ASL Detection using Deep Learning

Gandhar Deshpande
Shruthi Pasumarthi
Mikhail Nikitin
Trevor Goad

The DataSet

- The dataset chosen is the ASL dataset from the Kaggle website. The ASL folder consists of images which are already divided into a training dataset and a test dataset.
- The images are 200x200 pixels in size and are in the RGB format.
- There are 29 classes in this dataset, 26 alphabets and 3 other classes, namely space, delete and nothing.
- The images are divided into two parts, the training dataset and the test dataset.
- The training dataset consists of 87000 images, wherein each class has 3000 images.
- The test dataset consists of 29 images, 1 image from each class.

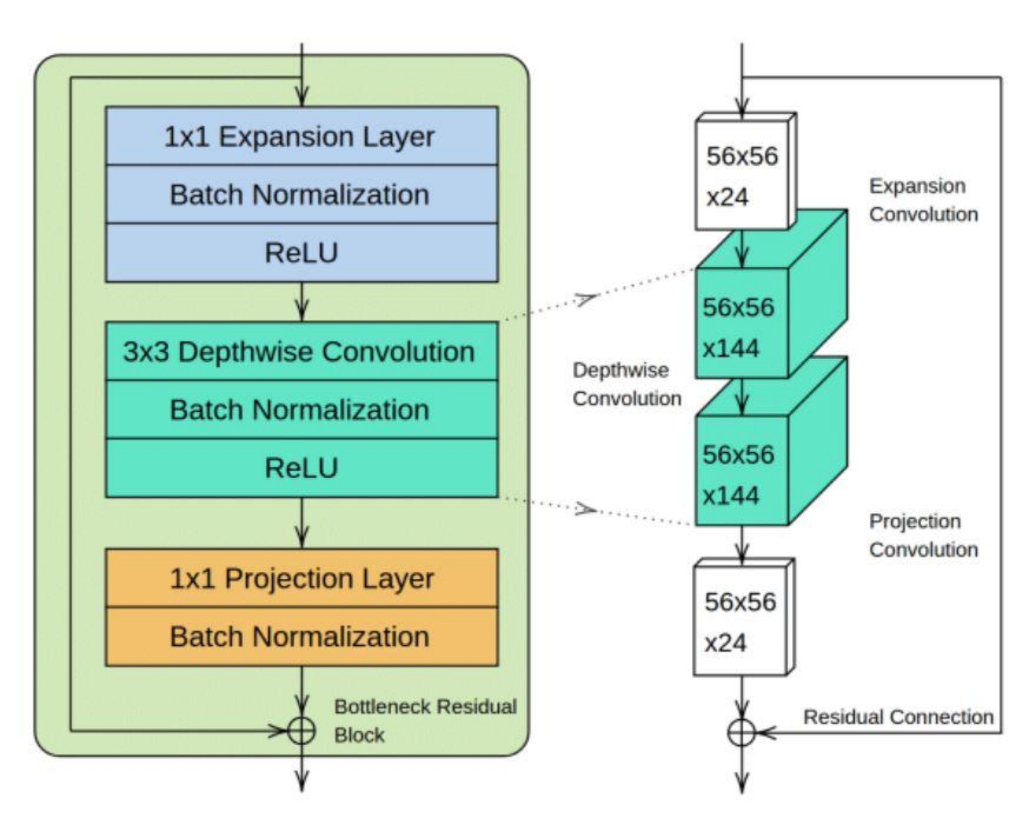
The DataSet

Models & Training

- We chose 3 models:
    - MobileNet-V2- Trained on NVIDIA Tesla GPU on Google Collab
    - Resnet-9- Trained on the CPU
    - ResNet-50- Trained on the NVIDIA GTX 1650
- The training time and details varied quite significantly across all three networks.

MobileNet-V2

- This network was developed by Google.
- This network was chosen because it is a very light model which can be used on small devices such as mobile phones.



