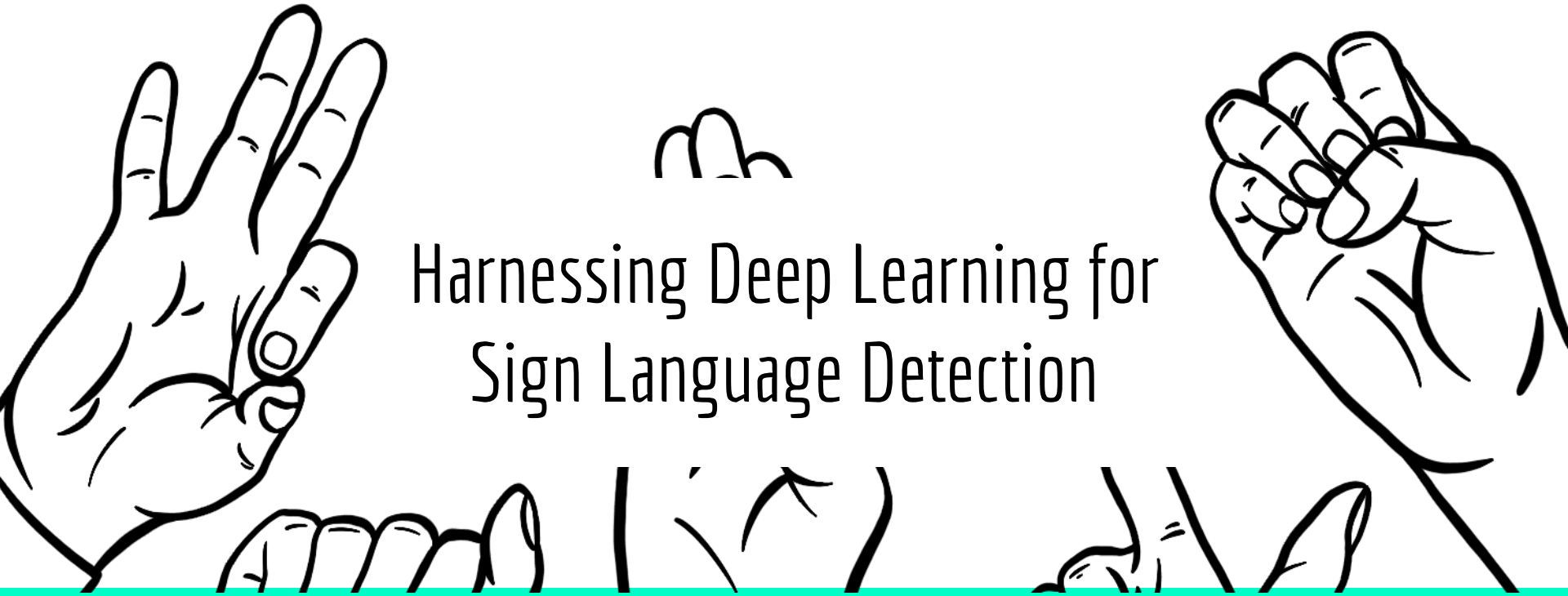


Harnessing Deep Learning for Sign Language Detection

Team: Sign Network

Joon Jung, Srinidhi Manikantan, Heathvonn Styles, Runru Shen

Background

- Over half a million people speak ASL as their First language
- Different situations bring different challenges, on top of biological challenge
    - Born with the disability, Accident, child with disability...
- NN might be able to provide easier way to understand and interpret ASL, enhancing different challenges and situations that people face

Our Project Value is to assess how well NN Models are able to predict ASL signs on diverse datasets, accounting for difference in skin tone, lighting, color grading etc.

More info about the dataset

In this project, we introduced two distinct datasets utilized for training and testing purposes.

