PROJECT REPORT FOR DEEP LEARNING.

Baseline Model when Epochs = 50.

A screenshot of a computer

Description automatically generated with low confidence

Baseline Model when EPOCHS = 10

A screenshot of a computer

Description automatically generated with low confidence

Baseline Model when EPOCHS=100

A screenshot of a computer

Description automatically generated with low confidence

Summarization of the above points.

**The baseline GAN model was implemented using TensorFlow and Keras.**

**The quality of the generated images was compared after 10 epochs, 50 epochs, and 100 epochs.**

**As the number of epochs increased, the quality of the generated images also improved. After 100 epochs, the images generated by the GAN were of much better quality than those generated after 10 epochs.**

b) Used ReLU activation (instead of LeakyReLU in the Baseline) in the generator for all layers except for the output, which uses a Tanh activation. Report the difference in the quality of images and training time at 50 epochs compared with that of the Baseline

O/p Baseline model when used RELU Activation Function when EPOCHS=50

A screenshot of a computer

Description automatically generated with low confidence

Summarization of point2:

In this point, ReLU activation was used in the generator for all layers except for the output, which used a Tanh activation.

The training time was similar to the baseline GAN model, but the quality of the generated images was slightly worse here.

c)

Change the following hyperparameters:

(1) the dimensionality of the noise vector,

(2) the batch size (i.e., the number of images per forward/backward pass),

(3) the learning rates, and

(4) the momentum terms.

Report the difference in the quality of images and training time at 50 epochs compared with that of the Baseline. Justify your choices of hyperparameter values

1. the dimensionality of the noise vector
2. The dimensionality of the noise vector determines the complexity of the input to the generator. A higher dimensional noise vector may allow the generator to generate more complex and diverse images, but may also increase training time. A lower dimensional noise vector may limit the generator's ability to generate diverse images.
3. To determine the optimal noise vector dimensionality, it's recommended to try a range of values (e.g., 50, 100, 200, etc.) and compare the quality of the generated images and training time. The optimal value will depend on the specific dataset and architecture used.

A picture containing text, font, screenshot

Description automatically generated

1. the batch size

2.1 ) Batch size: The batch size determines the number of images per forward/backward pass during training. A higher batch size may allow the model to converge faster, but may also require more memory and limit the diversity of the generated images. A lower batch size may allow for more diverse images, but may also require more training time.

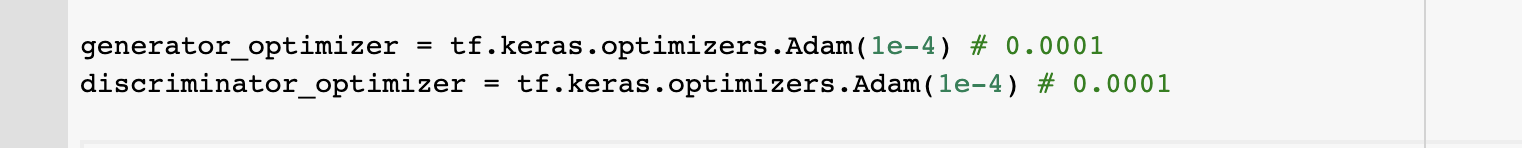
2.2 ) To determine the optimal batch size, it's recommended to try a range of values (e.g., 32, 64, 128, etc.) and compare the quality of the generated images and training time. The optimal value will depend on the specific dataset and architecture used.

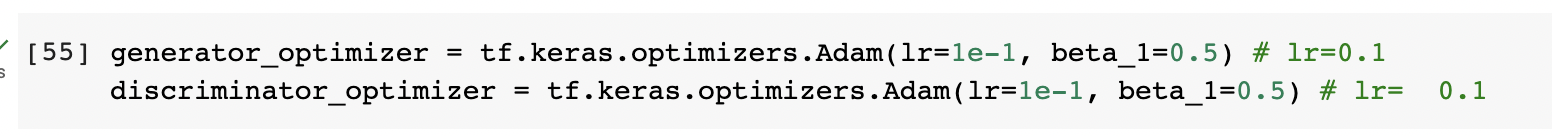
A picture containing text, font, screenshot, white

Description automatically generated

1. the learning rates

Learning rate: The learning rate determines the step size during gradient descent. A higher learning rate may cause the model to overshoot the optimal solution, while a lower learning rate may cause the model to converge too slowly.





