



**University
of Dayton**

ECE 595 Homework #4
7/27/2021
Christopher Couture Del Valle

Introduction:

Students were assigned to understand utilizing a denoising auto-encoder for cleaning noisy documents and generating new samples with generative models.

Methodology:

There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles or methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a separate sheet is much better from the point of view of the quality but requires giving more instructions and, more importantly, cost this type of acquisition is used. Guiding rulers printed on the sheet used for this reason. Light rectangles can be removed more easily whenever the handwritten text touches the rulers. Nevertheless, be taken into account: The best way to print these light rectangles

There are several classic spatial filters for reducing frequency noise from images. The mean filter, the median opening filter are frequently used. The mean filter is a filter that replaces the pixel values with the neighborhood of the image noise but blurs the image edges. The median of the pixel neighborhood for each pixel, thereby reducing noise. Finally, the opening closing filter is a mathematical morphology that combines the same number of erosion and dilation morph to eliminate small objects from images.

The main goal was to train a neural network in a situation to clean image from a noisy one. In this particular case

There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles or methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a separate sheet is much better from the point of view of the quality but requires giving more instructions and, more importantly, cost this type of acquisition is used. Guiding rulers printed on the sheet used for this reason. Light rectangles can be removed more easily whenever the handwritten text touches the rulers. Nevertheless, be taken into account: The best way to print these light rectangles

There are several classic spatial filters for reducing frequency noise from images. The mean filter, the median opening filter are frequently used. The mean filter is a filter that replaces the pixel values with the neighborhood of the image noise but blurs the image edges. The median of the pixel neighborhood for each pixel, thereby reducing noise. Finally, the opening closing filter is a mathematical morphology that combines the same number of erosion and dilation morph to eliminate small objects from images.

The main goal was to train a neural network in a situation to clean image from a noisy one. In this particular case

Figure 1: Example images from document denoising dataset: noisy image on the left and clear images on the right.

Task 1 was to implement and evaluate an Auto-Encoder (AE) model for denoising the document files. The database was provided on Isidore including training and testing samples and images are shown in Figure 1. After training the model successfully, qualitative and quantitative analysis was provided on testing samples.

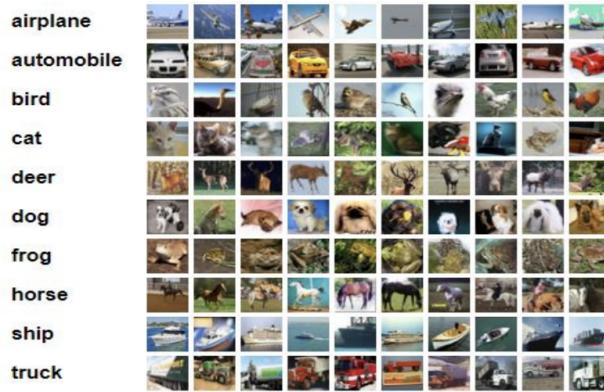


Figure 2: Cifar-10 Dataset

Task 2 was to implement generative models: The Convolutional Variational Auto-Encoder (VAE) and Deep Convolutional Generative Adversarial Network (DCGAN). The models were trained and tested on CIFAR 10 dataset and the example images were shown in Figure 2. A qualitative comparison was made between the two models.

Deep Learning Architecture:

