Assignment overview

The overarching goal of this assignment is to produce a research report in which you implement, analyse, and discuss various Neural Network techniques. You will be guided through the process of producing this report, which will provide you with experience in report writing that will be useful in any research project you might be involved in later in life.

All of your report, including code and Markdown/text, ***must*** be written up in ***this*** notebook. This is not typical for research, but is solely for the purpose of this assignment. Please make sure you change the title of this file so that XXXXXX is replaced by your candidate number. You can use code cells to write code to implement, train, test, and analyse your NNs, as well as to generate figures to plot data and the results of your experiments. You can use Markdown/text cells to describe and discuss the modelling choices you make, the methods you use, and the experiments you conduct. So that we can mark your reports with greater consistency, please ***do not***:

- rearrange the sequence of cells in this notebook.
- delete any cells, including the ones explaining what you need to do.

If you want to add more code cells, for example to help organise the figures you want to show, then please add them directly after the code cells that have already been provided.

Please provide verbose comments throughout your code so that it is easy for us to interpret what you are attempting to achieve with your code. Long comments are useful at the beginning of a block of code. Short comments, e.g. to explain the purpose of a new variable, or one of several steps in some analyses, are useful on every few lines of code, if not on every line. Please do not use the code cells for writing extensive sentences/paragraphs that should instead be in the Markdown/text cells.

---

keyboard_arrow_down

Abstract/Introduction (instructions) - 15 MARKS

Use the next Markdown/text cell to write a short introduction to your report. This should include:

- a brief description of the topic (image classification) and of the dataset being used (CIFAR10 dataset). (2 MARKS)
- a brief description of how the CIFAR10 dataset has aided the development of neural network techniques, with examples. (3 MARKS)
- a descriptive overview of what the goal of your report is, including what you investigated. (5 MARKS)
- a summary of your major findings. (3 MARKS)
- two or more relevant references. (2 MARKS)

Image classification is a classic computer vision problem, with the aim of identifying the contents of images. The CIFAR 10 dataset is a set of 60,000 labelled images with a size of 32x32 pixels; each image has 1 of 10 labels.

The CIFAR10 dataset has aided neural network development by being an easily accessible set of labelled images that allows anyone to use. Since 2010, the error rate of image recognition with the CIFAR10 dataset has dropped from 21.1% to 0.5%, meaning that CIFAR10 has played a large part in developing image recognition techniques.

This report aims to investigate several neural network techniques when applied to image classification, including: varying learning rates, and implementing dropout and transverse learning.

As the keras implementation of the Adam optimiser used in this report uses a default value. When investigating learning rates, values surrounding this default were used to see if any of these values would result in a higher performing network. When applying dropout, keras also has an implementation with a default value. Similar to the optimiser, values surrounding the default value were tested to see which would produce the most accurate network. The convolutional layers in the model when investigating trasverse learning used weights learnt when testing the base model. The model was then trained using the dropout rate with the best average performance and compared to a model trained without dropout.

It was found that the default learning rate was not the optimal for the model as it had the second highest accuracy and shared second highest loss. The scheduler created achieved a higher accuracy faster over no using one, the average accuracy achieved was also more consistent between trials.

A dropout rate of 0 game the worst performance out of all tested values but when using transfer learning, the networks with a dropout rate of 0 performed better.

// references

---

keyboard_arrow_down

Methodology (instructions) - 55 MARKS

Use the next cells in this Methodology section to describe and demonstrate the details of what you did, in practice, for your research. Cite at least two academic papers that support your model choices. The overarching prinicple of writing the Methodology is to ***provide sufficient details for someone to replicate your model and to reproduce your results, without having to resort to your code***. You must include at least these components in the Methodology:

- Data - Decribe the dataset, including how it is divided into training, validation, and test sets. Describe any pre-processing you perform on the data, and explain any advantages or disadvantages to your choice of pre-processing.

- Architecture - Describe the architecture of your model, including all relevant hyperparameters. The architecture must include 3 convolutional layers followed by two fully connected layers. Include a figure with labels to illustrate the architecture.
- Loss function - Describe the loss function(s) you are using, and explain any advantages or disadvantages there are with respect to the classification task.
- Optimiser - Describe the optimiser(s) you are using, including its hyperparameters, and explain any advantages or disadvantages there are to using that optimser.
- Experiments - Describe how you conducted each experiment, including any changes made to the baseline model that has already been described in the other Methodology sections. Explain the methods used for training the model and for assessing its performance on validation/test data.

Data (7 MARKS)

CIFAR10 is a set of 60,000 32x32 pixel colour images, each with one of 10 labels including modes of transport and animals. The array of 32x32x3 images was preprocessed by batching the images using the "xception" function in the keras "applications" library. This was largely due to simplicity but also increased the accuracy of the model.

The data for experiment 1 was split into 3 sections, two of which were added together, allowing the networks to train on 2/3 of the training data, with the last 1/3 for validation. For experiment 2, the data was split in half, with half of the data to train the networks and the other half to validate. This was to ensure that both halves of the experiment had the same amount of data to train on (the second half using the validation set from the first half to train and the training set to validate).

No other preprocessing was done as the model achieved approx. 70% accuracy without. If any more was to be added, padding the images would help preserve some of the positional information about the images, possibly increasing the accuracy.

Architecture (17 MARKS)

The network used consists of three sets of a convolutional layers (with biases initialised as zeros) combined with a max pooling layer, followed by one flattening layer, and two fully connected layers.

All convolutional layers use a 3x3 filter with a stride of 1 and no padding and the ReLU function. The small filter and stride value was chosen to preserve locational information about the image. If the either is too large, this data can be lost. No padding was needed as the stride of 1 meant that all image values are considered.

The first max pooling layer uses a 3x3 filter to downsample the data, while the others use a 2x2 filter. No padding is used when pooling. The stride value matched the pool size.

After the combination of the convolutional layers and max pooling layers, the data is a 3D array with shape 64x1x1. This is converted to a 64-element long vector for the fully connected layers to use by a flattening layer.

