Athanasios Liaskas, al3415, 01079253

1. Introduction to Convolutional Neural Networks Architectures

Task 1.1. In this tutorial we compare the performance of various architecture configurations. We study the VGG16, VGG19 architectures and their performances under various conditions. We start with the VGG16 architecture with an image size of (64x64) and train it from scratch (model #1). Then we train the full architecture again but we initialise it with pre-trained weights from ImageNet (model #2). Moreover, we report results when fine-tuning ImageNet weights by freezing all but the last dense layers and retraining the model (model #3). The goal of this experiment, is to determine whether loading the last dense layers with weights from Imagenet will results in a superior performance of the network, or whether allowing a higher degree of freedom to those layers will result in greater performance.

Through using an early stopping callback, we reduce the training time, and we report the best accuracy on the validation set for each model and not necessarily the final one. This is visible when looking at Figure 1, where for instance the accuracy of model #2 peaks during epoch #3 and then decreases. We use the early stopping callback for model #2 to report its highest accuracy. A similar procedure is carried out for the other models that also reach maximum accuracy in only a couple of epochs, and the best scores are reported in Table 1.

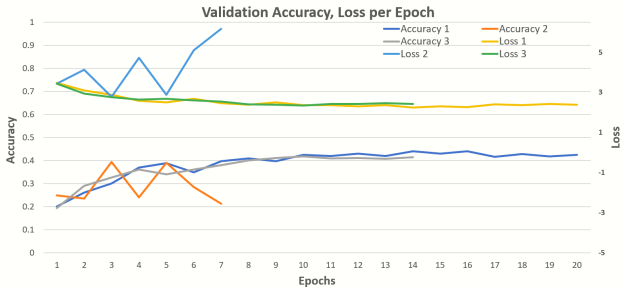

Figure 1. Comparison of loss and accuracy per epoch, for models #1, #2, #3, through a training cycle of 20 epochs.

We also want to investigate the effect the input image size has on the performance of the network. We train the VGG models that also do fine-tune (a.k.a. freezing) again, but now using input images of size 224x224. As the input size is 224x224, we preserve all dense layers but the very last one. We can easily compare the 64x64 and 224x224 (VGG16,19) models that use ImageNet and fine-tuning models since the only difference among them is the image size. As we can see in Table 1, the VGG models that have an image size of 224x224, perform on average slightly better than those with 64x64 images. That is an expected result since the default input size of the VGG networks is indeed 224x224. However, the improvement in performance is marginal, which could be expected since the 224x224 images that we provide as input are not truly of that resolution but are simply resized images up-scaled through our *lambda* layer, so no extra information is contained in the 224x224 images when compared to the 64x64 ones.

We compute the *inference time* by calculating the amount of time taken to for each network to process one image so that it is able to infer a result (i.e. make a decision about it). We report the time taken for each network to process an image in the *testing step* (i.e. time taken for one sample of input). It is clear from the results in Table 1 , that models that include the fine-tuning process are significantly slower than those that either are initialized with weights from scratch or from ImageNet. Moreover, we can see that those faster models outperform the in regards to accuracy.

Model	Weights	Image	Accuracy (%)	Time
VGG19	IN	64x64	53.79	0.346
VGG16	scratch	64x64	42.13	0.335
VGG19	IN-freezing	224x224	42.05	3
VGG16	IN-freezing	224x224	41.95	2
VGG19	IN-freezing	64x64	41.61	3
VGG16	IN-freezing	64x64	40.67	2

Table 1. Best performing VGG16, VGG19 architectures, ranked by accuracy. The *pre-trained* weights are from the ImageNet (IN) dataset. The time column signifies the *Inference Time* taken for each network and is in units of milliseconds (ms).

2. Recurrent Neural Networks (RNN)

Task 2.1. In this task we studied the impact of the *window_size* variable. We trained our model we report the predictions for the number of passengers for the months contained in the unseen test data. We experimented with a large

range of window sizes as can be seen in appendix Figure 9. We chose the best performing window size values with the least MSE to plot in Figure 2. The best performing window size was that of size 10 with a MSE of 39.9%.