Autoencoder

The autoencoder was created utilizing an input layer, 3 encoded layers, 3 decoded layers and an output layer. The input layer consisted of the input image size shape which was then fed to the first layer of the encoder. Each encoded layer utilized the previous and consisted of 3 Conv2d and 3 max pooling. The latent view would then pass off to the decoded layers which utilized Conv2d and upsampling 2d to produce a final output layer.

```
1 # input layer
2 input_layer = Input(shape=(258, 540,1))
3
4 # encoding architecture
5 encoded_layer1 = Conv2D(128, (3, 3), activation='relu', padding='same')(input_layer)
6 encoded_layer1 = MaxPool2D( (2, 2), padding= 'same')(encoded_layer1)
7 encoded_layer2 = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded_layer1)
8 encoded_layer2 = MaxPool2D( (2, 2), padding= 'same')(encoded_layer2)
9 encoded_layer3 = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded_layer2)
10 latent_view     = MaxPool2D( (2, 2), padding= 'same')(encoded_layer3)

1 # decoding architecture
2 decoded_layer1 = Conv2D(32, (3, 3), activation='relu', padding='same')(latent_view)
3 decoded_layer1 = UpSampling2D((2, 2))(decoded_layer1)
4 decoded_layer2 = Conv2D(64, (3, 3), activation='relu', padding='same')(decoded_layer1)
5 decoded_layer2 = UpSampling2D((2, 2))(decoded_layer2)
6 decoded_layer3 = Conv2D(128, (3, 3), activation='relu')(decoded_layer2)
7 decoded_layer3 = UpSampling2D((2, 2))(decoded_layer3)
8 output_layer   = Conv2D(1, (3, 3), padding='same')(decoded_layer3)

1 # compile the model
2 model = Model(input_layer, output_layer)
3 model.compile(optimizer='adam', loss='mse')
```

Figure 3: Autoencoder Model Python Syntax

Model: "model"

Layer (type)	Output Shape	Param #
<hr/>		
input_1 (InputLayer)	[(None, 260, 540, 3)]	0
conv2d (Conv2D)	(None, 260, 540, 128)	3584
max_pooling2d (MaxPooling2D)	(None, 130, 270, 128)	0
conv2d_1 (Conv2D)	(None, 130, 270, 64)	73792
max_pooling2d_1 (MaxPooling2D)	(None, 65, 135, 64)	0
conv2d_2 (Conv2D)	(None, 65, 135, 32)	18464
max_pooling2d_2 (MaxPooling2D)	(None, 33, 68, 32)	0
conv2d_3 (Conv2D)	(None, 33, 68, 32)	9248
up_sampling2d (UpSampling2D)	(None, 66, 136, 32)	0
conv2d_4 (Conv2D)	(None, 66, 136, 64)	18496
up_sampling2d_1 (UpSampling2D)	(None, 132, 272, 64)	0
conv2d_5 (Conv2D)	(None, 130, 270, 128)	73856
up_sampling2d_2 (UpSampling2D)	(None, 260, 540, 128)	0
conv2d_6 (Conv2D)	(None, 260, 540, 1)	1153
<hr/>		
Total params: 198,593		
Trainable params: 198,593		
Non-trainable params: 0		

Figure 4: Autoencoder Model Architecture

VAE

The variational autoencoder, a generative model was created in two parts similar to the ae. The decoder was created with the decoder taking the latent distribution sample as input. The original decoder layer outputted the total pixel size of the input image which was in this case $32 \times 32 \times 3 = 3072$.

Model: "model"

Layer (type)	Output Shape	Param #
input_decoder (InputLayer)	[(None, 2)]	0
intermediate_decoder (Dense)	(None, 512)	1536
original_decoder (Dense)	(None, 3072)	1575936
<hr/>		
Total params: 1,577,472		
Trainable params: 1,577,472		
Non-trainable params: 0		
<hr/>		

Figure 5: Decoder Model architecture

The vae was then compiled utilizing the previously created decoder model utilizing the input image size and y value. The model was then compiled utilizing RMSprop as the optimizer.

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 3072)]	0	
intermediate_encoder (Dense)	(None, 512)	1573376	input[0][0]
latent_encoder (Dense)	(None, 2)	1026	intermediate_encoder[0][0]
dense (Dense)	(None, 2)	6	latent_encoder[0][0]
dense_1 (Dense)	(None, 2)	6	latent_encoder[0][0]
lambda (Lambda)	(None, 2)	0	dense[0][0] dense_1[0][0]
model (Functional)	(None, 3072)	1577472	lambda[0][0]
custom_variational_layer (Custo	(None, 3072)	0	input[0][0] model[0][0]
<hr/>			
Total params: 3,151,886			
Trainable params: 3,151,886			
Non-trainable params: 0			
<hr/>			

