Performance Evaluation of Generative Adversarial Networks with Transfer Learning

Adam Price
University of Derby
May 2021

Contents

1 Introduction **2**
1.1 Aims . . . 2
1.2 Objectives . . . 2

2 Literature Review **3**
2.1 Generative Modelling . . . 3
2.2 Generative Adversarial Network . . . 3
2.2.1 Summary . . . 5
2.3 GANs with Convolutional Neural Networks . . . 5
2.4 Transfer Learning . . . 6

3 Methodology **7**
3.1 Quick Overview . . . 7
3.2 Dataset . . . 7
3.3 Model Setup . . . 8
3.4 Transfer Learning Setup . . . 10
3.5 Classifier Setup . . . 11

4 Results and Analysis **12**
4.1 Visual Analysis . . . 12
4.2 Classifier Results . . . 13

5 Conclusion **14**

1 Introduction

There has been a huge interest and growth of research into Generative Adversarial Networks (GANs), a type of generative modelling since the first GAN paper was proposed by Ian Goodfellow in 2014 [5]. Generative modelling is a way of learning from any kind of data distribution using unsupervised learning with the aim of generating new data points with variations from a true distribution of a training dataset [15]. GANs have been applied to various applications such as computer vision, natural language processing, video synthesis, image synthesis and much more achieving impressive results with little training time compared to older generative modelling approaches [7].

Among all the use cases for GANs, computer vision and image synthesis have been the most studied topic, research into this area has already demonstrated huge potential when using the GAN modelling method for image synthesis tasks, with the DCGAN paper having the most notable impact in the GAN field by using convolutional neural networks for the architecture and achieving high-resolution results [14]. Also the stylegan2 paper by NVIDIA, which makes use of an adaptive discriminator within the GAN to limit overfitting and better stabilise the model [8]. There is another paper given by Goodfellow in 2017, the original researcher for GANs where he describes the importance of Generative Adversarial Networks and generative modelling in general explaining why more research needs to be put into the area of study [4].

Because generative adversarial networks have received quite a lot of interest in the last few years with everyone trying to improve or change the GAN process, more people than ever want to get into generative modelling and build models to generative images that look like real-life objects and this paper aims to provide insight on what GANs are, how they work and focus on whether it is better to create a new network following these resources yourself or take a preexisting trained model and use transfer learning with a new dataset.

1.1 Aims

This paper aims to investigate and explore the methods used in generative modelling to generate images based on a given dataset, specifically looking at the generative adversarial network approach for generative modelling. Then evaluate whether it is better to build and train your own GAN model from scratch or take a pre-trained model trained on a different but similar dataset and use transfer learning to tune the model to a new dataset and compare the results.

1.2 Objectives

- Thoroughly review the literature related to generative adversarial networks and transfer learning to identify current works and how they contribute to my question and solution.

- Explain the methodology for setting up the GANs and transfer learning models and how their performance can be compared.
- Evaluate and conclude the results between a built model and pre-trained GAN model and give an insight on which method should be preferred for newcomers trying to build generative models quickly.

2 Literature Review

This section provides a review of some of the core methodologies and technologies used in the current works that relate to my aims and objectives. The literature review will explore what is generative modelling, how generative adversarial models are built and trained, and how transfer learning can be applied to models to speed up the required training time for getting results.

2.1 Generative Modelling

Generative modelling is a way of learning from any kind of data distribution using unsupervised learning with the aim of generating new data points with variations from a true distribution of a training set. It is not always possible to learn the exact distribution of the data so there needs to be a way to model the distribution as close to the true distribution as possible. This is where deep generative modelling techniques come into play by using the power of neural networks to learn a function that can estimate the distribution [15].

Generative models can generate new data instances from capturing the joint probability p(X, Y), or p(X) without class labels. Discriminative models discriminate between data instances by capturing the conditional probability p(Y | X) [1]. For example, a generative model could create new images of animals by learning from a collection of animal images and the discriminative model can tell what animal is shown in the image.

Research into generative modelling split into many different approaches, each technique with its benefits and downfalls. The two most used techniques for generative modelling are Variational Autoencoders (VAE) which works by trying to maximise the lower bound of the data log-likelihood and Generative Adversarial Networks (GAN) which try to achieve an equilibrium between a generator and a discriminator [2]. This project will focus on the latter method for the generative model and transfer learning evaluation.

