Shuru Wang, Alice Chu

Deliverable 2

Dataset

This project is designed to predict the corresponding alphabet of the input sign alphabet images using CNN algorithm for classification. We changed the dataset to this one. This dataset contains 27455 training images and 7172 test images all with a shape of 28 x 28 pixels such that each pixel has grayscale value between 0 - 255. These images belong to the 25 classes of English Alphabet starting from A to Y. The classes are labeled from A to Y respectively. For preprocessing, we will divide each pixel by 255 to convert it into a value between 0 to 1.

Framework and Tools

We use Keras library to implement CNN, numpy for linear algebra, pandas and sklearn for data preprocessing, matplotlib to visualize the images. The CNN includes an input layer, a convolutional layer, a pooling layer, and an output layer. We use a gray scale input layer because color is not important, max-pooling to reduce the number of parameters to optimize.

We plan to implement four layers: an input layer, a convolutional layer with ReLu as activation function, a max pooling layer, and a fully connected output layer. The output layer will use softmax as an activation function with 25 neurons. The dataset will be splitted into ratio 80:10:10 for train: valid:test. One regularization is to use dropout to reduce overfitting and improve generalization error. For optimization, we will use Adam optimizer as it has faster computation time and requires fewer parameters for tuning.

Hyperparameters and Design

Hyperparameters of the CNN architecture include kernel (filter) size, a matrix of weights used to convolve patch of input and measure how close it resembles a geometric feature; pooling-layer parameters, which have the same parameters as a convolutional layer; number of neurons in the layer; and the number of hidden layers.

We use large odd-size filters (3x3,5x5), large because the number of pixels in the dataset are large and odd-size by convention. The number of neurons in the input layer is the same as the number of features (columns in the data), while that in the output layer is the same as the number of class labels (since we are using softmax), which is 25. The optimal size of the hidden layers chosen is between the size of the input and size of the output layers. We will compare the accuracy scores of both training dataset and testing datasets. If the train accuracy is higher than test accuracy, then the model is overfitting.

Preliminary results

Using the architecture described above (one convolutional layer, one pooling layer, one dropout (regularization) layer, we obtained 81.07% training accuracy:

Layer (type)	Output Shape	Param #
conv2d_17 (Conv2D)	(None, 24, 24, 64)	1664
max_pooling2d_17 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_24 (Dropout)	(None, 12, 12, 64)	0
flatten_7 (Flatten)	(None, 9216)	0
dense_14 (Dense)	(None, 600)	5530200
dropout_25 (Dropout)	(None, 600)	0
dense_15 (Dense)	(None, 25)	15025

Total params: 5,546,889
Trainable params: 5,546,889
Non-trainable params: 0

```
687/687 [==============================] - 52s 75ms/step - loss: 0.6830 - accuracy: 0.8107
```

 # CNN Architecture Code

```
cnn_model.add(Conv2D(64, (5, 5), input_shape = (28,28,1),
activation='relu'))
cnn_model.add(MaxPooling2D(pool_size = (2, 2)))
cnn_model.add(Dropout(0.20))
cnn_model.add(Flatten())
cnn_model.add(Dense(units = 600, activation = 'relu'))
cnn_model.add(Dropout(0.20))
cnn_model.add(Dense(units = 25, activation = 'softmax'))
```

Future Plan

The dataset requires fewer preprocessing procedures and more options are available such as how many layers in implementation. In the future, we might implement additional layers to increase the precision and explore how the number of neurons, filter size, pooling size, and dropout rate affect the accuracy.

Shuru Wang, Alice Chu

References

https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7

https://analyticsindiamag.com/hands-on-guide-to-sign-language-classification-using-cnn/

https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

https://medium.com/@romainvrh/sign-language-classifier-using-cnn-e5c2fb99ef51