MLP Coursework 4: Final Report

Group 2 : s1415713, s1448320, s1401631

Abstract

We investigate Generative Adversarial Networks (GAN) in the task of unsupervised feature learning utilised in face-swapping, inspired by the recently popular algorithm referred to as *deepfakes*. We focus on the performance of modified network architectures, including XGAN (Royer et al., 2017) and Multi-Scale Dense CNNs (Pelt & Sethian, 2017) on the final transformation result. We also investigate the impact of frequent discriminator updates (Chintala et al., 2016) and batch normalization (Ioffe & Shegedy, 2015) on the images produced. We introduce two data sets of varying complexity, which contain images depicting Donald Trump and Hillary Clinton. Our experiments show improved results when amending our baseline model with batch normalization, Pixel Shuffle and more frequent discriminator updates. Substandard performance is observed when using XGAN and Mixed-Scale Dense CNNs. We find training our model on images of greater size than 64 x 64 infeasible, given the time constraints and computational limitations. We identify the optimal performing model as the baseline with the additions of batch normalization, Pixel Shuffle and frequent discriminator updates. This architecture achieves average generator losses 13% lower than the baseline model, whilst also producing images of superior quality.

1. Introduction

In the previous coursework, we investigated the performance of Generative Adversarial Network (GAN) (Goodfellow, 2014) on the task of face-swapping. This was inspired by the *deepfakes* algorithm, which caused great controversy due to its use in the production of counterfeit pornographic videos featuring faces of celebrities.

We tested our model (GitHub - shaonlu, 2018) on four distinct datasets with varying complexity. The first two datasets consisted of a series of images of Donald Trump and Hillary Clinton (Jansenh.stackstorage.com, 2017). The remaining datasets were composed of images of Albert and Karol, two members of the group. We introduced the notion of an *easy* dataset, which consisted of images taken with the same camera, from a similar angle, with uniform backgrounds, lighting conditions, a narrow array of facial expressions as well as no foreign objects. On the contrary, the *hard* datasets were composed of images obtained from a wide variety of sources. The dataset contained depictions of the individuals from different stages of their lives, with a wide array of backgrounds, lighting conditions, angles and facial expressions.

Our experiments consisted of face-swapping images of Trump & Clinton as well as Karol & Albert. We achieved satisfactory performance when training our model on the *easy* datasets, however, we noted room for further improvement. Several concepts have been proposed in recent studies as potential improvements (Chintala et al.,2016) to the original GAN model architecture designed by Goodfellow (Goodfellow, 2014). In particular, we aimed to implement Adaptive Dropout (Ba & Frey, 2013), batch normalization (Ioffe & Shegedy, 2015) and the sub-pixel method also known as Pixel Shuffle (Shi et al.,2016).

Due to the issues described in *Section 3.1*, we abandoned our intent to incorporate Adaptive Dropout (Ba & Frey, 2013) in our model, instead investigating the benefits of using Cross-GAN (XGAN) (Royer et al.,2017) and Mixed-Scale Dense CNN (Pelt & Sethian, 2017), further described in *Section 3.4* and *Section 3.5* respectively. The codebase for this was adapted from the implementation created by the author of the original face-swapping GAN model (GitHub - shaonlu, 2018).

Our research objectives are therefore centred around investigating the potential benefits of using batch normalization, Pixel Shuffle, XGAN and Mixed-Scale Dense CNNs in our default GAN model. Following the suggestions of Chintala, we further aim to understand the implications of more frequent discriminator updates on the final results achieved (Chintala et al.,2016). Furthermore, in order to produce counterfeit images which can be used for real-world applications, we test our model with images of higher resolution (128 x 128). The results of our research are discussed in Experiments 1 - 5 (*Section 4*). Finally, we aim to produce a video of the Donald Trump - Hillary Clinton Presidential Debate with swapped faces and compare our results to those achieved by Dale on a similar task (Dale et al., 2017).

2. Related Work

The bulk of our research has focused on the task of image-to-image translation, in particular, face-swapping. Automated face-swapping has been attempted as early as 2008 (Bitouk et al., 2008). It has many uses, including film and photography editing, image combination, face de-identification, as well as a multitude of applications in creative work. Bitouk's work was described in detail in the introduction of our interim report.

Despite multiple attempts to automate this process, the resulting solutions suffered from several major limitations. Given an input image, Bitouk's implementation was able to find a suitable replacement face from a pool of images and complete the translation, however, it performed poorly when provided with a particular pair of faces for swapping (Bitouk et al., 2008). Furthermore, it was unable to consistently carry out swaps with subjects of different gender and age.

Dale extended this concept to videos, by attempting to replace the faces of two individuals performing a similar task. This introduced further complexity to the problem, in the form of temporal alignment, changing facial expressions as well as temporal consistency (Dale et al., 2011). His approach also suffered from severe limitations, such as the need for consistent illumination in the target and source videos, high frequency lighting, the lack of hard shadows and the need for an orthographic camera (Dale et al., 2011).

Models based on Generative Adversarial Networks (GAN) (Goodfellow, 2014) have resolved many of these issues. They have seen extensive usage since 2014 for the task of image-to-image translation (Dong et al., 2017). Dong defines "image-to-image translation" as an automated transformation of an image from its original form to a synthetic form, whilst maintaining the structure and semantics of the original image (Dong et al., 2017). His architecture utilises deep convolutional neural networks and conditional GANs. Dong's model achieves significant generality and is able to translate images with large variance - tasks unachievable by most of the previously described models (Dong et al.,2017).

Nevertheless, GANs suffer from several prevalent issues, including an unstable training process, the difficulty of achieving a Nash equilibrium, which occurs due to the gradient-descent based training protocol (Goodfellow, 2016), as well as mode collapse, described in detail in Experiment 3 (*Section 4*) (Arjovsky, Chintala & Bottou, 2017). Furthermore, the deep architecture of GAN models makes them extremely susceptible to hyperparameter changes, whilst also creating a large number of trainable parameters (Pelt & Sethian, 2017). This makes GAN architectures not task-agnostic, as a model created for a particular task cannot be easily adapted in other domains (Pelt & Sethian, 2017).

Several alterations of the original GAN model (Goodfellow, 2014) have been proposed to resolve these issues. These include Wasserstein GAN (WGAN), which aimed to reduce the impact of mode collapse and improve the stability of learning (Gulrajani, 2017). Bicycle GAN (BGAN) also attempted to tackle the issue of mode collapse, whilst also producing more diverse results and encouraging bijective consistency between the latent encoding and output modes (Zhu et al., 2018). Unrolled GAN (U-GAN) on the other hand, stabilizes the training of GANs through the use of complex recurrent generators, whilst also increasing the coverage and diverseness of the data produced by the generator (Metz, 2016).

Cross-GAN (XGAN) attempts to tackle the issue of semantic style transfer and preserve the semantics in the learned embedding space (Royer et al., 2017). Therefore XGAN, allows us to retain semantic feature-level information instead of solely focusing on the pixel-level content of the image. Finally, Mixed-Scale Dense CNNs (Pelt & Sethian, 2017) have

been proposed in a broader context of deep convolutional networks, as a way of resolving the oversensitivity to hyperparameter changes and the task-specific nature of deep CNNs (*Section 3.5*).

We aim to further investigate the potential improvements made by XGAN (Royer et. al, 2017) and the Mixed-Scale Dense model (Pelt & Sethian, 2017) in our experiments. We also aim to understand the enhancements made by the addition of the sub-pixel convolutional layer (Pixel Shuffle) to our model (Shi et al., 2016). Chintala recommends this as an improvement to GAN image-to-image translation, which enhances the quality of the generated images by aggregating the feature maps from the low-resolution space and generating the super-resolution image in a single step (Chintala et al.,2016).

3. Methodology

3.1. Adaptive Dropout

One of the goals identified in the interim report was to implement Adaptive Dropout (Ba & Frey, 2013) in our decoder and assess its impact on the performance of the model. Adaptive dropout is an alternative dropout model which does not have a fixed exclusion probability. Instead, a binary belief network is overlayed on the DNN and trained concurrently. As training progresses, the binary belief network stochastically adapts its architecture based on the input (Ba & Frey, 2013). This is described in detail in Section 3.6 of our interim report.

