CNN for ASL Sign Language Classification

Aidan Aug, Alan Zhang, Shreya Wadhwa, Trisha Karani

Applications: Image Data

Problem Definition

Problem Definition

What problem were you trying to solve or understand?

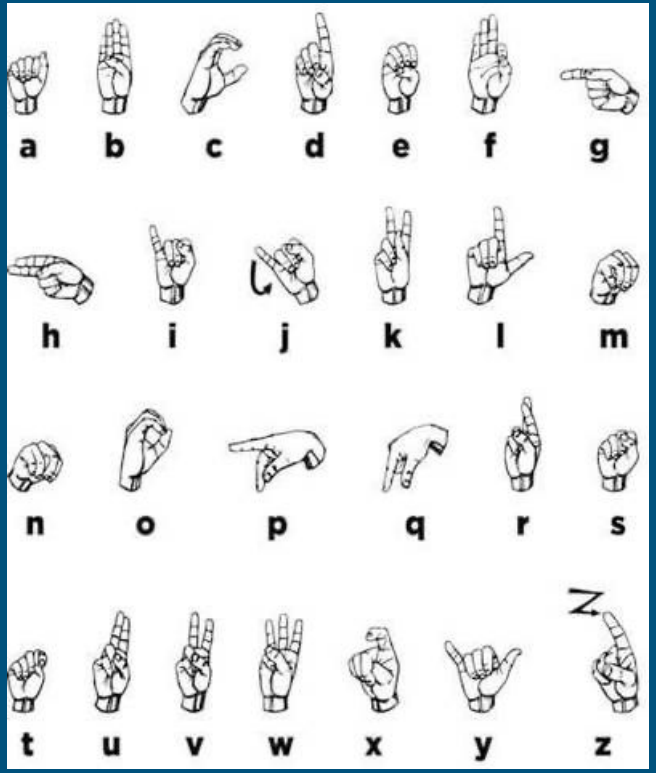
We aim to build a model that can return the alphabet corresponding to the hand gesture as seen in any image

What kind of data did you work with?

The input to our algorithm will be images of hand configurations of ASL letters.

How does this fit into the concepts of the course?

We extracted the labels for each image from the image name, so this is a supervised classification problem.



Why is this an interesting problem?

What are the real-world implications of this data and task? What ethical implications does this problem have?

The ability to communicate using ASL can increase awareness about the hard of hearing community. We want to explore the area of sign language translation: how people who might not be immediately fluent with ASL can communicate with people who rely on sign language. There are no ethical implications to this problem, but in the future we would like to add hand images with different complexities to remove bias on the bases of race.

How is this problem similar to others we've seen in lecture/breakout/homework?

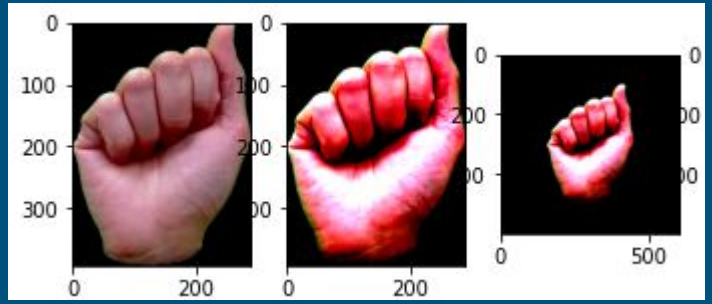
This is a supervised classification problem, which is similar to HW3 where we were given fruit images with labels and we were supposed to build different models for the same.

What makes this problem unique?

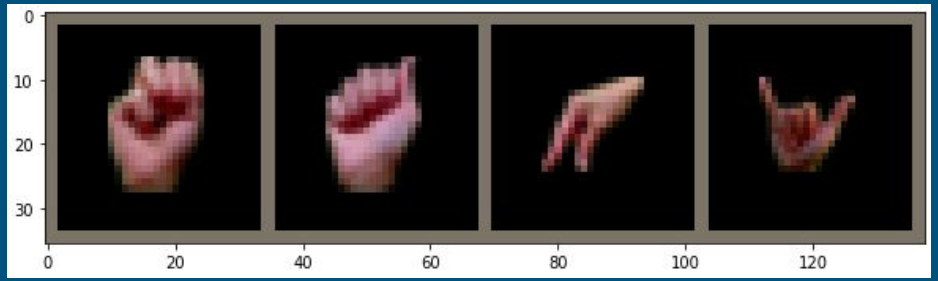
There has been a lot of discussion about being more inclusive towards the hard of hearing community by encouraging more people to learn ASL, but few about how we can make use to ML to facilitate the ability to do the same without having to learn the ASL in its entirety.

