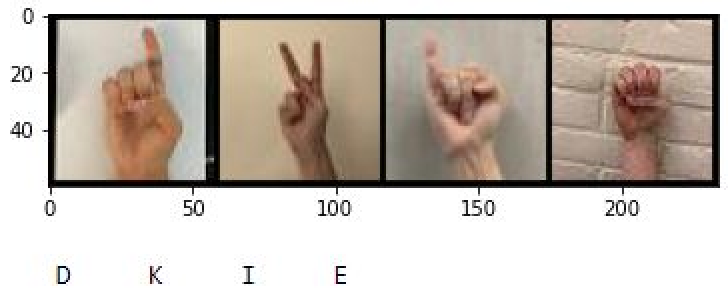
Esmat Sahak
ECE324

Assignment 4: Classification of ASL Images

2. Input Dataset

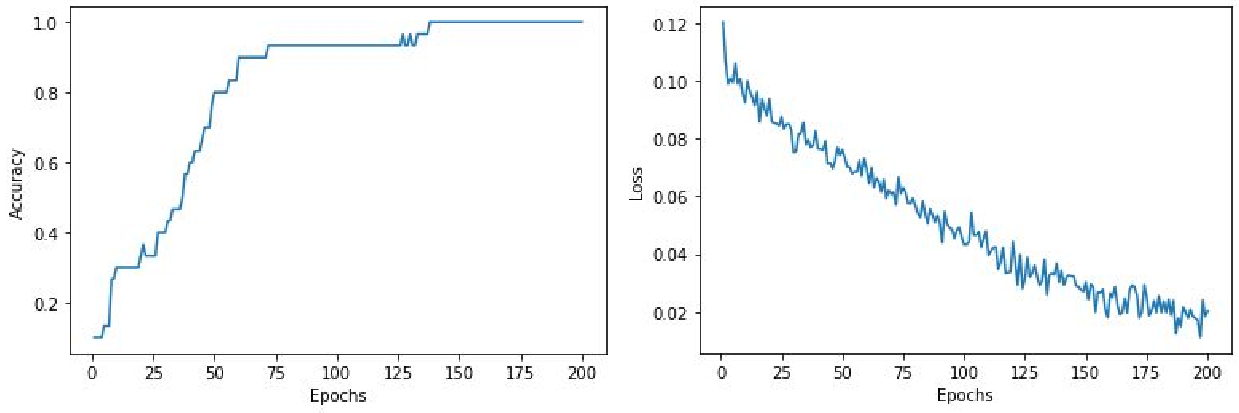
Print out and record those same four images, along with their ground-truth labels.



3. Model Building and Testing

Plots of loss and accuracy (of this training data only) vs. epoch.

Epochs = 200, Batch size = 10, Learning rate = 0.01, Seed = 0



The execution time of the full training loop, measured using the time library.

Time: 26.001931s

The number of epochs required to achieve 100% accuracy on the training data.

Epochs = 150

The size of this network, in terms of: 1) The total number of parameters to be trained, and 2) the amount of computer memory required to store all of the parameters (weights and biases) in all layers of the network. T

Number of parameters to be trained: 98,830

Amount of computer memory: 0.38 Mb

Esmat Sahak
ECE324

4.1. Data Loading and Splitting

Which method is better? Why?
The first method of putting some of everyone's samples into both the training and validation set is better as the training set is more representative of the entire set which will in turn improve training and validation accuracies.

What fraction of train/validation split did you use? Why?
I used a fraction of 0.33 such that for every three pictures taken per person of a letter, it is more likely that 2 of them are represented in the training set while the remaining picture is represented in the validation set.

Report the mean and standard deviation of the image training data that you use.
Mean: tensor([0.0027, -0.0007, -0.0027])
Standard Deviation: tensor([0.8384, 0.8899, 0.8775])

4.3: Hyperparameter Search

Indicate which 12 combinations you used, and explain why you chose them.
I started with one layer and adjusted other hyperparameters before adjusting the number of convolutional layers as I hypothesized that the number of convolutional layers would have the most impact on accuracy. This would help me converge to the best set of the remaining variables (last combination).

