Investigative Study on CNN & VGG Generative Models for Face2Sketch and Sketch2Face

Ben Mullally *36155273*

***Abstract*— This report shows a comparison of different CNN architectures applied to the Sketch2Face and Face2Sketch problem of image generation. There is copious research already using state-of-the-art techniques such as GANs to produce excellent results however these methods are often very resource intensive. Transfer learning using pre-trained models such as VGG16, VGG19, and VGGFace is an alternative method that is explored to extract features from images and then using custom convolutional decoders to produce image reconstructions. This process is much quicker and can produce promising results, especially using VGGFace on Sketch2Face. The fine-tuning of models using dropout and regularization techniques to improve generalizability to a test set is also proven to be effective for both tasks. The Sketch2Face problem was the main focus, giving superb results given the timescale of research, but examples of application to the Face2Sketch problem are also shown.**

***Index Terms*— VGG16, VGG19, VGGFace, CNN, Sketch2Face, Face2Sketch, Transfer Learning**

I. Introduction

This study covers the training of different deep learning models on image data of people's faces and their corresponding sketches (see Fig. 1). The aims were to learn generally about how models can produce sketches from faces (Face2Sketch) and vice versa (Sketch2Face). These two simple problems of image generation are extremely useful in many real-world applications. For example, it is obvious that Sketch2Face can be used in criminal face recognition from eye-witness reports which could massively improve the accuracy of their search for possible suspects since witness sketches are notoriously prone to problems [1]. Being able to create realistic images or sketches of people is also useful in the entertainment industry for use in animation, or just for people who want to create artwork using these procedures. Huge corporations like Disney are in fact pouring resources into models for several applications of image generation in their movies and use deep learning models to do so [2].

There is currently not a superior model in Face2Sketch nor Sketch2Face synthesis as research is plentiful, and a lot of approaches have produced excellent results however most are limited by the computational power required to train on the images. With this short project, this paper aims to explore the use of pretrained, highly regarded convolutional network-based models and the use of transfer learning to provide a possible alternative to sketch to face synthesis that can be further explored. Transfer learning uses pretrained models as the basis of information for another model to be trained to then solve a specific task. It is of use in many applications of deep learning using convolutional neural networks (CNN) such as in medicine, agriculture, and weapons for example [3]. Transfer learning significantly reduces training time and so this paper aims to explore the use of pretrained models VGG16, VGG19, and VGGFace as a possible solution to the problems, with more exploration leaning towards Sketch2Face. The first two models are 16-layer and 19-layer CNNs respectively and are highly regarded as some of the best image classification models in the field. They were trained on the ImageNET dataset on over fourteen million images belonging to thousands of categories [4]. VGGFace has a very similar architecture produced by the same Visual Geometry Group at the University of Oxford. Its major difference is that it was trained on human faces for facial recognition so it may be more effective for the problem at hand. The aim is to use the defined parameters of the models to produce features from sketches that can be passed through convolutional layers much like in a CNN to produce realistic photos. This requires partitioning the models in order to remove their final deeply connected layers that would usually classify images (e.g. cats and dogs), replacing them with a layered network that produces an RGB image.

This process of transfer learning on convolutional networks for image classification is something that has been explored before such as skin disease recognition [5] and VGG16 has obviously been used in many applications of classification such as pot-hole detection [6] and plant disease identification [7]. This paper explores the use of VGG models on image generation, using it as a feature extraction method from sketches in the popular CUHK dataset to train a small convolutional network with much quicker results than that of current modern approaches such as GANs. It aims to produce facial image generation from sketches and compare the results between the two models as well as that of a simplistic CNN. Some parameter changes such as kernel regularizers, and the addition of dropout layers was also explored in order to improve the generalizability of the models. The results clearly showed that image generation from sketches using the proposed method was possible but was definitely limited for VGG16 and VGG19. However, the results from VGGFace were excellent and generation to the human eye showed very good promise. The Face2Sketch problem using the same methods was much more limited. Results were obtained but the sketches produced were poor and maybe not a strength of the VGG models. More research and time would be needed to explore improvements of the technique in this regard.

The report will begin with a review of the current state-of-the-art models in sketch face synthesis and discuss the motivation of the developed research. There will then be a discussion of methodologies of CNNs and the more complex VGG16, VGG19, and

VGGFace, along with detail on the theory behind the new method and how models were trained. Finally, there will be the results of experimentation on the datasets chosen, with discussion on the comparison of the different methodologies.

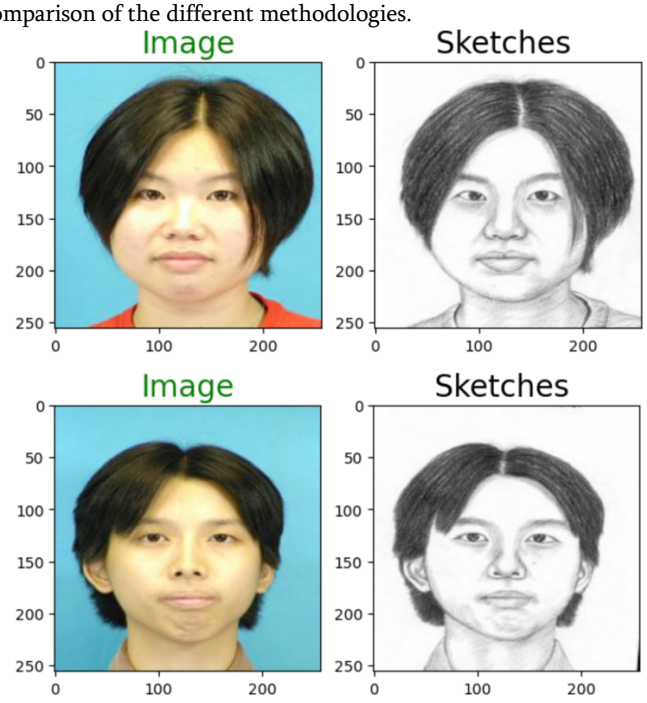

Fig. 1. Example faces and sketches from CUHK dataset used.