Dataset and Preprocessing

Dataset



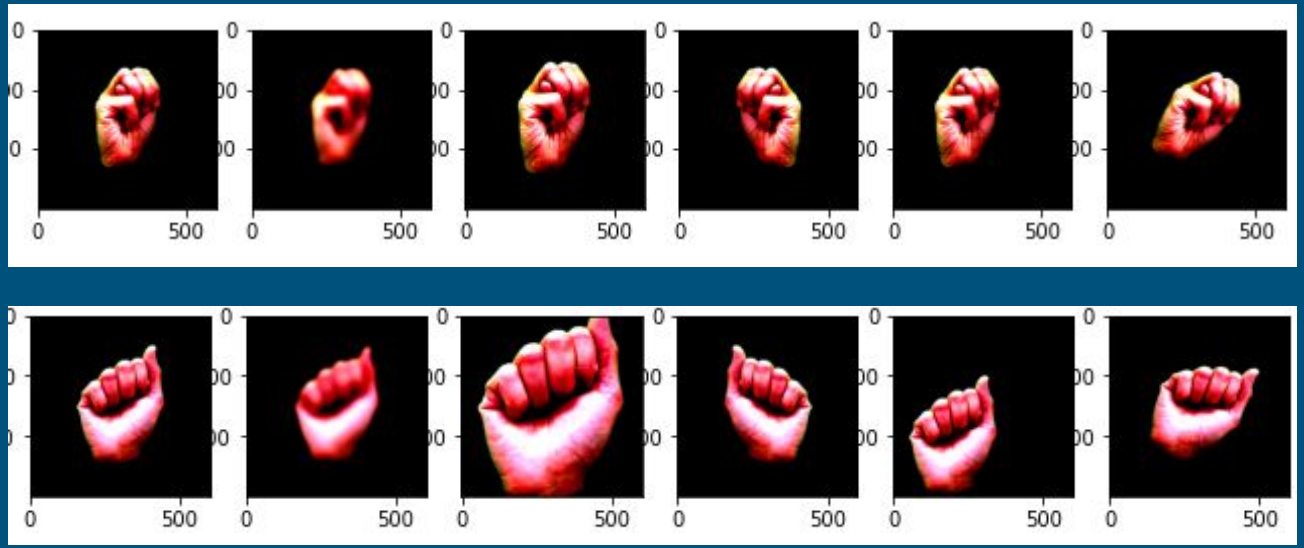
- The data set is the MU Hand Images ASL dataset
  - We will be using a 900 image subset
- The images range from 200-500 pixels per side
- Preprocessing
  - Extract labels from file names
  - Normalize
  - Resize them all to be 600 x 600, by padding with black pixels
  - Compress them to be 32x32 before feeding them into our CNN



Dataset - Augmentation

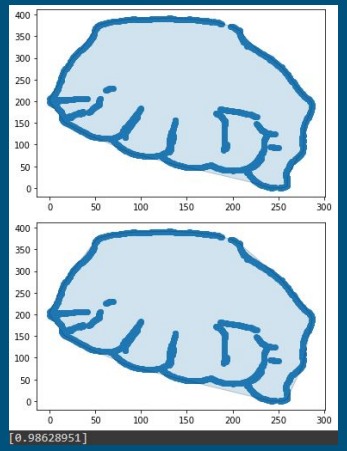
We do a variety of augmentations to our images, including

- Blur
- Scaling
- Flipping
- Translation
- Rotation

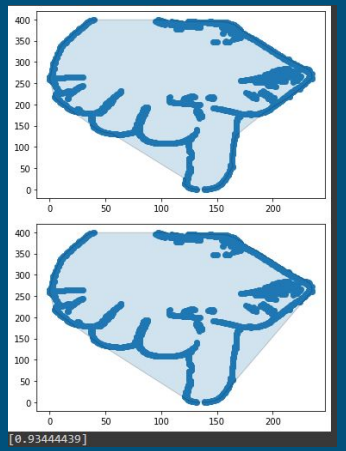


Dataset - Features

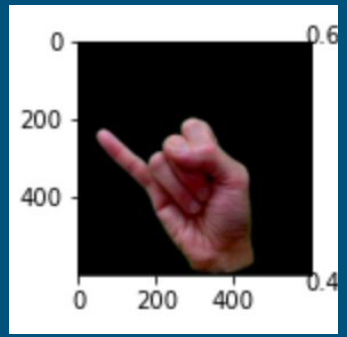
- Ratio of hand-to-background
  - Simply count number of black pixels vs non-black pixels
- Convexity of the hand
  - Ratio of the area of the convex hull and the alphashape
  - First, use canny edge detector, the calculate the areas
  - Convex hull is smallest convex set that contains all the points
  - alphashape allows for non-convexities