We have successfully implemented and tested the *AdaptiveDropout* layer in the Keras framework. Unfortunately, we were unable to include it in our decoder model. This is due to the underlying architecture of Keras. The *AdaptiveDropout* layer uses the weights from the previous layer in the model while calculating its output. However, in Keras, we define the model architecture without access to the weight matrix values until the training process commences. The *AdaptiveDropout* layer expects a Tensor with live data, but instead receives a placeholder Tensor with an undefined shape of the form *(?,?,?,?)*. Despite our best efforts, we were unable to resolve this issue. We attach our *AdaptiveDropout* layer code as part of the submission.

Instead, we investigate the potential benefits of the XGAN (Royer et al., 2017) and Mixed-Scale Dense CNN (Pelt & Sethian, 2017) models described below.

3.2. Batch Normalization

As previously mentioned in our interim report (Section 2.1), one of our experiments aimed to investigate the influence of batch normalization (Ioffe & Szegedy, 2015) on stabilising the training process of our GAN model. Batch normalization is a widely used technique which combats the problem of internal covariate shift. It also helps with hyperparameter and weight initialization, by normalising the activations at each layer (Ioffe & Shegedy, 2015). Finally, batch normalization allows us to use higher learning rate values and in turn achieve faster network convergence.

We expect to improve the overall performance of our network by incorporating batch normalization within the GAN's architecture to make it more robust to parameter initialisation. We applied batch normalisation between every convolutional layer and activation unit within both the generator and the discriminator as proposed by the current studies (Chintala, Redford & Metz, 2016).

3.3. Pixel Shuffle

One of the methods mentioned in research as a potential improvement to the performance of GAN networks was the replacement of the transposed convolutional layer within the decoder of the generator with its sub-pixel convolutional counterpart (Chintala et al., 2016). This method has been proposed by Shi and promises significantly faster computations and improvements in the quality of the up-scaled images (Shi et al., 2016).

In order to upscale the downscaled image generated by the encoder (i.e. decoding operation) and project feature maps to a higher-dimensional space, our baseline network used a transposed convolutional layer- often referred to as a fractionally-strided convolutional layer (Zeiler et al., 2010). Transposed convolutions simply swap the forward and backward passes of the convolution with strides. This operation allows us to upscale the input by filling $s-1$ zeros between each input unit, followed by a convolution with kernel k rotated by 180 degrees. The output is hence upscaled to the size determined by the formula in Equation 1 (Dumoulin and Visin, 2016).

$$o' = s(i'-1) + k - 2p \tag{1}$$

This operation works well in the case of classification where only a feedforward pass is needed in order to extract necessary features of different scales (Noh, Hong & Han, 2015). However, the operation is slightly more complicated in case of autoencoders where both down and upsampling operations are needed in a single feedforward pass (Xu et al., 2014) (Turchenko, Chalmers & Luczak, 2017). The biggest issue of the transposed convolutional layer is the fact that it inserts zeros between each of the input units which need to be filled with appropriate values through the process of learning. It is also likely that these zero values might lack any gradient information that can be backpropagated (Noh, Hong & Han, 2015).

In order to resolve the issue of insufficient gradient information required for backpropagation, we have introduced a sub-pixel convolutional layer (Shi et al., 2016) in place of the previously described transposed convolution. The sub-pixel layer consists of a standard convolutional layer followed by an activation unit and a `PixelShuffle` component which performs a periodic shuffle operation (Shi et al., 2016). This way the network performs more convolutions in lower resolution and resizes the resulting feature maps into an upscaled image without the need to perform additional computations on the zero-filled units. In other words, the sub-pixel convolutional layer aggregates low-dimensional feature maps to build a higher resolution image in a single step.

Hence, the whole operation of transforming a low-resolution image (i.e. downscaled; I^{LR}) into the super-resolution (upscaled; I^{HR}) can be described by a formula in Equation 2. Here, PS represents the periodic shuffle operator which resizes the input of size $H \times W \times C \cdot r^2$ (where r^2 denotes the number of activation patterns) into a $rH \times rW \times C$ tensor. This operation was greatly inspired by a *tiled convolution* (Ng et al., 2010) and can be described by the formula in Equation 3. Thus, the convolutional operator W_L in Equation 2 has a shape of $n_{L-1} \times r^2 C \times k_L \times k_L$ with k_L representing the size of the filter.

$$I^{SR} = f^L(I^{LR}) = PS(W_L \cdot f^{L-1}(I^{LR}) + b_l) \tag{2}$$

$$PS(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, C \cdot r \cdot mod(y,r), C \cdot r \cdot mod(x,r)+c} \tag{3}$$

The sub-pixel convolutional layer makes the resizing operations significantly more efficient due to the phase shift operation, providing a $log_2 r^2$ speed-up compared to the deconvolutional layer in case of training time as well as a r^2 speed-up compared to other upscaling methods such as max-unpooling (Turchenko, Chalmers and Luczak, 2017).

Furthermore, according to recent experiments (Shi et al., 2016) (Aitken et al., 2017), the sub-pixel layer yields substantially smoother and more detailed high-resolution images which we hope will improve the quality of the images generated by our GAN network. The results of our experiments have been presented in *Section 4*. We based our code on the *t-ae* implementation of the `PixelShuffler` layer class (Gist, 2018).

3.4. XGAN

To remind the reader, a standard GAN network is composed of two simultaneously trained components - the generator and discriminator (Goodfellow, 2014). The generator aims to learn the distribution p_g over the training data x by sampling a random noise vector z from a uniform distribution $p_z(z)$. Using a differentiable network, the input is then mapped to data space $G(z;\theta_g)$. The discriminator is essentially a classifier $D(x;\theta_d)$ with a sigmoid cross-entropy function which aims to differentiate between real and fake data (Dong et al., 2017).

As explained in our previous report (*Section 4*), we incorporated the *LSGAN* loss function in our baseline GAN due to its potential of generating higher quality images and more stable performance during the learning process when compared to the standard GAN loss (Mao, 2016). *LSGAN* is a variation of the original GAN loss function, which uses least squares loss instead of cross entropy loss. Its exact formula is presented in Equation 4 and 5.

$$\min_D V_{LSGAN}(D) \tag{4}$$
$$= \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(\mathbf{x})-b)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(\mathbf{z}))-a)^2],$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(\mathbf{z}))-c)^2] \tag{5}$$

The main idea behind *LSGAN* and its least squares loss function is that it provides a smooth and non-saturating gradient in the discriminator. This loss function penalises datapoints that are far away from the decision boundary, proportionally to the distance; even though they have been classified correctly. This is a significant change from the cross-entropy

loss which does not take into account the distance, but only the sign of the datapoint, treating points which are distant from the decision boundary in the same way as those much closer to it (Mao, 2016). *LSGAN*, therefore, tends to yield more satisfactory results, stabilising the GAN's training process which is considered a significant challenge in many other image-to-image translation tasks (Goodfellow, 2016).

We can define face-swapping as a more general task of semantic style transfer where, given two unpaired collections of images, we aim to learn to map an image from one domain into the style of another domain whilst preserving its semantic content (Zhao, Rosin and Lai, 2017). The main difference in this approach is that we are trying to preserve a higher-level semantic content in the feature space rather than the pixel-level structure. This approach presents promising results in the face-to-cartoon task with a potential to be used in related tasks such as sketches-to-images or even text-to-image translation (Royer et al., 2017).

In order to further improve the efficiency of our GAN, we decided to incorporate an alternative loss function inspired by *XGAN* (Royer et al., 2017). Similarly to our baseline network, *XGAN* (i.e. *Cross-GAN*) is a dual adversarial auto-encoder. However, due to its semantic consistency loss, it is able to simultaneously learn the domain-to-domain translation, capturing the shared representation of the common domain-specific content in an unsupervised way.

