# Udai Arneja, ua518, 01512300

## 1. Convolutional Neural Network Training

**a. Data Augmentation** Table 1, reports the validation accuracy for CNN classification with various data augmentation techniques applied. With no data augmentation the model shows a good validation accuracy, but this occurs very early on in the training ( epoch 13) after which there is clear overfitting (see Figure X). This is likely due to the model memorising the data very quickly after which, the generalisation error grows greatly. When a small amount of data augmentation is applied the model takes longer to learn the data but generalises well after 40 epoches. This produces the best results with limited overfitting. Additionally, if the data augmentation strategy is too aggressive the model struggles to learn the data effectively producing a poor performance.

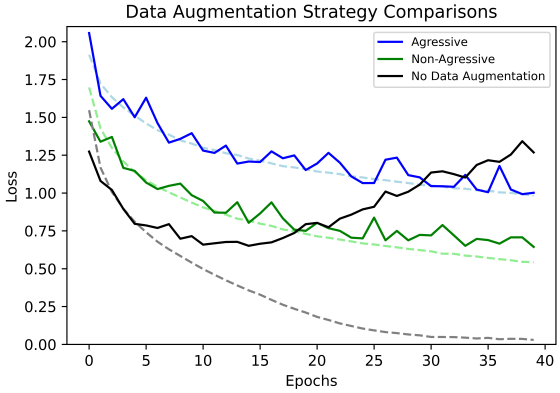


Figure 1. Three Data Augmentation Strategies. Dotted line: Training Loss, Solid line: Validation Loss

**b. Batch Normalisation and Dropout Layers** The effects of using just Dropout Layers or both Dropout and Batch Normalisation on validation accuracy is largely affected by the positioning of the layers, as compared to just using Batch Normalisation which seemed relatively insensitive to changes in positioning. Implementing Dropout Layers causes a loss in information and hence implementation in earlier layers means more information is lost, producing poorer validation accuracy's. Changing the dropout rate (see Table 8), has minor affects on the validation accuracy

| Model | Validation Accuracy (%) |
|---|---|
| No Changes | 79.29 |
| Aggressive Data Aug | 66.60 |
| Non-Aggressive Data Aug | 82.68 |
| Batch N. (1,2) Only | 80.72 |
| Dropout (2,3) Only | 82.24 |
| Dropout & Batch N. (2,3) | 82.26 |
| Zeros Initialisation | 10.00 |
| Xavier Uniform Initialisation | 79.45 |

Table 1. Comparing the highest validation accuracy of Models with stated changes over 40 epochs. Note, a dropout rate of 0.3 is used in all cases above, see below for detail.

- 0.3 marginally outperformed both 0.1 and 0.6, hence a dropout rate of 0.3 is used throughout.

**c. Kernel Initialisation** This shows the extreme effects of the vanishing gradient problem, the weights in the model are equal or close to 0 and hence experience no training, resulting in a validation accuracy of 10%.

**d. Learning Rates** Figure 2 indicates that as the learning rate increases the model converges at a faster rate (the loss of black line decreases the fastest). Additionally, larger learning rates show less smooth declines in loss as the model is trained, indicating the loss increases shortly before decreasing again. This is likely due to the training step being too large (due to the large learning rate), such that the model overshoots the gradient descent and the loss increases.

## 2. Common CNN Architectures

**a. Training Methods for Common Models** When developing a model for a common problem, such as image classification, there are usually existing models that can adapted for more specific uses. To apply these models they can either be trained from scratch (random weights), frozen with only the output dense layers trained or further training or tuning the entire model. Figure 3 shows these approaches for the applying a common model to a modified dataset. As expected, the model trained from random weights will have a large training accuracy as it memorises the dataset, the validation accuracy remains low as this will not gener-

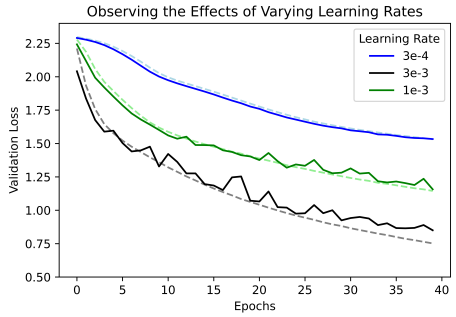


Figure 2. Training (dotted line) and validation (solid line) losses. Early stopping was applied with a patience of 10.

alise well. Comparatively, the loaded models perform significantly better as they have been trained on large datasets and have good generalisations. Both the tuning and transfer methods of applying a model also face rapid overfitting, with tuning method outperforming the transfer model when validated. This is likely due to the tuning model becoming more accustomed to the image type. A further, common CNN architecture ResNet50 was applied to this dataset under a tuning method, and performed similarly to the VGG16 equivalent model.

