GOLDSMITHS, UNIVERSITY OF LONDON

DISSERTATION

Autoencoding Video Frames

Author:
Terence BROAD

Supervisor:
Dr. Mick GRIERSON

A dissertation submitted in partial fulfillment of the requirements for the degree of Msci Creative Computing

in the

Department of Computing

May 12, 2016

Abstract

This report details the implementation of an autoencoder trained with a learned similarity metric - one that is capable of modelling a complex distribution of natural images - training it on frames from selected films, and using it to reconstruct video sequences by passing each frame through the autoencoder and re-sequencing the output frames in-order. This is primarily an artistic exploration of the representational capacity of the current state of the art in generative models and is a novel application of autoencoders. This model is trained on, and used to reconstruct the films *Blade Runner* and *A Scanner Darkly*, producing new artworks in their own right. Experiments passing other videos through these models is carried out, demonstrating the potential of this method to become a new technique in the production of experimental image and video.

Acknowledgements

I would like to thank my parents and Grandfather for all the support they have given me over the years, and particularly for both giving me the financial support I needed in helping to purchase the two new computers that were essential for me carrying out this project.

I would like to thank Emma for the support this year, and all the previous years that we have been studying together. Hopefully I will now be able to offer her more support over the summer while she is completing her dissertation.

But in particular I would like to extend my gratitude to Mick. I can't believe how much I have learned and progressed in the past four years at Goldsmiths, and he has been central to my development throughout my time here. I am particularly grateful that he offered me the opportunity to extend my time at Goldsmiths doing an additional year of research, which is in itself a rare and unique opportunity. But on top of all that I will be eternally grateful that he pushed me towards researching deep learning at such an exciting and pivotal time.

Contents

List of Figures

Chapter 1

Introduction

This report details the efforts in building an autoencoder capable of modelling a complex distribution of natural images, training it on frames from selected films, and using it to reconstruct video sequences by passing each frame through the autoencoder and re-sequencing the output frames in-order. This is primarily an artistic exploration of the representational cacity of the current state of the art in generative models - as far as I am aware - this is a novel application of autoencoders, and has the potential to become a new method in the production of experimental image and film.

1.1 Motivation

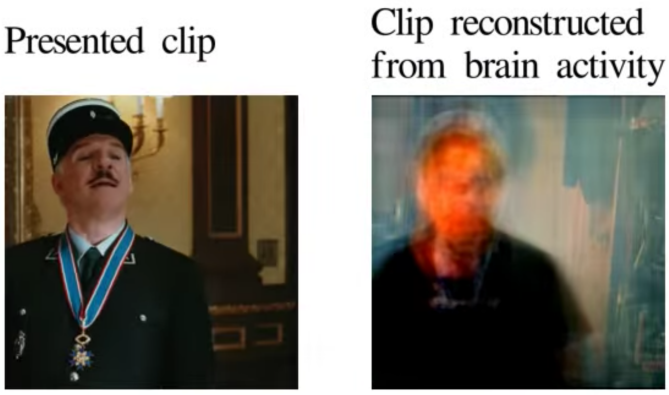

FIGURE 1.1: Reconstruction of visual experiences from brain activity evoked by natural movies (Nishimoto et al., 2011).

I originally had to idea of trying to view videos through artificial neural networks after seeing the talk “Mindreading in Modern Neuroscience” by John-Dylan Haynes (2013) at Ars Electronica. In the talk he demonstrated the ability to reconstruct images of what people were seeing, using fMRI and a Bayesian decoder that combines frames from the closest seen matches in the previously seen posterior clips (see figure 1.1), I found the results astonishing and wondered if it was possible to do the same thing with artificial neural networks. I had considered doing attempting to do this for my undergraduate dissertation, but even as recently as last year, development in neural networks was not at the stage where this would have been possible. Even in September 2015, when I started investigating generative

models, it seemed like models for complex natural images where a long way off. Luckily, in the past 6 months there has been amazing advances in the development of generative models that have made this work possible.

1.2 Initial Research

Although I had the idea to reconstruct video with neural networks several years ago, at the beginning of the academic year this did not seem feasable, and I had spent most of my time understanding the two best generative models at that time (purely to research image generation); DRAW (section 2.2.3) and LAPGAN (section 2.3.1). My research into DRAW, led me to reading about DARN (section 2.2.3) and finding the work done to combine that with an LSTM to do binarized video prediction of Atari games (Graves and Freitas, 2015), this led me to attempt to reproduce this, with an eye on furthering it by improving the image model combined with the LSTM to handle more natural images.

However when DCGAN (section 2.3.2) was published it was obvious that using a convolutional model was the best way to produce natural images. I then turned my attention to finding some way of combining the advarsarial approach with an autoencoder, and maybe even using another network to compare the similarity of the images produced by an autoencoder with the real images. Within a month however, a paper was published that combined both of these approaches in a very surprising and elegant way (section 2.4.1). Therefore once again, I turned my attention to implementing this method, with the idea that would combine in it with an LSTM and do video prediction with this more sophisticated autoencoder model. Unfortunately due to the time taking to build this model, getting it functioning in a stable manner, and then the time taken training the models, I was not able to progress with the line of research. But - perhaps serendipitously - it led me back to the original idea I had years ago; reconstructing videos with neural networks.

1.3 Final Idea

Once it was clear that I would restrict my focus to using autoencoders to reconstruct images, I should put some thought into *what* videos I was going to reconstruct. I remembered a talk I went to at LCC in 2014 by the author and Professor of media theory Charlie Gere. At the end of the talk Gere spent quite some time talking about the film *Blade Runner* - the adaption of Philip K. Dick's novel *Do Androids Dream of Electric Sheep?* - and how it reflects on the limits of human finitude and experience. Given the subject matter, it was obvious that *Blade Runner* was the most pertinent choice of film to be reconstructed using a neural network (I expand on this in the artistic motivation - section 3.3). The second film I chose to reconstruct was *A Scanner Darkly* (another Phillip K. Dick adaption) which is an interesting film to model not only because of the subject matter, but as it was animated using the interpolated rotorscope method.

Chapter 2

Context

2.1 History of Artificial Neural Networks

Artificial neural networks are a family of computational models that are loosely inspired by biological neural networks. There is no strict formal definition of what an artificial neural network is, but statistical models that have multiple units connected by a set of adaptive weights, that are tuned by a learning algorithm, which can then approximate non-linear functions upon inputs are often classified as artificial neural networks. The units in artificial neural networks are often known as 'neurons' or 'nodes', for the rest of this report they will referred to as nodes. These nodes perform a function on a weighted sum of their inputs, this function is known as the activation function and is most often a nonlinear function such as the sigmoidal function or tanh.

Statistical models that can be classified as 'neural' date back to when McCulloch and Pitts (1943) created a computational model for biological neural networks using mathematics which they called threshold logic. A notable milestone was the development of the perceptron by Rosenblatt (1958), which was a simple two layer network with a learning algorithm using addition and subtraction. This garnered a lot of attention and excitement at the time, but research stagnated after Minsky and Papert (1969) published a paper proving that perceptrons could not learn exclusive-or circuit, and that computers didn't have the processing power capable of to effectively handle the long run time needed for large neural networks. It was not until Werbos (1974) developed the backpropagation algorithm that perceptrons could effectively learn the exclusive-or circuit, causing a resurgence in the field of developing multi-layered perception (MLP) networks.

2.1.1 Deep Learning

Once again in the 1990's research interest in neural networks dwindled as researchers developed other machine learning methods such as support vector machines that were more efficient to train (in hindsight interest probably dropped because of a lack of computational power and insufficiently large datasets). It was not until Hinton and Salakhutdinov (2006) first developed a method that outperformed principal components analysis at reducing the dimensionality of data - using a deep autoencoder that had been pre-trained

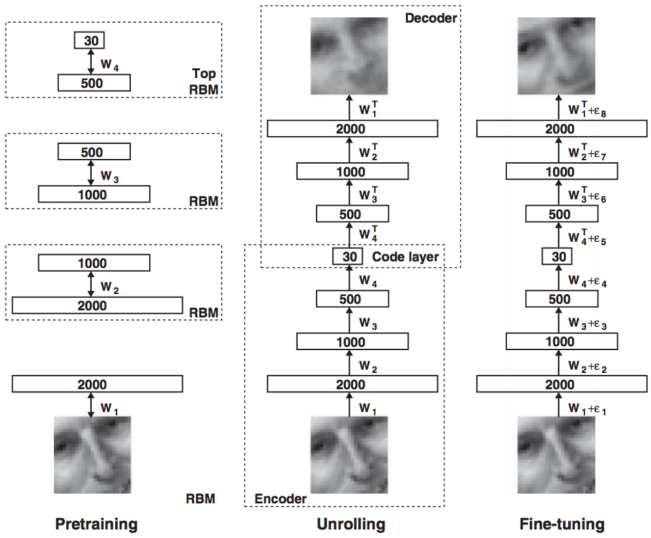

FIGURE 2.1: Unrolling a stack of restricted Boltzmann Machines into an autoencoder (Hinton and Salakhutdinov, 2006).

as a stack of restricted boltzmann machines - that a resurgence in interest in neural networks happened, giving rise to the field of deep learning. Although 'deep' neural networks (networks with many layers) had been developed prior to this, they had not particularly useful as it had been difficult to backpropagate error derivatives through many layers, and there was not sufficient computational power to train these networks for a long time efficiently.

Although it had been proven that a multi-layered percetron network with a single layer containing a finite amount of nodes could approximate any continuous function (Gybenko, 1989), deep neural network models are much more efficient and flexible in approximating complex functions. However, the most import capability afforded to deep neural network models is the ability for the network the learn a hierarchy of increasing complex and abstract features. This was first demonstrated visually by Lee et al. (2009) with their convolutional deep belief networks (see figure 2.2).

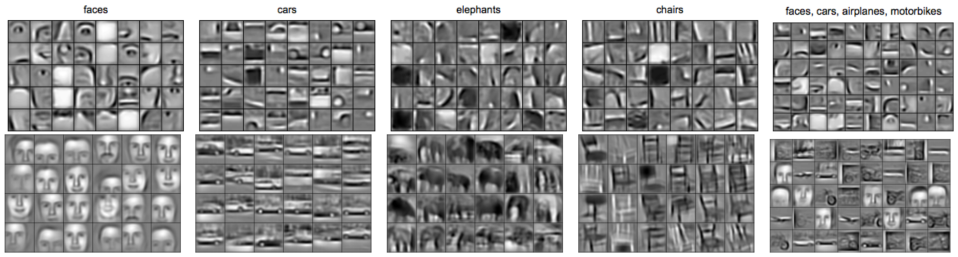

FIGURE 2.2: Hierarchy of features in convolutional deep belief networks (Lee et al., 2009)

Since 2006, various types of deep neural networks have made significant breakthroughs in many domains of machine learning, and are now the state

of the art in most of these fields. The most notable breakthrough being the use of a deep convolutional neural network to significantly outperform state of natural image classification from the ImageNet 2010 competition (Krizhevsky, Sutskever, and Hinton, 2012). Since then, deep convolutional neural networks are the de facto method for image classification and can now outperform humans in certain tasks in image classification (He et al., 2015).

2.1.2 The Development Towards Generative Models

Although deep convolutional neural networks are very powerful at image classification tasks, some of the criticisms leveled at them is that they are black box methods (meaning it is difficult to understand the mechanics of how they work), and that they learn to correlate features statistically useful for classifying objects but are not components of the object themselves. It has also been demonstrated that it is easy to train another network to manipulate an image imperceptibly in order to fool another convolutional neural network to make and incorrect classification (Szegedy et al., 2013). In response to this, there has been a drive to develop neural networks that can generate images well, the theory being that if a model can hallucinate well it has a much deeper understanding of the world (Freitas, 2015).

2.2 Autoencoders

Autoencoders are neural networks with two components, one that restructures data into an encoding of reduced dimensionality, and another that reconstructs the data from the encoding. The network is given a data input x the encoder encodes it into a latent representation z and the decoder reconstructs the data input $\bar{x}$, the cost function used to train the network is then defined as the mean squared error between the input x and the reconstruction $\bar{x}$.

$$argmin = \|x - \bar{x}\|^2 \tag{2.1}$$

Originally these models were used for reducing the dimensionality of data and mapping it to meaningful encodings (Hinton and Salakhutdinov, 2006), but recently they have been used for learning generative models of data.

