Ata Yardimci, ay1517, 01356750

1. CNN Architectures

Task 1.1: VGG on Tiny-ImageNet: [13][15] In Figure 1, we observe that training from scratch or with pre-trained weights from 'imagenet' and not freezing any layers gives a very oscillatory accuracy behaviour as there are many parameters to train. However, near to the end of training, they both converge to a similar accuracy value of about 60%.

Fine-tuning the weights by freezing all layers but the dense layers gave us a smoother and steadier response which is due to having much less trainable parameters. While it converged to an accuracy worse but close to the ones we got from training all the parameters, it also yielded less loss at the end of the training. (See Appendix A) In Table 1, we can see that fine-tuning the model also had less than half the training time per epoch and converged in less epochs compared to previous training methods.

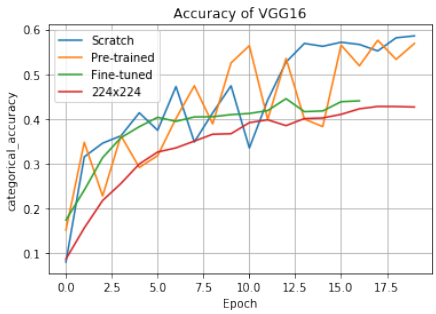


Figure 1. Accuracy of VGG16 with its weights trained with different methods (Loss in Appendix A as well as the plots for VGG19)

When we fine-tuned the model with an input image of size 64x64, we had about 40 million weights of which 26 million were trainable and made up the the last three dense layers. However, when we used an image of size 224x224, we only had to fine-tune the last of the dense layers which had 200 neurons and 820,000 trainable weights. However, total number of parameters this model has is now 135 million and even though most aren't trainable, this large number causes back-propagation to happen very slowly. The loss and accuracy graphs were smoother when we used a larger image size but converged to similar values as the fine-tuned model for the smaller image size.

Table 1 gives the accuracies of different models and their training times. Best results in terms of accuracy and training time for VGG16 and VGG19 are highlighted in green.

VGG16 [13]	Test Acc	Time (min)	Epochs	t/epoch (s)
scratch	58.0%	21.5	20	64
pre-trained	56.3%	21.4	20	64
fine-tuned	44.1%	7.60	17	27
224x224	42.2%	72.3	20	217

VGG19 [13]	Test Acc	Time (min)	Epochs	t/epoch (s)
scratch	54.8%	25.8	20	77
pre-trained	55.5%	25.8	20	78
fine-tuned	40.7%	10.4	19	33
224x224	40.7%	85.8	20	257

Table 1. Accuracies and inference times for different VGG models

2. Recurrent Neural Networks

Task 2.1: RNN Regression: In figure 2, we compare the prediction plots of different models that are trained with a range of window sizes against the true plot of the number of passengers on a plane in a given month. [6]

As the window size increases, number of training and test samples decreases, and the model waits for a longer sequence before making a prediction. Thus, plots start later in time for larger windows. We also see that predictions for window sizes 1, 5 and 10 are closer to the true value than the ones for window sizes 20 and 30. (See also test errors for different window sizes in Appendix B) One other observation to note is that as the window size increases, the plots seem to flatten.

The best performance we have seems to be for a window size of 10 which could mean that the model for this particular data moreso depends on about last ten samples.

Task 2.2: Text Embeddings Importance: In figure 3, we have accuracy plots of models that use different embedding methods to classify the IMDB Sentiment Dataset.[9] We observe that, all three methods experimented steadily converge to about 100% accuracy in training. However, the observed validation accuracy behaviour is slightly different.

