jj1220, 01860908

1. Network Training

Task 1.1 In the original run without data augmentation in Figure 11, the training loss demonstrates a decreasing trend with respect to epoch as expected. The validation loss decreases initially but rises quickly at the end with the lowest validation accuracy of 0.7967 in Table 1 because the original dataset is relatively small which could easily overfit and results in large gap between the validation and training loss. Zooming with height 0.2 increases number of training examples and hence improves validation accuracy and generalisation in Figure 12, but the gap is still significant. Flip horizontally and zooming with height 0.3 is a more aggressive approach which further increases number of training data and results in best validation accuracy of 0.8301 and best generalisation between the three approaches shown in Figure 13. It is noticeable that data augmentation improves validation accuracy at a cost of decreasing training accuracy as data augmentation adds more variety to the model.

Tuning Technique	Train Acc.	Val Acc.
No Augmentation	0.9890	0.7967
Zoom (0.2)	0.9596	0.8093
Horizontal flip zoom (0.3)	0.8910	0.8301
Conv+Act+Pool+Drop	0.8162	0.8168
Conv+Act+BN+Pool	0.9916	0.7901
Conv+Act+BN+Pool+Drop	0.8310	0.8373
Zero kernel initialization	0.09998	0.1000

Table 1. Accuracy with different tuning techniques

Task 1.2 Shown in Figure 14 Dropout with a factor of 0.3 in between every pooling layer and the next convolution layer has a close training and validation accuracy (0.8162 & 0.8168) in Table 1 as deactivating random neurons prevents the network from replying heavily on specific inputs so it has to spread the weights to learn more general features hence improves generalisation. However better generalisation comes with the price of losing accuracy. Applying batch normalization after every activation function has significantly increased the speed of convergence shown in Figure 15. This is because layer outputs are transformed into a Gaussian distribution which solves exploding and vanishing gradient problem with large input values in the following layer. Since weights are now within a smaller range, higher learning rates can be taken and thus reduces convergence speed. Combining dropout and batch normalization in the order Conv+Act+BN+Pool+Dropout achieves the best validation accuracy 0.8373 with good generalisation in Table 1.

Task 1.3 Initializing all weights with zeros means all hidden units will have an equal influence on the cost, forbidding different neurons from learning different features [4]. Hence, the performance is very poor shown in Table 1 with both training and validation accuracy around 0.1.

Task 1.4 Shown in Figure 1, the gap between training and validation loss curve is small as SGD has stronger regularization effects and thus outperforms Adam in generalization. Furthermore, as learning rate decreases, both losses increases, but the gap in between are reduced and therefore leads to better generalization.

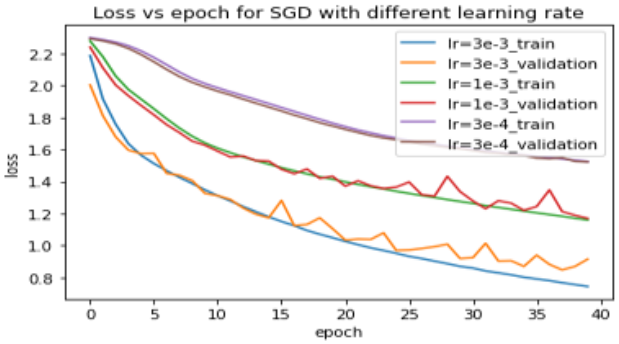

Figure 1. SGD with different learning rates

2. CNN Architecture

Task 1.1 As shown in Table 2, transfer learning initialized with pre-trained weights from ImageNet has the fastest training time with 8s/epoch. This is because it aids the network by taking advantages of wide range of features learned on ImageNet and only trains the classification stage [5]. Even though features in the first few layers can be well transferred as they detect more general patterns, later layers that are responsible for more dataset-specific and complicated patterns could decrease accuracy as ImageNet and Tiny-ImageNet have different resolutions and distributions of objects [5], so the test accuracy has significantly decreased to 0.473 seen in Figure 2.. Fine tuning on the other hand achieves the highest test accuracy 0.545 among the three approaches as it not only has good initialization but also it allows the network to adapt the new dataset by retraining the whole architecture including the feature extraction stage. Finally, DenseNet169 is a much deeper network

	VGG16	Transfer	FineTune	Dense169
Train Acc.	0.946	0.968	0.949	0.989
Test Acc.	0.542	0.473	0.545	0.586
Infer(ms)	0.2275	0.2165	0.2132	0.8744
Epochs	11	10	11	10
Time/e(s)	18	8	18	69

Table 2. VGG models with different strategies and DenseNet169

but with less parameters and as layers are connected to each other using densely connected blocks which brings the benefits of moderating the vanishing-gradient and reinforcing feature propagation [2]. Therefore it achieves the highest training and validation accuracy stated in Table 2. However, DenseNet169 has the longest training time of 69s/epoch due to large number of densely connected layers.

