Machine learning assignment 4

i.

a. See Appendix for code

b.

The input image is 32x32 in size and is shown in figure 2.

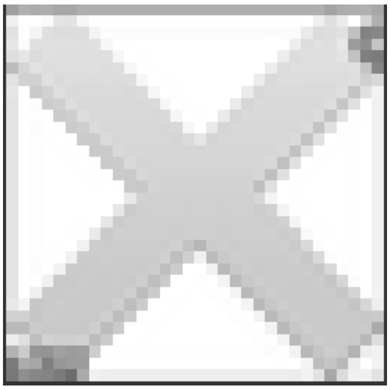

Figure 1

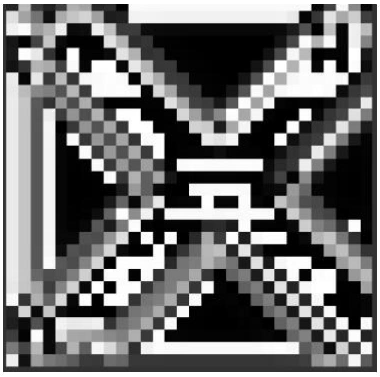

Figure 2

Figure 3 shows the result of convolving the input image with kernel 1. Kernel 1 produces a matrix of size 30X30, due to the 3x3 kernel cutting off 1 row from the top and bottom and one column from each side, causing the output to reduce by two in each dimension. Kernel 1is able to provide good features to represent the image. It can be seen there are some feature which are useless inside the ‘x’ shape, but the kernel was able to capture the fundamental idea of the image.

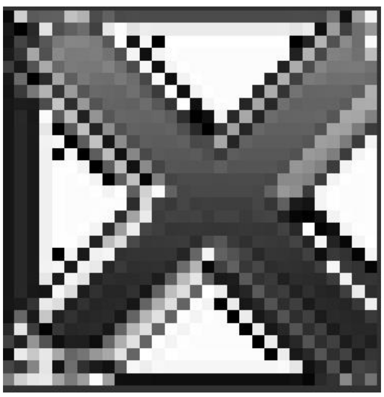

Figure 3

Figure 4 shows the output when the input image is convolved with kernel 2. Similarly to figure 3 the image size is 30x30. It can be seen for this kernel, that the features of the image are more efficient in creating a better representation of the image. Kernel 2 would be better for this particular image.

ii.

a.

There are 4 convolutional layers with one dropout layer and one dense layer at the output. The four convolutional layers have a 3x3 kernel. The first convolution layer has an output of 32x32x16. The second layer has an output an output of 16x16x16. The third convolutional layer has an output of 16x16x32. The fourth convolution layer has an output of 8x8x32. The dropout layer has an output of 2048. The final dense layer has an output of 10 for the 10 classes.

b.

i.

Layer	Parameters
1st Convolution layer	448
2nd Convolution layer	2,320
3rd Convolution layer	4,640
4th Convolution layer	9,248
Dropout	0
Dense	20,490

Total number of trainable params: 37,146

The dense layer has most params : 20490. This was expected as the dense layer is considered a fully connected layer. This means that every output of a dense layer is connected to every input of the following layer, meaning they will have lots more parameters than a convolution layer as the convolutional layer is connected to a sub set of the previous layer called a kernel which size is usually 3x3.

Training: 58%
Test Data: 50%
Baseline: 14.3%

The test data performs with an accuracy which will be very rarely be higher than the training data, as the model has been specifically trained to perform well on the training data. It is seen that the model achieves an accuracy of approximately 58% on the training data, while it performs with a 48 % accuracy on the test data. The better the representation of the test data the training data is, the closer the test and training accuracy will be. It's important that the training data is a good representation of the actual use case for the model.

The baseline model used was a majority class predictor. The data wasn't split into training and testing for the baseline. The baseline achieved a 14.3% accuracy on the dataset, which is significantly less than the CNN. This clarifies that the CNN extra complexity and resources needed to develop it was worth it as the result are significantly better than the baseline.

ii.

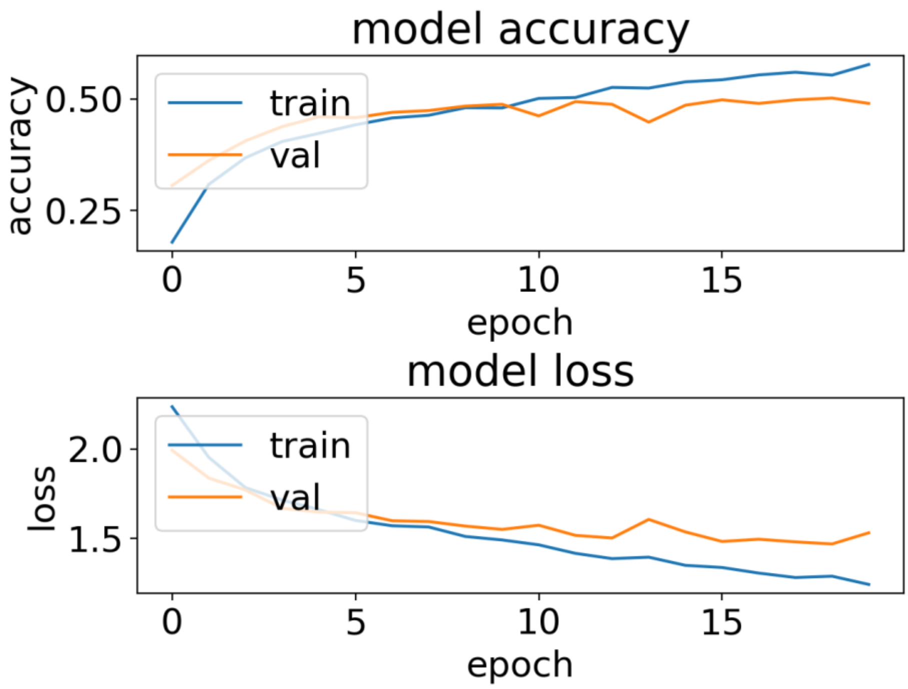

Figure 4

Figure 1 shows the accuracy and loss graph for the model. Figure 5 can be used to identify a overfitting or underfitting by the model. It's clear from figure 5 that the training data performs better than the test data as expected. The shape of both plots is near the ideal. For the accuracy plot the test data accuracy starts low and grows rapidly initially before it plateaus and converges to a value of roughly 50%. For the loss plots, the test accuracy starts high and rapidly drops before it plateaus to a value near zero. There is a small amount of overfitting seen as the training and test accuracy curves tend to diverge from around epoch 11.
Underfitting – if the accuracy of the training and validation is low, you know there is underfitting. This can be down to the model being too simplistic or poor data/features being used to train the data. E.G. trying to predict somebody's weight by using their favourite colour as a feature. Underfitting graphs will tend to have a growing training loss as with increasing epoch, the training and test loss will be close to each other at the final epoch, and there will be a sudden drop in the training loss in the final epochs. Figure 5 doesn't display these features, thus from the analysis of this graph there does seem to be any underfitting.