When the model is trained with trainable embeddings initialized at random, the validation accuracy goes down in an oscillatory way. When they are initialized with GloVe

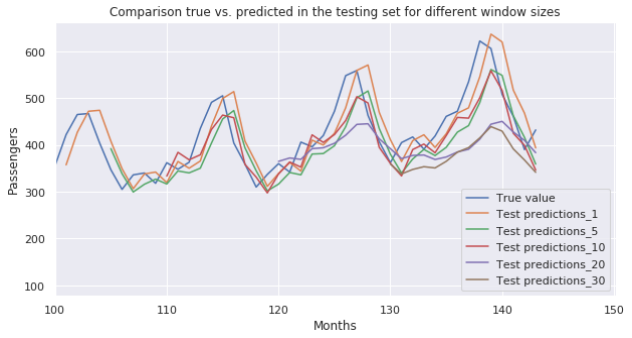


Figure 2. Test prediction curves of models trained with different window sizes

embeddings, [11] the plot starts at a higher percentage but quickly over-fits and goes down. Overfitting is a result of having too many free parameters to train. When the model is trained with non-trainable embeddings initialized with GloVe, the plot steadily increases from a lower level and stays at a higher percentage at the end of training; less trainable parameters prevent over-fitting in this case. The validation plots show the importance of meaningful embeddings.

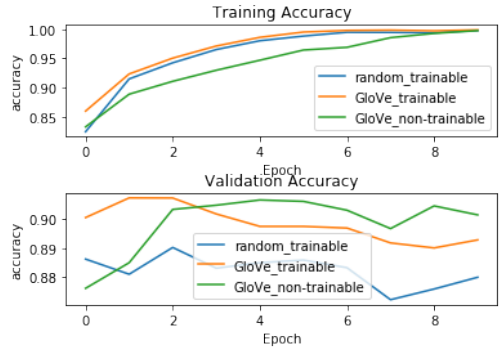


Figure 3. Training and validation accuracy curves over 10 epochs for different embedding methods

Ten closest words to "action" according to different embeddings we use are reported in Table 2. When the training was repeated, none of the closest words changed for embedding 3 as expected since it wasn't trained during the training process. Some words changed for the second embedding and all changed for the first. We also observe that while some of the words are same for embeddings 2 and 3 like "actions", "taken" and "moves", all of the words similar to "action" in embedding 1 are different, the reason being that it's too unpredictable with random initialization.

Task 2.3: Text Generation: In Figure 4, we have the BLEU scores for different temperature values in the task of generating text resembling the scripts of Game of Thrones.

1	2	3
random_trainable	**GloVe_trainable**	**GloVe_non-trainable**
pursuit	actions	actions
white	moves	taken
more	which	take
adult	similar	result
stranger	taken	taking
cruel	brought	moves
fights	calling	instead
persons	response	which
before	rather	but
getting	director's	yet

Table 2. Closest ten words to "action" for different embeddings

BLEU is usually used in more constrainted tasks unlike text generation where there are many grammatically and semantically correct possibilities.[10] However, in Figure 4, we do see a correlation between BLEU and the temperature values. The score is at its highest when the temperature is about 0.2-0.3 and drops steadily when it moves away from these values. This suggests a low level of variability in prediction generates writing more similar to the scripts.

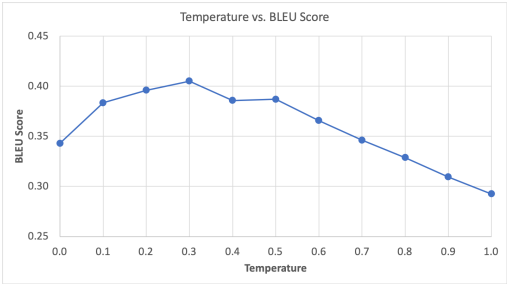


Figure 4. BLEU score for different temperature values

The generated sentences (Table 3) are grammatically correct when T (temperature) is 0 or 0.5 but don't make much sense in context. When T is 1, the variability is too high and some of the words are even meaningless.

