Generating new images out of datasets, using the Generator and Discriminator in GANs

How effectively do the hyperparameters of a GANs influence its ability to generate high-quality images and its time complexity?

Word Count: 3943

Computer Science [Higher Level] Extended Essay

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Table of contents:

1. Introduction…….....2
2. Background Information……..…….3
   - 2.1 Understanding Artificial Intelligence……..3
   - 2.2 Understanding Machine Learning and its Types.....4
   - 2.3 Understanding Deep Learning…….....8
   - 2.4 Understanding Neural Networks……....9
   - 2.5 Generative Adversarial Networks (GANs).....11
3. Experiment…….……..12
   - 3.1 Methodology……..12
   - 3.3 Limitations of The Experiment……..16
   - 3.3 Results and Analysis……..17
     - 3.3.1 Buffer Size……..17
     - 3.3.2 Batch Size……..……..21
     - 3.3.3 Learning Rate…….25
4. Further Research Opportunities……..28
5. Conclusion……..29
6. Works Cited……30
7. Appendix……33

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

1 Introduction

Generative Adversarial Networks or GANs are said to be the most lucrative recent technology within the machine learning and deep learning subset. Yann LeCun, a french pioneer in Computer Science says about GANs, "GANs is the most interesting idea in the last 10 years in Machine Learning ". GANs were developed by Ian Goodfellow in the year 2014 and have seen many applications ever since.[1] Among its greatest applications is image generation. Generating an image, one that is unique, and one that doesn't share a direct likeness to any other existing image before it, is a major advancement within the field of machine learning, Using the idea of GANs, an individual may produce a brand new image, that is merely inspired by existing images that may be entered as training datasets.

Then there are hyperparameters, which are adjustable settings that determine how a GAN learns to generate images. These parameters include the optimizer type, epochs[2], number of hidden layers, learning rate and others.

This essay seeks to investigate the effect of tuning these hyperparameters on the ability of GANs to generate high-quality images and the time required for training. The findings of this study could provide insights into the best practices for tuning GAN hyperparameters and enhance the understanding of GANs behaviour. The three main parameters that I will be focussing on are Batch size, Buffer size, and Learning Rate.

---

[1] Ahmed, Ryan. "TensorFlow 2.0 Practical Advanced." *Udemy*, Udemy, 21 2 2021, https://www.udemy.com/course/tensorflow-2-practical-advanced/learn/lecture/16266470#content. Accessed 28 November 2022.
[2] Baeldung. "Epoch in Neural Networks." *Baeldung*, Baeldung, 11 November 2022, https://www.baeldung.com/cs/epoch-neural-networks. Accessed 21 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

2. Background Research

2.1 Understanding Artificial Intelligence

Artificial Intelligence (AI), can, in essence, be defined as an automated technology that attempts to replicate human intelligence. It is seen everywhere in the modern world, where it is able to solve logical problems that are well-defined. It is important to note that AI is a general field that encompasses other subfields such as machine learning, deep learning, and so on. It is not necessary that an AI's approach to solving a problem would have to include 'learning', but rather what we find is, many approaches that are present, that would still be considered AI, don't fall under the subfield of machine learning, which approaches problems based on feedback and learning. An example of this would be conditional statements within programming languages, such as the IF ELSE statement, which wouldn't require any learning for it to execute or solve a problem.

Today, AI can be seen in different technologies such as image processing, natural language translation, speech recognition, and more. These technologies have been implemented in various different fields and can fulfil their objective of being an automated entity that goes on to solve tasks that humans would regularly take a longer period to solve.[3] [4]

[3] Burns, Ed. "What is Artificial Intelligence (AI)? | Definition from TechTarget." *TechTarget*, 2020, https://dokumen.pub/ai-and-machine-learning-for-coders-a-programmers-guide-to-artificial-intelligence-1nbsped-1492078190-9781492078197.html. Accessed 19 December 2022.

[4] Chollet, Francois. *Deep Learning with Python, Second Edition*. Manning, 2021. Accessed 19 December 2022

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

2.2 Understanding Machine Learning and its types

Machine learning (ML), is a subfield of AI, hence every feature associated with AI, would also apply to machine learning. The key differentiating factor is that, in the case of ML, it executes tasks based on feedback from which it produces the next output iteration, hence the name Machine Learning.

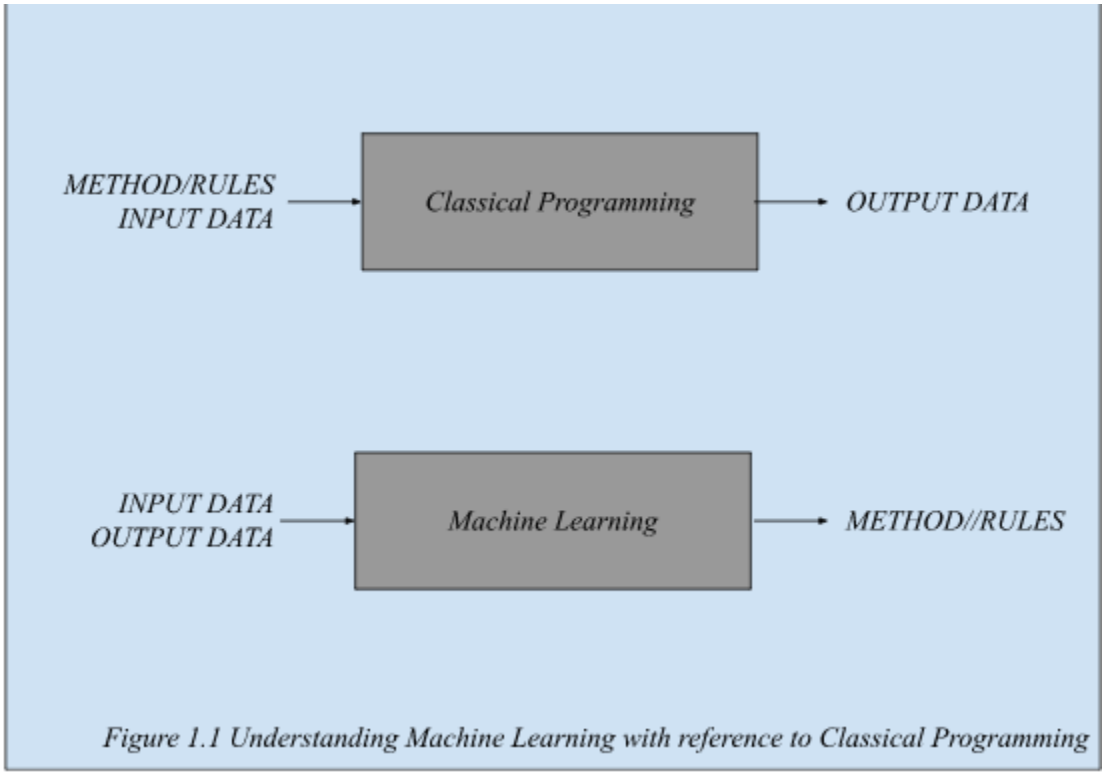


Figure 1.1 Understanding Machine Learning with reference to Classical Programming

Classical programming would require a human to establish the rules for the computer, and this can be called a computer program. The step-by-step information that a programmer would embed within their computer program, is what the program would use to

execute a given problem. This approach possesses many different limitations such as limited problems it can solve, and even more fundamental issues such as the way certain problems may be presented, hence these programs would require very well-defined problems.

In machine learning, the machine is trained to produce different methods of achieving a given output from a given input. This is different from traditional programs where the machine only gives an output based on explicit rules that it has been programmed with. Machine learning enables efficient and high-quality outputs by allowing the machine to execute tasks in the fastest possible way.