Architecture of the MobileNet-V2

Training Details and Results

Training Time:	1 hr 4 minutes
Epochs:	2
GPU:	NVIDIA Tesla (Google Colab)
Training Accuracy:	99%
Validation Accuracy:	97%
Number of Parameters:	124M
Number of MACs:	318.99MMac

MobileNet V2 Training

```
[transforms.Resize((224,224)),
 transforms.RandomRotation(30),
 transforms.RandomInvert(0.5),
 transforms.ToTensor(),
 transforms.Normalize([0.485, 0.456, 0.406],
                      [0.229, 0.224, 0.225])])
```

Data Augmentation

```
Sequential(
  (0): Dropout(p=0.6, inplace=False)
  (1): Linear(in_features=1280, out_features=29, bias=True)
  (2): LogSoftmax(dim=1)
)
```

Dropout

Data Loader

```
trainloader = torch.utils.data.DataLoader(train_data, batch_size=512, shuffle=True)
testloader = torch.utils.data.DataLoader(test_data, batch_size=512)
```

MobileNet V2 Training

2 Epochs, split into 142 steps each

```
------------------------------ START OF EPOCH [ 1 ] >>> LR =  0.0005 ------------------------------

Epoch (1 of 2) ... Step  (  5 of 142) ... Train loss: 3.315 ... Test loss: 2.854 ... Test accuracy: 0.312
Epoch (1 of 2) ... Step  ( 10 of 142) ... Train loss: 2.717 ... Test loss: 2.022 ... Test accuracy: 0.532
Epoch (1 of 2) ... Step  ( 15 of 142) ... Train loss: 2.078 ... Test loss: 1.412 ... Test accuracy: 0.649
Epoch (1 of 2) ... Step  ( 20 of 142) ... Train loss: 1.558 ... Test loss: 1.053 ... Test accuracy: 0.750
Epoch (1 of 2) ... Step  ( 25 of 142) ... Train loss: 1.228 ... Test loss: 0.840 ... Test accuracy: 0.785
Epoch (1 of 2) ... Step  ( 30 of 142) ... Train loss: 0.980 ... Test loss: 0.693 ... Test accuracy: 0.848
Epoch (1 of 2) ... Step  ( 35 of 142) ... Train loss: 0.762 ... Test loss: 0.584 ... Test accuracy: 0.883
Epoch (1 of 2) ... Step  ( 40 of 142) ... Train loss: 0.656 ... Test loss: 0.512 ... Test accuracy: 0.897
Epoch (1 of 2) ... Step  ( 45 of 142) ... Train loss: 0.591 ... Test loss: 0.450 ... Test accuracy: 0.899
Epoch (1 of 2) ... Step  ( 50 of 142) ... Train loss: 0.487 ... Test loss: 0.399 ... Test accuracy: 0.913
Epoch (1 of 2) ... Step  ( 55 of 142) ... Train loss: 0.428 ... Test loss: 0.373 ... Test accuracy: 0.910
Epoch (1 of 2) ... Step  ( 60 of 142) ... Train loss: 0.383 ... Test loss: 0.331 ... Test accuracy: 0.914
Epoch (1 of 2) ... Step  ( 65 of 142) ... Train loss: 0.315 ... Test loss: 0.298 ... Test accuracy: 0.933
Epoch (1 of 2) ... Step  ( 70 of 142) ... Train loss: 0.305 ... Test loss: 0.272 ... Test accuracy: 0.938
Epoch (1 of 2) ... Step  ( 75 of 142) ... Train loss: 0.288 ... Test loss: 0.275 ... Test accuracy: 0.927
Epoch (1 of 2) ... Step  ( 80 of 142) ... Train loss: 0.256 ... Test loss: 0.251 ... Test accuracy: 0.926
Epoch (1 of 2) ... Step  ( 85 of 142) ... Train loss: 0.245 ... Test loss: 0.219 ... Test accuracy: 0.940
Epoch (1 of 2) ... Step  ( 90 of 142) ... Train loss: 0.224 ... Test loss: 0.195 ... Test accuracy: 0.948
Epoch (1 of 2) ... Step  ( 95 of 142) ... Train loss: 0.212 ... Test loss: 0.186 ... Test accuracy: 0.948
Epoch (1 of 2) ... Step  (100 of 142) ... Train loss: 0.204 ... Test loss: 0.176 ... Test accuracy: 0.952
Epoch (1 of 2) ... Step  (105 of 142) ... Train loss: 0.187 ... Test loss: 0.205 ... Test accuracy: 0.939
Epoch (1 of 2) ... Step  (110 of 142) ... Train loss: 0.182 ... Test loss: 0.186 ... Test accuracy: 0.948
Epoch (1 of 2) ... Step  (115 of 142) ... Train loss: 0.165 ... Test loss: 0.177 ... Test accuracy: 0.948
Epoch (1 of 2) ... Step  (120 of 142) ... Train loss: 0.172 ... Test loss: 0.164 ... Test accuracy: 0.952
Epoch (1 of 2) ... Step  (125 of 142) ... Train loss: 0.156 ... Test loss: 0.152 ... Test accuracy: 0.955
