Conor Gagliardi ML A4 Writeup
Q1)

Accuracy Table of Linear Models with various Learning rates			
		Learning Rates	
	0.5	0.25	0.125
Ls (Linear Softmax Normalization)	91.79% 0.6702s	91.56% 0.6720s	91.13% 0.6748s
LrLs (Linear ReLu activation with Linear Softmax Normalization)	9.80% 1.8193s	96.56% 1.7848s	95.89% 1.7820s
L2s (2 Linear Layer without activation between)	9.80% 1.6669s	9.80% 1.6572s	91.50% 1.6393s

Ls is a linear model with a single layer, whereas LrLs and L2s each have two layers. LrLs with a learning rate of 0.5 and L2s with learning rates of 0.5 and 0.25 are of interest. These results indicate that the models failed to converge with the specified learning rates, resulting in an accuracy of approximately 10% (possibly picking 0 by default when not converging, which may account for 9.8% of the testing data in this instance). The models failed when the learning rate was increased because the number of parameters used were greater than in Ls. To account for the higher number of parameters, Charniak wrote, “to reflect the larger number of parameters, we need to lower the learning rate by a factor of 10.” Also said in Wk9-Lecture1, it is preferable to begin with a lower learning rate because an excessively high learning rate will result in “jumps into distant parts of loss function space and not follow the loss surface.” In light of this, it is evident that a "sweet spot" for the learning rates was 0.5 for Ls, 0.25 for LrLs, and 0.125 for L2s was the only one of the three that succeeded. L2s only worked with the lowest learning rate because, in the absence of an activation function (RELU in these linear examples) the implicit calls to compute the forward pass are effectively performing two forward passes for every one backward pass (like LrLs) without an activation function in between. The first of which has more parameters than LS, and with larger learning rates, this causes the gradient to jump into distant regions of the loss function space. Charniak also described how the two-layer solution without an activation function could be implemented with a single layer due to matrix math properties, therefore the resulting accuracy of Ls and L2s will be fairly similar. LrLs has the highest accuracy because additional layers permit increasingly complex data generalization approximations (The difference of using a single line to identify a number vs. 2 lines in this case).

Ls takes the least amount of time due to its single forward pass and lack of activation function application (less math operations than the others). Both two-layer models have more initial parameters. L2s calculates two forward passes and joins the results with matrix multiplication, resulting in approximately double the time required by Ls plus a little amount of additional time for matrix multiplication. Lastly, LrLs applies a RELU in addition to two layers of operations, causing it to take the most time. Between different learning rates, each model takes roughly the same amount of time, differing slightly by learning rates affecting time needed to converge (or not converge).

Q2)

Accuracy Table of Convolutional Models with Various # of Batches			
		# of Batches	
	1000	2000	4000
CrLs (Convolutional ReLu activation with Linear Softmax Normalization)	92.18% 2.2158s	94.07% 4.2144s	96.65% 8.3364s
CrCLs (Convolutional ReLu activation with Convolutional and Linear Softmax Normalization)	97.61% 7.5583s	97.72% 14.7426s	98.30% 29.2466s

As the number of batches grows, the trend in accuracy is also growing. 1000 is the default number of batches. Comparing convolutional models to linear models involves comparing the column of 1000 batches to the table column learning rate of 0.125 for the linear model. In this instance, the model with the highest performance is CrCLs, followed by LrLs, CrLs, L2s, and Ls. In general, however, the convolutional models yielded greater accuracy than the linear models.

The parameters of the models CrLs and Ls have “essentially remained constant” different by roughly 64 due to the weights from the filters in CrLs The weights in the first layer of each is set to be 784 x 10. LrLs and L2Ls have the most parameters because they are dense networks in which every node in one layer is connected to every node in the following layer. Additionally, the initialization of the first layer in double linear models is 784 x 784, which is the largest initialization of any model. CrCLs uses more parameters than CrLs but less than LrLs and L2Ls. Its initialized starting weight was 1568 x 10. Each model's trainable parameters were calculated as follows: Ls-8,634 LrLs-624,858 L2s-624,858 CrLs-8,728 (+10 because the bias is not kept in a tf.Variable) and CrCLs-19,706 (+10 for the same reason). These results validate the estimated parameters. Despite the differences in parameters, Convolutional models took consistently longer to train due to the use of multiple filters which are used to split up the data into nearby areas, specifically, the area of the kernel calculations.. Each increase of double the number of batches intuitively doubled the total time taken for training. The roughly doubling of the parameters from CrLs to CrCLs along with the additional network layer, and adding the max pooling function, resulted in the time taken being about triple.