There are three different types of ML, which are Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

Supervised learning involves an expert in a particular field, who in this case is the supervisor, feeding in the example datasets and information to process for which the supervisor already knows the answers. A real-life example of this learning strategy is training for pattern recognition.[5] [6]

[5] Chollet, Francois. *Deep Learning with Python, Second Edition*. Manning, 2021. Accessed 19 December 2022

[6] Sarker, Iqbal H. "Machine Learning: Algorithms, Real-World Applications and Research Directions." *SpringerLink*, SpringerLink, 22 March 2021, https://link.springer.com/article/10.1007/s42979-021-00592-x. Accessed 19 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

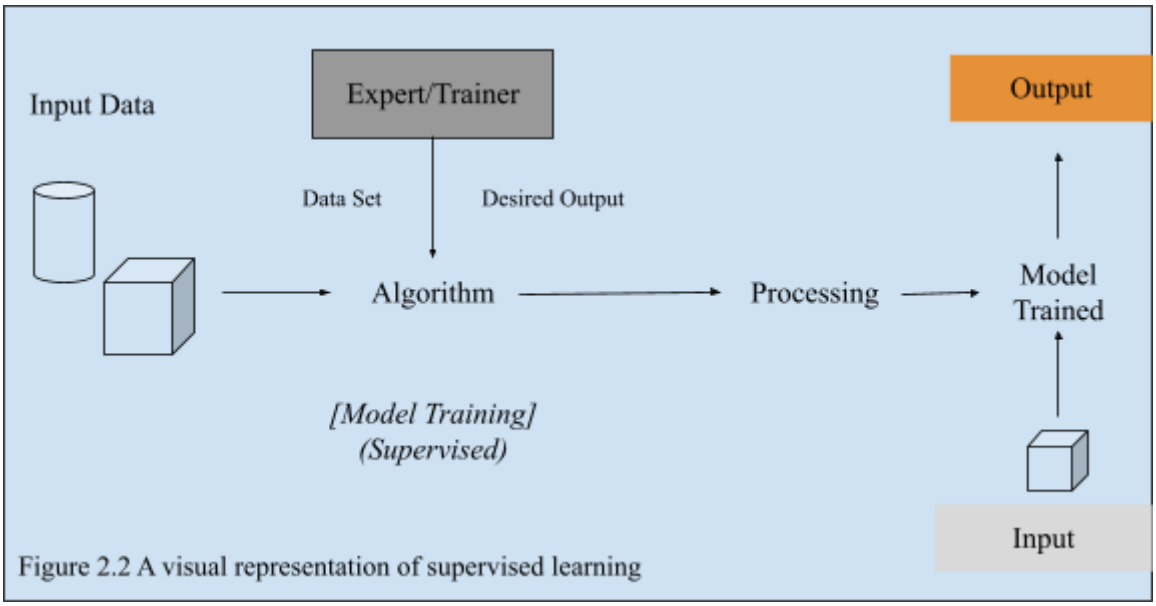


Figure 2.2 A visual representation of supervised learning

On the other hand, Unsupervised learning is required when there are no example datasets with known answers. An example is when the clustering algorithm is performed, where the ANN (Artificial Neural Networks) divides a set of elements into groups according to similarities between the data. This is useful to find insights from the provided data.

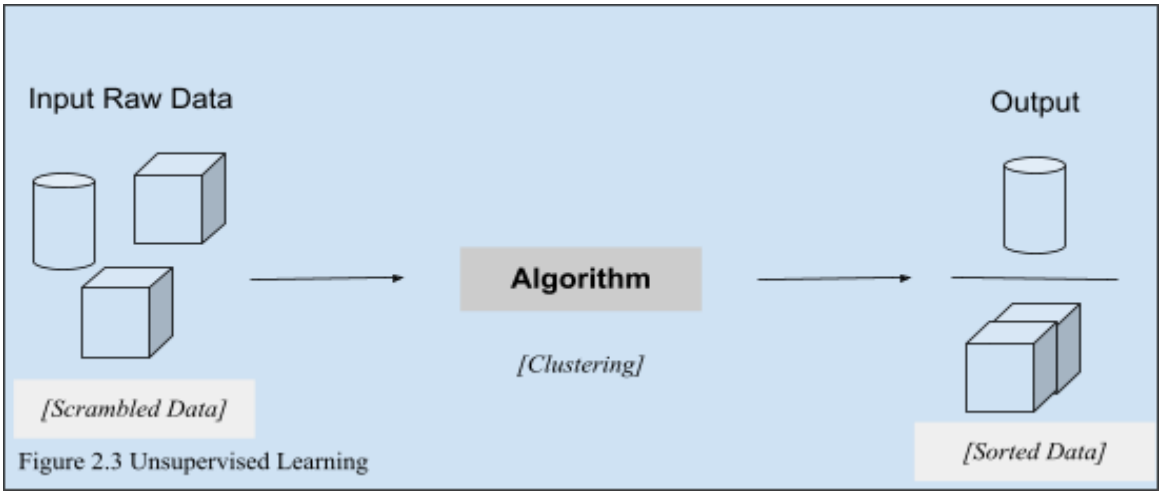


Figure 2.3 Unsupervised Learning

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Then there is reinforcement learning, which learns in an interactive environment, with trial and error utilising feedback that is received from its actions and experiences. A few recommender systems can be seen as reinforcement learning positive behaviour, such as rating content, which is rewarded with better recommendations. [7]

2.3 Understanding Deep Learning

Deep learning (DL) is a subfield of ML and it is a different approach to learning representation from data that emphasises the learning of successive layers with increasingly meaningful representations. It gets the name 'Deep' learning, from its concept of successive layers of representation. These layers that contribute to a model can be referred to as the depth of the model. Modern deep learning models consist of tens if not hundrerds of successive layers of representations, and they are learned all from their exposure to the training input data as opposed to shallow learning, whose model is based on a few successive layers of representation, sometimes even two layers.

These layered representations are learned with models called neural networks, structured in literal layers stacked on top of each other. [8]

[7] International Baccalaureate. *Computer science Case study: May I recommend the following?* International Baccalaureate Diploma Programme, 2021, https://computersciencewiki.org/images/b/bc/2023_case_study.pdf. Accessed 20 December 2022
[8] Chollet, Francois. *Deep Learning with Python, Second Edition*. Manning, 2021. Accessed 19 December 2022

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

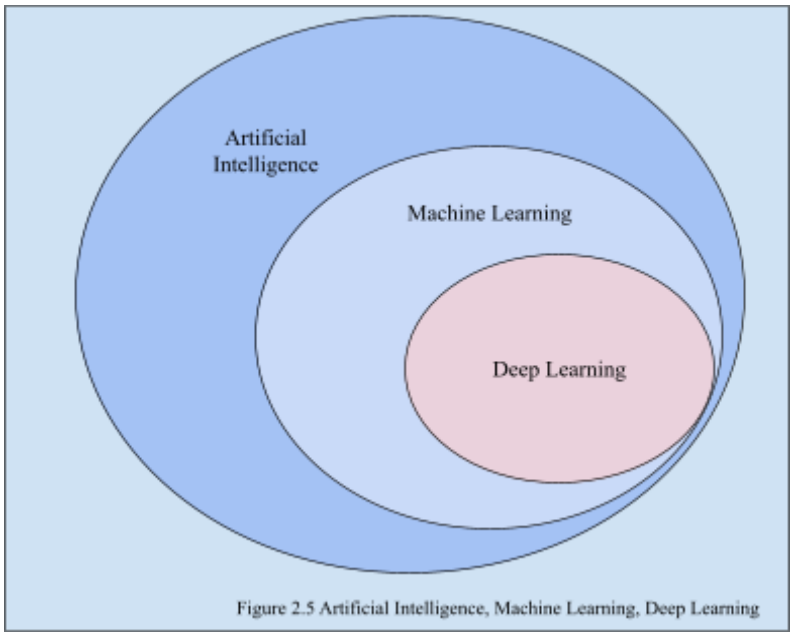


Figure 2.5 Artificial Intelligence, Machine Learning, Deep Learning

2.4 Understanding Neural Networks

The term Neural Networks also known as Artificial Neural Networks (ANNs) is about neurobiology, although not all deep learning models are models of the brain, as there is no evidence that the brain approaches or executes tasks like existing deep learning models. ANNs are a subset of ML and are the heart of DL algorithms.

ANNs are composed of node layers, which contain an input layer, one or more hidden layers, and finally an output layer. Each node can be understood as an artificial neuron interconnected with others and has an associated weight and threshold. [9]

[9] IBM. “What are Neural Networks?” *IBM*, IBM, 2022, https://www.ibm.com/in-en/topics/neural-networks. Accessed 19 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