It clear that a good understanding of whether the model is underfitting or overfitting can be got from the analysing figure 5. This will be helpful for understanding the model and to help improve its performance.

iii.

Training points	Training time(s)	Train Accuracy	Val Accuracy
5,000	57	58%	50%
10,000	109	60%	57%

20,000	235	63%	62%
40,000	436	68%	66%

The table above shows the training times, the accuracy for the training and test datasets for the corresponding training dataset size. The training and validation accuracy increased with increasing dataset size. This was expected once the added data contained useful features or added to pre-existing features. It also shows that the training time increases. This was also expected as the more training data there is, the more data will needed to be processed and optimised for the model. There will be more computations for the computer to do and thus will take the computer longer to train.

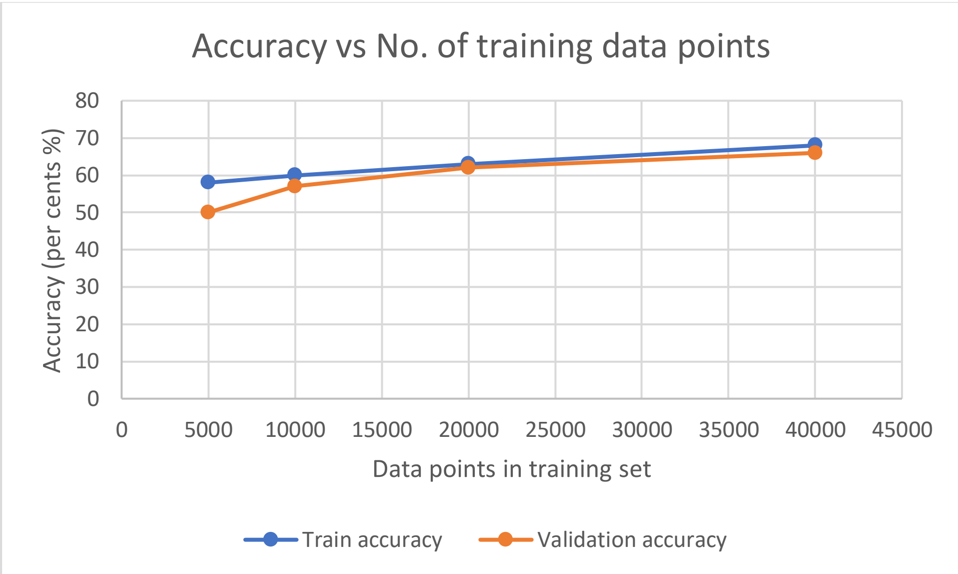

Figure 5

Figure 6 shows that the training accuracy increases in a linear fashion from 58 % to 68% for increasing the training data points from 5,000 to 40,000. This data was got from the learning graphs of each datapoint. It shows that the test accuracy increases more quickly between 5,000 and 10,000 data points and then levels off, to increase linearly at a similar rate to the training data. The result of increasing the data points, is that the testing accuracy is nearly identical to the training accuracy for the largest dataset of 40,000, thus reducing overfitting. This was expected as by increasing the training data, it will produce a more representable dataset once the added data was relevant which would be fair to assume in this case.

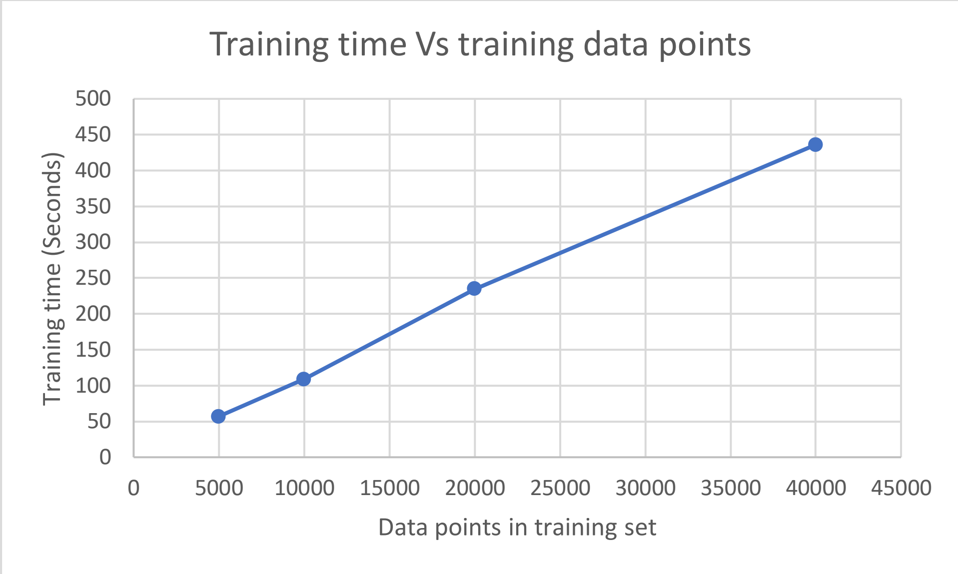

Figure 6

Figure 3 shows the increase in training time with the increasing training dataset size. The graphs appear to be nearly linear, pointing to the fact the computations increase in a linear fashion with increasing training dataset size.

iv.

L1 weight	Training time (s)	Training Accuracy	Test Accuracy
0	52	58%	47%
0.0001	60	58%	52%
0.001	55	52%	46%
0.01	58	42%	40%
0.1	57	29%	35%
1	57	23%	18%

The table above shows the training time along with the training and testing accuracies for various L1 weight value between 0 and 1. It can be seen that for in general both the train and testing accuracy decreases significantly with increasing L1 weight value. The only slight outlier from that trend is the L1 weight value of 0.0001 which is slightly better test accuracy than L1 weight value of 0. The training time remains relatively stable between 52 and 60 seconds for all L1 weight values.