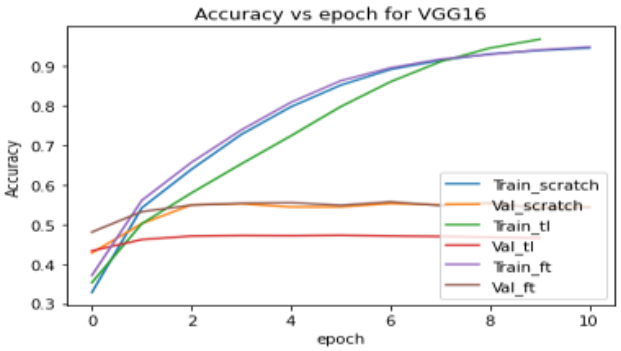

Figure 2. Accuracy for VGG16

3. RNN

Task 1.1 As demonstrated in Figure 3, increasing the window size makes the prediction curves smoother and the training loss tends to decrease in Table 8. This is because larger window size uses more previous data as input and thus helps identify long-term trend so prediction is more accurate with decreasing loss. However, if the window size is too larger (W=15), it starts to overfit and stop following the trend, resulting in poor performance in the test set.

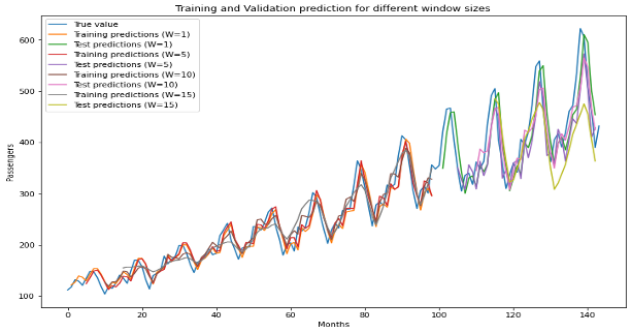

Figure 3. Predictions for different window sizes

Task 2.1 In Table 3, embedding of dimensionality 1 without LSTM has the lowest training accuracy and a relatively good validation accuracy. This is because a low dimensionality means there are limited number of trainable parameters (5002) in the encoded vectors and thus the training accuracy will not be optimal. However, it is less likely to overfit, corresponding to the small gap between the two accuracy curves shown in Figure 17. Increasing dimensionality to 300 along with LSTM has increased the number of trainable parameters to 1,570,251 as there are more features in each encoded vector. Also, LSTM memorizes relevant historical information with the presence of the three gates. Therefore, the training accuracy is significantly higher (0.99) but at a cost of losing some validation accuracy with increasing risk of overfiting, corresponding to the larger gap between the accuracy curves shown in Figure 18. LSTM initialized with GloVe demonstrated the best validation accuracy (0.87) among the three as GloVe not only captures local information but also integrates with global information through co-occurrence matrix between words and context [1]. Also, initializing with GloVe has reduced the trainable parameters to 70,251 which balances the trade off between accuracy and overfitting seen in Figure 19.

Embedding	Train acc	Val acc
Dimension 1	0.91	0.85
Dimension 300 with LSTM	0.99	0.84
LSTM with GloVe	0.95	0.87

Table 3. Accuracy for different embedding techniques

Task 2.2 Embeddings of dimensionality 1 without LSTM cannot differentiate positive and negative sentiments and identifies both as relatively negative (closer to 0) as it produces the exact same scores for both ones shown in Table 4. LSTM with GloVe successfully produces a score closer to 1 (0.83) for positive and a score closer to 0 (0.01) for negative. Also, embedding without LSTM encodes the semantic meaning of words whereas GloVe pays more attention on word structure and contexts using the co-ocurrence matrix. The differences can be seen in Table 9, the most 10 similar words of '8' and 'play' encoded with embedding without LSTM all have positive sentiment and similar meanings whereas encoding using GloVe embedding shows words that are similar in structure and appear in similar contexts.