T	Generated Sentence
0.0	"d how many missing me to the throne room. TYRION: I don't know what you want to stay to the great ma"
0.5	"d leading in the North. He took me as well. QHORIN: The Lord of Light was everything those gods ride"
0.5	"d we're standing from the men we'll have a good man. RAMSAY: You wanted to leave the little brother."
0.5	"d he was best he sees her baby. SAM: I don't care before you killed a fair in the North and your pri"
1.0	"d the Watch dintnis guarding on Varys. MARGALOR: I would happen to hurt middler enough for everywies"
1.0	"d I was never would see up you east. TOMMEN: You held to rush the gate. SELYSE: We've betraged her n"
1.0	"yway. The reshy INT. A COMP [EDRISERNESS A sentant says he pulls them. RICKARD KARSTARK: He's a murd"

Table 3. Generated text with different temperature values

3. Auto-encoders

Task 3.1: Non-linear Transformations for Representation Learning: For the linear auto-encoder defined in the tutorial which had two neurons in its representation layer, we had an MSE around 0.0557 for the reconstructed images of the MNIST data set.[8] In Figure 5, we try to give the representation more capacity by adding non-linearities such as ReLU and Sigmoid activations and increase the number of neurons in the representation layer.

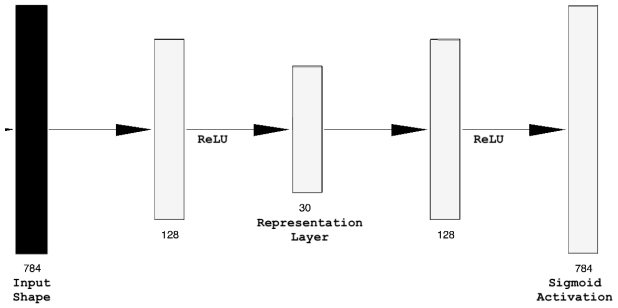


Figure 5. Architecture for Representation Learning for the MNIST data set [8] with MSE = 0.0082

In Figure 6, we have four examples of the reconstructed images. On the left are the actual test images, in the middle the reconstructed images using the non-linear model in Figure 5, and on the right are the reconstructed images using the linear model. We validate the low MSE of the non-linear model against the linear model and observe that using non-linear activations and increasing the dimensionality of the representation indeed results in better capacity and results.

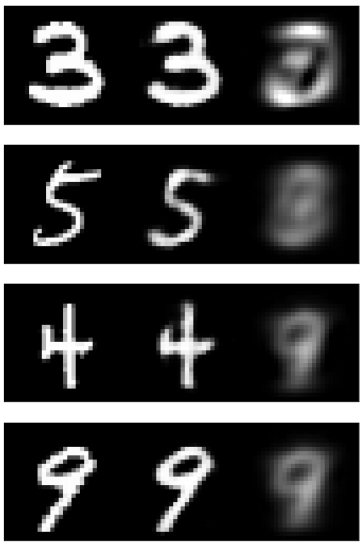


Figure 6. Test images vs. Reconstructed images of the Non-linear Model vs. Reconstructed images of the Linear Model

Features are plotted in the representation space using a PCA approach in Figure 7. Although the representation space has a dimensionality of 30, this approach gives us a good way to visualize the clusters. Then in clustering the data points in this dimension, we get an accuracy of 53.2%.

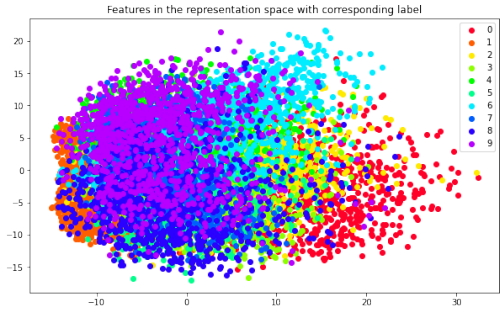


Figure 7. T3.1. Feature distribution.....

Task 3.2: Custom Loss Functions From Table 4, we observe that models trained with Mean Absolute Error and 1/PSNR as loss functions perform better quantitatively as they're more robust to outliers. We can support this claim qualitatively by checking the reconstructed images in Figure 8. In the figure, we also observe that although MSE and SSIM trained models have similar quantitative behavior, SSIM which is a perception-based model that considers structural information reconstructs a more noiseless image. L0-norm wasn't used since it wasn't differentiable.