The first fully connected layers takes the result of the flattening layer and applies the "ReLU" activation function, outputting a 32 element vector. This is parsed to the final fully connected layer, applying the "softmax" activation function and returning a probability distribution for the 10 different labels. Both fully connected layers' biases initialise as zeros.

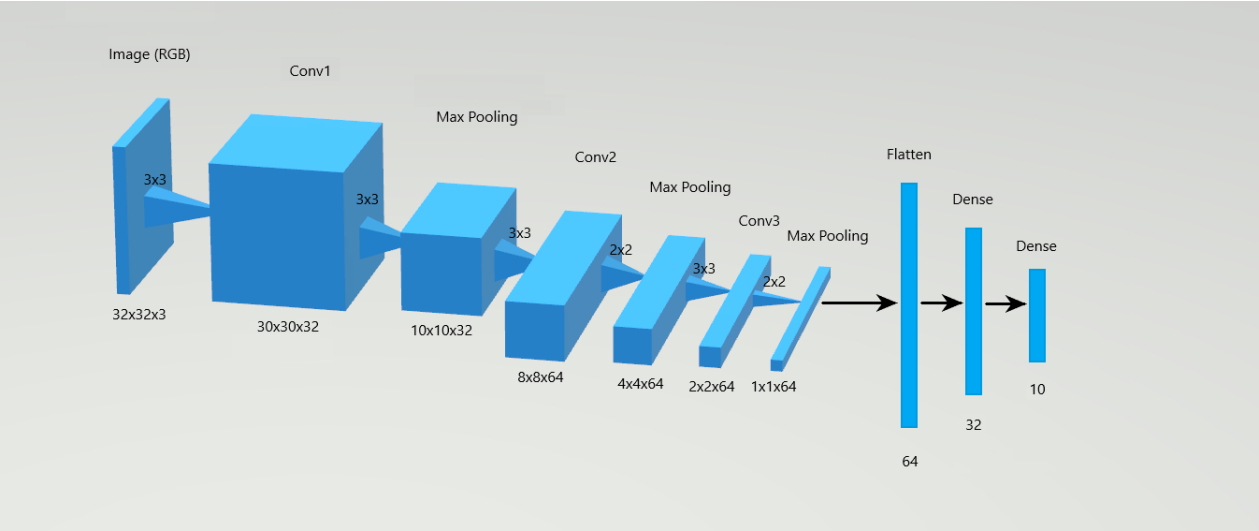

Figure 1 – The architecture of the model

Brackets around the conv and dense layers

Loss function (3 MARKS)

The "sparce categorical cross entropy" loss function is used when computing the cross-entropy loss between labels and predictions. Cross-entropy is used as it tries to minimise the negative log probability of the correct class, meaning that it is well suited for a multi-class classification task. * as opposed to [x] (reference) *

If the labels are stored as vectors (“[1, 0, 0],[0, 1, 0]…”), regular cross entropy is more fitting, but if the labels are stored as indices (“[0], [1], [2]...”), sparce cross-entropy is more fitting. The categorical variant is used because this is a multiclass problem. For the likelihood of a data item being a single class, non-categorical cross-entropy would be used.

Optimiser (4 MARKS)

// describe optimiser - optimised for X - using accuracy metric // reference "all tested optimisers"

The Adam optimiser uses stochastic gradient descent, allowing it to quickly converge on a moderately accurate answer. It was chosen because it is

"computationally efficient, [and] has little memory requirement" (Diederik P., et. al.). This means that Adam provides fast convergence speeds, so the number of epochs can be reduced, and more data can be gathered from a greater number of trials over the same interval.

The default learning rate for the keras implementation of Adam is 0.001, meaning that, where unspecified, a learning rate of 0.001 is used.

---

keyboard_arrow_down

Experiments

Experiment 1 (8 MARKS)

The default learning rate for the keras implementation of 'Adam' is 0.001 so values for potential learning rates were chosen around this value as there is a high likelihood that these values would not lead the network to become unstable. Values of "[0.0005, 0.001, 0.0015, 0.002, 0.0025]" were chosen. All other hyperparameters were kept to the default values with their keras implementation.

The learning rates and results arrays were created, and the number of epochs and trials were declared (20 and 5 respectively).

For each of the 5 trials of each learning rate, a new model was created using the architecture stated above to prevent any previous training skewing results.

The validation accuracy over each trial is recorded and an average computed, producing an average validation accuracy for every epoch for each given learning rate.

For the scheduled learning part of this experiment, the learning rate with the steepest initial loss gradient was chosen as the initial learning rate. The learning rate was changed twice to the rate found to have the lowest average loss increase to reduce overfitting.

Experiment 2 (8 MARKS)

Transfer learning uses the weights from a network trained on a different task to initialise the new network weights. This experiment aims to find whether dropout will positively affect the accuracy of a network using transfer learning by using weights previously trained a model.

The training data was split in half for training and validation. This experiment calls for the model to be trained using one half of the data to find how dropout rate affects performance and the other half to conduct transfer learning. The data used to train one part of the experiment was used to validate the other half.

5 different dropout rates (centred around the default value of 0.1) were tested over 5 trials. The average accuracy and loss is calculated for each dropout rate and plotted for analysis.

For the transfer learning half of this experiment, the training and validation data are swapped as transfer learning is used when applying weights learnt on one dataset to a network using another.

The weights used in the transfer learning task were copied from the base model after it was trained using the training and validation data used in the first half of this experiment.

The convolutional layers these weights were applied to were frozen, preventing any changes. As a network was created for each trial, the fully connected layer weights did not need to be reinitialised.

After this, one network was trained with a dropout rate of 0 (no dropout) and the other using the non-zero rate that gave the best average performance in part 1. This trial was run 5 times with averages computed and plotted for analysis.

Results (instructions) - 55 MARKS

Use the Results section to summarise your findings from the experiments. For each experiment, use the Markdown/text cell to describe and explain your results, and use the code cell (and additional code cells if necessary) to conduct the experiment and produce figures to show your results.