2.2.1 Denoising Autoencoders

Denoising autoencoders (Vincent et al., 2008) take a partially corrupted input and attempt to reconstruct the original undistorted input, forcing the autoencoder to learn more robust and general features (an approach that was later extended to corrupting the hidden layers any given network with the dropout training method (Srivastava et al., 2014)). A preliminary stochastic mapping is performed $x \rightarrow x'$ before the network encodes and reconstructs

the input. However the network is trained to reconstruct the origin input $L(x, \bar{x}')$ rather than the corrupted input $L(x', \bar{x}')$.

2.2.2 Variational Autoencoders

Variational autoencoders (independently developed by Kingma and Welling (2013) and Rezende, Mohamed, and Wierstra (2014)) inherit the same architecture from the previously described autoencoder models; but place a strict assumption on the distribution of latent variables, taking a variational bayesian approach to learning the latent representation. This model assumes the original data given to the network is generated by a directed graphical model $p(x|z)$ and that the encoder is learning an approximation $q_\phi(z|x)$ to the posterior distribution $p_\theta(z|x)$, where ϕ and θ denote the parameters of encoder and decoder. The objective function of a variational autoencoder is given as:

$$L(x) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}(\log p_\theta(x|z)) \tag{2.2}$$

The second term in the equation $E_{q_\phi(z|x)}$ is the reconstruction error, this is often the log of the mean squared error between the original input and reconstruction cost (as given in equation 2.1), this is referred to as the log-likelihood. D_{KL} is defined as the Kullback-Leibler divergence, measuring the divergence of the approximate posterior from the prior distribution. The prior over the latent variables is given as a centred isotropic multivariate Gaussian $p_\theta(z) = \mathcal{N}(0, I)$. The equation 2.2 thus expresses a lower-bound on the log likelihood, crucially variational autoencoders maximise a lower-bound on the log-likelihood rather than maximising the likelihood directly.

In order to efficiently adapt $q_\phi(z|x)$ to improve the reconstruction, the latent representation z is a deterministic function of ϕ and noise ϵ. When the latent representation z approximates the posterior distribution that has a univariate Gaussian prior:

$$z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2) \tag{2.3}$$

Then z can be reparameterized as:

$$z = \mu + \sigma\epsilon \tag{2.4}$$

In the case the prior distribution on z is given as:

$$p_\theta(z) = \prod_{i=1}^{N} \mathcal{N}(z_i|\ 0, 1) \tag{2.5}$$

And the approximate posterior distributions is normal and factorised:

$$q_\phi(z|x) = \prod_{i=1}^{N} \mathcal{N}(z_i|\ \mu_i(x), \sigma_i^2(x)) \tag{2.6}$$

The Kullback-Leibler divergence term integrates to:

$$D_{KL} = \frac{1}{2} \sum_{i=1}^{N} 1 + \log(\sigma_i^2(x)) - \mu_i^2(x) - \sigma_i^2(x) \tag{2.7}$$

A complete derivation of this can be found in the paper by Kingma and Welling (2013).

2.2.3 Extended Autoencoder Models

There are a number of generative models that take the basic premise of an autoencoder and expand on the architecture in unique and complex ways. They will not be covered in great detail as this project does not directly build on them, but they are worth noting due to their generative capabilities.

Deep AutoRegressive Networks

Deep AutoRegressive Networks (Gregor et al., 2013) takes the standard autoencoder architecture but adds successive layers of stochastic binary units equipped with autoregressive connections, enabling the model to be sampled efficiently using ancestral sampling. The cost function is based minimum description length principle - minimising the amount of stochastic binary units used to encode an output - which can be seen as maximising a variational lower bound on the log-likelihood. Deep AutoRegressive Networks are only capable of represented binary images, and at the time were the state of the art for generative modelling of MNIST and binarized frames from Atari 2600 games. In a lecture by Alex Graves (2015) he demonstrates a model where an LSTM (a type of recurrent neural network) is intertwined between the encoder and decoder of a DARN network, the system is able to predict sucessive frames in Atari games, and hallucinate long sequences of these frames when it is repeatedly given frames it has produced.

Deep Recurrent Attentive Writer

The Deep Recurrent Attentive Writer (DRAW) (Gregor et al., 2015) superseded DARN as the state of the art in generating MNIST, and when trained on the Google Street View House Number dataset it generated images that can not be distinguished by the naked eye. DRAW utilises a novel spatial attention mechanism that mimics the foveation of the human eye. Combined with a sequential variational auto-encoding framework that allows for the iterative construction of complex images (see figure 2.3). The encoder and decoder networks are recurrent neural networks - specifically Long Short Term Memory networks - that exchange a sequence of codes to iteratively

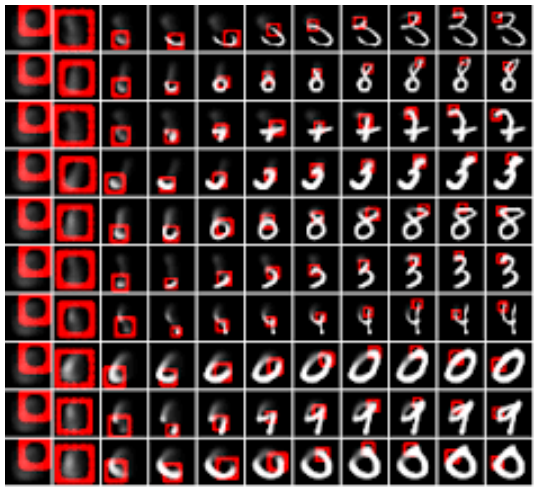

FIGURE 2.3: A Trained DRAW network generating MNIST digits. Each row show succesive stages in the generation of a single digit. (Gregor et al., 2015)

construct the image; as opposed to one code representing one image, which is the standard case for autoencoders.

One-Shot Generalisation in Deep Generative Models

In the paper "One-Shot Generalisation in Deep Generative Models" Rezende et al. (2016) attempt to combine the representation power of of deep learning and the inferential power of bayesian reasoning. They develop a class of sequential generative models that are built on the principles of feed back and attention, capable of generating convincing and diverse samples after seeing only one example of an object class. In addition to this they combine the sequential generative model (built up the the DRAW architecture - 2.2.3) with spatial transformer networks (Jaderberg, Simonyan, Zisserman, et al., 2015), a learnable module using convolutional neural networks that are capable of spatial manipulation of data within a larger network. The combination of these two approaches has undoubtably created the best generative model to date for generating samples of a single class of object (see figure 2.4).

FIGURE 2.4: Generated faces from spatial transformer networks combined with an additive canvas with feedback and attention over 32 steps (Rezende et al., 2016).

2.3 Generative Adversarial Networks

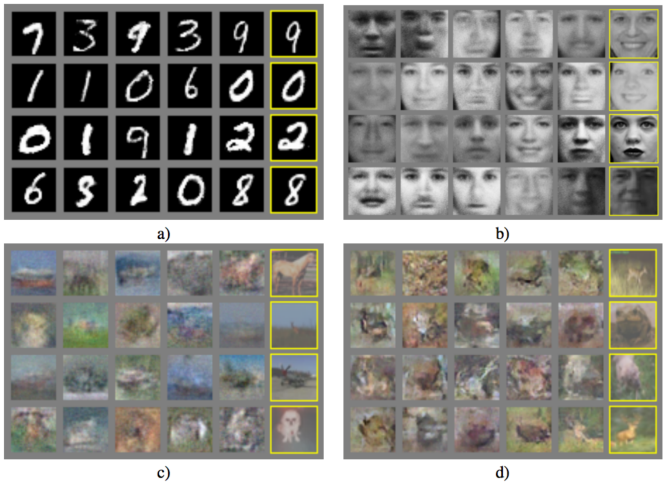

FIGURE 2.5: Images created using generative adversarial networks: a) MNIST dataset, b) TFD dataset, c) and d) CIFAR-10 dataset using convolutional discriminator and deconvolutional generator (Goodfellow et al., 2014).

Generative Adversarial Networks, introduced by Goodfellow et al. (2014) offers an alternative approach to generating images with neural networks. Instead of using markov-chain monte-carlo to maximise the likelihood of the data, or even maximising a variational lower bound, this technique simply plays a game between two networks. A generator network G takes random numbers and converts it into samples and a discriminator network D that attempts to differentiate between samples from the real data, and samples from G. Akin to counterfeiters G producing fake bank notes and police D trying to determine whether a bank note is real or fake. The training procedure for D is to maximise the probability of assigning correct labels for training examples from the data and samples from G.

The training procedure for G is to maximise the probability of D making a mistake, or in other words to minimise $\log(1-D(G(z)))$, where z is the prior input noise variables and x is the data. This can be defined as a two-player minimax game with a value function $V(G,D)$:

$$\min_G \max_D V(G,D) = \mathbf{E}_{x\ p_{data}(x)}[\log D(x)] + \mathbf{E}_{z\ p_z(z)}[\log(1-D(G(x))] \quad (2.8)$$

The difficulty in training GAN's is that it is essential to balance the rate of learning of both the generator and discriminator in order that they learn at the same rate. If the discriminator learns too quickly, it will quickly start to classify every generated sample as being fake and the error signals propagated through the discriminator will become meaningless. Likewise if the generator learns to fool the discriminator too quickly, it may be fooling a

discriminator network that is yet to have developed a sophisticated understanding of the distribution of the data, and the quality of the generated samples will cease to improve.

In the original paper describing Generative Adversarial Networks, Goodfellow et al. (2014) use feedforward multilayer perceptron networks using a combination of rectified linear units[1] and sigmoidal units for the activation functions of the generator network, while the discriminator network uses a combination of maxout[2] activations using dropout[3] as a regulariser. This approach produces good results on the MNIST and Toroto Face Datasets, but presents noisy, incoherent samples on the CIFAR-10 dataset of natural images (see figure 2.5).

2.3.1 Laplacian Pyramid of Generative Adversarial Networks

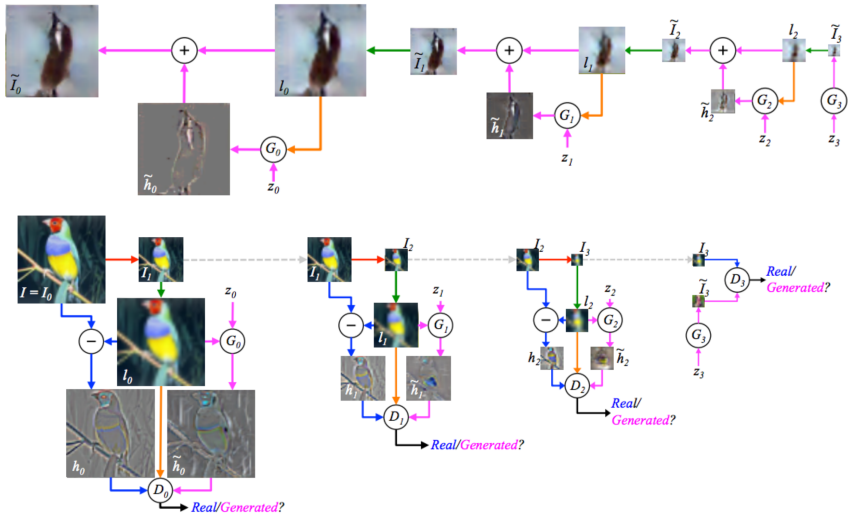

FIGURE 2.6: Top: Sequence of steps in generating samples from LAPGAN. Bottom: Sequence of steps in training LAPGAN, at every stage, a data sample is down-sampled and the discriminator network compares the the true laplacian transformation to form the real image and the one generated by the generator network (Denton, Chintala, Fergus, et al., 2015).

Denton, Chintala, Fergus, et al. (2015) extend the GAN approach to develop the first model capable of producing high quality samples of natural images. They present an architecture of cascading generator and discriminator networks, spaced one octave apart. The networks generate and discriminate band-pass images that act as up-sampling operators which smooths and expands an image to be twice the size. To train the network real data samples (in this case 64x64)are down-sampled to half the size, the true Laplacian band-pass image is extracted and one is generated by the generator network, the discriminator network processes both samples and both networks

[1]Rectified linear units were first described by Nair and Hinton (2010).
[2]Maxout activation functions were first described by Goodfellow et al. (2013).
[3]Using dropout as a regulariser was first described by Srivastava et al. (2014).

are trained using the standard GAN objective function described in equation 2.8. This process is repeated down to a 4x4 image, in which there is a traditional GAN generator network that also generates a 4x4 image for comparison. Once training is completed the network can be sampled by generating a 4x4 seed image and upscaling the image to twice the size and combining it with the generated band-pass image at each stage in the network pyramid.