II. Literature Review

There is already excellent past research using similar datasets applied in this paper (and on additional datasets) for face to sketch synthesis [8]. Several methods such as the MrFSPS method have been developed for excellent performance, generating images very close to the true sketches, where the differences become negligible to the human eye. The same model was performed for sketch to face generation producing some positive results however there was clear limitations such as the overall accuracy and realism of the generated photos. Furthermore, the dataset was limited to producing grayscale images which simplifies the process and gives less of a challenge. The motivation of this report was to tackle the difficult problem of photo generation with colour from black and white sketches as it is a much more useful application that can be used in many scenarios as discussed earlier. Face sketch photo synthesis is a popular approach whereby the model takes a face from a new photo and produces a sketch of it to match with the sketches already stored in a database or vice versa where a photo is generated from a sketch instead [9]. The aim is to create a scenario where traditional face recognition methods can be used then for identification. For example, this is applicable in criminal identification using police databases with faces in them and the corresponding witness sketches. There are again clear limitations to the process. There is unlikely to be large databases of sketches of people from the public however there are databases of real photos (e.g. passports, driving license, etc.). This motivates image generation from sketches to be the primary goal to then be used in image recognition from existing photo databases. However, image identification in this manner is known to be an "open-set identification problem where training set and test set classes are disjoint" [10, 11]. Excellent performance of image generation can be achieved in training from specific dataset collections with ideal lighting, backgrounds, etc. The challenge is to generalize results for test sets from those datasets (which is discussed in this paper) and then produce further accuracy on sketches from outside the database. This will then be applicable in fields such as witness sketches and be of great use to the public. CMOS-GAN is an interesting recent paper that tackles the problem of insufficient paired training data by using an encoder-decoder approach to build a robust cross-modality model from paired and un-paired training data [12].

Further research has also been done into using bionic face-sketch generators to improve coarse sketches from humans into clearer and more accurate sketches of the actual images [13]. This could be very useful as it could provide training photos with corresponding sketches that have more defining facial features that could boost the accuracy of the generation of new faces and should be explored further in this field.

In terms of photo synthesis from sketches, there is also great advancements being made, mainly by using variations of generative adversarial networks (GANs) [14]. The most well-known image generators to create deepfakes are all using these networks for example and detecting these fake images has become another massive area for research [15, 16]. Image synthesis in the models is done by the generative side of the network, where it tries to 'fool' its partner, the discriminator, into believing a generated image is in fact a real person. Most modern approaches for this problem use a combination of models such as U-Net or P-Net along with the GAN to first improve recognition of features within sketches before inputting into the network [17, 18]. These composition-aided approaches require a decomposition of the face (P-Net for example) which adds an extra layer for the model increasing complexity and can be difficult to produce accurate segmentation for analysis. If the segmentation is done properly however, the results are often very realistic face generations that could easily fool a human.

Other strategies have also been developed with GANs such as exploiting an intermediate latent space between the sketches and faces, and attention-based networks that focus on learning regions of the face and limit background information influence (CSA-GAN) [19, 20].

GANs are, unlike most other approaches, not trained to minimize the differences between the sketch-photo pairs. This makes the training process time-consuming and is often not feasible in real-world, time-restricted environments. This motivated the exploration in this paper of VGG16, a pretrained 16 layer CNN, and using transfer learning to speed up training time. VGG16 and its larger-layer version VGG19 are often used for classification of images, and, despite being developed in 2014, are still in use regularly to this day for numerous tasks [21, 22]. These pre-trained models are excellent at extracting features from images, which proposed the idea of using it for extracting information from the sketches in this problem to gather important features for use in training. The process of transfer learning can then be applied whereby the pre-trained model quickly extracts features that can then be passed through another network to produce a required output. A further model called VGGFace by the same research group at the University of Oxford was created but trained specifically on photos of humans [23]. As a result, this model was also applied to the problem of Sketch2Face and Face2Sketch to see

if it had superior feature extraction to the other methods. VGGFace2 was also a further improvement of the model using a larger, more diverse dataset but was not further explored [24].

III. Preliminary on Methodologies

Three different methodologies were applied to both Sketch2Face and Face2Sketch problems: CNN, VGG16/19, and VGGFace. The models are all convolutional-based networks with a lot of similarities but the subtle differences in architecture and how they were trained make noticeable disparities in results. Furthermore, extra research was done into improving the generalizability of the models, and so there will be a comparison of not only the different techniques but also variations in implementation.

A. Convolutional Neural Network (CNN)

CNNs differ to other forms of artificial neural networks (ANNs) but the overall structure is similar [25]. It is made of interconnected layers, where each layer is comprised of neurons, as shown in figure 2. Typically, these neurons receive an input from the layer before, process it using a defined function, and then send it to its connected neuron in the next layer. All the neurons have specific connections from one layer to another, and each are fundamental in the learning process. Throughout training, the associated weights of the connections change, each time fine-tuning what features of the inputted data are more important to classifying the output correctly via supervised learning. Input data can be of any form but must be broken down into a vector or matrix of some kind in order for the network to perform its functions. The limitations of a simple autoencoder for image processing tasks is that it is often difficult to process the large amount of data in images as a vector. For example, one of the images tested in this study is an RGB (3 channels) of 256x256 pixels therefore it must input 196,608 pixels as features for training which leads to the numbers of weights becoming extremely large over just a few layers. This leads to slow computation time and often results are not worth the wait. A deep autoencoder with 3 dense layers was tested on the Sketch2Face and Face2Sketch tasks but each epoch took considerably long and the image generation after learning was very poor therefore it was not included in the results.

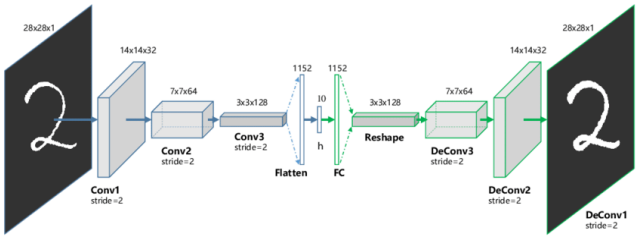

Fig. 2. CNN typical architecture. Taken from [26].