Experiment 1 (17 MARKS)

Write up results for Experiment 1 here

The average accuracy of the networks using the 0.001 learning rate to be the highest, the rate of 0.0005 and 0.0015 being second highest, the accuracy reducing the further from the default value the learning rate was. This was not the case.

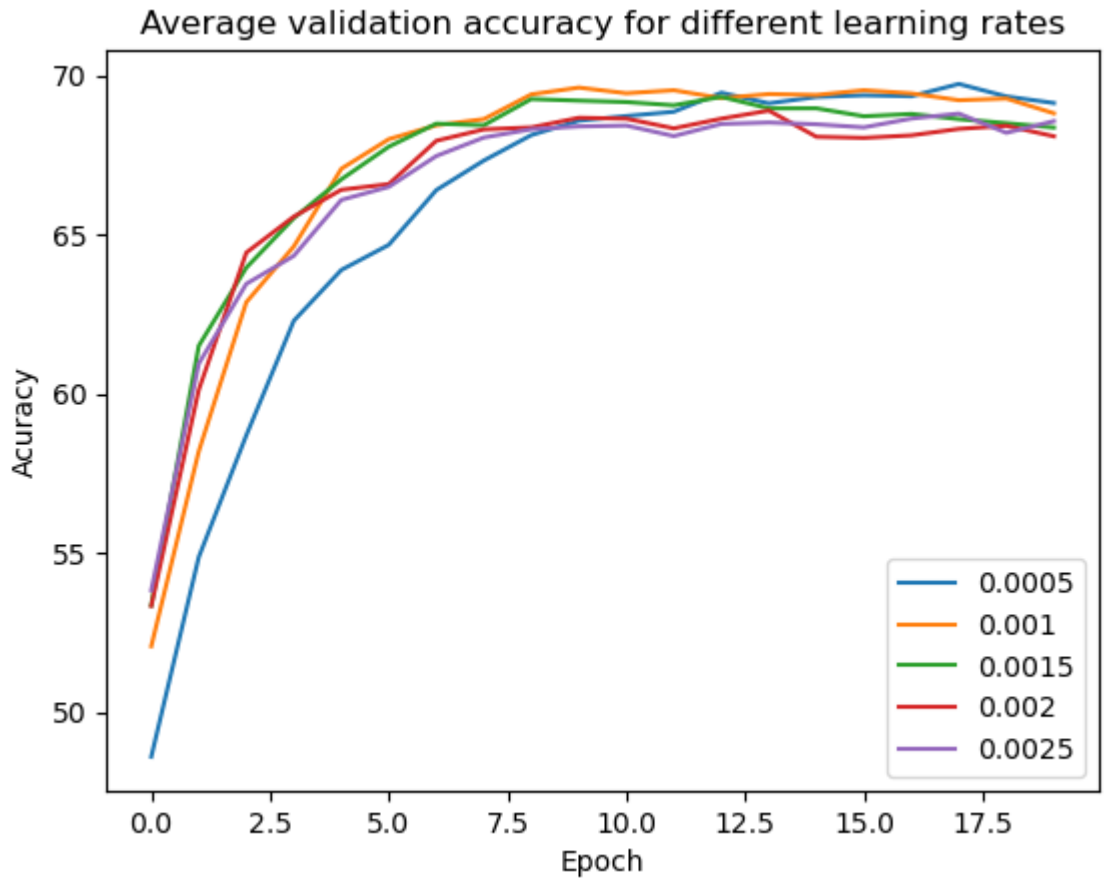

Figure 2

Figure 2 shows the initial gradient for all learning rates except 0.0005 to be similar, with the accuracy for the 0.0005 rate only approaching the others after 8 epochs. The smaller learning rate means that the weights and biases are changed by a lower amount, mean they take longer to get influenced. Because of this, finer adjustments to biases can be made, increasing the potential accuracy. When the accuracy reached that of the 0.001 learning rate (68.9% (3 s.f.)), all other networks have started to overfit so their accuracy is reducing. Because it takes longer to be influence, a low learning rate will make a network more resistant to overfitting. The 0.0005 rate ends with the highest accuracy of 69.1% (3 s.f.).

The average accuracy for the 0.0025 and 0.002 learning rates are very similar throughout all 20 epochs. Their gradients start to shallow after 2 epochs, and again after 7, until they start to lose accuracy after 10.

The 0.0015 learning rate plot has the highest initial gradient and keeps increasing in accuracy until epoch 8. After this, the average accuracy decreases until, after 20 epochs, it has the second lowest accuracy with 68.4% (3 s.f.).

The default value (0.001) had the second lowest initial accuracy gradient but had the highest accuracy after 4 epochs with 67.1% (3 s.f.). From epoch 10 onwards, the accuracy plateaued until epoch 16 where if dropped from 69.5% (3 s.f.) to 68.8% (3 s.f.).

The learning rate of 0.0005 ended with the highest average accuracy after 20 epochs and was least affected by overfitting. The learning rate of 0.001 had the highest accuracy with 69.9% (3 s.f.) and found it after only 8 epochs.