Epoch (1 of 2) ... Step  (130 of 142) ... Train loss: 0.150 ... Test loss: 0.153 ... Test accuracy: 0.953
Epoch (1 of 2) ... Step  (135 of 142) ... Train loss: 0.136 ... Test loss: 0.146 ... Test accuracy: 0.957
Epoch (1 of 2) ... Step  (140 of 142) ... Train loss: 0.133 ... Test loss: 0.139 ... Test accuracy: 0.957
Epoch (1 of 2) ... Step  (142 of 142) ... Train loss: 0.054 ... Test loss: 0.130 ... Test accuracy: 0.962
```

```
------------------------------ START OF EPOCH [ 2 ] >>> LR =  5e-05 ------------------------------
Epoch (2 of 2) ... Step  (  5 of 142) ... Train loss: 0.148 ... Test loss: 0.123 ... Test accuracy: 0.965
Epoch (2 of 2) ... Step  ( 10 of 142) ... Train loss: 0.114 ... Test loss: 0.121 ... Test accuracy: 0.966
Epoch (2 of 2) ... Step  ( 15 of 142) ... Train loss: 0.109 ... Test loss: 0.119 ... Test accuracy: 0.966
Epoch (2 of 2) ... Step  ( 20 of 142) ... Train loss: 0.116 ... Test loss: 0.118 ... Test accuracy: 0.967
Epoch (2 of 2) ... Step  ( 25 of 142) ... Train loss: 0.099 ... Test loss: 0.116 ... Test accuracy: 0.968
Epoch (2 of 2) ... Step  ( 30 of 142) ... Train loss: 0.109 ... Test loss: 0.116 ... Test accuracy: 0.968
Epoch (2 of 2) ... Step  ( 35 of 142) ... Train loss: 0.104 ... Test loss: 0.117 ... Test accuracy: 0.967
Epoch (2 of 2) ... Step  ( 40 of 142) ... Train loss: 0.096 ... Test loss: 0.116 ... Test accuracy: 0.968
Epoch (2 of 2) ... Step  ( 45 of 142) ... Train loss: 0.104 ... Test loss: 0.115 ... Test accuracy: 0.968
Epoch (2 of 2) ... Step  ( 50 of 142) ... Train loss: 0.110 ... Test loss: 0.113 ... Test accuracy: 0.970
Epoch (2 of 2) ... Step  ( 55 of 142) ... Train loss: 0.103 ... Test loss: 0.112 ... Test accuracy: 0.970
Epoch (2 of 2) ... Step  ( 60 of 142) ... Train loss: 0.100 ... Test loss: 0.111 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  ( 65 of 142) ... Train loss: 0.098 ... Test loss: 0.110 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  ( 70 of 142) ... Train loss: 0.093 ... Test loss: 0.112 ... Test accuracy: 0.970
Epoch (2 of 2) ... Step  ( 75 of 142) ... Train loss: 0.086 ... Test loss: 0.111 ... Test accuracy: 0.970
Epoch (2 of 2) ... Step  ( 80 of 142) ... Train loss: 0.093 ... Test loss: 0.109 ... Test accuracy: 0.972
Epoch (2 of 2) ... Step  ( 85 of 142) ... Train loss: 0.097 ... Test loss: 0.106 ... Test accuracy: 0.973
Epoch (2 of 2) ... Step  ( 90 of 142) ... Train loss: 0.083 ... Test loss: 0.104 ... Test accuracy: 0.973
Epoch (2 of 2) ... Step  ( 95 of 142) ... Train loss: 0.098 ... Test loss: 0.105 ... Test accuracy: 0.973
Epoch (2 of 2) ... Step  (100 of 142) ... Train loss: 0.093 ... Test loss: 0.105 ... Test accuracy: 0.973
Epoch (2 of 2) ... Step  (105 of 142) ... Train loss: 0.089 ... Test loss: 0.107 ... Test accuracy: 0.972
Epoch (2 of 2) ... Step  (110 of 142) ... Train loss: 0.083 ... Test loss: 0.107 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (115 of 142) ... Train loss: 0.102 ... Test loss: 0.106 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (120 of 142) ... Train loss: 0.087 ... Test loss: 0.107 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (125 of 142) ... Train loss: 0.091 ... Test loss: 0.108 ... Test accuracy: 0.970
Epoch (2 of 2) ... Step  (130 of 142) ... Train loss: 0.098 ... Test loss: 0.108 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (135 of 142) ... Train loss: 0.093 ... Test loss: 0.107 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (140 of 142) ... Train loss: 0.085 ... Test loss: 0.106 ... Test accuracy: 0.971
Epoch (2 of 2) ... Step  (142 of 142) ... Train loss: 0.033 ... Test loss: 0.106 ... Test accuracy: 0.972
```

