Final Exam

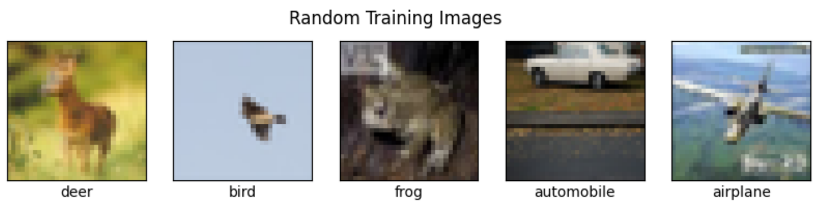
Joe Shymanski

May 3, 2022

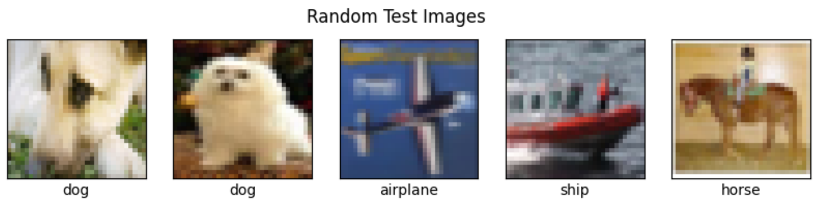
1 Question 1

All data was normalized between 0 and 1.

1.1 a



(a)



(b)

Figure 1: Random images from CIFAR-10 data set

1.2 b

Components:

```
[[ 0.02683309  0.02796728  0.03068162 ...  0.01685011  0.0174886
   0.01913861]
 [-0.01331717 -0.01572706 -0.01840725 ...  0.02451899  0.02341633
   0.02394079]
 [-0.01907806 -0.01777064 -0.01515922 ... -0.02054518 -0.01925073
  -0.01583303]
 ...
 [ 0.01585476  0.01858785  0.01453274 ...  0.01122816  0.01631268
   0.01773613]
 [ 0.00783289  0.01970625  0.02687812 ... -0.01493002 -0.00406169
  -0.01751752]
 [-0.00529874 -0.00483493 -0.01382035 ... -0.01237675 -0.00903651
  -0.00199737]]
```

Variation explained ratio per component:

```
[0.29322478 0.11188372 0.06871613 0.03787763 0.03672929 0.03110855
 0.02505076 0.02161501 0.02121662 0.01629588 0.01390965 0.01134555
 0.01070742 0.010156   0.00928799 0.00877039 0.00802561 0.00766501
 0.00743857 0.00682502 0.00618252 0.00561294 0.00520935 0.00498732
 0.0048207  0.0046245  0.00454447 0.00431502 0.00426929 0.00399022
 0.00357593 0.00355408 0.00343434 0.00327653 0.00320925 0.00296224
 0.00272837 0.00268145 0.00259115 0.00247102 0.00240025 0.00234498
 0.00230033 0.00223637 0.00217373 0.00214357 0.00199422 0.00198208
 0.00193471 0.00190456 0.0018546  0.00181029 0.00179442 0.00175792
 0.00172242 0.00165052 0.00158631 0.00158234 0.00155935 0.00155003
 0.00149835 0.00145711 0.00142211 0.00140444 0.00138574 0.00136638
 0.00133613 0.00131907 0.00130208 0.00129406 0.00124192 0.00121662
 0.00120807 0.00117642 0.00115986 0.0011277  0.00111932 0.00109491
 0.00105623 0.00104495 0.00102457 0.0010007  0.00098762 0.0009834
 0.00096552 0.00093599 0.00092894 0.00091714 0.00089934 0.00089314
 0.00088171 0.00087449 0.00084907 0.00083015 0.00082007 0.00081273
 0.00080502 0.00080052 0.0007855  0.00077259 0.00075046 0.00074208
 0.00072734 0.00071802 0.00070957 0.00070119 0.00068193 0.00067836
 0.00067271 0.00065689 0.0006463  0.00064022 0.00062657 0.00062082
 0.00060778 0.00060359 0.00058301 0.00057805 0.00057132 0.00056752]
```

Total variation explained ratio: 0.9312566707159597

Pooling	Layers	Accuracy	Loss
Max	2	0.6929	0.8912
Max	3	0.7128	0.8352
Max	4	0.7188	0.8283
Average	2	0.6564	0.9915
Average	3	0.6815	0.9219
Average	4	0.6242	1.0525

