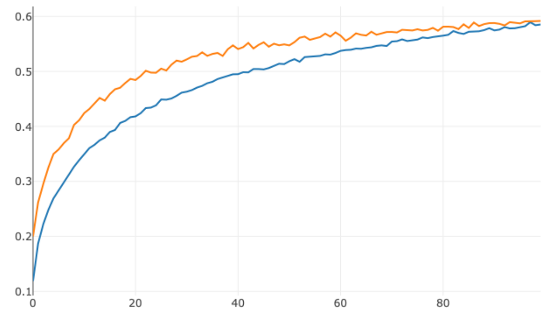
Machine Learning Assignment

Fatema Alkhanaizi

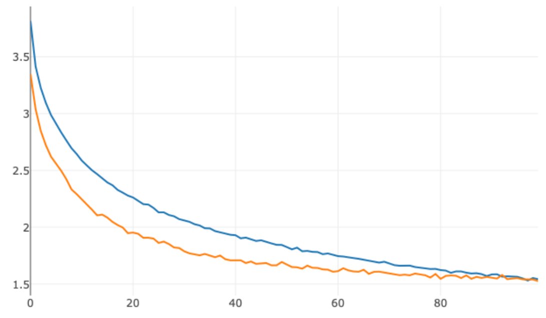
March 22, 2019

Building a Classifier

The plots in Figure 1 are for the best CNN architecture implmeneted for the classifier. The best accuracy for the test data was 59%. Figure 2 show the architecture of this network. A key feature for this classifier was the inclusion of randomnization for the training dataset. The data was randomnly rotated, cropped and flipped, this had significally improved the results as shown in Figure 3. It is clear from the plots that without randomnising the dataset the test accuracy quickly converges although the training accuracy is improving and reaching a value close to 98%. With randomnisation the test accuracy was continuesly improving along with the training accuracy although at a much slower pace.



(a) accuracy plot



(b) loss plot

Figure 1: Best CNN results

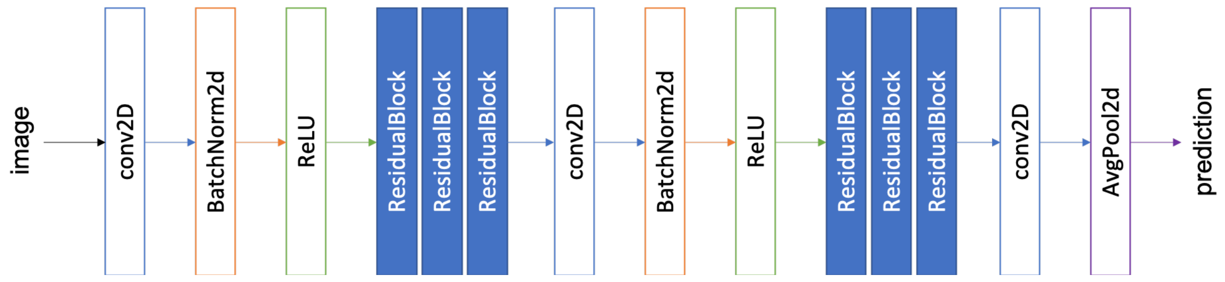
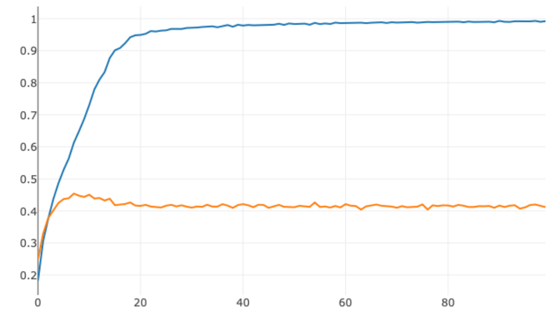
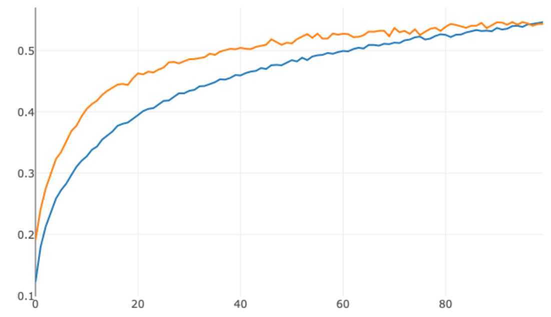


