Radboud University

Visualising Learnt Representations of k-GANs

Thesis BSc Artificial Intelligence

Author:
Rory Quinn-Bailie

Supervisor:
Umut Guclu

July 2019

Contents

1 Introduction

1.1 The Black Box Problem

Neural networks have been a powerful advancement in the field of Machine Learning, profoundly transforming the state of the art in several important areas of machine learning research[1][2]. With the growing popularity of these networks, specifically deep neural nets (DNNs), increasing the comprehensibility of these technologies is of vital importance now more than ever. DNNs have had successful application in many real-world areas, such as the medical domain [3]. There are wide-ranging implications concerning the increased application of deep learning in the real world, and the lack of transparency of DNNs is a pressing and complex issue that is prevalent with this application. As long as the internal representations and learning process of these networks remain opaque, DNNs will encounter restrictions and obstacles in several fields. Other issues such as the relationship between data and privacy also intertwine here[4]. Lack of interpretability is a well-known shortcoming in deep learning, and the reliability and usabilility concerns associated with this are well-established[5].

This issue contains deep intricacies that are beyond the scope of this paper, but at one level the problem involves a lack of explanation of the causal connections in a neural network, as well as an absence of insight into the decision processes behind any conclusions a network may draw. This is aptly referred to as the **Black Box** problem[6] (Although neural networks are not complete black boxes in the sense there is some interpretability of their behaviour, see 1.2). This is a vulnerability that leaves deep learning applications prone to multiple problems. One such problem is adversarial attacks, which have demonstrated that the robustness of deep networks is a lot weaker than previously thought [7]. Here changing a few pixels values of an input image, for example, completely shifts the classification probabilities of a network, while the change between original and adversarial image is unrecognisible to the human eye. Therefore, due to the extensive internal structure of a DNN, with its potentially millions of parameters, network transparency proves a complex and critical problem to solve, but also an extremely important and rewarding one. One tool that can give aid into beginning to resolve this extensive challenge is visualisation techniques.

1.2 Using Visualisation to Better Understand DNNs

The importance and the pressing nature of improving neural network transparency has been introduced and established. Next, the question of how to tackle this problem naturally arises. Can one open the black box of DNNs and if so, what tools are necessary for this task? There are several approaches that have been researched and applied with some success in illuminating DNN learning and furthering Explainable Artificial Intelligence[8] [6][7]. These works have illustrated that indeed neural networks are not complete black boxes, in other words that their workings are somewhat interpretable. Interpretation is prone to error however, and this work has only begun to excavate the area of transparency in deep learning.

The approach of utilising visualisation techniques is of growing popularity and has seen some illusive applications in the field of deep learning [9]. The research area is young and has already seen impressive discoveries. One possibility in apply visual techniques to DNNs is to visualise learned representations of a network. In other words, what features the network has deemed important its respective task and has therefore encoded into its weighted connections, whether on the neuron or layer level. Visualisation of learned representations is one effective approach to gain insight into the internal mechanisms of DNNs, whether during or after training the network.

1.2.1 Previous Work

In terms of increasing network transparency, DNN visualization has already proven extremely useful in gaining insight into the inner workings of neural nets[10]. Great strides have been made in improving the intricacies that feature visualisation entails, namely optimisation and regularisation methods [10]. Multiple approaches have been made in visualising components of DNNs. Feature visualisation can be done at the level of a single neuron,taking a whole layer or , in the case of convolutional nets, visualising an entire channel. The level at which one intends to investigate the networks inner workings will determine which technique is applied. Taking the approach of visualising layer by layer, for example, can help build an insight into how a DNNs representations develop as one traverses deeper into the network. These techniques have resulted in valuable progress for this direction of research [10]. Visualisation is a powerful tool that has been developing steadily and may prove deeply beneficial in the long process toward building a more complete picture of DNNs.

In terms of a specific example,Olah et al's comprehensive paper is complimented by an interactive website that truly shows the power of a tool like visualisation. The authors explore optimisation as a method of effective feature visualisation, emphasising the different facets of this approach and its challenges [10].

The process of visualisation involves reconstructing features encoded inside the layers of a DNN. Methods of inverting representations have been successfully applied to obtain knowledge about what information these layers encode [11]. This conceptually straightforward and elegant approach consists of taking an input of random noise and to adjust this input such that the activation of a certain neuron or layer is maximized [12] (example in figure 1). This technique alone is relatively ineffective at giving clear visualisations however, but has be improved upon greatly via regularization techniques, such as taking a natural image prior[13] [9]. Without this prior maximally activated inputs are often unrecognizable[14], making visualisation with this method alone inefficient and inaccurate. However, with tools like regularization, such as taking a natural image prior when updating the values of the (initially random) pixel vector, this approach has been shown to create incredibly useful visualisations. This improved route to visualising has been a substantial step in the research field. This has lead to researchers being able to draw inferences about the networks learning process. Yosinki and his colleagues showed that DNN representations are interpretable even at the neuronal level, as well as showing the network learns features not explicity trained for (such as text detectors when being trained on discerning if an image is a library). This is a clear indication of promising beginnings for DNN visualisations and includes concrete examples of when and how this method is useful [9].

