issue. We tested not using normalisation at all and found that the network would not train. Weight normalisation was found to remove this problem; however, and we believe that it is due to not being a variant of batch normalisation. Weight normalisation was also found to improve image quality in general due to the network not struggling to manage the issues introduced by layer normalisation.

Figure 5.9b shows another instance of normalisation-caused artefacts. While most pix2pixHD configurations do not present these issues, these seem to appear only when prior layer information is used without optical flow (indices 12 & 13). This may be a result of a single discriminator being used for both image quality and temporal consistency, as described in section 4.5.2, and when paired with this discriminator pix2pixHD is unable to act on the feedback given by the discriminator. Karras *et al.* [2020] came to this same conclusion when analysing StyleGAN [Karras *et al.* 2019], and determined that the cause of their synthesis artefacts were indeed caused by instance normalisation, where the network could not learn correctly and these artefacts were manifested as a result of this.

Figure 5.9c shows an artefact that is present in index 6, where a discriminator is added to the CRN without prior frame usage. This is the only instance of artefacts when using weight normalisation, and it is suspected to be related to the reasons mentioned above. Karras *et al.* [2020] proposed a method called *weight demodulation* and adjustments to their network training method, which completely removed their artefacts; however, the testing of this

method was deemed out of scope for this work as this issue only appeared in one configuration when using weight normalisation, and the implementation of weight demodulation is not trivial.



(a) CRN synthesis artefact present in every frame caused by layer normalisation, changes colour depending on frame content, but does not move (index 1).



(b) pix2pixHD synthesis artefact present only in prior frame configurations, moving across the top of the frame over subsequent frames (indices 12 & 13).



(c) CRN synthesis artefact present in only one configuration with weight normalisation and not visible on every frame, though always in the same location with the same shape (index 6).

Figure 5.9: Different synthesis artefacts encountered in certain network on the same video frame. configurations

5.3.2 The effect of the feature encoder

Table 5.8: FVD scores for network configurations relating to feature encoder effects.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
4	1386.111	CRN	1	Weight	MNet	Edge	F.E.	-	-	-	-	-	Link

Refer to table 5.1 for table notes.

Table 5.8 is available for easier comparisons of network configurations details. Wang *et al.* [2018b] added the feature encoder, a more direct means of specifying the characteristics of the synthesised images, as a means of allowing the user to hand-modify the styled of objects in the scene. While this certainly had the intended effect, we have observed that this component is far more important than was originally thought.

As discussed in section 3.2.1, the CRN is proposed to have nine output images in an attempt to introduce diversity into the image synthesis process. Our results show that while there are some minor differences between images produced, each class still only has one style. Index 3, with only a single output image, has a single style for each class and applies varying amounts of darkness to the objects in what appears to be an attempt to add shadow. When the feature encoder is added in index 4, this increases the range of styles that can be applied to the different classes of objects, as well as different instances of the class in the same scene. An example of these differences can be seen in figure 5.10. This is due to an expansion in the latent space of the network, where instead of the latent space being the one-hot encoded segmentation annotation, which only offers information about class and position, the addition of the feature encoder inputs allows the network to explicitly pair different feature encodings to different styles, and perhaps even other characteristics such as the orientation of the object. This allows a more directed learning process that shifts the problem of image synthesis closer to a one-to-one problem, from its previous position as a one-to-many problem, as more information in the input to the network allows for less variance in the output.



Figure 5.10: Example of a difference the feature encoder can make to the synthesised output. An easily noticed difference in nearly all configurations is the opportunity for different coloured cars.

5.3.3 Perceptual loss issues

Table 5.9: FVD scores for network configurations relating to perceptual loss issues.

Index	FVD	Net.	O.I.	Norm	Percept.	Edge	F.E.	Res.	Disc.	Prior.	Twin.	Flow.	Video Link
1	1885.094	CRN	9 ¹	Layer	VGG	-	-	-	-	-	-	-	Link
2	1609.033	CRN	1	Weight	VGG	-	-	-	-	-	-	-	Link
3	1488.244	CRN	1	Weight	MNet	-	-	-	-	-	-	-	Link
6	1317.829	CRN	1	Weight	MNet	Edge	F.E.	Res.	Disc.	-	-	-	Link
10	1388.469	pix2pixHD	1	Inst.	-	Edge	F.E.	Res. ²	Disc.	-	-	-	Link
11	1304.637	pix2pixHD	1	Inst.	MNet	Edge	F.E.	Res. ²	Disc.	-	-	-	Link

Refer to table 5.1 for table notes.