Q3)

Accuracy Table of Convolutional Models with 3 Rounds				
		Metrics		
	Maximum Accuracy	Minimum Accuracy	Average Accuracy	Standard Deviation
Ls	92.04% 1.9411s	91.13%	91.60%	0.46%
LrLs	97.15% 5.0646s	95.89%	96.61%	0.65%
L2s	9.80% 4.6951s	9.80%	9.80%	0.00%
CrLs	95.83% 5.9941s	92.49%	94.44%	1.74%
CrCLs	98.42% 23.0361s	97.68%	98.12%	0.39%

The average accuracies and time taken are consistent with previous models overall. Compared to specific hyperparameters from the average of the 3 round test (1000 batches, 0.125 learning rate, 1 round), Ls differs by 0.47%, LrLs differs by 0.72 %, L2s fails to converge with 3 rounds, CrLs differs by 2.26%, and CrCLs differs by 0.51%. When comparing the minimum accuracy from the 3-round test, which reflects the first round in every case, the accuracies are similar with minor variations due to tensorflow reshuffling the testing data (for linear models, the minimum accuracy is identical because they aren't using reshuffled tests). Due to three rounds of training as opposed to one cycle for the prior tests, all execution timings are approximately three times longer than those of the individual model tests.

The variance across multiple rounds is due to the fact that the accuracy of all models is lowest in the first round and increases with each subsequent round. For CrLs, the improvement in accuracy was the largest, resulting in the greatest standard deviation. This relates to the amount of training necessary for a model to approach its best estimate of data generalization. The models with the smallest standard deviation are the most stable. The most stable model is CrCLs, which improves its accuracy the least per round. As the number of rounds increases and the model's accuracy subsequently increases, the standard deviation would decrease as more results neared the model's peak accuracy. However, if the number of rounds causes overfitting, the accuracy may decrease, leading the standard deviation to increase.

Bonus) Re-do Q2 experiment

(FASHION MNIST VERSION) Accuracy Table of Convolutional Models with Various # of Batches (Learning rate 0.125, 1 round, random seed of 7)				
		# of Batches		
	1000	2000	4000	6000
CrLs (Convolutional ReLu activation with Linear Softmax Normalization)	82.7701% 13.1011s	84.6260% 15.0004s	85.1929% 19.1493s	86.7348% 8.3364s
CrCLs (Convolutional ReLu activation with Convolutional and Linear Softmax Normalization)	86.248% 30.9709s	87.2401% 38.9215s	87.7104% 55.7553s	89.1885% 71.1311s

In this experiment for fashion_mnist the dependent variable was the number of batches. Independent variables were the models used (CrLs, and CrCLs), Learning rate (0.125), number of rounds (1), and the random seed (7 - apparently it's lucky). Similarly to question 2, as the number of batches increased the resulting accuracy of either model increased. Unlike regular MNIST, Fashion_MNIST appears to be more ambiguous and It is likely difficult to get scores that approach regular MNIST accuracy scores. This is likely due to the complexity of information from an article of clothing versus a 2d drawn number.

Similar to question 2, the CrCLS takes proportionally the same amount of time more than CrLs ("The roughly doubling of the parameters from CrLs to CrCLs along with the additional network layer, and adding the max pooling function, resulted in the time taken being about triple."). Additionally, the number of batches is proportionally adding batches to calculating, adding time for every extra batch calculation.

If I were to repeat this experiment, I would estimate that adding additional rounds of testing in addition to a large batch size would yield the best accuracy.]