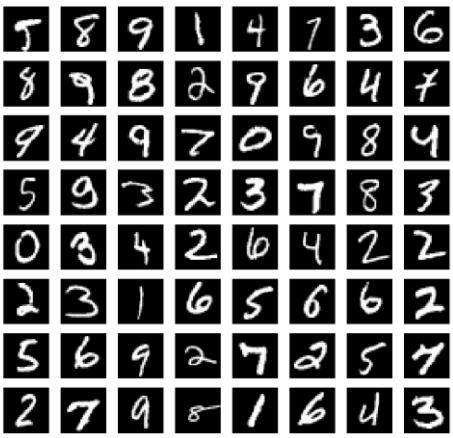
Taha Sukru Karabacakoglu
16-125-692

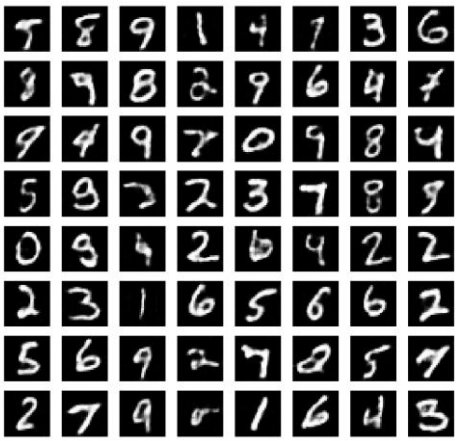
Denoising Autoencoder

In this part, we implemented 3 different noise types to an autoencoder. We chose epoch size as 10. Our observations and results are like as follows:

No Noise

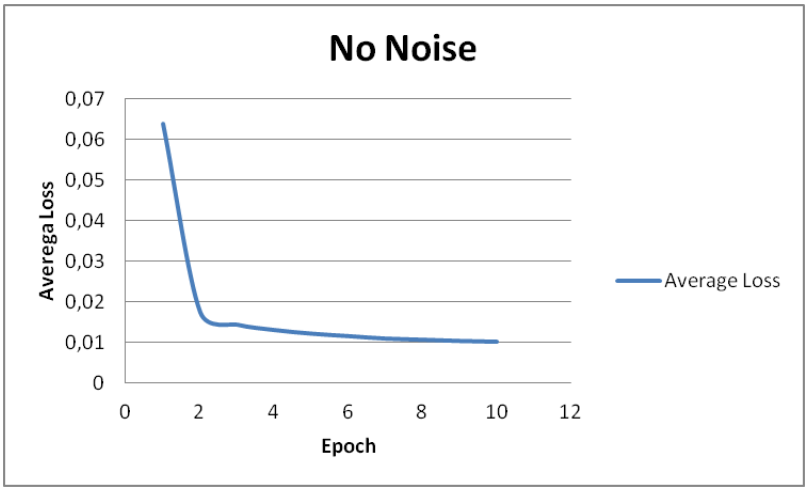
In this part, we analyzed results for standard autoencoder without noise. Inputs are shown on the left image, and outputs are shown on the right image.





We observed following losses:

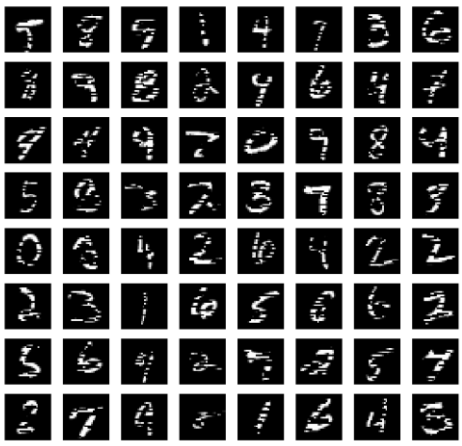
No Noise	
Epoch	Average Loss
1	0,0638
2	0,0176
3	0,0145
4	0,0132
5	0,0123
6	0,0117
7	0,0111
8	0,0108
9	0,0105
10	0,0103

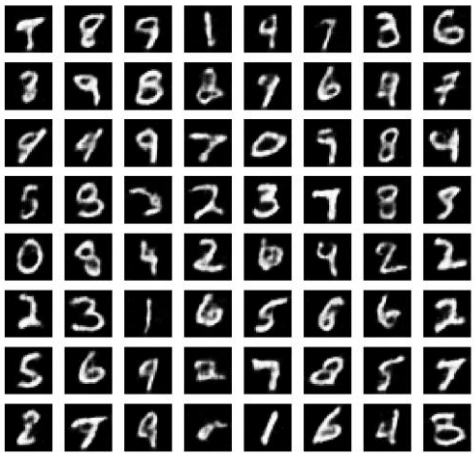


As expected, we achieved the lowest loss without noise.