2.2 Generative Adversarial Network

First proposed by Ian J. Goodfellow and colleagues from the University of Montreal, the generative adversarial network (GAN) was developed with the intention of estimating generative models via an adversarial process in which two models are trained simultaneously, a generative model and discriminative model. The generator is pitted against the discriminator as an adversary which learns

to tell if samples came from the generator distribution of the real data distribution. This process is adversarial training, both networks improve by working against each other in a game like fashion [5].

The objective of the generator model is to generate images by learning from the sample distribution to trick the discriminator into thinking the images came from the real samples, and the objective of the discriminator is to decide whether the images it samples are generated or from the original sample set. The weights of the discriminator are updated based on its ability to correctly classify where the images came from and over time through errors and backpropagation it should become better at detecting generated samples from the real dataset. So, for the generator to fool the discriminator it is forced to improve the generated samples to match the data distribution more effectively and produce more realistic samples. This process is repeated until a perfect equilibrium is reached and the discriminator can no longer distinguish the generated images from the real ones. At this point, the output for the generator model should be at the best it can at being able to reproduce images from the original data distributions [5, 13]. Figure 1 visualises the core setup of the GAN explained described above.

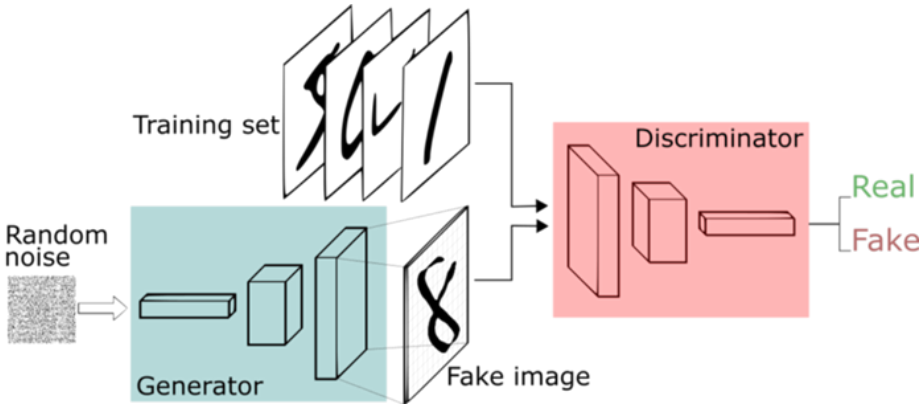

Figure 1: A Visual representation of a generative adversarial network setup described by Ian Goodfellow in his paper 'Generative Adversarial Nets'. Credit Thalles Silva

Both networks start with randomised weights so they can be in equilibrium state as the training begins, as the generator would capture more of training data distribution, the discriminator would always be unsure whether the images it receives are generated or not[1].

Conditional generative models which allow for the output to be conditioned to be a specific thing or object can be obtained by simply adding a conditional label vector as input along with the noise input vector to both the generator and discriminator making it a conditional GAN. By conditioning the GAN with additional information, it is possible to direct the data generating process and select the desired features in an image or type of object in the output instead of

those features being random based on the sample of noise and can increase the training time in datasets [13]. The conditioning labels used for the models can be a simple label vector that represents classes in a dataset that are assigned to different objects.

Using different techniques when training the GAN combined with different network architectures such as in the deep convolution GAN paper which is more suited for image features extraction, GANs have seen major improvements in results in the recent years when generating realistic high-resolution images [14, 4].

2.2.1 Summary

- GANs consist of two networks a generator and discriminator which trained against each other.
- The generator is trying to maximise the probability of the discriminator mistaking fake from real samples by learning from the sample distribution.
- The discriminator is trying to minimise the probability of mistaking generated samples from the real samples.

2.3 GANs with Convolutional Neural Networks

One of the earliest works after the original 'generative adversarial nets' paper to employ the GAN training method for generative modelling with a network architecture which was not a simple fully connected network was the paper 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks' by Alec Radford in 2016 [14]. Convolutional Neural Networks are used for both the generator and the discriminator. CNNs are generally used in computer vision tasks such as object detection and face recognition [6], so it makes sense to train a GAN with a network that is better at extracting features in an image for a generative model which generates new images based on learning from a sample distribution of images.