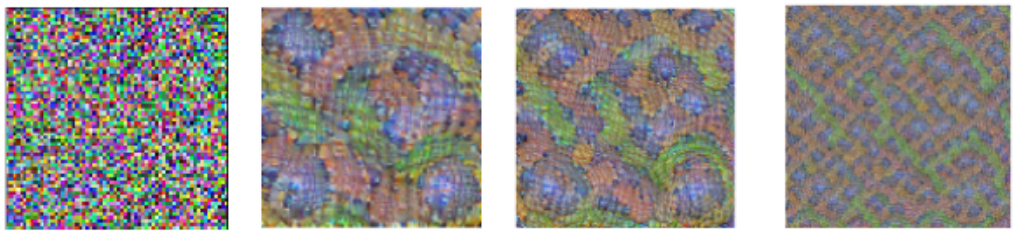


Figure 1: An initial image of random noise being optimised for a particular neuron of VGG-16 (2.1.4)

1.3 GANs - Learning & Challenges

One specific type of DNN that can benefit greatly from visualisation is GANs. Generative Adversarial Networks (GANs) have established themselves as a powerful tool in the field of machine learning. In the relatively short period of time since their popularisation they have had numerous compelling applications, especially in the sphere of photo-realistic image generation [15][16].They have also been applied with great success to image reconstruction[17], text-to-image synthesis [18] and a wide variety of other domains [19]. The improvement in the state of the art is swift and impressive. This makes the research field of GANs a compelling one to participate in and also makes GANs an area fit for attempting to create transparency and gain insight into their learning process. GANs involve regular classification, a popular application of neural networks, but also data generation. This creates complexity when quantifying GAN learning, and visualisation techniques can help give clarity in this regard. Specifically this paper's goal is to use these techniques to visualise the learnt representations that generative adversarial networks encode.

1.3.1 GANs Explained

The goal of a GAN is to model some probability distribution. If trained correctly, the model will be able to successfully generate realistic samples from that distribution. For example, a GAN could be trained on a dataset of pictures of faces, modelling the distribution of that data. After successful training it would be able to generate new faces not seen previously in the dataset. GANs consist of two adversarial networks, competing against over a shared cost function. One network attempts to minimise while the other attempts to maximise this objective function. One of these networks is the discriminator, whose role is to successfully differentiate between actual data samples taken from the dataset, and samples created by the generator. Its output is therefore a probability value that the given sample is real. Hence, the generators mission is to fool the discriminator by producing realistic samples. The objective function is used to measure the difference between the real data distribution and the one created by the generator model..

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\big[logD(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[log(1-D(G(z)))\big]$$

Above is the objective function utilised in this paper, taken from the DCGAN implementation [20]. Here the generator is attempting to minimise an error function while the discriminators goal is to maximise it. This formula consists of the summation of two terms. The first, $\mathbb{E}_{x\sim p_{data}(x)}\big[logD(x)\big]$, is the discriminator attempting to maximise the probability value the discriminator returns for data sampled from the real distribution. The second term, $\mathbb{E}_{z\sim p_z(z)}\big[log(1-D(G(z)))\big]$, the discriminator tries to minimise the probability value given for data sampled from the generator. Therefore the discriminator's goal is to maximise the objective function. The generator's aim is the opposite, namely to minimise the first term and maximise the second. If these networks compete successfully and with stability, the generator successfully models the data distribution and be used to generate indistinguishable samples (at least indistinguishable to the discriminator- the quality of data provided and the power of both networks constitutes whether or not generated data is indistinguishable by the human eye).

1.3.2 Challenges in GAN Learning

There are numerous trade−offs and questions to be considered when training GANs [21]. The original GAN model [22] encounters several potential issues during training, some of which are hard to identify and counter[23][24]. Specifically, GAN training encounters frequent issues in terms of the general instability of training. This makes the process of creating generative models a potentially slow and challenging one, and their implementation requires much tuning and trail-and-error. Such issues include non-convergence of the objective function, vanishing gradient problems and mode collapse. Mode collapse is expanded upon further in the next section(1.4), an interesting problem that this paper aims to give insight into.

Given these concerns in the effective and efficient implementation of GANS, within the sphere of generative neural networks there has been deep interest and effort shown in tackling these issues. Out of this research several effective solutions have been created, being applied to improving GAN stabilisation, as well as to eliminating the vanishing gradient problem, and effective approaches to combating the mode collapse problem encountered by the generator [25] [26] [27].

The evaluation of GANs is also a challenging topic. Reaching a conclusive clear metric is difficult compared to, for example, a DNN trained for image recognition, where the error between a predicated class and the actual class is intuitively computable. In generated class samples, effectively quantifying just how realistic generated samples are proves a difficult and poignant issue. The diversity of generated samples is also important so that the network accurately models the given data and not just a subset of its classes.One widely-used metric for this measurement is Inception score[27]. This is useful in that it measures both the quality of samples within a class as well as the diversity of samples produced. Keeping evaluation empirically grounded is of course essential as GANs continue to develop [28].

1.4 k-GANs

As mentioned in section 1.3, one difficult challenge commonly encountered in GAN training is that of mode collapse. This issue relates to a lack of sample diversity which implies the generator does not accurate model the actual distribution of the data. This is due to the fact that data is almost always multimodal. The generator may learn to model only one mode or cluster of the data distribution, and may even switch between clusters during training. This proves a near-impossible problem to note during training and difficult to identify and fix.