Random Masking

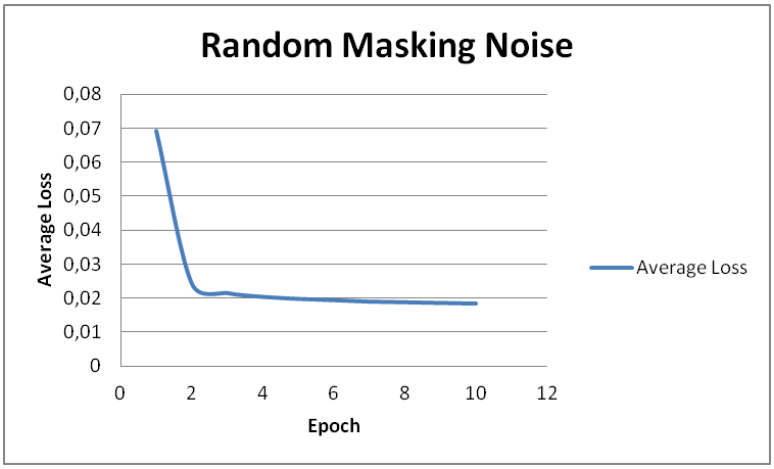
In this part, we analyzed results for denoising autoencoder with Random Masking noise. Inputs are shown on the left image, and outputs are shown on the right image.





We observed following losses:

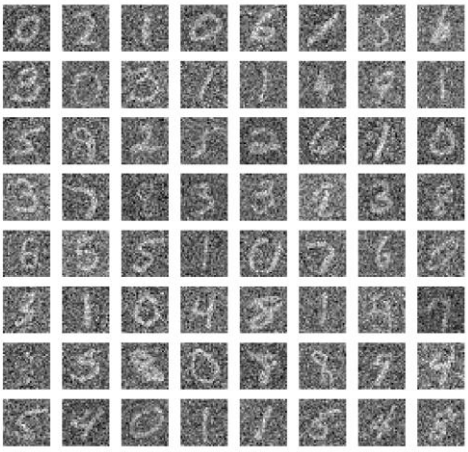
Random Masking Noise	
Epoch	Average Loss
1	0,0693
2	0,0243
3	0,0215
4	0,0204
5	0,0198
6	0,0194
7	0,0190
8	0,0188
9	0,0186
10	0,0184

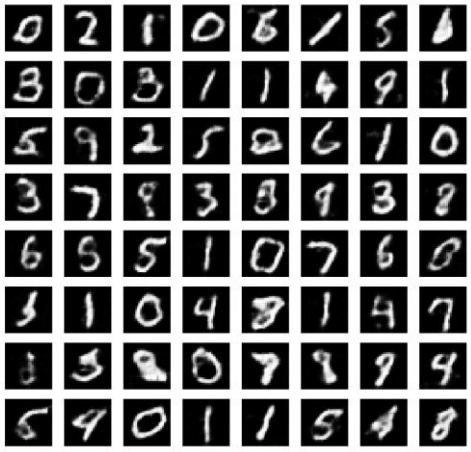


This one has losses greater than without noise. That's because in this type of noise, we are removing some parts of the image. In another words, we are setting some pixel values to zero. We are not changing all pixels, also we are not adding extreme values. Therefore, this one is the easiest noise to handle.