Figure 2: Best CNN architecture



(a) accuracy plot for non-randomnized training dataset

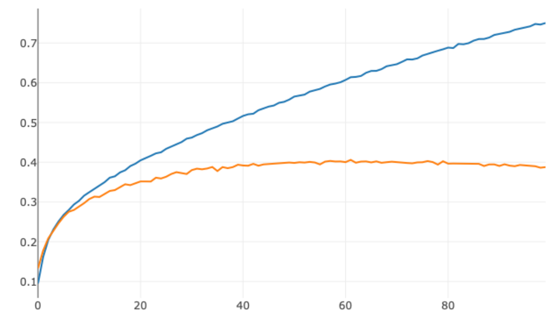


(b) accuracy plot for randomnized training dataset

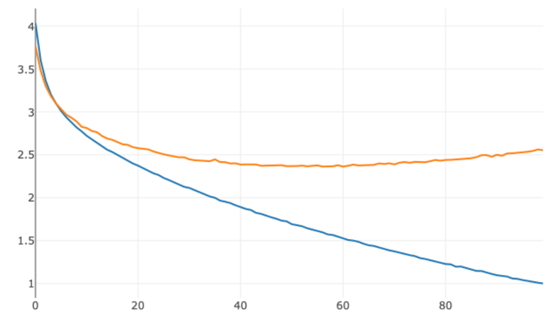
Figure 3: The network accuracy with and without randomnising the training dataset

Other Architectures Tested

As first steps, the CNN discussed in this module lectures was tested. The code of this CNN was taken from the provided material and modified accordingly to fit the purpose of this assignment. The first modification included was adding rescalling and normalizing both training and testing datasets. The batch size was increased to 64 as well. The number of epochs was also increased to 100. The results for those modifications included in Figure 4. The accuracy obtained for this CNN was 39%. The architecture of this network is shown in Figure 5.



(a) accuracy plot



(b) loss plot

Figure 4: CNN with 3 convolutional network layers, training dataset was not randomnized

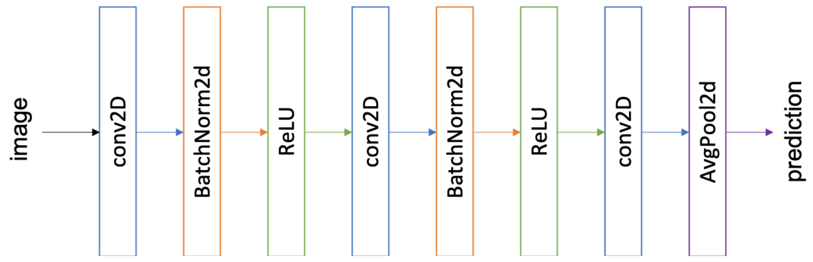
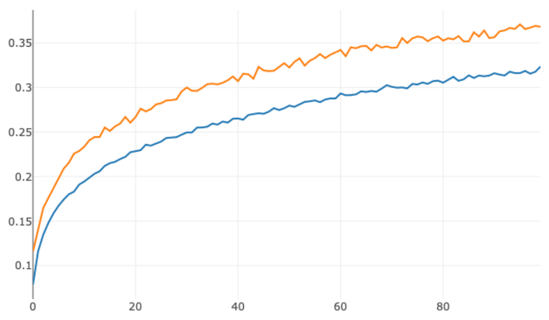
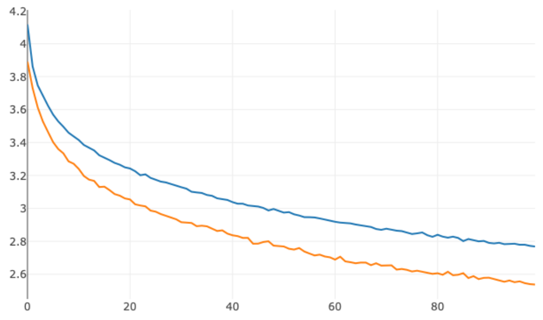


Figure 5: Basic CNN architecture

After that randomnisation was included to the training dataset; Figure 6. The resulted test accuracy, 37% was not better than the previous test accuracy, however the convergance of the accuracy was significally slowed and thus with more complex network architecture and increased epochs this setup could provide better results than the previous setup.