2.3.2 Deep Convolutional Generative Adversarial Networks

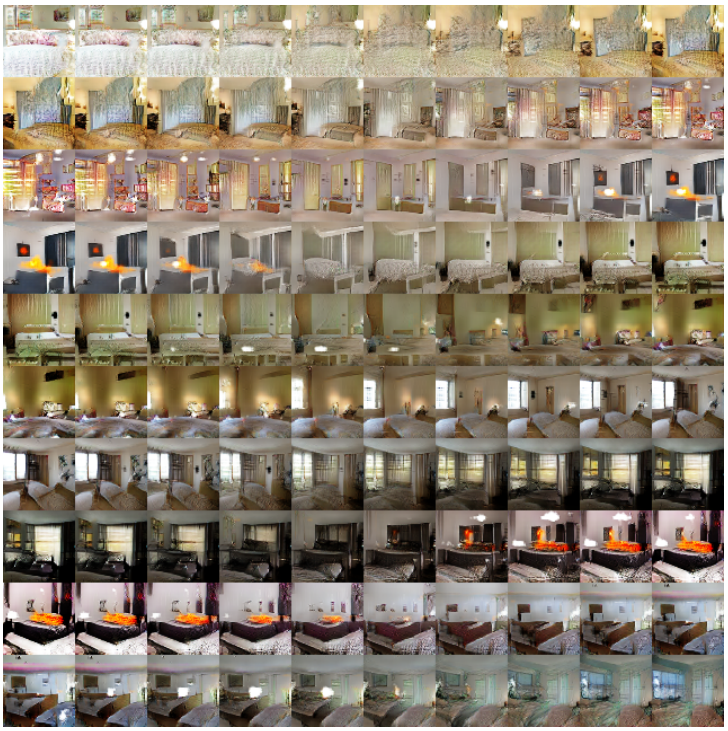

FIGURE 2.7: Interpolating between different points in the latent space of a DCGAN trained on the LSUN bedrooms dataset (Radford, Metz, and Chintala, 2015).

It was widely assumed that convolutional neural networks were not suitable for the generation of images (Freitas, 2015), despite their vast superiority for approaching image recognition. This was because of the standard use of deterministic pooling layers (usually max-pooling) in convolutional neural networks. Max-pooling layers are used to spatially down-sample image representations at increasingly higher layers in the network, but because they only pick the maximum pixel value from a windowed region (usually a 4x4 region), spatial information is lost and the process is not invertible. However, Radford, Metz, and Chintala (2015) demonstrated it was possible to use convolutional neural networks to generate realistic natural images. To overcome the problem of lost spatial information incurred by the use of max-pooling layers, their model just has layers of strided convolutions, rather than alternating layers of convolutions and max-pooling. This allows both the discriminator and generator network to learn their own spatial down-sampling and up-sampling.

The approach of an all convolutional network was first developed by Springenberg et al. (2014), in an attempt to simplify CNNs, and yields competitive or state of the art performance on multiple different object recognition datasets. In addition to using all convolutional networks, Radford, Metz, and Chintala (2015) eliminate the traditional fully connected layers used at the top CNN's, instead opting for the top layer of the discriminator network to be re-shaped linearly and fed into a sigmoid function. The generated network is instantiated by a 100 digit random number from a uniform noise distribution Z, and is re-shaped and projected onto a 4-dimensional tensor that acts as the first layer in the convolutional stack. Batch normalisation (Ioffe and Szegedy, 2015) is implemented in the DCGAN model and is used as a regulariser; helping gradient flow in deep models, compensating for poor initialisation of the network and allowing for the use of much higher learning rates. The generator network uses ReLU (rectified linear units (Nair and Hinton, 2010)) with the exception of the output layer which has a tanh activation. The discriminator network uses leaky rectified activation units (Maas, Hannun, and Ng, 2013) which was found to work especially well for higher resolution image modelling. Training is performed using the ADAM optimiser (Kingma and Ba, 2014) with a learning rate of 0.0002.

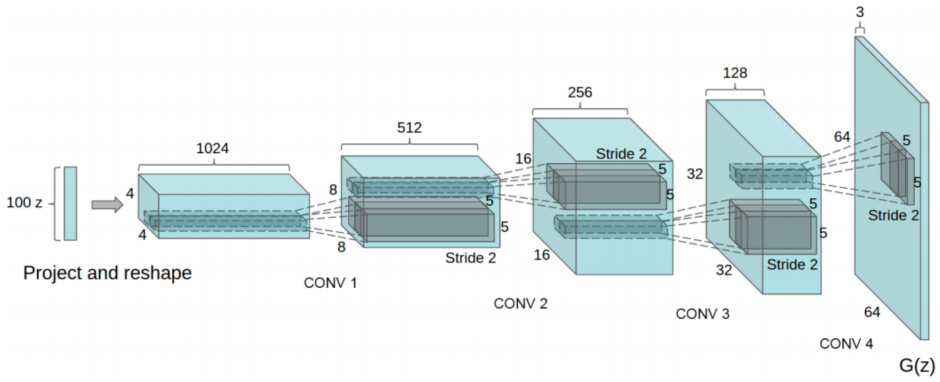

FIGURE 2.8: Diagram of the generator network in the DCGAN model, a 100 dimensional uniform distribution Z is reshaped and projected onto a Tensor representing 1024 4x4 image regions. A set of four fractionally strided convolution layers upscale the image to a 64x64 colour image (Radford, Metz, and Chintala, 2015).

2.4 Combining Autoencoders and Adversarial Networks

There were several attempts in 2015 to combine the GAN approach with an autoencoder. Larsen and Sønderby (2015) demonstrated a variational autoencoder that alternated its training as a GAN, with the decoder of the VAE acting as the generator in the GAN architeture. This allowed them to generate samples of faces with increased higher frequency content than a VAE (which tends to produce smoothed samples). Makhzani et al. (2015) take a similar approach, but also use the GAN in order to shape the latent space according to classification labels, able to shape the latent space directly

into gaussian distributions or even something completely arbitrary like a swiss roll distribution.

2.4.1 Autoencoding With A Learned Similarity Metric

Following on from the original work by Larsen and Sønderby (2015) in alternating a VAE with a GAN, Larsen, Sønderby, and Winther (2015) build upon this model, they use the feature representations in the GAN discriminator as the basis for the VAE reconstruction objective. This is quite a significant advance because the pixel-wise euclidean distance between two images is not a good feature representation of the similarity between images because it is not modelled on the properties of human visual perception. A small translation may result in a large pixel-wise error but would be imperceptible to the human eye, and as this is the basis objective function for a VAE, it results in the model learning low frequency, smoothed representations (see figure 2.10), to compensate for this invariance. Therefore they introduce a Gaussian observation model (euclidean distance) for the distance in response in the higher levels of the discriminator network, providing a *sufficiently invariant* representation of the images. This technique is used as the main inspiration for this project (although it is not a direct implementation) so a thorough description of the network architecture and training procedure is outlined in chapter 4.

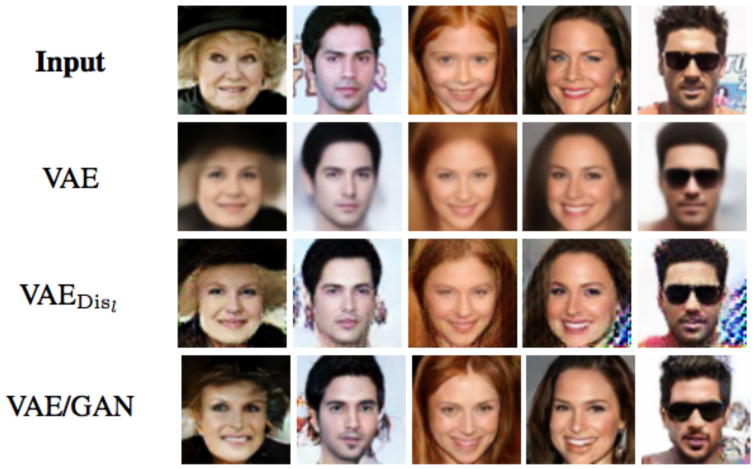

FIGURE 2.9: Results from different autoencoder models trained on the celebA dataset (Ziwei Liu and Tang, 2015). 1st row: Input image. 2nd row: Reconstruction from VAE. 3rd row: Reconstruction from VAE trained with discriminator distance metric. 4th row: Reconstruction with VAE model trained with discriminator distance metric that is additionally balanced with the GAN objective function. (Larsen, Sønderby, and Winther, 2015).

2.5 Other Techniques For Reconstructing Video

Figure 1.1 in the previous chapter illustrates the work by Nishimoto et al. (2011) for reconstructing video of what people are seeing fMRI data. This is not reconstructing videos directly, but is still worth noting. They have a bayesian framework that encodes voxel responses of brain blood-flow information, semantic information recorded from the visual processing parts of the brain (occipital cortex anterior) and image priors from the training images shown to subjects. In testing they then use voxel responses and the semantic information ascertained from the fMRI to approximately reconstruct the video that the subject is watching. This technique is not using a artificial neural network, but rather hand-engineered image filters.

Casey and Grierson (2007) present a system for real-time matching of an audio input stream to a database of continuous audio or video. They perform nearest neighbour searching on variable-length sequences of audio features, allowing real time reconstruction of an audio stream. In addition to this, they describe an application called REMIX-TV where both the audio and video of a film corpus is analysed, and video can be new video sequences can be reconstructed using correlated audio features from an audio-stream. Grierson (2009) develops on this work with PLUNDERMATICS with more sophisticated methods for feature extraction, segmentation and filtering. Audio onset-detection, motion-based shot boundary detection and matched filtering are all used to enable audiovisual segmentation. Allowing for various real-time audiovisual interaction technologies.

Mital, Grierson, and Smith (2013) extend this approach further to synthesis a target image using a corpus of images. The image is synthesised in fragments that are matched from the database extracted from the corpus based on shape and colour similarity. Adjusting the parameters of the synthesis process can result in aesthetic stylisations associated with Impressionist, Cubist, and Abstract Expressionist paintings. The fragment matching is performed using nearest neighbour.

FIGURE 2.10: Corpus-based visual synthesis: an approach for artistic stylisation. Results from Mital, Grierson, and Smith (2013) reconstructing videos from a corpus of source images. In this case a clip from Akira Kurosawa's "Dreams" is reconstructed using Vann Gogh's "Langlois Bridge at Arles" as the corpus. Left: Input video. Right: Reconstructed video.

Chapter 3

Research Questions

3.1 Aim

The aim for this project is to implement the most sophisticated existing autoencoder model for producing images; which is capable of modelling a wide distribution of natural images, therefore being the most suitable model for modelling the distribution of images (individual frames) of any given video. With the explicit intent of using these models to recreate the videos used to train the autoencoder - by autoencoding every frame and reconstructing the whole video in the original sequence - and autoencoding and reconstructing other videos that the models have not been trained on. The artistic motivation for using autoencoders to model and reproduce individual films is outlined in section 3.3. The technical challenges facing this task are outline in the next section.

3.2 Technical Challenges

In the aim I stated that I wanted to implement the most sophisticated autoencoder model that is flexible enough to model a wide distribution of natural images. Therefore I have chosen to implement the model of autoencoding with a learned similarity metric (Larsen, Sønderby, and Winther, 2015 - see section 2.4.1). This has many advantages: it can be modelled as a convolutional generative network, the best available model for natural images; it uses a third - discriminator - network that learns to discriminate from real images and generated images, that can learn to model and distinguish image style, and uses this network to measure reconstruction error in a more sophisticated way than the traditional pixel-wise euclidean distance.

Implementing this model alone is a substantial technical challenge. I have chosen to do this in Google's recently released machine learning framework Tensorflow (Abadi et al., 2015), which is still in relatively early stages of development. Training models with adverserial architectures is also notoriously difficult (Goodfellow et al., 2014), the rate of learning for each network must be finely balanced, in order for all the networks to learn and develop at the same rate. When building the model, it is also challenging to optimise for efficiency, while attempting to optimise the aesthetic quality of the results from the model.

The specific challenge to modelling video frames - as opposed to modelling a dataset one classification of image object all aligned in the centre of the image - is that there is a wide variety of composition regarding lighting and configuration of objects in a scene. This variety will certainly be a challenge for the model to represent (one which is interesting to explore artistically), but another challenge is that a lot of the frames in a film are very similar, say the camera being fixed on one actors face for an extended monologue. Balancing the models capability to represent a large variety of scenes, while also representing fine-tuned differences between vert similar frames will be a unique challenge facing this project.

Ideally I would be reconstructing video frames in the highest resolution possible, unfortunately, however, the limitations of memory available on modern GPU's, and the time taken to train these models, restricts the resolution that these generative models are able to represent. The standard resolution for the current state of the art models (such as DCGAN - section 2.3.2) is 64x64 pixels. For the reconstruction of video, it is obviously preferable to model images that are in a non-square aspect ratio. I will also pursue modelling images at a high a resolution as possible within the practical limitations of physical memory and training time.