DCGAN generative model takes in a 100-dimensional uniform distribution vector and passes it through a series of convolutional layers using upscaling the image resolution by double each time until it has fully converted the vector of random noise into a 64x64 image. Using DCGAN provides impressive results for natural-looking images for an unsupervised learning model. The trained discriminator becomes a good CNN image classifier that shows competitive performance to other unsupervised algorithms for creating image classifiers [14]. Techniques used in this paper will make up the basis of the models that will be implemented in the models later in this project.

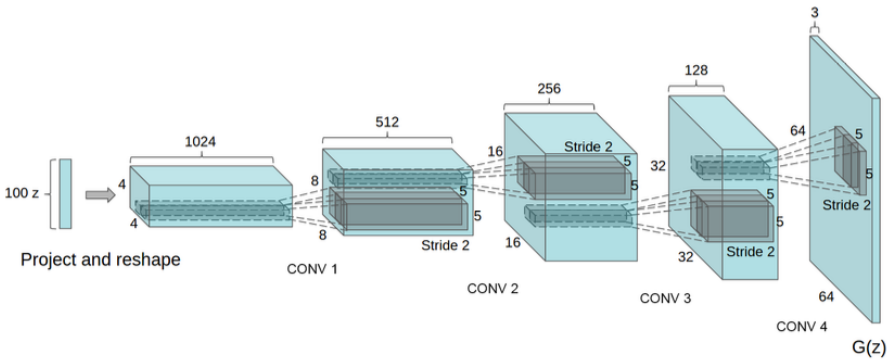

Figure 2: A Visual representation of the generator model in the DCGAN paper which makes use of multiple convolutional layers.

2.4 Transfer Learning

There are a lot of tasks within deep learning that require a huge of amount time and compute power to train models for, training deep convolutional neural network models can take days or even weeks on high-resolution or challenging datasets with very few samples to learn from [10]. To combat the long training times, a technique for deep neural network training called transfer learning became popular, where a model trained for one task is repurposed and trained for another, weights in the model used for one task can act as a starting point for the training process for a new task. It does not usually matter what kind of dataset the model was trained on beforehand but in general, transfer learning works best when the two problems being solved are similar [11].

"Modern deep neural networks exhibit a curious phenomenon: when trained on images, they all tend to learn first-layer features that resemble either Gabor filters or colour blobs" [18]. This appears to happen to the first couple layers regardless of the dataset used or the cost function applied to the network, the features on the first layer or two tend to be general layers for computer vision tasks. The final layers in a neural network model contain features computed that greatly depend on the dataset chosen for the task. This means, unlike the first layers the last layers are feature specific and are used at to end for the final classification of objects or object generation [18].

With using transfer leaning on one of the GAN models in the comparison later, hopefully, the model with it can make use of the generalisation layers at the beginning of the neural network and train faster than the model without pre-trained layers. Once the scores of both networks are displayed on a plot the advantages of transfer learning should be clear.

3 Methodology

3.1 Quick Overview

For the comparison, I will be using an untrained model built in python using the same network architecture found in the paper 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks' [14] and taking a pre-trained model which was trained on a similar dataset to the new dataset which will be used to train both models. Additionally, to visually comparing the results by checking the produced images, I will create an image classifier trained in the same dataset to test which generative model can produce the most images that can pass recognised by the classifier.

All the code for this project will be written in python and the models will be built using TensorFlow with the Keras API which simplifies the model building process by providing an interface with the core TensorFlow code allowing me just to describe the model architecture.

