Max Wickham, mw1919, 01717673

1. Tutorial 3 Network Training

Task1: Tuning a Classification Model Figures 7, 8, 9 in the appendix show the validation and training accuracy and loss for each of the different augmentation strategies. The soft strategy was a random translation with a height and width factor of 0.2 and a bilinear interpolation. Conversely, the aggressive strategy used was the same random translation accompanied with a random rotation of 0.2 and a random zoom with a height of 0.3. Table 1 shows each strategies validation accuracy.

The soft augmentation strategy had the best result and the hard augmentation strategy gave the worst result. When looking at the loss curves for the three strategies it can be noted that only when using no augmentation does the validation loss start to increase with later epochs suggesting that overfitting is starting to occur at this point. In the other strategies, both of which use augmentation the loss is still decreasing by the last epoch suggestion that adding augmentation results in either more epochs or a more complex model needed to give overfitting due to the larger input space. This is good as it prevents the model from being too specific to the training data.

The poor performance of the hard augmentation may be due to the dataset being standardised at a similar zoom and rotation level meaning that this augmentation is forcing the model to be able to classify a much greater range of possible inputs requiring a more complex model or more epochs to get the same classification accuracy.

Batch normalisation was added after each of the max pooling payers following the convolution layers, dropout layers were added in the same place with a dropout value of 0.2 used. Dropout came first when using both. Table 1 shows adding both batch normalisation and dropout was the most effective strategy. According to arXiv:1502.03167 [5] batch normalisation prevents "small changes from being amplified into larger suboptimal changed" which can make the model more stable giving a higher accuracy. As said in the slides [7] dropout decreases overfitting so also increases accuracy in this case.

Initialisation all the weights to zero gave an accuracy of exactly 10%. This suggests that the input to the model has no effect on the output after training as the same number is always predicted. This is due to the zero values having no gradient so the weights aren't adjusted.

Figure 1 shows the loss for each of the learning rates. The larger the learning rate used the faster the loss decreases which is due to greater steps being taken along the gradient to reduce the loss [7]. None of the three strategies used have any sign of overfitting with the validation loss always being very close to the training loss for a given learning rate. In this case the best learning rate tried is 0.003.

Strategy	Validation Accuracy
No Augmentation	78%
Soft Augmentation	82%
Hard Augmentation	73%
Dropout	82%
Batch Normalisation	80 %
Batch Normalisation and Dropout	84%
Zero Weight Initialisation	10.000%

Table 1. T3.1 Validation accuracy for different models strategies

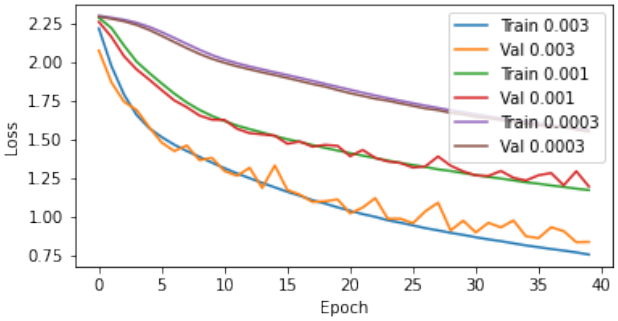

Figure 1. T3.1.4 Validation and Training Loss For Different Learning Rates

2. Tutorial 4 Introduction to Common Neural Network Architectures

Model	Test Acc.%	Time(s)	Epochs	Inference(ms)
Scratch	53.8	531	9	0.40
Pre Train	47.0	660	20	0.40
Fine Tuning	54.4	531	9	0.39
DenseNet121	55.4	644	7	0.72

Table 2. T3.1 Results for different training methods

Task1: Classification on Tiny Image Net As can be seen from table 2 and figure 2, using a pre-trained VGG-16 model and freezing the weights gave significantly worse