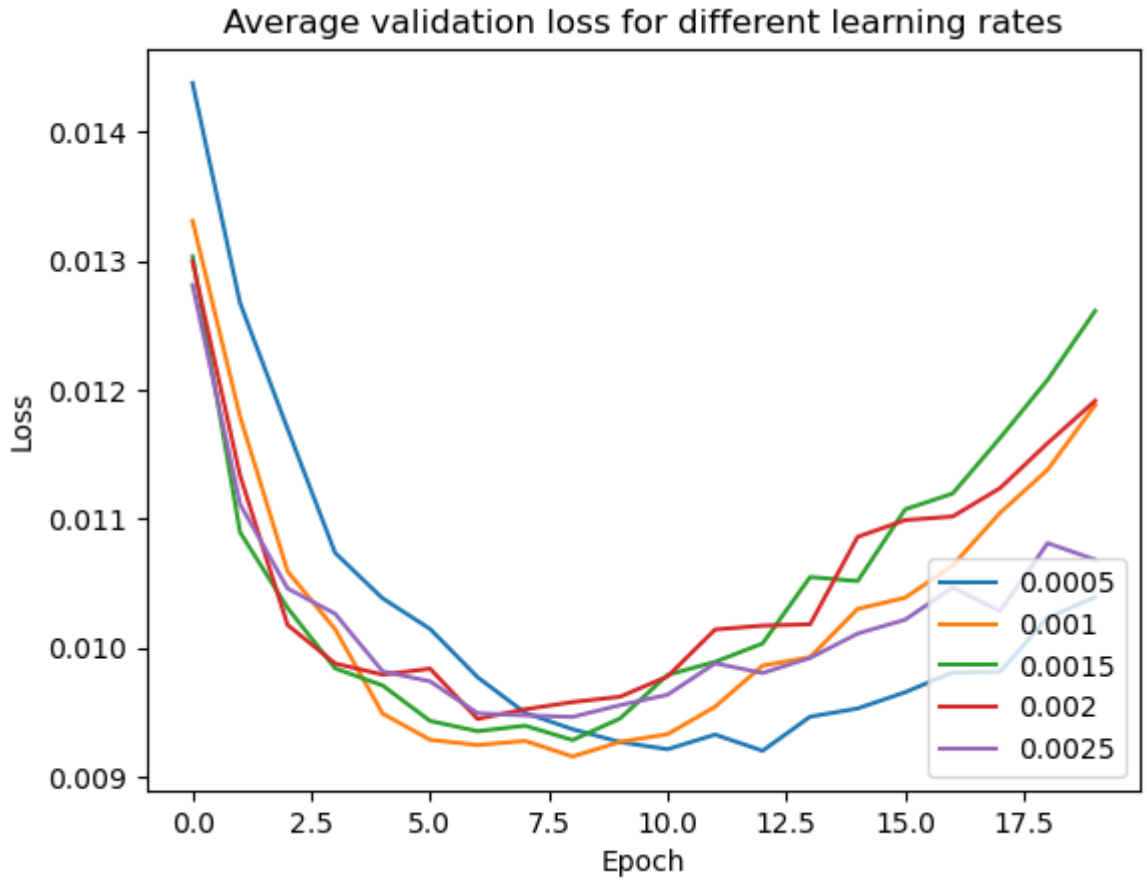

Figure 3

When compared to their average loss (figure 3), the loss of each learning rate corresponds to when it's accuracy starts to decrease. The loss of the 0.0005 rate reaches its minimum after 12 epochs, ending with the lowest loss of 0.0106 (3 s.f.).

The 0.001 and 0.002 curves had the joint second highest loss with 0.0120 (3 s.f.) with the 0.002 networks overfitting after 6 epochs, while the 0.001 curve shows overfitting after 8. The 0.0015 curve shows that most overfitting with a loss of 0.00932 after 8 epochs and a loss of 0.0124 (3 s.f.) after 20 epochs.

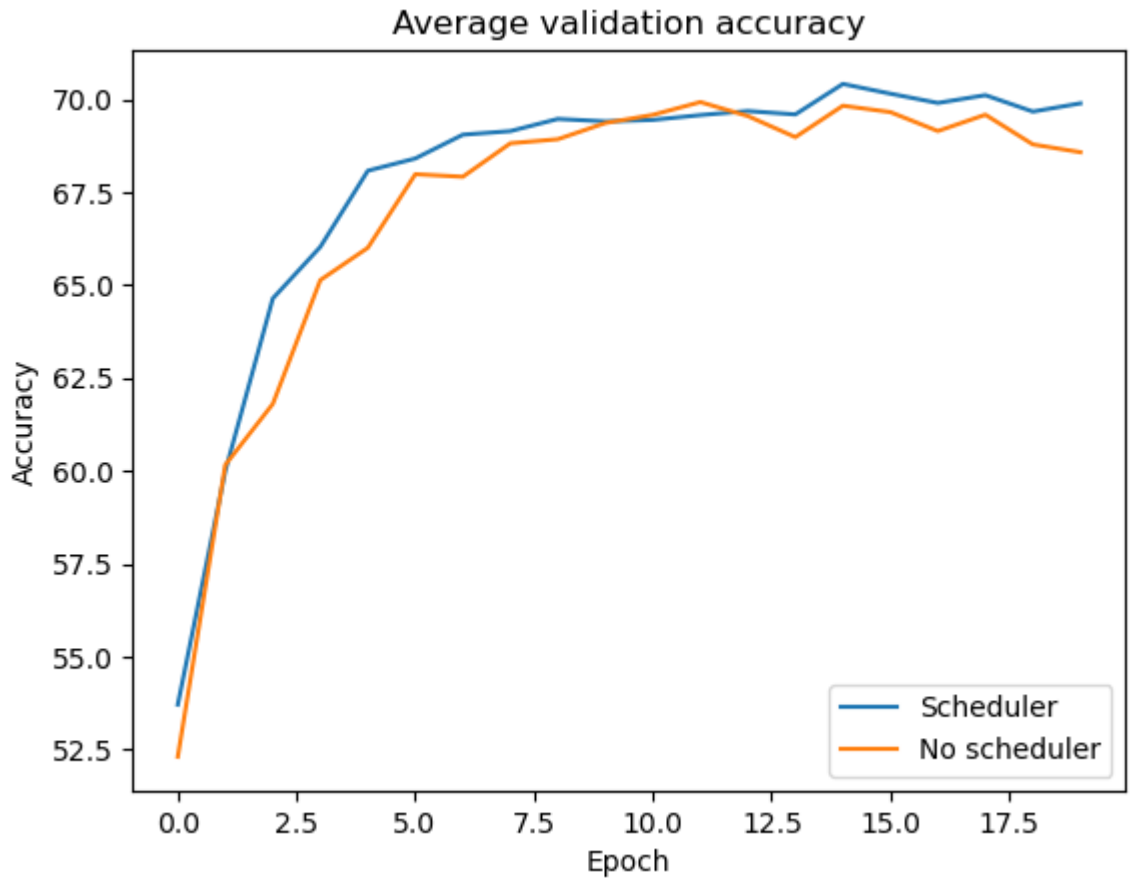

Figure 4

The average accuracy (figure 4) for networks using the scheduler was higher than those without over most epochs. After 9 epochs, the scheduler accuracy curve plateaus at 69.5% (3 s.f.) until the 14th epoch, when the smallest learning rate is used, when the accuracy reaches of 70.4% (3 s.f.). The scheduler started to overfit after 15 epochs, but as the lowest learning rate was in effect, the overfitting rate was low, leaving the accuracy at 69.9% (3 s.f.) after 20 epochs. The accuracy curve for the non-scheduled learning rate was lower than the scheduled curve until it's maximum value of 69.9% (3 s.f.) was reached. After this, the model started to overfit, ending with an accuracy of 68.6% (3 s.f.).