When training a GAN model for digit recognition using TensorFlow and Keras, it's important to choose an appropriate learning rate value.

A learning rate that's too high may cause the model to overshoot the optimal solution, while a learning rate that's too low may cause the model to converge too slowly or get stuck in a suboptimal solution.

The optimal learning rate value may vary depending on the specific architecture and dataset used, but in general, it's recommended to start with a small value (e.g., 0.0002) and gradually increase it if the model is not converging fast enough.

It's also common to use learning rate schedules, which gradually decrease the learning rate over time to allow the model to converge more smoothly. Some popular learning rate schedules include step decay, exponential decay, and cosine annealing.

Overall, selecting an appropriate learning rate value and schedule is an important part of training GAN models for digit recognition using TensorFlow and Keras.

1. Momentum.
2. In the context of GANs using TensorFlow and Keras, the momentum term typically refers to the momentum parameter used in the optimization algorithm for updating the discriminator and generator networks during training.
3. In gradient descent optimization algorithms, such as stochastic gradient descent (SGD) and its variants like Adam, the update to the network weights is based on the gradient of the loss function with respect to the weights. In addition to the gradient, momentum can be introduced to the update rule to accelerate the optimization process and avoid local minima.
4. In the context of GANs using TensorFlow and Keras, the momentum term typically refers to the momentum parameter used in the optimization algorithm for updating the discriminator and generator networks during training.
5. In gradient descent optimization algorithms, such as stochastic gradient descent (SGD) and its variants like Adam, the update to the network weights is based on the gradient of the loss function with respect to the weights. In addition to the gradient, momentum can be introduced to the update rule to accelerate the optimization process and avoid local minima.
6. The momentum term is a hyperparameter that controls the contribution of the previous update step to the current update step. It is typically set to a value between 0 and 1, with a higher value resulting in a larger contribution from the previous update step. A value of 0 means no momentum, while a value of 1 means that the previous update step has the same weight as the current update step.

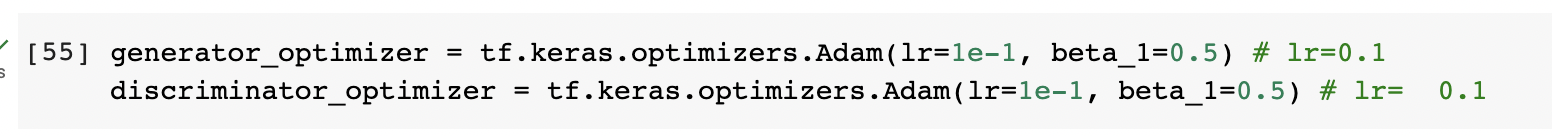
To use momentum in a GAN training process, you can specify the value of the momentum parameter when defining the optimizer for the discriminator and generator networks. For example, in TensorFlow and Keras, you can set the momentum parameter for the Adam optimizer using the beta\_1 parameter. The default value of beta\_1 is 0.9, which is a commonly used value for momentum.

Momentum terms: The momentum terms determine the contribution of the previous gradients to the current update. A higher momentum may help the model converge faster, but may also cause overshooting. A lower momentum may help the model converge more smoothly, but may also require more training time.

To determine the optimal momentum terms, it's recommended to try a range of values (e.g., 0.5, 0.9, 0.99, etc.) and compare the quality of the generated images and training time. The optimal value will depend on the specific dataset and architecture used.

A screenshot of a computer code

Description automatically generated with low confidence



A screenshot of a computer

Description automatically generated with medium confidence

In a Nutshell,

After trying a range of values for each hyperparameter, it's recommended to select the best hyperparameters based on the quality of the generated images and training time at a fixed number of epochs (e.g., 50 epochs) compared to the baseline model.

The hyperparameter values that provide the best balance of image quality and training time should be selected.