CNNs were built for the handling of images, reducing the number of parameters in training with better learning and therefore are often superior to ANNs. The ANN will likely overfit to the data due to its densely connected layers. The CNN has an architecture to account for the dimensionality of images, allowing processes to be handled in a 3-dimensional matrix (often called a tensor). The overall architecture of the network is shown in figure x. There is first an input layer that takes the individual pixels from an image. The next layers are to be seen as blocks of convolutional and pooling layers, one always followed by the other. The convolutional layer contains a set of kernels (filters) that are applied to the image pixels producing a feature map and throughout training, the weightings of the filters adapt to learn the images features. Since the images in use here are RGB, the convolutional layer kernels also must have a depth of 3, for each colour channel [27]. To process the feature maps created, the layers use a non-linear activation function (often a rectified linear unit, ReLu) on the result of the local weighted sum [28]. The pooling layer applies downsampling to the output, further reducing the dimensionality of the feature space, therefore the number of weights also. Often the max-pooling layer can be replaced by using a stride of 2 within the convolutional layer, reducing the size of the feature maps by half. This technique was used in the code implementation. After several blocks of these layers, the features are distilled into a much smaller information space and within a lot of CNNs, the next step would be to pass through the outputs into deeply connected layers to produce a simplistic classification output that is of just one dimension. However, since the output in this case must be a reconstructed image, the process of the convolutional and pooling blocks is repeated, this time increasing the feature space to return to the size of the full image dimensions for an output. This is referred to as a convolutional autoencoder, where the first step encodes the image into its core features, and the next step decodes them, mapping them to the new desired outputs (e.g. pixels of the face, if the sketch is the input). At each epoch (iteration) in training, the weight parameters of the network adapt on the whole training set and over time, the networks begin to learn how to map sketches into realistic photos or vice versa. The CNN can either increase its dimensions in the middle before decoding (used for the Face2Sketch) or reduce its dimensions instead (Sketch2Face) depending on the problem. For example, since sketches are simpler versions of the faces, the increase in the sparse feature space technique was used, also using less layers.

The architecture implemented for Face2Sketch was an encoder with 3 convolutional layers (filters = 16, 64, & 256) followed by a decoder of 5 convolutional layers with two dropout layers (filters = 256, 64, 16, 8, & 3).

The architecture implemented for Sketch2Face was an encoder of 6 convolutional layers (filters = 16, 32, 64, 256, 512, & 1024) and then the reverse for the decoder. The encoder used batch normalization on the 2nd, 4th, and 5th layers and the decoder applied a dropout of 0.1 on the 1st and 3rd layers. The number of parameters due to the increased number of neurons also increases in this implementation leading to processing per epoch doubling.

B. VGG16 & VGG19

VGG16 was developed as a very deep convolutional neural network and so all its architecture is based off CNNs. Most advanced models using this technique at the time were using large convolutional filters/kernels which were effective but made the networks shallow. Alternatively, VGG16 used smaller kernels, reducing the number of parameters, and allowing for the number of convolutional layers to increase to 13 [29]. The architecture was trained for three weeks on ImageNet images with millions of images within its database. This resulted in a model that had been trained for extracting features from images, allowing the use of these learnt weights in transfer learning. The '16' refers to the total number of convolutional and dense layers, as the final 3 layers are made of

densely connected layers. The model first requires input images to be reshaped to be of a 224x224x3 dimensionality and thereafter comes the repetition of convolutional layers and max pooling as shown in figure 3, resulting in a feature space of 7x7x512. It uses the smallest possible convolutional filters (3x3) to detect left, right, up, and down differences in the image. The filters only have stride of 1 also, meaning more information is kept per iteration. By stacking these convolutional layers and increasing the number of them at each block the further down you go, the model can learn very complex features.

VGG19 was another model developed by the University of Oxford but with slightly different configurations, with the main focus being on three extra convolutional layers, increasing the overall complexity of the network. This often leads to VGG19 having better overall accuracy at a cost for extra computation and memory cost.

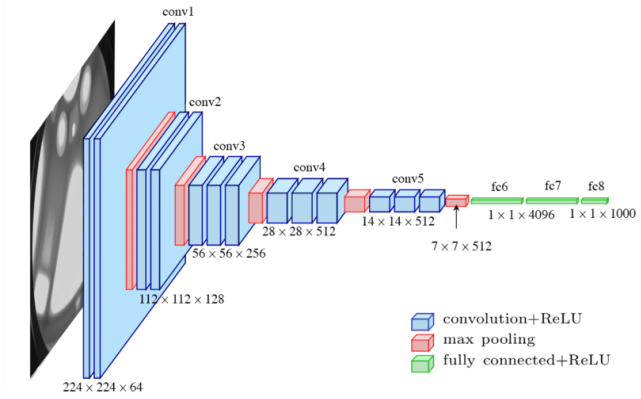

Fig. 3. VGG16 architecture. Taken from [30].

Usually, the networks have three more densely connected layers to map to a final one-dimensional output however, since the task was image reconstruction, a custom decoder was instead created using convolutional and up-sampling layers to produce the 224x224x3 image output needed, replacing the dense layers. This technique is called transfer learning and its use was inspired by an informative YouTube video on image colorization [31]. Transfer learning is a simple method where one uses a pretrained model and its parameters to very quickly extract features from the inputs, basically forming the 'encoder' layer of any autoencoder [32]. In this case, the VGG16 and VGG19 models have been used to extract 7x7x512 feature spaces for every image in the training set, whether it's a Face2Sketch or Sketch2Face. This results in much better feature extraction than a normal CNN encoder. The VGG models could even be retrained on the datasets if there was enough time since the models have hundreds of millions of parameters. The new decoder takes much longer to train than the encoder, taking the feature spaces and mapping them to the output sketch or face, depending on the problem. The structure of this decoder for VGG16 and VGG19 Sketch2Face was firstly two convolutional layers (256 and then 128 filters) followed by an up-sampling layer. Then, a convolutional layer with 64 filters followed by another up-sampling layer, which is then repeated for 32, 16 and 3 filters respectively. Each convolutional layer uses a ReLu activation function except the last layer that uses a tanh activation function to produce the final image. Since Face2Sketch was a much simpler reconstruction, the layer construction differed slightly to minimize loss. The final layer's activation function had to be changed to a ReLu to produce valid outputs for the sketch. These models were tested as a baseline and then further alterations were made to try improving accuracy as discussed in the improving generalizability section.