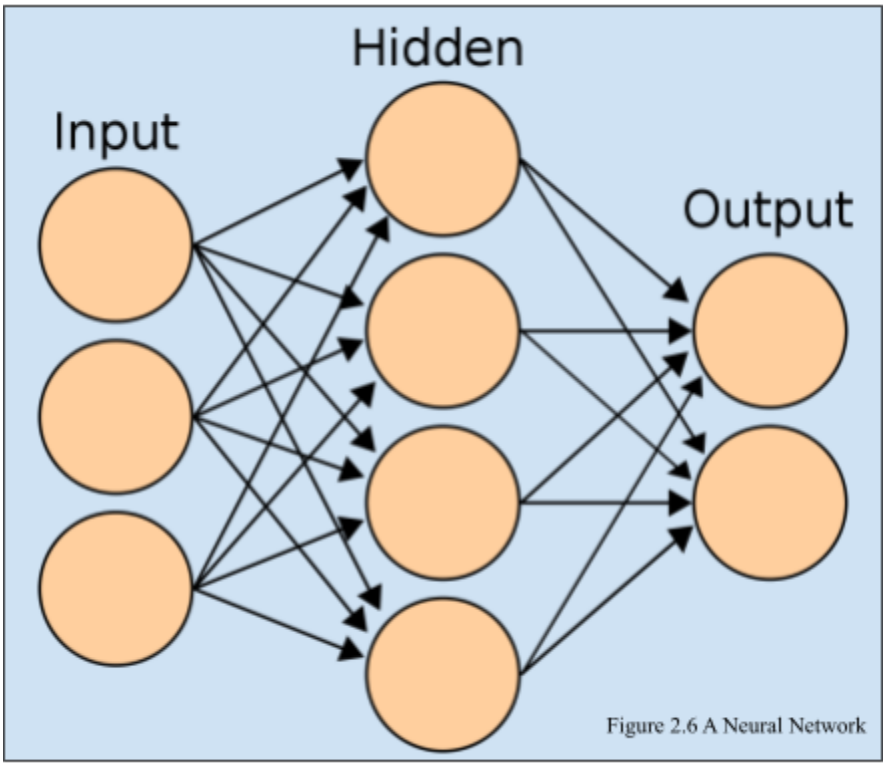


Figure 2.6 A Neural Network

DL and ANNs tend to be used interchangeably, rather the case is that a neural network that has three or more nodes can be termed as a DL algorithm, and this is with the inclusion of the input and output nodes. Anything lower than this can be classed as a basic Neural Network.

Neural Networks or Artificial Neural Networks have been applied to complex tasks which range from speech recognition to tasks within computational biology such as the classification of cancers, prediction of protein secondary structure, and gene prediction.[10]

[10] Chollet, Francois. *Deep Learning with Python, Second Edition*. Manning, 2021. Accessed 19 December 2022

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

2.5 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), were introduced by Ian Goodfellow in the year 2014. They allow for the generation of rather realistic synthetic images by causing the generated images to be statistically almost indistinguishable from the original images. GAN is an effective type of generative model, which has been a subject of great interest in the ML field of study.

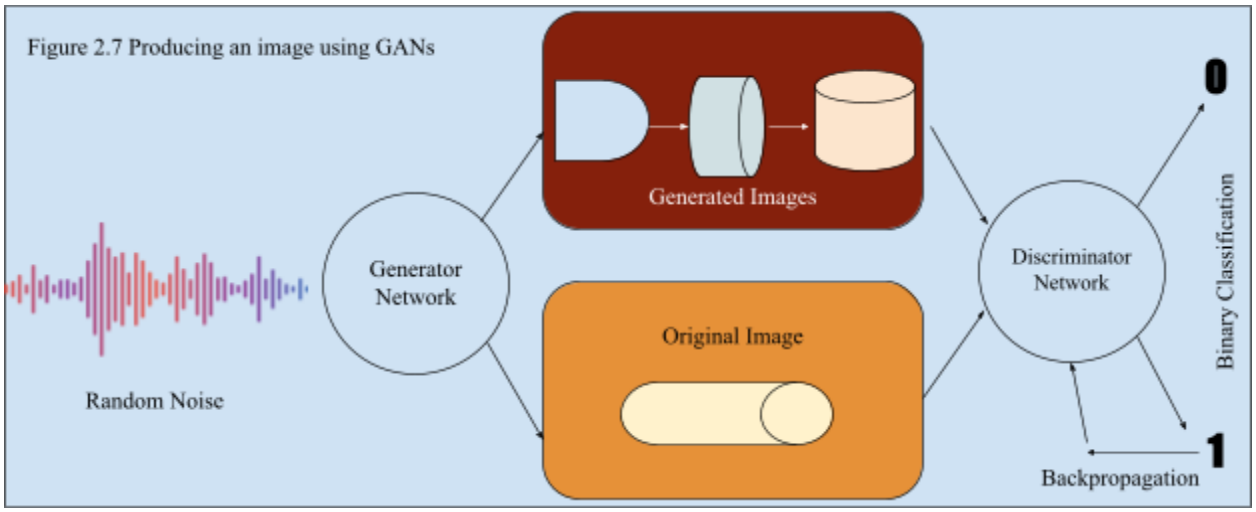
A GAN consists of two competing networks, a generator network, and a discriminator network. It is important to note that both of these elements are Artificial Neural Networks.

The generator attempts to produce random synthetic outputs such as images of real-life objects, whilst the discriminator tries to classify between the original image and the generated image. As iterations pass, the case with both networks is that they improve which, in the end, will give us the result being an image produced by a generator that will be almost visually and statistically indistinguishable from the original.

The way a discriminator classifies the original image and the produced image is by using an algorithm called binary classification, where it labels the produced image as 0, and when the produced images get indistinguishable it will label it as 1 and the program would terminate

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

In essence, GANs are neural networks that learn to choose from a special distribution from which we get the term generative, and they do so by setting up a competition, and there we get the term adversarial.[11]



In addition to this play a crucial role in the performance of GANs. As mentioned earlier, we are going to be focussing mostly on three hyperparameters, which are: The choice of learning rate, optimizer type, and a number of hidden layers that can greatly impact the ability of the GAN to generate high-quality images and the time required for training. By carefully tuning these hyperparameters, it is possible to achieve better results with faster training times. In this study, we aim to investigate the effects of different hyperparameter configurations on GAN performance.

[11] Ahmed, Ryan. "TensorFlow 2.0 Practical Advanced." *Udemy*, Udemy, 21 2 2021, https://www.udemy.com/course/tensorflow-2-practical-advanced/learn/lecture/16266470#content. Accessed 28 November 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

3 Experiment

3.1 Methodology

For this portion of the study, we will be using TensorFlow, an open-source machine-learning framework developed by Google, to train and evaluate GANs.[12] The specific hyperparameters that we will be adjusting and tuning are the batch size, buffer size and learning rate.

The batch size is the number of samples processed in one forward/backward pass of the network.[13] The buffer size is the maximum number of elements that will be buffered when prefetching datasets.[14] The learning rate is a hyperparameter that determines the step size at each iteration while moving toward a lower loss function. It controls how much the model weights are adjusted with respect to the gradient of the loss.[15]

For the dataset, we will be using the MNIST (Modified National Institute of Standards and Technology) dataset, which is a large database of handwritten digits used for training various machine learning-enabled models. It contains a total of 60,000 examples and a test of 10,000 examples. This will be the only dataset used in the experiment, as including too many variables can make it difficult to isolate the effects of each variable.

---

[12] Google TensorFlow. "GAN Model - Colaboratory." *Google Colab*, Google Colab, 2019, https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb. Accessed 19 December 2023.

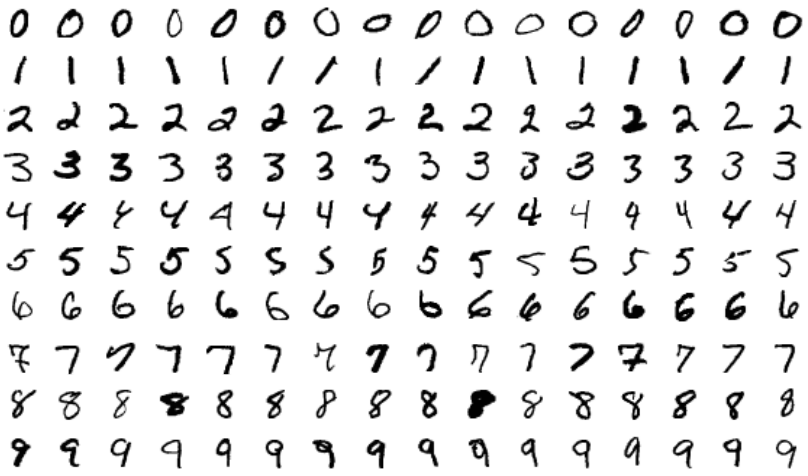
[13] Brownlee, Jason. "Difference Between a Batch and an Epoch in a Neural Network - MachineLearningMastery.com." *Machine Learning Mastery*, 10 August 2022, https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. Accessed 28 December 2022.

[14] TensorFlow. "TensorFlow v2.11.0." *TensorFlow*, 18 November 2022, https://www.tensorflow.org/api_docs/python/tf/data/experimental/shuffle_and_repeat. Accessed 28 December 2022.

[15] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. *Google Books*, https://www.google.com/books/edition/Machine_Learning/NZP6AQAAQBAJ?hl=en&gbpv=0&kptab=sideways. Accessed 28 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Figure 3.1 Sample images from the MNIST Dataset



[16]

We will be using the in-built GPU of the machine for hardware acceleration, with other background applications and tasks being minimal or inactive. The code is based on Python, which will be executed using Google Colaboratory, a machine learning and data analysis tool that allows a combination of executable Python code on the web.[17] The code we will be using is from the GANs subpage of the Google TensorFlow Core program.[18]

It is important to note that, when these hyperparameters are being tested, they are being tested individually, which in essence means that, when one hyperparameter is being tested, the other hyperparameters remain in their defunct configuration. The default configuration of the buffer size, batch size and learning rate are 60,000, 256, and 1e-4 respectively.

