Creating Autoencoders for Face Generation

1 Introduction

Autoencoders are neural network models that compress input data into a low-dimensional latent representation through an encoder and then attempt to reconstruct it using a decoder. Autoencoders are unsupervised models that are typically used to extract features from the input for tasks such as sentence translation and image reconstruction. In many cases, an autoencoder architecture is used to train models which will be used for generative tasks. Here, the models are trained using both the encoder and decoder, but only the decoder is used for generation. This is common in image generation, where the latent vector is randomly sampled or specifically chosen to create a new image.

In this project, a range of convolutional autoencoders were trained on the CelebA dataset to generate pictures. The final goal of this project is to generate non-existent people, however, several model architectures were tested to examine a model's capability to reconstruct images and to generate faces. These variations include but are not limited to input resolution, latent vector dimension, and activation functions. Four final models will be presented, each showcasing distinct architectures with different strengths and weaknesses. The assessment criteria for each model include the realism of generated faces, including: a complete head, hair, a neck, accurate facial structure, and image clarity.

2 Dataset

The CelebA dataset consists of 202,599 facial images, split into 162,770 training, 19,867 validation, and 19,962 testing images, originally sized at 178×218. As the images are coloured, every image is separated into three channels representing red, green, and blue. Each image is assigned a label of 40 attributes spanning facial features such as 'big lips' and 'pointy nose', as well as characteristics like 'young'. The attribute labels are not balanced, with some appearing more frequently than others. The labels are not entirely accurate, with some images missing correlated labels such as 'double chin' but not 'chubby' [5]. The faces in the images are not always centred and visually clear, as some contain external features that cover the face, like hands, trophies, or drinking glasses. This can cause problems during image reconstruction, as the models cannot correctly identify the face. This does not affect image generation as there are not enough unique images to affect a model's parameters.

3 Architecture

3.1 Design Considerations

Throughout development, several model architectures were tested to check different component configurations, such as activation functions and filter sizes.

The first step involved selecting the input resolution. A variety of image sizes were considered: 32×32, 64×64, 96×96, and 128×128. Greater sizes require larger depth models to use effectively, which increases the computational costs significantly. 32×32 images appear too blurry and do not have

enough pixels to capture finer details in facial structure, while 96×96 were avoided to ensure clean downsampling was possible during the convolution layers to 1×1. The chosen resolutions are 64×64 and 128×128. 64×64 images are a middle ground between the other resolution sizes, which is useful for a less demanding model. As there are four times as many pixels, facial detail should sharper than in the 32×32 models. Some graininess is still visible, but the facial structure is recognisable. The 128×128 resolution images are four times larger than 64×64 pictures. The added pixels should lead to a better understanding of finer facial details, and the projected images should be sharper. The difference between these resolutions is seen in Figure 1, where the larger resolution shows more detail and better image quality. It is important to mention that the smaller the image scale, the less this difference is seen.

Figure 1: CelebA testing images showing 64×64 and 128×128 resolutions.

In the design phase, the activation functions that were tested were ReLU and Leaky ReLU, defined as $\text{ReLU}(x) = \max(0, x)$ and $\text{LeakyReLU}(x) = \max(\alpha x, x)$ where x is the input and α is a constant. ReLU is the simplest activation function that adds non-linearity while also being a solution to the vanishing gradient problem that other functions suffer from, such as sigmoid and Tanh. However, since ReLU sets all negative values to 0, it can cause the dying ReLU problem. This occurs when negative values pass through ReLU, causing neurons to output 0, leading to no parameter updates during gradient descent. Leaky ReLU is a solution to this issue. Negative values pass through the activation function as αx, allowing neurons to contribute in learning, effectively making the model stronger.

The activation function in the output layer is always the sigmoid function. All of the images were scaled to the range $[0, 1]$ and sigmoid maps each input to $[0, 1]$. This makes it easier for direct comparison between the output and the true output during reconstruction training. Additionally, in classification of independent labels, sigmoid is used to give the probability of each label.