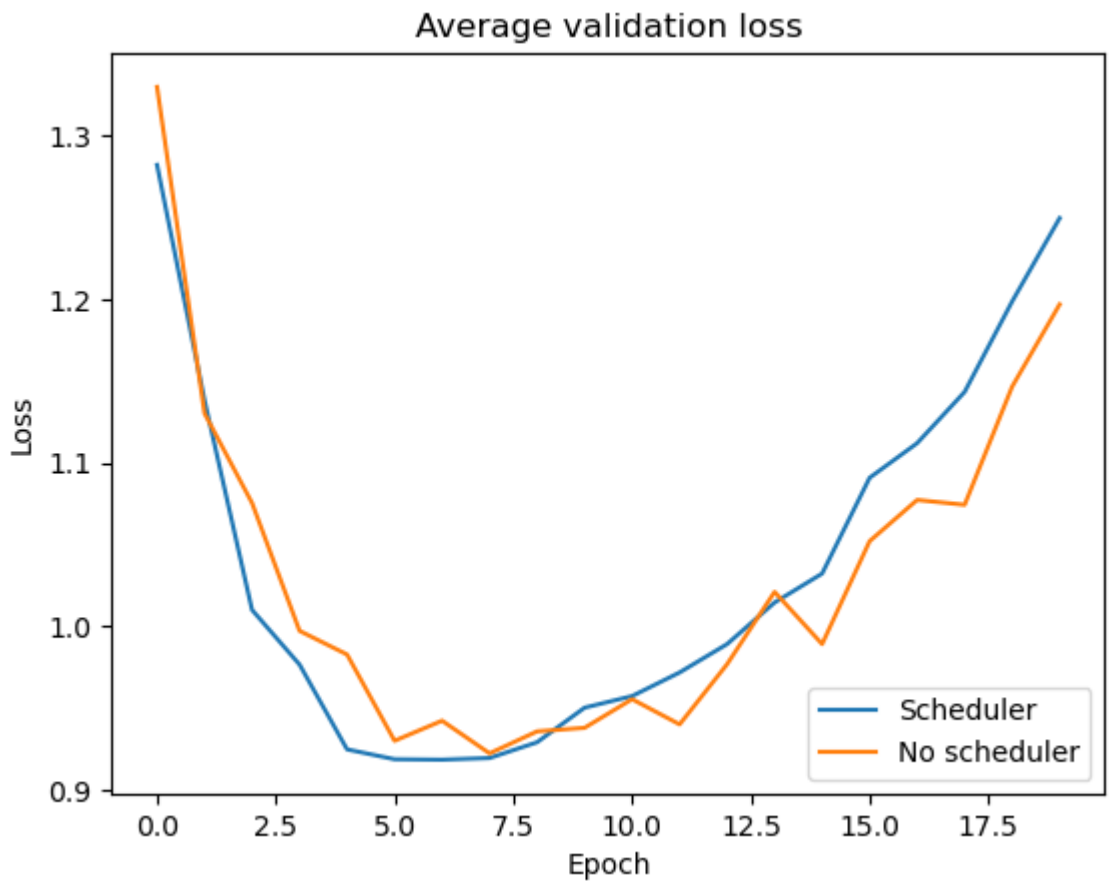

Figure 5

Figure 5 shows the scheduled loss curve below the non-scheduled curve until epoch 8, reached a minimum loss of 0.936 (3 s.f.). After 8 epochs, the scheduler had a higher loss than the non-scheduled curve with 1.25 (3 s.f.) and 1.20 (3 s.f.) respectively. The scheduled curve is smoother, suggesting more consistent loss across trials, while the non-scheduled curve suggests substantial variations.

Experiment 2 (19 MARKS)