C. VGGFace

The main problem with applying VGG16 and VGG19 to the problem at hand is that the models were trained for feature recognition across millions of different types of photos and so the subtle differences between human faces may not be best suited to its strengths. Thankfully, in 2015 the same Visual Geometry Group at the University of Oxford trained a model called VGGFace which focused on deep face recognition [33]. They accumulated a dataset of 2.6 million faces from 2.6 thousand people by variations of images of the same people (mainly popular movie stars) and trained a model similar to VGG16 but specifically for face recognition. The architecture is shown in figure 4, consisting of 11 blocks, each with a linear operator followed by non-linearities such as max-pooling and ReLu. However, since the aim is to abstract features from our images, the model is again pruned after the first 8 blocks of convolutional layers, leaving a 7x7x512 feature space. The exact same approach of transfer learning was used as before but now the feature space should be more appropriate due to how the weights were trained.

D. Improving Generalizability

It was noticeable during training that the VGG16 and VGG19 models in particular performed extremely well on training data however lead to overfitting for Sketch2Face since generalization was poor on the test set (discussed further in results section). Adaptations of the decoder model was investigated for comparison with hope of improved performance. On the other hand, Face2Sketch tuning was not tested as thoroughly. Three different strategies were explored:

1) Decreasing the number of layers/neurons- This is a very simple way to reduce the information reconstructed and was more useful on Face2Sketch since the image complexity decreasing was the aim. For example, the CNN used only 8 layers in comparison to the 14 used for Sketch2Face. A further technique is to stop training when performance on a validation set starts to worsen but since epochs were only small in experiments, this technique was not used.
2) Dropout – This is the process of 'turning off' neurons in specified layers during training based on a defined probability. This halts learning for that iteration of very influential neurons, hence the model learns more information overall, leading to robustness in image variation. For large CNNs, this dropout probability is usually most effective with larger values such as 0.5 [33]. However, since the decoder was small (1.5 million parameters), smaller values were used (0.01 for the first two layers of Sketch2Face on VGG16 and VGG19). VGGFace and the CNN both used a dropout of 0.1 at several layers for better generalizability. The focus of dropout is on the much larger layers at the beginning since there are more weights for the dropout to be effective [34].
3) Regularization- There are two main regularizers that were explored: L1 and L2. They are applied again to more complex layers but L2 was more effective since it is often used in image

problems. Regularizers effectively penalize larger, more influential weights during training to improve the accuracy of smaller weights, therefore overall generalizability. The L1 norm takes the norm of the weights using the following formula:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{n=1}^{N} (p_n - y_n)^2 + \lambda \parallel \mathbf{w} \parallel_2,$$

L2 on the other hand uses:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{n=1}^{N} (p_n - y_n)^2 + \lambda \parallel \mathbf{w} \parallel_1,$$

The lambda value is a defined parameter that weights the penalization, leading to 'weight decay' as the weights are update linearly [35]. L2 with lambda=0.01 was the chosen regularizer in Sketch2Face for VGG16 and VGG19 but other models did not use it. The M is defined as the number of training examples whereas the N is the number of neurons, with outputs p_n and desired outputs y_n.

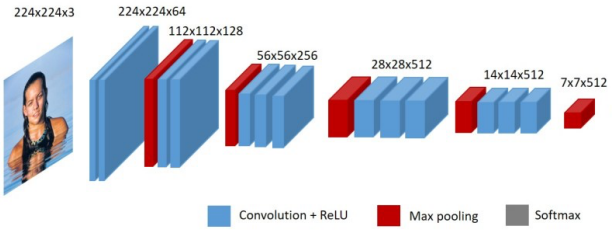

Fig. 4. VGGFace architecture. Taken from [36].

IV. Experimental Results

A. Experimental Setup

The dataset used for analysis was the popular CUHK dataset, containing 88 training images and 100 testing images, both sketches and faces of University of Hong Kong volunteers. To increase the size of the training set, the images were rotated and flipped to form 8 different variations of the same image, forming a total set size of 704. The images were all RGB images of dimensions 256x256x3 and so the images were all divided by 255 first to normalize them between 0 and 1, reducing the computational complexity of the tensor multiplications and weight changes involved in the process of training. Furthermore, the VGG models all required the images to be of 224x224x3 dimensions therefore they were resized before input.

Training is done by passing through the sketch images as inputs, encoding them using the corresponding encoder for each method to produce a feature space, and then decoding to produce the corresponding face. This process is the same but obviously images swap for Face2Sketch. The outputs and inputs are ensured to be the same dimensionality as the image similarity measures used in testing all require this. The measures used were RMSE, PSNR, FSIM, SRE, SAM, and UIQ. ISSM and SSIM were also used however they were not useful for this task, producing values of 0 and 1 respectively for all models therefore they were left out of results. Brief explanations of the methods used will follow [1]:

- Root Mean Square Error (RMSE): Gives the rms difference in pixels, a value of zero means they are identical.
- Peak Signal-to-Noise Ratio (PSNR): = 10log(R^2/MSE) where R is the max pixel value possible in the image. Its output is in logarithmic decibels, with values high between 60 and 80 being good.
- Feature Similarity Indexing Method (FSIM): Compares structural and feature difference measures to produce a value between 0 and 1 where 1 is perfect feature similarity.
- Signal to Reconstruction Error Ratio (SRE): Calculates the error relative to the power/brightness of the image. Higher values (in decibels) show better reconstruction.
- Spectral Angle Mapper (SAM): Uses physics methods of spectral similarity between images using the angle between spectra, treating them as vectors in space, making it resistant to illumination effects. Lower values show better similarity.
- Universal Image Quality index (UIQ): Combines loss of correlation, luminance distortion, and contrast distortion into one metric to give an index between 0 and 1 (perfect match).

The number of epochs for training varied per model as training time per epoch differed due to the number of parameters involved. Since the CNN was made from scratch, the time per epoch was 1-2 minutes whereas the VGG models took between 2 and 12 seconds per epoch to decode as the feature creation from the pretrained models were rapid. This allowed the VGG models to be trained on more epochs due to time constraints but remember this process was only on the decoder and not training of the encoder since those weights were predefined.

Model	Sketch2Face	Face2Sketch
CNN (dropout)	20	50
VGG16	10000	1000
VGG16 (dropout)	10000	1000
VGG19	10000	1000
VGG19 (dropout)	1000	1000
VGGFace	1000	1000
VGGFace (dropout)	1000	1000

Fig. 5. Number of epochs in training per model.