3.3 Artistic Motivation

One of the fundamental goals of artificial intelligence - dating back to the work of Alan Turing (1950) - is to build machines that can think like humans. This is a milestone that has yet been realised, despite human levels of performance being achieved and exceeded in certain tasks. One of the criticisms of traditional approaches in AI is that they were processing human-defined symbolic representations, rather than processing real world data. One of the great successes of deep learning is the development of general purpose algorithms that can process many forms of raw real-world data. It is telling that Google DeepMind - in their pursuit of true artificial general intelligence - are training their AI systems in increasingly complex interactive virtual worlds[1]. The push toward developing systems that can negotiate increasingly complex virtual worlds - and ultimately the real world - is almost certainly influenced by the theory of embodied cognition.

Embodiment and enactivism are almost certainly the best frameworks within which we can understand the mind and the mechanics of perception (Noe, 2004), and may ultimately lead to the development of true artificial general intelligence. However, these are only theories of cognition, and other theories of cognition can be modelled - at least allegorically – given the implicit assumptions made about how an agent is structured and the way in which it interacts with its environment. In particular, disembodied models

[1]DeepMind made their first major breaktrough with an general purpose AI system that could learn to play many different ATARI games without any prior knowledge of the game (Mnih et al., 2015). They have now developed a 3D maze environment which can be continually modified to add tasks of ever increasing complexity (Mnih et al., 2016).

of cognition such as Cartesian dualism - which holds that the mind is a non-physical and therefore, non-spatial substance (Descartes, 1967 [1641]) - can be explored.

The visual model of the modern era, which can be associated with the Renaissance notions of perspective in the visual arts, and the Cartesian idea of subjective rationality in philosophy; combined, these are known as Cartesian perspectivalism, which as Martin Jay (1988) argues, is the prevalent scopic regime of modernity. The ability to represent the world mathematically as projected onto a flat plane, has dominated peoples understanding of the world since the Renaissance. This theory of mind, combined with the use of the camera obscura by artists in reproducing images, ultimately let to the development of the photographic camera. Ironically, this disembodied theory of perception gradually had to be discarded to take into account the physiology of vision, i.e. the physical and chemical processes needed to fix images (Gere, 2015).

While completely disembodied perception had to be discarded to take account for the physical realities of image making. The fundamental characteristics of Cartesian perspectivalism (flat, rectilinear, monoscopic images - as opposed to the stereoscopic, concave, saccadic nature of human vision) continued to dominate the majority of mass media through the 20th century and through to this day. It is only recently, through advances in virtual reality, augmented reality and 360-degree video, that we are starting to see a shift away from media that had inherited the characteristics of Cartesian perspectivalism.

Thus, for a project that is trying to expose the phenomenology of disembodied artificial perception, it is fitting to train it on media that indirectly inherits so much from the Cartesian ideas of subjective rationality. In addition to this, there could not be a more apt film to explore these themes with than *Blade Runner* (1982). The Ridley Scott adaption of the Phillp K. Dick novel *Do Androids Dream of Electric Sheep?* (1982 [1968]), which was one of the first novels to explore the themes of artificial subjectivity, and which repeatedly depicts eyes, photographs and other symbols alluding to perception.

The other film chosen to model for this project is *A Scanner Darkly* (2006) - another adaption of a Phillp K. Dick novel (2011 [1977]). This story also explores themes of the nature of reality, and is particularly interesting for being reconstructed with a neural network as every frame of the film has already been reconstructed (hand traced over the original film) by an animator.

Chapter 4

Method

4.1 Model Overview

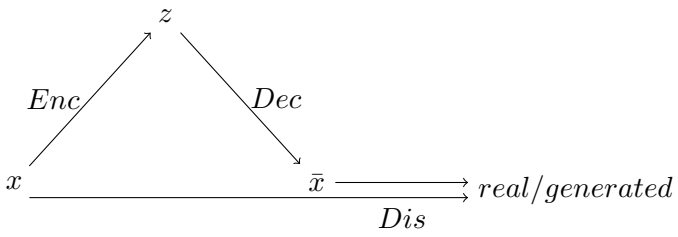

FIGURE 4.1: Overview of autoencoder model with discriminator network.

The model consists of three convolutional neural networks: the encoder Enc, the decoder Dec, and the discriminator Dis. The encoder Enc encoders a data sample x into a latent representation z. The decoder Dec decodes the latent representation back into data space to produce a generated sample $\bar{x}$. The encoder Enc and decoder Dec constitute the autoencoder.

$$z \sim Enc(x) = q(z|x), \quad \bar{x} \sim Dec(z) = p(x|z) \tag{4.1}$$

The discriminator Dis networks assigns probability $y = Dis(x) \in [0, 1]$ that x is a real data sample and probability $1 - y$ that $\bar{x}$ and x_p are generated data samples. The GAN objective of assigning the best possible discrimination between real and generated data samples is used the train the discriminator network Dec. The covariance between the discriminators Dec response to the real data sample x and reconstructed sample $\bar{x}$ is used as the reconstruction error used to train the encoder Enc and decoder Dec. A full description of the training procedure and various objective functions is detailed in section 4.3.

The model is trained using batch normalisation with a batch size of 12, significantly smaller than the batch size of 64 described by Larsen, Sønderby, and Winther (2015) (the learned similarity autoencoding paper). This is because those models were trained on images of size 64x64, whereas this model is trained on images of size 256x144. Therefore the batch size had to be reduced for the model to be within the memory restriction of the GPU. The model is trained with random mini-batches of frames from a

film (the training dataset) that have been scaled to 256x144. Once trained, the model processes mini-batches of video frames in order, and the reconstructed frames $\bar{x}$ are resequenced into a video using FFmpeg (Bellard, Niedermayer, et al., 2012). The model was implemented in Python using Google's machine learning framework TensorFlow (Abadi et al., 2015).

The image resolution of 256x144 was chosen as it was the largest resolution at a 16:9 aspect ratio that could be performed with a reasonable mini-batch size for batch normalisation. The aspect ratio of 16:9 was chosen as it is the standard aspect ratio for online video. The film *Blade Runner* was filmed at an aspect ratio of 2.35:1, the standard for widescreen cinema. As this is wider than 16:9, regrettably the frames had to be cropped to a 16:9 aspect ratio. In addition to this, the 2.35:1 aspect ratio does not scale well to dimensions of small integers needed to model images at high level layers in convolutional neural networks.

4.2 Network Architecture

The model consists of three convolutional neural networks. The encoder network Enc and the discriminator network Dis take a 4$^{\text{th}}$-order Tensor of size 256x144x3x12 as input, with the final layer of the decoder network Dec being the same dimensions. The final layer of the decoder network Dec being the same dimensions as the images in the training dataset, which have a resolution of 256x144 and have 3 colour channels: red, green and blue (RGB). The fourth dimension of the Tensor accounts for the fact that the model is being trained using batch normalisation in mini-batches of 12. All of the networks share the weight parameters and and bias parameters across the 4$^{\text{th}}$ dimension spanning the size of the mini-batch. Therefore the model is effectively a network to auto-encode 3$^{\text{rd}}$-order Tensors, i.e. modelling single images. The dimension of latent variables z is 200, smaller than the 2048 described by Larsen, Sønderby, and Winther (2015).

4.2.1 Encoder Network

The encoder network Enc has five layers. It takes as input a mini-batch of data samples x, 4$^{\text{th}}$-order Tensor of size 256x144x3x12. The first four layers are convolutional layers and the final layer represents the latent representation z prior to it being a function of noise ϵ as in the variational autoencoder re-parameterisation criterion (see section 2.2.2). Therefore the final layer of the encoder has two outputs, μ and σ, z is ascertained by the function $z = \mu + \sigma\epsilon$.

The four convolutional layers use strided convolutions (as opposed to the convolutions and pooling used in traditional conv-nets), using a convolution filter size of 5x5 and strides of 2 in both the x and y directions. As strides of 2 are being used, the x and y dimensions are half of the previous layer for each convolutional layer, reducing the x and y dimensions from the input size of 256x144 to a size of 16x9 in the last convolutional layer. The number of convolutional filters in the first convolutional layer is 80, this number

doubles in each layer with the last layer having 640 convolutional filters. The activation function used in the four convolution layers is the leaky rectified linear unit (*leakyReLU*, full description in section 4.2.4). This differs from the model described by Larsen, Sønderby, and Winther (2015) which uses rectified linear units in all convolutional layers.

The final convolutional layer in the encoder network Enc is reshaped (flattened) from a 4th-order Tensor of size 16x9x640x12 to a linear (ignoring the batch depth) 2nd-order Tensor of size 92160x12. The figure 92160 being the sum $16 \times 9 \times 640$. The final layer in the encoder network Enc is fully connected to the flattened convolutional layer. The final layer is a 2nd-order Tensor of size 200x12, 200 being the dimension of latent variables z. The activation function used in the final layer is the tanh function[1], this was found to work best in practice in this implementation. This differs from the ReLU activation function used in the final layer described by Larsen, Sønderby, and Winther (2015).

Architecture of encoder network

Layer	1st	2nd	3rd	4th	Layer Type	Activation Function	BNorm
Input	256	114	3	12	Sample x	-	-
1	128	72	80	12	Conv $\downarrow$	LeakyReLU	BNorm
2	64	36	160	12	Conv $\downarrow$	LeakyReLU	BNorm
3	62	18	320	12	Conv $\downarrow$	LeakyReLU	BNorm
4	16	9	640	12	Conv $\downarrow$	LeakyReLU	BNorm
5	200	12	-	-	Fully Conn	tanh	-

FIGURE 4.2: *Table 1.* Architecture of encoder network Enc including input data sample layer. $\downarrow$ represents convolutional down-sampling with strided convolutions. The network trained with batch normalisation with a batch size of 12.

4.2.2 Decoder Network

The decoder network Dec has five layers. The input is the latent representation z a 2nd-order Tensor of dimensions 200x12, with the latent representation z having a dimension of 200. This is projected (as this is a matrix multiplication this is essentially the same as being fully connected) onto a linear 2nd-order Tensor of dimensions 92160x12, that is reshaped to a 4th-order Tensor of dimensions 16x9x640x12. This first layer is used as the start of the convolution stack. This is the same procedure as is described for the generator in the DCGAN model (Radford, Metz, and Chintala, 2015).

The next four layers are fractially-strided transposed convolutions[2]. The x and y dimensions of these layers double at each stage with the final layer

[1] In the first build of this model, no activation function was used in the final layer. This led to the KL-divergence exploding at an unpredictable point, causing extremely large error derivatives being propagated through the encoder Enc, effectively stopping learning and eventually eroding all of the effective weights learned from training.

[2] Commonly reffered to as deconvolution in machine learning literature (Zeiler et al., 2010), but this is actually the transpose of a convolution, rather than an actual deconvolution.

being the dimension 256x144x3x12, the same dimension as the data samples. The first four layers in the network use rectified linear units as their activation functions (*ReLU*, full description in section 4.2.4), this is the same activation function used by Larsen, Sønderby, and Winther (2015) in their decoder. The final output layer uses the Tanh as the activation function which was shown by Radford, Metz, and Chintala (2015) to learn more quickly and to saturate and cover the colour space of the training distribution. Sampling the final output layer produces the mini-batch of generated data samples $\bar{x}$ and x_p.

Architecture of decoder network

Layer	1st	2nd	3rd	4th	Layer Type	Activation Function	BNorm
Input	200	12	-	-	Latent Rep z, z_p	-	-
1	16	9	640	12	Fully Conn	ReLU	BNorm
2	32	18	320	12	Conv $\uparrow$	ReLU	BNorm
3	64	36	180	12	Conv $\uparrow$	ReLU	BNorm
4	128	72	80	12	Conv $\uparrow$	ReLU	BNorm
5	256	144	3	12	Conv $\uparrow$	Tanh	-

FIGURE 4.3: *Table 2.* Architecture of decoder network Dec including input latent representation z. $\uparrow$ represents convolutional up-sampling with fractional-strided convolutions. The last layer 5 is the output and is sampled as an images. The network trained with batch normalisation with a batch size of 12.

4.2.3 Discriminator Network

The discriminator network Dec takes as its input a mini-batch of either real data samples x, or generated data samples $\bar{x}$ or x_p. The architecture of the discriminator network is almost identical to that of the encoder network Enc (see section 4.2.1), except the final convolutional layer is simply flattened and fed into a single sigmoid activation function, which assigns probability to the data sample being real or fake.