---

[16] Kaggle. “MNIST Dataset.” *Kaggle*, Kaggle, 2022, https://www.kaggle.com/competitions/digit-recognizer. Accessed 19 December 2022.

[17] Google. “Google Colab.” *Google Research*, Google Research, 2017, https://research.google.com/colaboratory/faq.html. Accessed 1 December 2023.

[18] “Deep Convolutional Generative Adversarial Network.” *TensorFlow*, 15 December 2022, https://www.tensorflow.org/tutorials/generative/dcgan. Accessed 19 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

The results will be analysed both visually and statistically. The visual analysis will be based on its similarity to the original dataset and the recognizability of the digits. Overall, an output of 16 digits would be executed. The statistical analysis will be based on the discriminator and generator loss values. In machine learning, Loss values are a measure of how well a machine learning model is performing on a task. They represent the difference between the model's predicted outputs and actual outputs and are used to adjunct the model's parameters to improve its performance.[19]

The generator model is trained using the adversarial loss function, which compares the discriminator's predictions on the generated images to a vector of 1s (indicating that the images are real). The generator tries to minimise this loss, which encourages it to create images that can fool the discriminator.

The discriminator model is trained using the binary cross-entropy loss function, which measures the difference between the discriminator's predictions and the true labels (1 for real images, 0 for fake images). The discriminator tries to minimise this loss, which helps it learn to correctly distinguish between real and fake images. The loss values can be plotted on a line graph, and we can use this to understand how well the model is performing. Whilst the model is being executed, the time of execution is being recorded by the code executer, and this will be used to see if the time complexity of execution is affected by the change of hyperparameters.

[19] Brownlee, Jason. "Loss and Loss Functions for Training Deep Learning Neural Networks - MachineLearningMastery.com." *Machine Learning Mastery*, 28 January 2019, https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. Accessed 7 January 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

[20]

Below are the configurations that will be tested:\

Table 3.1 Configurations for testing the effect of buffer size

	Buffer Size	Batch Size	Learning Rate
Configuration 1	30,000	256	1e-4
Configuration 2	60,000	256	1e-4
Configuration 3	120,000	256	1e-4

Table 3.2 Configuration for testing the effect of buffer size

	Buffer Size	Batch Size	Learning Rate
Configuration 1	60,000	128	1e-4
Configuration 2	60,000	256	1e-4
Configuration 3	60,000	512	1e-4

Table 3.2 Configuration for testing the effect of Learning Rate

	Buffer Size	Batch Size	Learning Rate
Configuration 1	60,000	256	5e-5
Configuration 2	60,000	256	1e-4
Configuration 3	60,000	256	2e-4

[20] O'Reilly Media. “4. Generative Adversarial Networks - Generative Deep Learning [Book].” *O'Reilly Media*, O'Reilly Media, 2018, https://www.oreilly.com/library/view/generative-deep-learning/9781492041931/ch04.html. Accessed 7 January 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

3.2 Limitations of The Experiment

The experiment's findings may be limited by the exclusive use of the MNIST dataset, though it contains a large number of sample images, the effect of the parameters on the data set may be exclusive to this dataset. Therefore, the results obtained from this dataset may not be readily applicable to other datasets or real-world images. Additionally, most real-world projects are based on much larger datasets and iterations.[21] With regards to the output, the 16 images that the model would be generating, would more often than not, feature a different set of digits within them, and this would make it more difficult to pass a verdict on which image is greater in detail and quality.

Concerning the hyperparameters, The experiment only tests a limited number of hyperparameters, including the batch size, buffer size, and learning rate. Other hyperparameters, such as learning rate and activation functions, are not considered. Therefore, the results may not provide a comprehensive understanding of how all hyperparameters influence GAN performance and time complexity.

With regard to hyperparameters, specifically, the hyperparameters batch size and buffer size would be primarily affected due to hardware limitations, for they require a higher level of memory allocation and computational resources to process the dataset.

[21] *Midjourney*, 2022, https://www.midjourney.com/home/?callbackUrl=%2Fapp%2F. Accessed 7 January 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

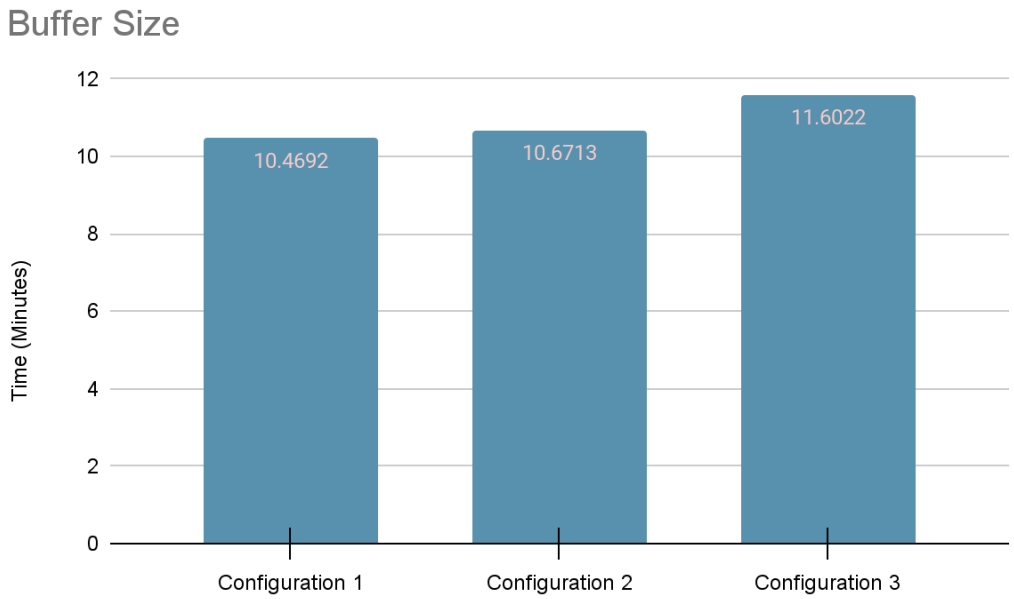
Lastly, the experiment design itself, since the experiments test each hyperparameter individually, which may not reflect a real-world scenario where there would be multiple hyperparameters adjusted as the code is being executed simultaneously in more advanced software platforms, and these methods give more fine-tuned parameters for a particular data set, which in turns maximises the performance of the model.

That being said, all of these limitations affect the legitimacy of the experiment results on a very minimal scale and this experiment is very much appropriate for the scale of this project.

3.3 Results and Analysis

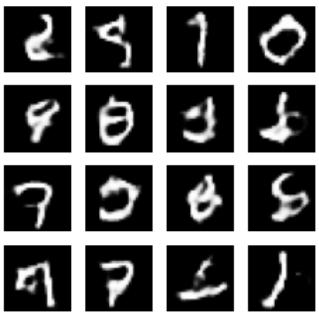
3.3.1 Buffer Size

Graph 3.1 Time taken for 50 epochs

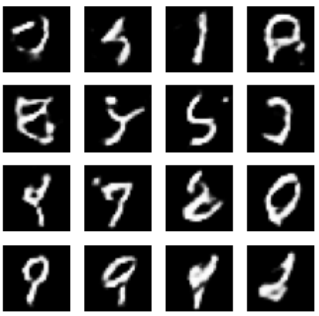


How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

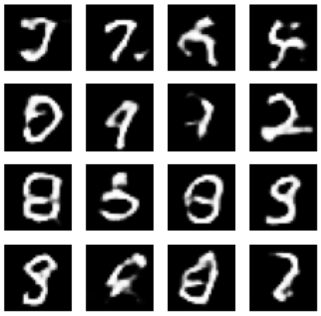
Figure 3.2 The 50th Epoch for the respective buffer sizes and the Loss Values