There have been several solution proposed in the literature to solve this problem. These variations of GANs have been implemented with quite a bit of success, eliminating or greatly reducing mode collapse[18][26][24]. One such solution is to use an ensemble of GANs to model the data-space.

k-GANs is one such ensemble of generative networks. The authors aim to improve upon the empirical aspect of ensemble techniques in GANs. They apply optimal transport theory [29] to do this, ensuring each k in the ensemble models a differing section of the data space. In order to investigate how each k models the data space, this paper aims to visualise the learnt representations of each GAN in the ensemble. This may lend some insight into the nature of the specific k-GANs model as well as ensemble GANs in general.

A more specific route to explore this issue is to draw a comparison between the visualisations created for each k generator-discriminator pair of the k-GAN ensemble and those created for one GAN that models the whole dataspace (in the interest of fairness this is simply a k-GAN model where $k = 1$). In order to create a fair comparison k of these single ($k = 1$) all-data GANs will be trained and their models combined (where

k is the number of GANs in the k-GAN ensemble). Visually comparing the results of encoded representations for this seperate models, as well a difference in performance between these models could be an indication that k-GAN clusters are modelling separate parts of the data space. Initially visualisations could give insight into the specifics of this learning, lending aid to the k-GANs authors work.

2 Methods

In this section the specific process of visualisation utilised in this paper is explained. The pipeline structure of the visualisation mechanism is illustrated with figures and a step-by-step description of each part of the process is given for clarity. The architectures of the models used in this process are also introduced and an overview of their structure is given, along with motivating their reason for being chosen.

2.1 Network Architectures

2.1.1 k-GANs

A k-GAN ensemble consists of a collection of Wasserstein GANs, a variation of GAN that increases the general stability of vanilla GAN learning as well as mode collapse issues[26]. k of these GANs model are trained on separate, non-overlapping partitions of the data. Choosing the number of GANs (k) to train requires hyperparameter tuning. A higher k will also be resource intensive, which influences this papers choice of ensemble size. Each discriminator consists of two convolutional layers and a final linear layer, producing a scalar probability output. The generator network has a linear input layer with three deconvolutional layers following, outputting a generated image.

2.1.2 Visualisation Ensemble

The visualisation process will require an array of networks as pictured in figure 2. As mentioned in 1.2.1, maximising activation by backpropagating error directly to the input image may produce unclear and uniterpretable images. A prior network will therefore be used to increase the realism and interpretability of the visualisations. The prior specifically used in this paper is a generator network taken from the DCGAN model[20].

This generator is trained to reproduce an input image given some activation vector from a layer. In other words, when an image is passed through a network, it causes some activation in each layer. Attempting to reconstruct one of these layer's activations is the role of the prior model. In the example 2, the network is trained on face data. This means when a random latent vector is passed as input to the generator, it produces some random output. Instead of random pixels however, the image produced contains some facial structure, e.g it is constrained by the prior to have some semblance to an image of a face.

Here the generator network acts as the prior for maximising the activation of some layer j in the k-GAN model, thus visualising what representations that layer finds important and is encoding. An initially random input image is generated by the prior (taking a random latent vector as input) and incrementally updated to maximise activation. The image is passed through the network one is visualising, and the activation from layer j is computed. Using the maximisation of the average activation of this layer as a loss function, the latent vector is then updated, the prior then generating an image that causes higher layer activation. Through this process a visual representation of the encoded features at layer j of the k-GAN discriminator can be produced. From this, insights and observations can be made about the networks learned features.

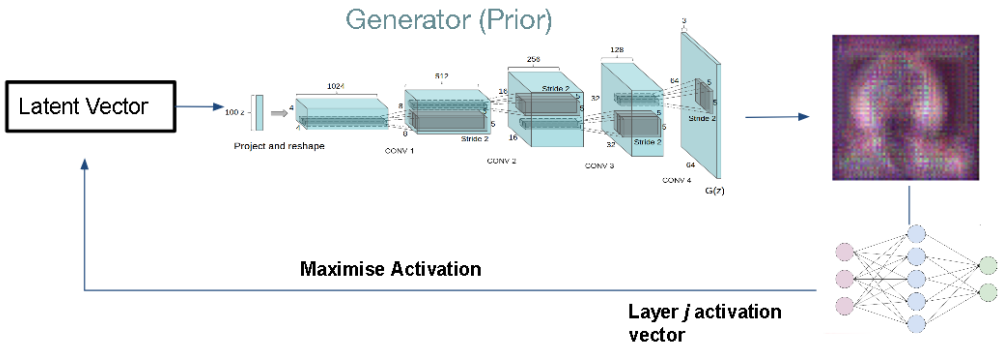


Figure 2: A basic diagram of the overall architecture of the visualisation process.