Train Dataset

- Comprises **29 features**, including **26 alphabets and 3 special categories**
- We started out with 3000 images for each feature sized at **200x200 pixels**
- Through meticulous data processing, we've **magnified our dataset sevenfold, resulting in 21,000 images per alphabet**
- This **totals up to 600,000 images** for train dataset

Test Dataset

- Comprises **26 features**, including **26 alphabets**
- We have 70 images for each feature sized at **200x200 pixels**
- This **totals up to 1800 images** for test dataset

Data Processing

Pre-processing techniques were applied to the training dataset to augment various hand signs, thereby enhancing the size and variety of the image dataset utilized for training the neural network





Pre-Processing Methods

Thresholding, edge tracing, as well as adjustments to image brightness and sharpness were done to produce 5 variations for every image in the train dataset.

Goal

The primary objective behind augmenting the hand signs was to mitigate any inherent biases within the dataset and improve the overall predictive accuracy of the models

MODELS

Explored a mix of models:

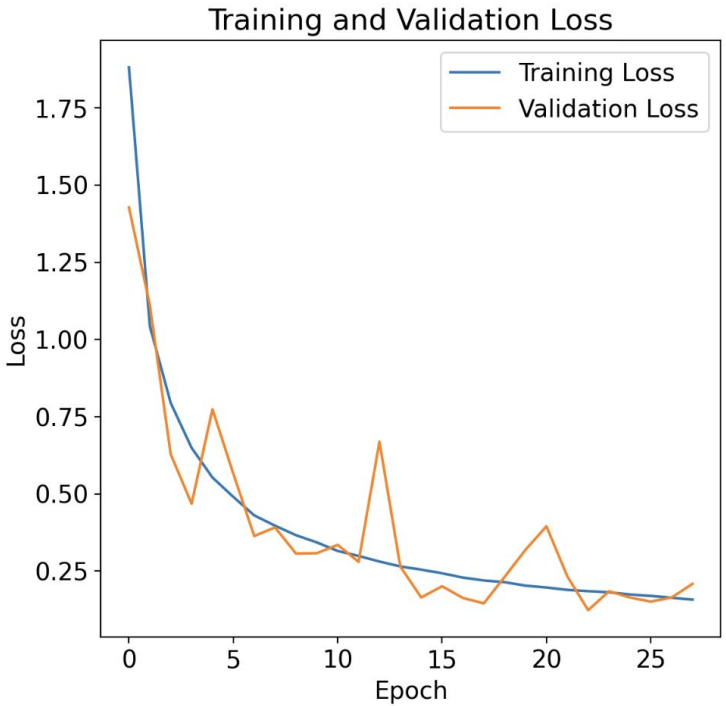
Models made from scratch
2 Layer CNN Model

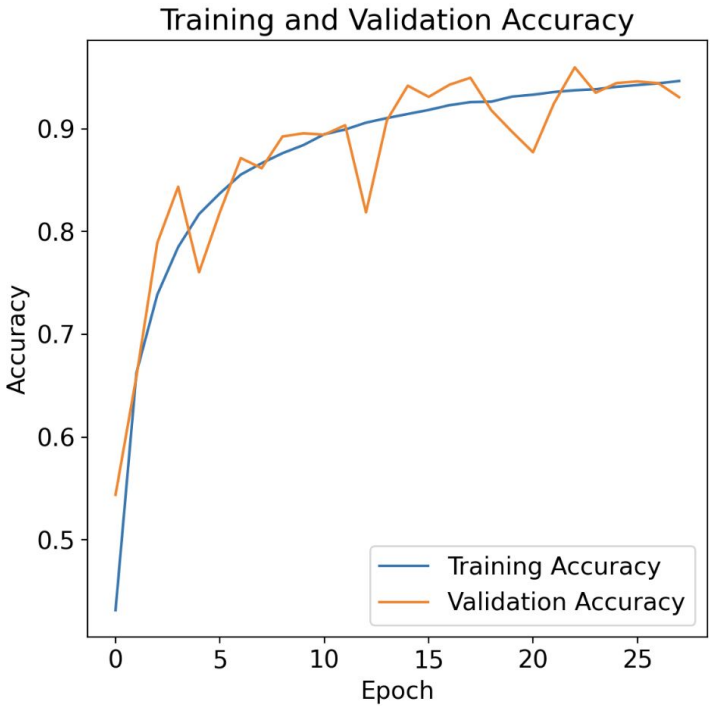
Pretrained Models
ResNet50 Model

```
class ConvolutionalNetwork(LightningModule):

    Notebook editor cells ):
        super(ConvolutionalNetwork, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 3, 1)
        self.conv2 = nn.Conv2d(6, 16, 3, 1)
        self.fc1 = nn.Linear(16 * 54 * 54, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 20)
        self.fc4 = nn.Linear(20, len(class_names))

    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2, 2)
        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2, 2)
        X = X.view(-1, 16 * 54 * 54)
        X = F.relu(self.fc1(X))
        X = F.relu(self.fc2(X))
        X = F.relu(self.fc3(X))
        X = self.fc4(X)
        return F.log_softmax(X, dim=1)

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=0.001)
        return optimizer

    def training_step(self, train_batch, batch_idx):
        X, y = train_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("train_loss", loss)
        self.log("train_acc", acc)
        return loss
```