In case of the model architecture, *XGAN* is similar to our baseline network, consisting of an encoder e_1 which takes inputs in domain D_1 (i.e. face A) with its output being fed into the decoder d_2 producing the output in domain D_2 (i.e. face B). The same principle applies to the face B-to-A translation. It is important to note that both encoders are partially shared. However, the main addition is the semantic consistency loss which encourages the model to preserve the semantics in the learned space ($e_1 \approx e_2 \circ g_{1\to2}$; $e_2 \approx e_1 \circ g_{2\to1}$) and retaining semantic feature-level information rather than pixel-level content.

This being said, the *XGAN* loss function which the model aims to minimise consists of four components and can be described as follows (Equation 6):

$$L_{XGAN} = L_{rec} + \alpha_{dann}L_{dann} + \alpha_{sem}L_{sem} + \alpha_{gan}L_{gan} \tag{6}$$

L_{rec} is defined as *reconstruction loss* which allows the input to be reconstructed by the auto-encoder by forcing the model to encode enough information about each of the domains for that input. L_{rec} is essentially a sum of each of the domain's construction losses (i.e. $L_{rec} = L_{rec,1} + L_{rec,2}$), defined as in Equation 7.

$$L_{rec} = L_{rec,1} + L_{rec,2} = \mathbb{E}_{x\sim p_{D_1}}(\|\mathbf{x} - d_1(e_1(\mathbf{x}))\|_2) + \mathbb{E}_{x\sim p_{D_2}}(\|\mathbf{x} - d_2(e_2(\mathbf{x}))\|_2) \tag{7}$$

Moreover, L_{dann} donates the *domain-advarsarial loss* between two domains which encourages the embeddings learned by the encoder to lie in the same subspace (Ganin et al., 2015) and, according to the authors, guarantees the soundness of the cross-domain transformations $g_{1\to2}$ as well as $g_{2\to1}$. We presented its definition in Equation 8, where c_{dann} represents a binary classifier on top of the embedding layer, which aims to categorise encoded images from both domains with l being its cross-entropy loss (Royer et al., 2017).

$$L_{dann} = \mathbb{E}_{p_{D_1}} l(1, c_{dann}(e_1(\mathbf{x}))) + \mathbb{E}_{p_{D_2}} l(2, c_{dann}(e_2(\mathbf{x}))) \tag{8}$$

Furthermore, L_{sem} defines the previously discussed *semantic consistency loss*, which can be described as a loss of cosine distance of embeddings, used in order to preserve the semantics of the input. L_{sem} maps the embedding of an input and the embedding of its translated counterpart to the same point, reinforcing the action of the domain-advarsarial loss (i.e. L_{dann}). Considering that the pixel-level data is a suboptimal method, the semantic consistency loss uses feature-level information instead, which encourages the network to preserve the learned embedding during domain translation (Royer et al., 2017). This loss is described in the Equation 9.

$$\begin{aligned} L_{sem} = L_{sem,1\to2} + L_{sem,2\to1} = \mathbb{E}_{\mathbf{x}\sim p_{D_1}}\|e_1(\mathbf{x}) - e_2(g_{1\to2}(\mathbf{x}))\| \\ +\mathbb{E}_{\mathbf{x}\sim p_{D_2}}\|e_1(\mathbf{x}) - e_2(g_{2\to1}(\mathbf{x}))\| \end{aligned} \tag{9}$$

Finally, L_{GAN} is a standard GAN loss function component which encourages the model to generate more realistic samples by simultaneously training the generator and discriminator with the former aiming to *fool* the latter (Goodfellow, 2014). L_{GAN} is a sum of losses of transforming one domain into the other (i.e. $L_{GAN} = L_{GAN,1\to2} + L_{GAN,2\to1}$) and is defined by the Equation 10.

$$\begin{aligned} L_{GAN,1\to2} = \mathbb{E}_{\mathbf{x}\sim p_{D_2}}(log(D_{1\to2}(\mathbf{x}))) \\ +\mathbb{E}_{\mathbf{x}\sim p_{D_1}}(log(1 - D_{1\to2}(g_{1\to2}(\mathbf{x})))) \end{aligned} \tag{10}$$

3.5. Mixed-Scale Dense CNN

Mixed-Scale Dense CNNs were proposed in December 2017 by Pelt and Sethian, in order to simplify the model architecture of deep convolutional neural networks (DCNN) used in image processing tasks. The authors highlight the complex structure of DCNNs used for image processing, including the need for downscaling and upscaling, additional layer types such as dense layers, and a large number of intermediate trainable parameters and temporary images. This model complexity leads to the extreme sensitivity of the network in terms of the layers used and the chosen hyperparameter values. It is therefore difficult to achieve optimal performance for a particular problem, as this requires tweaking too many variables. Furthermore, networks which have been configured to perform well on a specific task cannot be easily adapted to achieve similar results on a slightly different domain (Pelt & Sethian, 2017).

Instead, the authors propose a network architecture which utilises dilated convolutions for feature extraction, rather than upscaling and downscaling as seen in our original model. This is achieved by using multiple scales in a single layer and creating dense connections between all intermediate images. Furthermore, all layers of the network rely on the same set of operations and connections, which further decreases the amount of tuning required. Pelt & Sethian highlight the robustness and versatility of this model when compared to traditional DNNs used in image processing tasks.

3.5.1. Mixed-Scaling

We introduce the notion of a dilated convolution $D_{h,s}$ with dilation $s \in \mathbb{Z}^+$. In the above notation, h indicates the dilated filter which is nonzero only at multiples of s pixels from the centre of the input. Next, rather than associating each layer of the DCNN with a predefined scale, as is the case in our original model, we use the mixed scale approach. In this approach, each channel of a feature map within a layer will use a different scale. The convolution operations performed on each channel of the output image at a particular layer are specified using the following equation:

$$g_{ij}(z_{i-1}) = \sum_{k=0}^{c_{i-1}} = D_{\mathbf{h}_{ijk},\mathbf{s}_{ij}}\mathbf{z}_{i-1}^k \tag{11}$$

where s_{ij} refers to the dilation used, z_{i-1} refers to the previously computed feature maps, and D_h refers to the dilation used at the particular layer and channel of the output image.

This approach carries several advantages over the upscaling/downscaling model used in our previous experiments. First of all, the use of relatively large dilations in shallow layers of the network allows us to learn large-scale information about the input early on. This can then be used in deeper layers of the network to improve the results achieved in these layers. Furthermore, due to the lack of upscaling, we do not need to learn additional parameters during training which are used in the upscaling operation. This makes our model easier to train and less susceptible to parameter changes. Finally, the mixed-scale DCCN can be applied to a larger subset of problems. This is because the network is able to learn the combinations of dilations it should apply throughout the training process (Pelt & Sethian, 2017).

3.5.2. Densely Connected CNN

Next, we introduce the notion of a densely connected convolutional network. Due to the use of the mixed-scale approach, all of the image features in our network are of size m rows by n columns - the same size as our input and output images. Therefore, in the process of computing a feature map for a given layer, we are able to utilise all previously computed feature maps, rather than relying solely on the output of the previous layer in our network. In order to compute the output of a layer z_i, we first calculate the output of an individual channel j as follows:

$$z_i^j = \sigma\left(g_{ij}\left(\{z_0, ..., z_{i-1}\}\right) + b_{ij}\right) \tag{12}$$

where σ refers to the sigmoid function, b_{ij} is used to indicate the bias, and $g_{ij} : \mathbb{R}^{m_{i-1}\times n_{i-1}\times c_{i-1}} \to \mathbb{R}^{m_i\times n_i}$ is used to perform convolutions on the channels of the input feature maps, using distinct filters for each channel. We then perform a summation of the resulting images in a pixelwise manner.

g_{ij} can therefore be defined using the following equation:

$$g_{ij}(\{z_0, ..., z_{i-1}\}) = \sum_{l=0}^{i-1} \sum_{k=0}^{c_{l-1}} D_{\mathbf{h}_{ijkl}, \mathbf{s}_{ij}} \mathbf{z}_l^k \tag{13}$$

Therefore, a densely connected CNN (DCCNN) can be defined as a network where we use all of the feature maps to produce the final output image, rather than only the feature map of the previous layer. This provides us with several advantages. Finally, once we detect a useful feature in a particular feature map, we are able to use it in any layer of our CNN, without having to propagate it through the network. This limits the number of feature maps and model parameters needed to achieve satisfactory results when using a DCCNN. Consequently, our DCCNN is less susceptible to changes in hyperparameters and layout architecture. Furthermore, densely connected CNNs can be effectively trained with smaller training sets. Hence, due to our relatively small data sets, we expect to see improved results when comparing the model using DCCNN with our original model.