Batch normalisation is a model component that addresses the internal covariate shift problem. The distribution between layers can change during training, which can result in less stability and slower learning. By normalising the outputs per batch after every layer, the distribution remains the same throughout the network. This stops the original issue but creates a new one, since the normalised values can be below 0. This means Leaky ReLU should be used to avoid the dying ReLU problem.

Downsampling occurs in models to try to extract features from different segments of the input while condensing important prior information. The methods considered were max pooling and strided convolutions. Max pooling involves splitting the input into patches and picking the maximum values for each patch [1]. Strided convolutions reduce input size by applying filters that skip positions in the image, resulting in downsampling while extracting learnt features. Max pooling can improve shift-invariance, a model's ability to produce good outputs when the input is slightly changed, but it is possible for important features to be discarded and no learning occurs during pooling. Strided convolutions allow the model to learn which features to keep during downsampling, increasing the performance of the model, especially on larger inputs. However, stride can cause lower accuracy if downsampling occurs too quickly, as important information can be missed [1] [3]. For upsampling, only strided transposed convolutions were considered. Since the decoder is responsible for image generation, learning to upscale images with more features should result in better generated images.

Padding is an extra part of a convolution layer that maintains the input size during a convolution

layer. Without padding, convolution layers reduce input size by ignoring pixels on the outside of an image. This can lead to unnatural sampling and loss of features. With padding, the spatial size is maintained, so the features are kept and sampling is smoother.

The number of channels in each convolution layer determines how many features the network can learn from the input. In early testing, the number of channels ranged from approximately 10 to 100, but this proved to be too few to express the complexity of facial structures. In later models, the channel counts ranged from 8 to 1024 based on input resolution. 128×128 images required larger models as there were more features to extract. Models with better generation tended to be larger, however, the difference between 512 and 1024 channels was not visible enough to justify the additional training time. Additionally, the difference between the encoder and decoder channels likely leads to better image generation. A larger decoder will reconstruct data more accurately, which should lead to better image generation. A larger encoder should produce structured outputs for the decoder. If this is sampled for generation, the decoder will not understand the latent vector and will create poor images. A less trained model may perform better for image generation but worse in reconstruction.

The encoder output, a latent vector, is an important design choice. A mixture of sizes was considered from 40 to 1024. For image reconstruction, a larger bottleneck leaves more information for the decoder to use, leading to better reconstruction. For image generation, a smaller latent space is better as only the most important information is saved, in this case, facial features. When generating images, the latent vectors are sampled using a distribution from the encoder's output. Larger latent spaces can confuse the decoder as there will be a range of values it has not seen during training, leading to distorted images. Smaller latent spaces reduce this problem.

An interesting idea involved using an encoder trained for classification and transferring it to image reconstruction. The labels learnt during training could help create a better latent representation of an image, which should help in reconstruction training for the decoder. As CelebA includes labels for each image, it should be possible to create an autoencoder model in this style, but there is a misrepresentation of the labels in the dataset, which could be an issue. This model could also be used to create custom made generated faces by selecting specific attributes. However, randomly sampled latent vectors will cause an issue for the decoder, since it will expect realistic vectors, whereas random sampling will lead to impossible attribute combinations.

3.2 Discarded Models

Through many different model designs, there have been aspects of model architecture that have been considered but have led to consistently poor results. These models may be useful for image reconstruction, but not for image generation which is the desired goal. Dropout is a regularisation technique used to prevent overfitting and provides the network with a nonlinear path to the output. In testing, models that included dropout performed worse than models without it.

A trialled model excluded the use of padding in both the encoder and decoder. Excluding padding in the encoder does not cause problems with image generation, however, the exclusion of padding in the decoder severely impacts it. In Figure 2, the output contains pronounced square-shaped regions that disrupt facial fluidity.

Figure 2: Generated images from an autoencoder model using no padding in the decoder.

