Sign Language Images Classification

Nhung Nguyen & Tina Nguyen

12/09/2019

Machine Learning II, Fall 2019

Agenda

- Project motivation and goal
- Dataset
- Approach
- Result
- Conclusion

Motivation

- Growing technology to help people with disability
- Application:
    - UNI:”detecting hand and finger gestures with its specialized camera algorithm, then converting it to the text in very short time to provide meaning of a given sign language” by reference

Goal

- Create an effective neural network to recognize the alphabets in sign language through images

Dataset

- Sign Language Dataset from Kaggle
- 28x28 grayscale images
- No J or Z
- Train-test ratio ~ 4:1

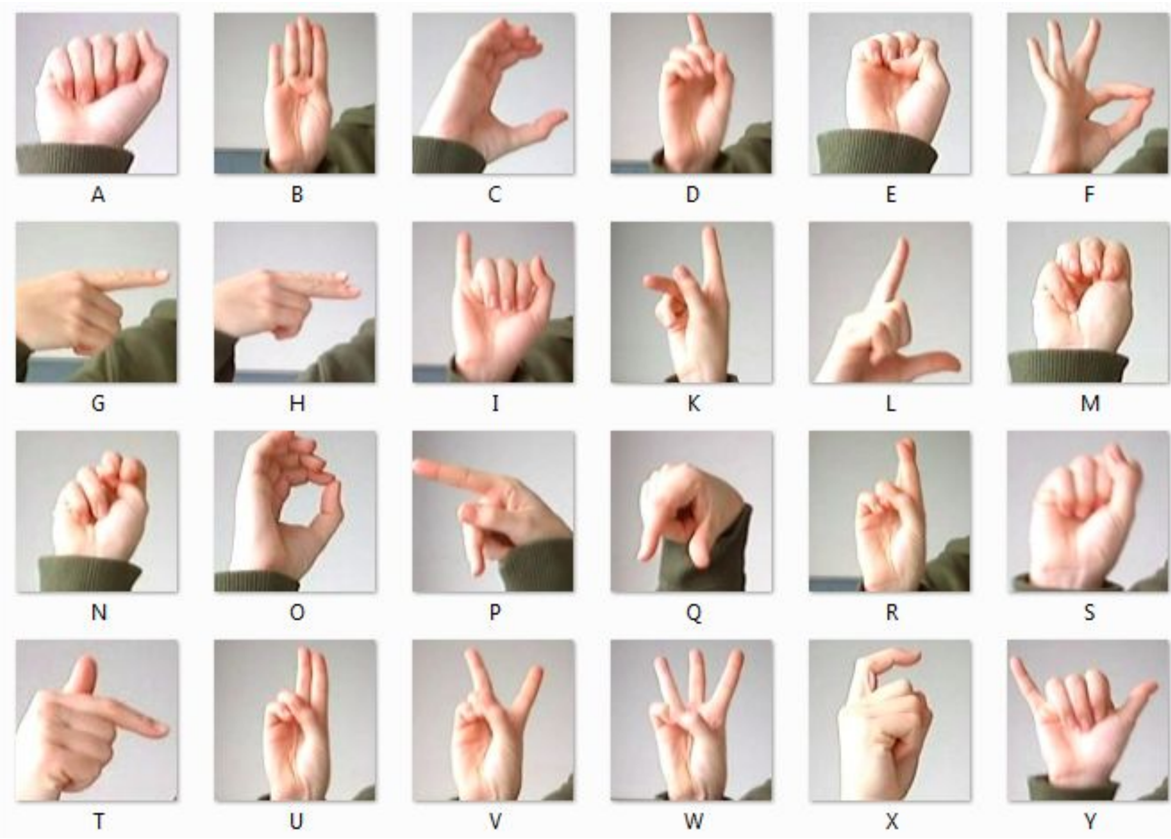


Image source

Data Preprocessing

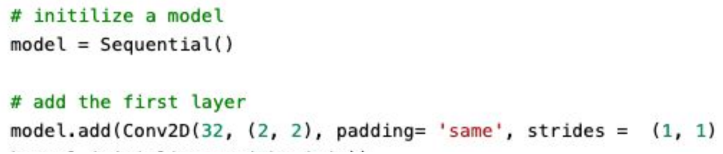
- Maintain original images size since the data was properly preprocessed
- Reshape data to feed into model
- Split train dataset into train and validation with ratio 7:3