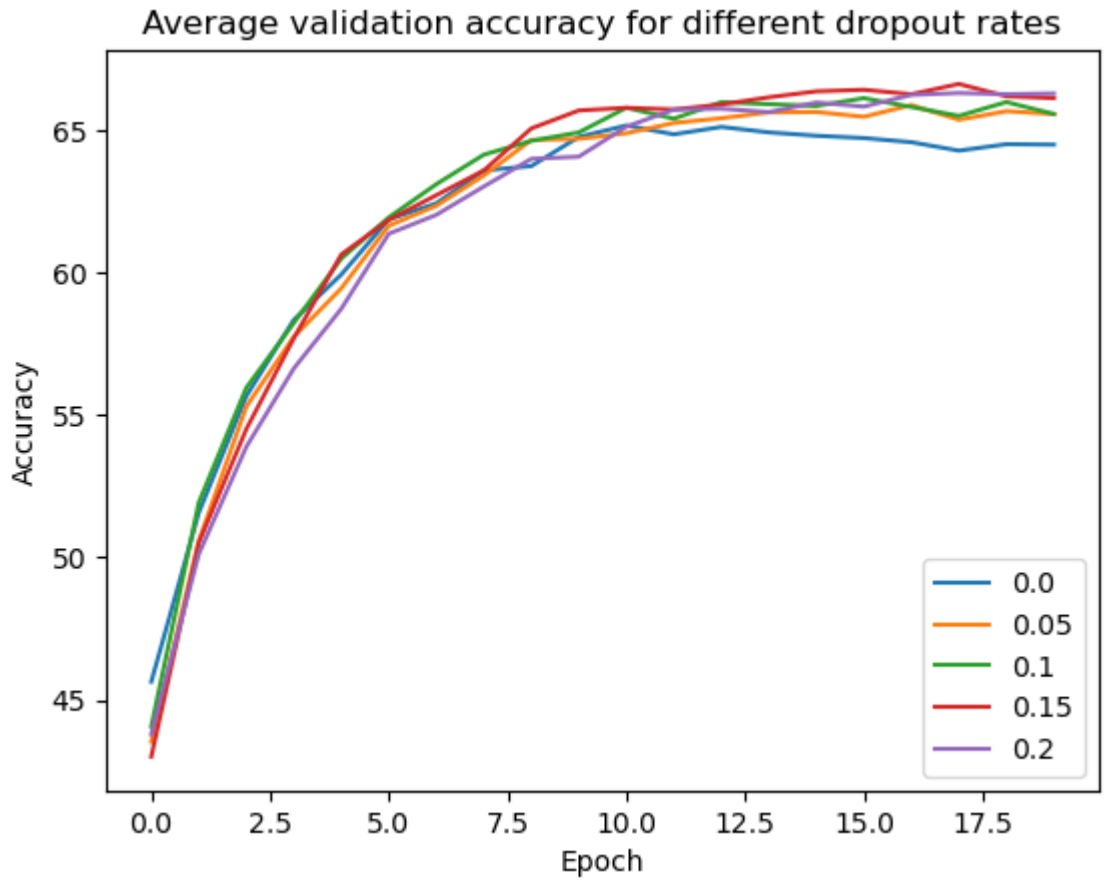

Figure 6

Figure 6 shows that average validation accuracy for the different dropout rates used ended at approximately 65% after 20 epochs, with the highest (dropout rate of 0.2) having an average accuracy with 66.3% (3 s.f.), and the lowest (dropout rate of 0) having the lowest accuracy with 64.5% (3 s.f.).

The dropout rate of 0.1 had the fastest convergence rate but ended only 4th most accurate with 65.6% (3 s.f.). The 0.2 dropout rate had the slowest convergence rate, taking 18 epochs before it had the highest accuracy. The slow high accuracy but shallowest gradient is likely due to dropout disabling connections between different neurons, giving each neuron a better chance to for their biases to become for effective while reducing the likelihood of overfitting.

When comparing the 0 and 0.2 dropout rates, the 0 dropout rate networks reached an average accuracy of 65.2% (3 s.f.) faster than with 0.2 dropout. After 10 epochs, however, the average accuracy for the 0.2 dropout rate keeps increasing, whereas the 0 dropout rate average decreases. This is because the 0.2 dropout networks are not overfitting as much as the 0 dropout networks.

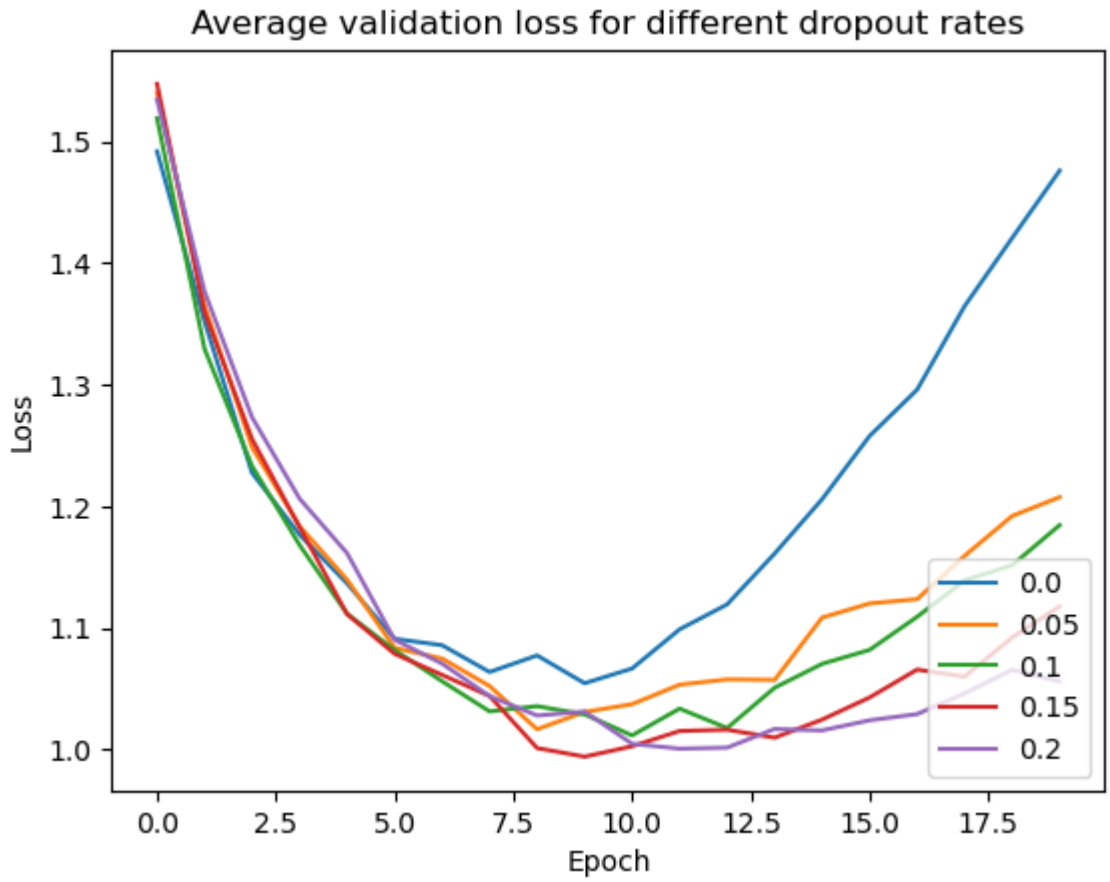

Figure 7

Figure 7 shows the average validation loss for the different dropout rates tested. All dropout rates performed similarly until 5 epochs (with a range of average loss from 1.11 (3 s.f.) to 1.16 (3 s.f.)), afterwards however, the average loss diverges as different dropout rates cause the networks to overfit at different rates.