Loss Function	MSE
MSE	0.00199
MAE	0.00161
1/PSNR[1]	0.00160
SSIM[14]	0.00205

Table 4. MSE of models trained with different loss functions

Figure 8. **True Image** - **MSE** - **MAE** - **1/PSNR** - **SSIM** Denoised images (in that order)

4. Variational Autoencoders and Generative Adversarial Networks

Task 4.1: MNIST generation using VAE and GAN In Table 5, we have the Inception Scores [2] and Mean absolute errors for different generative models. We observe an improvement in the measures of the VAE model when the latent dimensionality is increased from 2 to 20 as this gives the network more capacity for more accurate and diverse images. After adding an extra layer both to the encoder and the decoder, we observe an improvement in the IS but not in the MAE. For this reason, we argue that the extra layers provide the network with more diversity than accuracy.

The GAN model with an initial random sample dimension of 10 (randomDim=10) performs the best out of the experimented models in Table 5, and the one with randomDim=1 performs the worst. While the generated samples of the GAN model with randomDim=10 were quite diverse and accurate, the other GAN model lacked diversity and generated less accurate images. By making the initial sample of dimension 1, we cause the the uniformity of $p(y)$ to be disrupted, and this in turn yields a low IS.

Model	Inception Score	MAE
Baseline VAE	4.6367	0.1022
VAE with latent_dim=20	5.9133	0.0413
VAE with extra layers	6.9296	0.0437
GAN with randomDim=10	8.2804	-
GAN with randomDim=1	2.9127	-

Table 5. Measures of VAEs and GANs with different parameters

Task 4.2: Quantitative VS Qualitative Results Here we compare a Conditional GAN network (cGAN) [5] against a UNet [12] Autoencoder (MAE) trained with MAE loss in the task of coloring B&W images. While cGAN model yields a mean absolute error of 0.0458, the MAE model yields an error of 0.0449 when they're evaluated with CIFAR10 [7] test dataset. Although their performances are quantitatively close (in "mae" sense), in Figure 9 we can see that cGAN model generates more realistic images.

GANs do not sample the closest point or the best fit to the data set but rather minimize the overall distance between the real and generated distribution, which yields better results.[4] Colors don't align with the real image for the cGAN images, which may cause a higher "mae" loss, but the generated image is sharper and more realistic as training with an adversarial loss is better in image generation.

5. Deep Reinforcement Learning

Task 5.1: On-policy vs. Off-policy We train a RL Agent to play the CartPole[3] game using Q Learning (off-policy) and SARSA (on-policy) with ϵ-greedy and softmax policy.

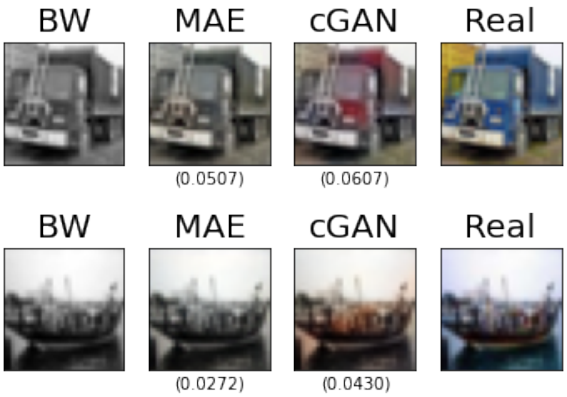


Figure 9. Colored images generated by MAE and cGAN models with the MAE errors given below the generated images

We observe in Figure 10 that although the performances of Q Learning and SARSA don't vary much from each other, learning with ϵ-greedy policy is better in overall than learning with softmax policy for this particular task. We also note that each method has a local maxima around training episodes 150-200, and SARSA with softmax policy has a relatively larger variance in its performance.