ResNet

- Earlier, the deeper networks created would have a much poorer performance than the shallow networks.
- This happened due to the vanishing gradient problem, wherein the gradient would reduce so much that the updates from the training has no effect. This in turn made the training ineffective.
- ResNet models have skip connections due to which the performance of the deeper connections boosted.
- The resnet models also solved the vanishing gradient problem.

ResNet-50

- The Resnet-50 has 50 resblocks.
- This is a combination of several convolutional, max pooling and average pooling.
- The ResNet networks are big networks, but their performance is really good.
- The ResNet-50 is a slightly different compared to the other resnets, the skip connections in this network skip 3 layers and there are a few 1x1 convolutional layers which help control the size of the feature maps across the channels.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

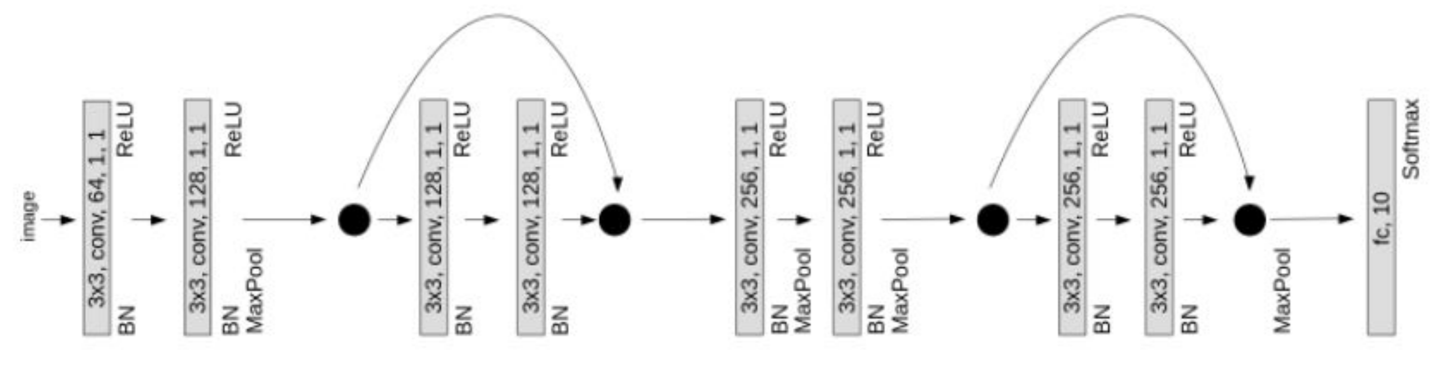
Architecture of the ResNet-50

Training Details and Results

Training Time:	2 hr 37 minutes
Epochs:	50
GPU:	NVIDIA GTX 1650
Training Accuracy:	87%
Validation Accuracy:	88%
Number of Parameters:	25.56M
Number of MACs:	0.34 GMac

ResNet-9

- This is a smaller network in the ResNet family of networks.
- This model also like the other networks in the family gives a ver high accuracy.
- The ResNet-9 model also has skip connections that skip over 2 layers.
- There are a mix of max pooling layers and batch convolutional layers.
- The optimizer used for training this dataset is the Adam Optimizer.