Finally, we combine our densely connected CNN with the mixed-scale approach described in *Section 3.4.1* in order to introduce the mixed-scale densely connected CNN. In this model, the feature set is computed by applying equation 3 to all previously defined feature sets. The following operations are used:

1. Dilated convolutions (3 x 3-pixel filters using channel-specific dilations)
2. Summing over all the pixels of the output images produced by the convolutions
3. Adding a predefined bias to every pixel
4. ReLU Activation function

Finally, to compute the output of the network, we apply the above operations to all feature sets, using 1 x 1-pixel filters. Therefore, the channels of the image outputted by the network are obtained as a result of linear combinations of all the channels of the feature sets. Finally, we apply an activation function. This process is summarised in the equation below:

$$y^k = \sigma'(\sum_{i,j} w_{ijk} z_i^j + b'_k) \tag{14}$$

where w_{ijk} refers to the weights and b'_k refers to the biases - parameters which are learned throughout the training process.

The numerous advantages of this model, such as the ability to produce accurate results using a small number of feature sets and network parameters and the ability to apply the model to a multitude of problems make it a viable solution for various domains - such as image classification, feature detection and instance segmentation (Pelt & Sethian, 2017). Pelt and Sethian test their model on the CamVid dataset (Brostow et al., 2008), performing the image classification task on 11 classes. Their model achieves an 87% global accuracy on this task, overperforming the U-Net (Ronneberger, Fischer and Brox, 2015) and SegNet (Badrinarayanan, Kendall and Cipolla, 2015) networks which were used as benchmarks, whilst requiring 10 times less trainable network parameters.

We will incorporate the mixed-scale dense CNN model in our decoders. This will be combined with the XGAN model described in *Section 3.3*.

3.6. Evaluation Metrics

Recent studies fail to specify a standard evaluation metric used for measuring the performance of GANs. The original paper introducing GANs (Goodfellow, 2014) proposed using log-likelihoods as an evaluation metric. However, this approach is not suitable for higher dimensional data, and therefore cannot be used for our domain (Goodfellow, 2014) (Theis et al., 2016). Another proposal made by Durugkar was to use the nearest neighbour classification algorithm on the pixel data of the generated and original images (Durugkar, Gemp & Mahadevan, 2016). However, Theis states that this method produces inaccurate results and therefore is not viable for evaluating tasks involving image translation (Theis et al.,2016).

3.6.1. GAM

Durugkar proposed the Generative Adversarial Metric (GAM) - given two generative adversarial models M_1 and M_2, we measure the performance by placing the generator of M_1 which we denote as G_1 in a position in which it is attempting to fool the discriminator of M_2, denoted as D_2. In a similar manner, the generator of M_2, denoted as G_2 attempts to fool the discriminator of M_1 (Durugkar, Gemp & Mahadevan, 2016). We calculate the following ratio:

$$r_{test} = \frac{\epsilon(D_1(x_{test}))}{\epsilon(D_2(x_{test}))} \tag{15}$$

where $\epsilon(.)$ computes the classification error rate. This ratio allows us to assess which of the two networks performs better at generalization. Next, we introduce:

$$r_{samples} = \frac{\epsilon(D_1(G_2(z)))}{\epsilon(D_2 G_1(z)))} \tag{16}$$

where z denotes the latent code vector of the autoencoder (generator). This ratio allows us to determine which model can fool the other model more frequently, as we feed the discriminators with images generated by the opposing GANs generator.

Finally, to obtain the winner, the following conditions are evaluated:

$$winner = \begin{cases} M_1 & \text{if } r_{sample} < 1 \text{ and } r_{test} \approx 1 \\ M_2 & \text{if } r_{sample} > 1 \text{ and } r_{test} \approx 1 \\ Tie & otherwise \end{cases} \tag{17}$$

However, GAM does not evaluate image quality itself, it simply compares the performance of two GAN models, using average losses as the metric of comparison. Therefore, this method is still not a reliable metric for assessing the quality of the generated images themselves.

Furthermore, due to time constraints, we were unable to implement this in our models. This is due to the fact that we came across this technique after running our experiments and would need to re-run them in order to incorporate this metric in our results.

Therefore, we opted for visual inspection of the final images - a metric used in many recent studies, including work carried out by Royer, Goodfellow and Borji (Royer et al.,2017) (Goodfellow, 2016) (Borji, 2018).

4. Experiments

All of our experiments use the Trump & Clinton (T & C) *hard* and *easy* datasets, described in *Section 3.1* and *Section 3.2* of our interim report. We have made this decision due to the time constraints associated with running our experiments on four distinct data sets. Furthermore, the datasets containing images Karol and Albert, described in *Section 3.4* and *Section 3.5* of our interim report, contained a limited number of images. It was difficult to obtain additional images for these datasets. In order to resolve this issue, we decided to use only the Trump and Clinton datasets and amended them by adding 4000 additional images. The images for the *easy* dataset were sourced from the video of the Trump - Clinton debate used previously. In the case of the *hard* dataset, we obtained images from Wikimedia (Commons.wikimedia.org, 2018).

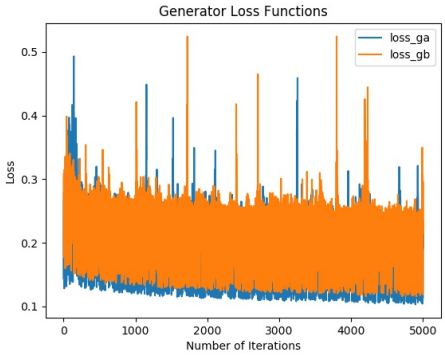
Hyperparameters: All experiments with the exception of Experiment 1 use a learning rate of 0.0001 for the discriminator and generator. Experiment 1 uses a learning rate of 0.001 in order to understand the benefits of batch normalization in our model. Experiments 1,3,4,5 use a mini-batch size of 128 and experiment 2 uses a mini-batch size of 64, due to the memory limitations of the GPU when using larger sized input images. All experiments are run on 5000 iterations due to time constraints.

The average losses for the discriminator (ADL) and generator (AGL) are summarised in *Table 2*.

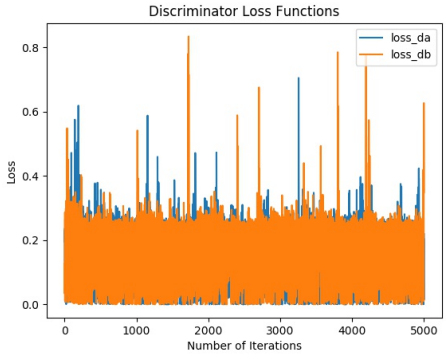
Experiment 1 - Batch Normalization

Motivation: Batch normalization is a technique used to combat the problem of internal covariate shift. It allows us to use higher learning rates and makes the model less sensitive to hyperparameter changes, thus requiring less tuning (Iofe and Shegedy, 2015). The motivation of this experiment is to understand whether batch normalization will make our GAN model more robust to outlier activations, and speed up the training process.