2.1.3 The Prior Model

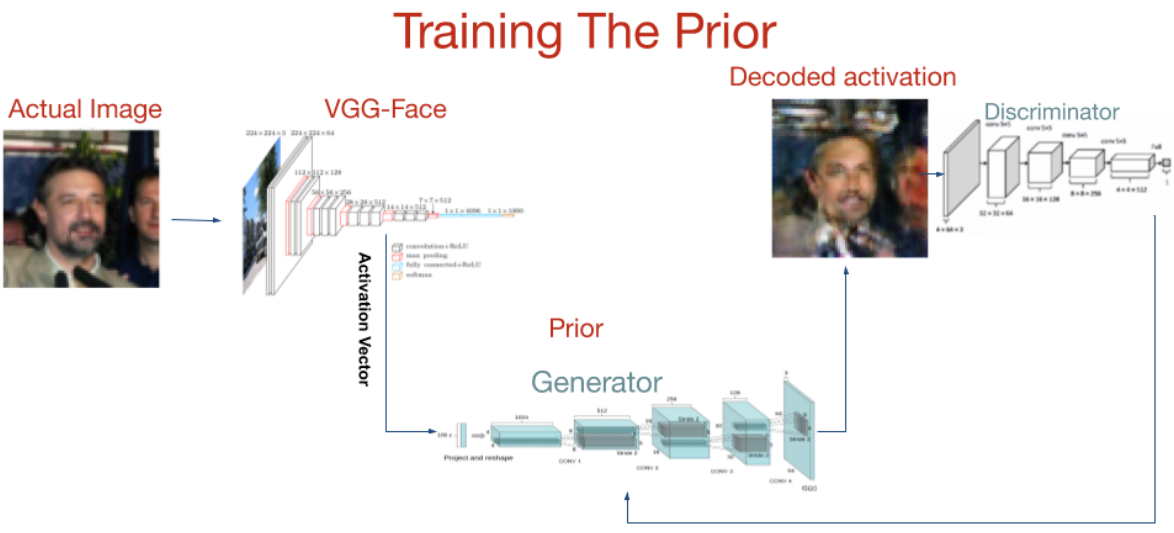


Figure 3: Figure of the training process for the prior network

A popular and impressive improvement to GANs was introduced by Radford et al. in 2015[20]. This generative model extended GANs by incorporating convolutional networks into the architecture. The DCGAN model does not utilise max pooling (instead using convolutional stride) and does not incorporate any fully connected layers. It is a simple and compact network, making it straightforward and efficient to implement, while also producing great results on benchmark tasks [20]. This makes it a popular and strong GAN variation. Convolutional layers are used in the discriminator to reduce an input image to a scalar probability value.Deconvolutional layers are used in the generator network to transform a latent vector input into an image. The discriminator takes as input an rgb image (3x64x64) outputting a scalar probability value. The generator takes a 100 dimension latent vector as input, producing an RGB image (3x64x64).

This DCGAN architecture, shown in figure 4, is used for the prior network, transforming layer activations into their (decoded) image representations. To facilitate this

the generator's latent space is a dataset of vector activations (as opposed to the normally implemented latent space of a gaussian distribution), their original input image being used as a label. Thus the generator will learn how to decode activation vectors into their original input image.

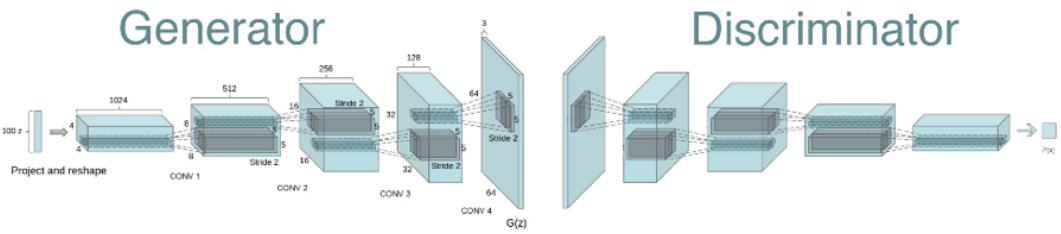


Figure 4: DCGAN Generator and Discriminator Models [20]

In order to ensure a correct implementation of the architecture, the network is first test-trained on face data, namely the fashion MNIST dataset [30] (Pipeline in figure 3. This involved first passing fashion MNIST images through a pretrained VGG-Face model (see 2.1.4) and collecting the activations from a layer of this network. This activation vector has the original input image as its label. This data is then used to train the DCGAN model. Once trained, the generator from the model can decode layer activations into the original input image (see example in figure 11). This was first done with a different dataset (fashion-MNIST data as opposed to LFW) due to the multiple components of the visualisation ensemble. It seemed beneficial to ensure that if future errors occured during the implementation and integrating of the entire visualisation structure, it would be somewhat clear that the prior model was not the source of error, and also for the fact face visualisations are more clearly interpretable to see the model is decoding correctly. The Fashion-MNIST dataset consists of 60,000 training images (see figure **??**) and 10,000 testing images (64x64 size, grayscale). There are 10 output classes for these images, each class label being a different clothing type. (See Appendix for fashion MNIST training results).

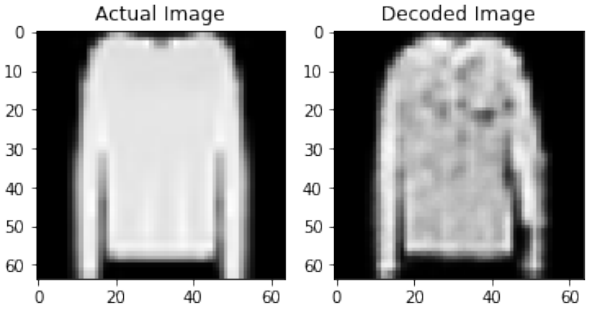


Figure 5:

Prior network trained on Fashion MNIST data.