Epochs = 100*, Seed = 0, Test Size = 0.2*, Kernel Size = 3, Loss Function = MSE, Activation Function: ReLU									
Batch Size	**Learning Rate**	**Conv. Layers**	**Number of Kernels**	**First Layer Neurons**	**Validation Accuracy**	**Training Time (s)**	**Minimum Epochs**	**# of Parameters**	**Memory (Mb)**
32	0.01	1	10	32	0.786370	472.78	70	233,922	0.89
32	0.1	1	10	32	0.768021	445.24	30	233,922	0.89
4	0.01	2	30	32	0.871560	1595.99	40	147,572	0.56
32	0.1	2	10	32	0.842726	590.03	60	47,632	0.18
32	0.01	2	10	8	0.688073	1154.19	> 200	12,808	0.05
4	0.1	2	10	32	0.840105	620.96	20	47,632	0.18
4	0.01	2	30	8	0.829620	2397.65	70	43,628	0.17
4	0.1	4	10	32	0.834862	887.56	40	3,692	0.01
32	0.01	4	30	32	0.753604	2939.19	> 200	26,522	0.10
4	0.01	4	30	32	0.889908	1749.23	80	26,552	0.10
32	0.01	4	10	32	0.471822	1239.92	> 200	3,692	0.01
4	0.01	4	30	32	0.889908	1656.68	90	26,552	0.10

* **Note:** for three cases, 200 epochs were used but did not converge within 200 epochs regardless.
* Test size was refined to obtain higher training accuracies while maintaining an adequate size for the validation set.

Esmat Sahak
ECE324

Comment on the relative merits of different networks with respect to these metrics.

Batch size: A smaller batch size reduces the number of gradient calculations improving runtime.

Learning rate: A learning rate of 0.01 is more accurate but the steps are lower so it may take more steps to reach the minimum.

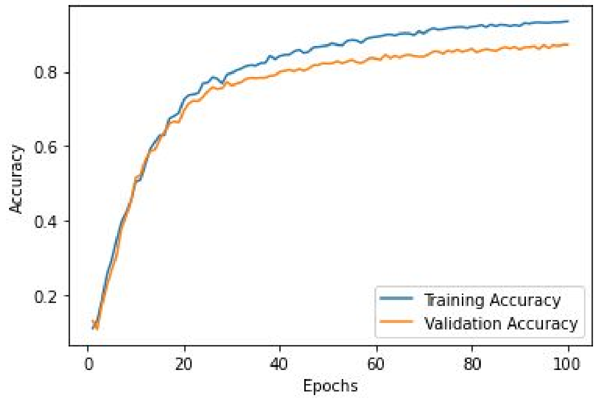
Convolutional layers: More layers usually give more accuracy and reduce the number of parameters, saving memory.

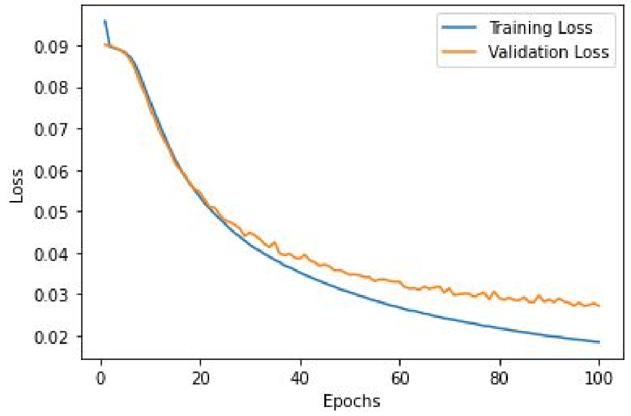
Number of kernels: Higher values generally increased the accuracy but at the cost of more memory.

First layer neurons: More neurons means more parameters consuming memory but usually increases accuracy.

For the network that you select as best, provide the loss and accuracy vs. epoch plots, for both the training data and validation data.

Conv. Layers = 4, Batch Size = 4, Learning Rate = 0.01, Number of Kernels = 30, First Layer Neurons = 32





5. Batch Normalization and Cross Entropy Loss

Give all of the same metrics reported above in a table.