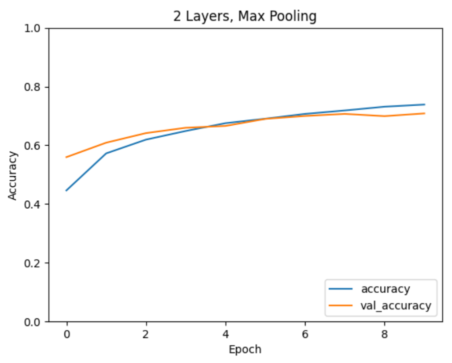
Table 1: Test performance for different CNN structures

1.3 c

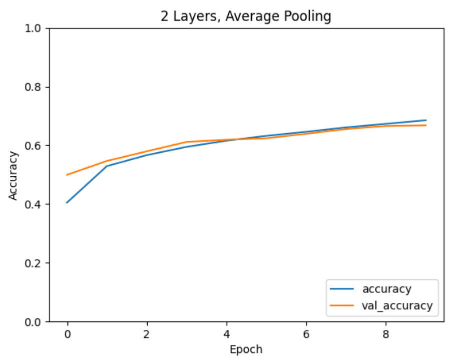
The CNNs tested consisted of two, three, and four layers. No matter the number of layers, the first convolutional layer contained 32 filters and each subsequent convolutional layer contained 64 filters. All convolutional layers used a 3×3 kernel and ReLU as the activation function. A pooling layer was inserted after every convolutional layer with a pool size of 2×2. After the last convolutional layer, a flattened layer was inserted followed by a dense output layer of 10 units and softmax activation. In order to ensure that the network with four layers did not run out of image dimensions before the final layer, each convolutional layer specified `padding="same"` for all networks. Each model was then compiled with the Adam optimizer, sparse categorical cross-entropy loss function, and accuracy metric. Finally, each model was trained using a batch size of 128, 10 epochs, and a 10% validation split.

According to Figure 2 and Table 1, it appears that max pooling performs better than average pooling across all CNN structures. Average pooling may show slightly less overfitting than max pooling, but the latter still outperformed the former on the test set. Max pooling likely does better since it only retains the starkest features from the image. This is useful for this kind of classification problem due to the sheer difference between the 10 classes; dog and airplane should theoretically have no features in common whatsoever. Average pooling may create some scenarios where the starkest, differentiating features are combined with insignificant ones, blurring the lines between the classes.

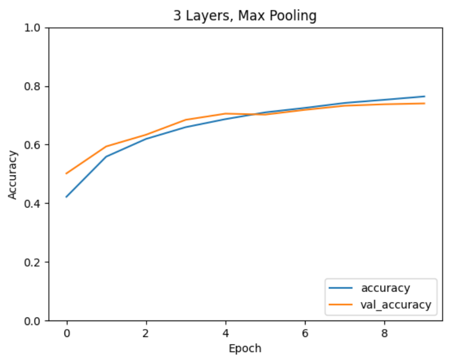
The model with 4 convolutional layers yielded the best results under max pooling. This may be due to the fact that a wider range of filters needed to be learned since the 10 classes contain vastly different features. Effectively piecing together these features takes more layers to do so.



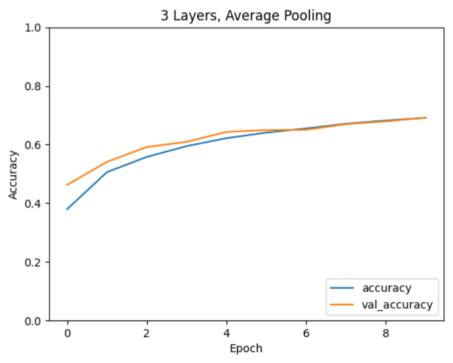
(a)



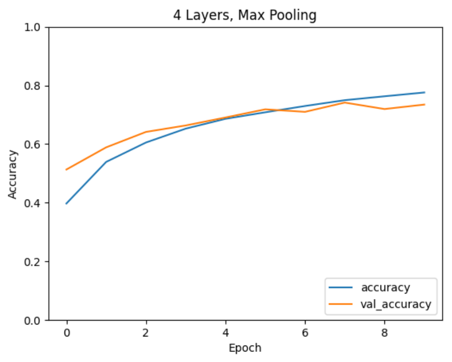
(b)