A series of discarded models tested different bottlenecks for the latent vector. As hypothesised, the larger latent vectors produce good image reconstructions but fail in image generation. Most of the generated images are distorted and contain only segments of facial structure, mainly the mouth or eyes. The images are also scattered in colour because of the random noise from sampling. In Figure 3, there

is partial success in the fourth image, as some facial features: the eyes, mouth, nose, and eyebrows, are clear, but the rest of the face is not generated.

Figure 3: Generated images from an autoencoder model using a sampled 512-dimensional latent vector.

3.3 Final Model Architectures

The final four models were designed to maintain a balance between architectural differences and similarities to ensure that a level of comparison is possible. Each model was made for image generation as the final task to complete.

3.3.1 Autoencoder Model 1

Model 1 is a baseline autoencoder trained on 64×64 resolution images. It features a pretrained encoder for classification of the 40 CelebA attributes, meaning the latent vector dimension is 40. The encoder uses a sequence of six convolution layers, with 8 to 256 channels, while downsampling to 1×1, followed by two linear layers. To ensure clean downsampling, 3×3 filters are used in combination with padding. ReLU is used with max pooling to create the most efficient and smallest model. The decoder is built symmetrically but uses strided convolutions for upsampling. In the transposed convolution layers, there are 4×4 filters with padding for clean upsampling. Sigmoid is used in the encoder from the classification pretraining, and is also used as the decoder output for easy comparison between the true $[0, 1]$ inputs.

3.3.2 Autoencoder Model 2

Model 2 is an autoencoder trained on 64×64 resolution images without a pretrained encoder. Strided convolutions are used in the encoder to keep downsampling features. No padding is used to reduce the spatial size faster. As a result, there are five convolution layers, ranging from 16 to 256 to remain comparable to Model 1. The latent vector size increases to 84 to preserve more information. The decoder is elongated, containing an extra linear and transposed convolution layer. The deeper channels from 512 to 16 should result in better generation. Like in model 1, the same filter and padding sizes are used for clean upsampling. ReLU is used throughout the entire model, with sigmoid as the output function in the decoder.

3.3.3 Autoencoder Model 3

Model 3 is the first 128×128 input model and features a pretrained encoder similar to Model 1. As the inputs are larger, seven strided convolution layers are used with channels ranging from 8 to 512. The filter sizes in the early layers are 5×5 with two padding to capture broader image details, while the deeper layers have 3×3 filters with one padding. The padding is used for clean downsampling. ReLU is used throughout the model, but to accommodate the larger layers in the decoder, LeakyReLU is used to keep neurons stable. Apart from that, the decoder is symmetrical with the encoder. Sigmoid is used at the output of the encoder and decoder. Unlike Model 1, the encoder will be frozen during training to preserve learnt classification features.

3.3.4 Autoencoder Model 4

Model 4 is a 128×128 input model. There are six convolution layers, downsampling to 2×2 instead of 1×1 to prevent the channels from being too large. The number of channels ranges from 16 to 512. The latent vector dimension is set to 128, which accommodates the increase in input size. This model uses the same filter and padding sizes as Model 1, but after each layer, batch normalisation occurs in tandem with Leaky ReLU. This is done to stabilise the model, especially the output distribution for the encoder. The decoder mirrors the structure of the encoder, but it is possible to increase the number of layers and channel ceiling to seven and 1024, which should make the model better. Sigmoid is used in the output of the decoder.

4 Training

Training involves several components that need to be considered before model training begins. A series of batch sizes was tested, ranging from 64 to 512 for each model. Larger batch sizes are processed faster, equating to less time per epoch, but there are fewer parameter updates per epoch, resulting in slower training. Based on practical experience, a 128 batch size provides a good balance between learning speed and time spent per epoch. Each model uses the same dataset, but with different resolution images. The dataset is split into training and validation sets, where training is used for model learning and validation is used to spot issues like overfitting. The testing set was exclusively used for final accuracy tests or image reconstructions. In each model, including encoder pretraining, the learning rate is different and has to be optimised. To choose the appropriate learning rate, the models were trained up to 20 epochs from a range of $[0.002, 0.0001]$. Learning rates above and below these values cause overshooting or very slow learning.