Gaussian Noise

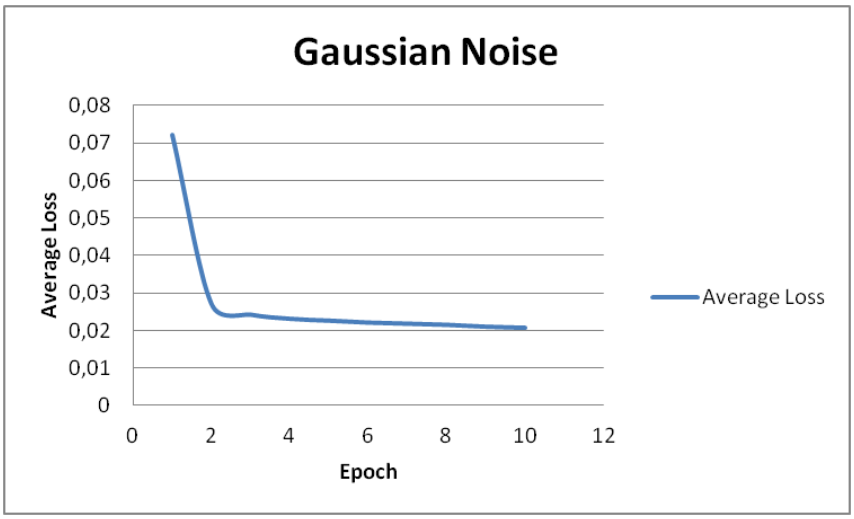
In this part, we analyzed results for denoising autoencoder with Gaussian noise. Inputs are shown on the left image, and outputs are shown on the right image.





We observed following losses:

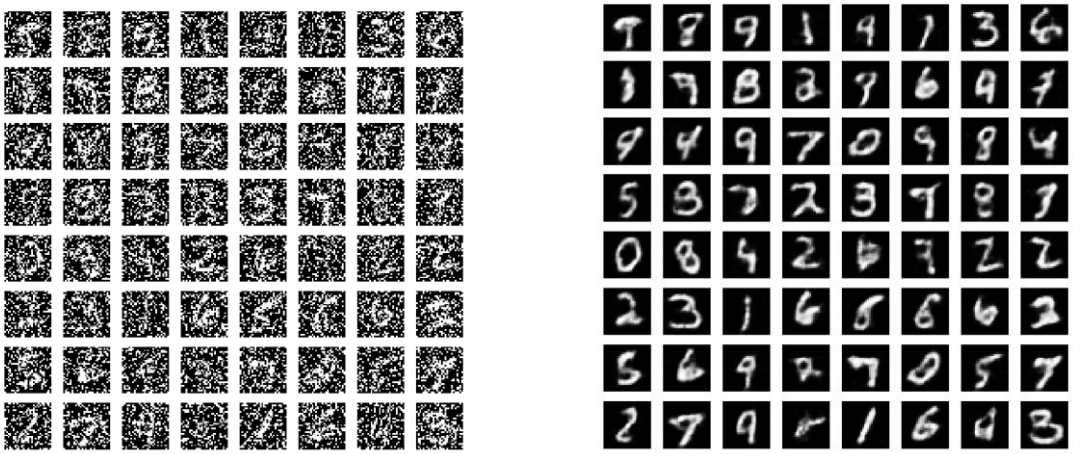
Gaussian Noise	
Epoch	Average Loss
1	0,0721
2	0,0271
3	0,0243
4	0,0232
5	0,0227
6	0,0222
7	0,0219
8	0,0216
9	0,0211
10	0,0208



This one has losses greater than Random Masking noise. That's because in this type of noise, we are adding some values to all pixels with some deviation. However, we are not dealing with extreme values, this is the second hardest noise to overcome.

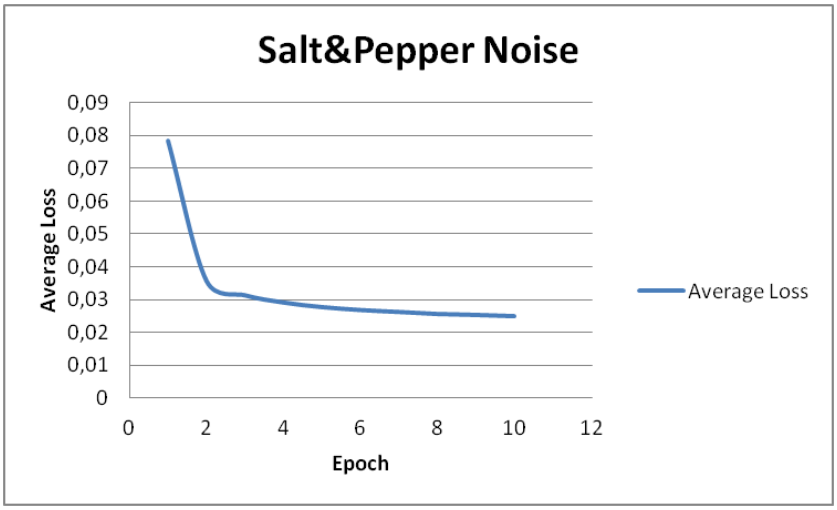
Salt&Pepper Noise

In this part, we analyzed results for denoising autoencoder with Salt&Pepper noise. Inputs are shown on the left image, and outputs are shown on the right image.