It can be seen that a value of L1 = 0.01 is closest the test accuracy is to the training accuracy with only a 2% difference, thus is suggest this value provides the least amount of overfitting. The drawback with L1 is to achieve the minimal amount of overfitting the accuracy has been compromised to 42% and 40% for training and testing respectively. Adding more data will have the benefit of both increasing the training and test accuracies, while also reducing the differential between the training and testing accuracy, thus reducing overfitting. For this reason adding more data will always better to combat overfitting than regularisation. Adding more data can be expensive in terms of resources and time so it will not always be available. When this is the case L1 provides a good compromise.

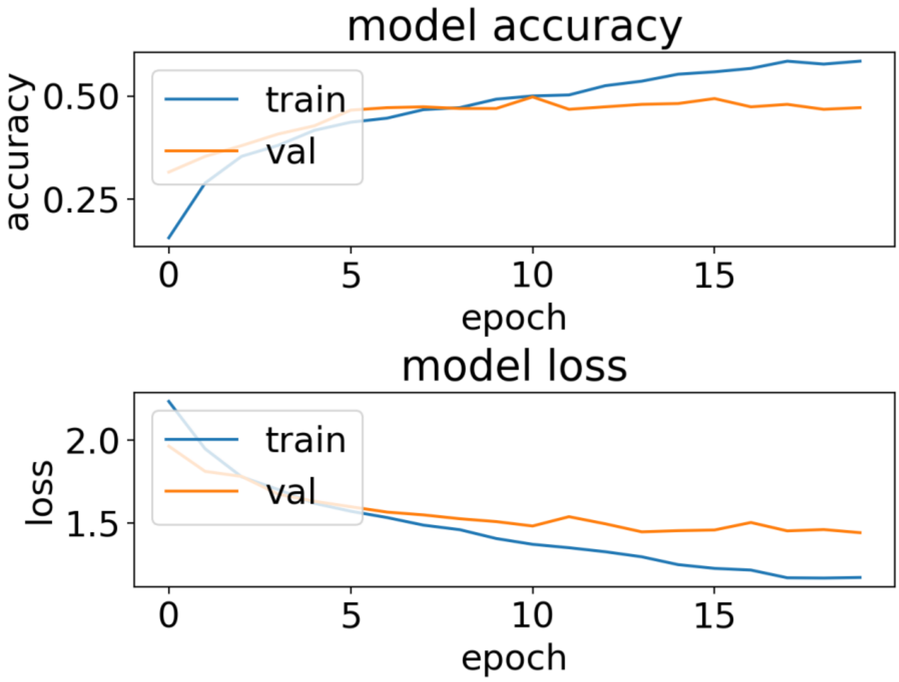

Figure 7 L1 = 0

Figure 8 shows the learning graph L1 = 0. There is a bit of overfitting after epoch 10 as the training and testing accuracies begin to diverge. It achieves an accuracy of 58% and 47% for training and testing respectively.

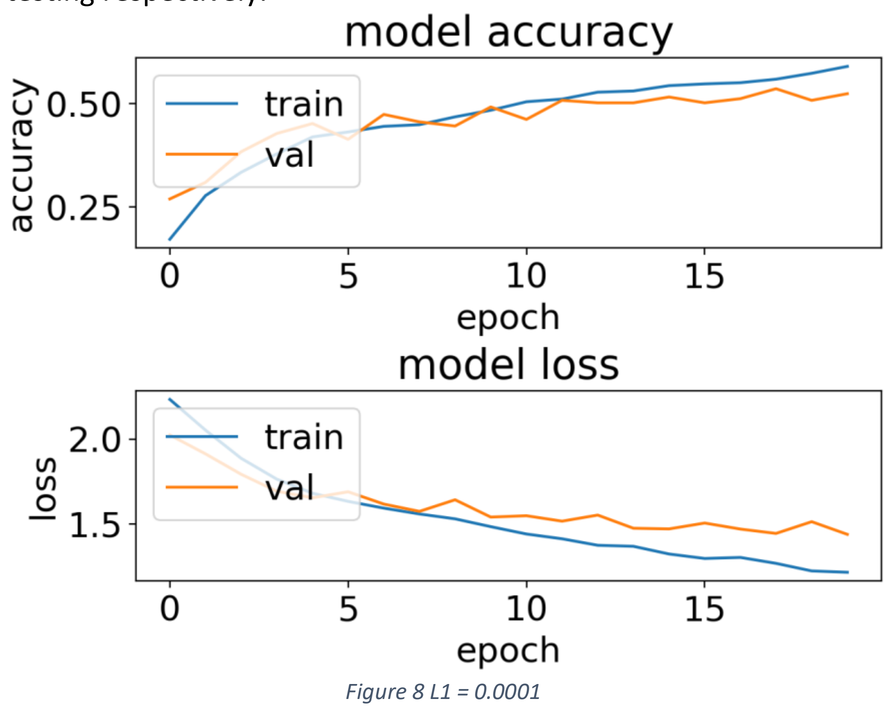

Figure 8 L1 = 0.0001

Figure 9 shows the learning graph L1 = 0.0001. There is a bit of overfitting after epoch 11 as the training and testing accuracies begin to diverge. This is less than L1 = 0. It achieves an accuracy of 58% and 52% for training and testing respectively. This is the highest training and testing accuracy showing that this is the optimal L1 value.