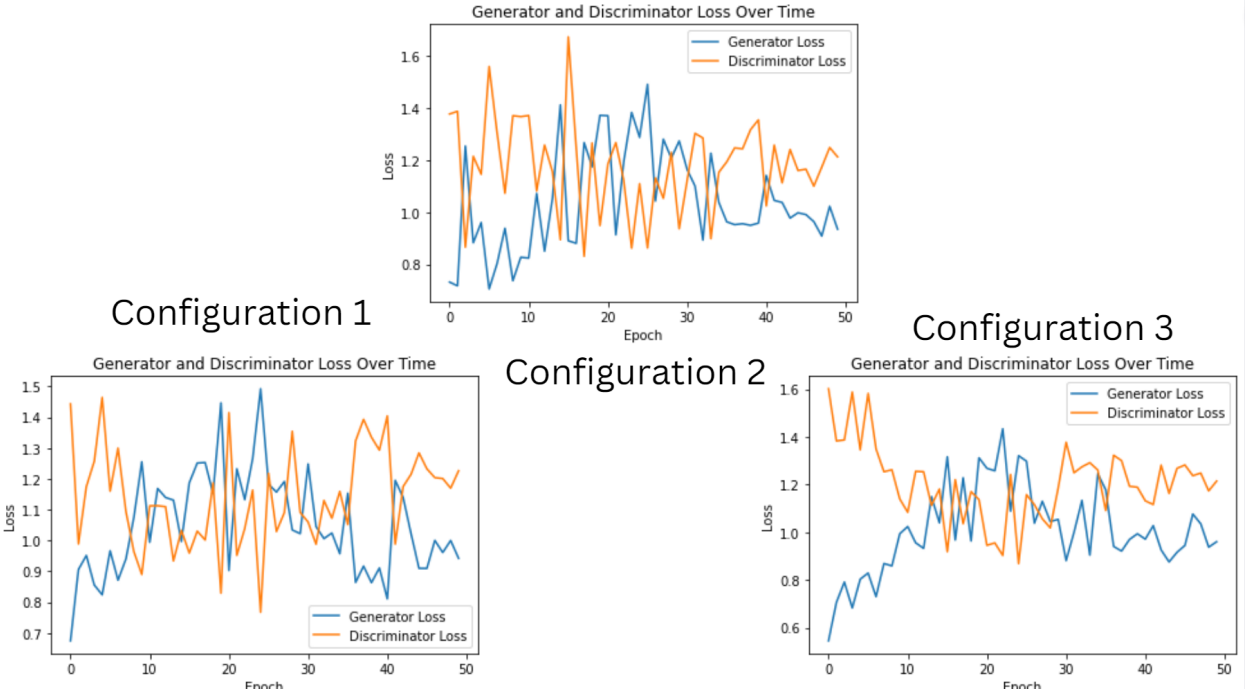
Configuration 1



Configuration 2



Configuration 3



How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

In this experiment, we found that increasing the buffer size did not significantly improve the final image quality, but it did increase the time taken to complete the training. This is because the buffer size determines the number of elements that will be buffered when prefetching datasets. A larger buffer size means more elements will be prefetched, which can increase the time taken to complete each iteration. Therefore, if time is a critical factor, it might be preferable to use a smaller buffer size, even if it means sacrificing some image quality and this would be especially ideal when working with large-scale projects where factors such as time and image quality, tend to compound into something much larger, for longer executions with more advanced systems and algorithms.[22]

With regards to the loss values, in a general scenario, it would be ideal that the generator loss values decrease over time, and this means that the generating is getting better at producing realistic images. At the same we want the Discriminator loss to increase, indicating that the discriminator is better at distinguishing between real and fake.

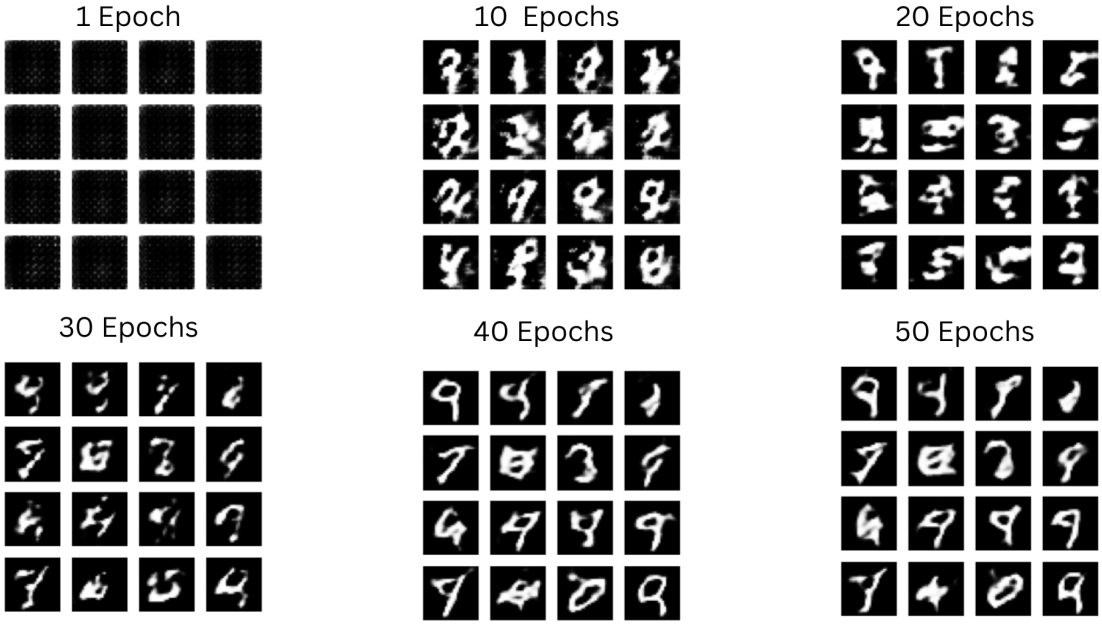
The loss plot showed that during the early stages of training, the generator and discriminator values fluctuated significantly. However, as training progressed, the generator value tended to decrease while the discriminator value continued to fluctuate. When the execution was reaching close to reaching the 50th epoch then it could be seen that the generator loss value starts decreasing and the discriminator value starts to increase. This trend indicates that the discriminator is becoming more effective at distinguishing between real and fake images, while

[22] "What is the meaning of longer buffer size?" *The Healthy Journal*, 2020, https://www.thehealthyjournal.com/faq/what-is-the-meaning-of-longer-buffer-size. Accessed 2 February 2023.

the generator is having a harder time generating realistic images. In other words, as the training progresses, the discriminator becomes more skilled at identifying fake images, which makes it more difficult for the generator to generate high-quality images, and eventually (with more iterations), both curves would flatten.

The other observation that could be made is that the entire execution for all three configurations was free from instability and uncertainty in execution. Instability and uncertainty in the context of this experiment would be the idea of unpredictability with regard to the output of the execution that is taking place. It is important to address the issue of stability and consistency as this would give the idea of the probable performance of a model with a higher level of stability.[23]

Figure 3.3 The progress the model makes throughout the iterations (Configuration 2)



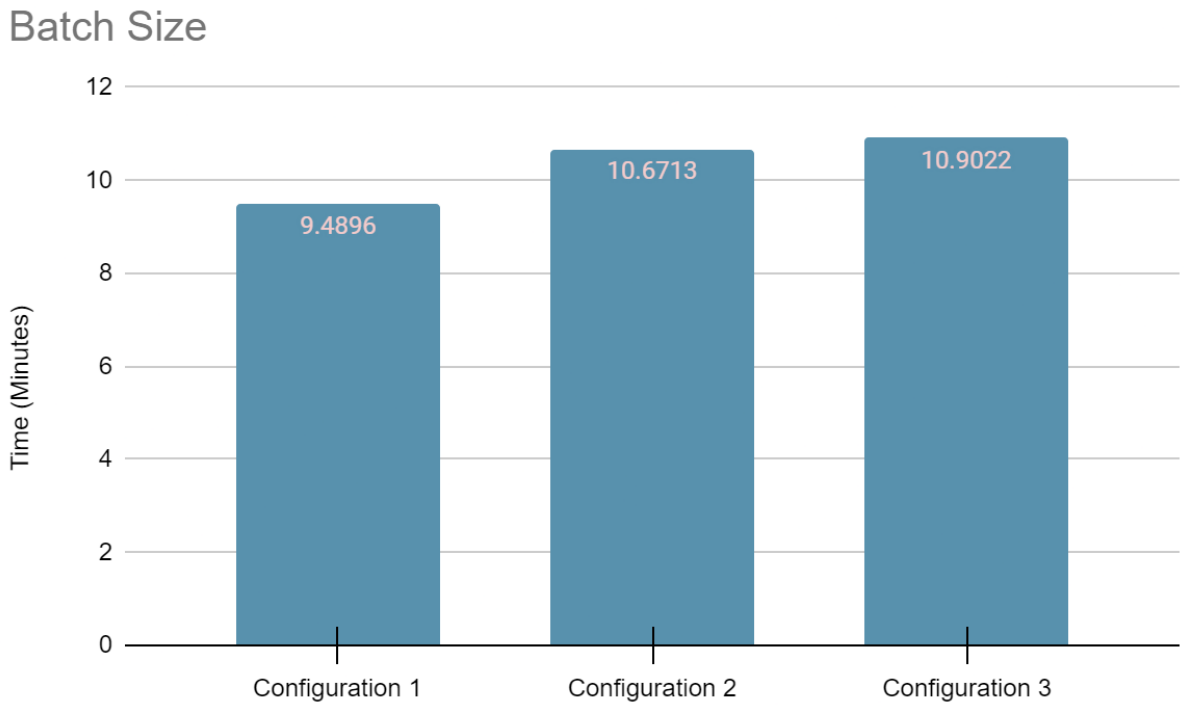
[23] Brownlee, Jason. "How to Identify and Diagnose GAN Failure Modes - MachineLearningMastery.com." *Machine Learning Mastery*, 8 July 2019, https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/. Accessed 20 December 2022.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Here we can see the improvements made by the generator throughout the 50 iterations and how it reached a set of recognisable numerical digits from something that is of no real value. This also illustrates the way in which GANs work, in order to produce a statistically and visually similar if not indistinguishable image from the dataset.