Description: This experiment was carried out using the baseline model described in *Section 4.1* of the interim report and the T & C datasets described in *Section 3.1* and *Section 3.2* of our interim report. The experiment is run on both the *hard* and *easy* datasets, in order to understand

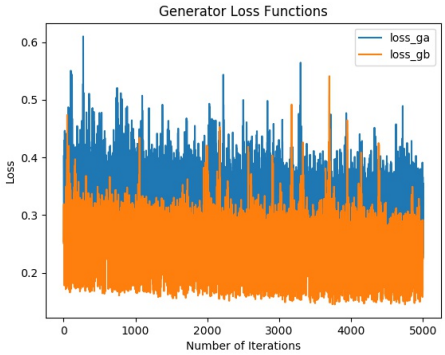


(a) Generator Loss on Final Model Easy

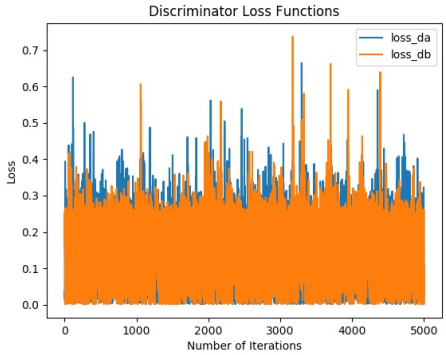


(b) Discriminator Loss on Final Model Easy

Figure 1. Loss funtion for the Final Model trained using the *easy* dataset (*loss_ga* and *loss_da* indicate generator and discriminator losses respectively for the Trump Sample, *loss_gb* and *loss_db* indicate losses for the Clinton Sample)



(a) Generator Loss on Final Model Hard



(b) Discriminator Loss on Final Model Hard

Figure 2. Loss funtion for the Final Model trained using the *hard* dataset (*loss_ga* and *loss_da* indicate generator and discriminator losses respectively for the Trump Sample, *loss_gb* and *loss_db* indicate losses for the Clinton Sample)

Model	loss_da	loss_db	loss_ga	loss_gb
Baseline T & C Easy	0.1757	0.1776	0.2395	0.2596
Batchnorm T & C Easy	0.1746	0.1769	**0.2069**	0.2205
Larger Input T & C Easy	0.1807	0.1829	0.2895	0.3129
Freq. Discr. T & C Easy	0.1744	0.1733	0.2127	0.2277
PixelShuffle T & C Easy	0.1714	0.1753	0.2102	**0.2195**
XGAN +MS T & C Easy	**0.1392**	**0.1493**	0.4732	0.5104
Baseline T & C Hard	0.1729	0.1774	0.3523	0.2668
Batchnorm T & C Hard	0.1680	0.1733	0.3347	0.2560
Larger Input T & C Hard	0.1740	0.1797	0.5121	0.4036
Freq. Discr. T & C Hard	0.1746	0.1762	0.3510	0.2649
PixelShuffle T & C Hard	0.1728	0.1710	**0.3308**	**0.2521**
XGAN + MS T & C Hard	**0.1229**	**0.1328**	0.5323	0.5522

Table 1. Averages loss in baseline & experiments 1-5 - T & C refers to Trump (Face A) & Clinton (Face B), Freq. Discr. refers to Frequent Discriminator Updates (*Experiment 3)*, MS refers to Mixed-Scale Dense CNN (*Experiment 5*)

how inhibiting factors such as varying backgrounds, lighting conditions and watermarks affect the benefits obtained from using batch normalization. We added a batch normalization layer between each convolutional layer and activation unit in both the generator and the discriminator - as recommended by the current studies (Chintala, Radford & Metz, 2016). We use a learning rate of 0.001 - as suggested by Iofe and Shegedy. All other experiments use 0.0001 as their learning rate.

Results:

Easy: The average generator loss is similar for both faces, 0.2069 in the case of Trump and 0.2205 in the case of Clinton. We observe a substantial decrease in generator losses - 14% decrease for generator A, and 15% in the case of generator B. The average discriminator loss is 0.1746 for Trump and 0.1769 in the case of Clinton and does not change in value throughout training - similarly as in the case of the baseline. We observe a minute decrease in the average discriminator losses. The average losses for the generator reach their optimal value after about 1500 iterations. In the case of our baseline model, we observed a steady decline in generator losses for about 4500 iterations. Therefore, convergence occurs much faster when batch normalization is used. The graphs for the baseline model were provided in the interim report (*Figure 1*). Finally, the training process took almost 1.7 times longer than in the case of the baseline.

Hard: The average generator losses obtained are 0.3347 in the case of Trump and 0.2560 in the case of Clinton. We observe a small decrease in generator losses - 5% decrease for generator A, and 4% in the case of generator B. The average discriminator loss is 0.1746 for Trump and 0.1769 in the case of Clinton. Similarly to the easy dataset, we observe small decreases in the average discriminator losses. Faster convergence can also be observed in this case, as the average generator losses achieve their optimal value after about 2000 iterations when compared to 3500 in the baseline model (*Figure 2*). This model also took significantly longer to train than the baseline.

Conclusions: The decrease in generator losses observed in both the *easy* and *hard* datasets can be attributed to the regularization properties of batch normalization and the smaller impact of weight initialization on final results. Due to the depth of our network and a large amount of trainable parameters, it is extremely sensitive to hyperparameter and architecture changes. Therefore, we believe that the small amendments made by batch normalization result in such a substantial improvement in the average generator losses. This vulnerability to small changes in hyperparameters and the network architecture is the main motivation for the mixed-scale dense CNN model proposed by Pelt and Sethian (Pelt & Sethian, 2017).

The smaller decrease in average generator losses seen in the *hard* dataset implies that the effects of batch normalization are somewhat hindered by the complexity of the dataset. Nevertheless, we still observe a positive

impact on the overall performance.

The faster convergence in average generator losses observed in both datasets is the result of the higher learning rate used and the positive effect of batch normalization on gradient flow throughout the network - as gradients in the network are less dependent on the initial parameter values and their scale (Iofe and Shegedy, 2015).

Experiment 2 - Larger Image Size

Motivation: The images used in the baseline experiments had a relatively small size of 64 x 64. This made them unusable in any real-life setting, such as producing a face-swapped video. Therefore, the aim of this experiment was to understand if similar results can be obtained whilst using a more practical image size, which would produce results we could use for our face-swapped video of the Trump-Clinton Presidential Debate.

Description: This experiment was carried out using the baseline model described in *Section 4.1* of our interim report. The image size used was doubled from 64 x 64 to 128 x 128.

Results:

Easy: The average generator loss is similar for both faces, 0.2895 in the case of Trump and 0.3129 in the case of Clinton. Both average generator losses see a substantial increase - 21% for Trump, and 20% in the case of Clinton. The discriminator losses grow negligibly when compared to the baseline model. We observe discriminator losses of 0.1807 and 0.1829 for Trump and Clinton respectively. The time required to complete the same amount of iterations as the baseline model has significantly increased by a factor of 2.5. This configuration also takes longer to complete than the model which uses batch normalization, described in Experiment 1.

Hard: The average generator losses obtained are 0.5121 in the case of Trump and 0.4036 in the case of Clinton. We observe a considerable rise in generator losses - 45% for generator A, and 51% in the case of generator B. This is a substantially higher increase when compared to the easy dataset described above. The average discriminator loss is 0.1740 and 0.1797 for Trump and Clinton respectively. This is lower than in the case of the easy dataset. This configuration also required about 2.5 times longer to complete than the baseline model.

Conclusions: We observe a substantial increase in the average losses for both generators. This can be observed in both data sets. Due to the larger input dimension size, and consequently larger output dimensions, the generator must produce more data to output a fake image. This means that in order to generate a counterfeit of comparable quality, it must process more pixels from the training data. Therefore, the model requires a greater number of iterations to achieve the same performance as when using a smaller input size.

Moreover, a greater number of computations must be carried out on the larger input, which in turn leads to more time required to complete a single iteration. However, due to time constraints, our experiment is run on 5000 iterations - the same as the baseline model. This may explain the increase in average generator losses observed.

It is also worth noting that the *hard* dataset exhibits a percentage increase which substantially surpasses that of the *easy* dataset. This signifies that the complexity of the dataset has an elevating impact on the generator losses. However, the time required to train the model on both data sets was relatively similar, which indicates that the data set has no impact on the time needed for training. We conclude that in order to achieve viable quality of the face-swapped images, which can then be used in real-world applications, the model must be trained for a significantly larger number of iterations. This is unfortunately infeasible in our case, given the constraints on time and computational power. Due to this, we were unable to meet our final goal of producing a video of the Trump-Clinton Presidential Debate with swapped faces.

The small increase in the average discriminator losses in the case of the *easy* dataset shows that, counterintuitively, when the generator performance deteriorates, this does not always result in better discriminator performance. However, when the degradation of generator performance is substantial, as in the case of the *hard* dataset, we observe a decrease in discriminator losses. This behaviour is expected, as the discriminator finds it easier to identify the poorer quality counterfeits produced by an underperforming generator.