Architecture of discriminator network

Layer	1st	2nd	3rd	4th	Layer Type	Activation Function	BNorm
Input	256	114	3	12	Sample $x, \bar{x}, x_p$	-	-
1	128	72	80	12	Conv $\downarrow$	LeakyReLU	BNorm
2	64	36	160	12	Conv $\downarrow$	LeakyReLU	BNorm
3	62	18	320	12	Conv $\downarrow$	LeakyReLU	BNorm
4	16	9	640	12	Conv $\downarrow$	LeakyReLU	BNorm
5	1	12	-	-	Fully Conn	sigmoid	-

FIGURE 4.4: *Table 3.* Architecture of discriminator network Dis including input data sample layer. $\downarrow$ represents convolutional down-sampling with strided convolutions. The final convolutional layer is flattened and fed into a single sigmoid output. The network trained with batch normalisation with a batch size of 12.

4.2.4 Activation Functions

There are four activation functions used in this model: sigmoid, Tanh, ReLU and LeakyReLU. Definitions of ReLU and LeakyReLU are given in the following subsections. The sigmoid and Tanh activation functions are standard bounded nonlinear activation functions used in machine learning. The output of a node is calculated as the weighted sum of the inputs w, passed through one of these activation functions.

The sigmoid function is given as:

$$S(w) = \frac{1}{1 + e^{w}} \tag{4.2}$$

The Tanh function is given as:

$$tanh(w) = \frac{1 - e^{-2w}}{1 + e^{-2w}} \tag{4.3}$$

Rectified Linear Unit

The rectified linear unit *(ReLU)* is defined as:

$$f(w) = max(0, w) \tag{4.4}$$

It was first proposed by Nair and Hinton (2010) and is very common in deep learning, especially convolutional neural networks because of its efficiency. It is also argued to be more biological plausible that complex non-linear functions (Glorot, Bordes, and Bengio, 2011). In this model it is used in the most of the convolutional layers of the decoder network Dec.

Leaky Rectified Linear Unit

The leaky rectified linear unit *(LeakyReLU)* is very similar to the rectified linear unit but allows a small, non-zero gradient when the unit is not active. It was first proposed by Maas, Hannun, and Ng (2013). The function is defined as:

$$f(w) = \begin{cases} w, & \text{if } w > 0 \\ 0.01, & \text{if } w \leq 0 \end{cases} \tag{4.5}$$

In the model, this activation function is used in the convolutional layers of the encoder network Enc and discriminator network Dis.

4.3 Objective Functions

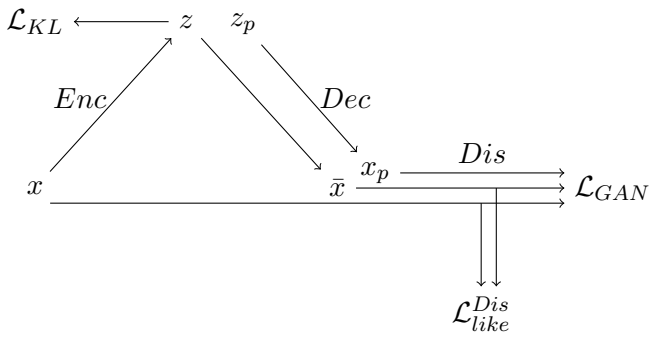

FIGURE 4.5: Overview of autoencoder model showing flow of data and points at which objective functions are calculated.

There are three objective functions: $\mathcal{L}_{KL}$, $\mathcal{L}_{GAN}$ and $\mathcal{L}^{Dis}_{like}$ that are calculated in the training procedure, these three objective functions are used in different in different combinations and propagated as the error derivatives through the three networks Enc, Dec and Dis.

4.3.1 Kullback-Leibler Divergence

The objective function $\mathcal{L}_{KL}$ is the Kullback-Leibler divergence that is used to impose a strict prior on the distribution over the latent variables $p(z)$, the is used as one of the components of the objective function in the training criterion of a variational autoencoder (see section 2.2.2). In this case the prior is a centred isotropic multivariate Gaussian:

$$p(z) = \mathcal{N}(0, I) \tag{4.6}$$

The Kullback-Leibler divergence measures the difference between the prior distribution of latent variables $p(z)$ and the approximate posterior distribution of latent variables $q(z|x)$ generated by the encoder network Enc given data sample x.

$$\mathcal{L}_{KL} = D_{KL}(q(z|x)||p(z)) \tag{4.7}$$

Following the reparameterization trick described by Kingma and Welling (2013), where the latent variable z is reparameterized as a deterministic function of noise ϵ (see section 2.2.2), the Kullback-Leibler divergence is calculated using the equation:

$$\mathcal{L}_{KL} = \frac{1}{2}\sum_{i=1}^{N} 1 + log(\sigma_i^2(x)) - \mu_i^2(x) - \sigma_i^2(x) \tag{4.8}$$

4.3.2 GAN Objective

The objective function $\mathcal{L}_{GAN}$ is the GAN objective (generative adversarial networks, see section 2.3). The discriminator network Dis observes real data samples x and generated data samples $\bar{x}$ and x_p. The GAN objective is for the discriminator network to learn the best possible binary classifier to discriminate between real and generated data samples. This helps force the decoder Dec to fit the true data distribution. This is calculated by maximising the binary cross entropy for real data samples and minimising the binary cross entropy for generated data samples:

$$\mathcal{L}_{GAN} = \log(Dis(x)) + \log(1 - Dis(\bar{x})) + \log(1 - Dis(x_p)) \tag{4.9}$$

4.3.3 Discriminator Covariance

The objective function $\mathcal{L}_{like}^{Dis}$ is the covariance between the discriminators response to real data samples $Dis(x)$ and reconstructed data samples $Dis(\bar{x})$. This objective function is used to replace the pixel-wise reconstruction cost that is used in traditional autoencoders. This objective function is the learned similarity measure (Larsen, Sønderby, and Winther, 2015) that drastically improves the performance of modelling natural images compared to variational autoencoders.

To calculate $\mathcal{L}_{like}^{Dis}$, the hidden representation of layer l of the discriminator network given a real data sample $Dis_l(x)$ is compared to the discrinators response given the reconstructed data sample $Dis_l(\bar{x})$. This is done using a Gaussian observation model for $Dis_l(x)$ with a mean of $Dis_l(\bar{x})$ and identity covariance[3]:

$$p(Dis_l(x)|z) = \mathcal{N}(Dis_l(x)|Dis_l(\bar{x}), I) \tag{4.10}$$

In this implementation the observation model is performed for the 4 convolutional layers of the discriminator network Dis but not the final sigmoid layer. The mean of these values is taken by summing them and dividing by the number of observed layers n, which in this case is 4:

$$\mathcal{L}_{like}^{Dis} = \frac{1}{n}\sum_{l=1}^{n}\mathcal{N}(Dis_l(x)|Dis_l(\bar{x}), I) \tag{4.11}$$

Calculating the mean of the covariance of several layers is an extension to the method described by Larsen, Sønderby, and Winther (2015) and is performed as it improves the capability of the decoder Dec to represent the dataset over multiple levels of representation.

[3] In practice this is simply calculated as being the mean squared error (MSE).

4.4 Respective Error Gradients

Each of three networks in the model is trained with different combinations of the three objective functions. Once the network parameters θ_{Enc}, θ_{Dec} and θ_{Dis} have been initialised, a forward pass of the network is performed from a mini-batch x. In the backward pass the parameters of the networks are updated according to the respective error gradients. The parameters of the discriminator θ_{Dis} is simply updated using the GAN objective function:

$$\theta_{Dis} \leftarrow \nabla_{\theta_{Dis}}(\mathcal{L}_{GAN}) \tag{4.12}$$

The parameters of the encoder network θ_{Enc} are updated using the sum of the KL-divergence $\mathcal{L}_{KL}$ and the discriminator covariance $\mathcal{L}_{like}^{Dis}$:

$$\theta_{Enc} \leftarrow \nabla_{\theta_{Enc}}(\mathcal{L}_{KL} + \mathcal{L}_{like}^{Dis}) \tag{4.13}$$

The parameters of the decoder network θ_{Dec} are updated using the weighted combination of the discriminator covariance $\mathcal{L}_{like}^{Dis}$ and the GAN objective $\mathcal{L}_{GAN}$. This is weighted using the parameter γ which in the case of this implementation a value of 2.5 was found to be the most stable. Larsen, Sønderby, and Winther (2015) interpret this is as weighting the *content* and *style* of the generated samples:

$$\theta_{Dec} \leftarrow \nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{like}^{Dis} - \mathcal{L}_{GAN}) \tag{4.14}$$

4.5 Training Procedure

To begin with, the parameters for the three networks θ_{Enc}, θ_{Dec} and θ_{Dis} are initialised. The dataset F (all of the frames from a film) is loaded into the software. As the all the frames of the dataset F are in sequence, if you were to sample mini-batches in order from this dataset, the model would effectively be learning an averaged image over the number frames in the batch size. Therefore the frames in the dataset F are randomly shuffled to give the randomised dataset F_r. Mini-batches of data samples x are sequentially sampled from the dataset F_r, performing a forward and backward pass through the model until one epoch is complete. Then parameters may be changed (see Fine Tuning - 4.5.1) and more epochs of the dataset F_r can be performed.

A mini-batch of real data samples x is loaded into the model, the encoder Enc encodes x into a latent representation z. The objective function $\mathcal{L}_{KL}$ is obtained by calculating the KL-divergence of the approximate posterior $z \sim q(z|x)$ from the prior distribution $p(z) = \mathcal{N}(0, I)$. The decoder Dec reconstructs the mini-batch of data samples from the latent representation z to produce the reconstructed data samples $\bar{x}$. In addition to this random latent variables are sampled from the prior distribution z_p and the decoder Dec decodes z_p to create the generated data samples x_p. The discriminator Dis processes the real data samples x and the generated samples $\bar{x}$ and

x_p. The binary cross entropy of the decoders response to the data samples is calculated and combined to calculate the GAN objective function $\mathcal{L}_{GAN}$. The covariance of the four hidden convolutional layers in the discriminator response to the real data samples x and reconstructed data samples $\bar{x}$ is calculated to ascertain the image similarity objective function $\mathcal{L}_{like}^{Dis}$. The parameters for the three networks θ_{Enc}, θ_{Dec} and θ_{Dis} are then updated using their respective error gradients (section 4.4) using the *Adam* optimiser (Kingma and Ba, 2014) with a momentum of 0.5 and a learning rate of 0.0002. This is repeated until every mini-batch from the randomised dataset F_r has been processed (1 epoch).

Algorithm 1 Training procedure

$F \leftarrow$ Load dataset of film frames
$F_r \leftarrow$ Randomise order of frames
θ_{Enc}, θ_{Dec} and $\theta_{Dis} \leftarrow$ initialise network parameters
repeat
  $x \leftarrow$ mini-batch of data samples from F_r
  $z \leftarrow Enc(x)$
  $\mathcal{L}_{KL} \leftarrow D_{KL}(q(z|x)||p(z))$
  $\bar{x} \leftarrow Dec(z)$
  $z_p \leftarrow$ samples from prior $p(z) = \mathcal{N}(0, I)$
  $x_p \leftarrow Dec(z_p)$
  $\mathcal{L}_{GAN} \leftarrow \log(Dis(x)) + \log(1 - Dis(\bar{x})) + \log(1 - Dis(x_p))$
  $\mathcal{L}_{like}^{Dis} \leftarrow \frac{1}{n}\sum_{l=1}^{n} \mathcal{N}(Dis_l(x)|Dis_l(\bar{x}), I)$
  Update network parameters according to respective gradients:
  $\theta_{Enc} \leftarrow \nabla_{\theta_{Enc}}(\mathcal{L}_{KL} + \mathcal{L}_{like}^{Dis})$
  $\theta_{Dec} \leftarrow \nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{like}^{Dis} - \mathcal{L}_{GAN})$
  $\theta_{Dis} \leftarrow \nabla_{\theta_{Dis}}(\mathcal{L}_{GAN})$
until The defined number of epochs of F_r is completed

4.5.1 Fine Tuning

After training using the algorithm described, then running the network and re-sequencing the film the network was trained on. It was discovered that the model collapses large periods of similar frames (i.e a fixed camera shot of a character talking) into one representation; usually the most common frame (thus giving long sequences of an actors face not blinking or making facial gestures). This is unsurprising as the distribution of a dataset F of film frames is likely to be a lot more skewed and concentrated on a lot of very similar frames, than say a hand picked evenly distributed dataset of images of one subject matter.