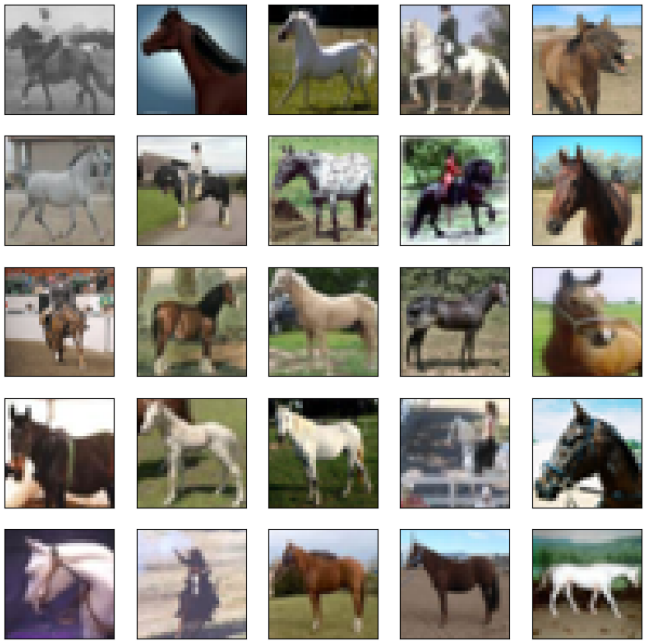

Figure 3: Real horse Images from the CIFAR-10 dataset

3.2 Dataset

I will be using the CIFAR-10 dataset for this experiment which is a labelled subset of the tiny 80 million tiny images set collected by Alex Krizhevsky and Vinod Nair [9]. I will be using the horse set only from this dataset which

contains 6000 images to try and minimise any variables in the comparison. The scale of this dataset is 32x32 which is considered a low resolution even when compared to the first GAN paper by Ian Goodfellow [5]. But with limited hardware and online GPU compute hours, the lower resolution allows for me to train the models faster and get the results relatively quickly. Figure 3 shows a few random images from the horse CIFAR-10 dataset to give a visual example of what the GANs in the later stages of the project will be trying to recreate.

3.3 Model Setup

Starting with the untrained model we need to define the generator and discriminator architecture for the GAN. The core model architecture and approach for the networks are inspired by the DCGAN paper because they achieved pretty good results with convolution neural networks when creating stable models that do not collapse into random noise.

The guidelines for a stable deep convolutional GAN architecture described in the DCGAN paper which the model created will be following.

- Use fractional-strided convolutional layers for up-sampling in the generator and use strided convolutional layers for down-sampling in the discriminator.
- Use batch normalisation layers in both the generator and the discriminator.
- Do not included any fully connected layers expect for the input and the output of the GAN model.
- Use ReLU activation for all layers in the generator expect for the output layer.
- Use LeakyReLU activation for all layers in the discriminator.
- Gaussian Weight Initialization, the paper reports that in testing they found initialising all the weights using zero-centred gaussian distribution with a standard deviation of 0.02 provides the highest chances of a successful GAN generation.
- Adam Stochastic Gradient Descent, the standard algorithm used to optimise the weights of convolutional neural networks.

Listing 1: Generator network architecture

```
def build_generator_model():
    model = Sequential()

    # foundation for 4x4 image
    model.add(Dense(512 * 4 * 4, kernel_initializer=init,
        input_dim=LATENT_DIM))
    model.add(LeakyReLU(alpha=0.2))
```

```
    model.add(Reshape((4, 4, 512)))
    # upsample to 8x8
    model.add(Conv2DTranspose(256, (4,4), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))
    # upsample to 16x16
    model.add(Conv2DTranspose(128, (4,4), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))
    # upsample to 32x32
    model.add(Conv2DTranspose(128, (4,4), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))

    # output layer
    model.add(Conv2D(3, (3,3), kernel_initializer=init,
        activation='tanh', padding='same'))
    return model
```

The generator network takes in a latent dimension vector 100 and is scaled up multiple times into a 32x32x3 image using convolutional layers. $100 \rightarrow 8192 \rightarrow 4x4x512 \rightarrow 8x8x256 \rightarrow 16x16x128 \rightarrow 32x32x128 \rightarrow 32x32x3$.

Listing 2: Discriminator network architecture

```
def build_discriminator_model():
    model = Sequential()
    # Input Layer
    model.add(Conv2D(64, (3,3), kernel_initializer=init,
        padding='same', input_shape=(32, 32, 3)))
    model.add(LeakyReLU(alpha=0.2))

    model.add(Conv2D(128, (3,3), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))

    model.add(Conv2D(128, (3,3), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))

    model.add(Conv2D(256, (3,3), kernel_initializer=init,
        strides=(2,2), padding='same'))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))
```

```
model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(1, kernel_initializer=init, activation='sigmoid'))

# compile model
opt = Adam(lr=0.0002, beta_1=0.5)
model.compile(loss='binary_crossentropy', optimizer=opt,
    metrics=['accuracy'])
return model
```

The discriminator network takes in a 32x32x3 image and converts it into a single value, real or fake. Together, both the the networks ar become the GAN.