Results

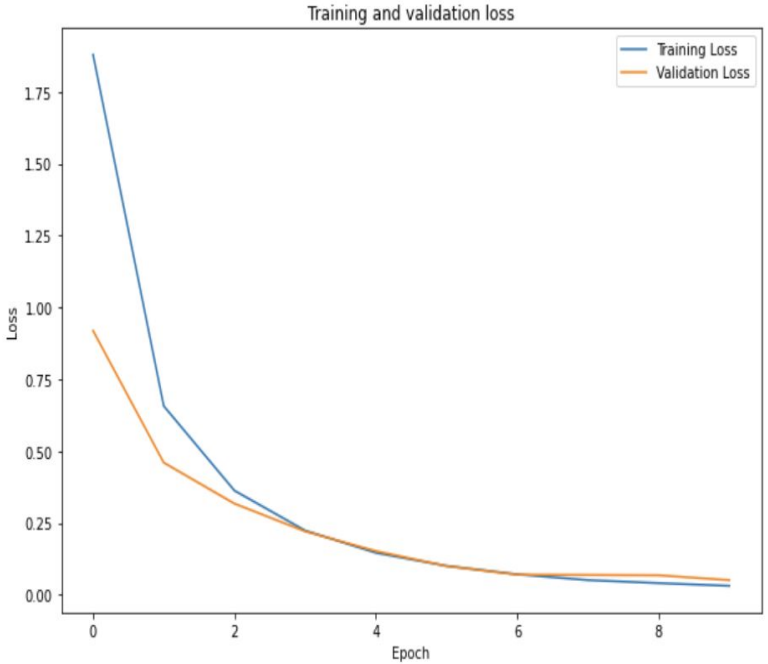
Using Unprocessed Images - Sequential Model made from scratch

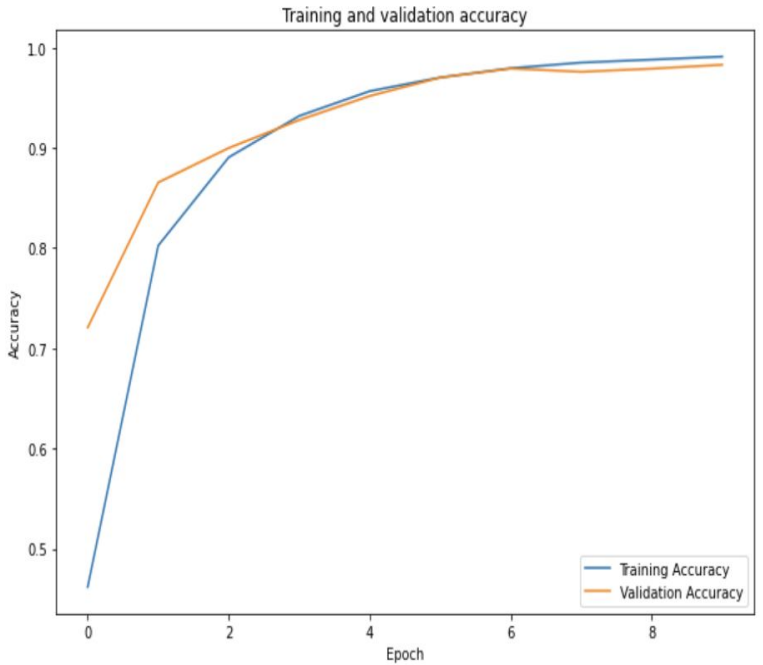




Results

Using Unprocessed Images - Pretrained Resnet 50 Model





Results

Using Processed Images - 1 Epoch took 2 hours and 43 minutes

Test metric	DataLoader 0
test_acc	0.4907635450363159
test_loss	1.5825673341751099