Epochs was kept consistent for Face2Sketch methods and overall training time was approximately one hour whereas there was much more variation in Sketch2Face. This was due to the clarity of the face outputs becoming much clearer after more epochs and this problem itself was explored much more rigorously due to the aims as defined in the introduction. Due to the time restriction of this project also, it was difficult to maximize the training process as the tuning and training of parameters is an endless procedure.

The running environment for these experiments was tensor-flow 2.10 on an Apple MacBook Pro M1 Max Laptop, but only using the CPU version of the library, so it was not completely optimised for GPU use but was still considerably quicker than running it on Google Colab.

B. Training of Your Method

Each method was trained to minimize the mean-squared error (MSE) and each method converged to zero over time which was successful. The loss function does not exactly determine that the model is getting more accurate but, in this case, just that the MSE

between result and the true images was reducing. It is still a good sign of the model getting better but other loss functions could have been investigated further. Analysis of the loss and accuracy over epochs will be split into the two problems defined.

Sketch2Face

There is a clear convergence of training loss towards zero for all models however validation loss seems to stay slightly above it and although changes are not noticeable, the values were decreasing slowly over epochs (shown in figure 7). This is to be expected as the validation set is not being trained upon therefore having losses very close to training values is seen as a good result. The accuracies show the same pattern but more of a convergence to 1 rather than 0 (as shown in figure 8). The accuracy and validation accuracy of these

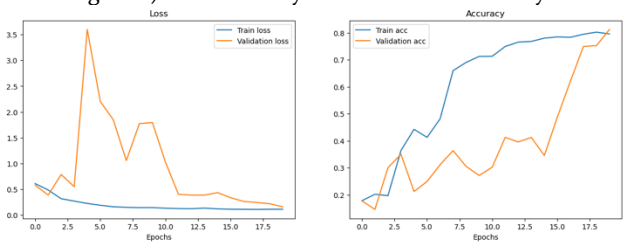

Fig. 6. CNN loss and accuracy over epochs.

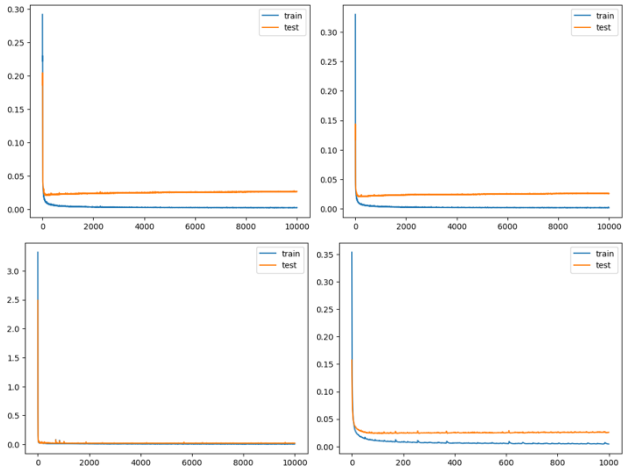

Fig. 7. Loss over epochs of VGG16 (top-left) & VGG19 (top-right), and regularized versions (bottom).

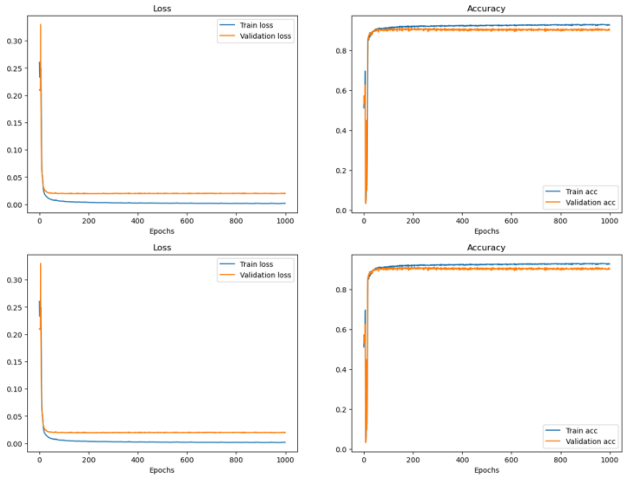

Fig. 8. Loss and accuracy over epochs of VGGFace (top) & VGGFace with dropout (bottom).

models generally tending to values around 90% showing again that the faces outputted were accurate. Throughout training it was apparent that the standard VGG models were poorly generalizing to the validation set as loss did not decrease at the same rate and accuracy was higher for training set (sometimes up to 3% difference). Further conclusions from the images, as shown in the test results section, showed it was clear that the models were overfitting hence a kernel regularizer was used for the VGG16 and VGG19 models and dropout was applied to VGGFace. This improved visual results and decreased loss. The loss for the CNN model shown in figure 7 shows random spikes but ten also convergence and with more epochs the training and test losses will likely tend to 0 also.

Face2Sketch

Variations in loss and accuracy in the first epochs was very normal as shown in the graphs of figure 9. This is due to the models random initialization and then gradual updates until the loss starts to decrease and the accuracy increase. It is apparent in the Face2Sketch problem that the accuracy for the training and validation set plateaus after a certain number of iterations, resulting in minimum gain in results. This was a problem because the accuracy was only at 30% and although loss was decreasing, it did not seem to affect it. Future developments could explore this problem and try to solve it. Overall, Face2Sketch was not focused on as much, but results were still interesting. The VGG models were always more likely to be better at collecting features from sketches to produce faces due to their training and perhaps this overcomplication caused poor performance for Face2Sketch.