Experiment 3 - Frequent Discriminator Updates

Motivation: The goal of this experiment was to investigate the viability of the suggestions made by Chintala & Goodfellow (Chintala et. al, 2016) (Goodfellow, 2017). They advocate for multiple discriminator updates per single generator update. This technique is present in several GAN models, most notably WGAN (Gulrajani, 2017) and Unrolled GAN (Metz, 2016). It carries several advantages over updating the discriminator and generator with equal frequency. In an ideal situation, we aim to achieve discriminator convergence. However, this is extremely expensive and practically infeasible. Nevertheless updating the discriminator more frequently is a small step in this direction. Furthermore, the creators of WGAN claim that this technique leads to greater learning stability and a smaller impact of mode collapse on the final output (Arjovsky, Chintala & Bottou, 2017). Mode collapse is a prevalent issue encountered while training GANs, where the generator focuses its efforts on particular features of the image, rather than giving uniform weight to all modes of the data distribution (Goodfellow, 2016).

Description: This experiment was carried out using the default model described in *Section 4.1* of our interim report, with one additional discriminator update per generator update. Chintala states that it is extremely difficult to determine the optimal ratio of discriminator / generator updates. However, due to time and computing-power limitations, we perform two discriminator updates for each generator update (Chintala et. al, 2016).

Results:

Easy: The average generator loss is similar for both faces, 0.2127 in the case of Trump and 0.2277 in the case of Clinton. We observe a relatively significant decrease in generator losses - 11% for generator A, and 12% in the case of generator B. The average discriminator loss is 0.1744 for Trump and 0.1733 in the case of Clinton and does not change in value throughout training. A minimal decrease in average discriminator losses can be noted. The training process took marginally longer than in the case of the baseline.

Hard: The average generator loss for Trump has not changed significantly and stands at 0.3510. Similarly, the average generator loss for the Clinton images has dropped marginally by 0.7% to 0.2649. The average discriminator losses have changed marginally for both sets of images. We note a loss of 0.1746 for Trump (1% increase), and 0.1762 (1% decrease) in the case of Clinton.

Conclusions: The overall decrease in average generator losses can be attributed to the diminishing impact of mode collapse in our model. The images in the *easy* data set exhibit low diversity, and consequently low number of "key-points" or distinctive features. Therefore, the generator maps a given set of "key-points" to the same image. As as a result, the discriminator will reject most of these images. By updating the discriminator more frequently, we provide it with the opportunity to achieve greater convergence on the forged images and therefore become more sensitive to minimal differences in the underlying data distribution of the counterfeit images. Thus, the discriminator becomes more capable of distinguishing very similar images. This explains the improved performance on the easy dataset. Furthermore, this model achieves similar performance to the one using batch normalization when compared on the *easy* datasets.

The negligible decrease in average discriminator losses is similar to the results observed in previous experiments and cements our conclusion that small changes in generator losses do not impact discriminator performance. The more frequent discriminator updates also seem to not impact the average loss achieved by the discriminator, however, we should note that obtaining the optimal ratio of discriminator/generator updates is difficult and requires further research, as indicated by Chintala (Chintala et. al, 2016).

The model trained on the *hard* dataset behaves as expected. We observe a marginal decrease in generator losses for both sets of images. We ascribe this to the significant diversity of the *hard* dataset. As a result of this, the baseline model trained on this dataset is unlikely to suffer from mode collapse to the same degree as the one trained on the *easy* data. This is intuitive, as diverse data will have a significant number of "key-points". Furthermore, these points will vary significantly for all images in the dataset. It is unlikely for two distinct sets of "key-points" to be mapped to a single image. Consequently, the benefits of tools used to reduce the impact of mode collapse are marginal. The discriminator losses obtained on the *hard* dataset align with our expectations and are similar to the values observed in previous experiments.

Experiment 4 - Pixel Shuffle

Motivation: The goal of this experiment was to investigate the impact of adding sub-pixel convolutional layers (Pixel Shuffle) to our model. This technique was proposed as a method of improving the quality of the final generated images, by aggregating the feature maps from the low-resolution space and generating the super-resolution image in a single step (Shi et al.,2016). We add this layer to the decoder of our generator, as suggested by Chintala (Chintala et al., 2016). This technique is further described in *Section 3.3*.

Description: This experiment was carried out using the default model described in *Section 4.1*, with added convolutional and Pixel Shuffle layers in place of the transpose convolutional layer in the decoder of the generator.

Results:

Easy: The average generator loss is similar for both faces, 0.2102 in the case of Trump and 0.2195 in the case of Clinton. We observe a decline in generator losses - 12% and 15% for Trump and Clinton respectively. The discriminator losses exhibit a negligible decline when compared to the baseline model. We note discriminator losses of 0.1714 and 0.1753 for Trump and Clinton respectively. The time required to complete the training process does not differ substantially from the baseline model.

Hard: The average generator losses recorded are 0.3308 in the case of Trump and 0.2521 in the case of Clinton. We note that the generator losses have fallen for both image sets - 6% for generator A, and 5% in the case of generator B. The average discriminator loss is 0.1728 and 0.1710 for Trump and Clinton respectively.

Conclusions: The generator losses for both datasets have fallen by a considerable amount, which indicates that Pixel Shuffle has had a beneficial impact on the performance of our model. It is worth noting that the benefits observed are greater in the case of the *easy* dataset, which indicates that a highly diverse dataset can limit the benefits obtained from the use of sub-pixel convolution layers in the model. Furthermore, the results recorded are virtually indistinguishable from those seen in Experiment 1, when using batch normalization. However, the time required for training this configuration was substantially lower and comparable to that of the baseline. Finally, the discriminator losses have witnessed marginal changes, in line with previous experiments.

Experiment 5 - XGAN & Mixed-Scale CNN

Motivation: Due to our inability to incorporate Adaptive Dropout (Ba & Frey, 2013) in our model, we decided to experiment with XGAN (Royer et al.,2017) & Mixed-Scale Dense CNNs (Pelt & Sethian, 2017). We describe both of these concepts in detail in our methodology (*Section 3.4* & *Section 3.5* respectively). Mixed-scale densely connected CNNs aim to utilise all previously extracted features when computing the feature map of a particular layer (Pelt & Sethian, 2017). XGAN allows us to retain semantic feature-level information instead of solely focusing on pixel-level content (Royer et al.,2017). Therefore, both of these techniques focus on utilising higher-level characteristics of the image, rather than focusing only on features extracted in the previous layer or pixel-level content of the image. The Mixed-Scale dense CNN approach allows us to limit the sensitivity of our network to parameter changes and make our model applicable to a wider array of image processing tasks (Pelt & Sethian, 2017). This also reduces the number of trainable hyperparameters and consequently allows the model to train faster. We combine these two models in this experiment as, without the reduced number of trainable hyperparameters and the time benefits of Mixed-Scale dense CNNs, the XGAN network model takes infeasibly long to train due to its complex architecture.

Description: This experiment was carried out using the default model described in *Section 4.1* with the following changes: the mixed-scale dense CNN model was added to the decoder for the purpose of output refinement, the domain adversarial loss and semantic consistency loss *(Section 3.4)* were added to the overall loss function. The LSGAN loss function was replaced with the XGAN loss function *(Section 3.4)*.

Results:

Easy: The average generator loss observed is significantly higher than in previous experiments, 0.4732 for Trump and 0.5104 for Clinton. This constitutes a 98% increase on the Trump imageset, and a 97% rise on the Clinton sample. The discriminator losses are lower than in any previously observed configuration - 0.1392 for Trump and 0.1493 for Clinton. These results are 20% lower than the baseline in the Trump case, and 16% lower for Clinton. This model trained slightly longer than the baseline model.