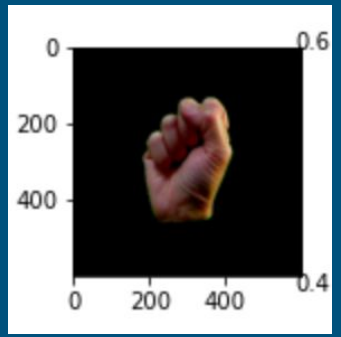
98.62%



93.44%



21.4752%



13.57%

- Both of these are related the to the number of fingers present in the image

Methods

Methods - Baseline

"Deep Convolutional Networks for Gesture Recognition in American Sign Language"

(Vivek Bheda, N. Dianna Radpour 2017)

1. Similar Dataset
2. Data preprocessing and augmentation
3. CNN
    a. 6 Convolutional layers
    b. 3 MaxPools
    c. 5 Dropout
    d. 2 Fully-connected Layers
4. **Highest Accuracy: 82.5%**
    a. 3250 total images
    b. Train-dev test split

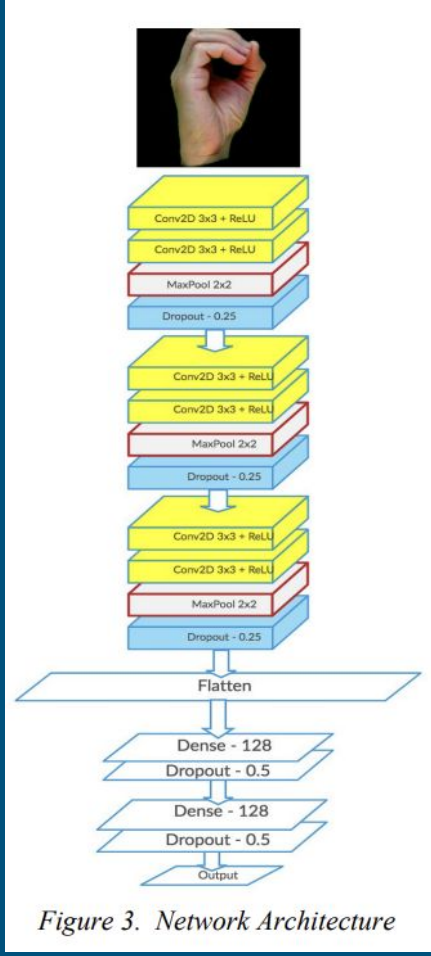


Figure 3. Network Architecture

Use this model structure and accuracy as baselines for methods and results

https://arxiv.org/pdf/1710.06836.pdf

Methods - Model Specifications

Hypothesis Class:

- *Neural Networks* (universal approximators)
- Convolutional Neural Network (CNN) (for images)

CNN Structure: *LeNet (1998) Basis*

Loss Function: *Cross Entropy Loss*

Optimization Approach: Stochastic Gradient Descent

Learning Rate: Hyperparameter (0.001)

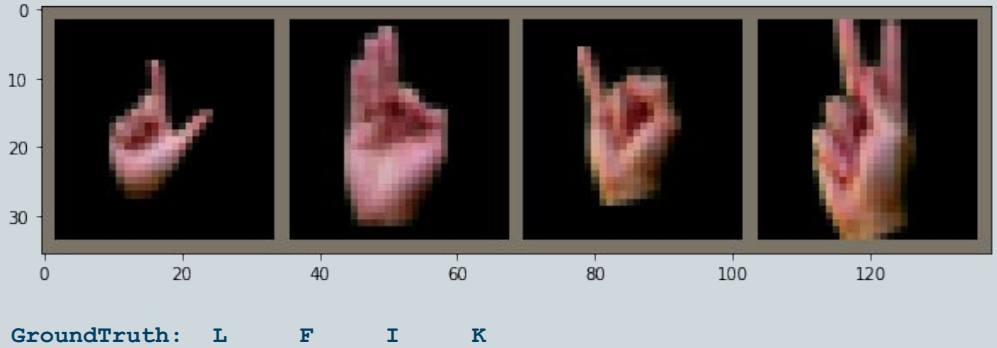
```
class LeNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 26)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

lr = 0.001  # Hyperparameter
momentum = 0.9 # Standard should be close to 1
epochs = 25 # Number iterations through dataset
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(lenet_with_aug.parameters(),
lr=lr, momentum=momentum)
```