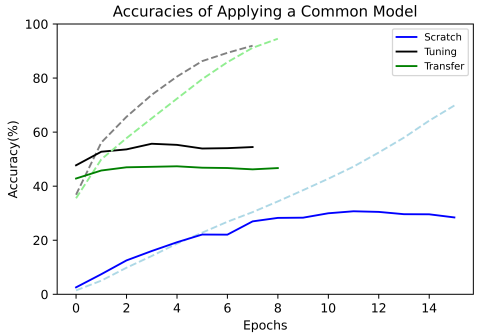


Figure 3. Training (dotted line) and validation (solid line) categorical accuracies. 'Scratch' model uses a random initialisation of weights. 'Tuning' model initially uses the 'imagenet' weights which are trained further. 'Transfer' model uses 'imagenet' initial weights but only trains the output dense layers. Early stopping was applied with a patience of 4.

| Model | Validation Acc. (%) | Training Times (sec - epochs) | Inference Time per Image (ms) |
|---|---|---|---|
| Scratch | 29.95 | 918.08 — 15 | 0.34 |
| Tuning | 55.06 | 458.43 — 7 | 0.31 |
| Transfer | 47.45 | 196.63 — 8 | 0.31 |
| ResNet50 | 53.86 | 421.87 — 7 | 0.47 |

Table 2. Observing effectiveness of methods of applying trained models to new datasets.

Additionally, Table 2 indicates the total training times for applying the above models to the new dataset, giving an insight on smaller level into which methods are most resource intensive.

## 3. Recurrent Neural Networks

**a. Regression - Effects of varying Window Sizes** In Figure 4 the predicted next values in a time series test set are shown for varying window sizes. The figure indicates that as window size increases, the predictions worsen up to a point at which they increase again, before dropping off at window size 14.

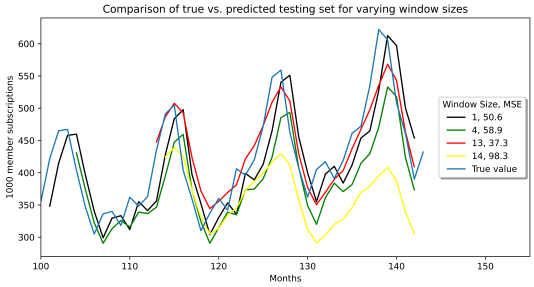


Figure 4. Training (dotted line) and validation (solid line) accuracies. Early stopping was applied with a patience of 10.

**b. Text Embeddings** Embeddings transform text data into numerical vectors that represent both the semantic and properties of the word. LSTMs overcome the short-term memory limitation of RNNs by enabling multiple layers to interact. The GloVe [4] embeddings used have been trained on Wikipedia and Gigaword resources to produce word vectors.

From Table 3, it can be seen that while the test accuracies of the models are similar, models with LSTMs provide considerably better classification predictions of Positive or Negative Sentiments when given a statement. Additionally, model (2) which has trainable embeddings performed slightly better than model (3) (which also uses LSTMs) when predicting positive sentiment, despite a lower test accuracy. Comparatively, for negative sentiment model (3) outperformed model(2) by a factor of 10. This could perhaps be due to the training dataset being strongly positive, with very little negative sentiments to learn from. And hence in this case, the GloVe embeddings, being pre-trained will perform better.

**c. Text Generation** The BLEU Score quantitatively evaluates generated text by finding contiguous word elements in the reference text. Temperature is a trade-off between the variability of the predicted word next output distribution and probability of the individual word. It can be generally

| Model | Test Accuracy (%) | Positive Sentiment | Negative Sentiment |
|---|---|---|---|
| (1) | 85.17 | 0.0000046 | 0.0000046 |
| (2) | 83.19 | 0.79 | 0.0047 |
| (3) | 86.70 | 0.77 | 0.039 |

Table 3. Model (1): No LSTM & Average Pooling, Model (2): LSTM & Trainable Embeddings, Model (3): LSTM, Non-Trainable Embeddings & Glove Initialisation

observed that both models do not perform well, with the generated text making little sense in most cases. The figure shows that the character model performs worse as the temperature increases while the word model remains relatively constant. This indicates that for the word model, the most probable prediction also tends to have a similar output distribution as the current model. Comparatively, in the character model the predicted characters with similar distributions to the current model have a poor performance generating text and are notably different to those predictions that are most probable. In the word model the grammar seems to be somewhat coherent for all temperatures, as with the character model at lower temperatures. As the temperature increases in the character model, both the spellings and grammar falls quickly no longer resembling the English language at a temperature of 1.3.