Embedding	Positive score	Negative score
Without LSTM	0.36	0.36
LSTM with GloVe	0.83	0.01

Table 4. Sentiment prediction for different embedding techniques

Task 3.1 Increasing temperature decreases BLEU scores for both character and word level seen in Figure 4 and Figure 5. This is because higher temperature introduces more variety in text generation and therefore BLEU score as a measure of the similarity between the generated text and the reference text decreases. However, word level has a more fluctuating curve. Also, increasing the temperature when

generating sentences will make the sentences less coherent and grammatically incorrect in both levels. However, higher temperature in the character level makes spelling mistakes and produces nonsense as it introduces more variety in character level while higher temperature in word level does not make spelling mistakes but the generated sentences are still grammatically incorrect due to more variety in word level.

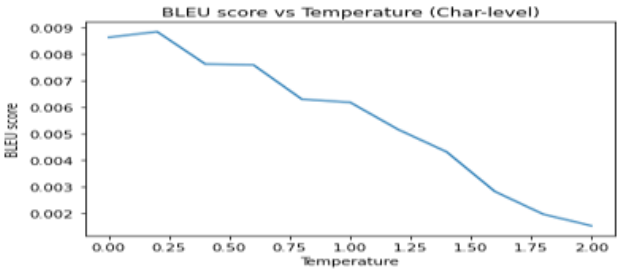

Figure 4. BLEU score with respect to temperature for char-level

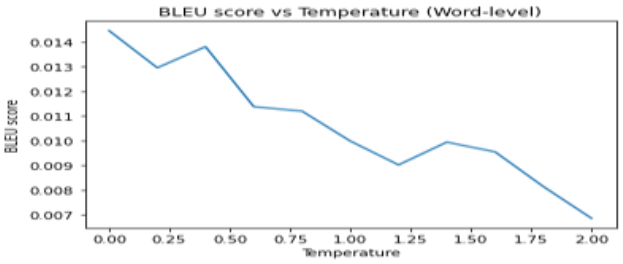

Figure 5. BLEU score with respect to temperature for word-level

4. Autoencoders

Task 1.1 The accuracy results of PCA is the worst among the three approaches seen in Table 5 as PCA is a linear reduction technique, equavalent to a linear shallow autoencoder that finds the principle components with either maximum variance or minimum error. Therefore it can only capture linear relations and some features are lost due to its lossy behavior. Non-linear autoencoders achieves better accuracy performance than PCA as it can capture non-linear relations using non-linear activation functions. Also, with the addition of loss functions and optimizer, weights of the dense layers can be back-propagated to reduce loss. Convolutional autoencoder builds upon the advantages of the Non-linear autoencoder and achieves the best performance because convolutional layers can capture the spatial information within pixels, making it more powerful in extracting features and reconstructing the original input. The initial performance are similar (around 0.81 accuracy and 0.026 MSE in Table 5) because the autoencoders are shallow with only a few layers. The modified non-linear autoencoder in Figure 6 has added two dense layers of 1024 and 2048 neurons in both encoder and decoder and has accuracy of 0.94 and much lower MSE. The modified convolutional antoencoder in Figure 7 consists of three convolutional layers of size 32, 64, 128 and a series of batch normalization, max-pooling and upsampling in both encoder and decoder. An extra convolutional layer is deployed at the end of the decoder to improve smoothness of the image. This model has achieved the highest accuracy around 0.96 and lowest MSE among all the models.

	Accuracy		MSE	
Model	Train	Test	Train	Test
PCA	0.8101	0.8155	0.0258	0.0256
Non-linear	0.8112	0.8176	0.0259	0.0256
Conv	0.8162	0.8186	0.0286	0.0285
Non-linear2	0.9441	0.9404	0.0069	0.0078
Conv2	0.9570	0.9600	0.0061	0.0069

Table 5. Accuracy and MSE for different autoencoders

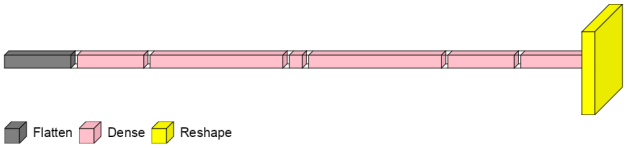

Figure 6. Modified non-convolutional autoencoder

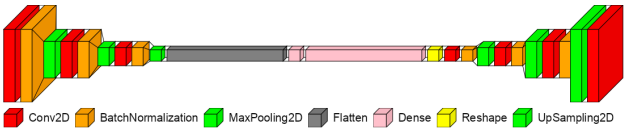

Figure 7. Modified convolutional autoencoder

Task 2.1 Both loss functions perform well in terms of MSE shown in Table 6. However, it is noticable that MAE has a lower MSE and the denoised image is cleaner in Figure 8 than 1/PSNR which still has noticeable noise presented in Figure 9. This is because MAE is less sensitive to the outlier while PSNR depends on mean squared error which is more vulnerable to larger values.