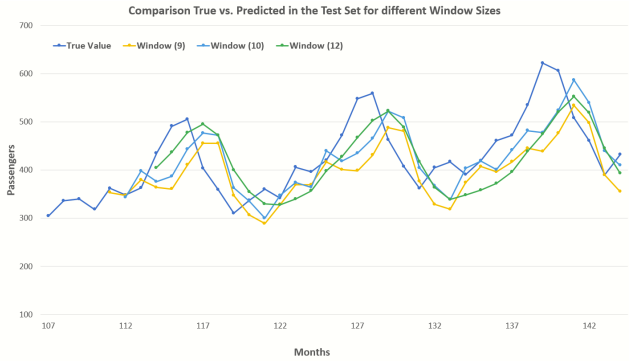

Figure 2. Test curves for different models for different window size values.

One side-effect of choosing a large window size is that you will have predictions for less months, since the RNN will consume more historical information per prediction it makes. This can be seen in Figure 2 since for the first 9, 10, 12 months we have no predictions. However, for a bigger test dataset this would not be a significant and having a lower MSE would be our priority.

Task 2.2. In this task we studied the importance of **embeddings**. We experimented with three different models: Model #1 had randomly initialised but trainable embeddings and the other two (#2, #3) had embeddings initialised with GloVe that were trainable, and non-trainable (frozen) respectively. Figure 3 contains the training, validation accuracy curves. Paying close attention to Figure 3 and specifically at the validation set accuracy curves, we can see that for the first model we need to do an early callback to save the best performing model, which occurs after epoch number one, since after that we see a drop in the validation accuracy. Similarly, we repeat this for the other two cases. For the first two models, which have trainable embeddings, we can observe the training process taking place since the training accuracy progressively increases, contrary to the third model (right of Figure 3) with non-trainable GloVe embeddings which remains stable in accuracy. As the training accuracy increases the two models, over-fit the training dataset which is why they then perform worse in the validation set in terms of accuracy.

The words from the Glove trained embeddings are much more concise and concentrated in a small neighborhood in the embeddings space. This can inferred from the fact that for example in model #3 words “taken”, “take” are the top 2 and 3 words while the word “actions” is the top word for both models #2, #3. Conversely, the non-trained model’s words are fairly accurate but are much more sparsely placed among the embeddings space. For example, words ”every”, ”films” (e.g. “actions films”) are associated with the word “action“ correctly, but if we take the distance between every, and films we can assume it will much larger than that between “actions’ and “taken”.

	Model #1	Model #2	Model #3
word #1	every	actions	actions
word #2	films	calling	taken
word #3	joseph	response	take

Table 2. The closest words to the word “*action*” that were generated from the three models.

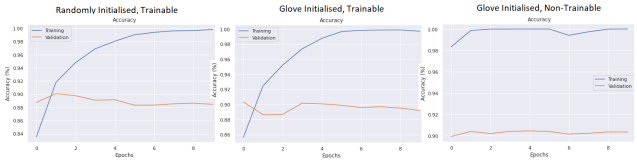

Figure 3. Training and validation accuracy curves over 10 epochs, where validation set is the test set.

Task 2.3. The scope of this task is to further delve into the relationship between the ***BLEU*** score metric and the ***temperature*** used within our RNN when we use it to generate text. Figure 4 outlines this relationship that we were able to plot by experimenting with the temperature values and observing the resulting BLEU score. Observing Figure 4 more closely we attempted to fit a linear and a quadratic line through the data to help us identify this relationship between the two variables. As is evident from the figure the linear trendline outlines an almost indirectly proportional relationship since the higher the temperature the lower the BLEU score. The quadratic trendline yields a much more closely matched fit to the data hinting a potential relationship between the temperature and the Square of the BLEU score. The BLEU score peaks, at around a temperature equal to 0.3.

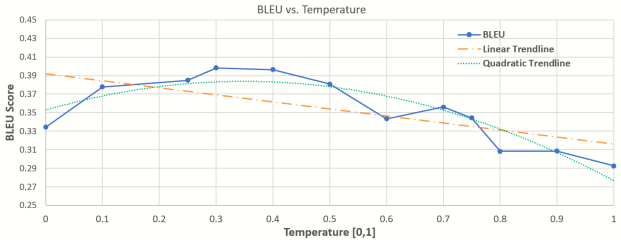

Figure 4. BLEU score variation for different temperature values, between [0,1].