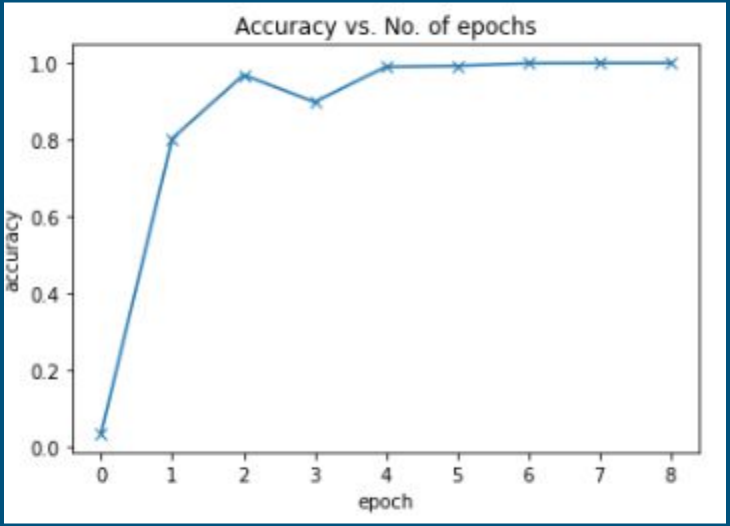
Architecture of ResNet-9

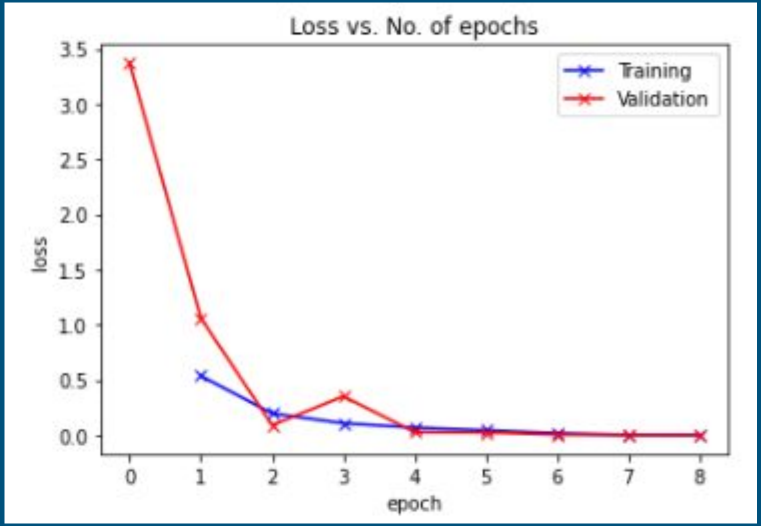
Training Details and Results

Training Time:	8 hr 35 minutes
Epochs:	7
GPU:	Nil (CPU)
Training Accuracy:	100%
Validation Accuracy:	100%
Number of Parameters:	6.63M
Number of MACs:	1.52 GMac

```
100%|██████████| 1032/1032 [57:31<00:00,  3.34s/it]
100%|██████████| 55/55 [02:13<00:00,  2.43s/it]

Epoch [0], last_lr: 0.00395, train_loss: 0.5363, val_loss: 1.0588, val_acc: 0.8034

100%|██████████| 1032/1032 [54:51<00:00,  3.19s/it]
100%|██████████| 55/55 [02:03<00:00,  2.25s/it]

Epoch [1], last_lr: 0.00936, train_loss: 0.1959, val_loss: 0.0897, val_acc: 0.9688

100%|██████████| 1032/1032 [55:22<00:00,  3.22s/it]
100%|██████████| 55/55 [02:05<00:00,  2.27s/it]

Epoch [2], last_lr: 0.00972, train_loss: 0.1072, val_loss: 0.3507, val_acc: 0.8984

100%|██████████| 1032/1032 [54:33<00:00,  3.17s/it]
100%|██████████| 55/55 [02:02<00:00,  2.24s/it]

Epoch [3], last_lr: 0.00812, train_loss: 0.0673, val_loss: 0.0273, val_acc: 0.9900

100%|██████████| 1032/1032 [57:51<00:00,  3.36s/it]
100%|██████████| 55/55 [02:01<00:00,  2.20s/it]

Epoch [4], last_lr: 0.00556, train_loss: 0.0432, val_loss: 0.0245, val_acc: 0.9926

100%|██████████| 1032/1032 [1:03:38<00:00,  3.70s/it]
100%|██████████| 55/55 [02:08<00:00,  2.34s/it]

Epoch [5], last_lr: 0.00283, train_loss: 0.0167, val_loss: 0.0038, val_acc: 0.9994

100%|██████████| 1032/1032 [1:13:30<00:00,  4.27s/it]
100%|██████████| 55/55 [02:27<00:00,  2.68s/it]

Epoch [6], last_lr: 0.00077, train_loss: 0.0022, val_loss: 0.0006, val_acc: 0.9999

100%|██████████| 1032/1032 [1:20:23<00:00,  4.67s/it]
100%|██████████| 55/55 [02:20<00:00,  2.55s/it]

Epoch [7], last_lr: 0.00000, train_loss: 0.0003, val_loss: 0.0003, val_acc: 1.0000
Wall time: 8h 35min 5s
```