One possible cause of this problem (similar frames collapsing into one representation) was that the magnitude of the noise ϵ - which the latent representation z is a deterministic function of: $z = \mu + \sigma\epsilon$ - was too high. In the original build ϵ was sampled from a normal distribution with a mean μ of 0 and a standard deviation σ of 1: $\epsilon \sim \mathcal{N}(0, 1)$. Results were improved by running the network with noise ϵ having a standard deviation σ of 0.25: $\epsilon \sim \mathcal{N}(0, 0.25^2)$.

Results were improved further by gradually reducing the standard deviation σ of the noise ϵ after each epoch. This allows the encoder Enc to efficiently shape and manipulate the distribution of latent variables z in the early stages of training, while allowing the decoder Dec to learn fine-grained differences between similar frames in the latter stages of training. Reducing to propensity for the system to collapse long sequences into one representation, ensuring there is continuous variation in the resequenced video.

This addition to the training procedure is one of the novel contributions of this project.

4.6 Running The Model

Once the model has been trained, sequential mini-batches x from the ordered training dataset F or an alternative dataset of sequentially ordered video frames F_{alt} can be used. The encoder Enc encodes the frames into the latent representation z. In this case no noise ϵ is added to the latent representation z[4], although a very small amount (with a standard deviation σ of something like 0.0001) may be added to inject some variation to the reconstructed video sequence. The decoder Dec decodes the latent representation z to produce the reconstructed frames $\bar{x}$. The discriminator Dis is not used when the model is not training.

Once one epoch of the ordered dataset F or F_{alt} is complete, the complete set of reconstructed outputs are saved into an external directory as the new ordered and reconstructed set $\bar{F}$ or $\bar{F}_{alt}$. These are then resequenced into a video using FFmpeg (Bellard, Niedermayer, et al., 2012).

[4]This is performed by setting the standard deviation σ of noise ϵ to 0, thus $z = \mu + \sigma\epsilon$ simply equals μ with is the same as the encoder Enc output simply being z.

Chapter 5

Results

This section gives examples of samples during the training of the model trained on *Blade Runner* (5.1) and *A Scanner Darkly* (5.2) and their final reconstructed video sequences. Section 5.3 show samples from reconstructions of other videos ran the trained models. Links to download all of the reconstructed videos can be found in Appendix A.

5.1 Blade Runner

The model was trained on the film *Blade Runner* (1982) for 6 epochs. The training regime using the fine tuning algorithm (gradually reducing noise ϵ) was as follows:

Parameters for training model on *Blade Runner*

Epoch	Standard Deviation σ of Noise ϵ
1	0.25
2	0.25
3	0.01
4	0.01
5	0.05
6	0.05

The sides of the film *Blade Runner* was cropped to give the correct aspect ratio 16:9. The credit sequences at the beginning and end of the film were trimmed from the film, as in early experiments it was apparent that modelling the distribution of images of high contrast detailed text alongside varied natural images was putting too much strain on the model. Results were significantly improved by removing the credit sequences from the training dataset and this also stabilised training.

The trimmed film was converted into a dataset of still images at the resolution 256x144 using FFmpeg (Bellard, Niedermayer, et al., 2012). The trimmed film (1:h52m) resulted in a dataset with 157224 individual frames. With a batch size of 12 this gives 13102 mini-batches to be processed for one epoch.

It takes about 13 seconds to process one mini-batch during training on a 4GB NVIDIA GTX 960 graphics card. So to train for 6 epochs of *Blade Runner* took approximately 283 hours or 11 days and 20 hours.

5.1.1 Samples From Training

The model exports the mini-batch x, the reconstructed samples $\bar{x}$ and the generated samples x_p (from random latent variables z_p) at regular intervals. Examples of these samples over the course of the training are given in this section.

1st Epoch

FIGURE 5.1: Samples from training on *Blade Runner* after 1 epoch. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

2nd Epoch

FIGURE 5.2: Samples from training on *Blade Runner* after 2 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\tilde{x}$. Bottom: random samples x_p.

3rd Epoch

FIGURE 5.3: Samples from training on *Blade Runner* after 3 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

4th Epoch

FIGURE 5.4: Samples from training on *Blade Runner* after 4 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\tilde{x}$. Bottom: random samples x_p.

5th Epoch

FIGURE 5.5: Samples from training on *Blade Runner* after 5 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

6th Epoch

FIGURE 5.6: Samples from training on *Blade Runner* after 6 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\tilde{x}$. Bottom: random samples x_p.

5.1.2 Reconstructed Film

After training, one epoch through the film *Blade Runner* is performed without noise ϵ and without training. It takes about 2 seconds to process one mini-batch when not training. Therefore it took approximately 7 hours 30 minutes to reconstruct every frame in the film.

FIGURE 5.7: Real and generated samples from the first half of *Blade Runner* in steps of 4000 frames. Alternating real then reconstructed samples.

FIGURE 5.8: Real and generated samples from the second half of *Blade Runner* in steps of 4000 frames. Alternating real then reconstructed samples.

5.2 A Scanner Darkly

The model was also trained on the film *A Scanner Darkly* (2006) for 6 epochs. The training regime using the fine tuning algorithm (gradually reducing noise ϵ) was as follows:

Parameters for training model on *A Scanner Darkly*

Epoch	Standard Deviation σ of Noise ϵ
1	0.25
2	0.1
3	0.1
4	0.05
5	0.05
6	0.05

Unlike *Blade Runner*, *A Scanner Darkly* was filmed at a aspect ratio of 16:9. The credit sequences at the end of the film were trimmed from the film.

The trimmed film was converted into a dataset of still images at the resolution 256x144 using FFmpeg (Bellard, Niedermayer, et al., 2012). The trimmed film (1:h35m) resulted in a dataset with 137871 individual frames. With a batch size of 12 this gives 11489 mini-batches to be processed for one epoch.

It takes about 13 seconds to process one mini-batch during training on a 4GB NVIDIA GTX 960 graphics card. So to train for 6 epochs of *A Scanner Darkly* took approximately 249 hours or 10 days and 9 hours.

5.2.1 Samples From Training

The model exports the mini-batch x, the reconstructed samples $\bar{x}$ and the generated samples x_p from random latent variables z_p at regular intervals. Examples of these samples over the course of the training are given in this section.

1st Epoch

FIGURE 5.9: Samples from training on *A Scanner Darkly* after 1 epoch. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

2nd Epoch

FIGURE 5.10: Samples from training on *A Scanner Darkly* after 2 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

3rd Epoch

FIGURE 5.11: Samples from training on *A Scanner Darkly* after 3 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

4th Epoch

FIGURE 5.12: Samples from training on *A Scanner Darkly* after 4 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

5th Epoch

FIGURE 5.13: Samples from training on *A Scanner Darkly* after 5 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

6th Epoch

FIGURE 5.14: Samples from training on *A Scanner Darkly* after 6 epochs. Top: mini-batch of training data x. Middle: reconstructed samples $\bar{x}$. Bottom: random samples x_p.

5.2.2 Reconstructed Film

After training, one epoch through the film *A Scanner Darkly* is performed without noise ϵ and without training. It takes about 2 seconds to process one mini-batch when not training. Therefore it took approximately 6 hours 38 minutes to reconstruct every frame in the film.

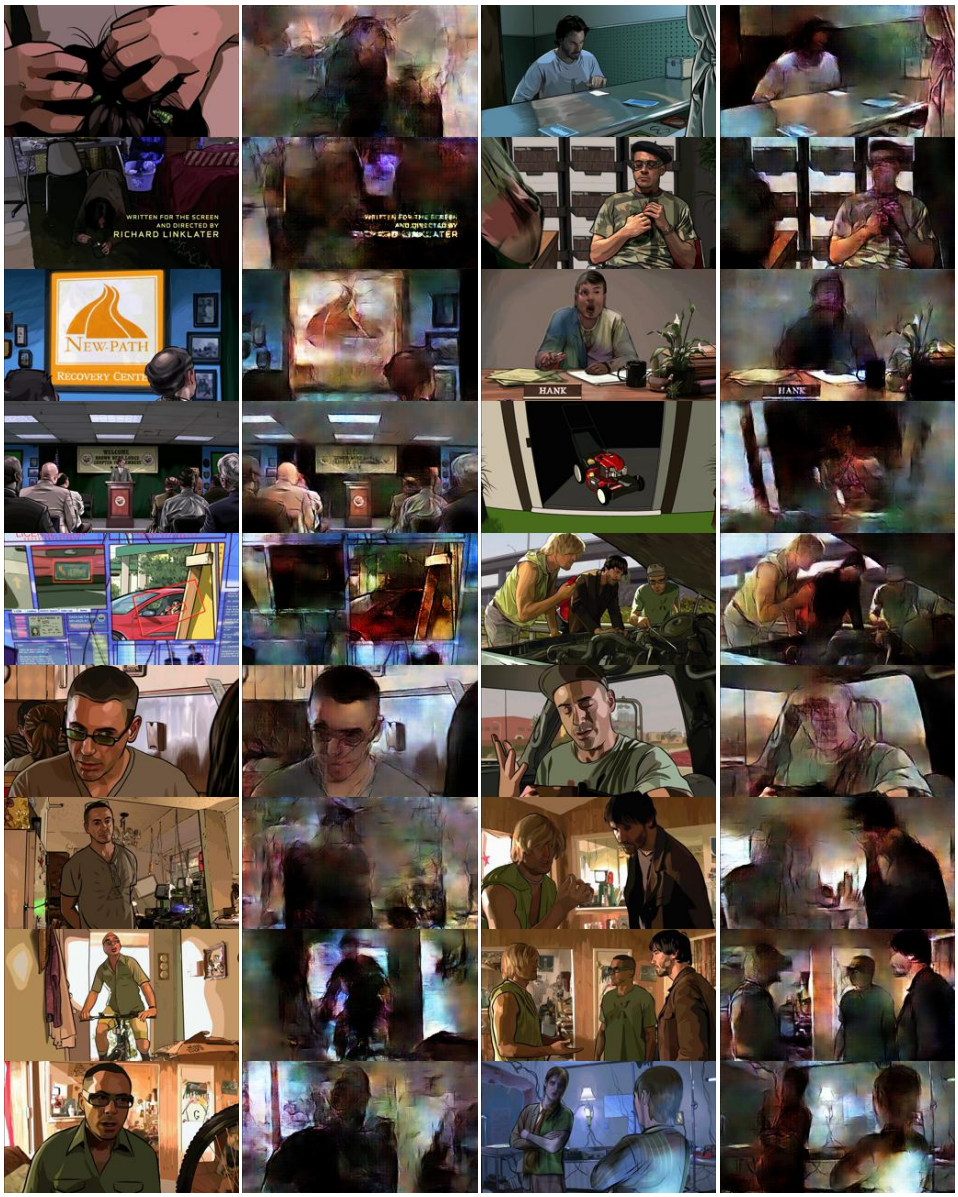

FIGURE 5.15: Real and generated samples from the first half of *A Scanner Darkly* in steps of 4000 frames. Alternating real then reconstructed samples.

FIGURE 5.16: Real and generated samples from the second half of *A Scanner Darkly* in steps of 4000 frames. Alternating real then reconstructed samples.

5.3 Feeding Other Videos Through The Models

As well as reconstructing the training videos, one of the goals of this project was to experiment with feeding other video through the models to see how this model fairs in generating new material, and to explore how the model *reinterprets* other films. The next two sub-sections show *Blade Runner* ran through *A Scanner Darkly* and *A Scanner Darkly* ran through *Blade Runner* respectively. The following two sub-sections show two further videos ran through the model trained on *Blade Runner*.

5.3.1 Blade Runner Through A Scanner Darkly

FIGURE 5.17: Real and generated samples from *Blade Runner* ran through the model trained on *A Scanner Darkly* in steps of 8000 frames. Alternating real then reconstructed samples.

5.3.2 A Scanner Darkly Through Blade Runner

FIGURE 5.18: Real and generated samples from *A Scanner Darkly* ran through the model trained on *Blade Runner* in steps of 7500 frames. Alternating real then reconstructed samples.

5.3.3 1984 Apple Advert Through Blade Runner

The famous Apple *1984* Super Bowl advert that was made to introduce the Macintosh was directed by Ridley Scott (1984) who also directed *Blade Runner* (1982). It is believed that it was the success of *Blade Runner* that motivated Steve Jobs (the former co-founder and CEO of Apple Computer Inc) to commission Ridley Scott to direct the advert for a record breaking expense (Gere, 2002, 2015). The visual aesthetic of the *1984* advert has a lot in common with *Blade Runner* so it was a good choice for testing the model with a video of a similar style.