Models 1 and 3 were pretrained for attribute classification of 40 separate classes using binary cross-entropy loss. Each model was given a maximum of 30 epochs, as most models converged earlier, and further training resulted in diminishing returns. The encoder models were tested for individual label accuracy for the entire testing set.

For the autoencoder, each model was trained to reconstruct the input. In this task, both the mean squared error loss (MSE) and binary cross-entropy loss (BCE) can be used in training. MSE measures the pixel difference between the output and the true output, while BCE measures the probability for a pixel to match the normalised output. MSE treats all pixels equally, which can result in poor reconstructions of important facial features, while BCE treats wide disparities between the output and the true output with more severity. MSE creates more detailed reconstructions, however, the results are less consistent, and the generations are worse than when using BCE. BCE is the loss function for each model. Testing each autoencoder model required both visual inspection of the reconstructed and randomly generated images.

The maximum number of epochs for training is set to 100. As this dataset has a diverse range of images, the models struggle to converge over many epochs. From observation, larger models consistently learnt over 300 epochs, however, the loss and visual difference between 100 and 300 is small, for both MSE and BCE. Additionally, the encoder for Model 3 was frozen during training to prevent learning new information.

5 Results

The performance of each model is evaluated across three sections depending on the model architecture: attribute classification, image reconstruction, and image generation.

5.1 Attribute Classification

Model 1 and Model 3 were designed to include an encoder trained for attribute classification on the CelebA images. Model 1 was trained for the full 30 epochs at a learning rate of 0.0005. This occurred for 36.37 minutes with an average of 1.21 minutes per epoch. This model did not overfit but showed steady training until epoch 15, when learning slowed. The minimum validation loss obtained for similar models was around 0.2220, while Model 1 obtained 0.2265. In terms of accuracy per label, Model 1 secured 89.60%, which is similar to other models. In variations of this model, strided convolutions lead to worse results than with max pooling, however, the difference is around 0.3%.

The encoder for Model 3 converged by epoch 17 at a learning rate of 0.0003. The total training time took 22.92 minutes with 1.35 minutes per epoch. This is longer per epoch than Model 1, but that is to be expected. This model converged at a validation loss of 0.2178 with an accuracy of 89.98%, which is 0.38% better, or quantified to 3035 more correct labels based on the testing data and 40 attributes per image.

5.2 Image reconstruction

The results for each model contain a final validation loss after training. This value is not a complete representation of a model's performance for reconstruction. A model can achieve a lower validation loss by reconstructing background colours rather than having a detailed face. A good reconstruction does not mean that a model will do well in generation.

Model 1 was trained for 100 epochs at a learning rate of 0.001. The total training time was 122.77 minutes, averaging 1.23 minutes per epoch. This is the smallest model which takes the shortest amount of time to train. The final validation loss was 0.5107. The model did see constant fluctuations in validation loss, but overall learning decreased at a steady rate. Model 2 was trained for 100 epochs at a learning rate of 0.00065. The training time for this model was 132.93 minutes and 1.33 minutes per epoch, which is slightly longer than Model 1 due to the larger decoder. The training appeared smoother than in the first model and ended with a lower validation loss of 0.5043.

Model 3 is the first model with high input resolution. This model was trained with a frozen encoder, resulting in fast overfitting. The model was observed for 25 epochs, while convergence occurred at epoch 10 with a validation loss of 0.6358. This model was trained at a learning rate of 0.0005 for 13.45 minutes and 1.35 minutes per epoch. Attempts to reduce the validation loss were made, including parameter tuning, decreasing and increasing the capacity of the decoder, and changes in architectural design. These changes did improve the validation loss, however, the improvements were not large enough to justify changes. Model 4 is the largest model contributing to the longest training time of 138.03 minutes and 1.38 minutes per epoch. This model was trained for 100 epochs at a 0.00045 learning rate, obtaining 0.4927 validation loss. The validation loss is expected to be the lowest in this model since it is the deepest. The total training time across 100 epochs did not change very much per model, but the time per epoch did. If the models were trained until convergence, the time difference would be significantly larger.