Figure 6: VAE Model architecture

DCGAN

The DCGAN, another generative model, utilized a discriminator and generator. The job of the generator is to spawn ‘fake’ images that look like the training images. The job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator.

```
def build_generator(self):

    model = Sequential()

    model.add(Dense(256, input_dim=self.latent_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(np.prod(self.img_shape), activation='tanh'))
    model.add(Reshape(self.img_shape))

    model.summary()

    noise = Input(shape=(self.latent_dim,))
    img = model(noise)

    return Model(noise, img)

def build_discriminator(self):

    model = Sequential()

    model.add(Flatten(input_shape=self.img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()

    img = Input(shape=self.img_shape)
    validity = model(img)
    return Model(img, validity)
```

Figure 7: DCGAN Generator and Discriminator Model Syntax

```
... Model: "sequential"
```

Layer (type)	Output Shape	Param #
<hr/>		
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 512)	1573376
leaky_re_lu (LeakyReLU)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
dense_2 (Dense)	(None, 1)	257
<hr/>		
Total params: 1,704,961		
Trainable params: 1,704,961		
Non-trainable params: 0		

Figure 8: DCGAN Discriminator Model

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
<hr/>		
dense_3 (Dense)	(None, 256)	25856
leaky_re_lu_2 (LeakyReLU)	(None, 256)	0
batch_normalization (BatchNo	(None, 256)	1024
dense_4 (Dense)	(None, 512)	131584
leaky_re_lu_3 (LeakyReLU)	(None, 512)	0
batch_normalization_1 (Batch	(None, 512)	2048
dense_5 (Dense)	(None, 1024)	525312
leaky_re_lu_4 (LeakyReLU)	(None, 1024)	0
batch_normalization_2 (Batch	(None, 1024)	4096
dense_6 (Dense)	(None, 3072)	3148800
reshape (Reshape)	(None, 32, 32, 3)	0
<hr/>		
Total params: 3,838,720		
Trainable params: 3,835,136		
Non-trainable params: 3,584		

Figure 9: DCGAN Generator Model

Results + Discussion & Comparison:

Autoencoder

The autoencoder did not initially perform that well due to several factors. Initially the incorrect image size was being extrapolated giving outputting some bad training values. This issue was addressed by extrapolating a 250,540 image and epochs were progressively increased. High RAM usage was addressed by lowering the batch size.

```
Epoch 1/25
9/9 [=====] - 52s 554ms/step - loss: 0.1712 - val_loss: 0.0779
Epoch 2/25
9/9 [=====] - 4s 485ms/step - loss: 0.0819 - val_loss: 0.0870
Epoch 3/25
9/9 [=====] - 4s 488ms/step - loss: 0.0682 - val_loss: 0.0756
Epoch 4/25
9/9 [=====] - 4s 490ms/step - loss: 0.0612 - val_loss: 0.0794
Epoch 5/25
9/9 [=====] - 4s 495ms/step - loss: 0.0583 - val_loss: 0.0863
Epoch 6/25
9/9 [=====] - 4s 511ms/step - loss: 0.0571 - val_loss: 0.0809
Epoch 7/25
9/9 [=====] - 4s 501ms/step - loss: 0.0556 - val_loss: 0.0818
Epoch 8/25
9/9 [=====] - 4s 514ms/step - loss: 0.0541 - val_loss: 0.0812
Epoch 9/25
9/9 [=====] - 5s 520ms/step - loss: 0.0552 - val_loss: 0.0890
Epoch 10/25
9/9 [=====] - 4s 516ms/step - loss: 0.0555 - val_loss: 0.0793
Epoch 11/25
9/9 [=====] - 4s 502ms/step - loss: 0.0542 - val_loss: 0.0802
Epoch 12/25
9/9 [=====] - 4s 500ms/step - loss: 0.0527 - val_loss: 0.0837
Epoch 13/25
9/9 [=====] - 4s 495ms/step - loss: 0.0525 - val_loss: 0.0805
```

Figure 10: Autoencoder Training Output 25 epochs

After 25 epochs, the loss had only decreased to .05 with validation loss being around .079. I determined this was the case compared to the lecture examples due to the images being much larger so I had to increase the number of epochs.