Here is a summary of the other techniques that help improve the stability of a GAN without changing the network architecture are listed in the DCGAN paper as well as the NIPS paper by Ian Goodfellow himself on how to improve GANs [4].

- Scale image values between -1 and 1.
- Label smoothing, using soft labels for identifying the real and fake images when passing them into the GAN can have a regularising effect whilst training the model reducing the chances of overfitting and potentially leading to fast training times. Values $0.8 - 1.2$ are used to represent 1 and values -0.2 and 0.2 are used to represent 0.
- Label Noise, introducing minor errors in the labelling of the fakes and real images where the fakes are marked and real and vice versa can also help with the overfitting issue.

3.4 Transfer Learning Setup

For the GAN using transfer learning, a model trained on a different dataset is required, so a model trained on the LSUN dataset [19] is used. But the network is trained only on the cat images only from the dataset. This model was chosen for this because the core neural network architecture for both the generator and the discriminator are almost identical to the GAN setup without transfer learning. The untrained GAN was influenced by this architecture setup so the comparison can be relevant enough and the results are meaningful. Also, the dataset itself is similar to the new dataset, cats to horses. It is better to do transfer learning on similar problems in computer vision [11].

One major difference between the models is that the model trained for the LSUN CAT dataset is 128x128 and the CIFAR-10 dataset is 32x32, so the model needs to be shorted down to the 32x32 layer. With the first layers taken from the model already trained on the LSUN CAT dataset hopefully, the training process can take advantage of the generalisation layers which should improve

the speed of the GAN to distinguish colours and shapes. With both the GANs setup to generate 32x32 horse image. In the final model, the classifier is needed to accurately compare the performances over time and tell which method is the best.

3.5 Classifier Setup

To compare the results more accurately from both GAN models than simply visualising the differences to tell which model provided the best outcome, an additional network can be created to test which GAN generates the most realistic looking images.

A classifier model can classify items or objects of a dataset that it is trained on, for example, if we take the cifar10 dataset and trained the classifier on all 10 classes it would try to classify any 32x32 image passed into the network as one of those classes. For this project, since only the horse images from that cifiar10 dataset are being used to train the GAN models, a classifier needs to be trained which can simply tell whether a sample is a picture of a horse or not, i.e. real or fake looking.

The implementation for the classifier network taken mostly from the TensorFlow site [17] for creating a basic convolution neural network classifier with a few changes to the code to make it better suited for this project. The biggest change is relabelling the dataset so the classifier can be used to detect whether an image is real or fake all classes that are not the horse are labelled as 0 and the horse samples are labelled as 1.

Although the initial accuracy of the classifier was around 70-80% each train, by adding additional features such as batch normalisation, dropout and image flipping to the network and its layers the classifier achieved an accuracy of 97% when telling the horse images from the rest of the dataset. 97% should be more than accurate enough for this rudimentary comparison model because the margins between the two GANs should not be that close based on initial findings.

Wrote additional code which can take a bunch of images at once and return the average results of those images which will be stored in a CSV file to get the results later.

4 Results and Analysis

4.1 Visual Analysis

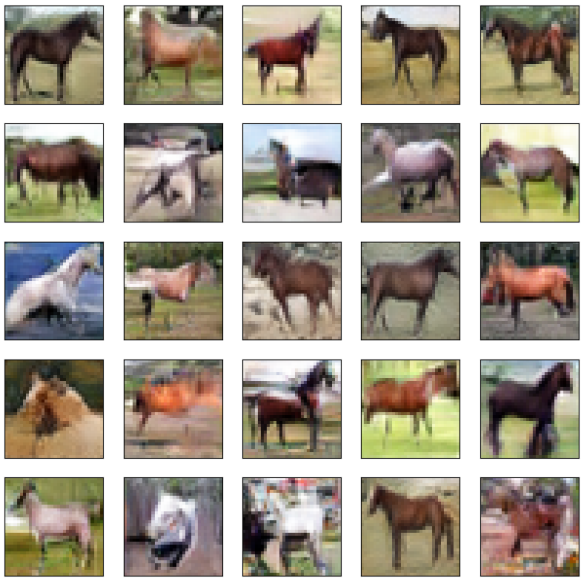

Figure 4: Generated samples from the untrained GAN.