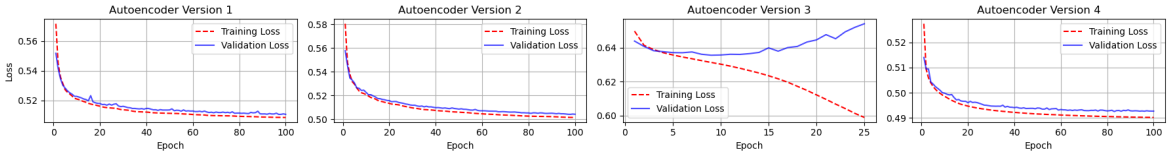

Figure 4: Training and validation loss for each autoencoder model.

In this case, the validation losses for each model present a clear ranking for image reconstruction as seen in Figure 5. Model 1 creates blurry but fine reconstructions. The colouring and positioning of the faces are mostly accurate, but the faces themselves lack detail and are generic. There are no signs

of pretraining, which means the classification parameters do not transfer to reconstruction. Model 2 creates sharper reconstructions, which is expected from a deeper model. When inspected, the facial expressions are better than in Model 1 as they are less generic, however, they still do not match the original. The positions of the faces are more accurate, and the backgrounds tend to match the original. This is most likely due to the larger latent vector.

Model 3 is the worst reconstruction model. The backgrounds are not reflected, and the positions of the faces are incorrect. This is due to the frozen encoder. In these images, common attributes from classification are represented, such as hair colours and open mouths. The faces rival Model 2 reconstructions, as some are more detailed, but in general, this is a poor model. Model 4 creates the best reconstructions. This is expected from the latent vector size, as more details are captured, especially in the hair flow. This can be seen further in the clothing as the colours and shape are more accurate than others, for example, the headband in the last image. This model creates the sharpest faces that mimic the original facial expressions to a good degree.

Figure 5: Original images followed by reconstructions by models in ascending model order.

5.3 Image Generation

The final goal of this project is to create a model that can create non-existent human faces. To do this, the latent vector is sampled by taking the encoder output from each autoencoder. The mean and standard deviation for the outputs are computed, and new latent vectors can be randomised using those statistics. In Figure 6, there are 12 generated random faces for each model.

Model 1 creates good generations of faces. Most of the faces are not cut off, and there is a neck that makes the image more realistic. The skin tone is well done, and there is even lighting in common places like the forehead. The hair, although textureless, is correctly shown in most images with suitable hair colours. The small latent vector is most likely the reason for good image generations. Model 2 has interesting image generations. The faces can be more detailed than in Model 1, but in general are more blurry. The larger latent vector is responsible for some of the background, as there are far more colours than in the first model. Another problem with this model is the hair, as it meshes with the background and tends to lack volume when present. Another problem with these images is the black eyes, which are most likely supposed to represent some sort of glasses, like in Figure 5.

Model 3 has very good facial generation. Overall, the faces created are better in image quality and are more detailed than any other model. This is partly due to this model being in a higher resolution, so the images will naturally be clearer. There are no problems with colour mixing, and the hair is realistic. Apart from the overexposure in some images and some issues with black eyes, this model has few flaws. Due to the frozen encoder, the decoder receives plausible values that could represent a real person, practically mimicking the 40 attributes. It is possible that these images could come off as low quality prison identification photos due to the clean backgrounds. Model 4 creates the least blurry images however, the faces themselves are not good. In each image, parts of the face are cut off, and there is little hair, which is a large part of what makes these pictures realistic. Most likely, the large latent vector captures hair across multiple dimensions, which means that multiple values need to line up correctly to get all the hair. This is also probably the case for the missing faces. There are a lot of

random colours that blend with the faces, however, this is on the lower end of this problem thanks to the normalisation of batches.