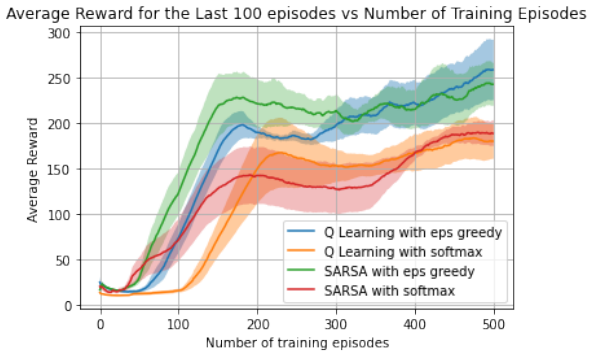


Figure 10. Average reward for the last 100 episodes for different learning methods (Means of multiple realizations with their standard deviations as shades are plotted for better representation)

6. Conclusions

This paper discusses several topics in Deep Learning: investigating CNN Architectures by fine-tuning their weights, RNN Networks for regression and text generation, importance of text embeddings, Auto-encoders for representation learning, custom loss functions, VAEs and GANs for image generation, and different learning methods in Deep Reinforcement Learning. The paper provides an overview on each topic, supporting its claims with empirical results. In the future the scope of the investigation can be extended to different architectures and methods for eclectic tasks.

References

[1] Peak signal-to-noise ratio.

[2] S. Barratt and R. Sharma. A note on the inception score, 2018.

[3] A. Barto, R. Sutton, and C. Anderson. Cartpole-v1, 1983.

[4] G. Delétang. How gans really work, 2019.

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

[6] jbrownlee. airline passengers.

[7] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset and the cifar-100 dataset.

[8] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database.

[9] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[10] K. Mikolajczyk and C. Caliberto. Ee3-25: Deep learning, 2020.

[11] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation, 2014.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

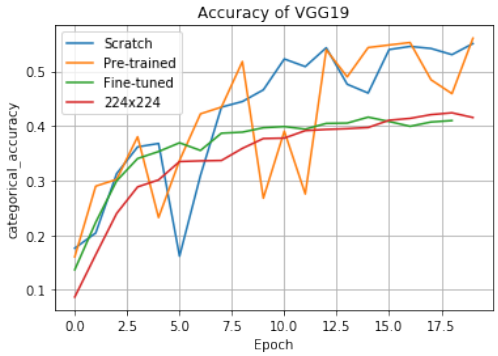
[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

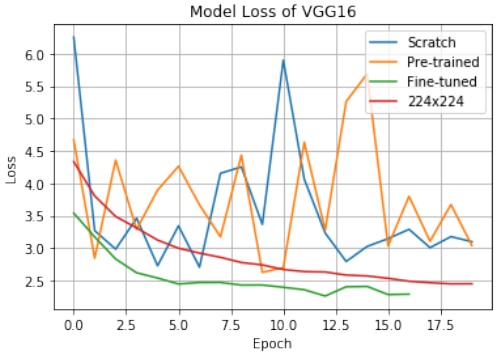
[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. The ssim index for image quality assessment, 2003.

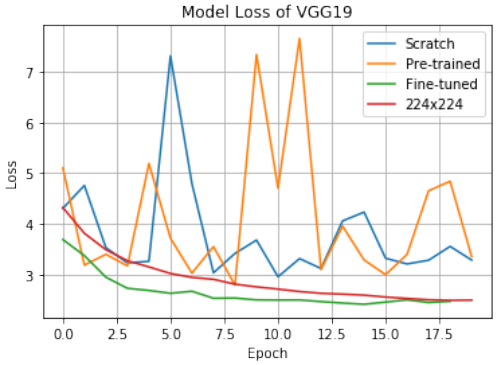
[15] J. Wu, Q. Zhang, and G. Xu. Tiny imagenet challenge.

7. Appendix

A. Accuracy of VGG19 and Loss of Different VGG Models







B. MSE errors for different models with different window sizes

Window Size	Test Error	Training Error
1	51.29	23.56
5	54.76	25.79
10	47.46	20.69
20	80.89	19.75
30	100.88	19.75

Table 6. Error vs Window Size