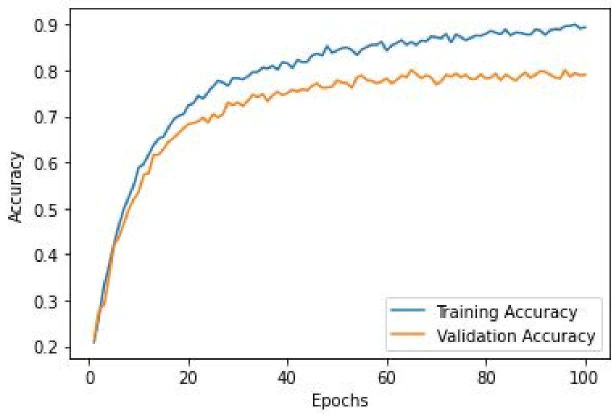
Epochs = 100, Seed = 0, Batch Size = 4, Test Size = 0.2, Kernel Size = 3 *					
	Validation Accuracy	**Training Time (s)**	**Minimum Epochs**	**# of Parameters**	**Memory (Mb)**
Batch Normalization	0.790301	2782.18	65	26,856	0.10
Cross Entropy Loss	0.906946	1893.02	50	26,552	0.10
Both BN and CEL	0.847287	2711.03	60	26,856	0.10

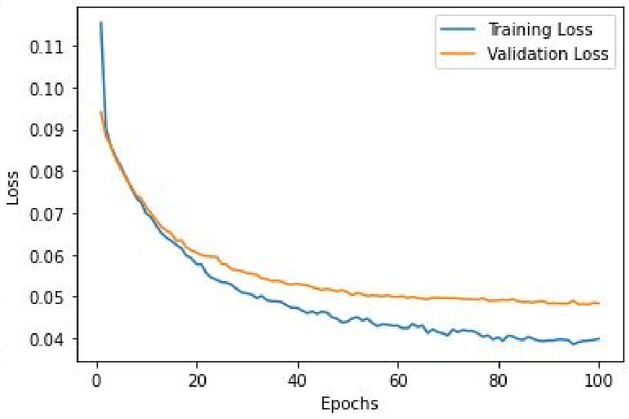
* Hyperparameters in addition to those listed as the best

Esmat Sahak
ECE324

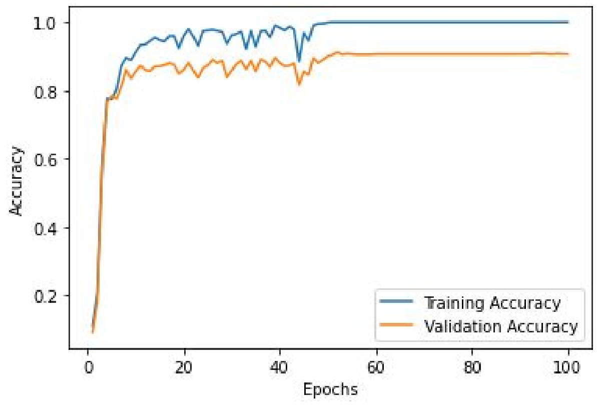
Provide both the loss and accuracy plots for training and validation.

