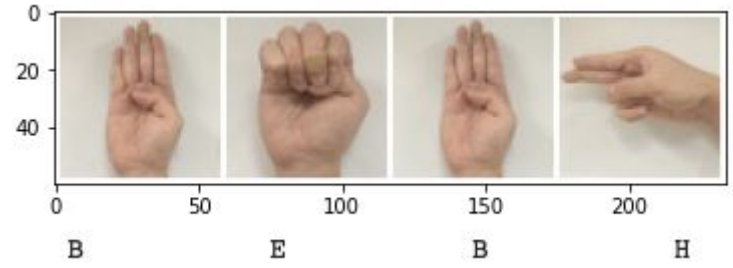
Project Report

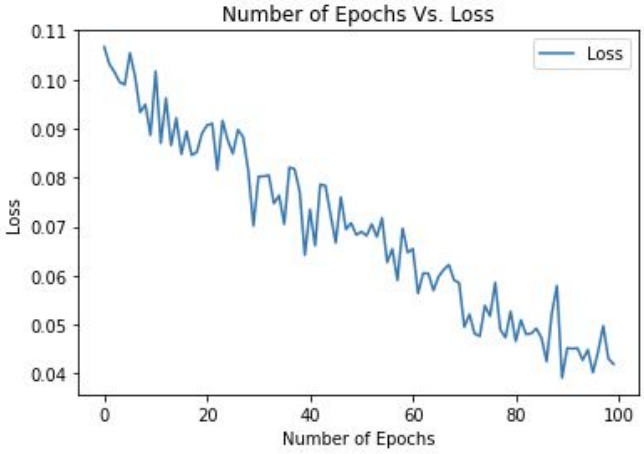
Input Datasets

Ground truth checking for a batch size of 4

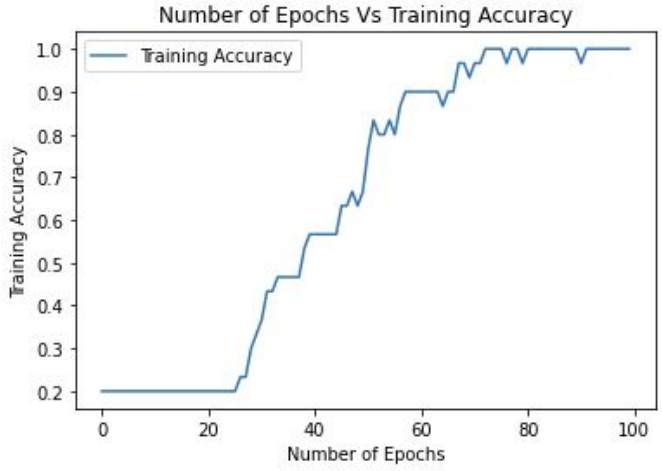


Model Building and Testing

Loss vs. Epochs



Accuracy vs. Epochs



Execution Time: 11.084 s

Number of epochs required to achieve 100%: ~72
Size of network --- Total number of parameters: 104,934
--- Memory required (Estimated total size): 0.59 MB

Using the Full Data Set

Data Loading and Splitting

1. Putting everyone's samples into both training and validation sets would be better. This is because we want to share the randomness of the datasets between both training and validation data so that the overall model would be more generalized. If some people's images are only put into training and some others' images are put in the validation set, the model will then have a higher bias due to the lack of randomness from the other set of data that is in the dataset.

2. I used a 80-20 split between training and validation. This is because the dataset is fairly average in size, thus there is no need to accommodate for the lack of validation images by splitting it with a high ratio; vice versa.