Model	loss_da	loss_db	loss_ga	loss_gb
Baseline T & C Easy	0.1757	0.1776	0.2395	0.2596
Final Model T & C Easy	**0.1701**	**0.1718**	**0.1907**	**0.2137**
Baseline T & C Hard	**0.1729**	0.1774	0.3523	0.2668
Final Model T & C Hard	0.1732	**0.1715**	**0.3253**	**0.2491**

Table 2. Average loss achieved by baseline and final model configurations - T & C refers to Trump (Face A) & Clinton (Face B))

Hard: The average generator loss for Trump stands at 0.5323, which constitutes a 51% surge when compared with the baseline. The Clinton generator loss is equal to 0.5522, a value 107% higher than the baseline. The discriminator losses have fallen for both samples. We observe discriminator losses of 0.1229 for the Trump imageset and 0.1328 for Clinton. This constitutes a 29% loss in the case of Trump and 25% for Clinton.

Conclusions: We observe a non-uniform increase in generator losses for both datasets. In the case of the *easy* dataset, we note a two-fold rise in losses. When training on the *hard* data, the losses obtained are of similar value as in the *easy* case. This behaviour is unexpected, and may perhaps indicate that the use of high-level semantic information and the use of all previously extracted features when computing the feature map of a particular layer negates the negative impact of data complexity on the training process. In fact, our model seems to perform similarly for both datasets.

This performance, however, is not ideal, and significantly worse than our baseline model. We believe that due to the complexity of the XGAN model with the added Mixed-Scale Dense convolutions, it takes significantly more iterations to train. This is likely compounded by the fact that both of these models make use of high-level image characteristics, which must be sourced from the whole network, which in turn requires longer training to achieve the same result. It is difficult for us to draw conclusions, due to the novelty of XGAN and consequent lack of research on this topic. However, we conclude that in order to fully understand this behaviour, the model must be trained significantly longer. We also believe that more meaningful results can be obtained when training the XGAN and Mixed-Scale Dense CNN models separately, however, this was infeasible in our case due to time and computational power constraints.

Finally, the observed behaviour of the discriminators for both datasets is not surprising when taking into account the generator loss values. Since the quality of the counterfeit images produced by the generator is relatively poor, the discriminator performs better, as it has less difficulty identifying the fake images.

5. Final Model

We have identified the optimal performing architecture as the baseline model with the following modifications:

1. batch normalization added between each convolutional layer and activation unit of both the generator and discriminator, as suggested by Chintala (Chintala, Radford & Metz, 2016) (Experiment 1)

2. two discriminator updates for each generator update, as recommended by Chintala and Goodfellow (Chintala et al., 2016) (Goodfellow, 2017) (Experiment 3)

3. a pair of convolutional layers & sub-pixel convolutional layers in place of the transpose convolutional layers in the decoder of the generator (Chintala et al., 2016) (Experiment 4)

This model achieves the performance seen in *Table 2*. We observe an average 14% decline in generator losses on the Trump images, and 12% in the case of Clinton. Furthermore, based on visual inspection, the final images produced by this model are superior in quality to those achieved by the baseline, as seen in *Figure 3*.

6. Conclusions

The following conclusions have been made:

1. As suggested by Chintala, batch normalization has a positive impact on the performance of our GAN (Chintala, Radford & Metz, 2016). Its stabilizing effect on the training process, allows us to use a higher learning rate and therefore achieve faster convergence as well as reduce the sensitivity of the network to hyperparameter changes and the values assigned to the weights and biases upon initialization (Ioffe & Shegedy, 2015). The final images produced are less susceptible to noise and constitute a more accurate representation of the target's face. However, batch normalization substantially increases the complexity of the model, and consequently the time required for training.

2. In order to obtain images which can be used in real-world scenarios, such as generating videos with swapped faces, the images outputted by our model must be of high resolution (ideally the same resolution as the target video). However, when training our model with images of size 128 x 128, we observe a vast increase in the time required to achieve the same level of convergence. The original code base (GitHub - shaoanlu, 2018) recommends using 50k iterations for this task, however we were limited to 5000 iterations due to the time constraints and available hardware. Therefore, in order to use our model for real-world problems, we require longer training and more computational resources. Due to this, we were unable to produce the face-swapped video of the Trump-Clinton Debate.

3. Updating the discriminator more frequently than the generator, as suggested by Chintala and Goodfellow (Chintala et al.,2016) (Goodfellow, 2017), leads to substantial performance improvements on datasets with small diversity. We observe significantly smaller average generator losses when training on the *easy* dataset. Frequent discriminator updates may reduce the impact of mode collapse, an issue prevalent in GANs trained on datasets with little diversity (Arjovsky, Chintala & Bottou, 2017). The benefits observed on highly diverse datasets are not as significant, however small improvements can still be noted. This trend is reflected in the final image quality, as we observe notable improvements when swapping images from the *easy* dataset. Face-swapping samples from the *hard* dataset produces results of similar quality as in the baseline model.

4. Sub-pixel convolutional layers (Pixel Shuffle) (Shi et al., 2016) achieve improvements in performance similar to batch normalization, without requiring lengthy training time. This suggests that altering the decoder of the generator, by aggregating feature maps from the low resolution space and generating the super-resolution image in a single step, rather than using transpose convolutions for upscaling, is likely to improve final image quality. The results obtained align with the suggestions made by Chintala for improving GAN performance (Chintala et al.,2016).

5. Cross-GAN (XGAN) (Royer et al., 2017) was proposed as an alternative GAN architecture, which aims to utilise the semantic feature-level information of the image, rather than focusing solely on its pixel-level content when generating the final output. Our experiments show that XGAN performs similarly on both datasets, which may indicate that the use of high-level semantic information negates the negative impact of the complexity of the data on the training process. However, when training for 5000 iterations we do not obtain satisfactory results. The image quality is distorted, which leads us to conclude that due to the complexity of the model, training must be carried out for a significantly greater number of iterations. It is worth noting that our XGAN experiment was run in conjunction with the Mixed-Scale Dense CNN approach (Pelt & Sethian, 2017). Although not ideal, this was necessary in this scenario, as running XGAN independently, without the time reduction benefits of Mixed-Scale Dense CNN took infeasible long to train. Nevertheless, we recognise that further research into the benefits of XGAN is required, and more meaningful results can be obtained when training an XGAN model independently.

6. Deep CNNs, such as the generator and discriminator of the GAN model suffer from extreme sensitivity to changes in hyperparameter values and small alterations to the architecture of the network (Pelt & Sethian, 2017). This makes it difficult to find the optimal hyperparameter configuration of these models, whilst also making them difficult to adapt to any other task than the one they were originally designed for (Pelt & Sethian, 2017). Furthermore, these models take long to train, due to the large number of trainable parameters. Mixed-Scale Dense CNNs were proposed as a solution to these issues by Pelt and Sethian. This tool is used in our XGAN experiments, leading to a significant reduction in the time required to train the model. However, this should be run solely alongside the baseline model in order to meaningfully assess its impact on model performance.

7. Given the computational resource constraints and time limitations, we identify the optimal performing model in our scenario as the baseline model with added batch normalization between each convolutional layer and activation unit of the generator, two discriminator updates for each generator update and a pair of convolutional layers and sub-pixel convolutional layers in place of the transpose convolutional layers in the decoder of the generator. This model achieves the lowest generator losses observed yet, an average 13% lower than the baseline. This can also be observed in terms of final image quality, as this model produces images superior to any of the other tested configurations, as seen in *Figure 3*.

8. Discriminator and generator losses exhibit an inversely proportional relationship. This trend can be observed across multiple experiments. We note that substantial increases in generator losses leads to improved discriminator performance. As the quality of the counterfeit images produced by the generator deteriorates, the discriminator finds it easier to distinguish them from the real images.

9. There is no standard evaluation metric for measuring the performance of GANs. Log-likelihoods have been proposed as a possible metric (Goodfellow, 2014), however, this approach is not suitable for higher dimensional data like images, and therefore cannot be used for our domain (Goodfellow, 2014) (Theis et al., 2016). Another proposal made by Durugkar was to use the nearest neighbour classification algorithm on the pixel data of the generated and original images (Durugkar, Gemp & Mahadevan, 2016). However, Theis states that this method produces inaccurate results and therefore is not viable for evaluating tasks involving image translation (Theis et al.,2016). GAM (Durugkar, Gemp & Mahadevan, 2016) has been suggested as a tool for comparing the performance of two GANs, however it cannot be used to assess the quality of the output itself. Overall, the method used is usually domain-specific (Selseng, 2017), making it difficult to identify a single uniform metric. However, due to the recent success and growth in popularity of GANs in image translation problems (Dong et al.,2017), there is a substantial need for a reliable evaluation metric for this problem domain.