Frameworks

- Keras

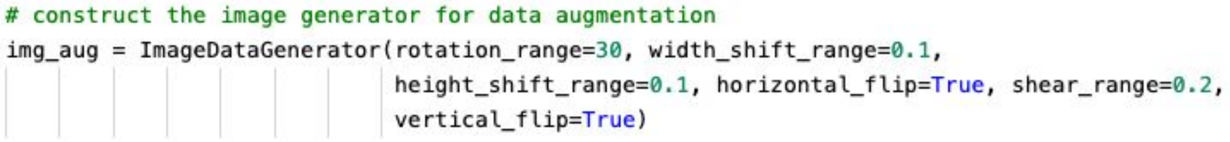
Keras Framework

- Use The Sequential API Keras with 4 layers



```
# initilize a model
model = Sequential()

# add the first layer
model.add(Conv2D(32, (2, 2), padding= 'same', strides =  (1, 1)
```

- Use ImageDataGenerator to transform data

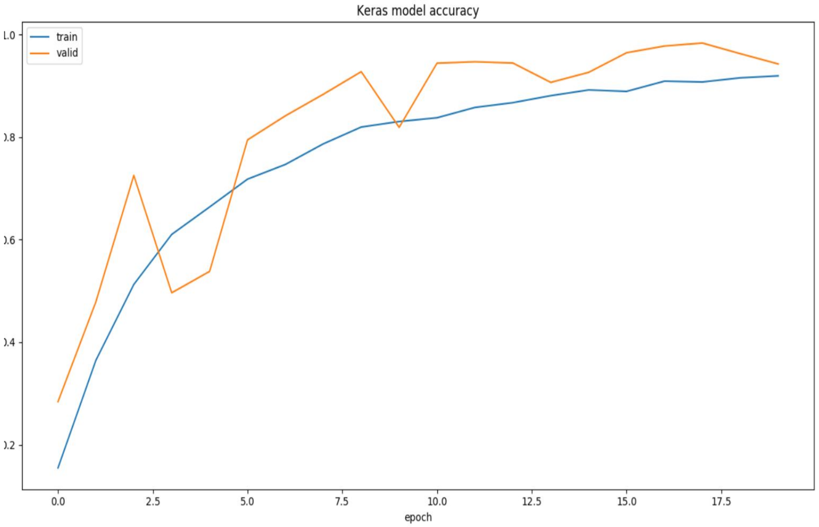


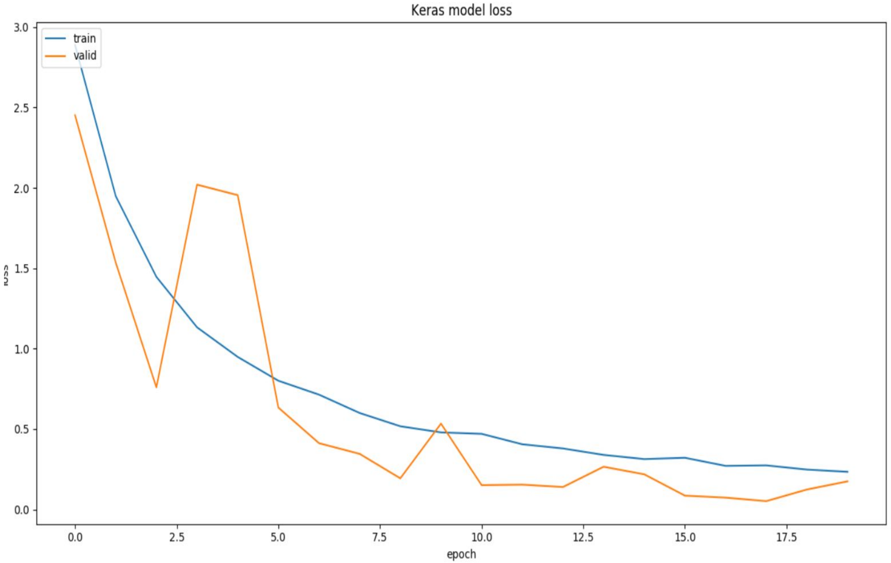
```
# construct the image generator for data augmentation
img_aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
                             height_shift_range=0.1, horizontal_flip=True, shear_range=0.2,
                             vertical_flip=True)
```