3. ```
   mean = [0.7458459809779424, 0.7445457143567186,
   0.7422643455354258]
   std = [0.11261820096134469, 0.11160372862894079,
   0.1122811851077723]
   ```

Default hyper-parameters

- *Activation Function:* ***ReLU***
- *Optimizer:* ***optim.SGD***
- *Batch Normalization:* ***No***
- *Loss Function:* ***MSELoss***
- *Size of kernels on each convolutional layer:* ***3 by 3***
- *Number of fully connected layers:* ***2***

Combination 1		**Result for combination 1**	
Number of Conv. layers	2	Validation Accuracy	0.842
Number of kernels on each conv. layer	10	Training Execution Time	377.58 s
Number of neurons on	32	Number of Epochs	30
   ```

first layer		required	
Learning rate	0.1	Number of parameters	56,000
Batch size	4	Memory required	0.59 MB
Combination 2		**Result for combination 2**	
Number of Conv. layers	2	Validation Accuracy	0.353
Number of kernels on each conv. layer	10	Training Execution Time	369.69 s
Number of neurons on first layer	8	Number of Epochs required	30
Learning rate	0.1	Number of parameters	14,804
Batch size	4	Memory required	0.43 MB
Combination 3		**Result for combination 3**	
Number of Conv. layers	2	Validation Accuracy	0.840
Number of kernels on each conv. layer	10	Training Execution Time	523.89 s
Number of neurons on first layer	32	Number of Epochs required	50
Learning rate	0.1	Number of parameters	56,000
Batch size	32	Memory required	0.59 MB
Combination 4		**Result for combination 4**	
Number of Conv. layers	2	Validation Accuracy	0.866
Number of kernels on each conv. layer	10	Training Execution Time	656.10 s
Number of neurons on first layer	32	Number of Epochs required	50
Learning rate	0.01	Number of parameters	111,824
Batch size	4	Memory required	0.8 MB
Combination 5		**Result for combination 5**	
Number of Conv. layers	2	Validation Accuracy	0.842

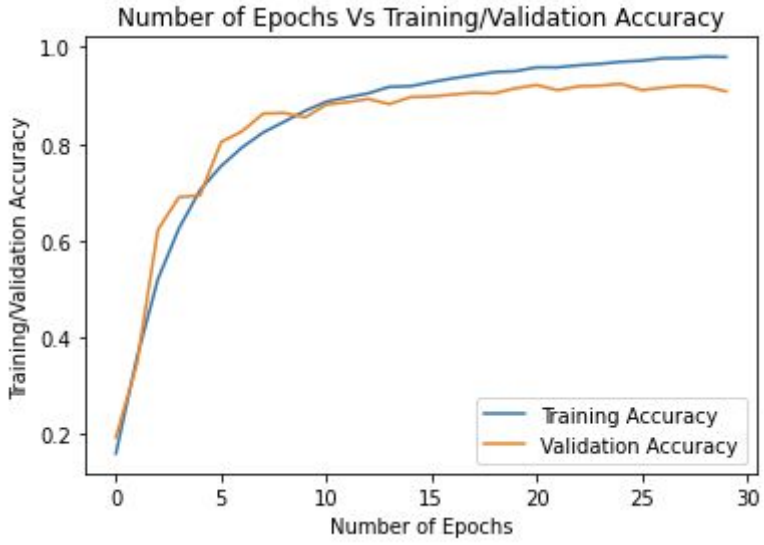
Number of kernels on each conv. layer	30	Training Execution Time	511.80 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	58,360
Batch size	4	Memory required	1.15 MB
Combination 6		**Result for combination 6**	
Number of Conv. layers	1	Validation Accuracy	0.796
Number of kernels on each conv. layer	30	Training Execution Time	499.36 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	701,410
Batch size	4	Memory required	3.55 MB
Combination 7		**Result for combination 7**	
Number of Conv. layers	1	Validation Accuracy	0.757
Number of kernels on each conv. layer	10	Training Execution Time	336.19 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	234,290
Batch size	4	Memory required	1.21 MB
Combination 8		**Result for combination 8**	
Number of Conv. layers	4	Validation Accuracy	0.813
Number of kernels on each conv. layer	10	Training Execution Time	393.44 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	5,020
Batch size	30	Memory required	0.43 MB
Combination 9		**Result for combination 9**	