Table 5.9 is available for easier comparisons of network configurations details. When compared to the results shown by [Chen and Koltun \[2017\]](#), the results of our CRN in index 1 leave much to be desired. While large amounts of time and effort were devoted to the exploration of this problem, we can only speculate on the causes; however, we are fairly certain that the problem lies in the implementation of the perceptual loss and differences in the layer normalisation implementations used between implementations.

Replacing layer normalisation with weight normalisation prevented the formation of artefacts in synthesised images, however as described in section 5.3.1 these artefacts are a dual problem, showing issues in both the network architecture and training methods. With this replacement, there is a general improvement in image quality and the artefacts are removed. While replacing VGG-19 with MobileNet-V2 does improve the image quality further as can be seen in index 3, both of these improvements should at least allow us to reach the quality of the original implementation; however, even after considerable testing, this was not the case.

The CRN does manage to learn advanced features in a similar manner to that of pix2pixHD when a discriminator is involved in the training process (index 6). Index 10 and 11 demonstrate the difference that adding the perceptual loss to pix2pixHD made, where this difference should be noticeable according to [Wang et al. \[2018b\]](#), however upon inspection it makes little to no difference to the training of pix2pixHD in our results. These two findings support our idea that the perceptual loss implementation is to blame, and not the CRN itself. One cause could be that PyTorch [\[Paszke et al. 2019\]](#) implements some operations with minor differences to the framework that is used in the official source code.



Figure 5.11: The difference between the synthesised output of the CRN’s original implementation and our implementation (index 1).

5.4 Conclusion

In this chapter, we discussed the results from our experiments, answered our research questions, and explored other important aspects in an attempt to further our understanding of the results and their implications.

Our qualitative analysis, focusing on our research questions, made consistent use of the FVD scores of the different network configurations to compare temporal consistency and investigate the reasons behind the scores that were achieved. We found that the CRN, with some minor modifications, can almost match pix2pixHD in temporal consistency, and that better image quality does not imply an equal improvement in temporal consistency. We found that porting components from pix2pixHD to the CRN did offer improvements

to the FVD score, and that it is possible to train the CRN with a discriminator without stability issues. Prior frame information was found to be the single most effective method of increasing temporal consistency in any configuration while the inclusion of optical flow estimation dramatically changed the outputs of both network architectures, with the CRN suffering a slight increase in FVD score for not being able to correctly use optical flow, whereas pix2pixHD used the extra capability to achieve the best FVD score. Finally, we found that when the CRN has the extra components from pix2pixHD, the FVD scores they achieve are close to each other.

In our qualitative analysis, we investigated how normalisation affects synthesis output, detailing the reasons why synthesis artefacts might appear depending on the network architecture. We analysed how the feature encoder has a greater effect than one might realise at first by allowing the network to learn better, and finally, we addressed the issues that we had with our implementation of the perceptual loss and its effects on the synthesis quality of then networks that used it.

Chapter 6

Conclusion

6.1 Discussion

Temporally consistent video synthesis is a difficult problem, requiring careful attention to each component and concept that is included in the network architecture. The current state of the art video synthesis networks are large, complicated, and require inaccessible hardware to train. Repurposing image synthesis networks for the task will not match the image quality or level of fine detail that specifically designed video synthesis networks achieve; however, these networks can, with the right modifications, make great strides in becoming temporally consistent.

We set out to compare two such networks, the Cascaded Refinement Network (CRN) [Chen and Koltun 2017], and pix2pixHD [Wang *et al.* 2018b], when tasked with synthesising temporally consistent video. pix2pixHD was created as a direct competitor to the CRN, built using components for which the CRN has no equivalent, as well as using a GAN architecture, something that Chen and Koltun [2017] explicitly decided against citing stability issues during training. We decided that to compare these networks, we would need to compare them both in their original forms, and create variants of the networks to investigate how different features influence the abilities of the different networks.

We took special interest in the CRN and investigated how it responded to improvements ported directly from pix2pixHD. We found that in its original configuration the CRN fell vastly short of matching pix2pixHD; however, when these improvements are included, the network’s ability to synthesise temporally consistent video increased, approaching the base pix2pixHD configuration.