Temperature	Generated Sentences
T = 0	- d how much are some searching and swear the street for the street for the stage. - TYRION: I don't kno
T = 0.5	- d my lady from the house. - SAM: Well, my lady. You had a lad soot. - SAM: I was a knight. - THEON: He ne
T = 1	- d your visrt with blood inside, I won't meading the mountes are time that he can see your parryiring

Table 3. The text sentences generated for various levels of Temperature

The input sequence is in the form of a question and ends with the characters "an" which is why in all generated sentences, the first predicted character is the most probable one i.e. "d" given how popular the word "and" is. Given the results from Figure 4 we would expect a lower Temperature value to produce the best sentence. Even though, all sentences follow to some extent the style of the script, indeed T=0.5 produces a semantically correct sentence that finishes the question of the input sentence with a sort and quick sentence, although makes a syntactical mistake since it has the same character performing dialogue twice and in adjacent sentences, instead of simply joining the two sentences. For large T values as we expected from Figure 4 the sentences are grammatically, syntactically and semantically incorrect as they carry little meaning.

3. Autoencoders

Task T3.1. In this task once again, use the MNIST test sets validation data. We then, compare the linear model from the tutorial, with our non-linear architecture model. The linear model is a simple linear Regression model, hence has limited power and does not perform satisfactorily. This is reflected in its MSE of "0.0544". The MSE of the of the linear model might seem small at first glance, but when compared to the MSE of "0.0094" that we achieve with our non-linear architecture we can see that it is almost one order of magnitude better. This is reflected in the quality of the images that the two models are able to reconstruct.

The architecture of our non-linear model outlined in Figure 4 was chosen experimentally. Namely, by varying the dimensionality and structure of layers in our model and measuring the MSE after every subsequent change. When developing the architecture the first step was to add *two extra fully-connected layers*, one before and one after the representation layer. This resulted in a slight improvement of the MSE. Adding more layers than that did not yield any significant improvement. Changing the activation functions to leaky-ReLu yielded a similar but slightly worsened MSE so that was abandoned. Finally, by increasing the dimensionality hyper-parameter we were able to progressively improve our MSE, finally choosing the upper bound *dimensionality of 50* which performed the best out achieving the smallest MSE.

Dense Layer	Description	
	Activation type	#Filters
Activation	ReLu	128
Activation	ReLu	30
Represtation	-	50
Activation	ReLu	128
Activation	Sigmoid	784

Table 4. Best performing architecture trained over 10 epochs, with *MSE* loss, and *adam* optimiser.

The images seen in Figure 5 are those reconstructed by our non-linear model of MSE="0.0094". Comparing them to the images reconstructed by the linear model as seen in appendix Figure 11, we can see that the improvement in the MSE is reflected in the our model's ability to reconstruct the images. The images from the non-linear model are far superior in resolution clarity as well as in terms of their definition of the various edges of the digits, when compared to the faintly visible digit images of the linear model.

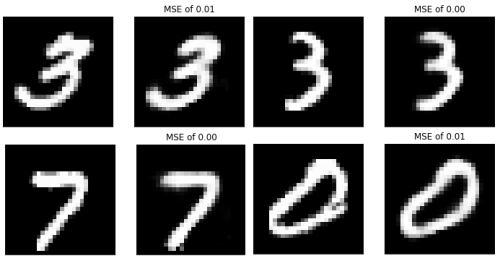

Figure 5. Reconstructed images

We then plotted the representations in the feature space, and through firstly using a the Principal Component Analysis (PCA) approach mapped them to a 2D space, which is important given that the features are of dimensionality of 50. As is evident in Figure 6, by simply relying on the PCA we cannot get good clustering of the data, since we can see a lot of overlapping of the different classes. This is due to the large dimensionality of our feature space as well as due to the non-linear nature of our model. However, given the low MSE we are confident that in the large dimension environment of the feature space the data will be clearly clustered, justifying the representation's good performance in reconstructing the images. Looking at how other architectures (that performed worse in terms of MSE) clustered the data after PCA we can see that simpler architectures created clusters that are more clearly defined, as seen in appendix Figure 13 but reconstructed images with inferior quality.