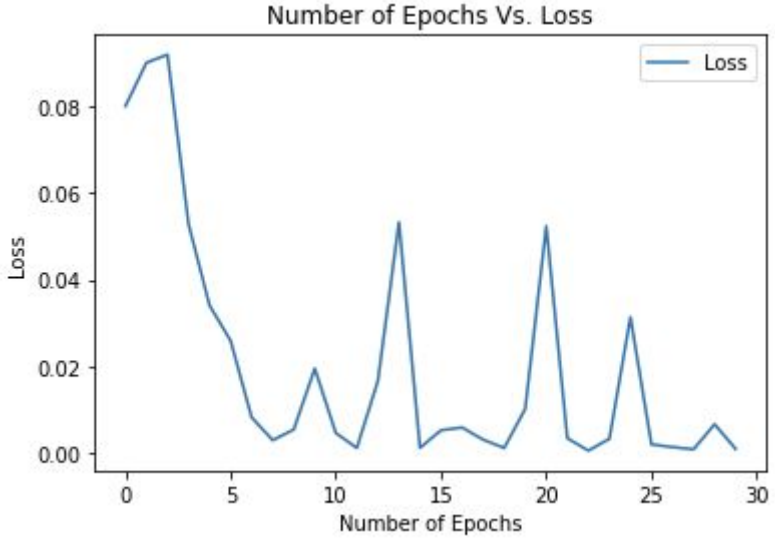
Number of Conv. layers	4	Validation Accuracy	0.908
Number of kernels on each conv. layer	30	Training Execution Time	586.551 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	29,800
Batch size	4	Memory required	1.19 MB
Combination 10		**Result for combination 10**	
Number of Conv. layers	4	Validation Accuracy	0.775
Number of kernels on each conv. layer	30	Training Execution Time	487.738 s
Number of neurons on first layer	32	Number of Epochs required	30
Learning rate	0.1	Number of parameters	29,800
Batch size	32	Memory required	1.19 MB
Combination 11		**Result for combination 11**	
Number of Conv. layers	4	Validation Accuracy	0.751
Number of kernels on each conv. layer	30	Training Execution Time	1023.012 s
Number of neurons on first layer	32	Number of Epochs required	50
Learning rate	0.01	Number of parameters	29,800
Batch size	4	Memory required	1.19 MB
Combination 12		**Result for combination 12**	
Number of Conv. layers	4	Validation Accuracy	0.462
Number of kernels on each conv. layer	30	Training Execution Time	779.427
Number of neurons on first layer	8	Number of Epochs required	40
Learning rate	0.1	Number of parameters	26,284

Batch size	4	Memory required	1.18

Overall, the above-illustrated 12 models utilized combinations of different hyper-parameters in order to explore potential optimal results. In doing so, there are the following observations: 1) With only 1 CNN layer, the validation result seems to be capped at a ceiling right below 80%, other hyper-parameters don't seem to affect the results that much; 2) With 2 CNN layers, overall decent validation accuracies can be achieved, except for the case of Combination 2 where the number of neurons in the first linear layer is only 8 and the batch size is also kept at a small value of 4; the other hyper-parameters such as number of kernels and learning rates do not seem to have that much of an impact on the final model's accuracy; 3) With 4 CNN layers, the best model is produced (see Combination 9 for specific hyper-parameters) with a final validation accuracy of 0.908. The models take longer to train due to the increased number of layers. Furthermore, other hyper-parameters seem to have a larger impact on the accuracy of the model compared to 1 and 2 layers of CNN. Specifically, when increased to batch size from 4 to 32 while keeping the remaining hyper-parameters the same, the model's accuracy went down significantly. This is also the case when the number of first layer neurons is decreased from 32 to 8.