Loss function	Train MSE	Val MSE
MAE	0.0054	0.0055
1/PSNR	0.0072	0.0073

Table 6. MSE table for different loss functions

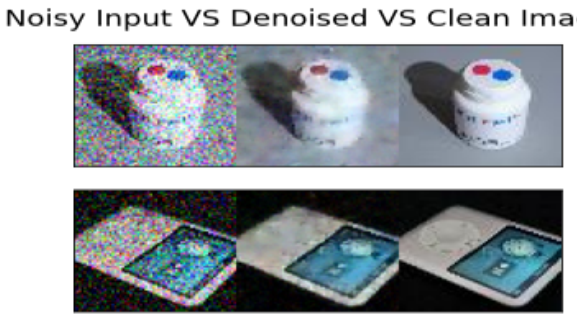

Figure 8. Denoised images with MAE loss

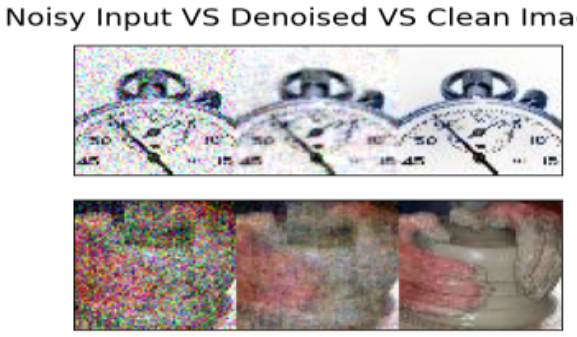

Figure 9. Denoised images with 1/PSNR loss

5. VAE and GAN

Task 1.1 Even though VAE trained with KL loss has a relatively higher MSE seen in Table 7 because the tradeoff between the reconstruction term and the KL loss has forbidden the decoders from focusing only on reconstructing the original input data, KL divergence has brought regularization to the encoded latent space to follow uniform distribution, forming separable dense clusters centered around the origin [3] as shown in Figure 20. Therefore, the decoder can generate more diverse output data with better quality seen in Figure 21, thus resulting in a higher IS score. VAE trained without KL loss only focuses on reconstructing loss and therefore has a lower MSE but may risks overfitting with a scattered latent space in Figure 22. Therefore it produces output with less diversity and quality seen in Figure 23 with a lower IS score. GAN on the other hand, simultaneously trains a generator responsible for generating 'fake' images and a discriminator responsible for distinguishing between the generated images and real training input images. Furthermore, the generator is updated with the gradient information produced by the discriminator to produce images with better quality and diversity seen in Figure 24 to 'fool' the discriminator. Therefore, GAN has a significantly higher IS score than VAE in Table 7. However, it is noticeable that the training time for GAN is significantly longer than VAE and it may also suffer from mode collapse.

Dimension 10	MSE	IS score
VAE without KL loss	0.0107	6.1559
VAE with KL loss	0.0115	7.3307
GAN	N/A	8.2308

Table 7. MSE and IS score for VAE and GAN

Task 2.1 From numerical perspective, both cGAN and UNet autoencoder with MAE loss has achieved relatively low and similar MAE 0.0449 and 0.0458. However, from visual perspective, even though cGAN shows similar performance with MAE in the background color seen from Figure 25 to Figure 28, it outperforms MAE in terms of coloring the objects and its details. Unlike MAE involves no learning process, cGAN takes black and white images as conditioned input and trains the generator to transform from this domain to RGB image domain. The adversarial structure allows discriminator to update the generator with information it has evaluated on the generated and real images and therefore the generator will learn and optimize its weight. Furthermore, the additional skip connections between the generator and the discriminator enable the discriminator to feed more information to the generator and thus producing more accurate RGB images.

6. Reinforcement Learning

Task 1.1 Seen in Figure 10, all agents have displayed a "Z" shape. This is because the agents are still learning and exploring at the beginning thus the average reward is flat and fluctuate in early epochs. Then the average reward drastically increases and then converges to an optimal value. The epsilon approach has a faster convergence speed because it is deterministic and only focuses on selecting the action with the highest Q value. Softmax on the other hand is a more stochastic approach with more variety in selecting actions according to probability distribution and therefore requires more epochs to converge to optimum.