	precision	recall	f1-score
A	0.97	1.00	0.99
B	0.99	0.96	0.98
C	0.99	1.00	0.99
D	1.00	0.99	0.99
E	0.93	0.99	0.96
F	1.00	1.00	1.00
G	0.95	1.00	0.97
H	1.00	0.96	0.98
I	1.00	0.98	0.99
J	1.00	1.00	1.00
K	0.96	1.00	0.98
L	1.00	1.00	1.00
M	1.00	0.97	0.98
N	0.99	0.97	0.98
O	0.99	0.97	0.98
P	1.00	0.98	0.99
Q	0.99	1.00	1.00
R	0.99	0.95	0.97
S	0.90	1.00	0.95
T	0.99	0.99	0.99
U	0.97	0.96	0.97
V	0.94	0.98	0.96
W	0.99	0.95	0.97
X	0.99	0.94	0.96
Y	0.99	1.00	1.00
Z	1.00	1.00	1.00
del	1.00	1.00	1.00
nothing	1.00	1.00	1.00
space	1.00	1.00	1.00
accuracy			0.98

	precision	recall	f1-score
A	0.9866	0.9933	0.9899
B	0.9899	0.9879	0.9889
C	0.9815	0.9938	0.9876
D	0.9958	0.9772	0.9864
E	0.9921	0.9882	0.9902
F	0.9815	0.9958	0.9886
G	0.9900	0.9960	0.9930
H	0.9788	0.9935	0.9861
I	0.9598	0.9938	0.9765
J	0.9820	0.9761	0.9791
K	0.9811	0.9628	0.9718
L	0.9918	1.0000	0.9959
M	0.9934	0.9678	0.9804
N	1.0000	0.9978	0.9989
O	0.9896	0.9896	0.9896
P	0.9890	0.9978	0.9934
Q	0.9960	0.9920	0.9940
R	0.9936	0.9789	0.9862
S	0.9881	0.9921	0.9901
T	0.9978	0.9850	0.9914
U	0.9755	0.9696	0.9725
V	0.9360	0.9679	0.9517
W	0.9956	0.9305	0.9619
X	0.9669	0.9979	0.9822
Y	0.9390	0.9830	0.9605
Z	0.9902	0.9980	0.9941
del	0.9850	0.9978	0.9914
nothing	0.9978	0.9957	0.9968
space	1.0000	0.9433	0.9708
accuracy			0.9841

	precision	recall	f1-score
0	1.00	1.00	1.00
1	1.00	0.97	0.98
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	0.99	0.99	0.99
5	1.00	1.00	1.00
6	1.00	0.97	0.99
7	1.00	0.99	0.99
8	0.99	1.00	0.99
9	0.99	1.00	1.00
10	0.99	0.96	0.98
11	0.99	1.00	1.00
12	1.00	0.99	0.99
13	0.99	1.00	0.99
14	1.00	0.99	1.00
15	0.98	1.00	0.99
16	1.00	1.00	1.00
17	0.97	0.98	0.98
18	0.99	0.99	0.99
19	1.00	0.99	0.99
20	0.98	0.98	0.98
21	0.95	0.99	0.97
22	0.99	1.00	1.00
23	1.00	1.00	1.00
24	1.00	1.00	1.00
25	1.00	1.00	1.00
26	1.00	1.00	1.00
27	1.00	1.00	1.00
28	1.00	1.00	1.00
accuracy			0.99

Future directions

- Video processing
- Pose Detection
- Full ASL