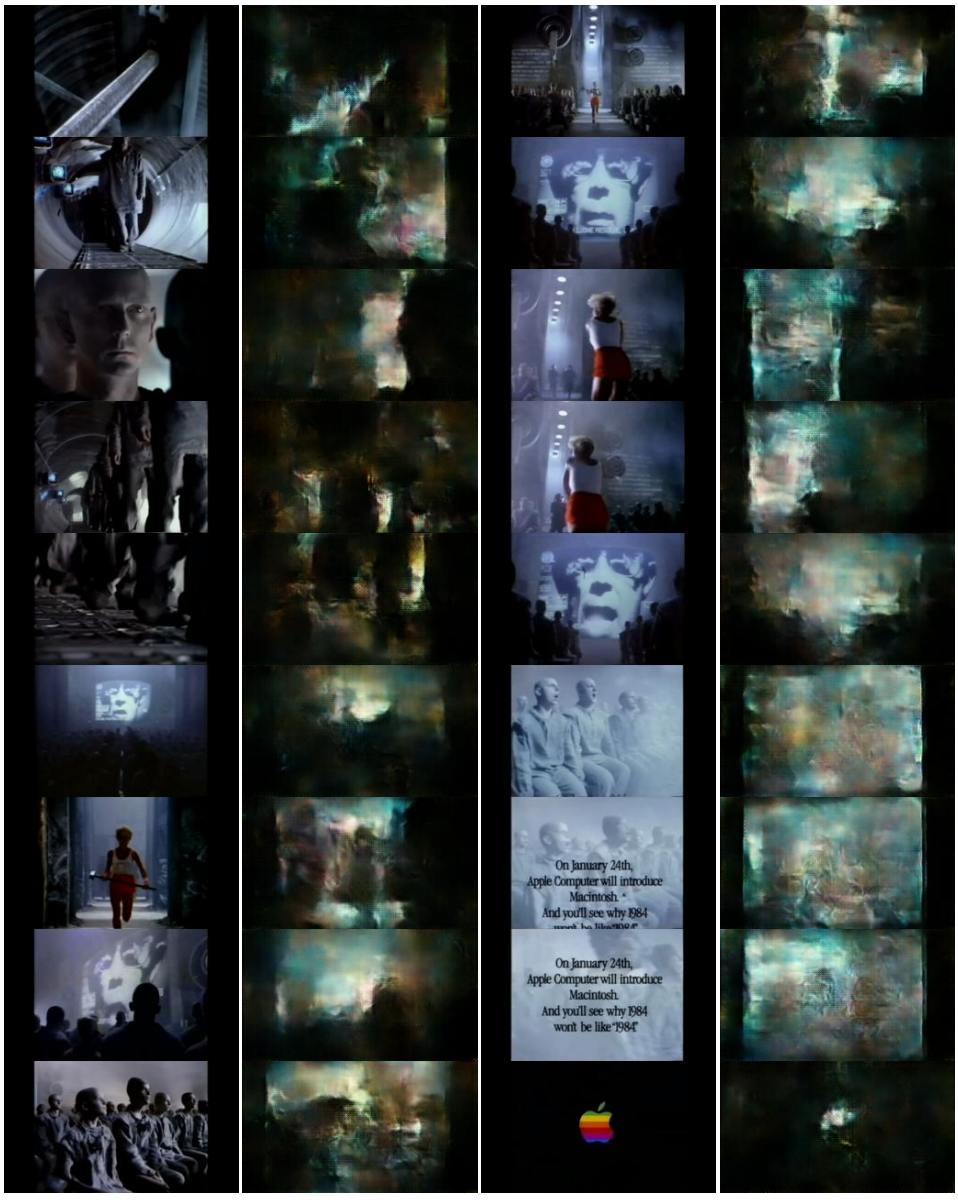

FIGURE 5.19: Real and generated samples from the Ridley Scott (1984) directed *1984* Apple Macintosh advert ran through the model trained on *Blade Runner* in steps of 100 frames. Alternating real then reconstructed samples.

5.3.4 Matrix III Through Blade Runner

John Whitney was a pioneer of computer graphics and animation. From the 1960's through to the 1990's Whitney made ground breaking and influential computer generated animations. The film Matrix III (1972) was one in a series of films that was demonstrating and exploring harmonic progression in animation. This film was chosen the run through the *Blade Runner* model to see wether theses harmonic progressions would translate to the reconstructed samples.

FIGURE 5.20: Real and generated samples from John Whitney's *Matrix III* (1972) ran through the model trained on *Blade Runner* in steps of 1000 frames. Alternating real then reconstructed samples.

Chapter 6

Evaluation

Assessing the effectiveness of generative models is difficult and problematic for high resolution natural images. Theis, Oord, and Bethge (2015) give a well-balanced account of the problems faced in trying to objective compare the quality of generative models. Their recommendation is that in the case of image synthesis, a subjective evaluation is the most appropriate means for assessing the effectiveness of a generative model. As this model was built to model high resolution non-square images[1] I unfortunately did not have time to restructure the model for smaller, square images in order to make a direct comparison to other models by training on the same datasets. I have however given a self contained qualitative assessment of the models effectiveness in the following section. I also regret that I was not able to do a systematic analysis of different configurations of training parameters, activation functions and optimisers; but due to the long times taken to train these models (the best part of two weeks for both films), this was not possible either.

The stated aim of this project however, was not *necessarily* to make the best generative model - although I did set out to implement the best available published model - but to take an existing state of the art model, implement it, and to apply train it on film frames at the highest resolution possible with currently available commercial technology. I feel that I largely achieved that aim. In section 6.3 I reflect on the reconstructed films as artworks and the potential for this method to become a new technique for experimental film making. Section 6.2 reflects on some of the challenges in implementing the model so that training was carried out in an efficient and stable manner.

6.1 Qualitative Assessment

Given that the current state of the art for generative models can only effectively model a restricted distribution of natural images (i.e. of class of object from one perspective), and bearing in mind this model is representing much higher resolution images, this model does a reasonably good job of modelling such a diverse set of images from the training data. The model obviously does a much better job of reconstructing images from the training dataset than from other datasets; individual frames reconstructed from

[1] High resolution compared to state of the art published results from other generative models; 256x144 compared to 64x64 (Radford, Metz, and Chintala, 2015; Larsen, Sønderby, and Winther, 2015).

films other than the training dataset are difficult to make out on their own, but when resequenced into a video they are temporally coherent and make interesting viewing. Results of reconstructing non-training dataset videos would obviously be greatly improved by having a much bigger and more diverse set of images in the training dataset but in turn this would remove a lot of the artistic meaning from the videos, and any artistic style inherent in the film that the model implicitly represents would be washed out as more videos were added to the training dataset. Maybe the model could be pre-trained on a very large, diverse set of training data from many films, then finally pre-trained for a shorter time on one particular film - as Alex Graves (2013) did with his handwriting generation model - but unfortunately this could not be done because of the excessively long time required to train the model on hundreds of hours of video.

6.1.1 Reconstructing the Test Dataset

The model does a reasonably good job at reconstructing most of the frames from the films. The resequenced videos are coherent and can be followed easily for the majority of their duration. The models do a good job of reconstructing prolonged scenes that are static and have a high contrast, this is not surprising as the model is effectively trained on frames from a static prolonged scene many more times than scenes where the camera position is moving. Traditionally this would be regarded as overfitting, but since the model is intentionally trained on a skewed and unevenly distributed dataset this is not a great cause for concern. The model struggles with dark, low-contrast scenes; especially when there is one are where there is a lot of variation frame to frame (i.e. an animated face that is moving), and curiously the models are unable to reconstruct completely black frames (see figure 6.1) even though there are some examples in the training datasets (this could surely be rectified by simply adding more black frames to the training dataset). The model also struggles to reconstruct faces when there is a lot of variation frame to frame, and has a tendency to collapse prologued static shots where this is some variation into one single representation. These limitations are described in detail the following sub-sections.

FIGURE 6.1: Failure of both models to reconstruct a completely black input. Left: Input. Middle: Reconstruction with the *Blade Runner* model. Right: Reconstruction with the *A Scanner Darkly* model.

Reconstructing Faces

Perhaps shortcomings in the reconstruction of faces are simply more noticeable to the naked eye than the reconstruction of other kinds of scenes; but

this is an important limitation, given that a large proportion of the scenes in these films are close-ups of people conversing. If the actors face is fixed over a long period without much variation the model can represent it. But if there is a lot of variation in a fixed shot - for instance if a character rotates or moves their head (see figure 6.2) - the model seem to struggle.

FIGURE 6.2: Samples from *Blade Runner* showing the transiting of the character Rachael rotating her head. Sampled in increments of 14 frames. Top: Samples from the film. Bottom: Reconstructed Samples.

This is a problem in the *Blade Runner* model but is even more pronounced in the model trained on *A Scanner Darkly* (see figure 6.3). This problem is probably exacerbated by the fact that all of the facial features are outlined (because every frame is traced by hand) and very high contrast. Thus the model attempts to render high contrast outlined facial features, but the variation and complexity of facial structures is too difficult for the model to reproduce.

FIGURE 6.3: Samples from *A Scanner Darkly* showing the transiting of the character Charles Freck changing facial expressions. Sampled in increments of 15 frames. Top: Samples from the film. Bottom: Reconstructed Samples.

Ultimately this is not that surprising. The objective function of the model is skewed toward simply matching the spatial distribution of colour and brightness in a given frame, not producing a convincing representation of classification of object. The original learned similarity autoencoding paper by Larsen, Sønderby, and Winther (2015) produced much better results of faces (see figure 2.10) but they were training on a dataset of faces that were all cropped, aligned and taken from the same head-on perspective. The only known way to rectify this problem would be to combine a generative model with spatial transformer networks (Jaderberg, Simonyan, Zisserman, et al., 2015) that allow a model to be invariant to translation, scale, rotation and warping. Rezende et al. (2016) demonstrated the power of combining spatial transformer networks with a generative model (see figure **??**), but this

would require a radically different and more complex approach to the generative model and would probably not be suitable for modelling the frames from an entire film as this approach is designed for one-shot generalisation of one particular class of object representation.

Collapsing Representations

FIGURE 6.4: Samples from *Blade Runner* showing a long sequence in the film that has been collapsed into one representation by the model. Sampled in increments of 75 frames (3 seconds apart). Top: Samples from the film. Bottom: Reconstructed Samples.

One of the shortcomings of the project that becomes apparent when the reconstructed films are viewed in full, is that the model has a tendency to collapse long sequences where the shot is static into one extended representation. This is obviously not a great problem when the sequence is just a shot of a static landscape in the film, but the model quite often does this when there is still some action in the scene. The model will either collapse the scene into a representation where an actors motion is blurred out into some kind of *mean* of their action, or sometimes just represents the most common pose (i.e. the *mode* of their poses). This problem was particularly prevalent in early experiments, but gradually reducing the amount of noise ϵ injected into the latent representation z over the course of training (see Fine Tuning - 4.5.1) greatly reduced the models tendency to do this[2]. In theory, this is because if there is too much noise ϵ injected into the latent representation z, it is impossible for the decoder to learn to model fine grained differences between similar frames. The models were never trained without any noise (this was a deliberate decision as training with noise is what allows variational autoencoders to learn so efficiently) but perhaps this might help in the latter stages of fine-tuned training.

A potential solution to this problem would be to *somehow* alter the objective function to deter the decoder from constructing the same representation for slightly different frames, but it isn't clear what the best way to do this would be. Perhaps the discriminator would be given labels containing sequential

[2]Evidence of this can been seen in the last link to a video in Appendix A that shows an early test which has not been trained with the fine tuning approach and tendency for the model to collapse representations is even more extreme (there is however a lot of noise in the video as that model was built without the ability to remove noise from the latent representation when not training).

information, or that the image similarity objective function (see Discriminator Covariance - 4.3.3) would not only compare the similarity between the real frame and reconstructed frame, but would compare previous and subsequent frames to asses whether the model is sufficiently modelling variation frame to frame. This however, would require quite a radical rethinking of the training procedure, and it is not immediately obvious how this would be performed at the same time as batch normalisation.

6.1.2 Reconstructing Alternative Datasets

Not surprisingly, the model does not reconstruct other videos nearly as well as it reconstructs the films used as the training datasets. It is difficult to recognise what is being represented in any individual frame viewed alone, but viewed along side the real frame the model is attempting to reconstruct - it is obvious that the model is generating something that is somewhat sensible. The resequenced videos are temporally coherent and it is possible to get a vague sense of what might be going on. In addition to that the reconstructed videos contain a lot of - albeit unintelligible - interesting and complex structures that capture the cinematic style of the films they were trained on (granted both *Blade Runner* and *A Scanner Darkly* are visually very distinctive). In terms of representing style, this is not nearly as successful as the recently published artistic style transfer for videos paper (Ruder, Dosovitskiy, and Brox, 2016); but that method is not strictly a generative model, and is only for transferring the style from one image onto a video, not for modelling a large distribution of images.