(a) accuracy plot



(b) loss plot

Figure 6: CNN with 3 convolutional network layers, training dataset was randomnized

As randomnisation of the training dataset proved to be promising however this came at the cost of an increased training time; from around 30 minutes to 1 hour. The CNN architecture was improved next. Residual blocks were added to the network; Figure 7. Figure 8 shows the accuracy and loss plots for this network. The test accuracy reached 54% a big improvment from the previous network.

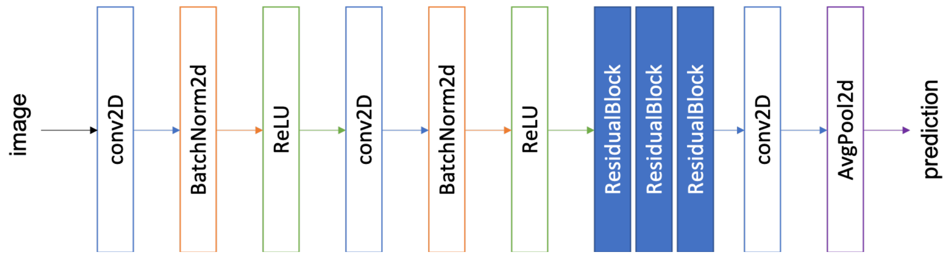
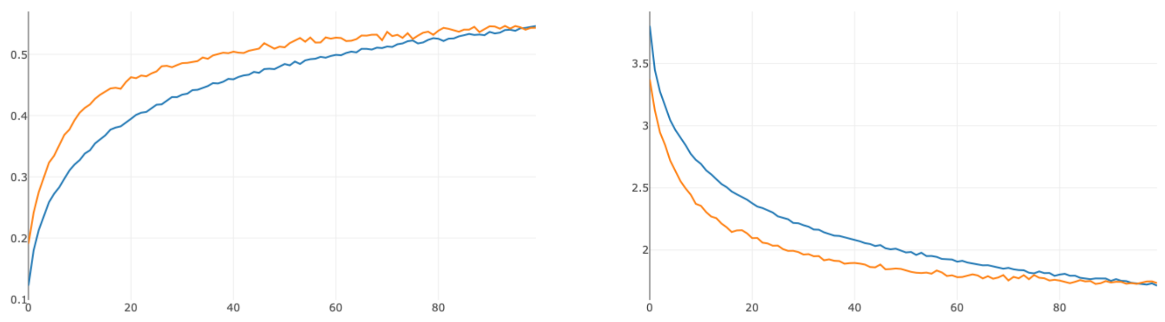


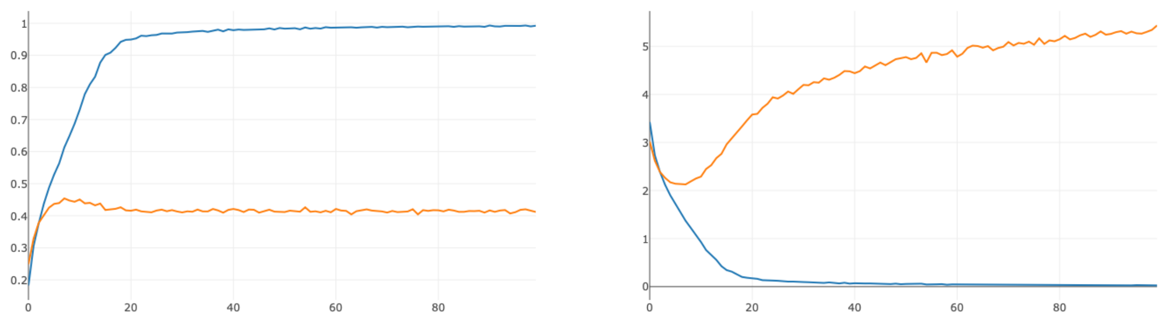
Figure 7: CNN with Residual Blocks architecture



(a) accuracy plot (b) loss plot

Figure 8: CNN with 3 Residual Blocks included, training dataset was randomnized

As a contrast, for non-randomised dataset Figure 9 shows the accuracy and loss plots. The test accuracy was 41% and the loss for the test dataset seemed to be getting worst with each epoch.