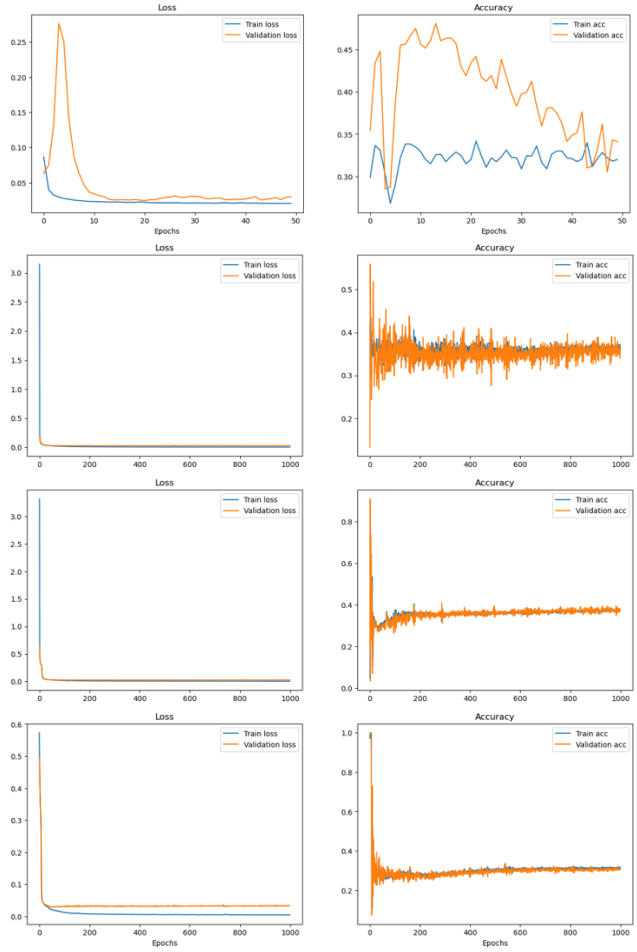

Fig. 9. Loss & accuracy over epochs for CNN (1st), VGG16 (2nd), VGG19 (3rd), VGGFace (4th).

Model	RMSE (x10^{-5})	PSNR	FSIM	SRE	SAM	UIQ
CNN	4.84	86.54	0.72	26.64	5.13	0.19
VGG16	4.47	87.15	0.76	26.35	5.04	0.18
VGG16 (dropout)	3.93	88.37	0.79	25.98	4.59	0.23
VGG19	4.61	86.88	0.75	26.22	5.25	0.17
VGG19 (dropout)	4.56	87.02	0.78	26.29	5.00	0.19
VGGFace	3.98	88.23	0.79	26.91	4.77	0.26
VGGFace (dropout)	**3.85**	**88.54**	**0.81**	**27.07**	**4.41**	**0.28**

Fig. 10. Image similarity measures across all models for Sketch2Face.

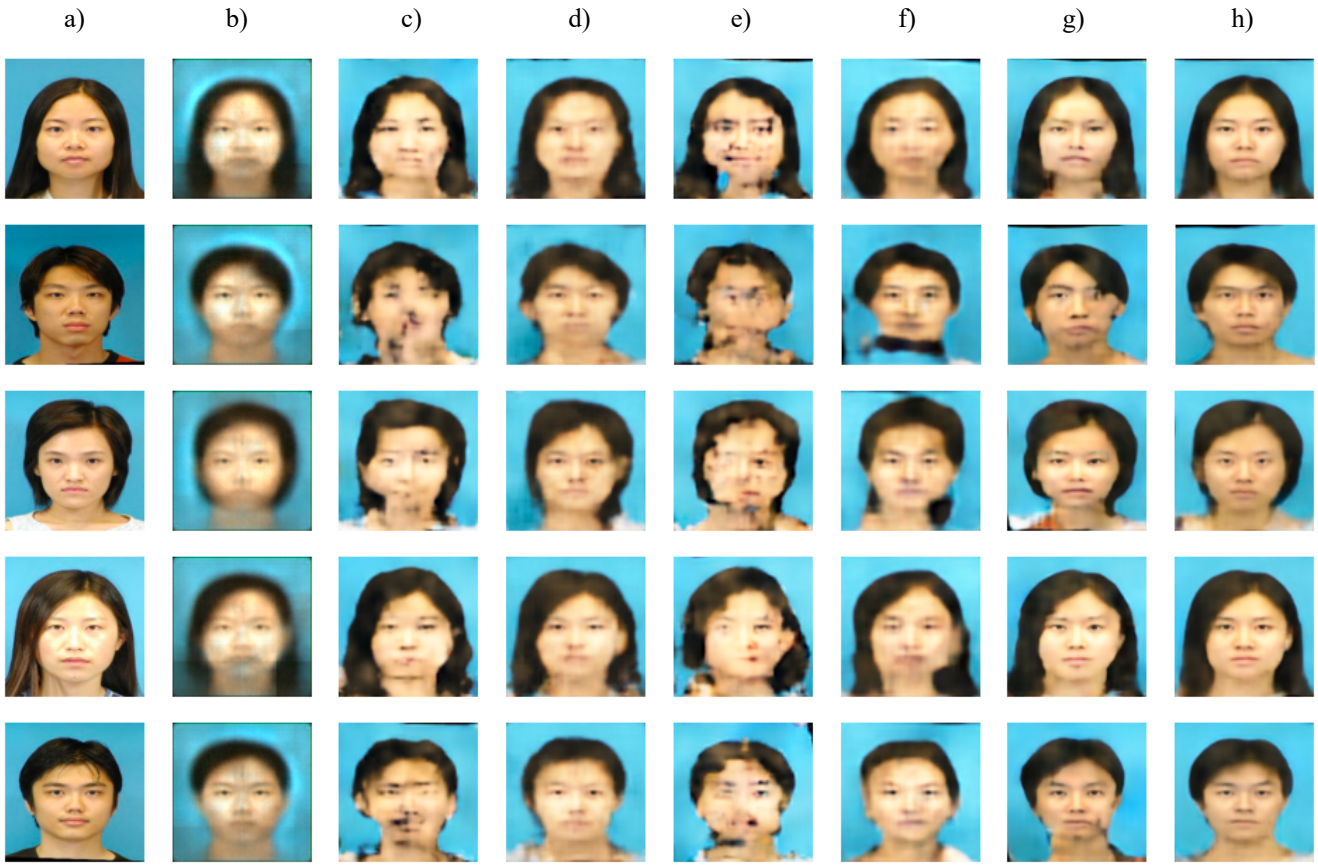

Fig. 11. a) True example test faces with Sketch2Face predictions: b) CNN, c) VGG16, d) VGG16 with dropout, e) VGG19, f) VGG19 with dropout, g) VGGFace, h) VGGFace with dropout

Model	RMSE (x10^{-5})	PSNR	FSIM	SRE	SAM	UIQ
CNN	**4.18**	**87.99**	**0.76**	**29.92**	0.75	**0.20**
VGG16	4.91	86.40	0.72	28.55	0.62	0.15
VGG19	4.92	86.39	0.73	28.54	0.63	0.15
VGGFace	4.91	86.39	0.74	28.54	**0.46**	0.17

Fig. 12. Image similarity measures across all models for Face2Sketch.