By then applying the majority method as seen in the right

of Figure 6 we can see a clearer clustering with an accuracy of 0.592. Observing the figure in more detail we can see hints of the clearly defined clustering when we transform back to the higher dimensional feature space.

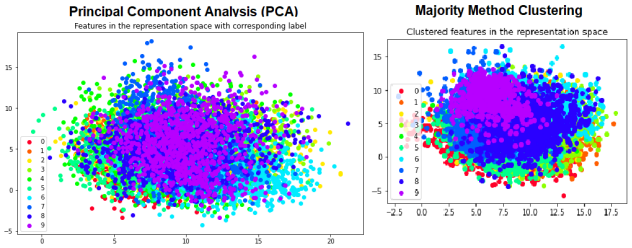

Figure 6. Figures depicting the Feature distribution when plotted in the Feature Space. Majority Method Clustering applied on the feature space in the right image.

Task T3.2. In this task we attempted to use three different loss function, to train our model to denoise images. We report in table Table 5 the MSE-metric results that we were able to yield with each Loss Function.

Loss Function	MSE (%)
SSIM	0.1975
MS-SSIM	0.1357
MSE/MAE	0.2589

Table 5. MSE metric for each of the three different Loss Functions.

We also report some examples of denoised images when using each of the Loss Functions in Figure 7.

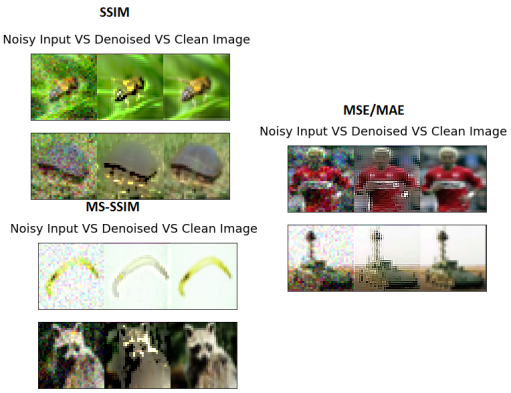

Figure 7. Denoised images

4. VAE-GAN

Task T4.1. Initially we start by comparing the *Inception Score (IS)*, *MAE* of the three different *VAE* models. Increasing the dimensionality for VAE #2 yields a much better IS as seen in Table 6. Adding the extra layer for VAE #3 yields only a minor improvement in terms of both IS and MAE.

The GAN #1 far outperforms the VAE by achieving a much higher IS, however when we then decrease its dimensionality to just "1" for GAN #2 there is massive drop in performance, performing worse than every other model.

Figure 8. Images produced by the cGAN and MAE models.

VAE model	Inception Score	MAE
#1	4.5902	0.3318
#2	5.7947	0.1990
#3	5.9600	0.1438
GAN model	Inception Score	-
#1	7.5375	-
#2	2.6625	-

Table 6. VAE and GAN Results in terms of (IS), MAE

Task T4.2. By taking a detailed look on the images of Figure 8, cGAN seems much more realistic and could easily fool a discriminator that is simulating how human viewers would evaluate the same images. Assuming we supply to a discriminator the Black and White image (so as the discriminator is not aware of the true colours of the image) it could easily take the images of cGAN to be real images since they are colored in a way that appears very real-life like even though in reality the coloring does not correspond to the actual colors of the *Real* image. The MSE for the cGAN is similar but slightly better than the MAE on average so for the majority of cases the cGAN performs slightly better than the MAE model.

5. Conclusions

Through this holistic overview of a large selection of methods for various deep-learning tasks, we determined in Tutorials 1,2 through a comparison of different weight and embeddings initialization techniques for our CNN, RNN respectively, that pre-trained weights, embeddings generally performed better as one would expect, given the vast amount of training and fine-tuning which they have undergone. Through Tutorials 3,4 we saw how we are able to extract meaningful features out of our dataset even without supervision that can then be used to generate new data.