- Optimizer: Adam
- Loss: Categorical_CrossEntropy

Result - Keras

- Accuracy on test set: 92%



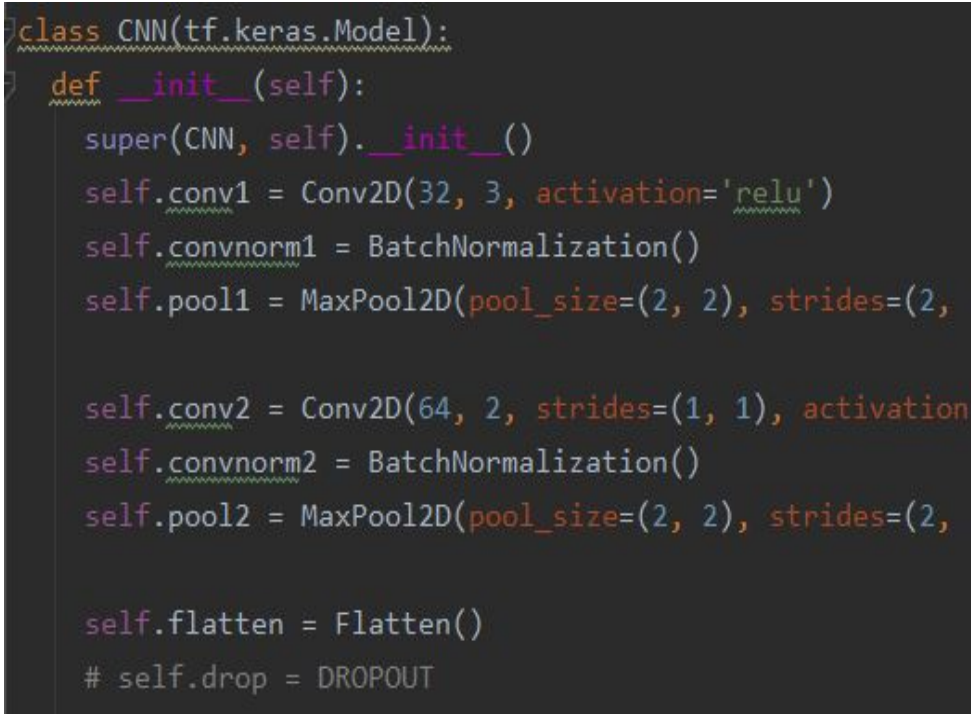


Frameworks

- Tensorflow -2.0

Tensorflow 2.0

- Model subclassing
- Optimizer: Adam vs SGD
- Loss: Sparse Categorical Cross Entropy
- Number of layers: 2