Fig. 13. a) True example test sketches with Face2Sketch predictions: b) CNN, c) VGG16, d) VGG19, e) VGGFace.

C. Test Results for Comparison

The 100 test images were passed through the learnt models for both Sketch2Face and Face2Sketch. The similarity measures as explained before were used to calculate scores for each image, and an average was taken as a summary of how well the models matched the true faces/sketches. Furthermore, sample images and the predictions are also provided to show to the human eye the quality. The image similarity metrics do not tell the full picture and being able to see the outputs for comparison is very helpful. Some models may produce better scores, but the outputs may not be capturing the images visually as well as others for example.

Sketch2Face

From the results in figure 10, it is conclusive that the VGGFace model with dropout performed the best on this problem, giving the best values across all measures. This is further evidenced by the output images, showing excellent reconstruction of faces from the sketches. Applying dropout and/or regularizers clearly improved model performance also, as metrics beat the standard models across all VGG versions. The CNN model, as expected, performed worse overall however results of metrics and images were still solid, and with more development and training time, could produce better outputs. The RMSE across all values was extremely small showing small loss, and all other metrics, especially for VGGFace with dropout, showed very good image generation. The UIQ values however showed some concern since they were not close to 1 however this is acceptable as the metric is often very sensitive and used mainly for slight differences in loss between images (e.g. JPEG conversion).

Figure 11 shows the differences in test samples produced from the models and visually shows the great performance of the VGGFace with dropout model. Genuine, realistic faces have been generated and although they do not perfectly match the original, they could quite easily fool a human. The original VGGFace also shows good results with obvious mistakes in some of the faces such as abnormalities in the shape and certain pixels missing. The same can be said for VGG16 and VGG19 but on much more extreme levels. The dropout/regularized versions of these models do show some accurate image cases, tending to be almost a 'blurred' version of the original image, but does struggle to generalize for all cases. Some of these weaker models also struggle with getting the gender correct clearly from the hair length. The longer hair is associated with the woman and so sometimes the genders clearly get disassociated from short haired woman to man and vice versa. The CNN faces capture general face and hair shape, but more specific features are repetitive for every image.

Face2Sketch

From the results in figure 12, it is conclusive that the CNN model performed best generating sketches. The example sketches further show that this model produced more realistic outputs and clearly there is further work to be done on the VGG models. The VGG models all performed extremely similar, with slightly worse scores in general than the CNN. Interestingly, VGGFace had a much better SAM score showing that perhaps the other outputs were overall affected by illuminance. Again, scores were similar to the Sketch2Face and on paper look good. Despite this, the actual example images in figure 8 show the poor sketches produced in the VGG models and the simplistic nature of the CNN sketches.

Figure 13 shows the sketches produced for each model. The CNN model clearly shows the best results but one could say they are too close to the original face. There is a lack of texture or lines whereas some of the other models do show this. General shape is good however there is a definite blur in the sketch quality of the VGG models. Again, gender is sometimes misclassified based on the hair whereas the CNN is immune to this since the features do not come from a pretrained model. There is often missing features such as eyes, noses or ears in the VGG16 and VGG19 model. The VGGFace model does show promise however, and with more parameter tuning and training, realistic sketches could become possible.

V. Discussion

Firstly, the Sketch2Face results were very promising, showing that transfer learning using VGG models is possible in generating faces from sketches. Research of the VGG models began with image colorization of the sketches by converting the RGB images into LAB images and trying to predict the AB colour channels from the L. This was then developed to instead generalize to inputting full sketches and mapping them to full faces using convolutional layers in a decoder. The VGG feature extraction allowed the decoder models to be easily trained quickly and experimented with leading to the discovery of dropout and regularizers to improve performance on test sets. This gradual process resulted in proving how effective these techniques could be as well as showing that transfer learning was possible on the problem. The VGGFace models used much smaller number of epochs to train with better results and shows that further training could lead to top-class image generation. As a result, VGGFace in particular can be used as a quick alternative to modern techniques such as GANs. Problems still do exist with some image generation, as mentioned above, but with more training time and better optimization of parameters this could be overcome.

On the other hand, Face2Sketch was worse, but some promising signs are there in the images. Clearly the shape and certain features were identified but the VGG models struggled to produce sketches on what should be a simpler task. Perhaps applying dropout or regularizer methodology like in the Sketch2Face models could result in better performance. With more time this could be explored since the CNN model did use these methods with better results.

The image similarity measures used back up the visual improvement in methodologies and give good evidence of the differences in performance across different criteria. The values obtained for the Face2Sketch were generous however. The overall results of the VGG models showed they were great at extracting features for complex reconstruction but were perhaps poor at simplifying facial features for sketches.

VI. Conclusion

This report has shown a comparison of CNN architectures from simple hand-made networks to pre-trained very deep VGG encoders. The use of transfer learning in combination with convolutional decoders has been applied on VGG16, VGG19, and VGGFace to produce quick training times with ease of construction. Furthermore, the use of dropout and regularizers have been shown to improve generalization of Sketch2Face. In particular, the VGGFace model with dropout gave excellent results producing very realistic faces that closely matched the original faces. The CNN was

limited on this problem however did perform best on the simpler problem of Face2Sketch. The VGG models did not perform as well on this problem as less attention was paid to it and so with further parameter fine-tuning, there is potential. One hypothesis for the failure was that the VGG models overcomplicated feature extraction for the sketches, but this needs to be further tested.

Due to time-constraints and computational performance giving slow training times, the project is limited but does show evidence of Sketch2Face generation in particular. The next steps would be to further develop the Face2Sketch performance, applying different techniques, and also apply the learnt models to further datasets outside the CUHK dataset to test further generalizability.

The link for the GitHub repository with all the code used for generating the 11 models and results is here:

https://github.com/benmullally/SCC413-Project-Repository

References

[1] Peng, C. et al. (2018) 'Face recognition from multiple stylistic sketches: Scenarios, datasets, and evaluation', Pattern recognition, 84, pp. 262–272. doi:10.1016/j.patcog.2018.07.014.

[2] Djelouah, A. et al. (2022) 'Training a Deep Remastering Model', Available at: https://studios.disneyresearch.com/2022/07/25/training-a-deep-remastering-model/

[3] Öztürk, C., Taşyürek, M. and Türkdamar, M.U. (2023) 'Transfer learning and fine-tuned transfer learning methods' effectiveness analyse in the CNN-based deep learning models', Concurrency and computation, 35(4), p. n/a–n/a. doi:10.1002/cpe.7542.