We also revisited the idea of a GAN architecture for the CRN, using the methods that Wang *et al.* [2018b] used to ensure training stability, and test the network’s capabilities. Adding pix2pixHD’s discriminator to the training process of the CRN, including the adjustments to the discriminator losses that Wang *et al.* [2018b] use for stability, we found that the network trained without issue, giving the CRN the ability to synthesise temporally consistent video to much the same standard as pix2pixHD.

Besides porting components from pix2pixHD to the CRN, we also created variants of both networks which include video-specific features. The first feature was the addition of prior frame information into the training and synthesis processes and thus found that regardless of the network configuration, this is the most crucial change that should be made. Even without an explicit temporal consistency component in the loss function for training the networks, the networks will adapt to reusing content, leading to significant improvements in temporal consistency. The second feature we added, optical flow, where the motion of pixels between frames is calculated and used by the network, offered interesting results; however, this addition breaks from the spirit of this work as it bloats the networks, pushing these networks further away from the slimmer image synthesis networks and closer to the large, complex video synthesis networks. We found that the CRN could not make proper use of optical flow, and thus suffered a slight loss in temporal consistency, whereas pix2pixHD made full use of the extra information and created the most temporally consistent video with the greatest temporal consistency of all our network configurations.

We also investigated the link between image quality and temporal consistency and found that an increase in individual frame detail and fidelity does not give the same increase in temporal consistency, since adding more details requires them to be consistent between frames, something not possible without prior frame information. We discussed how normalisation is required for a network to train; however, the incorrect choice of normalisation used can lead to synthesis artefacts, and that weight normalisation typically removes all of them. We analysed the effects the feature encoder from pix2pixHD has on the network training and determined that it is more important than Wang *et al.* [2018b] envisioned it to be. Finally, we addressed how our implementation of the perceptual loss, the CRN’s primary means of training, appears to function incorrectly and not perform as it should. Due to the perceptual loss being a component of most network configurations, if it is corrected and functions as well as the implementation by Chen and Koltun [2017], a reevaluation of each configuration would be of interest.

These results also show that had the CRN been created later, there would likely have been many alterations to the design, or simply, it would not have been created at all, given how the more recent network architectures are closer to that of pix2pixHD than the CRN. However, with these modifications, the CRN is capable enough to synthesise temporally consistent video to a similar standard as to that of pix2pixHD. While they may have similar scores when both networks are given prior frame information without optical flow, the CRN synthesised more pleasant footage than pix2pixHD.

6.2 Future work

A number of aspects of this work can be extended further. Since the release of vid2vid [Wang *et al.* 2018a], newer temporally consistent video synthesis networks have been proposed, such as World consistent vid2vid [Mallya *et al.* 2020]. These can possibly be a source of new video specific network features that can be added to image synthesis networks. The CRN and pix2pixHD networks have been able to adapt to newer features; however, since the networks were proposed there are likely newer, more powerful image synthesis networks that can be used instead.

With regards to our implementation of the networks, a primary point of further investigation is that of the lacklustre performance of the perceptual loss compared to the results presented by [Chen and Koltun \[2017\]](#). While both networks suffered equally by its ill performance, the effects this had on the FVD score for the networks and their various configurations are not well known. While there is a chance that the primary effect of this is simply in image quality and not temporal consistency, further investigation is required to know for sure.

Finally, there are two aspects of our experimental methodology that can be expanded upon. First, the selection of feature encoding settings, as detailed in [section 5.2](#), are done by hand, with a certain degree of randomness. This does have an effect on the reproducibility of the FVD scored for networks that make use of the feature encoder, and further investigation into a method of selecting and evaluating these configurations. Secondly, the issue of hardware availability used to train and test network configurations prevented us from running our final experiments more than once due to the large amount of time required to run them, as detailed in [section 5.2](#). While our training hardware, listed in [section 4.5.3](#), is already very powerful and expensive, state of the art networks are run on hardware many times as powerful, and many networks can be trained and tested at once. With access to this hardware, our experiments can be conducted multiple times for better accuracy.