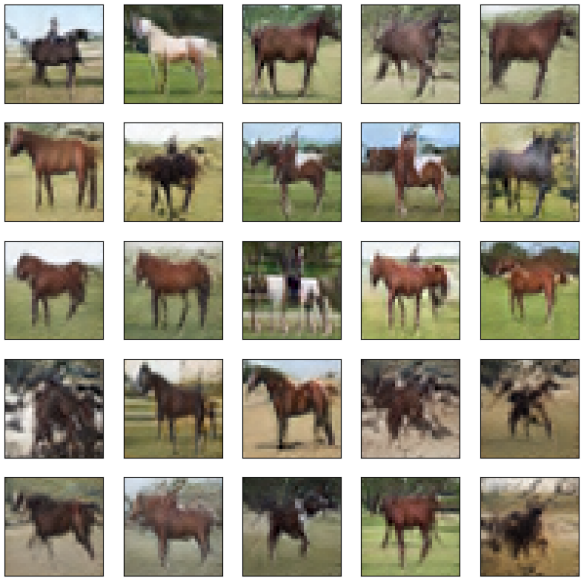

Figure 5: Generated samples from the GAN with transfer learning.

For the visual comparison, the image samples in figure 4 and figure 5 are taken from the latest snapshot of each model. Looking at the samples generated, the model created with transfer learning contains more stable horse images with fewer defects and errors from the generative modelling process, the legs and head being the most obvious when comparing the samples. The transfer learning model seems to prefer generating brown coloured horses, this could either be by chance due to the stochastic nature of the machine learning algorithm or the preference is from the initial weights trained on the original cat dataset due to the first layers in the network being generalisation layers with pick out colours and shapes. Or possibly more images were darker in the dataset when being trained the first time around give the model a preference for darker colours.

The model with transfer learning does seem to produce more realistic horse samples at first glance, but to confirm this with metrics, the classifier which was trained earlier on the same dataset can be used to validate the horse samples and provide a score to how well each model performed over time to give an idea of how long each model took to produce decent results.

4.2 Classifier Results

Both models trained for around 9-10 hours each. To get the performance over time for the models a snapshot of each model and the current weights are taken at roughly every hour increment of the training process. There is a slight variation in increments due to the epoch training time not being consistent but the max variation from the hour is no more than 5 minutes. 500 samples are generated with random input vectors for each of the saved incremental models from both GANs, 500 samples for each should be a big enough sample size to avoid any outlier results when collecting the samples. The classifier model will be fed each group of images and give a score for how many of the samples pass ass a real horse image.

With the data plotted onto a line graph in figure 6, the transfer learning GAN performs better at all stages and quickly gets a higher classifier score much sooner than the freshly trained GAN which in comparison has slow incremental improvements. The rapid increase from the 2 hours to 3 hours mark is most likely due to the model already containing multiple feature extraction weights in the convolutional neural network layers and the model just needed to be tweaked a little bit to go from creating cat images to creating horse images.

The transfer learning model seems to have levelled out without any significant gains in performance past the 6 hour mark whilst the GAN trained using randomised weights is still climbing to that point, possibly given enough time the model's score could improve but this experiment is about which GAN method provided good looking results in the quickest time.