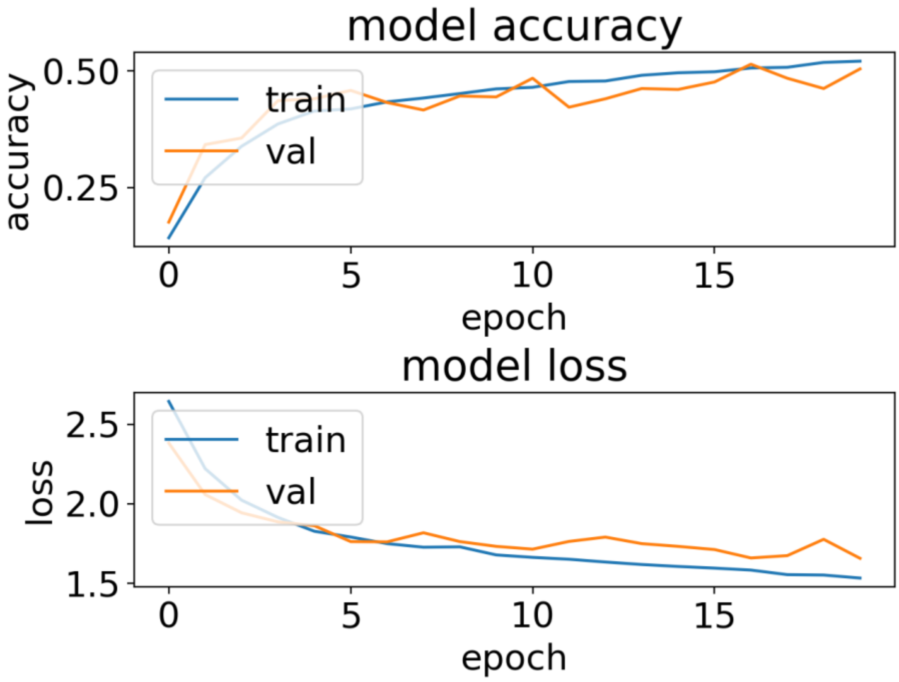

Figure 9 L1 = 0.001

Figure 10 shows the learning graph L1 = 0.001. This is the graph with the least overfitting as can be seen from figure 10, the training and testing accuracies are very close together. It achieves an accuracy of 42% and 40% for training and testing respectively.

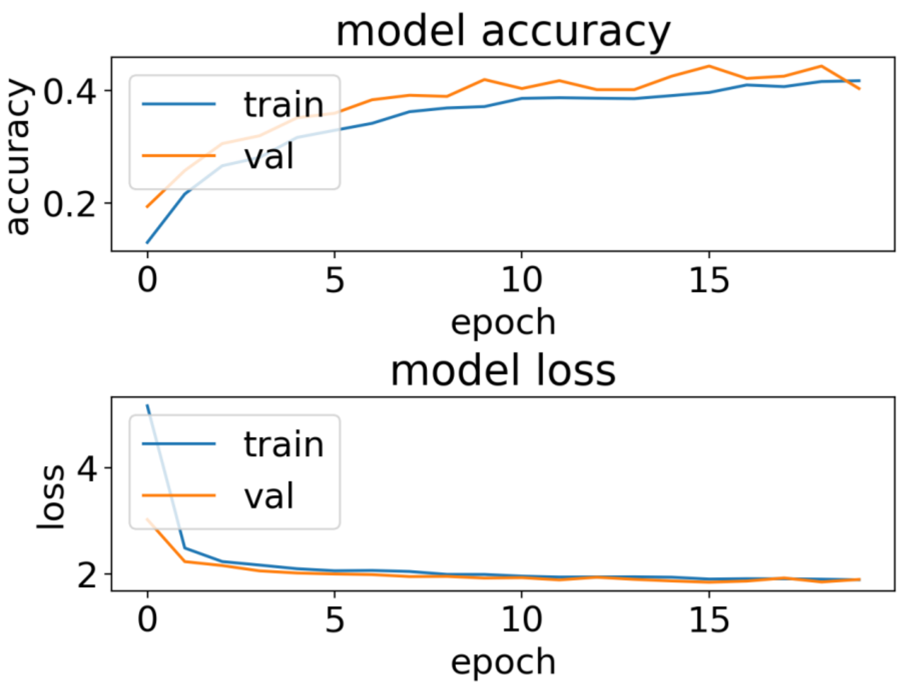

Figure 10 L1 = 0.01

Figure 11 shows the learning graph L1 = 0.01. Figure 11 has a minimal amount of overfitting as seen again the training and testing accuracies are very close to each other. It achieves an accuracy of 52% and 47% for training and testing respectively. This is lower than both L1 = 0 and 0.0001. This is a sign of potential underfitting as both accuracies are lower for increasing L1 value.

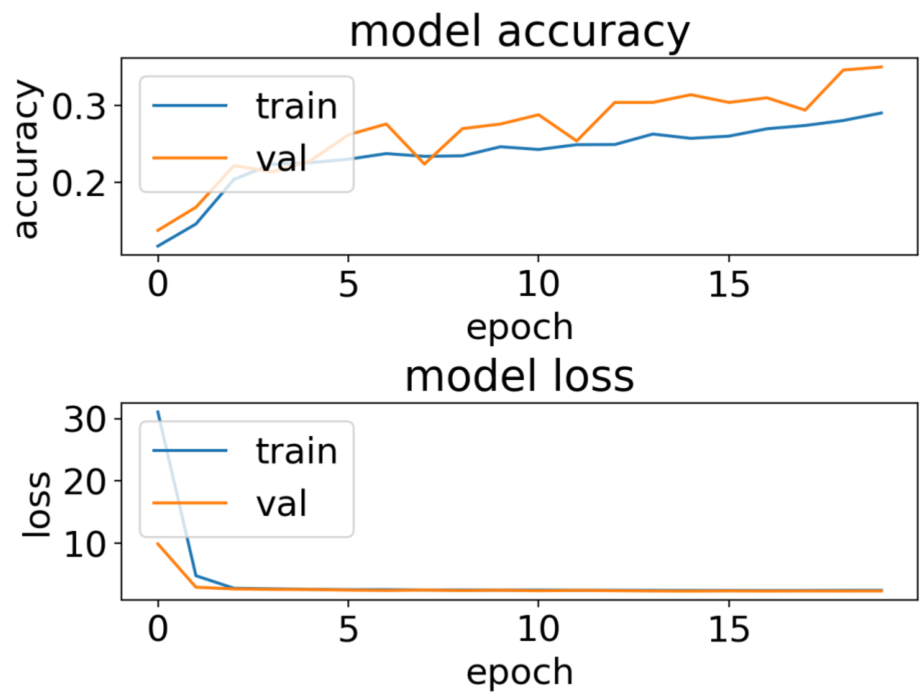

Figure 11 L1 = 0.1

Figure 12 shows the learning graph L1 = 0.1. It achieves an accuracy of 29% and 35% for training and testing respectively. Interestingly the test accuracy is higher than the training accuracy. This is a sign of further underfitting as both accuracies are lower for increasing L1 values.