Appendix A

Model components

A.1 Residual refinement module

While the base configuration of the CRN concatenates all input data into a single tensor, when using the residual configuration the activations from the prior residual block, the semantic and edge annotations, and the prior synthesised images if prior frames are being used, are all handled as separate inputs. These three inputs are each processed by 3x3 convolution operations to perform some initial processing as well as to adjust the number of channels of each input to be equal to the designated channel count for that refinement module. These three processed inputs are then summed together and are further processed by two ResNext [Xie *et al.* 2017] residual blocks, before continuing on to the next refinement module.

In the final module, an additional 3x3 convolution precedes the existing final 1x1 convolution operation. This is necessary as at this point the activations are the output of a residual block, and thus contain unprocessed data from the beginning of the module. Without this additional 3x3 convolution operation, there is a noticeable degradation of quality and smoothness. Additional 3x3 convolution operations can be added, however, this dramatically increases memory requirements and compute time, while not adding substantial benefit.

Bibliography

- [Abadi *et al.* 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org.
- [Arjovsky *et al.* 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.
- [Bojanowski *et al.* 2017] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.
- [Bradski 2000] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [Chen and Koltun 2017] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520, 2017.
- [Chen *et al.* 2018] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.
- [Cordts *et al.* 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [Dong *et al.* 2019] Haoye Dong, Xiaodan Liang, Xiaohui Shen, Bowen Wu, Bing-Cheng Chen, and Jian Yin. Fw-gan: Flow-navigated warping gan for video virtual try-on. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1161–1170, 2019.

- [Dosovitskiy *et al.* 2015] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.
- [Engel *et al.* 2019] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.
- [Gatys *et al.* 2016] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.
- [Goodfellow *et al.* 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [Haris *et al.* 2018] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1664–1673, 2018.
- [He *et al.* 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [He *et al.* 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.
- [Heusel *et al.* 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
- [Huang *et al.* 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [Hubel and Wiesel 1968] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.
- [Hur and Roth 2020] Junhwa Hur and Stefan Roth. Optical flow estimation in the deep learning age. In *Modelling Human Motion*, pages 119–140. Springer, 2020.
- [Huszár 2015] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.
- [Ilg *et al.* 2017] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep

- networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.
- [Ioffe and Szegedy 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [Isola *et al.* 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [Johnson *et al.* 2016] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.
- [Johnson *et al.* 2018] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.
- [Karras *et al.* 2019] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.
- [Karras *et al.* 2020] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.
- [LeCun *et al.* 1995] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [Lei Ba *et al.* 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [Long *et al.* 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [Maas *et al.* 2013] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.
- [MacQueen and others 1967] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [Mallya *et al.* 2020] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII* 16, pages 359–378. Springer, 2020.

- [Mao *et al.* 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.
- [Miyato *et al.* 2018] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.
- [Park *et al.* 2019] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.
- [Paszke *et al.* 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.
- [Pearson 1901] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- [Radford *et al.* 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [Ranjan and Black 2017] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4161–4170, 2017.
- [Salimans and Kingma 2016] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909, 2016.
- [Sandler *et al.* 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.
- [Santurkar *et al.* 2018] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in neural information processing systems*, pages 2483–2493, 2018.
- [Shah *et al.* 2017] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.
- [Simonyan and Zisserman 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [Szegedy *et al.* 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceed-*

- ings of the *IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [Ulyanov *et al.* 2016] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [Unterthiner *et al.* 2018] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.
- [Wan *et al.* 2020] Ziyu Wan, Bo Zhang, Dongdong Chen, Pan Zhang, Dong Chen, Jing Liao, and Fang Wen. Bringing old photos back to life. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2747–2757, 2020.
- [Wang *et al.* 2018a] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [Wang *et al.* 2018b] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.
- [Wang *et al.* 2019] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*, 2019.
- [Weinzaepfel *et al.* 2013] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013.
- [Wu and He 2018] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [Wu *et al.* 2019] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [Wulff *et al.* 2012] Jonas Wulff, Daniel J Butler, Garrett B Stanley, and Michael J Black. Lessons and insights from creating a synthetic optical flow benchmark. In *European Conference on Computer Vision*, pages 168–177. Springer, 2012.
- [Xie *et al.* 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.
- [Yang and Ramanan 2019] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Advances in neural information processing systems*, pages 794–805, 2019.

[Zhou *et al.* 2020] Yufan Zhou, Haiwei Dong, and Abdulmotaleb El Saddik. Deep learning in next-frame prediction: a benchmark review. *IEEE Access*, 8:69273–69283, 2020.