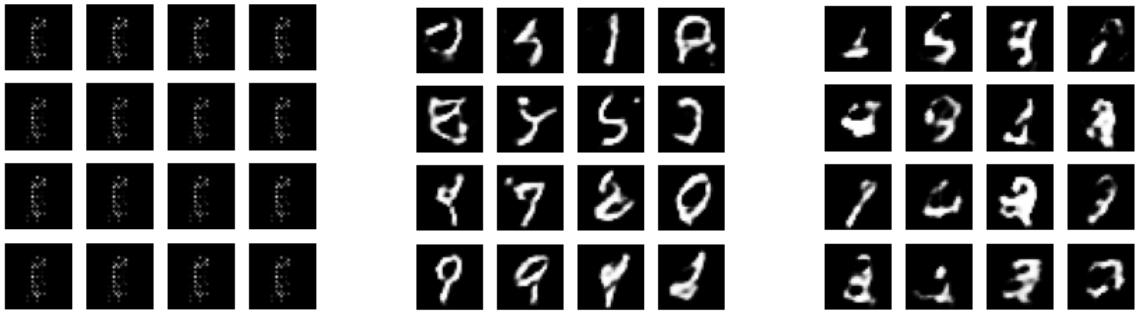
3.3.2 Batch Size

Graph 3.2 Time taken for 50 epochs

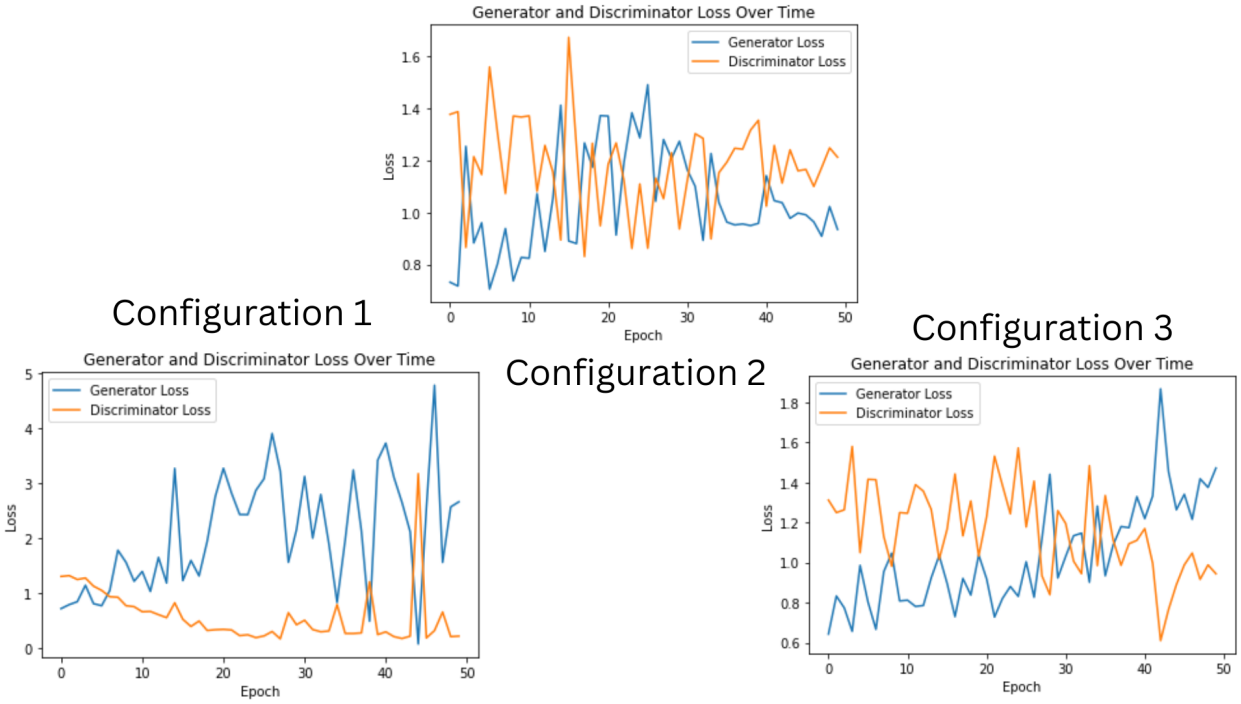


How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Figure 3.3 The 50th Epoch for the respective batch sizes and the Loss Values



Configuration 1 Configuration 2 Configuration 3



In this portion of the experiment, we evaluated the effect of changing the batch size parameter on the training of a GAN. We found that increasing the batch size led to an increase in the time taken to execute the training. On the other hand, lowering the batch size resulted in a decrease in

the final image quality, with the final output bearing no resemblance to the sample datasets, and mostly blank.

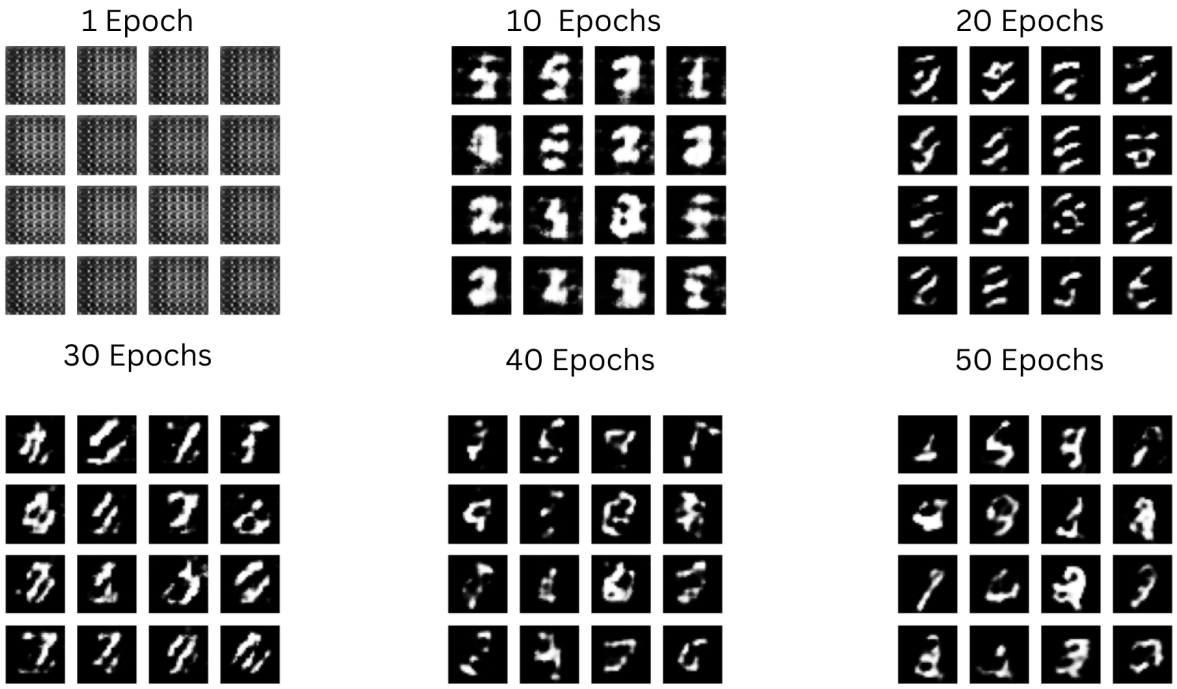
The increase in the batch size led to an increase in the time taken to complete the training. This was because the batch size determines the number of samples that are processed in each iteration. Larger batch sizes result in fewer iterations needed to complete an epoch, but each iteration takes longer to execute due to the increased number of samples. Therefore, if time is a critical factor, it might be preferable to use a smaller batch size, even if it means sacrificing some image quality.