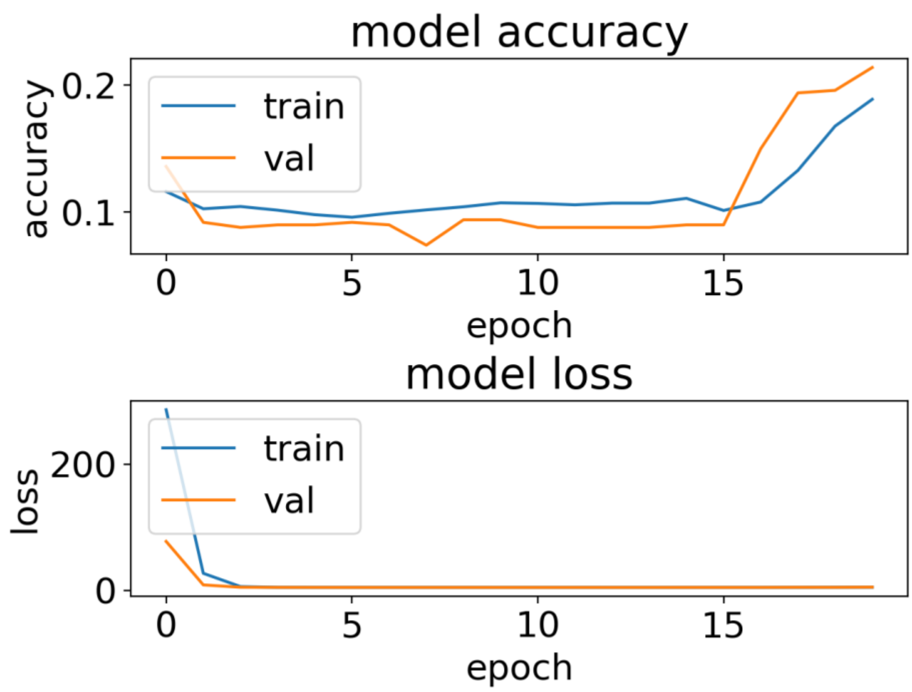

Figure 12 L1 = 1

Figure 13 shows the learning graph L1 = 1. It achieves an accuracy of 23% and 18% for training and testing respectively, the lowest of all the tested L1 values. This graphs shows for L1 = 1 there is significant underfitting.

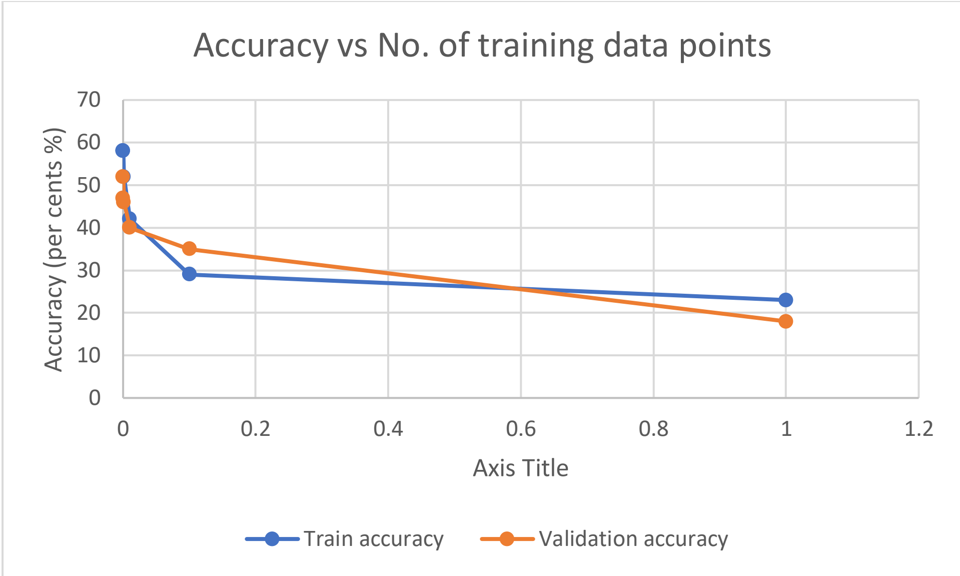

Figure 13

Figure 4 shows the accuracy for both the training and testing sets with varying L1 weight value from 0 to 1. The optimal value for L1 in this case is 0.0001. It decreases sharply beyond that point.

I think it is quite clear from the above plots that, increasing the training data is a much more efficient way to combat overfitting. Unfortunately, it will not always be easy to gain access to extra data, particular large amounts of useful data. Data is expensive to collect and can be challenging also. Using an L1 regularisation can provide a cheap and easy alternative to getting more data as L1 with a value of 0.0001 performs well with little overfitting seen from the analysis of figure 1. It will never be as good as getting extra data, and can be sensitive to changes in the L1 weight.

c.

i.

ii.

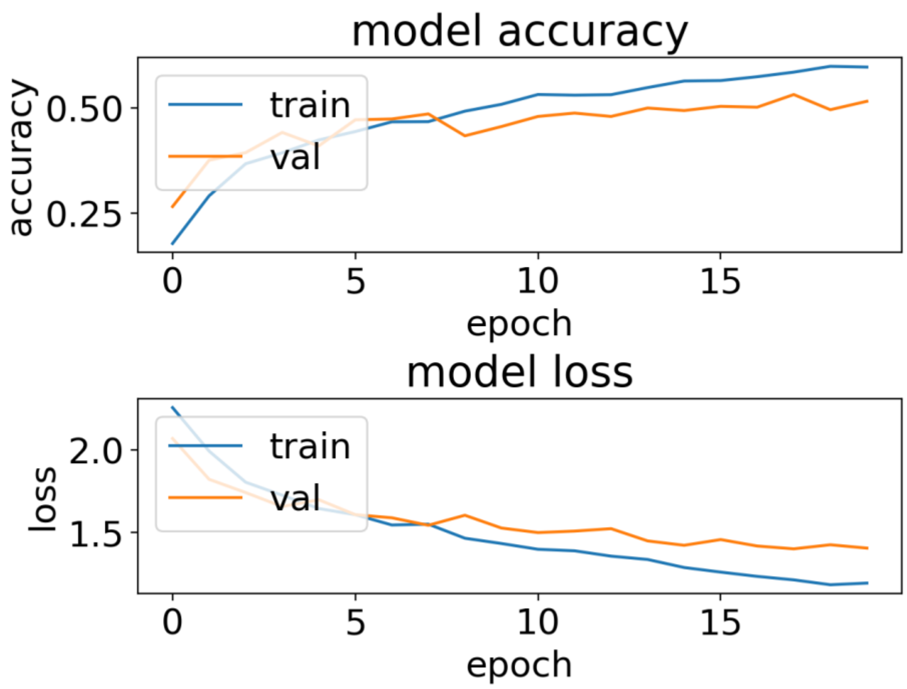

Figure 14

Model	Training time	Train Accuracy	Test Accuracy	Parameters
Strides	57 s	58%	50%	37,146
Max Pool	122 s	60%	52%	37,146