References

6. Appendix

Additional results and discussions are included here.

Task 6.1 Figure 9 Deviating slightly from the best performing window size results in big changes in the MSE. Attempting to increase or decrease the window size by 3, 4 units from the best performing value of 10 exploded the MSE value. Moving a further 5 units to either direction near double the MSE, upwards of 75%.

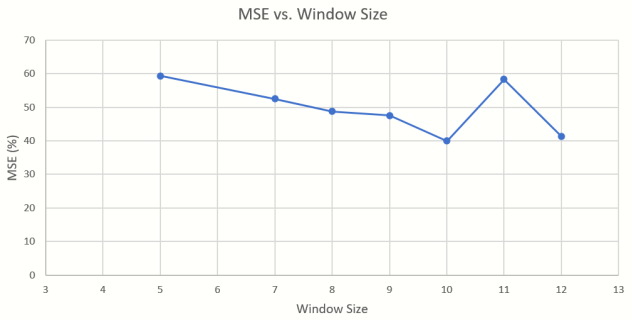

Figure 9. Line chart, depicting the variation in MSE as we change the Window Size.

Task 6.3 Figure 10 The generated text sequences exactly as they appeared in the console.

```
Temperature = 0
d how much are some searching and swear the street for the street for the stage.
TYRION: I don't kno
Temperature = 0.5
d my lady from the house.
SAM: Well, my lady. You have a lad soot.
SAM: I was a knight.
THEON: He ne
Temperature = 1
d your virst with blood inside, I won't meading the mountes are time that he can see your parryiring
```

Figure 10. Image, depicting the generated Game of Thrones text.

Task 7.1 Figure 11 Reconstructed images of the linear model as seen in Tutorial 7.

Best performing non-linear architecture as seen in terminal.

The PCA clustering achieved by a less complex architecture that performed worse in terms of MSE (MSE=0.469) but clusters the data more clearly in 2D space.

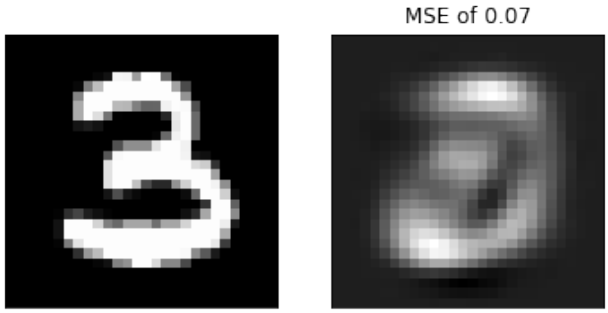

Figure 11. Linear model reconstructed-images.

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_90 (Dense)             (None, 128)               100480
_________________________________________________________________
dense_91 (Dense)             (None, 30)                3870
_________________________________________________________________
representation (Dense)       (None, 50)                1550
_________________________________________________________________
dense_92 (Dense)             (None, 128)               6528
_________________________________________________________________
dense_93 (Dense)             (None, 784)               101136
=================================================================
Total params: 213,564
Trainable params: 213,564
Non-trainable params: 0
```

Figure 12. Non-Linear model architecture.

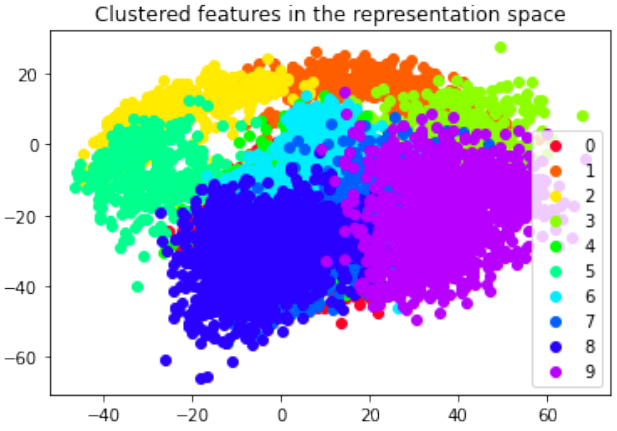

Figure 13. PCA clustering of simple images.