To summarise the results, the generative adversarial network which trained using transfer learning preformed a lot better than the one without being trained on a different dataset first. So, transfer learning should be used on GANs and

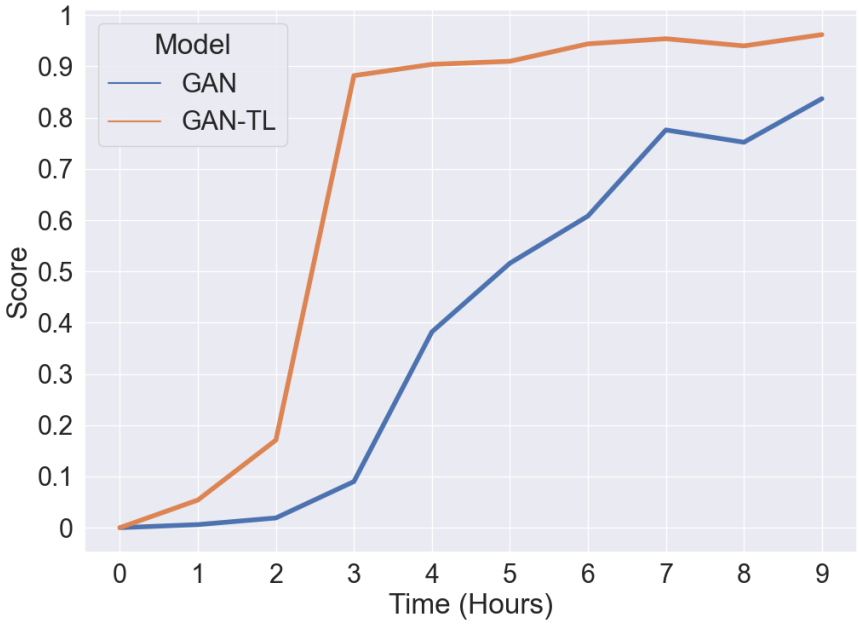

Figure 6: The results from the classifier model.

where possible to save time. Although results can vary widely depending on the starting and final dataset, if they are not similar like using a car trained GAN with the horse dataset, it may even take longer than starting from fresh if the model cannot use any of the pre-existing feature extraction weights because the dataset objects vary too much.

5 Conclusion

Generative adversarial networks take a very long time to train, especially high-resolution GANs. It is much quicker to take an existing GAN trained on a different dataset and use transfer learning to tweak the model weights to the new dataset. Therefore, where possible transfer learning should be used to save time. Interestingly the resulting samples from both GAN models are quite different from each other. Further testing could be done, like running this experiment several times and getting the average difference between both training methods. With the stochastic nature of machine learning algorithms, the ending result of this experiment could change, albeit probably unlikely because the scores for the models looked far outside the margin of error.

References

[1] Sean Augenstein et al. *Generative Models for Effective ML on Private, Decentralized Datasets*. 2020. arXiv: 1911.06679 [cs.LG].

[2] Sam Bond-Taylor et al. *Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models*. 2021. arXiv: 2103.04922 [cs.LG].

[3] Yaël Frégier and Jean-Baptiste Gouray. *Mind2Mind : transfer learning for GANs*. 2020. arXiv: 1906.11613 [cs.LG].

[4] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. arXiv: 1701.00160 [cs.LG].

[5] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[6] Jiuxiang Gu et al. *Recent Advances in Convolutional Neural Networks*. 2017. arXiv: 1512.07108 [cs.CV].

[7] He Huang, Philip S. Yu, and Changhu Wang. *An Introduction to Image Synthesis with Generative Adversarial Nets*. 2018. arXiv: 1803.04469 [cs.CV].

[8] Tero Karras et al. *Training Generative Adversarial Networks with Limited Data*. 2020. arXiv: 2006.06676 [cs.CV].

[9] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. Canadian Institute for Advanced Research, 2009.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[11] Quoc Le et al. "ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/233509073ed3432027d48b1a83f5fbd2-Paper.pdf.

[12] Puneet Mangla et al. *Data Instance Prior for Transfer Learning in GANs*. 2020. arXiv: 2012.04256 [cs.CV].

[13] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[14] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[15] Lars Ruthotto and Eldad Haber. *An Introduction to Deep Generative Modeling*. 2021. arXiv: 2103.05180 [cs.LG].

[16] Abhinav Sagar. *Generate High Resolution Images With Generative Variational Autoencoder*. 2020. arXiv: 2008.10399 [eess.IV].

[17] Tensorflow. *Convolutional Neural Network (CNN)*. 2021. URL: https://www.tensorflow.org/tutorials/images/cnn (visited on 03/19/2021).

[18] Jason Yosinski et al. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG].

[19] Fisher Yu et al. "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop". In: *arXiv preprint arXiv:1506.03365* (2015).