There exist several methods to design forms to be filled in. For instance, fields may be surrounded by boxes, by light rectangles or by guiding lines. Some specify where to write and, therefore, n of skew and overlapping with other parts of guides can be located on a separate sheet located below the form or they can be printed on the form. The use of guides on a separate sheet from the point of view of the quality of the form, but requires giving more instructions and restricts its use to tasks where this type of a

There are several classic spatial filters for eliminating high frequency noise from images: the median filter and the closing opening filter used. The mean filter is a lowpass or smoother that replaces the pixel values with the neighborhood; it reduces the image noise but blurs the image edges. The filter calculates the median of the pixel neighborhood, thereby reducing the blurring effect. Finally, the closing filter is a mathematical morphological operation that combines the same number of erosion and dilation operations in order to eliminate small objects.

There are several classic spatial filters for reducing frequency noise from images. The mean filter, the closing operator filter, are frequency filters. The closing operator filter is frequently used. This filter reduces the noise but blurs the image edges. It reduces the image noise but blurs the image edges, calculates the median of the pixel neighbourhood for each pixel. Finally, the median filter does not need morphological operations in order to eliminate small granular noise.

There exist several methods to design forms which can be filled in. For instance, fields may be surrounded by boxes, by light rectangles or by guiding lines. Some methods specify where to write and, therefore, the form is not skewed and overlapping with other parts of the form. Guides can be located on a separate sheet and the form can be located below the form or they can be printed directly on the form. The use of guides on a separate sheet is not very good from the point of view of the quality of the form, but it requires giving more instructions and, therefore, restricts its use to tasks where this type of

There are several classic spatial filters for reducing frequency noise from images. The mean filter, the closing opening filter are frequently used. The mean convolution filter is the most common of all. It removes the noise, but blurs the image; it calculates the median of the pixel neighborhood for eliminating the *Murphy effect*. Finally, the opening closing morphological operations are used for removing noise.

```
1 preds = model.predict(x_test[n:n+5])
2 f, ax = plt.subplots(1,5)
3 f.set_size_inches(258, 540)
4 for i, a in enumerate(range(n,n+5)):
5     ax[i].imshow(preds[i].reshape(258, 540))
6 plt.show()
```

*These would provide more or enough
for all needs. For instance, there may be
big themes, like Right relationships or the qualities
of love, that are repeated at several points. There
are also some good principles which certain parts
of the story can repeat again and again.
This way, the reader can remember
them better than those on their first
reading. The use of repetition is a valuable
tool that will help all the qualities of a
good story along more effectively and
impressively on the reader's mind.*

There are several elements of linear planning. High frequency noise, for example, the medium filter and the shortening command, are used to reduce the number of points. The point filter is a measure of the distance between the points; however, the number of points does not have to be the same as the number of pixels. Filter evaluates the number of the pixel points, then by reducing the filtering effect, changes. Filter is a mathematical transformation, which can increase the number of elements and decrease the number of points to eliminate point noise.

These are serious charges. We believe the Board has done its best to meet them. In particular, the Board appears to have been successful in advancing the human rights of the people of South Africa, and in the process of doing so, has demonstrated that it can be a force for good, and that the principles of racial equality and non-discrimination are not confined to the United States.

There were several incidents we observed but without proof. For instance, Police made the following remark: "The light transmission of the passing auto apparently controls the number plate, therefore, all drivers must understand that if a colored person passes near the inspection bay or in inspection, they must stop their car until the light has turned green." The color of plates are in sequential order. This is the reason why the light of each auto that passed us would be the specific color of each auto. Inspection drivers never make mistakes, because they are the ones who handle vehicles. Since those auto

There are many other ways to increase the value of your business. One way is to increase the number of customers you have. Another way is to increase the number of sales you make per customer. Still another way is to increase the average price per sale.