Lowering the batch size, on the other hand, had a detrimental effect on the final image quality. The final output had no resemblance to any of the sample datasets and was mostly blank. This indicates that the generator was unable to produce high-quality images when working with such small batches. The loss plot showed that during a large part of the test, the generator values were higher than the discriminator loss values. This trend indicates that the generator was struggling to produce realistic images, while the discriminator was becoming more effective at distinguishing between real and fake images.

With regards to the stability and the consistency of the model's performance, it could be said that when reducing the batch size, though the final iteration of the model, was excessively poor, it was consistent throughout, and it could be rather easily noticed that there is something out of order or defective, hence this may lead someone to stop the execution midway and try troubleshooting the issue in hand. Increasing the batch size did bring a lot of instability, for it could be noticed that the initial iteration of the model with this configuration, tends to produce promising results, and this can also be confirmed by the graph, where the generator loss values were for the most within the first half, lower than the discriminator value, until it spiked up, where the image started to lose its fidelity which in turn affected the final output.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

Figure 3.4 The progress the model makes throughout the iterations (Configuration 3)



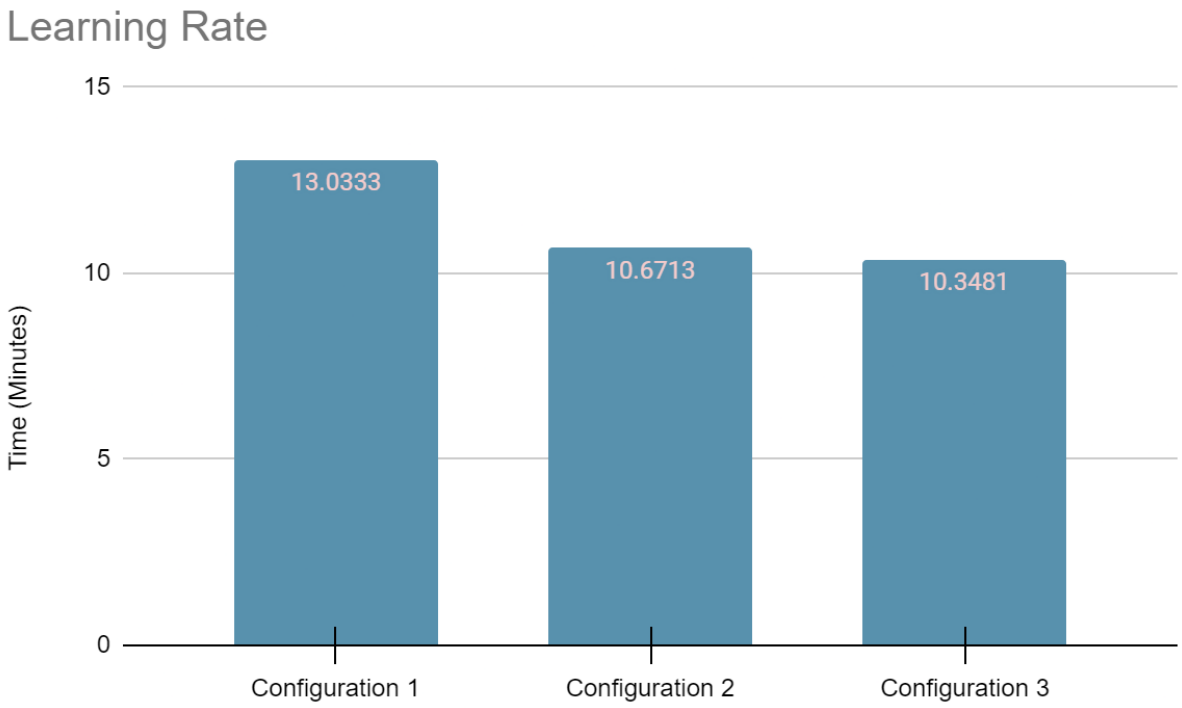
In conclusion, we found that changing the batch size can have a significant effect on the training of a GAN. Increasing the batch size led to longer training times while lowering the batch size resulted in a decrease in the final image quality. The optimal batch size depends on the specific requirements of the project, but in general, a larger batch size can lead to faster training times at the expense of image quality, while a smaller batch size can lead to higher image quality at the expense of longer training times.[24]

[24] Shen, Kevin. "Effect of batch size on training dynamics | by Kevin Shen | Mini Distill." *Medium*, Medium, 19 June 2018, https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e. Accessed 2 February 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

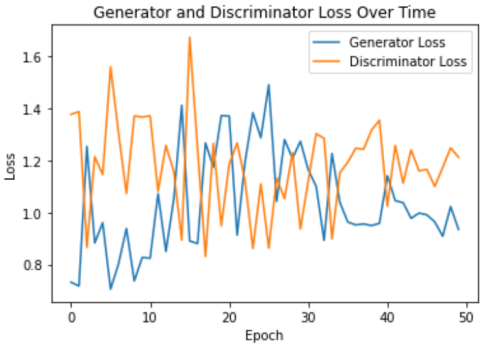
3.3.3 Learning Rate

Graph 3.3 Time taken for 50 epochs

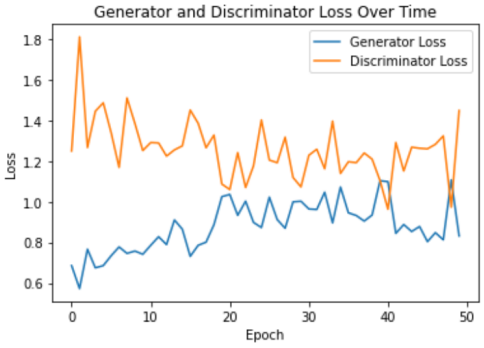


Configuration 1 Configuration 2 Configuration 3

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

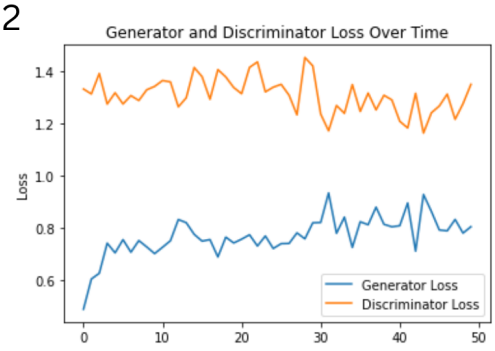


Configuration 1



Configuration 2

Configuration 3



In this experiment, we tested the impact of changing the learning rate on the performance of a GAN model. We found that increasing the learning rate helped to increase the speed of the execution while decreasing it led to a slower execution time. However, the images generated did not exhibit a significant improvement in quality. It is worth noting that this finding may be specific to the data set used in this experiment and could be improved with more iterations or other modifications to the model architecture.

When examining the loss plots for the model with the lower learning rate, we observed that the generator value was initially increasing while the discriminator value was decreasing until they intersected for a moment, and this happened twice. In the end, the generator value increased, and the discriminator value decreased. This behaviour suggests that the generator was initially successful in generating realistic images that the discriminator found difficult to differentiate

from the real ones. However, as training progressed, the discriminator became more skilled at identifying fake images, which made it more challenging for the generator to generate high-quality images.

On the other hand, the loss plot for the model with the higher learning rate showed that the two curves never intersect or even came close to each other. This suggests that the model was not able to achieve the same level of balance between the generator and discriminator as in the previous model. The final output was not better than the other configurations, indicating that simply increasing the learning rate does not always lead to improved performance.

In conclusion, changing the learning rate can have a significant impact on the speed of execution of a GAN model, but it does not always result in improved image quality. It is essential to experiment with different learning rates and carefully analyze the loss plots to identify the optimal value for the specific data set and model architecture.

Overall, it is important yet again to note that these are not the only hyperparameters, though they are present in every GAN model, the final results could be more diverse and profound when there are more hyperparameters being tested with different larger datasets, using advanced software.

That being said, we can still use these different tests to come to the conclusion that it is true that the hyperparameters of GAN affect its ability to generate high-quality images and the time taken

to execute different iterations. The influence of these hyperparameters could be both positive and negative and this depends on how one changes the parameters, to what extent, and the datasets they are being tested with.

4. Further Research Opportunities

While the results of this study provided insights into the effect of different hyperparameters on the performance of GANs, there were limitations to the study, such as the relatively small dataset and the use of a single architecture. Further research could explore the performance of GANs on larger datasets with more complex architectures and hyperparameters.

This could include exploring different regularisation techniques, optimizer algorithms, and architectures such as progressive GANs. The practical applications of GANs and hyperparameters include image and video generation, data augmentation, and domain adaptation, among others[25]. The exploration of new hyperparameters and architectures can help improve the performance of GANs in these applications.