We observed following losses:

Salt&Pepper Noise	
Epoch	Average Loss
1	0,0785
2	0,0356
3	0,0312
4	0,0290
5	0,0276
6	0,0267
7	0,0261
8	0,0255
9	0,0252
10	0,0248



This one has the highest losses. That's because in this type of noise, we are adding some extereme values and removing some values from pixels. Therefore, this is the hardest noise to overcome.

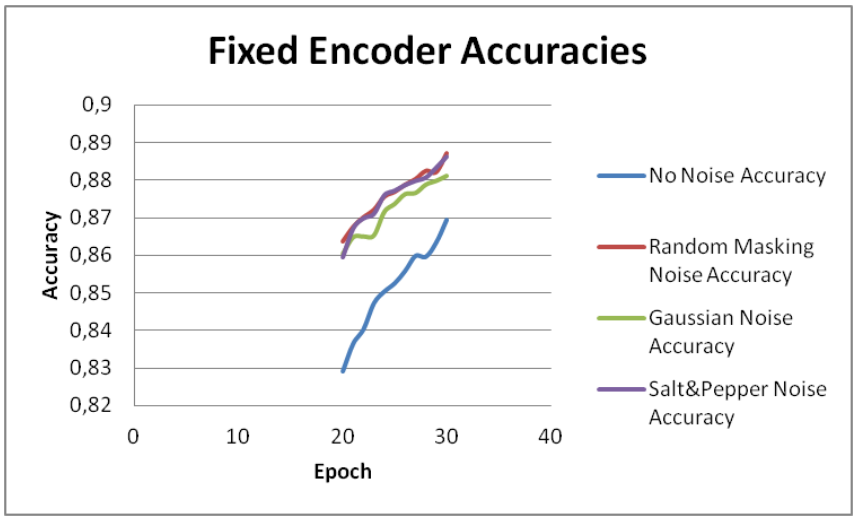
Transfer Learning

In this part, we are using our trained denoising autoencoder's encoder and trying to classify images. We chose epoch size as 30. In terms of simplicity, we mapped last 10 epochs. Our observations and results are like as follows:

Fixed Encoder Features

With fixed encoder, we exclude encoder from optimization. So, by that, only weights in linear classifier change after every epoch.

Epoch	No Noise Accuracy	Random Masking Noise Accuracy	Gaussian Noise Accuracy	Salt&Pepper Noise Accuracy
20	0,8291	0,8636	0,8605	0,8596
21	0,8365	0,8675	0,8649	0,8672
22	0,8404	0,8701	0,8651	0,8699
23	0,8473	0,8721	0,8654	0,8711
24	0,8504	0,8756	0,8716	0,8760
25	0,8526	0,8769	0,8737	0,8772
26	0,8559	0,8788	0,8763	0,8787
27	0,8599	0,8803	0,8766	0,8798
28	0,8597	0,8825	0,8788	0,8807
29	0,8635	0,8822	0,8798	0,8833
30	0,8695	0,8872	0,8811	0,8862