From the 9th epoch, the average loss for networks using 0 and 0.05 dropout rates increased, with the 0 dropout networks having an average loss of 1.48 (3 s.f.) after 20 epochs.

The networks with a dropout rate of 0.15 had the lowest loss with 0.994 (3 s.f.) but ended with a loss of 1.12 (3 s.f.), the second lowest.

0.2 was the best dropout rate over 20 epochs as the accuracy of the models ended the highest, and the loss the lowest. The high dropout rate means that the model didn't overfit much to the training data. The fact that the loss started to increase means that overfitting is happening, but because a high dropout rate means that there is a high chance for nodes to be ignored, the reduced learning rate this creates also means that it is slower to overfit any models with a high dropout rate.

Similar to experiment 1, loss was lowest at approximately 10 epochs.

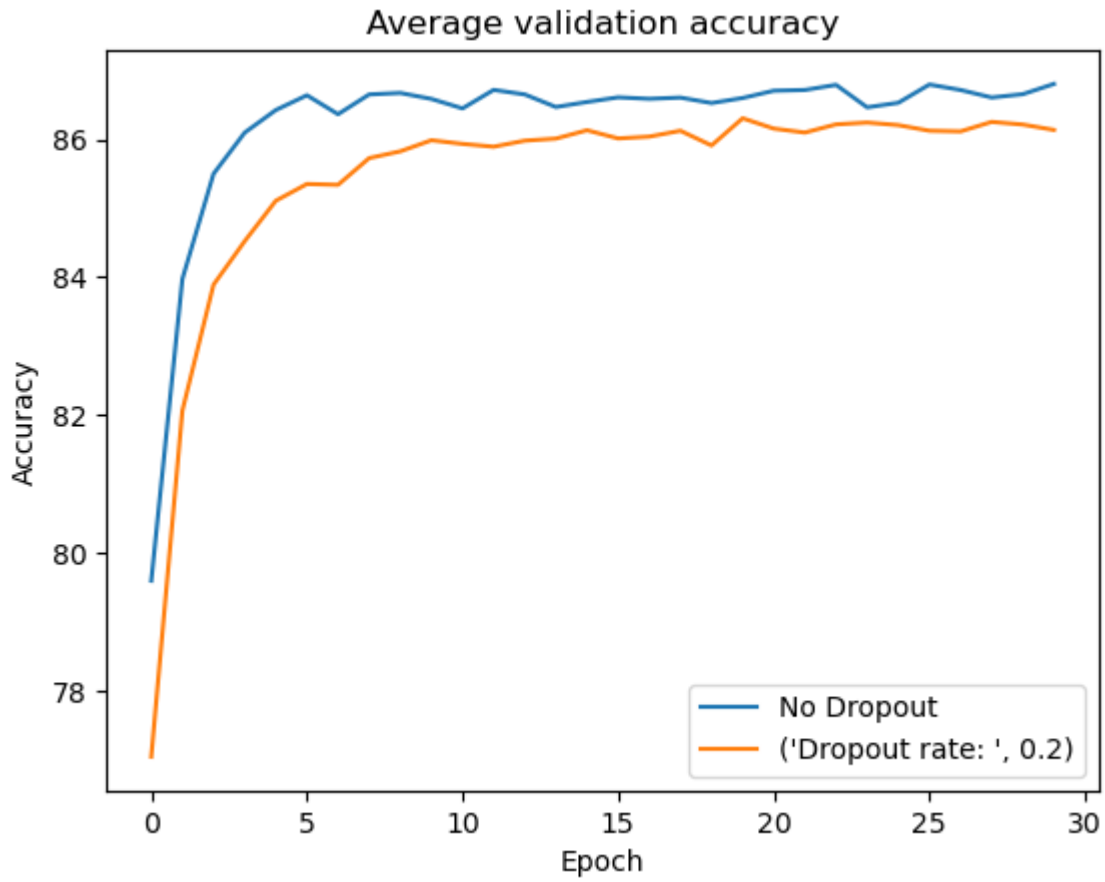

Figure 8

Figure 8 shows the average validation accuracy (for a model with a dropout rate of 0.2 and one with no dropout) when transfer learning. The networks without dropout had a higher average accuracy after every epoch and ended with the highest average accuracy of 86.8% (3 s.f.), whereas for the 0.2 dropout, the highest was 86.3% (3 s.f.), ending on 86.1% (3 s.f.), making it 30% more accurate than without transfer learning.

The curve accuracy when not using dropout is less smooth, implying that some networks were inconsistent with the others, or that the accuracy was inconsistent throughout each network.

The gradient of the accuracy curve for the 0.2 dropout networks decreased earlier than the 0 dropout curve but plateaued later (at epoch 9 rather than 5). The curve is much smoother, implying that the accuracy was more consistent across the different trials.