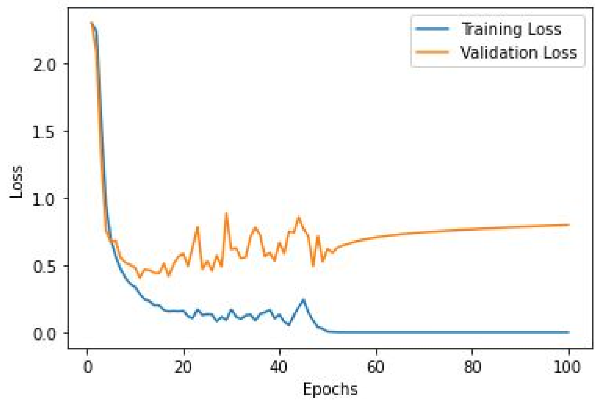
Batch Normalization



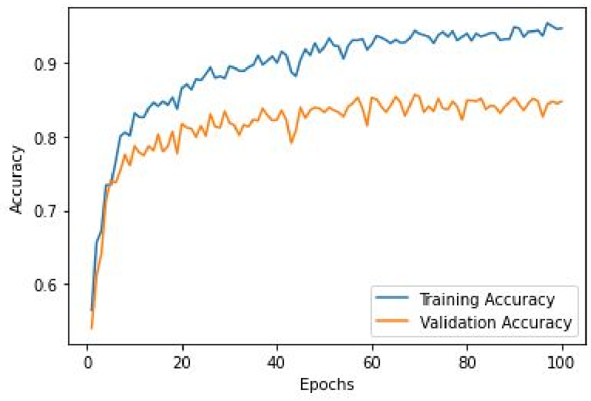


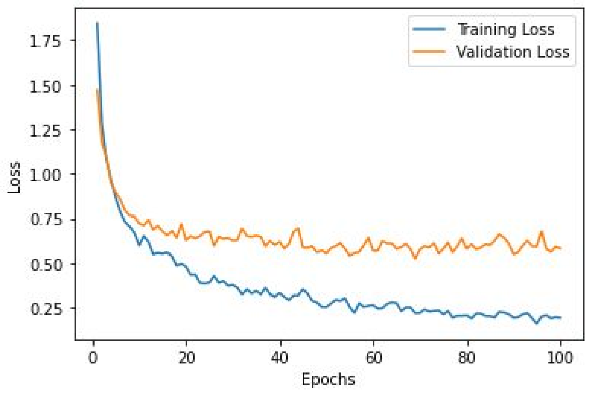
Cross Entropy Loss





Batch Normalization and Cross Entropy Loss





Discuss the effect of these two changes on the results.

Batch normalization: Stabilized the fluctuation in the loss and accuracy plots in a fewer number of epochs. More effective for larger batches in terms of improving accuracy: in test run with second last combination listed on table (0.47 validation accuracy), batch normalization resulted in 0.8 validation accuracy in the same number of epochs.

Esmat Sahak
ECE324

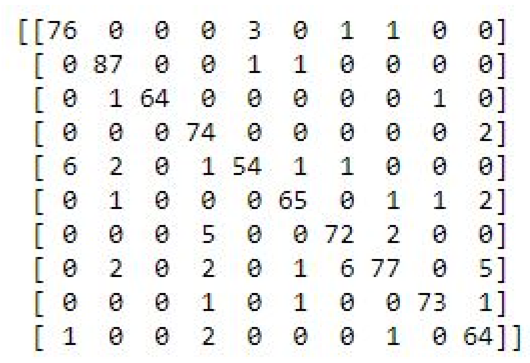
Cross entropy loss: Increases accuracy measures in fewer epochs but is less stable than MSE loss. The lower stability is due to harsher punishments for incorrect gradient steps. MSE loss is more suitable for regression type problems where the goal is to come close to a result as it has much lower loss values.
Batch normalization and cross entropy loss: Reduces the fluctuations due to the cross entropy loss function while combining their advantages.

6. Final Best Networks and Confusion Matrix

Hyperparameters for MyBest Network.
Batch Size = 4, **Learning Rate** = 0.01, **Epochs** = 29, **Seed** = 0, **Test Size** = 0.2, **Conv. Layers** = 4, **Kernels** = 50, **First Layer Neurons** = 16, **Kernel Size** = 3, **Loss Function** = cross entropy loss, **Activation Function** = ReLU, **Batch Normalization** = False
Validation Accuracy = **0.925295**, Training Accuracy = **1**

Hyperparameters for MyBestSmall Network.
Batch Size = 4, **Learning Rate** = 0.01, **Epochs** = 54, **Seed** = 0, **Test Size** = 0.2, **Conv. Layers** = 4, **Kernels** = 11, **First Layer Neurons** = 60, **Kernel Size** = 3, **Loss Function** = cross entropy loss, **Activation Function** = ReLU, **Batch Normalization** = False
Total Parameters = **4938**, Validation Accuracy = **0.861075**, Training Accuracy = **0.937418**

For your most accurate network, provide the confusion matrix.



```
[[76  0  0  0  3  0  1  1  0  0]
 [ 0 87  0  0  1  1  0  0  0  0]
 [ 0  1 64  0  0  0  0  0  1  0]
 [ 0  0  0 74  0  0  0  0  0  2]
 [ 6  2  0  1 54  1  1  0  0  0]
 [ 0  1  0  0  0 65  0  1  1  2]
 [ 0  0  0  5  0  0 72  2  0  0]
 [ 0  2  0  2  0  1  6 77  0  5]
 [ 0  0  0  1  0  1  0  0 73  1]
 [ 1  0  0  2  0  0  0  1  0 64]]
```

Looking at the confusion matrix, which two classes were the most confused for your model? Why?
There were two pairs of classes that were the most confused: A and E, G and H. The confusion is due to very similar hand signs for these pairs of letters: A and E can only be distinguished by the positioning of the thumb while G and H can only be separated based on the positioning of the middle finger.