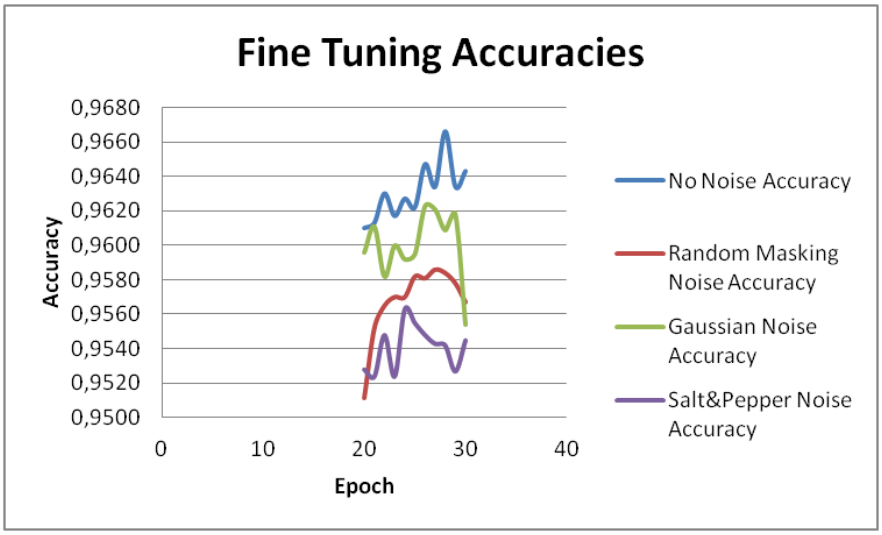


According to our results, adding noise to samples is increasing accuracy. Random masking and Salt&Pepper are so close to each other; however, random masking is slightly better than Salt&Pepper noise. We think that's because in Random masking, it's trying to complete missing parts of the pictures, but in Salt&Pepper noise adding extreme values are causing an increase in the loss. Features with noises are better than standard autoencoder.

Finetuning

With finetuning, we add encoder into optimizations. So, by that, we change weights in encoder and linear classifier after every epoch.

Epoch	No Noise Accuracy	Random Masking Noise Accuracy	Gaussian Noise Accuracy	Salt&Pepper Noise Accuracy
20	0,9610	0,9511	0,9596	0,9528
21	0,9613	0,9552	0,9611	0,9524
22	0,9630	0,9565	0,9582	0,9548
23	0,9617	0,9570	0,9600	0,9524
24	0,9627	0,9570	0,9592	0,9563
25	0,9622	0,9582	0,9595	0,9555
26	0,9647	0,9581	0,9623	0,9548
27	0,9634	0,9586	0,9621	0,9543
28	0,9666	0,9584	0,9609	0,9542
29	0,9634	0,9578	0,9618	0,9527
30	0,9643	0,9567	0,9554	0,9545

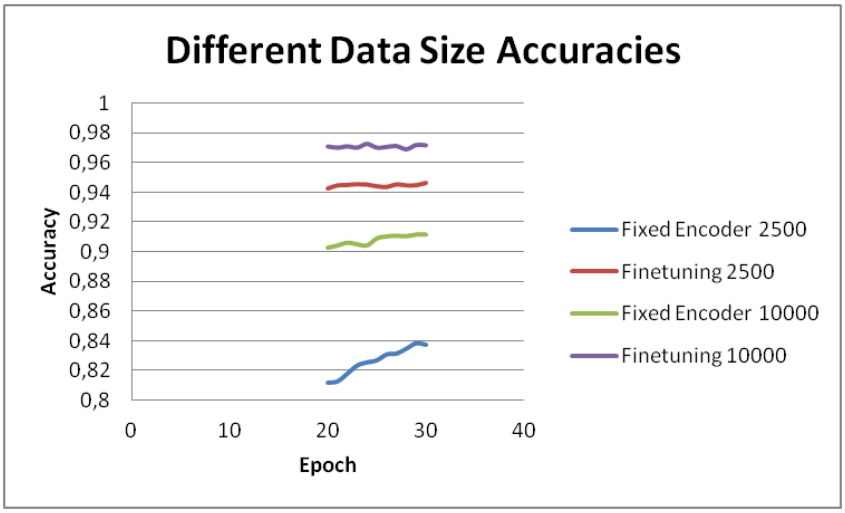


According to our results, in finetuning, adding noise to samples is decreasing accuracy. No noise encoder is having best results. With finetuning, we are also trying to adapt encoder to samples, and samples having noises increases loss. However, we are getting better accuracies by finetuning compared to fixed encoder.

Effect of Data size

In this part, we used random masking encoder to observe the effect of data size. We executed this on datasizes 2500 and 10000.

Epoch	Fixed Encoder 2500	Finetuning 2500	Fixed Encoder 10000	Finetuning 10000
20	0,8116	0,9423	0,9028	0,9710
21	0,8125	0,9444	0,9043	0,9702
22	0,8177	0,9447	0,9062	0,9711
23	0,8231	0,9452	0,9051	0,9702
24	0,8252	0,9449	0,9044	0,9728
25	0,8266	0,9438	0,9091	0,9702
26	0,8305	0,9434	0,9105	0,9707
27	0,8311	0,9450	0,9109	0,9713
28	0,8344	0,9444	0,9106	0,9691
29	0,8380	0,9445	0,9117	0,9720
30	0,8370	0,9460	0,9117	0,9718



As the data size increases, also accuracies increase. This is an expected behavior because we are increasing training data size. Finetuning still performs better than fixed encoder features.