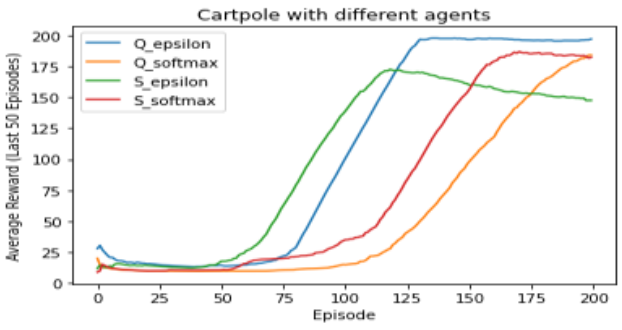

Figure 10. Cartpole with Q learning and SARSA

References

[1] J. Brown. Light on math-ml: An intuitive guide to understanding glove embeddings, 2021. towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010.

[2] S. R. K. He, X. Zhang and J. Sun. "deep residual learning for image recognition", 2016. arXiv:1608.06993 [cs.CV].

[3] M. Kumar. "intuitively understanding variational autoencoders," towards data science, 2020. Available: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf.

[4] K. . Kunin. Initializing neural networks, 2019. deeplearning.ai.

[5] S. Paul. Cnn transfer learning & fine-tuning," towards data science, 2019. Available: https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2.

7. Appendix

RL Task 1.1. The softmax policy is implemented by first estimating the action value using model.predict and then passing this value to the given softmax function to compute the softmax value and finally resampls the next action given the probability distribution calculated by softmax. SARSA on the other hand is on-policy which updates the current Q-value with respect to the next state and action chosen by the given policy. Therefore, I fist calculate the next action using model.predict and then compute the Q-values for the next action and state $Q(s_{t+1}, a_{t+1})$. Then, I update the target Q-value with the SARSA rule and use this to update the current Q-value given the equation $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$

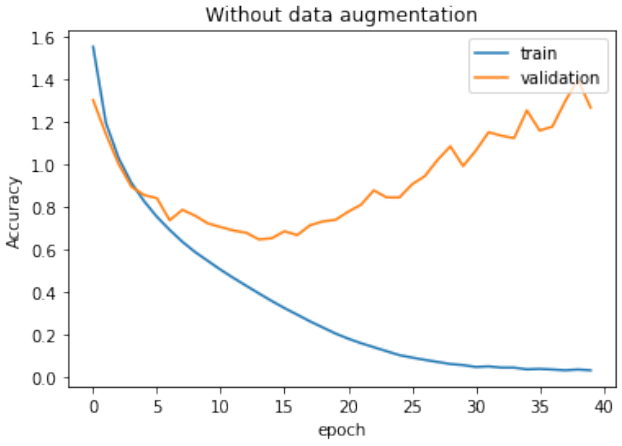

Figure 11. Loss curves without data augmentation

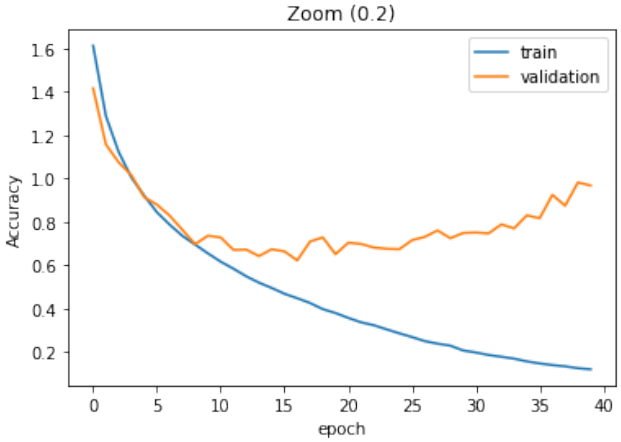

Figure 12. Data augmentation with Zoom (0.2)

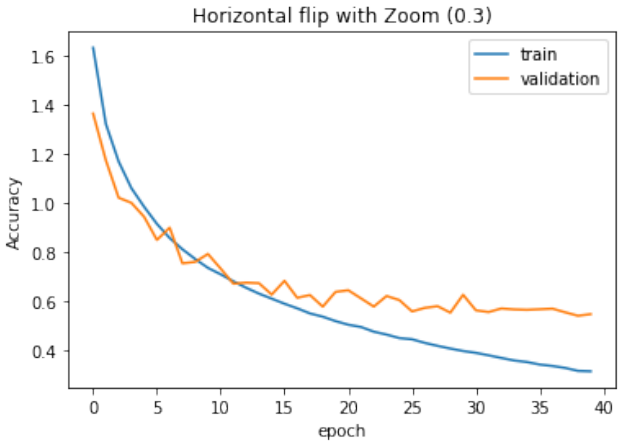

Figure 13. Horizontal flip with zoom (0.3)