[25] Köker, Raşit. "Generative Adversarial Network Technologies and Applications in Computer Vision." *Hindawi*, Hindawi, 2020 August 1, https://www.hindawi.com/journals/cin/2020/1459107/. Accessed 20 Febuary 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

5 Conclusion

In conclusion, the hyperparameters of GANs, including the batch size, buffer size, and learning rate, have a significant impact on the quality of the generated images and the time complexity of the model. Through the experimentation process, we found that increasing the batch size and buffer size can improve the image quality, but at the cost of increased time complexity. In contrast, increasing the learning rate can decrease the time complexity, but may not necessarily improve the image quality.

Despite the challenges faced due to the limitations of the dataset and system, this research provides valuable insights into the influence of hyperparameters on GANs. Future research can focus on utilising larger datasets and exploring more hyperparameters to improve the performance of GANs. The practical applications of GANs and hyperparameters in various fields, such as image and video synthesis, hold promising potential for future advancements.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

6. Works Cited

1. **Ahmed, Ryan. "TensorFlow 2.0 Practical Advanced." *Udemy*, Udemy, 21 2 2021, https://www.udemy.com/course/tensorflow-2-practical-advanced/learn/lecture/16266470#content. Accessed 28 November 2022.**
2. **Baeldung. "Epoch in Neural Networks." Baeldung, Baeldung, 11 November 2022, https://www.baeldung.com/cs/epoch-neural-networks. Accessed 21 December 2022.**
3. **Brownlee, Jason. "How to Identify and Diagnose GAN Failure Modes - MachineLearningMastery.com." *Machine Learning Mastery*, 8 July 2019, https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/. Accessed 20 December 2022.**
4. **Burns, Ed. "What is Artificial Intelligence (AI)? | Definition from TechTarget." *TechTarget*, 2020, https://dokumen.pub/ai-and-machine-learning-for-coders-a-programmers-guide-to-artificial-intelligence-1nbsped-1492078190-9781492078197.html. Accessed 19 December 2022.**
5. **Chollet, Francois. *Deep Learning with Python, Second Edition*. Manning, 2021.**
6. **"Deep Convolutional Generative Adversarial Network." *TensorFlow*, 15 December 2022, https://www.tensorflow.org/tutorials/generative/dcgan. Accessed 19 December 2022.**
7. **GitHub. "[Question] Batch, Buffer, Epoch, Time Horizon ? · Issue #1837 · Unity-Technologies/ml-agents." *GitHub*, GitHub, 18 March 2019, https://github.com/Unity-Technologies/ml-agents/issues/1837. Accessed 24 December 2022.**
8. **Google. "Google Colab." *Google Research*, Google Research, 2017, https://research.google.com/colaboratory/faq.html. Accessed 1 December 2023.**
9. **Google TensorFlow. "GAN Model - Colaboratory." *Google Colab*, Google Colab, 2019, https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb. Accessed 19 December 2023.**
10. **Huang, Yongchao. "Kullback-Leibler (KL) Divergence and Jensen-Shannon Divergence." *Yongchao Huang*, Yongchao Huang, 8 July 2020, https://yongchaohuang.github.io/2020-07-08-kl-divergence/. Accessed 19 December 2022.**
11. **IBM. "What are Neural Networks?" *IBM*, IBM, 2022, https://www.ibm.com/in-en/topics/neural-networks. Accessed 19 December 2022.**

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

12. International Baccalaureate. ***Computer science Case study: May I recommend the following?*** International Baccalaureate Diploma Programme, 2021, https://computersciencewiki.org/images/b/bc/2023_case_study.pdf.
13. Kaggle. "Fashion MNIST." ***Kaggle***, Kaggle, 2022, https://www.kaggle.com/datasets/zalando-research/fashionmnist. Accessed 19 December 2022.
14. Kaggle. "MNIST Dataset." ***Kaggle***, Kaggle, 2022, https://www.kaggle.com/competitions/digit-recognizer. Accessed 19 December 2022.
15. Kahng, Minsuk. "GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation." ***GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation***, GAN Lab, 2018, https://minsuk.com/research/papers/kahng-ganlab-vast2018.pdf. Accessed 19 December 2022.
16. "Lesson 12: Deep Learning Part 2 2018 - Generative Adversarial Networks (GANs)." ***Deep Learning Part 2 2018***, created by Jeremy Howard, season Generative Adversarial Networks (GANs), episode 12, 2018. Accessed 27 December 2022.
17. Mozilla. "Getting started with WebGL - Web APIs | MDN." ***MDN Web Docs***, 18 February 2015, https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Getting_started_with_WebGL. Accessed 7 January 2023.
18. Neptune.ai. "Understanding GAN Loss Functions - neptune.ai." ***Neptune.ai***, Neptune.ai, 2020, https://neptune.ai/blog/gan-loss-functions. Accessed 22 Febuary 2023.
19. Sarker, Iqbal H. "Machine Learning: Algorithms, Real-World Applications and Research Directions." ***SpringerLink***, SpringerLink, 22 March 2021, https://link.springer.com/article/10.1007/s42979-021-00592-x. Accessed 19 December 2022.
20. Murphy, Kevin P. ***Machine Learning: A Probabilistic Perspective***. MIT Press, 2012. ***Google Books***, https://www.google.com/books/edition/Machine_Learning/NZP6AQAAQBAJ?hl=en&gbpv=0&kptab=sideways. Accessed 28 December 2022
21. Brownlee, Jason. "Loss and Loss Functions for Training Deep Learning Neural Networks - MachineLearningMastery.com." Machine Learning Mastery, 28 January 2019, https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. Accessed 7 January 2023.
22. O'Reilly Media. "4. Generative Adversarial Networks - Generative Deep Learning [Book]." ***O'Reilly Media***, O'Reilly Media, 2018, https://www.oreilly.com/library/view/generative-deep-learning/9781492041931/ch04.html. Accessed 7 January 2023.

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

23. **Midjourney,2022,https://www.midjourney.com/home/?callbackUrl=%2Fapp%2F. Accessed 7 January 2023.**
24. **"What is the meaning of longer buffer size?" *The Healthy Journal*, 2020, https://www.thehealthyjournal.com/faq/what-is-the-meaning-of-longer-buffer-size. Accessed 2 February 2023.**
25. **Shen, Kevin. "Effect of batch size on training dynamics | by Kevin Shen | Mini Distill." *Medium*, Medium, 19 June 2018, https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e. Accessed 2 February 2023.**
26. **Köker, Raşit. "Generative Adversarial Network Technologies and Applications in Computer Vision." *Hindawi*, Hindawi, 2020 August 1, https://www.hindawi.com/journals/cin/2020/1459107/. Accessed 20 Febuary 2023.**

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

7. Appendix

Code for TensorFlow GAN model:

```
(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5  # Normalize the images to
[-1, 1]
BUFFER_SIZE = 60000
BATCH_SIZE = 256
# Batch and shuffle the data
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batc
h(BATCH_SIZE)
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)  # Note: None is the
batch size

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
```

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

```
    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2),
padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model
generator = make_generator_model()

noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)

plt.imshow(generated_image[0, :, :, 0], cmap='gray')
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                     input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model
discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)
# This method returns a helper function to compute cross entropy loss
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
def discriminator_loss(real_output, fake_output):
```

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

```
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,

discriminator_optimizer=discriminator_optimizer,
                                 generator=generator,
                                 discriminator=discriminator)
EPOCHS = 50
noise_dim = 100
num_examples_to_generate = 16

# You will reuse this seed overtime (so it's easier)
# to visualize progress in the animated GIF)
seed = tf.random.normal([num_examples_to_generate, noise_dim])
# Notice the use of `tf.function`
# This annotation causes the function to be "compiled".
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
      generated_images = generator(noise, training=True)

      real_output = discriminator(images, training=True)
      fake_output = discriminator(generated_images, training=True)

      gen_loss = generator_loss(fake_output)
      disc_loss = discriminator_loss(real_output, fake_output)
```

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity

```
    gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))

discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))
def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()

    for image_batch in dataset:
      train_step(image_batch)

    # Produce images for the GIF as you go
    display.clear_output(wait=True)
    generate_and_save_images(generator,
                             epoch + 1,
                             seed)

    # Save the model every 15 epochs
    if (epoch + 1) % 15 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)

    print ('Time for epoch {} is {} sec'.format(epoch + 1,
time.time()-start))

  # Generate after the final epoch
  display.clear_output(wait=True)
  generate_and_save_images(generator,
                           epochs,
                           seed)
train(train_dataset, EPOCHS)
```

How effectively do the hyperparameters of a GAN influence its ability to generate high-quality images and its time complexity