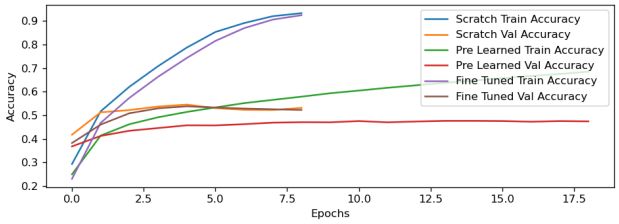

Figure 2. T4.1 Validation and Training Accuracy for Different Training Methods

performance than either training VGG-16 from scratch or fine tuning the weights. Despite having a faster training time per epoch even after twice the epochs as the other strategies the test, training and validation accuracy was roughly 8% lower. This suggests perhaps that the features learned by the pre trained VGG image net model do not transfer well to the dataset used here. The features do work to an extent as it is still possible to get a relatively high test accuracy compared to training from scratch and is certainly better than untrained features.

There is little difference in the results between training the model from scratch or fine tuning the pre trained model which corroborates the idea that the pre-learned features don't transfer that well to this dataset. Interestingly, the fine tuned training accuracy is lower than that of training from scratch but the test and validation accuracy is marginally higher. This suggests that the features in the fine tuned model are more generalised and less over fit to the data. The result of this indicates that fine tuning the weights is better than starting from scratch but doesn't make a huge difference in this case.

The DenseNet121 model provided by Keras was used as the alternative model with its pre-trained weights being fine tuned to give the results shown in table 2. This model gave a better test accuracy with an increase of about a percent combined however, with a longer training time per epoch and greater inference time. The increased accuracy is expected as according to the keras website [2] DenseNet121 does have a higher score on the ImageNet challenge than VGG16.

The GPU used for all the results reported for this task is a Tesla P100-PCIE-16GB.

3. Tutorial 5 Recurrent Neural Networks

Task1: RNN Regression The plots in figure 3 show the predictions made for different LSTM window sizes. A common theme between all three of the plots shown is that despite the window size they all seem to lag slightly behind the actual dataset values, with this being especially noticeable at month 125. A reason for this could be that the value predicted is most correlated to the score from the last month giving a prediction close to this value. When training LSTM of different dimensions were tried as well as combining multiple LSTMs along with dense layers and dropout. No model tried was able to give a significant improvement as compared to the original model however meaning significant more complexity and epochs may be needed to achieve more accurate results.

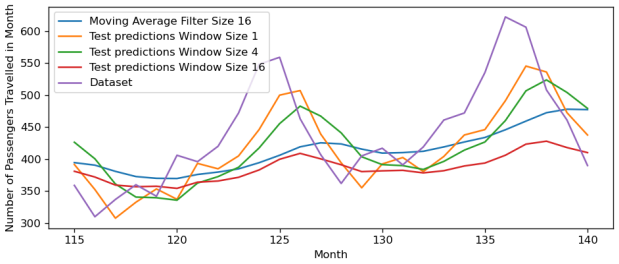

Figure 3. T5.1 Test predictions for different LSTM window sizes with a moving average filter and true values for reference

As can be seen in the plot the greater the window size the more smoothed out the predictions tend to be with the effect looking similar to that of a moving average filter of size 16, a plot of which has been added for reference in figure 3. This may indicate that the LSTM in weighting the effect of each input too evenly.

Task2: Text Embeddings Table 3 shows the test accuracy for the three different model types. As can be seen adding an LSTM and a larger embedding dimension made essentially no change to the test accuracy. Figures 10,11 and 12 in the appendix show the training and validation accuracy curves. For the two methods that used an LSTM the validation accuracy settled almost immediately suggesting that it is difficult to get an accuracy much higher than the one achieved. When using the custom embedding the validation loss starts to increase over time indicating that overfitting occurs, a problem that is removed when using the glove embedding due to the glove embedding being more generalised. The use of the glove embedding did give an increase in accuracy which is expected as the embedding better represents the meaning of a word as opposed to being over fit to the training data in the custom embeddings. The lack of an LSTM gave the same score in the example sentences given, "the movie is boring and not good" for the negative score and "the movie is good and not boring" for the positive score. This is expected as the two sentences use the exact same words and so without a sense of order the model has no way of inferring the different sentiment.