Keras estimates that there are 37,146 parameters for the model with max pooling which is the same as the original model with strides instead of pooling.

Figure 14 shows the learning graph for the model with the two max pooling layers. It's clear that both the loss and accuracy graphs look very similar to the original model with strides. The model with pooling achieves an accuracy of 60 % on the training set and 52 % on the testing set. This is 2 % better for both sets of training data when compared with the original model.

It's clear from the table that the training time has risen from 57 seconds to 122 seconds. This was expected as using max pooling, since all the different datapoints must be considered to find the max value for each pool. In comparison with strides, the strides simply skips every second element in both axis' if the stride is 2x2, meaning there is much less computations to compute thus leading to a faster training time.

It's clear that the extra computation time of using max pooling for an extra 2 per cent accuracy was worthwhile in this case as it only took an extra minute to train. For larger datasets or more complex networks this will not be the case as the training time between using max pooling and using strides could be hours, unless you have a high performance computer with GPU's. A high performance computer with a GPU and perhaps specific neural engine like the latest range of M1 Macs could still make it feasible to use max pooling .

I trained the model using 5 k data points. It achieved an accuracy of 49 % on the training set and 47% on the test set. These values are very similar to the thicker and shallower network. The thinner and shallower has 23,314 parameters. This means there would be less calculations to compute, thus it should be quicker to train. This is backed up by a 15 second faster training time of 42 seconds than the thicker and shallower network. The lesser amount of parameters would reduce the amount of feature which could reduce the accuracy of the model due it being simpler. This will also have the effect of reducing overfitting, which can be seen as the training and test accuracies are nearly identical.

d. (optional)

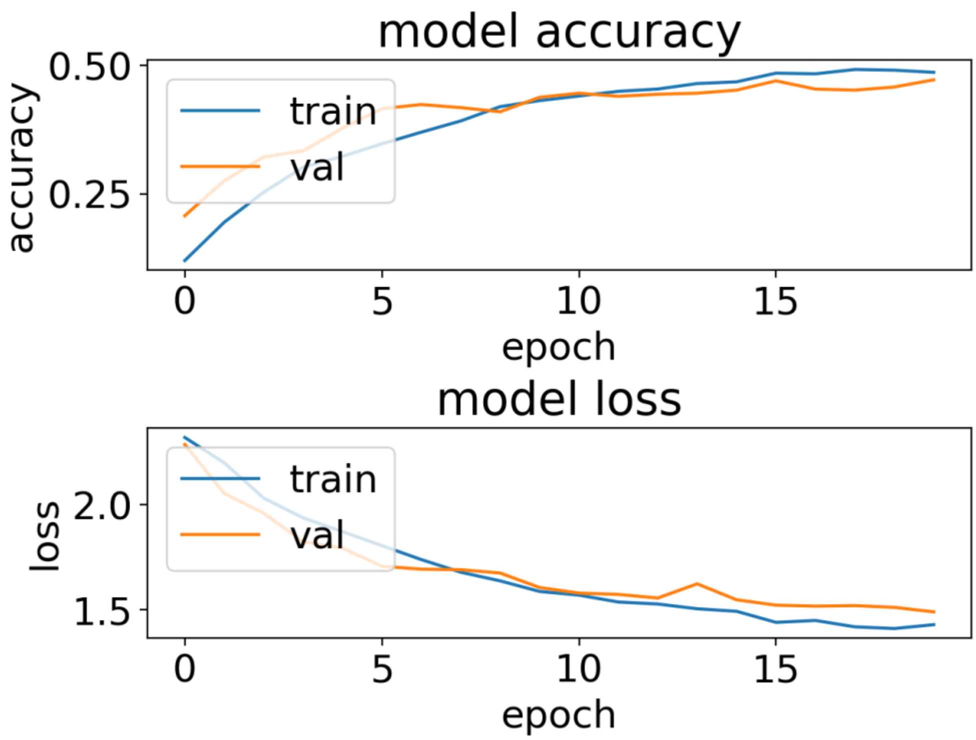

Figure 15

Figure 15 shows the learning graph for the thinner and deeper model. As mentioned the accuracy is very similar to the thicker and shallower model with the training and testing accuracy being slightly closer to each other.

Appendix - code

```python
def convolution(array1, kernel1):
    n = len(array1)
    k = len(kernel1)
    con_total = 0

    Convo = [[0 for x in range(n-k+1)] for y in range(n-k+1)]
    indent = int(k/2)
    e = 0
    f = 0
    for x in range(indent, n-indent):
        for y in range(indent, n-indent):
            con_total = 0

            c = 0
            d = 0
            for a in range(x-indent,x+indent+1):
                for b in range(x-indent,x+indent+1):
                    arr = array1[a][b]
                    #print("Array",arr)
                    ker = kernel1[d][c]
                    #print("Kernel", ker)
                    con_total = con_total+(arr*ker)
                    #print("Total Con",con_total)
                    if(c<2):
                        c = c+1
                    else:
                        c = 0
                        d = d+1
            Convo[e][f] = con_total
            print("Conv array:", Convo[f][e])
            #if(e<(n-2*indent)):
            if(e<n-2*indent-1):
                e = e + 1
            else:
                e = 0
                f = f+1
        print("Convo:", Convo)
```