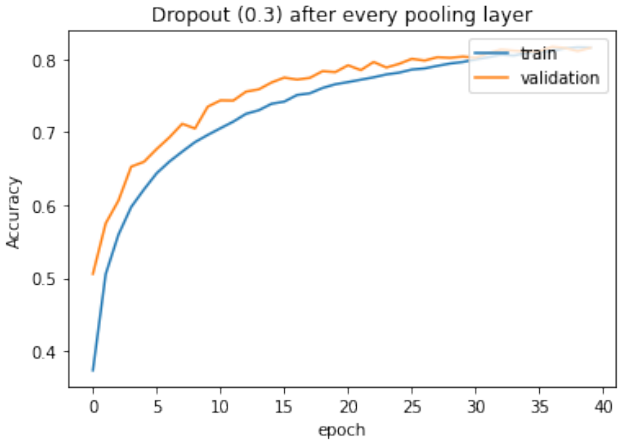

Figure 14. Dropout (0.3) between pooling and convolution layer

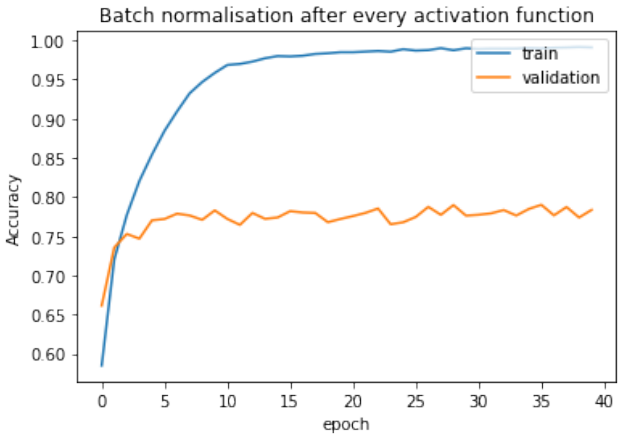

Figure 15. Batch normalization after every activation

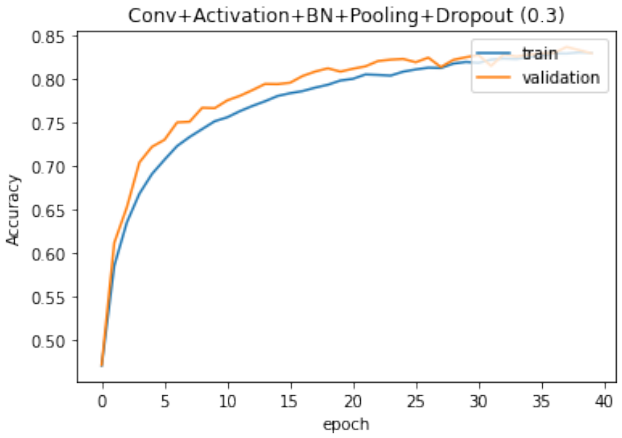

Figure 16. Conv+Activation+BN+Pooling+Dropout (0.3)

Window size	Train error (MSE)	Val error (MSE)
1	23.57	50.20
5	22.01	47.71
10	21.79	44.11
15	21.30	66.45