The list of words most similar to "book" was generated for the two LSTM types. When using the Glove embedding the words generated were of a similar meaning or genre to "book", for instance "novel" and "author". The custom embedding, on the other hand, had no similarity in meaning, for example "points" and "stick". This is due to the fact that the glove embedding is trained in such a way as to give

a similar embedding to words of a similar meaning and to allow combinations of words to give a summed embedding similar to words that have the same meaning as the words combined [8]. This is not the case in the custom embedding, which is simply trained to try contain sentiment within the embedding.

Model	Test Accuracy	Neg. Score	Pos. Score
No LSTM	85.24%	0.368	0.368
LSTM	85.11%	0.224	0.306
LSTM and Glove	86.82%	0.438	0.484

Table 3. T3.1 Test accuracy and score difference for different models

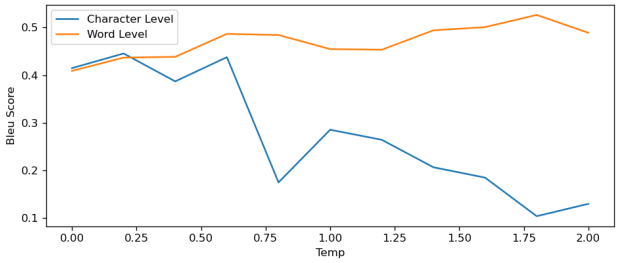

Figure 4. T5.1 Bleu score against temperature for character and word level models

Task3: Text Generation Figure 4 shows how the bleu score is affected by temperature. For the character level model the trend is clearly that the bleu score decreases with a higher temperature, whilst for the word level model the blue score seems to increase slightly as temperature is increased. Since the bleu score does not take into account sentence meaning it is expected that in the case of the character level model an increased prevalence of words that don't exist gives a lower bleu score as the score is a measure of the number of matching words between the true text and the generated [1]. Since the word level model doesn't give non existant words there is not much change in it's bleu score.

When increasing the temperature the generated sentences changed significantly. The sentences generated with extremely low temperatures contained a lot of repetition but only real words were generated and the grammatical accuracy and sentence meaning were not bad. As the temperature was increased so did the variation in the text along with the amount of incorrect words, accompanied with a decrease in overall sentence coherence. A temperature of about 1 gave the best combination of variation and accuracy for the character level model. For the word level model the problem of incorrect words is not present due to the words themselves being generated, meaning the change is only in variation and grammar. Since the grammar is bad for any temperature the change is not extremely noticeable and the decrease in repetition is the bigger change as the temperature is increased. The sentences generated do tend to make less sense the greater the temperature used to generate them.

4. Tutorial 6 Representational Learning

Task1: Non-linear Transformations Table 4 shows the results found for the three different encoding methods used. The final architecture of the autoencoders with no CNN was three dense layers with 256, 128 and 10 neurons respectively. The first two layers also employed RELU activation and the decoder architecture was the same as the encoding reversed. When using a CNN three different CNN layers were used with 128, 64 and 64 layers each and all CNN layers were of size 3x3. The CNN layers were followed by three dense layers of size 256,64 and 10. All layers used RELU activation apart from the last layer and the decoder used the same architecture in reverse.
As can be seen from table 4 the use of neural networks gave a large increase in accuracy as compared to PCA. The reason for the increased accuracy using neural nets is due to the ability to approximate non linear functions unlike linear PCA [9]. The none CNN strategy has an optimal solution that is the same as a non linear PCA [9]. Moving from a traditional NN to a CNN gave a small increase in performance in terms of accuracy although a fairly large decrease in MSE was found. This is expected as CNNs are better at extracting features from an image and so should be better at providing features for an encoding as compared to the traditional network.