Figure 16

Figure 1 shows my implementation of the convolution function. Array1 and Kernel1 are the image and kernel respectively. The first two for loops are for cycling through the image. It is indented since the kernel will cause to image to be clipped depending on the size of the kernel used. The second two for loops are for cycling though the pixels to be multiplied by the kernel. The con_total variable is the total value for a given pixel. This is stored in a given Convo array which is the computed matrix for the image array1 and kernel1. The if and else statement is for cycling though the kernel for each value in the image.

```
51 model = keras.Sequential()
52 model.add(Conv2D(16, (3,3), padding='same', input_shape=x_train.shape[1:],activation='relu'))
53 #model.add(Conv2D(16, (3,3), strides=(2, 2), padding='same', activation='relu'))
54 model.add(Conv2D(16, (3,3), padding='same', activation='relu'))
55 model.add(MaxPooling2D(pool_size=(2, 2)))
56 model.add(Conv2D(32, (3,3), padding='same', activation='relu'))
57 #model.add(Conv2D(32, (3,3), strides=(2,2), padding='same', activation='relu'))
58 model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
59 model.add(MaxPooling2D(pool_size=(2, 2)))
60 model.add(Dropout(0.5))
61 model.add(Flatten())
62 model.add(Dense(num_classes, activation='softmax',kernel_regularizer=regularizers.l1(0.0001)))
63 model.compile(loss="categorical_crossentropy", optimizer='adam', metrics=["accuracy"])
64 model.summary()
```

Figure 17

Figure 15 shows the implementation of the max pooling layers and removing the stride of two convolutional layers. There was not much to change. The stride parameter was simply removed from both convolutional layers. A max pooling layer was then added directly after the convolutional layers, seen in lines 55 and 59.

Appendix

I was unsure whether I should include the output of my convolution function for the two kernels, so I included them as an appendix.

Kernel 1

-474 700 314 291 317 177 164 296 302 257 250 250 249 249 249 249 249 249 249 249 249 249 250 250 250 265 321 249 135 264

-545 419 116 16 -84 -59 -45 -96 29 32 -14 -13 -14 -15 -15 -15 -15 -15 -15 -15 -15 -15 -14 -23 4 54 -29 -87 -51 64

-527 507 159 -77 -17 11 14 -18 -63 59 60 12 11 10 9 9 9 9 9 9 9 9 10 28 81 2 -64 7 -17 189

-490 514 68 -59 11 1 1 1 -30 -76 45 53 3 2 1 0 0 0 0 0 0 0 19 72 -5 -78 -8 6 -18 132

-486 254 -24 51 3 -1 3 2 12 -29 -77 49 54 3 2 1 0 0 0 0 0 19 75 -9 -71 0 4 4 -1 25

-494 563 -76 -86 63 -6 6 -1 -1 7 -25 -76 55 56 3 2 1 0 0 0 20 78 -1 -77 -5 8 -2 0 -4 13

-533 518 230 -201 -68 53 -9 -6 -3 -5 3 -40 -85 59 57 3 2 1 0 21 82 -3 -84 -9 -4 -5 -4 -7 9 58

-535 443 137 128 -172 -72 58 -5 -4 -3 -4 5 -40 -88 61 58 3 2 22 84 -6 -93
-8 -3 -4 -2 -5 12 55 -180
-533 454 89 50 150 -181 -83 56 1 2 3 3 3 -32 -91 65 59 24 86 -2 -95 -8
5 2 4 2 11 55 -186 -23
-534 453 87 -19 61 148 -177 -77 53 -1 -1 1 1 3 -42 -96 82 138 -3 -89 -10 4
0 0 -2 8 53 -188 -32 125
-534 453 88 -10 1 56 148 -173 -88 54 -1 -1 1 2 4 -46 -54 29 -89 -11 3 0
0 -3 16 51 -191 -28 124 28
-534 453 87 -11 8 -5 56 151 -177 -90 51 -3 -2 1 2 0 -22 -30 -10 3 0 0 -3
15 49 -195 -25 125 25 5
-534 453 87 -12 7 2 -5 57 153 -174 -85 59 -2 0 3 4 5 14 4 3 3 0 17
49 -198 -22 126 25 2 4
-534 453 87 -12 6 1 2 -5 57 153 -171 -96 53 2 -5 -3 -4 -4 -4 -3 -4 12 42 -
201 -19 127 25 2 1 3
-534 453 87 -12 6 0 1 2 -6 56 177 -161 -90 22 0 2 3 3 3 2 1 20 -189
4 139 25 3 1 0 3
-534 453 87 -12 6 0 0 1 1 26 154 95 9 0 0 -6 -5 -5 -5 -6 2 8 12 155
101 5 3 1 0 3
-534 453 87 -12 6 0 1 1 26 101 8 -128 -25 3 5 6 -2 7 7 -3 6 1 -61 -
110 78 71 5 2 1 3
-534 453 87 -12 6 1 2 14 101 -1 -116 -14 7 -6 -5 -5 -6 -7 -5 -4 -4 -4 4 -
50 -111 76 67 0 2 4
-534 453 87 -12 7 2 8 87 27 -135 -5 3 1 3 2 2 41 44 26 -4 4 3 2 10
-52 -128 107 50 -3 5
-534 453 87 -11 8 8 81 77 -107 -32 4 -1 0 -3 0 42 -86 -197 -2 29 -7 -1 -1
0 10 -75 -58 119 47 0
-534 453 88 -10 14 82 73 -121 -43 10 0 1 -2 8 53 -151 -73 193 -220 -27 38 2
0 0 1 2 -97 -65 118 51
-534 453 87 -6 88 76 -125 -44 0 -1 -1 -3 -2 41 -140 -131 181 146 116 -232 -30
26 -9 -2 -2 1 1 -99 -70 124
-533 454 102 78 82 -131 -49 6 -3 -3 3 -4 49 -133 -134 169 54 0 98 124 -228
-23 33 -3 4 -4 -2 7 -103 -71
-537 453 163 62 -124 -47 8 -1 0 -3 -2 51 -149 -131 170 54 -5 5 0 97 124 -
235 -30 35 -9 -1 -1 1 1 -95
-518 564 177 -157 -39 8 -1 0 -2 -1 51 -139 -130 172 54 -6 2 3 4 0 98 126 -
233 -28 37 -7 0 0 2 7
-476 469 10 -58 17 3 3 0 1 52 -139 -131 171 54 -6 2 1 0 2 4 0 98 126
-233 -28 37 -7 0 1 6
-481 268 -174 6 -14 -23 -21 -16 51 -147 -130 171 54 -6 2 1 0 0 0 2 4 0
98 126 -233 -28 37 -7 0 -7
-451 771 252 8 223 167 166 152 -119 -129 175 57 -3 5 4 3 3 3 3 3 5 7
3 101 129 -230 -24 40 -4 35
-389 324 184 -247 -183 -77 -161 -266 -38 162 47 -13 -4 -5 -6 -6 -6 -6 -6 -6 -6 -4
-2 -7 91 119 -242 -45 56 -6
-272 17 158 120 -146 92 58 -200 364 102 56 55 55 54 54 54 54 54 54 54 54
54 56 57 62 147 173 -148 -11 -163