Table 8. Train and val error for different window sizes

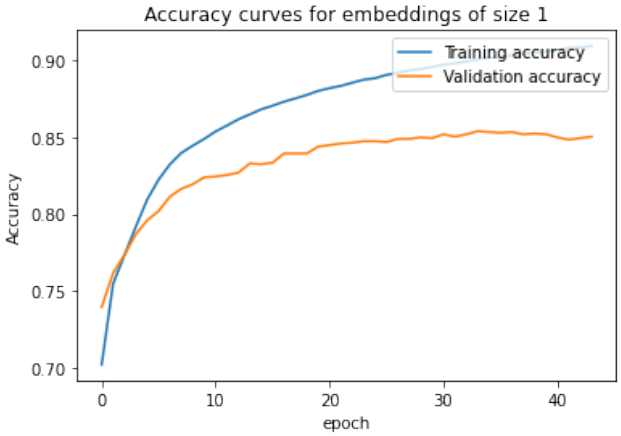

Figure 17. Accuracy curves for embeddings with dimension 1

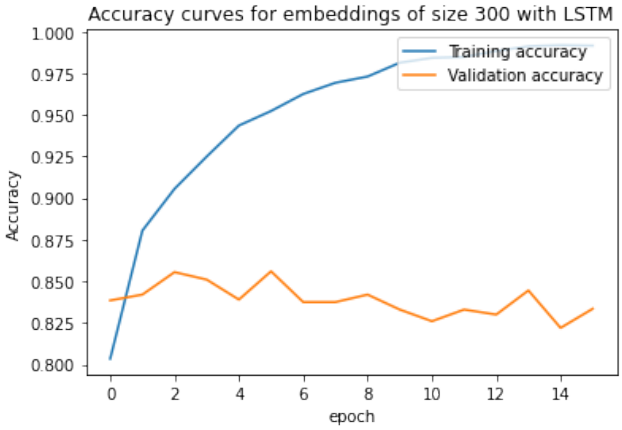

Figure 18. Accuracy curves for embeddings with LSTM

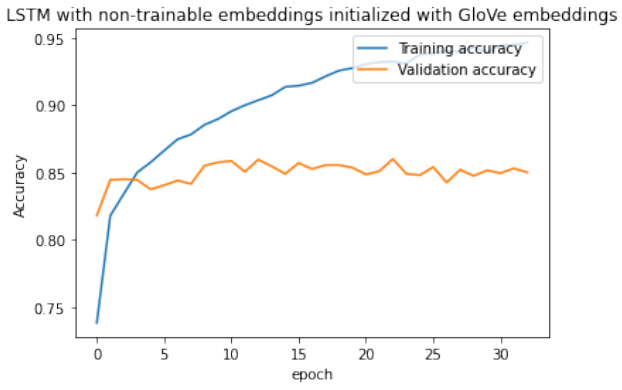

Figure 19. Accuracy curves for LSTM and GloVe embeddings

Most 10 similar words			
embedding 1		GloVe	
8	play	8	play
excellent	1950's	9	playing
highly	stunt	7	played
wonderful	nights	6	plays
perfect	tough	5	game
7	audiences	4	players
recommended	fallen	12	player
9	cuts	3	well
superb	titles	10	but
unexpected	costumes	16	going
today	about	13	starting