Architecture	Accuracy %	MSE Train	MSE Val
No CNN	91.67	0.0087	0.0089
CNN	96.09	0.0059	0.0068
PCA	81.48	0.0258	0.0256

Table 4. T6.1 MSE and accuracy for different encoder architectures

Task2: Custom Loss Functions Table 5 shows how the MSE is dependent on the loss function used and figure 5 shows the result filtering noise using a model trained with each loss function. The MS-SSIM and PSNR methods both give similar MSE and result in images that look very alike whereas the SSIM method gives an image that has fairly incorrect colours and gives a worse MSE. The SSIM function does give a similar level of noise removal in the image despite its colour inaccuracy. The increased performance of MS-SSIM vs SSIM is due to it working in a similar manner to SSIM by measuring structural similarity but operated at multiple scales giving a better metric [11]. The degradation in colour seen with SSIM may be due to this metric not only valuing similar pixel values but also similar variance and cross correlation within a patch of pixels [11]. The second two of which still give high scores if the overall average is shifted by a constant amount meaning a colour shift doesn't change two of these three metrics. PSNR may give a better MSE value due to its equations being directly related to MSE and so an improvement in PSNR gives an improvement in MSE [3]. The better visual results in figure 5 from

MS-SSIM vs PSNR may be due to MS-SSIM judging similarity in a manner close to that of a human [11]. Some more information on the implementation of the loss functions can be found in the appendix 7.1.

Loss Function	MSE Val
SSIM	0.00781
MS-SSIM	0.00562
PSNR	0.00552

Table 5. T6.2 MSE for different loss functions

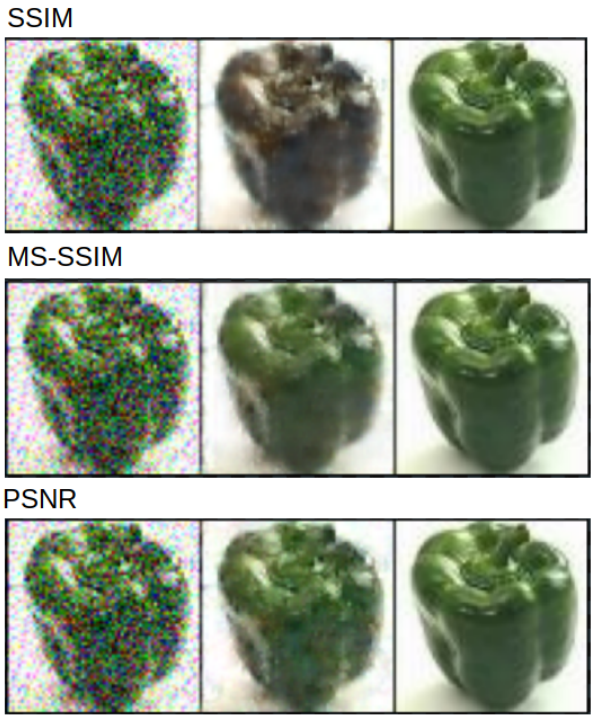

Figure 5. T5.1 Noisy Image, De-noised Image and Clean Image for different loss functions with SSIM resulting in bad colour accuracy and MS-SSIM giving the best result

5. Tutorial 7 Variational Autoencoders

Task 1: MNIST generation using VAE and GAN As can be seen in table 6 The use of KL divergence in the variational autoencoder model gave a slightly higher MSE but a noticeable improvement in the Inception Score. The reduction in MSE is expected as without the KL divergence the loss is simply based on the difference between the generated image and the correct image thus optimising the model to reduce this value [6].
If the distribution of the generated images is diverse in such a way that images with a uniform spread of labels are generated i.e. $p(y)$ is uniform then a higher inception score is given [10] as this is the distribution of the true data. Since the KL divergence layer loss function tries to optimise to such a distribution [10] its presence helps to increase the inception score as is found in table 6.