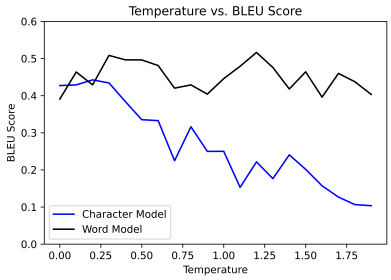


Figure 5. Evaluating the BLEU Score at different temperatures for a character and word text generation model.

## 4. Autoencoders

**a. Non-Linear Transformations for Representation Learning** The results indicate the clear advantages of using Non-Linear Transformations for Representation Learning. This is expected as linear transformations are limited in their expressive power as compared to non-linear transforms that can be stacked for more abstract representations, and hence better feature extracting [1]. In reducing images to a latent space with a smaller dimension, extracting useful features is important and hence the data type means that convolutional layers will allow for better feature extraction, as reflected in the results - the convolutional model performs better than the model without convolutional layers [3].

| Model | Encoder | Decoder |
|---|---|---|
| Conv. (1) | [D512, D128, D32] | [D32, D128, D512] |
| No-Conv. (2) | [C128,P,C256, P,C512,P,C512] | [D1024,D1024] |

Table 4. Architectures of Autoencoders in Table 5. Note, D512 is a Dense Layer with 512 neurons. P is a Max Pooling Layer. All kernel sizes are 3x3 for the Convolutional Layers.

| Model | Classifier Accuracy (%) | Training MSE | Validation MSE |
|---|---|---|---|
| PCA | 80.97 | 0.0258 | 0.0256 |
| Conv. (1) | 94.25 | 0.0088 | 0.0094 |
| No-Conv. (2) | 93.24 | 0.0099 | 0.010 |

Table 5. Comparing performance of autoencoders with different transformations methods

**b. Custom Loss Functions** From Table 6 it can be see than the MAE gives the most ideal Mean Squared Error Value, with $\frac{1}{PSNR}$ also achieving a good result. The Mean Absolute Error is a common loss function, used when training many models. The PSNR (peak signal to noise ratio) compares the compressed image to the original - evaluating its compression quality. The higher the PSNR, the higher the the quality of the reconstructed image. Table 6 also shows that the MS SSIM gives a high error, compared with the MAE and hence is perhaps not an appropriate loss function for this type of data.

| Model | Loss Function | MSE |
|---|---|---|
| UNet | MAE | 0.00589 |
| UNet | $\frac{1}{PSNR}$ | 0.00632 |
| UNet | SSIM | 0.00704 |
| UNet | MS SSIM | 0.0153 |

Table 6. Comparing the Mean Squared Error for different loss functions when trained using a UNet Model.

## 5. Variational Autoencoders & Generative Adversarial Networks

Variational Autoencoders (VAE) use the same principle as Autoencoders but apply this to generate new data, rather than for representation learning. Generative Adversarial Networks (GAN) are also used for data generation, but use a two step Generation - Discrimination process. To evaluate the generated data, the MSE, Inception Score and qualitative metrics are used.

**a. MNIST generation using VAE and GAN** From Table 7 it can be observed that the model with both the KL Divergence and Negative Log Likelihood losses seems to have a better Inception Score as compared to solely using the Negative Log Likelihood loss, with their MSE being

similar. The inception score of 7.336 (including KL) is greater than 6.053 (no KL), which correlates to a qualitative review of the prediction, indicating that the inception score is a useful evaluation tool. Additionally, the table shows that the GAN produces better generated data than the VAE, giving a higher Inception Score.

| Model | MSE | Inception Score |
|---|---|---|
| VAE with KL Div. Loss | 0.012 | 7.336 |
| VAE, no KL Div. Loss | 0.011 | 6.053 |
| GAN | - | 8.283 |

Table 7. T1.2 Comparing evaluation functions for data generation models.

**b. Quantitative vs. Qualitative Results** The two models - the cGAN model and the UNet Autoencoder model - are applied to generate coloured images from black and white ones. From a quantitative point of view the UNet model has a lower Mean Absolute Error (0.0449) than the cGAN (0.0458) and hence it would be evaluated as better. This is likely due to the MAE model producing very gray-scale based images, in order to reduce the MAE - grey-scaled images are on average the closest to any other colour.However, when taking a qualitative approach, it can be argued that the cGAN model is better [2] .From the results in Figure 6, the cGAN does colour the right segments of the image with the right colour scheme (front and side of the truck are red for example) as compared to the UNet model which provides a very limited colour change from the B&W image. While the colours provided by the cGAN model are wrong, it generates a better coloured image than the UNet model - emphasizing the importance of qualitative observations.