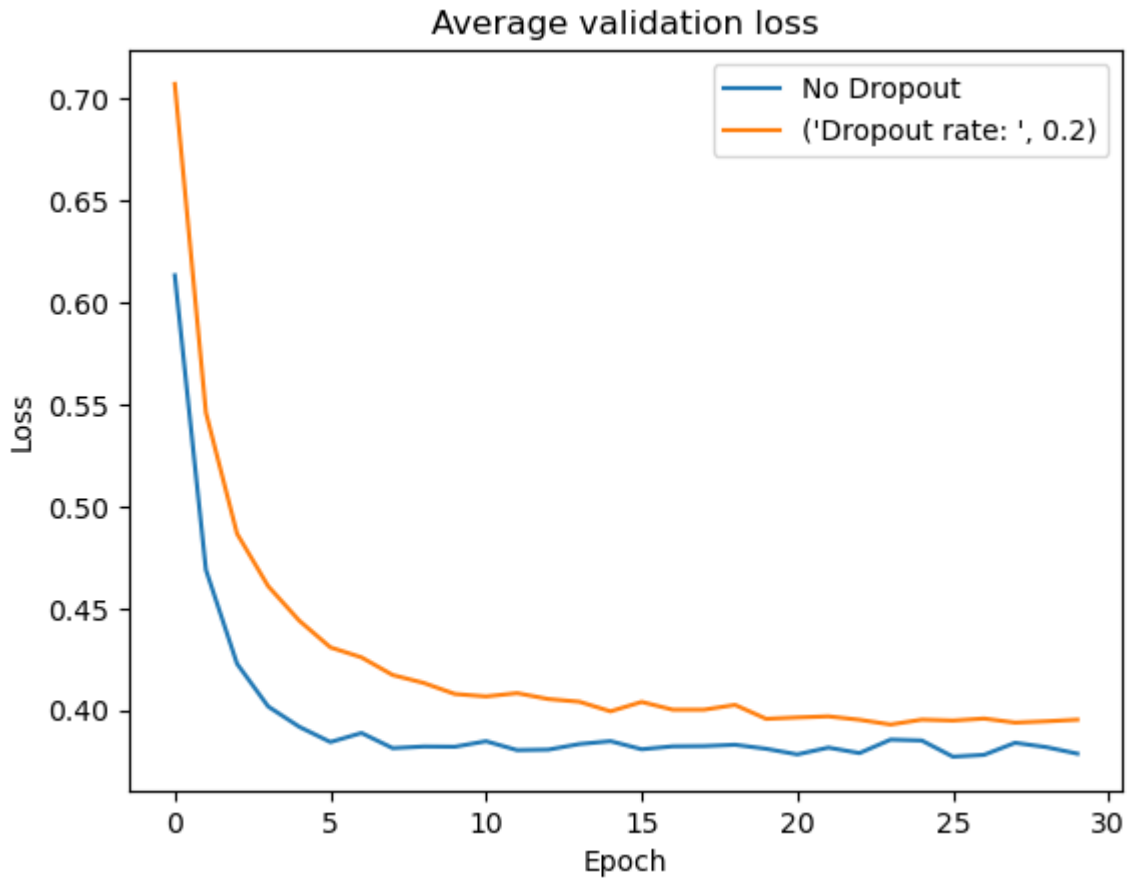

Figure 9

The average validation loss (Figure 9) shows that the 0 dropout loss quickly decreases to 0.384 (3 s.f.) after 6 epochs and plateaus, ending on 0.379 (3 s.f.). This happened 40% faster than without transfer learning while reducing loss by 66%.

Conclusions and Discussion (instructions) - 25 MARKS

In this section, you are expected to:

- briefly summarise and describe the conclusions from your experiments (8 MARKS).
- discuss whether or not your results are expected, providing scientific reasons (8 MARKS).
- discuss two or more alternative/additional methods that may enhance your model, with scientific reasons (4 MARKS).
- Reference two or more relevant academic publications that support your discussion. (4 MARKS)

Experiment 1

Using a scheduler to vary the learning rate improves performance of a network. This was expected as it allows a network to reach a high accuracy earlier during training while avoiding any local minima. After the model approached the global minimum and a high accuracy is found, the learning rate can be reduced increasing the networks accuracy while avoiding "bouncing" out of the found minima.

The shallowest end gradient was expected because the low learning rate grants a resistance to overfitting. It was expected that the 0.0025 learning rate curve had the highest minimum loss (0.00961 (3 s.f.)) but it was unexpected for it to end with the second lowest loss with 0.0149 (3 s.f.).

It was expected for the scheduler to improve accuracy and decrease performance of the model, however, it was not expected for the scheduler to end the training period with a greater loss.

Experiment 2

The data gathered during this experiment suggests that using dropout is good for training a network over a large number of epochs as it reduced the rate of overfitting, however, when transfer learning, it is better to not use dropout. While it was expected that the networks with a high dropout rate would be resistant to overfitting, the fact networks not using dropout having a higher average accuracy and lower loss when transfer learning was not expected. It was expected that the dropout-less networks would have a steeper accuracy curve (as more neurons would be taken into account when training for any given epoch) but that the networks with dropout would have a higher accuracy and lower loss after the training period (as seen in part 1 of this task).

It was expected that the 0.2 dropout average would have a shallower gradient but would decrease below the average loss for 0 dropout. This is not the case. At every epoch where the accuracy drastically decreased, the loss increased, creating a mirror of accuracy curves. The average accuracy of the 0 dropout networks becomes unstable when the loss becomes unstable. There are more differences in the loss the longer the networks were trained as some overfitting was introduced; the 0.2 dropout curve, however, plateaus after the 23rd epoch as it takes longer for any overfitting to take effect.

It was expected that the 0.2 dropout accuracy curve would be smoother and higher than the 0 dropout curve. This is because the dropout rate means that weights are effected less often (smoothing the curve) and because the accuracy after 20 epochs was greater in figure 5, it was expected that the 0.2 dropout average accuracy would be greater than the 0 dropout accuracy after 30 epochs, especially given that after 12 epochs, the accuracy of the 0 dropout rate networks started to decrease (figure 6).

Additional Methods

One method that could be used to improve the accuracy of the network would be to pad the images as part of preprocessing. This would help the locational information be preserved for longer through the networks. This, however, may work best on larger networks. Similarly, tests should be done with various convolutional filter sizes to see how these affect the accuracy and time taken to train as different filter sizes would change the amount of location-based information the network has.

---

keyboard_arrow_down

References (instructions)

Use the cell below to add your references. A good format to use for references is like this:

[AB Name], [CD Name], [EF Name] ([year]), [Article title], [Journal/Conference Name] [volume], [page numbers] or [article number] or [doi]

Some examples:

JEM Bennett, A Phillipides, T Nowotny (2021), Learning with reinforcement prediction errors in a model of the Drosophila mushroom body, Nat. Comms 12:2569, doi: 10.1038/s41467-021-22592-4

SO Kaba, AK Mondal, Y Zhang, Y Bengio, S Ravanbakhsh (2023), Proc. 40th Int. Conf. Machine Learning, 15546-15566

List your references here https://keras.io/

Diederik P., Kingma, Jimmy Ba (2015), Adam: A Method for Stochastic Optimization, ICLR 2015

https://arxiv.org/abs/1412.6980