Table 6 shows that the cGAN network gave a higher Inception score than either of the VAE methods. Although a cGan does not rigorously learn the likelihood distribution unlike the VAE with KL it implicitly learns it through aiming to minimise the difference between $P(X)$ and $P(Z|Y)$ [4]. In this case the GAN method has been more effective at creating accurate images and learning the distribution of images that the VAE methods, but this might not always be true.

Model	MSE	Inception
VAE KL	0.0120	7.29
VAE No KL	0.0110	5.95
GAN	na	8.13

Table 6. T6.1 MSE and Inception scores with and without KL divergence as well as for the GAN model

Task 2: Quantitative VS Qualitative Results Figure 13 in the appendix shows several combinations of images with colour added by the MAE model and by the cGAN model approach compared to the real counterpart. The MAE found for the cGAN model was 0.0458 and was 0.0449 for the MAE trained model. In the images shown the cGAN approach appears to lead to more variation in colours within the image and more saturation. In the case of the MAE model the lower saturation may be due to the model trying to reduce the absolute error from the true image and so chooses colours that are closer to the average of all possible colours to try and minimise the error if it is wrong, resulting in the lower MAE score found. In the case of the cGAN the model is not trying to create the exact same colours as the true image but is instead trying to create colours with the same probability of occurring as the true colours [4]. This means it is able use a greater saturation without risking higher errors in the same way as the MAE model as the error is not directly related to the difference in value between the colour it produces and the true colour.

6. Tutorial 8 Reinforcement Learning

Figure 6 shows the results found using SARSA and Q-Learning with ϵ-greedy and softmax. The modification made are given in the appendix in section7.2.

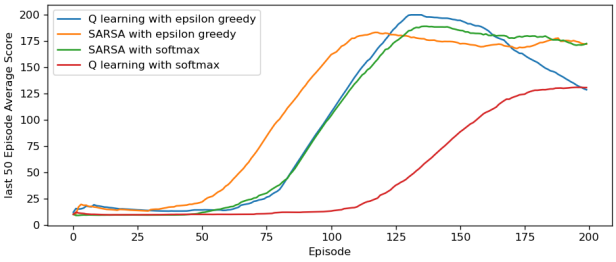

Figure 6. T5.1 Average Score of last 50 episodes for different RL techniques

References

[1] Bleu score. https://en.wikipedia.org/wiki/BLEU.

[2] Keras applications. https://keras.io/api/applications/.

[3] Peak signal-to-noise ratio as an image quality metric. https://www.ni.com/en-gb/innovations/white-papers/11/.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[6] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

[7] C. C. Krystian Mikolajczyk. Icl eee deep learning part3, 1-3.

[8] C. C. Krystian Mikolajczyk. Icl eee deep learning part5, 1-2.

[9] R. Z. Raquel Urtasun. Principal components analysis and autoencoders. https://www.cs.toronto.edu/~urtasun/courses/CSC411/14_pca.pdf.

[10] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. Variational approaches for auto-encoding generative adversarial networks, 2017.

[11] D. M. Rouse and S. S. Hemami. Understanding and simplifying the structural similarity metric. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.506.6577&rep=rep1&type=pdf.