2.1.4 VGG-16

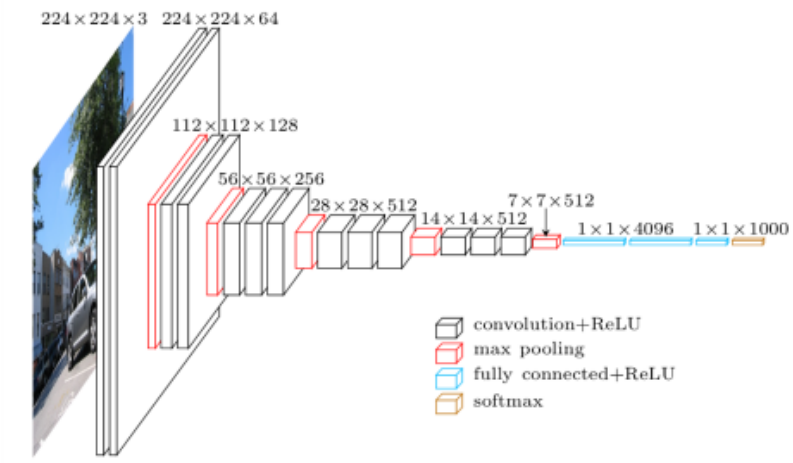


Figure 6: VGG-16 Model

In order to train the decoder, labelled data of layer activations must be collected. This involves training a network on image recognition, which, if successful, encodes features of these images into the networks layers. After training is completed, input images can be forward passed through the network and their corresponding layer activations can be collected, the activations containing encoded features associated with that input image. This constitutes the training dataset for the prior network.

A VGG-16 model [31], shown in figure 6, pretrained on ImageNet [32] is used for this purpose. ImageNet is a vast dataset (currently over 14 million annotated images) containing 1000 classes, and is the most widely used dataset for object recognition benchmarking, also proving effective in transfer learning [33]. VGG-16 is a deep convolutional network with 16 layers. 13 convolutional layers (along with 5 max pooling instances) are followed by 3 fully connected layers at the end of the network. The channel width of the convolutional layers is 64 initially, doubling after each pooling operation, finishing at 512 channels which are fed into the fully connected layers. This is a powerful network that has achieved excellent results in object recognition (top-5 accuracy of 92.3% on ImageNet [31])

2.2 Visualisation Techniques

2.2.1 Step by Step Explanation

Visualising the internal representations of each $k's$ discriminator involves optimising an input image through the discriminator network such that a particular layer is maximally activated (indeed singular neurons could also be visualised, but for the purpose of this paper layers are used). This pixel optimisation is done via the fixed prior network to ensure a regularised updating of pixel values, constraining the form of the image for increased realism as dicussed (as shown in figure 2).

For clarity and intuitive understanding, the following is a sequential explanation of visualising in this way. The process works as follows:

1. A random latent vector v is passed as input to the prior model, producing a random output image i (constrained under the context of faces are seen in 2)

2. This image vector i is forward passed through a trained k-GAN discriminator.

3. The activation vector of some layer j from this discriminator given input i is stored.

4. The average of this activation vector computed, and maximising the value of this vector is used as error for v, adjusting its values.

5. v is again passed through the prior, creating a different output which more maximally activates the discriminator layer j.

6. This process is repeated until some stopping criterion is reached (e.g until some number of iterations).

2.2.2 Generator Visualisation

In terms of visualising each $k's$ generator, the approach is more straight-forward than discriminator visualisation, as it won't involve decoding learned features in the networks layers and instead just generating images to see what features the generator has learned. This involves one-hot vector encoding on the input, e.g fixing one dimension of the latent space and then sampling from the space as is normally done in generation, effectively 'freezing' one dimension of the latent space. Multiple images are then sampled from the generator and through visual analysis inferences may be drawn about which feature(s) the generator has learned to produce, and which parts of the latent space the generator has mapped features of the data onto.

3 Experiments & Results

3.1 Dataset

The LFW dataset [34] was used to train the k-GAN and prior models. The deep funneling variant of this dataset is chose for the visualisation process due to its relative efficiency in training the dataset (less resource intensive than the vanilla LFW dataset) in training and testing, while also having the capacity to produce more illusive visualisations than the use of an simpler dataset such as fashion-MNIST. [35]. The LFW dataset consists of 13,000 images (see figure **??**). Each image is labelled with the name of the corresponding celebrity.

3.1.1 Preprocessing

Images were resized from 250x250 pixels to 128x128. Images were also normalised to contain values between 0 and 1.

3.1.2 Data for Prior Network Training

Training the prior network required collecting labelled activation data.This labelled data consists of some activation vector from a layer, with its original input image as a label. These activations were collected from a layer of the VGG model, and along with the input image were used to train the prior to transform (decode) encoded representations from a networks layer (activation vector) into an image (input image).

The activations from the final convolutional layer of the VGG-16 network were taken to train the prior network. This layer is chosen due to the fact it contains more abstract representations than earlier layers, having a richer array of encoded representations, resulting in more illustrative visualisations when decoded.

The prior generator takes as input the activation of the vgg layer. Thus, the latent space size is adjusted to facilitate this. Since there are 512 channels in the convolutional layer, the generator latent space size = 512. It outputs an rgb image

(512x8x8 activation vector $\longrightarrow$ 3x128x128 image).

The discriminator is trained as normal, taking an image as input and outputting a probability value that that image is a real sample.

(3x128x128 image vector $\longrightarrow$ scalar probability value).