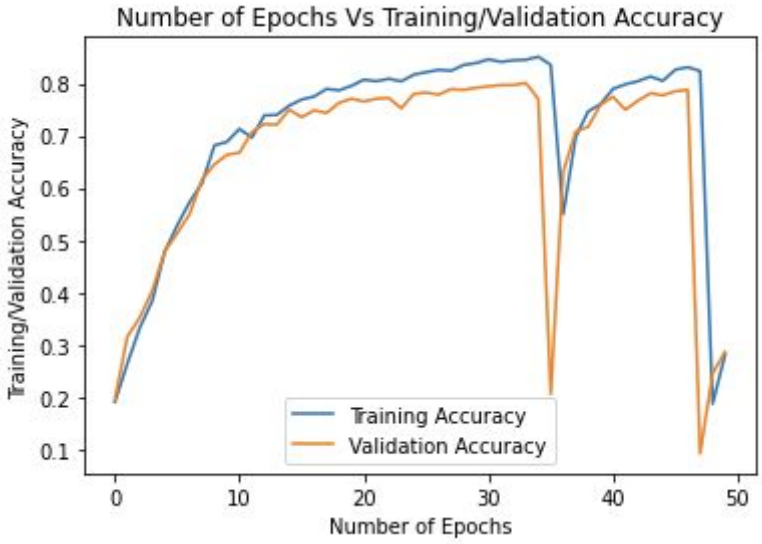
Best model plots

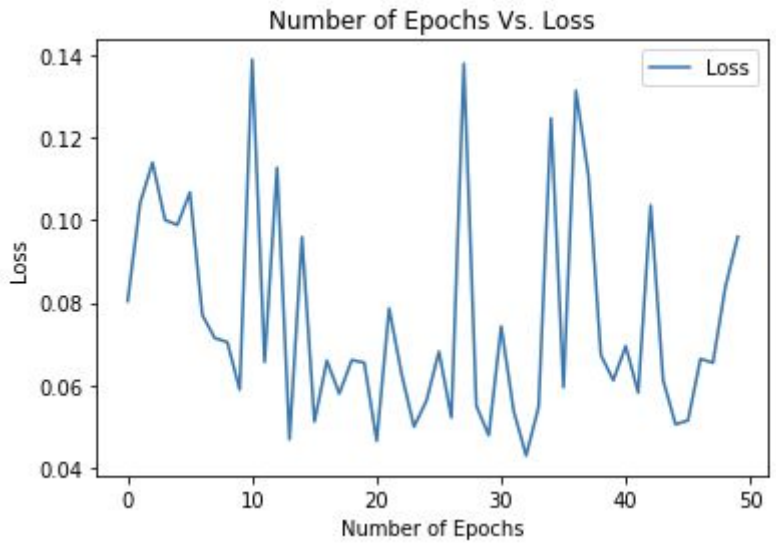




Batch Normalization and Cross Entropy Loss Function

1.





Batch Normalization Only		**Results**	
Number of Conv. layers	4	Validation Accuracy	0.287 (final epoch)
Number of kernels on each conv. layer	30	Training Execution Time	1383.36 s
Number of neurons on first layer	32	Number of Epochs required	50
Learning rate	0.1	Number of parameters	30,136
Batch size	4	Memory required	2.03 MB

Best small model summary:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 10, 54, 54]             280
         MaxPool2d-2          [-1, 10, 27, 27]               0
            Conv2d-3          [-1, 10, 25, 25]             910
         MaxPool2d-4          [-1, 10, 12, 12]               0
            Conv2d-5          [-1, 10, 10, 10]             910
         MaxPool2d-6            [-1, 10, 5, 5]               0
            Conv2d-7            [-1, 10, 3, 3]             910
         MaxPool2d-8            [-1, 10, 2, 2]               0
            Linear-9                  [-1, 30]           1,230
           Linear-10                  [-1, 16]             496
           Linear-11                  [-1, 10]             170
================================================================
Total params: 4,906
Trainable params: 4,906
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.04
Forward/backward pass size (MB): 0.35
Params size (MB): 0.02
Estimated Total Size (MB): 0.40
```