7. Appendix

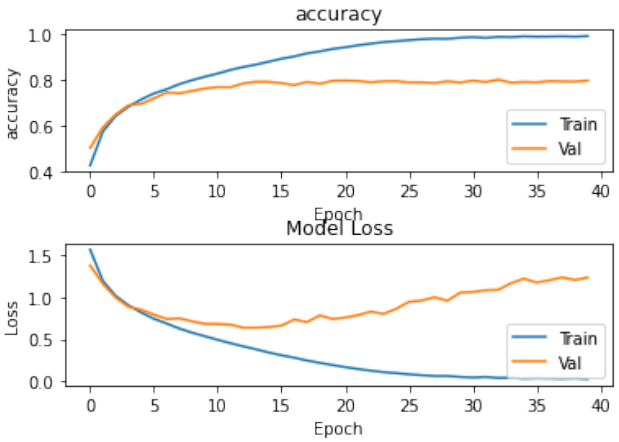

Figure 7. T2.1.1 Validation and Training Loss No Augmentation

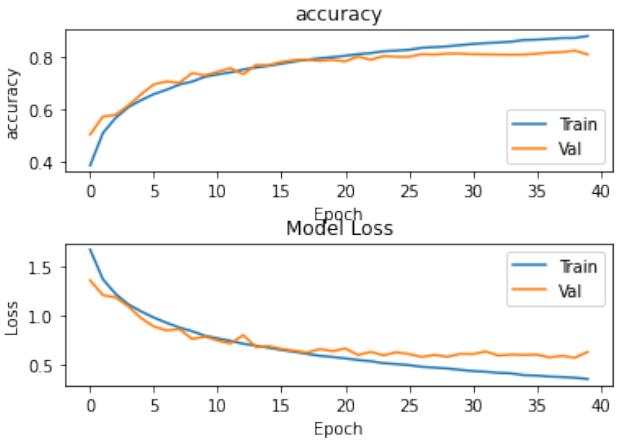

Figure 8. T2.1.1 Validation and Training Loss Soft Augmentation

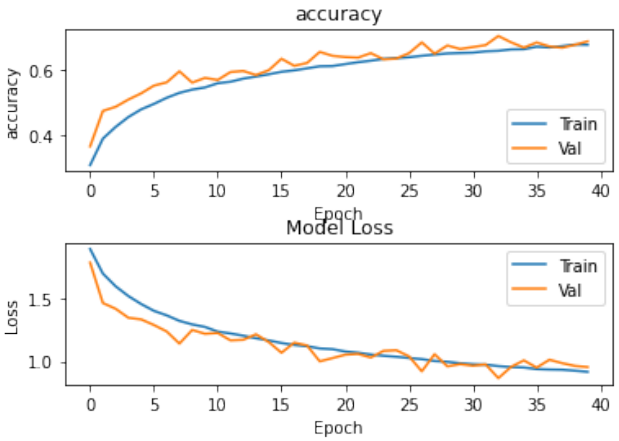

Figure 9. T2.1.1 Validation and Training Loss Hard Augmentation

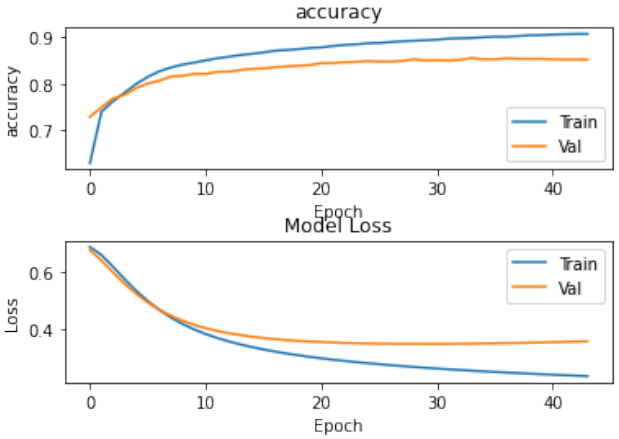

Figure 10. T2.1.1 Validation and Training Accuracy no LSTM

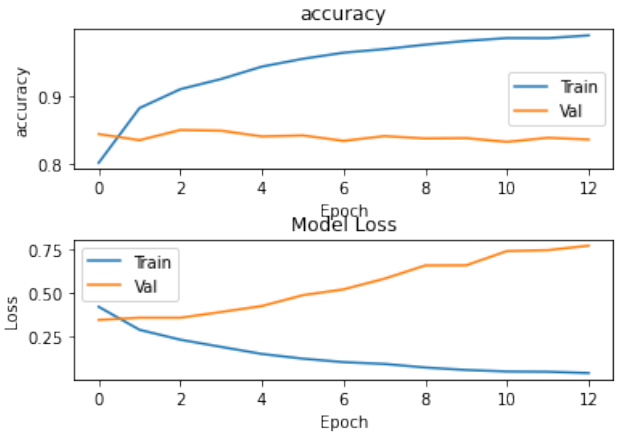

Figure 11. T2.1.1 Validation and Training Accuracy custom Embedding

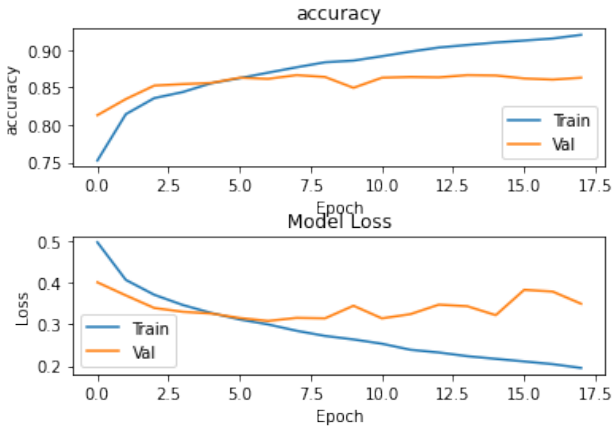

Figure 12. T2.1.1 Validation and Training Accuracy glove Embedding

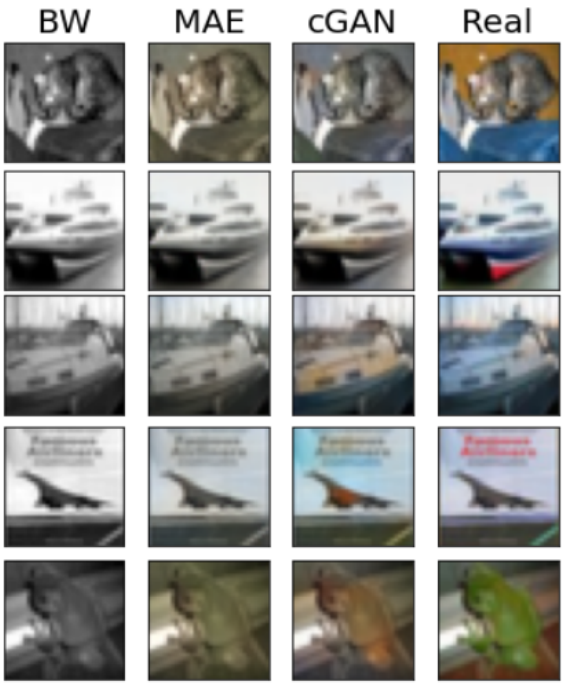

Figure 13. T7.2

7.1. Custom Loss Function

MS-SSIM and SSIM both give higher scores when two images are similar which was the opposite to what we want our loss functions to do. Thus $1 - L$ (where L is the output of these functions) was taken as 1 is the maximum score. It was found this method performed better than using $1/L$ in testing. In the case of PSNR $1/L$ was used as there is no limit on the value of PSNR with a higher value score being better.

7.2. DQNAgent Modifications

In order to implement softmax the "act" function had to be rewritten. The predicted scores for the two actions from the given state were found and then the softmax values of these scores were generated using teh given function. Finally a number between 0 and 1 was randomly generated. If this number was less than the first value given from the softmax output a 0 was returned otherwise a 1 was returned. When implementing SARSA the score chosen for a given next state was no longer just the highest score given by the two available actions. Instead the action taken was chosen in the same way as act function, (either using epsilon greedy or softmax discussed above). The action chosen using either of the two methods was then used to choose which next score was taken.