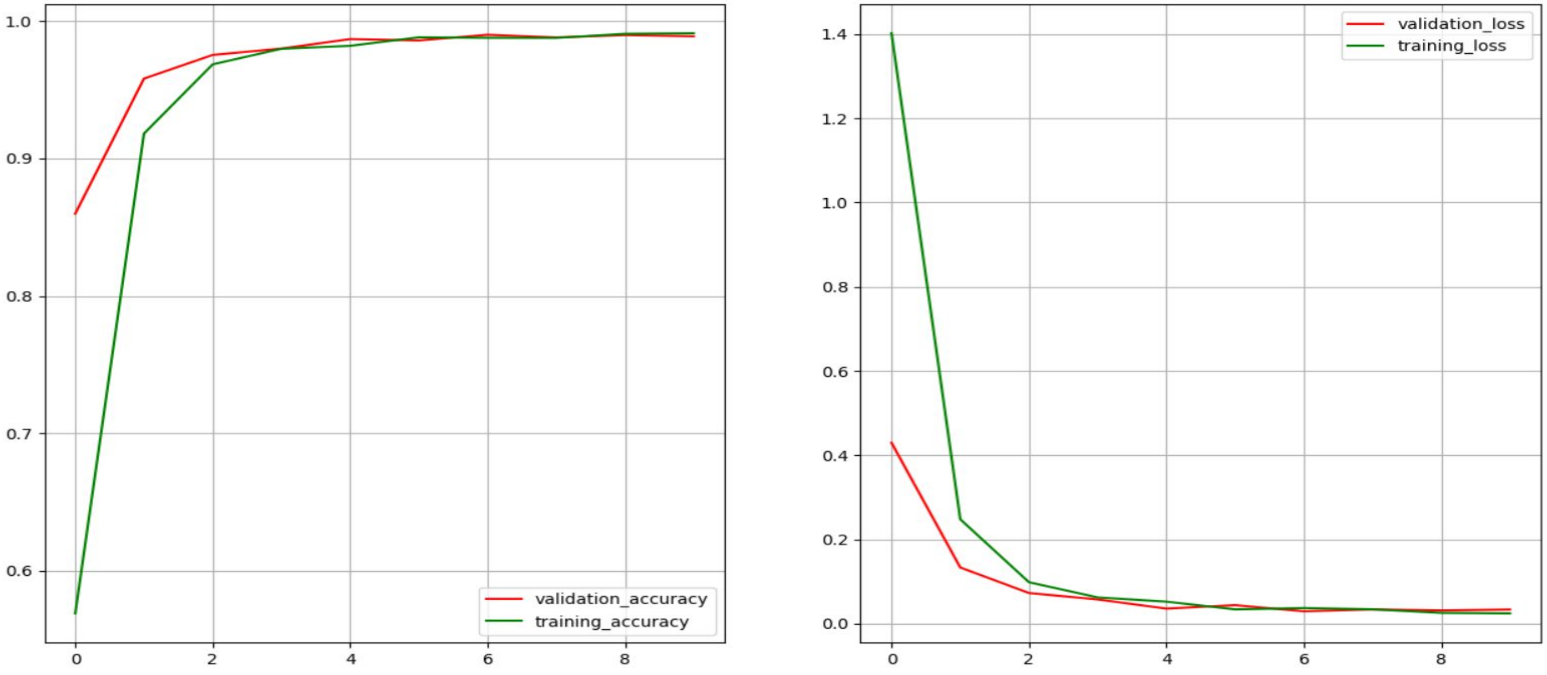


```
class CNN(tf.keras.Model):
  def __init__(self):
    super(CNN, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.convnorm1 = BatchNormalization()
    self.pool1 = MaxPool2D(pool_size=(2, 2), strides=(2,

    self.conv2 = Conv2D(64, 2, strides=(1, 1), activation
    self.convnorm2 = BatchNormalization()
    self.pool2 = MaxPool2D(pool_size=(2, 2), strides=(2,

    self.flatten = Flatten()
    # self.drop = DROPOUT
```

Result - Tensorflow 2.0

```
Accuracy on test set: 92.38%
```

ACCURACY / LOSS



Conclusion & Further Approach

- For this project, Tensorflow 2.0 is the winner!
- Future research: Attempt to translate videos of words utilizing knowledge gained from this project

Thank you!

Questions? Suggestion?