(a) accuracy plot (b) loss plot

Figure 9: CNN with 3 Residual Blocks included, training dataset was not randomnized

The following architectures were tested next:

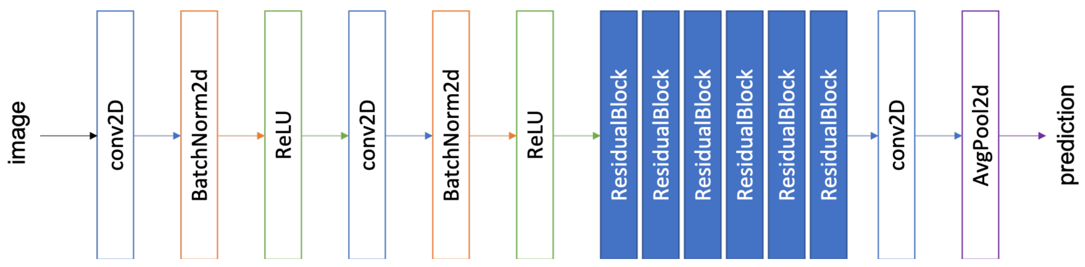


Figure 10: CNN with 6 consequetive Residual Blocks architecture

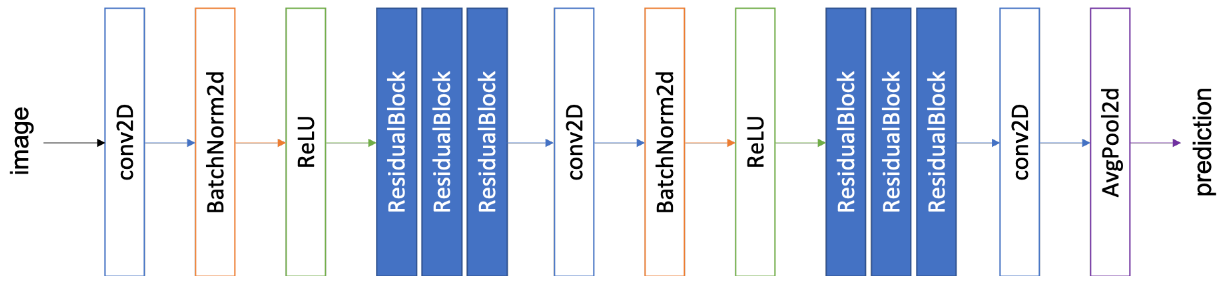
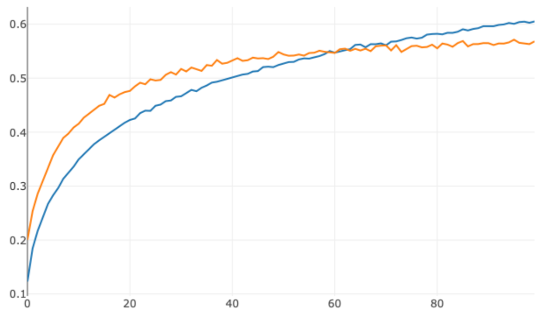
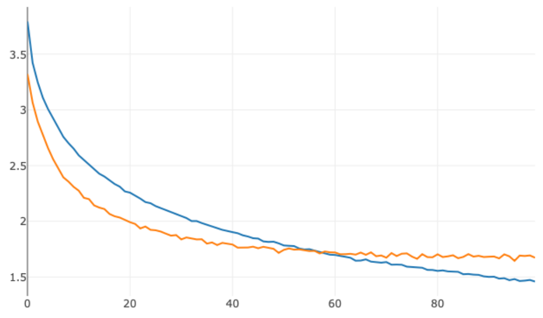


Figure 11: CNN with 3 consequetive Residual Blocks between the convolutional layers architecture

The architecture in Figure 10 resulted in a slightly improved test accuracy 56%, Figure 12. Also, the test accuracy converged at a much earlier epoch, around epoch 60, than the previous network.