Figure 11: Autoencoder Prediction Output 25 epochs



There are several classic spatial filters for reducing frequency noise from images. The mean filter, the closing opening filter are frequently used. The mean smoothing filter that replaces the pixel values with the It reduces the image noise but blurs the image edges calculates the median of the pixel neighbourhood for eaching the blurring effect. Finally, the opening, closing filter morphological filter that combines the same number of morphological operations in order to eliminate small structures.

Figure 12: Autoencoder Pred Image (L), Autoencoder Pred Image (R),

_____ The prediction output was nowhere near as clean as I would have liked. The image was left blurry while the only main improvement was creases were less visible and the coffee stain had been lightened.

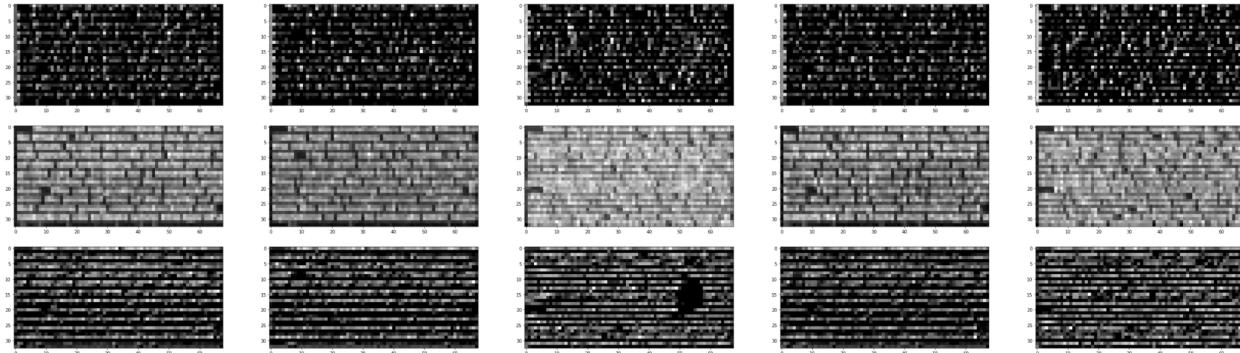


Figure 13: Autoencoder Quality Output 25 epochs

_____ I could tell the model was on the right track as it was identifying some lines of text and in particular noticed the coffee stain. This gave me the confidence to progressively tweak the model.

```

Epoch 91/100
9/9 [=====] - 4s 501ms/step - loss: 0.0283 - val_loss: 0.0279
Epoch 92/100
9/9 [=====] - 5s 519ms/step - loss: 0.0280 - val_loss: 0.0278
Epoch 93/100
9/9 [=====] - 4s 504ms/step - loss: 0.0279 - val_loss: 0.0276
Epoch 94/100
9/9 [=====] - 4s 516ms/step - loss: 0.0274 - val_loss: 0.0279
Epoch 95/100
9/9 [=====] - 4s 503ms/step - loss: 0.0275 - val_loss: 0.0275
Epoch 96/100
9/9 [=====] - 4s 507ms/step - loss: 0.0271 - val_loss: 0.0275
Epoch 97/100
9/9 [=====] - 4s 504ms/step - loss: 0.0272 - val_loss: 0.0272
Epoch 98/100
9/9 [=====] - 4s 505ms/step - loss: 0.0277 - val_loss: 0.0272
Epoch 99/100
9/9 [=====] - 4s 502ms/step - loss: 0.0266 - val_loss: 0.0271
Epoch 100/100
9/9 [=====] - 4s 515ms/step - loss: 0.0272 - val_loss: 0.0269

```

Figure 14: Autoencoder Training Output 100 epochs

Upon increasing the number of epochs I immediately noticed an immediate improvement in loss which dipped all the way down to .027. This indicated to me my hunch about image size requiring a larger run was correct.



Figure 15: Autoencoder Prediction Output 25 epochs

Upon increasing the number of epochs I immediately noticed an improvement in predicted image quality. The coffee stains were close to being completely removed in some places while the wrinkles completely gone from other images.