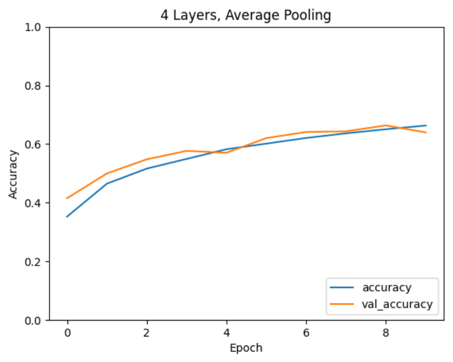
(c)



(d)



(e)



(f)

Figure 2: Training performance for different CNN structures

k	Layer	MSE	MAPE
3	LSTM	6.6031	30.5396
3	GRU	6.6688	30.8437
5	LSTM	5.8329	21.0585
5	GRU	6.3486	21.8717
7	LSTM	6.0237	28.5586
7	GRU	5.5061	27.9682
9	LSTM	6.5186	23.7990
9	GRU	6.9184	24.3292

Table 2: Test performance for different values of k

2 Question 2

2.1 a

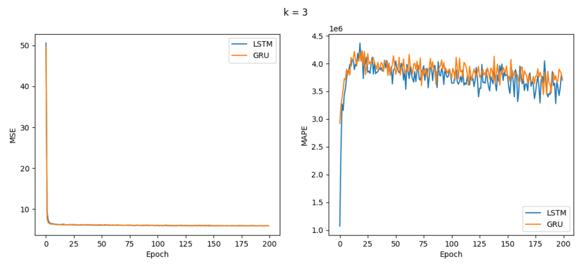
The code takes a random 20% of the samples of size k as the test set and the rest as the training set.

2.2 b

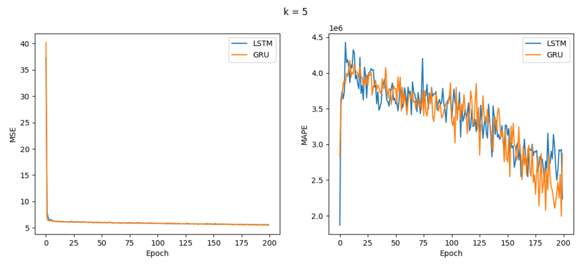
The LSTM created for this problem consists of a single LSTM layer of 50 units with ReLU activation connected to a dense output layer with a single unit. The model is then compiled with the Adam optimizer, mean squared error (MSE) loss function, and mean absolute percentage error (MAPE) metric. Finally, each model was fitted using 200 epochs for $k \in \{3, 5, 7, 9\}$. Since the number of samples depends on k, the training and test sets were recalculated with each value of k.

2.3 c

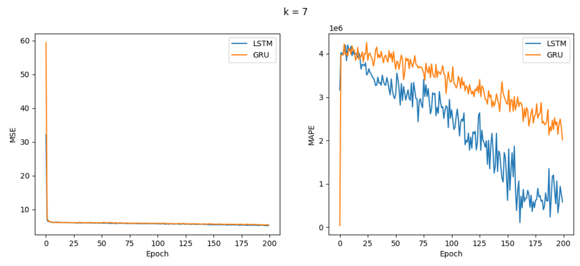
Both LSTMs and GRUs are gated units in RNNs. The difference is that LSTMs have three gates (input, output, and forget) while GRUs have two (reset and update). Since GRUs have less parameters, they are faster than LSTMs, though they may underperform for larger sequences. Figure 3 demonstrate how the two units perform similarly when the time sequence samples are fairly short. LSTM starts to pull ahead when k increases.



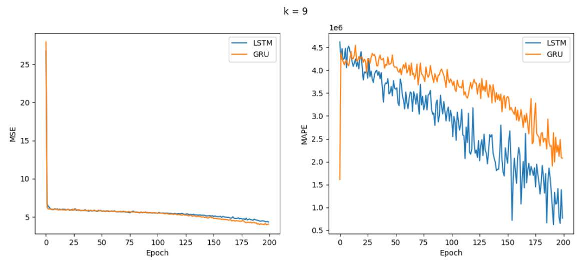
(a)



(b)



(c)



(d)

Figure 3: Training performance for different values of k