6.2 Efficiency and Stability

Regarding the efficiency of the implementation, it is quite difficult to assess how efficient it is as there are only two implementations of the same model, both implemented in frameworks I am unfamiliar with. Comparing implementations would most likely just be comparing the efficiency of the machine learning frameworks rather than the design of the implementations themselves. However, generally speaking TensorFlow does a good job of implementing models efficiently. As a programmer, you define the graph (model) in Python, then TensorFlow constructs an executes the graph in well defined and efficient C++ backend. Training the model was made more efficient when the optimiser was not sampling generated images (by extracting the Tensors and converting them to NumPy arrays) after every mini-batch, but only on regularly spaced intervals.

Getting the model to train in a consistent and stable manner was an incredibly difficult and testing ordeal. Training generative adversarial networks is a notoriously difficult thing to get right as the rate of learning for both the generator and the discriminator need to be finely balanced so they learn in step with each other. One of the recurring problems that plagued the development of this implementation was that after a potentially very long time of training well - like 8 hours for instance - one of the error gradients for one of the three networks would suddenly explode to a very large number, washing away all of the nuanced weights the network had learned. The network whose error gradient exploded would then never recover, effectively making further training of that network and the other two networks redundant.

An early attempt to fix the exploding gradient problem was to clip the error gradients to a fixed range (i.e. 0.00000001 - 100), this improved the stability of training, but ultimately did not rectify the problem[3]. This problem was rectified for the KL-divergence objective function $\mathcal{L}_{KL}$ by using the tanh activation function on the final layer of the encoder. For the GAN loss $\mathcal{L}_{GAN}$ however, the loss can still suddenly spike (when it gets every real/generated prediction wrong, which is very unusual) and not recover. When it did spike and didn't recover, one of the problems was the GAN loss was significantly higher than the learned similarity $\mathcal{L}_{like}^{Dis}$ so the with the error gradients clipped, the decoder gradient $\nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{like}^{Dis} - \mathcal{L}_{GAN})$ becomes zero and both networks stop learning. The only solution to this problem was allow the decoder error gradient to be negative, while continuing to restrict the encoder and discriminator error gradients to positive values. This is unfortunate because what ends up happening is that if the decoder suddenly gets very good at fooling the discriminator, some of that learning immediately gets undone. However this was ultimately unavoidable as this was the only way to get all networks to train in a harmonious, consistent and stable manner.

[3] Clipping the values in the output of the discriminator to [0.00000001 - 1] prior to performing the binary cross entropy did prevent the GAN objective function $\mathcal{L}_{GAN}$ exploding massively and causing a NaN error, as taking the log of 0 is not defined.

6.3 Artistic Evaluation

6.3.1 As A Method For Film-Making

Reconstructing videos that were not used as the training dataset certainly yields interesting results. The reconstructed frames may not be recognisable as being the same subject as the original samples, but when resequenced the reconstruct videos are complex, coherent and fruitful in repeatedly surprising ways. It can be safely assumed that with more training on much bigger and more diverse datasets the results in reconstructing unseen video samples would be improved. However when trained on solely one film, the model personifies the aesthetic qualities that is inherent in the distribution of images that make up one sole film, which is aesthetically and conceptually meaningful in its own right.

6.3.2 As An Artwork

The videos reconstructed from the models that they were trained on are - in my subjective opinion - very successful artworks. There are flaws to the reconstruction (which could probably be improved with more training and/or small modifications to the training procedure), but from a *conceptual* and aesthetic point of view these flaws are in no way problematic. In fact, they help to expose the models mechanics, limitations and implicit assumptions. The reconstruction of *Blade Runner* in particular, gives a great insight into the phenomenology of the models subjective, artificial perception.

Chapter 7

Conclusion

Despite having spent the majority of this year planning to implement an existing autoencoder model, and *then* to extend the model in a novel way to either do sequence prediction or to model *truly* high resolution images; I am still satisfied with the outcome of the project. This project is a novel application of an autoencoder, and has the potential to be the basis for a lot of exciting and creative work.

Prior to this year I had a very limited understanding of deep learning, I would be lying if I said it hasn't been extremely challenging wrapping my head around all of the theory and underlying mathematics (Linear algebra, Bayesian statistics) that I was previously unfamiliar with. Ultimately though, it has been very rewarding and I feel extremely lucky to be researching this field at an exciting time of such rapid development.

The results from training on and reconstructing the films *Blade Runner* and *A Scanner Darkly* significantly exceeded my expectations, it is quite remarkably how diverse the range of images are that can be faithfully reconstructed by the model. Obviously I had hoped to perform the reconstructions at a much higher resolution, but at the end of the day they still make for compelling viewing, and that - ultimately - is what is most important.

Appendix A

Links To Download Reconstructed Videos

Blade Runner reconstructed after 6 epochs of training: https://www.dropbox.com/s/wxlsqazfknluru2/Blade_Runner_Full_Reconstruction_6_Epochs.mp4?dl=0

A Scanner Darkly reconstructed after 6 epochs of training: https://www.dropbox.com/s/2uh0ia4quoahc13/A_Scanner_Darkly_Full_Reconstruction_6_Epochs.mp4?dl=0

Blade Runner reconstructed with the *A Scanner Darkly* model: https://www.dropbox.com/s/hsvyo9q1cbh0euf/Blade_Runner_Through_A_Scanner_Darkly.mp4?dl=0

A Scanner Darkly reconstructed with the *Blade Runner* model: https://www.dropbox.com/s/9afn7uysn3tsmxt/A_Scanner_Darkly_Through_Blade_Runner.mp4?dl=0

1984 Apple Macintosh advert reconstructed with the *Blade Runner* model: https://www.dropbox.com/s/3bbh93mqa91fybl/Apple_1984_Through_Blade_Runner.mp4?dl=0

Matrix III reconstructed with the *Blade Runner* model: https://www.dropbox.com/s/ywpntbyik958al2/Matrix_III_Through_Blade_Runner.mp4?dl=0

An early test of a reconstruction from a model trained on *Blade Runner* with no fine tuning and with noise injected into the latent representation: https://www.dropbox.com/s/owiywhn1d48azn1/Blade_Runner_Early_Test.mp4?dl=0

Bibliography

Abadi, Martın et al. (2015). “TensorFlow: Large-scale machine learning on heterogeneous systems, 2015”. In: *Software available from tensorflow. org*.

Bellard, Fabrice, M Niedermayer, et al. (2012). *FFmpeg*. `http://ffmpeg.org`.

Casey, Michael and Mick Grierson (2007). “Soundspotter/remix-tv: fast approximate matching for audio and video performance”. In: *Proc. of the International Computer Music Conference*.

Darkly, A Scanner (2006). “Directed by Richard Linklater”. In: *Warner Independent Pictures*.

Denton, Emily L, Soumith Chintala, Rob Fergus, et al. (2015). “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks”. In: *Advances in Neural Information Processing Systems*, pp. 1486–1494.

Descartes, René (1967 [1641]). “Meditations On First Philosophy (Meditation III)”. In: *Descartes: Philosophical Writings*.

Dick, Philip K (1982 [1968]). *Do Androids Dream Of Electric Sheep? - Blade Runner*. Random House LLC.

– (2011 [1977]). *A scanner darkly*. Houghton Mifflin Harcourt.

Freitas, Nando De (2015). *Deep Learning Lecture 10: Convolutional Neural Networks*. `https://www.youtube.com/watch?v=bEUX_56Lojc&list=PLjK8ddCbDMphIMSXn-w1IjyYpHU3DaUYw&index=10`. Accessed: 2016-04-07.

Gere, Charlie (2002). *Digital culture*. Reaktion Books.

– (2015). *Network Agora*. `https://vimeo.com/130532543`. Accessed: 2015-05-12.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). “Deep sparse rectifier neural networks”. In: *International Conference on Artificial Intelligence and Statistics*, pp. 315–323.

Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*, pp. 2672–2680.

Goodfellow, Ian J et al. (2013). “Maxout networks”. In: *arXiv preprint arXiv:1302.4389*.

Graves, Alex (2013). “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850*.

Graves, Alex and Nando De Freitas (2015). *Deep Learning Lecture 13: Alex Graves on Hallucination with RNNs*. `https://youtu.be/-yX1SYeDHbg?t=49m44s`. Accessed: 2015-12-10.

Gregor, Karol et al. (2013). “Deep autoregressive networks”. In: *arXiv preprint arXiv:1310.8499*.

Gregor, Karol et al. (2015). “DRAW: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623*.

Grierson, M (2009). “Plundermatics: real-time interactive media segmentation for audiovisual analysis, composition and performance”. In: *Proceedings of Electronic Visualisation and the Arts Conference. Computer Arts Society, London*.

Gybenko, G (1989). “Approximation by superposition of sigmoidal functions”. In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314.

Haynes, John-Dylan (2013). “Mindreading in Modern Neuroscience”. In: *Total Recall: The Evolution of Memory - Ars Electronica 2013*, pp. 18–24.

He, Kaiming et al. (2015). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786, pp. 504–507.

III, Matrix (1972). “Directed by John Whitney”. In: *Film, Independent*.

Ioffe, Sergey and Christian Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167*.

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, et al. (2015). “Spatial transformer networks”. In: *Advances in Neural Information Processing Systems*, pp. 2008–2016.

Jay, Martin (1988). “Scopic regimes of modernity”. In: *Vision and Visuality*, pp. 3–28.

Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P and Max Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Larsen, Anders Boesen Lindbo, Søren Kaae Sønderby, and Ole Winther (2015). “Autoencoding beyond pixels using a learned similarity metric”. In: *arXiv preprint arXiv:1512.09300*.

Larsen, Anders Boesen Lindbo and Søren Kaae Sønderby (2015). *Generating Faces with Torch*. `http://torch.ch/blog/2015/11/13/gan.html`. Accessed: 2016-05-02.

Lee, Honglak et al. (2009). “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 609–616.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng (2013). “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. ICML*. Vol. 30, p. 1.

Makhzani, Alireza et al. (2015). “Adversarial Autoencoders”. In: *arXiv preprint arXiv:1511.05644*.

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Minsky, Marvin and Seymour Papert (1969). “Perceptron: an introduction to computational geometry”. In: *The MIT Press, Cambridge, expanded edition* 19.88, p. 2.

Mital, Parag K, Mick Grierson, and Tim J Smith (2013). “Corpus-based visual synthesis: an approach for artistic stylization”. In: *Proceedings of the ACM Symposium on Applied Perception*. ACM, pp. 51–58.

Mnih, Volodymyr et al. (2015). “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540, pp. 529–533.

Mnih, Volodymyr et al. (2016). “Asynchronous Methods for Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1602.01783*.

Nair, Vinod and Geoffrey E Hinton (2010). “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

Nishimoto, Shinji et al. (2011). “Reconstructing visual experiences from brain activity evoked by natural movies”. In: *Current Biology* 21.19, pp. 1641–1646.

Noe, Alva (2004). *Action in perception*. MIT press.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *arXiv preprint arXiv:1511.06434*.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backpropagation and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082*.

Rezende, Danilo Jimenez et al. (2016). “One-Shot Generalization in Deep Generative Models”. In: *arXiv preprint arXiv:1603.05106*.

Rosenblatt, Frank (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6, p. 386.

Ruder, Manuel, Alexey Dosovitskiy, and Thomas Brox (2016). “Artistic style transfer for videos”. In: *arXiv preprint arXiv:1604.08610*.

Runner, Blade (1982). “Directed by Ridley Scott”. In: *Beverly Hills, CA: Ladd Company*.

Scott, Ridley (1984). *Apple Macintosh advertisement, 1984*.

Springenberg, Jost Tobias et al. (2014). “Striving for simplicity: The all convolutional net”. In: *arXiv preprint arXiv:1412.6806*.

Srivastava, Nitish et al. (2014). “Dropout: A simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Szegedy, Christian et al. (2013). “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199*.

Theis, Lucas, Aäron van den Oord, and Matthias Bethge (2015). “A note on the evaluation of generative models”. In: *arXiv preprint arXiv:1511.01844*.

Turing, Alan M (1950). “Computing machinery and intelligence”. In: *Mind* 59.236, pp. 433–460.

Vincent, Pascal et al. (2008). “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 1096–1103.

Werbos, Paul (1974). “Beyond regression: New tools for prediction and analysis in the behavioral sciences”. In:

Zeiler, Matthew D et al. (2010). “Deconvolutional networks”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, pp. 2528–2535.

Ziwei Liu Ping Luo, Xiaogang Wang and Xiaoou Tang (2015). “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*.