(a) accuracy plot



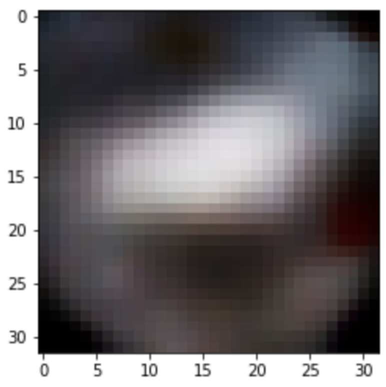
(b) loss plot

Figure 12: CNN with 6 Residual Blocks included, training dataset was randomnized

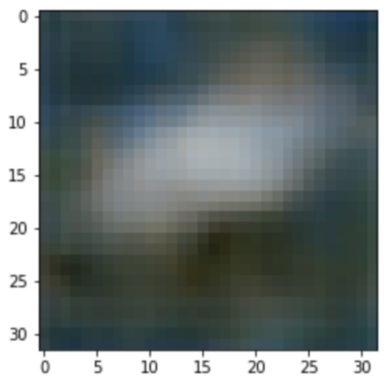
On the other hand for the architecture in Figure 11 which is the best architecture for this task, the test accuracy improved to 59%, Figure 1. Unlike the previous architecture the test accuracy did not converge around epoch 60 and kept improving along with the training accuracy. The parameters used of those networks were tested as well, however changing those parameters did not result in a much better performing classifier. The batch size was also modified however a smaller batch size did not improve the learning and an increase batch size did not provide a greater improvment.

Generating a Pegasus

For the intial steps for this task intial codes from the lectures materials were used, the simple Autocoder was trained after modifing the datsets to be rescalled and normalized and similarly to VAE model. Figure 13 shows the outputs for generating a pegasus with those two models.



(a) Basic Autocoder model



(b) VAE model

Figure 13: Generated Pegasus images

The VAE model showed much better results relatively to the basic autoencoder. The outline looks closer to a horse than the blob with the autoencoder image. For the next steps, the following architecture was used which is based on a paper by Larsen et al. [1], Figure 14. The idea behind this model is combine the GAN model features with the VAE model to provide a more realistic output rather than the blurry output from VAE model.

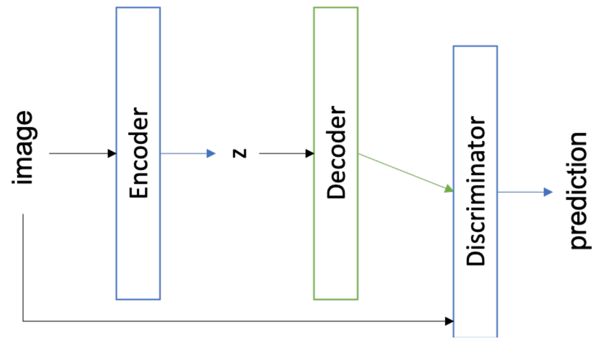
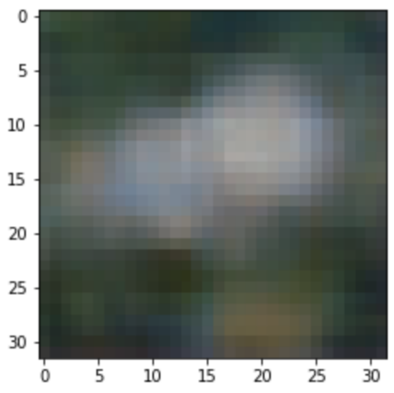


Figure 14: CNN with Residual Blocks architecture

For this case, the encoder and decoder are treated as seperate networks. The discriminator output is used for determining the loss for the decoder thus influencing the decoder weights. The weights for the encoder were carried as before with the VAE model. The weights for the discriminator were carried as with the GAN model. The output for the VAEGAN model shown in Figure 15


(a) VAEGAN model

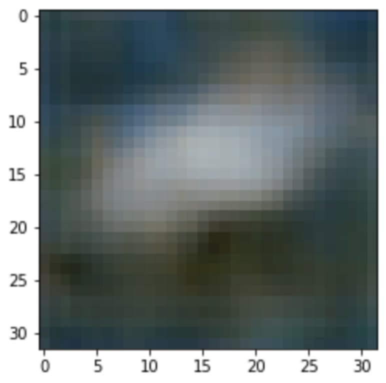

(b) VAE model

Figure 15: Generated Pegasus images

References

[1] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. “Autoencoding beyond pixels using a learned similarity metric”. In: *CoRR* abs/1512.09300 (2015). arXiv: `1512.09300`. URL: `http://arxiv.org/abs/1512.09300`.