(a) Baseline Model Easy

(b) Final Model Easy

(c) Baseline Model Hard

(d) Final Model Hard

Figure 3. Baseline & Final Model Image Transformation Results

7. References

References Aitken, A., Ledig, C., Theis, L., Caballero, J., Wang, Z. and Shi, W. (2017). Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize.

Allen, A. and Li, W. (2016). Generative Adversarial Denoising Autoencoder for Face Completion.

An Empirical Study on Evaluation Metrics of Generative Adversarial Networks. (2018). ICLR 2018 (under review).

Arjovsky, M., Chintala, S. and Bottou, L. (2017). Wasserstein GAN.

Ba, J. and Frey, B. (2013). Adaptive dropout for training deep neural networks. [online] Available at: https://papers.nips.cc/paper/5032-adaptive-dropout-for-training-deep-neural-networks.pdf [Accessed 12 Feb. 2018].

Badrinarayanan, V., Kendall, A. and Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.

Bengio, Y. (2016). Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space.

Bitouk, D., Kumar, N., Dhillon, S., Belhumeur, P. and Nayar, S. (2008). Face Swapping: Automatically Replacing Faces in Photographs. Columbia University.

Borji, A. (2018). Pros and Cons of GAN Evaluation Measures.

Brostow, G., Shotton, J., Fauqueur, J. and Roberto Cipolla, R. (2008). Segmentation and Recognition Using Structure from Motion Point Clouds. ECCV.

Chintala, S., Denton, E., Arjovsky, M. and Mathieu, M. (2016). soumith/ganhacks. [online] GitHub. Available at: https://github.com/soumith/ganhacks [Accessed 16 Feb. 2018].

Chintala, S., Radford, A. and Metz, L. (2016). UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. [online] Available at: https://arxiv.org/pdf/1511.06434.pdf [Accessed 12 Feb. 2018].

Commons.wikimedia.org. (2018). Donald Trump by year - Wikimedia Commons. [online] Available at: https://commons.wikimedia.org/wiki/Category:Donald_Trump_by_year [Accessed 20 Mar. 2018].

Dale, K., Sunkavalli, K., Johnson, M., Vlasic, D., Matusik, W. and Pfister, H. (2011). Video face replacement. ACM Transactions on Graphics, 30(6), p.1.

Doersch, C. (2016). Tutorial on Variational Autoencoders.

Dollar, P., Wojek, C., Schiele, B. and Perona, P. (2012). Pedestrian Detection: An Evaluation of the State of the Art. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(4), pp.743-761.

Dong, H., Neekhara, P., Wu, C. and Guo, Y. (2017). Unsupervised Image-to-Image Translation with Generative Adversarial Networks. [online] Available at: https://arxiv.org/pdf/1701.02676.pdf [Accessed 12 Feb. 2018].

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning.

Durugkar, I., Gemp, I. and Mahadevan, S. (2016). Generative Multi-Adversarial Networks.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M. and Lempitsky, V. (2015). Domain-Adversarial Training of Neural Networks.

Gist. (2018). PixelShuffler layer for Keras. [online] Available at: https://gist.github.com/t-ae/6e1016cc188104d123676ccef3264981 [Accessed 20 Mar. 2018].

GitHub - shaoanlu. (2018). shaoanlu/faceswap-GAN. [online] Available at: https://github.com/shaoanlu/faceswap-GAN [Accessed 12 Feb. 2018].

Goodfellow, I. (2014). Generative Adversarial Networks.

Goodfellow, I. (2016). Improved Techniques for Training GANs.

Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks.

Goodfellow, I. (2017). Why do people update the discriminator in GANs just once, and not until convergence, before updating the generator in practice?. [online] Quora. Available at: https://www.quora.com/Why-do-people-update-the-discriminator-in-GANs-just-once-and-not-until-convergence-before-updating-the-generator-in-practice [Accessed 20 Mar. 2018].

Gulrajani, I. (2017). Improved Training of Wasserstein GANs.

Hern, A. (2018). Reddit bans 'deepfakes' face-swap porn community. [online] the Guardian. Available at: https://www.theguardian.com/technology/2018/feb/08/reddit-bans-deepfakes-face-swap-porn-community [Accessed 16 Feb. 2018].

Hinton, G. (2009). Deep belief networks. Scholarpedia. [online] Available at: http://www.scholarpedia.org/article/Deep_belief_networks [Accessed 12 Feb. 2018].

Hitawala, S. (2018). Comparative Study on Generative Adversarial Networks.

Hoffman, J., Tzeng, E., Park, T., Zhu, J., Isola, P., Saenko, K., Efros, A. and Darrell, T. (2018). CyCADA: Cycle-Consistent Adversarial Domain Adaptation. ICLR 2018.

Jansenh.stackstorage.com. (2017). STACK. [online] Available at: https://jansenh.stackstorage.com/s/UayUugaE0GSda0y?dir=

Karpathy, A. (2015). Convolutional Neural Networks for Visual Recognition.

Karpathy, A., Abbeel, P., Brockman, G., Chen, P., Cheung, V., Duan, R., Goodfellow, I., Kingma, D., Ho, J., Houthooft, R., Salimans, T., Schulman, J., Sutskever, I. and Zaremba, W. (2016). Generative Models. [online] OpenAI Blog. Available at: https://blog.openai.com/generative-models/ [Accessed 15 Feb. 2018].

Korshunova, I., Shi, W., Dambre, J. and Theis, L. (2017). Fast Face-swap Using Convolutional Neural Networks.

Ledig, C. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.

Mao, X. (2016). Least Squares Generative Adversarial Networks.

Metz, L. (2016). Unrolled Generative Adversarial Networks.

Ng, A. and Jordan, M. (2014). On Discriminative vs Generative classifiers : A comparison of logistic regression and naive Bayes.

Ng, A., Wei Koh, P., Le, Q., Ngiam, J., Chen, Z. and Chia, D. (2010). Tiled convolutional neural networks.

Noh, H., Hong, S. and Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation.

Pelt, D. and Sethian, J. (2017). A mixed-scale dense convolutional neural network for image analysis. PNAS.

Radford, A. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

Ronneberger, O., Fischer, P. and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation.

Royer, A., Bousmalis, K., Gouws, S., Bertsch, F., Mosseri, I., Cole, F. and Murphy, K. (2017). XGAN: UNSUPERVISED IMAGE-TO-IMAGE TRANSLATION FOR MANY-TO-MANY MAPPINGS.

Selseng, S. (2017). Guiding the training of Generative Adversarial Networks.

Shi, W., Caballero, J., HuszÃąr, F., Totz, J., Aitken, A., Bishop, R., Rueckert, D. and Wang, Z. (2016). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.

Suwajanakorn, S., M. Seitz, S. and Kemelmacher-Shlizerman, I. (2015). What makes Tom Hanks look like Tom Hanks. 2015 IEEE International Conference on Computer Vision.

Theis, L., van den Oord, A. and Bethge, M. (2016). A note on the evaluation of generative models.

Thewlis, J., Bilen, H. and Vedaldi, A. (2017). Unsupervised learning of object frames by dense equivariant image labelling. [online] Available at: https://arxiv.org/pdf/1706.02932v2.pdf [Accessed 12 Feb. 2018].

Turchenko, V., Chalmers, E. and Luczak, A. (2017). A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe.

Xu, L., Ren, J., Liu, C. and Jia, J. (2014). Deep Convolutional Neural Network for Image Deconvolution.

Zeiler, M., Krishnan, D., Taylor, G. and Fergus, R. (2010). Deconvolutional Networks.

Zhao, H., Rosin, P. and Lai, Y. (2017). Automatic Semantic Style Transfer using Deep Convolutional Neural Networks and Soft Masks.

Zhu, J. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.

Zhu, J., Zhang, R., Pathak, D., Darrell, T., Efros, A., Wang, O. and Shechtman, E. (2018). Toward Multimodal Image-to-Image Translation.