Figure 7: Training examples for the LFW dataset

3.2 k-GAN Training

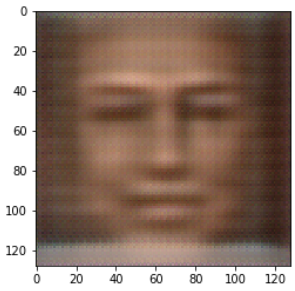
For training the k-GAN ensemble on the LFW data,$k = 3$ GANs were chosen. This number of generator-discriminator pairs was a realistic measure given the resource intensive nature of training a higher k, while also still training enough models to produce a sufficient array of visualisations. This allows the investigation of the difference in what section of the data the different ks modelled, as well as what features or collection of features they focused on during learning.

3.3 Results

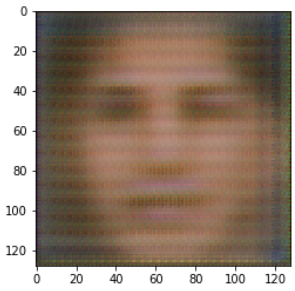
3.3.1 Generator Visualisation

From each medoid (the mean image that the generator models) of the 3 generators, it is clear the k-GAN model is working as intended in the facet of modelling different sections of the dataspace (see 8). Sampling from each generator makes this distinction even clearer (see 9). k=0 is attempting to model darker skin tone faces, while k=1 samples are more feminine in their appeareances. k=2 Also seems to be clustering based on skin tone, the samples appearing red in their skin colour.

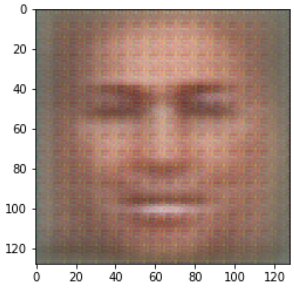
Training on this dataset illustrated the instability of GAN learning mentioned in the challenges of GAN learning (see 1.3). Training was unstable in terms of a greatly shifting medoid, and a lack of an increase in sample realism from early in training to later. At the end of training, the faces produced are noisy, especially in the k=2 generator where each sample produced seems to have a tile pattern of noise. Indeed samples from all



(a) k=0



(b) k=1

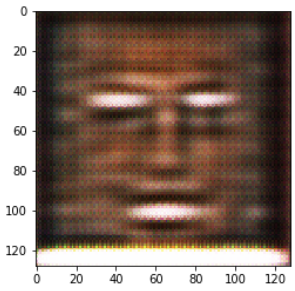


(c) k=2

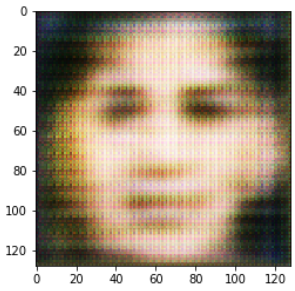
Figure 8: Medoid for each generator

generators appear to have this tile pattern. It is interesting nonetheless however that the differences in feature modelling is relatively clear, at least in respect to the k=1 model, where more feminine facial features are being encoded by the generator.

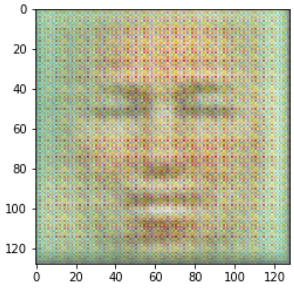
Freezing parts of the latent space (as mentioned in section 2.2.2) to produce varying feature values in samples in order to investigate learned features proved unsuccessful. Produced samples were noisy and indiscernable in their nature, giving no clear insight into the internal mapping each generator created onto the latent space.



(a) k=0



(b) k=1

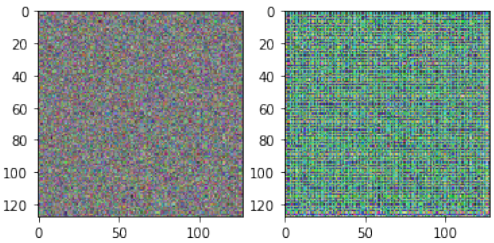


(c) k=2

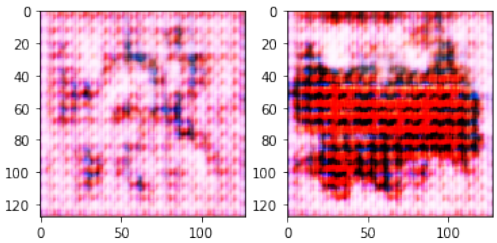
Figure 9: Random sample for each generator

3.3.2 Discriminator Visualisation

A large task in this paper is the visualisation of the model's k discriminators, using the structures and visualisation pipeline previously detailed(2). Due to the instability of learning however, the visual pipeline was not successful in reconstructing the discriminator's latent representations. Activation maximisation failed to illustrate any meaningful features of the discriminator's final convolutional layer (see 10). This is potentially due to an error in the implementation of maximisation, or the combined noise of the prior networks reconstructions and the noisy nature of the k-GAN discriminator representations.