There exist several methods to fill in fields to be filled in. For instance, by surrounding by bounding boxes, or by guiding rulers. These methods to write and, therefore, minimize and overlapping with other parts of guides can be located on a separate sheet that is located below the form or directly on the form. The use of sheet is much better from the point quality of the scanned image, but

There exist several methods to fill in fields to be filled in. For instance, by surrounding by bounding boxes, or by guiding rulers. These methods to write and, therefore, minimize and overlapping with other parts of guides can be located on a separate sheet that is located below the form or directly on the form. The use of sheet is much better from the point quality of the scanned image, but

Figure 16: Autoencoder Pred Image (L), Autoencoder Pred Image (R),

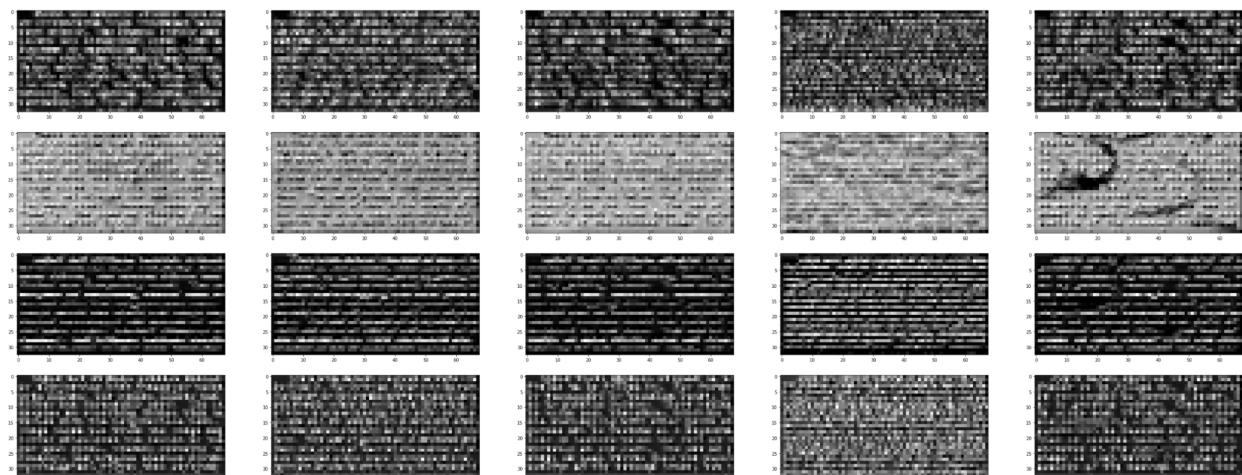


Figure 17: Autoencoder Quality Output 100 epochs

In terms of image quality metrics, the model was able to distinguish characters significantly better while also identifying creases, spills and other blemishes.

VAE

The VAE model did not perform that well similarly to the example given in lecture. Whereas the lecture model outputted a loss of .49, my model outputted a loss of 0.69. This was due to my implementation utilizing the cifar 10 dataset which utilizes three color channels. This makes the images more difficult to run through the model. The generative output at the end was also lacking in quality but was nonetheless generated properly.

```
Epoch 10/20
9375/9375 [=====] - 46s 5ms/step - loss: 0.6915 - val_loss: 0.6919
Epoch 11/20
9375/9375 [=====] - 49s 5ms/step - loss: 0.6916 - val_loss: 0.6919
Epoch 12/20
9375/9375 [=====] - 49s 5ms/step - loss: 0.6916 - val_loss: 0.6919
Epoch 13/20
9375/9375 [=====] - 48s 5ms/step - loss: 0.6915 - val_loss: 0.6918
Epoch 14/20
9375/9375 [=====] - 48s 5ms/step - loss: 0.6916 - val_loss: 0.6918
Epoch 15/20
9375/9375 [=====] - 46s 5ms/step - loss: 0.6915 - val_loss: 0.6919
Epoch 16/20
9375/9375 [=====] - 46s 5ms/step - loss: 0.6915 - val_loss: 0.6919
Epoch 17/20
9375/9375 [=====] - 47s 5ms/step - loss: 0.6915 - val_loss: 0.6919
Epoch 18/20
9375/9375 [=====] - 49s 5ms/step - loss: 0.6916 - val_loss: 0.6919
Epoch 19/20
9375/9375 [=====] - 49s 5ms/step - loss: 0.6916 - val_loss: 0.6919
Epoch 20/20
9375/9375 [=====] - 49s 5ms/step - loss: 0.6915 - val_loss: 0.6919
<keras.callbacks.History at 0x7f8ba04b91d0>
```