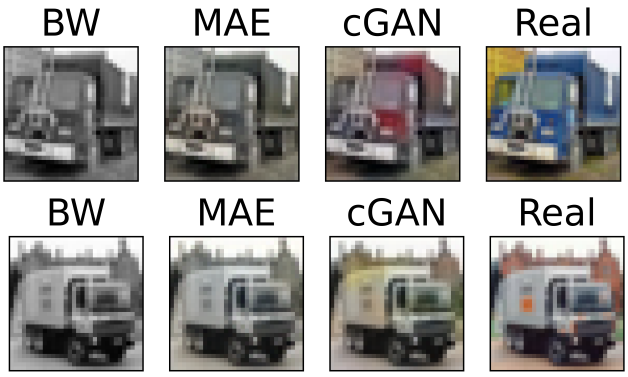


Figure 6. Comparing images generated from MAE models with cGAN models.

## 6. Reinforcement Learning

**a. On-policy vs. Off-policy** The results in Figure 2, provide clear insight into the comparison between on-policy (SARSA) and off-policy (Q-Learning).

The on-policy method (SARSA) shows a very different average-reward plot when the Action function applied changes from Softmax to $\epsilon$-Greedy, as compared to the off-policy method (Q-Learning) which, relatively, does not differ greatly. This is expected as the Q-Learning method already has a high variability in learning policy in learning, comparatively the SARSA method does not - and hence, when the $\epsilon$-Greedy function is applied the learning. Additionally, the figure indicates that the $\epsilon$-Greedy action function is beneficial for learning as the average-reward curve shifts to the left (receives higher rewards earlier).

The off-policy method, additionally, shows a kink at roughly 160 epoches, after which the average-reward begins to slightly decrease - in comparison, the on-policy method (SARSA) consistently has an increasing average-reward with only a decrease in the rate of this increase.

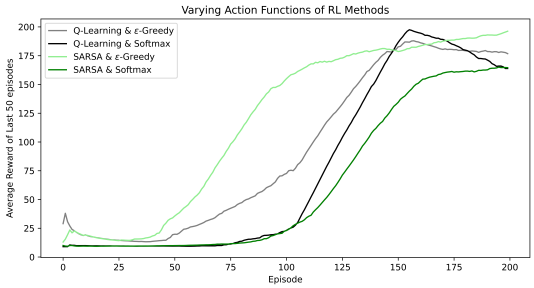


Figure 7. Reinforcement Learning for various strategies with different action functions.

Modifications made to DQNAgent :

When implmenting the Softmax Action function, a change was made to the 'act_method' which was made to randomly choose a value between any of the softmax values, while in $\epsilon$-Greedy a random value generator was used.

## References


[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. 2014.

[2] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014.

[3] A. Newson, A. Almansa, Y. Gousseau, and S. Ladjal. Taking apart autoencoders: How do they encode geometric shapes ? 2018.

[4] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.


## 7. Appendix

Additional results and discussions can be included here

| Model | Validation Accuracy (%) |
|---|---|
| Dropout (2,3 - 0.3) Only Model | 82.24 |
| Dropout (2,3 - 0.6) Only Model | 79.43 |
| Dropout (2,3 - 0.1) Only Model | 81.11 |

Table 8. Comparing the highest validation accuracy of Models (changing the dropout rate) over 40 epochs.

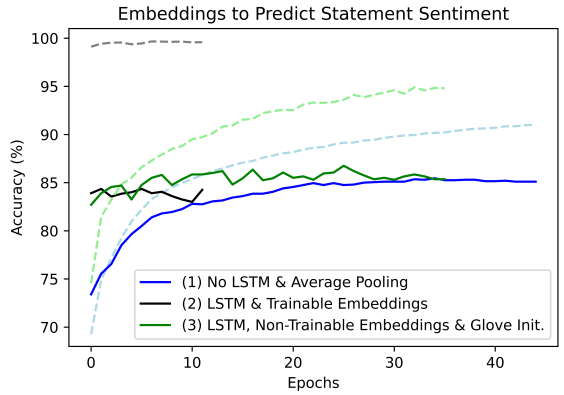


Figure 8. Training (dotted line) and validation (solid line) accuracies. Early stopping was applied with a patience of 10.