(a) Visualisation without a prior



(b) Visualisation with a prior

Figure 10: Activation maximisation for final layer of k=1 discriminator's convolutional layer, with and without a prior. Left: Initial input image, Right: Maximised Image

3.4 Discussion

The following paper aspired to apply the tool of visualisation techniques to visualising the representations encoded by a k-GAN ensemble. The intention of this was to visually analyse the features learned from each of the constituent GAN networks of the ensemble, and to aim to use this illustrations in drawing insights about how the ensemble partitions the dataset into sections during learning. The method of doing this is showing visually what features are learned by the different discriminator-generator pairs in the ensemble. From these visualisations conclusions may be drawn about the representations of the networks and general remarks of the learning process can be made. To outline this specifically, do the different models learn different parts of the dataspace? Can insights be made from the visualisations gathered? This would give credence to the tool of visualisation, which can be used to gain understanding in the area of DNN transparency and illuminate the black box problems encountered in deep learning. The findings made are outlined and discussed in this section, with shortcomings being examined in the Limitations section. Additional steps in the investigation this project undertook are explored in the Future Work section.

The nature of the k-GAN learning process, namely that each discriminator-generator pair models a non-overlapping section of the data distribution, means that insights are already given into the most prominent features in the dataset, e.g which one each model seperately focuses on, or finds important. With this in mind, training a k-GAN model with k=3 on LFW data indeed shows that this is how the ensemble seems to operate and learn. Results from generated samples (see figure 9) from each k support this claim that the model is learning different sections of the dataspace as generated samples from each k differ in the types of faces they appear to be learning e.g encoded different intervals of feature values.

3.5 Limitations

There were several shortcomings and oversights encountered in this project. In terms of the prior model, training on the fashion MNIST dataset provided much cleaner visual results due to its simplicity in resolution and structural differences of each class (see 5.1). This did not directly transfer to training on a more complex dataset as is seen in the results of the LFW prior.

One major limitation that undermined the clarity of results in this project was the challenging nature of training generative adversarial networks. The instabilty of GAN training is a cumbersome barrier and can be clearly seen in the produced visualisations in the results section. This instability and failure to produce realistic or interpretable results can be seen both the k-GAN and prior models, the k-GAN generators producing interpretable samples but somewhat lacking realism in the faces produced, while the prior failing to produce interpretable reconstructions in the activation maximisation process. In terms of analysing and approaching this failure, during training hyperparameter tuning was not performed to attempt to produce stronger samples, which would be a necessity in creating clearer visualisations (more realistic generated samples). Techniques such as adding input noise to both the generator and discriminator could also be applied, and were done so to the prior model (DCGAN architecture) but not to the k-GAN discriminators nor the generators.

The instability of training in the prior model could also be a result of the loss function in the generator. The generator in this model is essentially being trained for two purposes. Firstly, to produce realistic faces, and secondly, to accurately reconstruct faces from a specific activation vector. Therefore the loss from the discriminator (sample realism) is combined with the error difference in the sample image and the original vgg input image (reconstruction accuracy). Different approaches to combining and weighing

these loss functions could also be explored. First creating a generator that produces realistic samples and then specialising it to reconstruct specific activations is another approach to creating a stronger prior.

3.6 Future Work

There are many aspects of improving and strengthening the results found in this project that could be explored in future applications of visualisation or indeed in k-GAN training. The use of this form of visualisation could also be extended. For examples, applying these visualisation techniques to the process of a network learning to represent human faces could give insight into the internal learning processes and could even be applied in the field of neuroscience, a field of research closely interlinked with artificial intelligence.

During the research of literature in the technical aspects of visualising, another interesting aspect of using the visualisation technique arose. This concerns the subjectivity in the choice of prior when training an image to maximise a layers output. One limitation of visualising using an image prior is that a different prior may have to be used for visualising different data. Although creating more realistic and illustrative imagery whe compared to the use of other regularisation techniques , the problem in natural image priors is the lack of separation between which part of the visualisation is from the layer/neuron visualised, and which part is due to the prior, in other words, how much of a role each aspect contributes in producing the final visualisation. Investigations into this problem could lead to important insights into the use of a prior in visualisation regularisation, and indeed work is already being done on creating general priors that can be used across visualisations [10].

3.7 Conclusion

It is clear from this paper's implementation of k-GANs that this variant of generative adversarial nets can give great insight into the nature of datasets with complex features and that their visualisation can potentially give insight into the nature of DNN learning processes. The partitioning of the dataset into different, easily recognisible face types by the different k generators shows already gives the beginning of this insights. Due to the instability issues of GAN learning however care is needed in training and tuning hyperparameters to ensure a sufficient amount of feature encoding, which can then be extracted by visualisation techniques. In this sense this project encountered difficulties. Implementing activation maximisation also requires care in the choice of and training of the prior network, as well as hyperparameter tuning when updating gradients.

Overall however, the research direction of visualisation is one of great promise but requires careful consideration in its application. GANs are also a powerful tool that would benefit from feature visualisation, in both the discriminator, through activation maximisation, and the generator, through simply generating samples as well as hot encoding the latent space when creating samples.

4 References

[1] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[3] Qeethara Al-Shayea. Artificial neural networks in medical diagnosis. *Int J Comput Sci Issues*, 8:150–154, 02 2011.

[4] April Moreno Arellano, Wenrui Dai, Shuang Wang, Xiaoqian Jiang, and Lucila Ohno-Machado. Privacy policy and technology in biomedical data science. *Annual review of biomedical data science*, 1:115–129, 2018.

[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *NIPS Tutorial*, 2017.

[6] Julian D Olden and Donald A Jackson. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1):135 – 150, 2002.

[7] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[8] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

[9] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. 06 2015.

[10] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[11] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *preprint*, 12 2013.