Figure 18: VAE training output

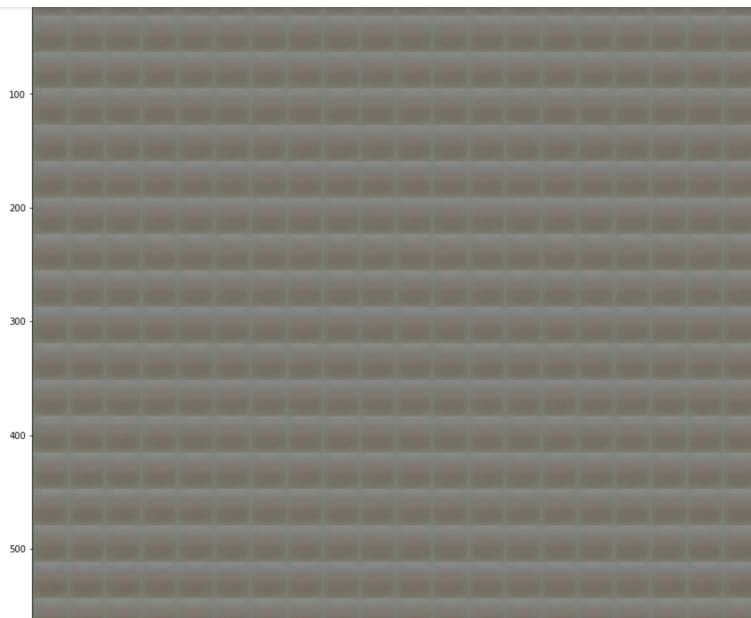


Figure 19: VAE Generative Output

DCGAN

_____ In comparison to VAE, the DCGAN yielded some rather solid results. Initially the generator had the discriminator fooled. With an accuracy of 50% this indicates the discriminator is only right half the time. As the epochs progressed and the generator images became more realistic, the Discriminator loss decreased, while the accuracy and generator loss increased. This indicated the DCGAN was functioning properly!

```
786 [D loss: 0.725293, acc.: 42.19%] [G loss: 0.802963]
787 [D loss: 0.719020, acc.: 39.06%] [G loss: 0.806783]
788 [D loss: 0.726040, acc.: 37.50%] [G loss: 0.804946]
789 [D loss: 0.701920, acc.: 50.00%] [G loss: 0.827112]
790 [D loss: 0.736028, acc.: 37.50%] [G loss: 0.796697]
791 [D loss: 0.747874, acc.: 32.81%] [G loss: 0.778332]
792 [D loss: 0.727529, acc.: 40.62%] [G loss: 0.804175]
793 [D loss: 0.706207, acc.: 48.44%] [G loss: 0.825914]
794 [D loss: 0.737578, acc.: 50.00%] [G loss: 0.832175]
795 [D loss: 0.724798, acc.: 46.88%] [G loss: 0.817256]
796 [D loss: 0.733033, acc.: 42.19%] [G loss: 0.821423]
797 [D loss: 0.712584, acc.: 51.56%] [G loss: 0.805397]
798 [D loss: 0.723849, acc.: 50.00%] [G loss: 0.791525]
799 [D loss: 0.724990, acc.: 45.31%] [G loss: 0.781141]
800 [D loss: 0.709014, acc.: 54.69%] [G loss: 0.772218]
```