Training of the ResNet-9



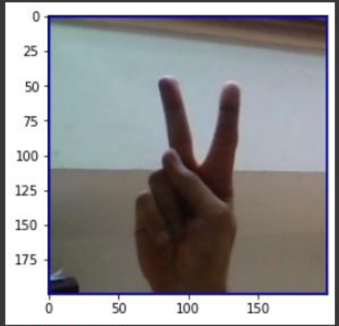


Performance Graphs

Comparison

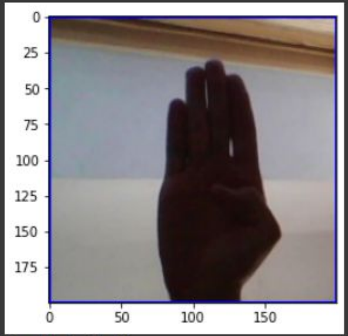
- Comparing the three models, The MobileNet-v2 was what we used for further research but ResNet-9 provided the best results
- Even though the accuracy may have been lower compared to the ResNet models, it has an edge over the ResNet models by being a less demanding network while providing competitive accuracy
- We felt that the ResNet -9 model was a bit fishy due to the 100% accuracy in the validation dataset.

MobileNetV2 Testing



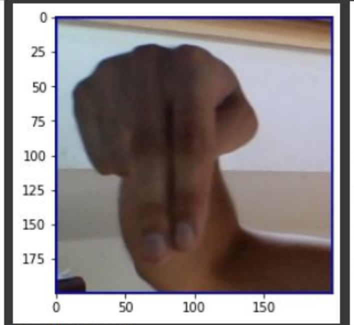
Predicted: K

Actual:
K_test.jpg



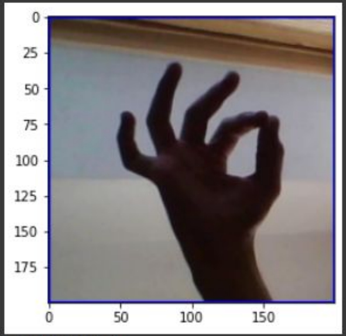
Predicted: B

Actual:
B_test.jpg



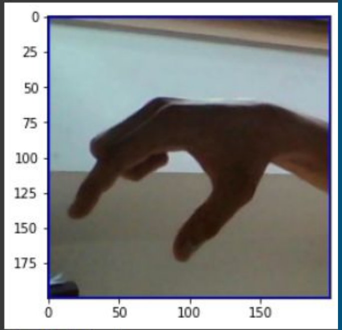
Predicted: N

Actual:
N_test.jpg



Predicted: F

Actual:
F_test.jpg

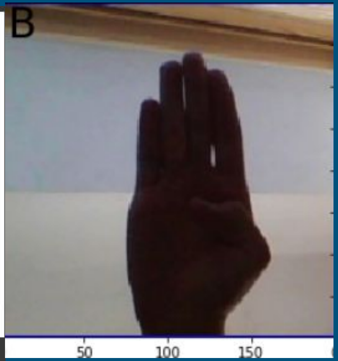


Predicted: Q

Actual:
Q_test.jpg

MobileNet-V2 Issues

- First version of this network could only see dark hands on light background, this was solved with an invert data augmentation
- When we tried to do webcam integration, the model would still not be very accurate. This is most likely due to the background of the dataset being static while a real background is much more dynamic



Future Works

This project can be expanded into a real-time application, wherein this model is deployed onto microcontrollers. These microcontrollers can be placed like a camera and when a person signs, this can be detected and the listener can easily translate the letters.

Another direction this project can be expanded into is by adding different Sign Languages, such as the British Sign Language, the Italian Sign Language, etc.

What We Learned

1. Different Networks can be more or less suited for an application
2. You need to be aware of the limitations of your dataset that you are using

   (For example, our dataset was limited in the background and presentation of the ASL signing)