[12] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[13] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[14] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[17] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[18] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[19] S. Minchul. Gans awesome applications. https://github.com/nashory/gans-awesome-applications, 2018.

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] Augustus Odena. Open questions about generative adversarial networks. *Distill*, 2019. https://distill.pub/2019/gan-open-problems.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[23] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[24] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3308–3318. Curran Associates, Inc., 2017.

[25] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[26] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[28] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[29] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[32] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[33] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

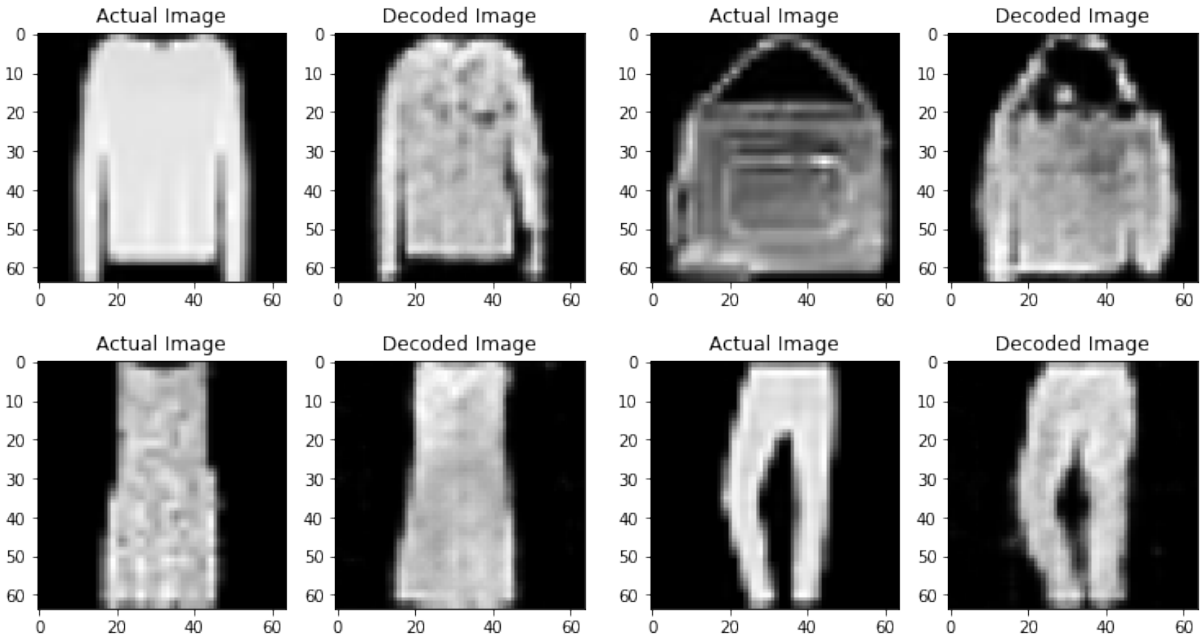
[34] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[35] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

5 Appendix

5.1 Prior Network Training

Below are some examples after training the prior network on fashion-MNIST. All 60,000 training examples were used to train the DCGAN architecture. 10 epochs of training were performed. Images were then randomly sampled from the dataset, being passed through the pretrained vgg model to collect the 13th layer's (final convolutional layer) activations. Then these activations were passed through the generator, creating the decoded images below. Beside each image is the real image label for each activation.



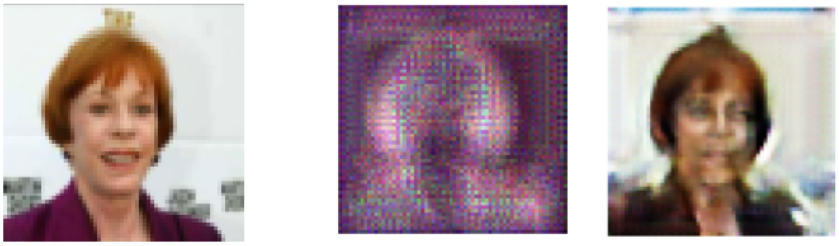
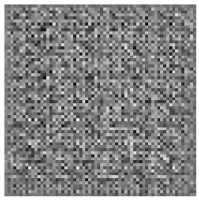


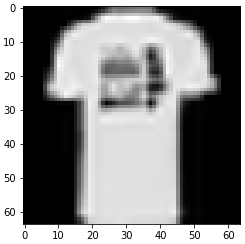
Figure 11:

Prior network trained on a small subset (1000 samples) of LFW data.
Left to Right: Actual image, decoded image at beginning of training, decoded image after 15 epochs of training. This example is probably overfitting, e.g the network is remembering as opposed to learning features.

5.1.1 Prior Learning Progression

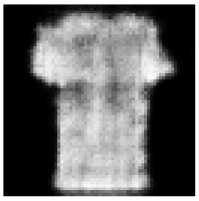


(a) Initial Input Noise

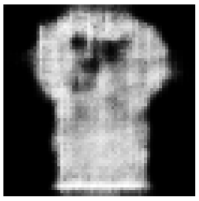


(b) Actual Image

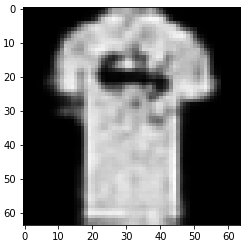
(c)



(d)



(e)



(f) Final Reconstruction

Figure 12: Progression of Prior Training for Fashion MNIST Data