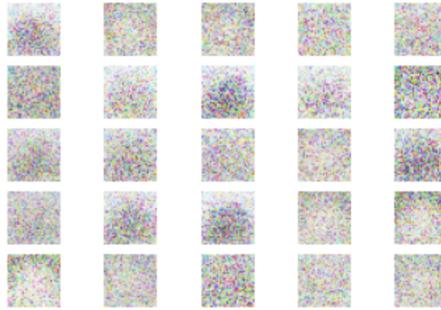


Figure 20: DCGAN Training Output and Generated Images 800 epochs

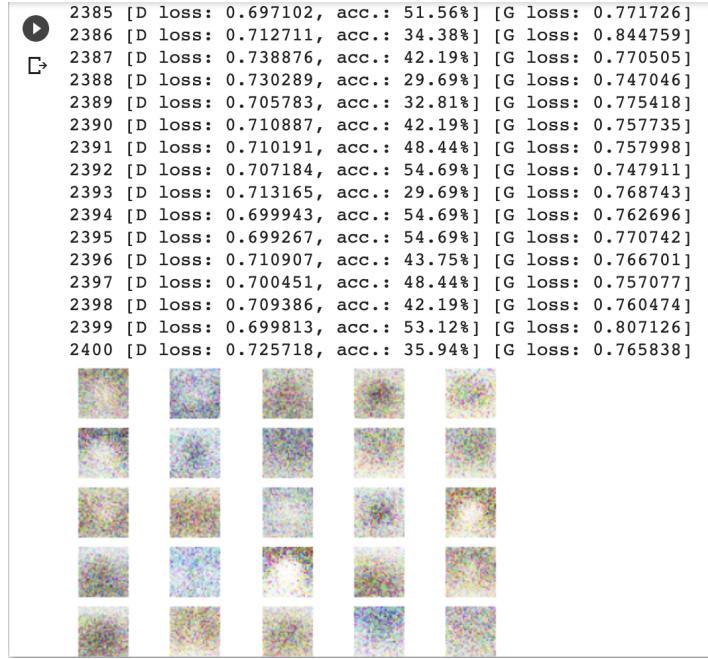


Figure 21: DCGAN Training Output and Generated Images 2400 epochs

```

9188 [D loss: 0.303323, acc.: 92.19%] [G loss: 2.704818]
9189 [D loss: 0.487477, acc.: 75.00%] [G loss: 1.681684]
9190 [D loss: 0.299526, acc.: 90.62%] [G loss: 2.741573]
9191 [D loss: 0.488861, acc.: 75.00%] [G loss: 2.628856]
9192 [D loss: 0.336703, acc.: 85.94%] [G loss: 2.931231]
9193 [D loss: 0.417487, acc.: 76.56%] [G loss: 1.667081]
9194 [D loss: 0.415616, acc.: 82.81%] [G loss: 1.815463]
9195 [D loss: 0.371422, acc.: 84.38%] [G loss: 2.357869]
9196 [D loss: 0.363875, acc.: 90.62%] [G loss: 2.928938]
9197 [D loss: 0.509812, acc.: 75.00%] [G loss: 1.586779]
9198 [D loss: 0.499059, acc.: 76.56%] [G loss: 1.821058]
9199 [D loss: 0.469546, acc.: 76.56%] [G loss: 1.738559]
9200 [D loss: 0.383822, acc.: 90.62%] [G loss: 2.150250]

```

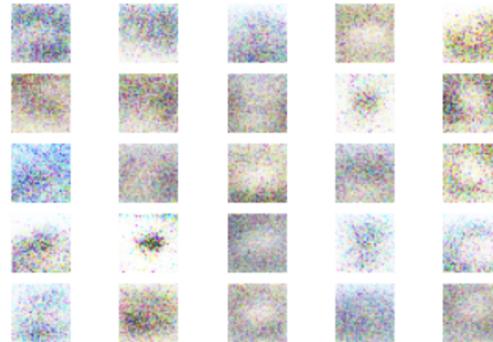


Figure 22: DCGAN Training Output and Generated Images 9200 epochs

Conclusion:

Homework 4 was another astounding success with images being mostly denoised and the generative DCGAN and VAE models being properly run, compiled and compared.