[4] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] Sadik, R. et al. (2023) 'An in-depth analysis of Convolutional Neural Network architectures with transfer learning for skin disease diagnosis', Healthcare Analytics, 3, p. 100143. doi:10.1016/j.health.2023.100143.

[6] Mohd Rum, S.N. and Rajaratinam, N. (2022) 'A Pothole Detection Using VGG16', Civil engineering and architecture, 10(3), pp. 1222–1232. doi:10.13189/cea.2022.100337.

[7] Krishnaswamy Rangarajan, A. and Purushothaman, R. (2020) 'Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM', Scientific reports, 10(1), pp. 2322–2322. doi:10.1038/s41598-020-59108-x.

[8] Peng, C. et al. (2016) 'Multiple Representations-Based Face Sketch-Photo Synthesis', IEEE transaction on neural networks and learning systems, 27(11), pp. 2201–2215. doi:10.1109/TNNLS.2015.2464681.

[9] Zhang, M. et al. (2019) 'Dual-Transfer Face Sketch-Photo Synthesis', IEEE transactions on image processing, 28(2), pp. 642–657. doi:10.1109/TIP.2018.2869688.

[10] Galea, C. (2022) 'Comments on "Domain Alignment Embedding Network for Sketch Face Recognition"', IEEE access, 10, pp. 71030–71034. doi:10.1109/ACCESS.2022.3188796.

[11] Y. Guo, L. Cao, C. Chen, K. Du and C. Fu. (2021) "Domain Alignment Embedding Network for Sketch Face Recognition," in IEEE Access, vol. 9, pp. 872-882, doi: 10.1109/ACCESS.2020.3047108.

[12] Yu, S. et al. (2023) 'CMOS-GAN: Semi-Supervised Generative Adversarial Model for Cross-Modality Face Image Synthesis', IEEE transactions on image processing, 32, pp. 144–158. doi:10.1109/TIP.2022.3226413.

[13] Zhang, M. et al. (2020) 'Bionic Face Sketch Generator', IEEE transactions on cybernetics, 50(6), pp. 2701–2714. doi:10.1109/TCYB.2019.2924589.

[14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. Communications of the ACM, 63(11), pp.139-144.

[15] Westerlund, M. (2019) 'The Emergence of Deepfake Technology: A Review' Technology Innovation Management Review. 9. 39-52. 10.22215/timreview/1282.

[16] Nguyen, T.T. et al. 'Deep learning for deepfakes creation and detection: A survey' Computer Vision and Image Understanding, Volume 223, 103525, ISSN 1077-3142, https://doi.org/10.1016/

[17] Yu, J. et al. (2021) 'Toward Realistic Face Photo-Sketch Synthesis via Composition-Aided GANs', IEEE transactions on cybernetics, 51(9), pp. 4350–4362. doi:10.1109/TCYB.2020.2972944.

[18] Peng, Y. et al. (2023) 'DiffFaceSketch: High-Fidelity Face Image Synthesis with Sketch-Guided Latent Diffusion Model'. doi:10.48550/arxiv.2302.06908.

[19] Bae, S. et al. (2022) 'Exploiting an Intermediate Latent Space between Photo and Sketch for Face Photo-Sketch Recognition', Sensors (Basel, Switzerland), 22(19), p. 7299. doi:10.3390/s22197299.

[20] Yadav, N.K., Singh, S.K. and Dubey, S.R. (2022) 'CSA-GAN: Cyclic synthesized attention guided generative adversarial network for face synthesis', Applied intelligence (Dordrecht, Netherlands), 52(11), pp. 12704–12723. doi:10.1007/s10489-021-03064-0.

[21] Ma, X. et al. (2023) 'Classification of seed corn ears based on custom lightweight convolutional neural network and improved training strategies', Engineering applications of artificial intelligence, 120. doi:10.1016/j.engappai.2023.105936.

[22] Xia, Y., Tang, M. and Tang, W. (2023) 'Fine-grained Potato Disease Identification Based on Contrastive Convolutional Neural Networks', Applied artificial intelligence, 37(1). doi:10.1080/08839514.2023.2166233.

[23] Parhki, O. (2015) 'Deep Face Recognition', British Machine Vision Conference

[24] Cao, Q., Shen, L., Xie, W., Parkhi, O.M. and Zisserman, A., 2018, May. Vggface2: A dataset for recognising faces across pose and age. In 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018) (pp. 67-74). IEEE.

[25] O'Shea, K. and Nash, R., 2015. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[26] Deep Clustering with Convolutional Autoencoders - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-structure-of-proposed-Convolutional-AutoEncoders-CAE-for-MNIST-In-the-middle-there_fig1_320658590 [accessed 9 Mar, 2023]

[27] Yamashita, R., Nishio, M., Do, R.K.G. et al. (2018) Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 https://doi.org/10.1007/s13244-018-0639-9

[28] LeCun, Y., Bengio, Y. & Hinton, G. (2015) Deep learning. Nature 521, 436–444 https://doi.org/10.1038/nature14539

[29] Simonyan, K. and Zisserman, A., (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[30] Khandelwal V. (2020) 'The Architecture and Implementation of VGG-16' Towards AI, Available at: https://towardsai.net/p/machine-learning/the-architecture-and-implementation-of-vgg-16

[31] DigitalSreeni (2020) '92 - Autoencoders using transfer learning - Image colorization' YouTube, Available at: https://www.youtube.com/watch?v=bIaT2X5Hd5k

[32] Bozinovski, S. (2020) 'Reminder of the First Paper on Transfer Learning in Neural Networks, 1976' Informatica 44 (2020) 291–302 https://doi.org/10.31449/inf.v44i3.2828

[33] Hinton, G. et al. (2014) 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting' Vol 15, 56. Available at: http://jmlr.org/papers/v15/srivastava14a.html

[34] Hinton, G.E. et al. (2012) 'Improving neural networks by preventing co-adaptation of feature detectors'. doi:10.48550/arxiv.1207.0580.

[35] Khan, S. et al. (2018) A Guide to Convolutional Neural Networks for Computer Vision. San Rafael: Morgan & Claypool Publishers.

[36] Serengil, S. (2018) 'Deep Face Recognition with Keras' Machine Learning. Available at: https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/