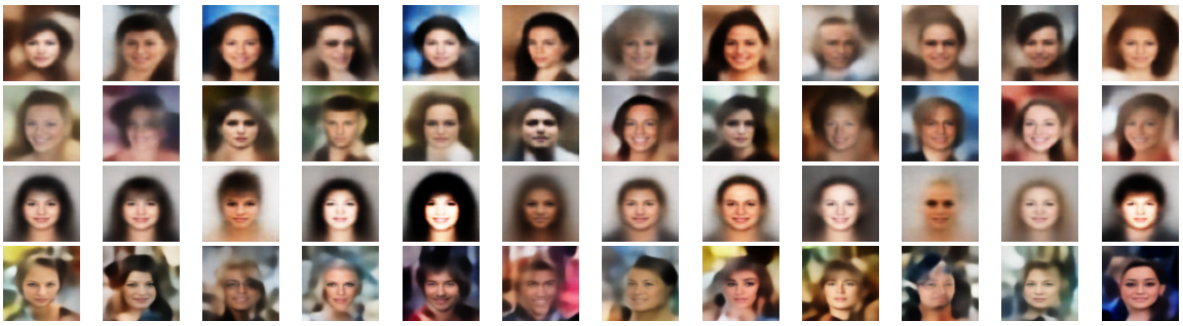

Figure 6: Generated images using sampled latent vectors by models in ascending model order.

The images in Figure 7 were made using a combination of attributes for the 40 CelebA attributes. The first three images are real combinations of attributes that can appear in the CelebA dataset, while the next three are impossible combinations, such as all categories or none. Both models received the same binary input vectors. Model 1 generates poor images as the encoder moves away from the strict outputs that represent the labels. Although the faces are visible, it is only an outline, and no real features are clear. Model 3 does a better job of showcasing the idea of custom inputs, as the faces partially match the expected output. The first face has feminine traits with black hair, while the second and third are males of different ages. However, it is interesting that two of the three impossible images display clear faces with visible features, since the encoder would never recreate this input.

Figure 7: Generated images using custom 40 attribute latent vectors for Model 1 (1-6) and 3 (7-12).

6 Conclusion

In conclusion, the best model for generating pictures of non-existent people is Model 3. Although the images are not as sharp as Model 4, the faces are shown in full consistently with different expressions and hair, making them realistic. While models 2 and 4 have worse generation, parts of their architecture can be used to improve Model 3. The deeper decoder from Model 2 is useful in more detailed reconstruction, and the normalisation layers from Model 4 help in training stability and consistent distribution. By increasing the number of channels in the convolution layers, the model should be able to learn more features in the encoder during classification. This would lead to a better latent representation of the input, which should lead to better decoder processing. As CelebA attributes are broad, it limits the potential of this idea. By creating more categories with more depth, such as different emotions, ethnicity, and skin tone, facial generation would significantly improve. This also addresses the skin tone issue in generations where darker skin tones are often whitened from the imbalance in the dataset. Another potential improvement includes additional training for the decoder by using the expected attribute outcomes and training for reconstruction. This would be similar to what modern generative models do, but on a significantly more basic level [4]. Although significantly more complex, it should be possible to use a super-resolution model to increase the resolution of a generated output, which should help with image clarity [2] [4]. Hair generation seems to be a consistent issue in each model, as it is less structured than facial features. Due to its high variability, hair is likely expressed as noise similar to the background. Model 3 handles hair well because it is an attribute, and the background is completely ignored. A potential fix to this problem would be to crop out the head in each image so the model can only focus on that.

References

[1] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[2] Brian B. Moser, Federico Raue, Stanislav Frolov, Sebastian Palacio, Jörn Hees, and Andreas Dengel. Hitchhiker's guide to super-resolution: Introduction and recent advances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9862–9882, 2023.

[3] Rachid Riad, Olivier Teboul, David Grangier, and Neil Zeghidour. Learning strides in convolutional neural networks, 2022.

[4] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[5] Haiyu Wu, Grace Bezold, Manuel Günther, Terrance Boult, Michael C. King, and Kevin W. Bowyer. Consistency and accuracy of celeba attribute values. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3258–3266, June 2023.