Kernel 2
10 1231 1114 1119 1120 961 946 1100 1122 1098 1100 1099 1099 1099 1099 1099 1099
1099 1099 1099 1099 1099 1099 1100 1099 1098 1139 1041 925 1013
25 1025 1065 995 847 864 877 833 1000 1020 995 1005 1003 1003 1003 1003 1003
1003 1003 1003 1003 1003 1004 995 1004 1039 925 834 879 920
20 1092 1069 855 889 902 901 892 856 1022 1042 1015 1025 1023 1023 1023 1023
1023 1023 1023 1023 1023 1023 1024 1059 948 850 907 878 1017
28 1078 935 849 906 891 892 894 883 848 1013 1040 1012 1022 1020 1020 1020 1020
1020 1020 1020 1020 1020 1057 946 840 898 893 881 956
62 793 852 938 873 891 888 888 899 879 843 1015 1040 1012 1022 1020 1020 1020
1020 1020 1020 1020 1058 940 843 901 888 889 890 882
24 1131 769 812 934 863 890 879 878 889 877 839 1018 1040 1012 1022 1020 1020
1020 1020 1020 1059 944 832 891 886 879 880 880 874
20 1082 1144 691 820 923 854 873 871 869 880 859 827 1021 1040 1012 1022 1020
1020 1020 1061 940 820 883 869 870 871 871 863 944
23 1053 1051 1101 709 814 921 852 869 867 865 877 856 820 1023 1040 1012 1022
1020 1062 935 808 880 865 867 867 868 860 940 685
22 1064 1048 1019 1116 702 801 916 855 871 869 868 871 860 814 1025 1040 1012
1064 936 803 876 868 869 869 871 855 936 678 898
22 1063 1050 999 1026 1116 704 803 911 850 866 864 864 867 848 809 1024 1078
929 806 871 864 864 864 865 850 932 673 891 1084
22 1063 1049 1010 1014 1024 1116 706 790 908 846 862 860 860 864 841 843 935
805 866 860 860 860 860 854 927 667 893 1084 1013
22 1063 1049 1008 1024 1012 1024 1117 701 785 903 841 858 856 856 859 847 826
861 856 856 856 856 850 922 661 895 1084 1012 1023
22 1063 1049 1008 1022 1022 1012 1024 1118 702 786 907 837 854 853 853 853 863
852 853 853 853 847 918 655 897 1084 1012 1022 1021
22 1063 1049 1008 1022 1020 1022 1012 1024 1118 704 773 896 837 843 843 843 842
843 843 843 839 906 650 899 1084 1012 1022 1020 1021
22 1063 1049 1008 1022 1020 1020 1022 1012 1023 1119 721 762 867 838 841 841 841
841 841 839 868 665 911 1083 1013 1022 1020 1020 1021
22 1063 1049 1008 1022 1020 1020 1020 1021 1020 1068 987 833 825 838 829 831 830
830 830 838 822 864 1050 1043 1013 1022 1020 1020 1021
22 1063 1049 1008 1022 1020 1020 1021 1020 1069 932 747 828 835 833 835 826 835
835 826 834 837 802 774 1025 1044 1012 1022 1020 1021
22 1063 1049 1008 1022 1020 1022 1012 1067 926 749 837 830 821 822 823 822 821
822 824 822 821 834 809 767 1027 1039 1012 1022 1021
22 1063 1049 1008 1022 1022 1012 1057 956 733 841 823 824 825 827 813 843 855
826 816 827 825 823 836 804 757 1059 1031 1012 1023
22 1063 1049 1008 1024 1012 1050 1009 758 816 820 820 820 821 805 870 756 610
851 834 809 821 820 818 833 782 825 1074 1028 1013
22 1063 1049 1010 1014 1050 1011 745 807 826 815 816 817 809 876 693 776 1089
609 823 839 813 817 816 814 831 759 825 1075 1029
22 1063 1050 999 1052 1013 739 803 814 811 811 814 797 862 699 742 1119 1050
1053 602 818 826 801 813 811 810 828 753 819 1078
22 1064 1048 1045 1015 732 797 817 806 806 817 790 867 702 734 1123 1023 1004
1049 1062 602 821 829 802 816 806 804 831 748 813

23 1051 1076 1002 734 798 818 806 808 810 791 869 686 736 1124 1022 1012 1025
1004 1048 1063 595 814 831 795 810 808 805 824 752
18 1127 1050 711 801 818 806 808 810 792 868 694 737 1124 1022 1012 1022 1020
1024 1004 1048 1064 596 814 832 796 810 808 806 826
41 997 829 771 823 807 809 811 792 869 694 736 1124 1022 1012 1022 1020 1020
1020 1024 1004 1048 1064 596 814 832 796 810 808 808
65 799 661 850 783 799 798 782 871 686 737 1124 1022 1012 1022 1020 1020 1020
1020 1020 1024 1004 1048 1064 596 814 832 796 811 799
13 1259 968 743 936 870 885 919 681 739 1125 1023 1013 1023 1021 1021 1021 1021
1021 1021 1021 1025 1005 1049 1065 597 815 834 795 821
71 819 901 485 530 622 560 539 785 1115 1015 1006 1016 1014 1014 1014 1014 1014
1014 1014 1014 1014 1018 998 1041 1058 591 795 837 830
127 463 741 745 421 635 670 509 1184 1039 1038 1039 1038 1038 1038 1038 1038
1038 1038 1038 1038 1038 1038 1041 1030 1065 1074 649 831 627