Table 9. Most 10 similar words for embedding 1 and GolVe

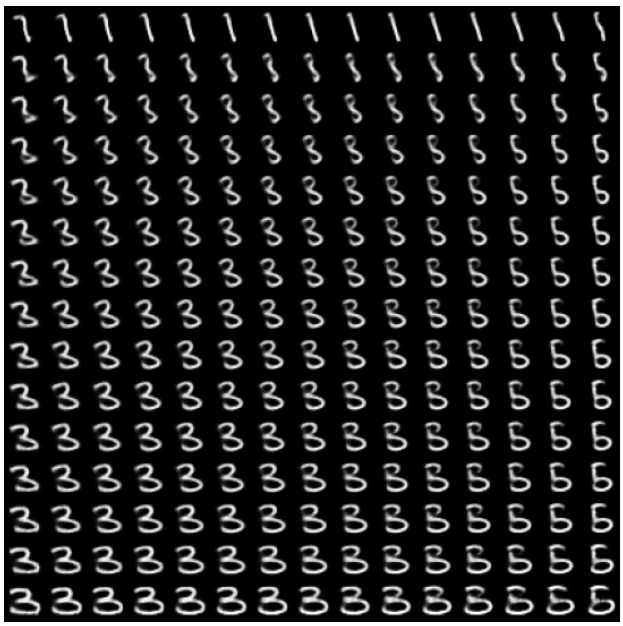

Figure 21. Data generated by VAE with KL loss

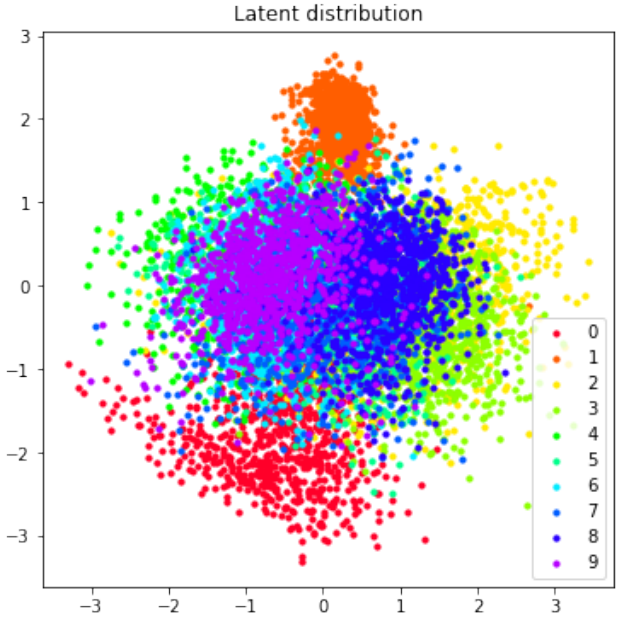

Figure 20. Distribution map of the encoded means for VAE with KL loss

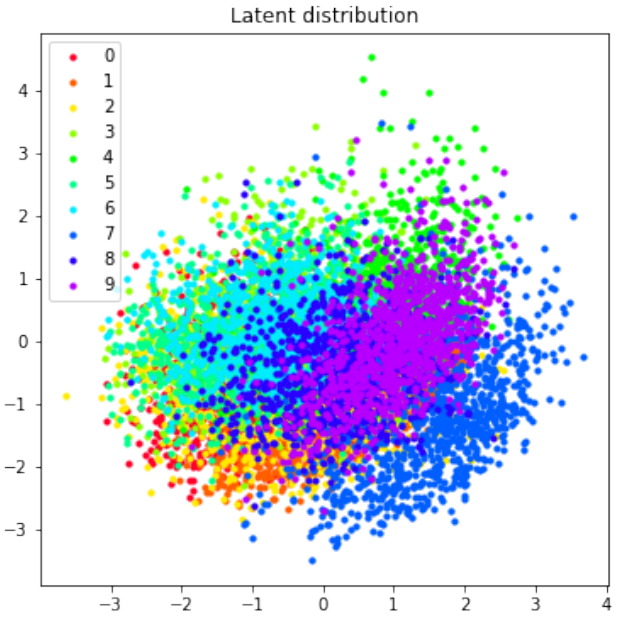

Figure 22. Distribution map of the encoded means for VAE without KL loss

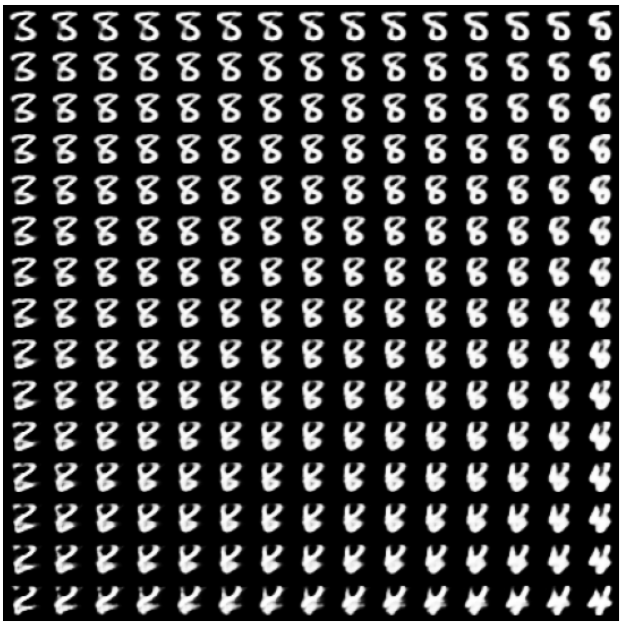

Figure 23. Data generated by VAE without KL loss

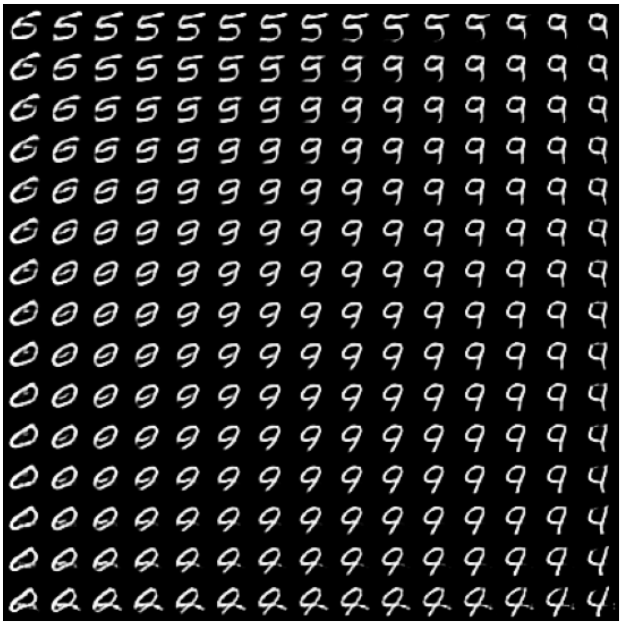

Figure 24. Data generated by GAN

Figure 25. Coloured images by MAE and cGAN compared to BW input and Real image 1

Figure 26. Coloured images by MAE and cGAN compared to BW input and Real image 2

Figure 27. Coloured images by MAE and cGAN compared to BW input and Real image 3

Figure 28. Coloured images by MAE and cGAN compared to BW input and Real image 4