Methods - Process

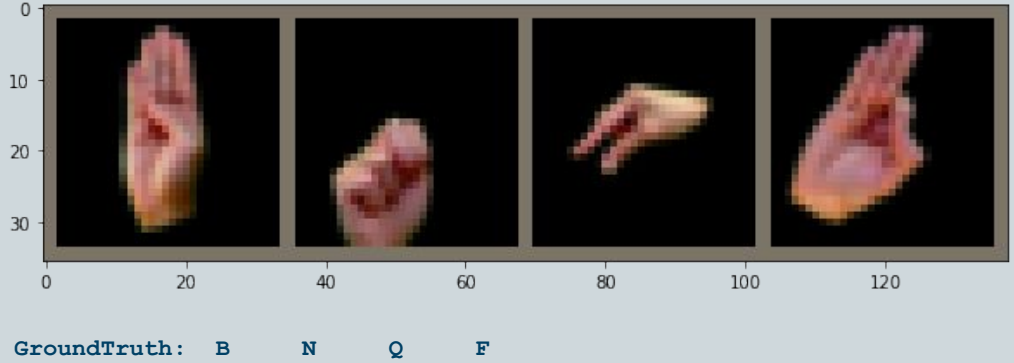
Process:

1. Preprocess data
   a. Normalized; 32 x 32 pixels
2. Create LeNet model (3 channels)
3. Train one model on original dataset
   a. Images as seen in dataset
4. Train second model on Data Augmented Dataset
   a. Each image has single augmentation
5. Test/compare models…

Original Dataset



Data Augmented Dataset



Results

Results

Predicted on two datasets:

1. Original images
2. Multiple augmentations images (each image has 1-3 augmentations applied)

	Model 1: LeNet trained on Original Dataset	Model 2: LeNet on trained on Augmented Dataset
Validation accuracy on original dataset	100%	87%
Prediction accuracy on multiple augmentation dataset	17%	57%

Deliverables

Must Accomplish	
Model with classification accuracy of 50%	✅
Identify 5 important data augmentation methods in sign-language classification.	✅
Identify potential themes/ image features common among misclassifications in our model.	💪🏽
Expect to accomplish	
Model with classification accuracy of 80%	✅
Identity transformations on our data that would not retain accuracy	✅
Create and compare at least two models with differing feature extraction techniques or network architectures	💪🏽

Deliverables

Would like to accomplish	
Create a majority vote classifier with multiple model types and parameters.	❌
Augment our model by including a broader dataset with sign-language words for model training instead of just letters and digits.	❌
Formal write-up	💪🏽

Future Steps

Will complete:

1. Identify themes in misclassified images
2. Identify efficacy of each augmentation
3. Train using feature engineering (convexity, hand-to-back)

Would like to complete:

4. Predict on real-world ASL images
5. Potentially implement other model types (ResNet) for further comparison

What we've learned

We learned more about CNNs, the underlying model that we use. In addition, we used many data augmentation methods, and will be adding features to our CNNs in addition to just the image.

We would've liked to look at more neural nets to test performance, since surprisingly, LeNet was better (and faster) than AlexNet while being a simpler structure, so we wonder whether there are other simple and good models out there

We still want to discover how adding the features we mentioned before, convexity and ratio of hand-to-background will affect our model

What we've learned

We should've conducted a more in depth literature search to see how we can build off of previous works.

With that being said, we would like feedback on whether our data augmentations seem reasonable. Are there any other data augmentations that we should consider? Could we make changes to our network architecture or add features that could be useful?

azhang42@jh.edu

References

If you mentioned any papers or software packages, please list them here

- Shin, Jungpil, et al. "American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation." Sensors 21.17 (2021): 5856.
- Barczak, A. L. C., Reyes, N. H., Abastillas, M., Piccio, A., & Susnjak, T. (2011). A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures. Research Letters in the Information and Mathematical Sciences, 15, 12–20.
- Software Packages:
  - Sklearn
  - Pytorch