Metis Deep Learning Module

Interpreting Sign Language Using CNNs

Abstract

American Sign Language/ASL is a primary means of communication for deaf, hard of hearing and hearing nonverbal children who are nonverbal due to conditions such as down syndrome, autism and other speech disorders. Deaf students are considered as a linguistic minority and I'm hoping with this project to promote awareness and acceptance of the language among others. Majority of the general population is unable to communicate in sign language and the aim of this project is to interpret signs from ASL into English alphabets. Aim of this project is to create an interpreter which can translate hand gestures into English alphabets.

Design

Data was imported into a pandas dataframe and initial exploration was done using the plt.imshow by reshaping the array into 28X28. To create a baseline a multiclass (25 classes) logistic regression was performed which performed moderately well but misclassified most classes. After the baseline, data was reshaped to input into a convolution neural network. Data scaling was performed using standard scalar and reshaped to fit into the CNN. Model performed extremely well and the need to transfer learning could not be justified. However, transfer learning was attempted with weights from imagenet which ended up performing no better than CNN. Future work includes deploying the model using Flask.

Data

Data is downloaded from Kaggle as a CSV file and consists of image information for each alphabet in sign language in the form of pixels for each picture. Test and train datasets were separate with about 20,000 images in train and 7000 in test.

Algorithms

- Logistic Regression
    - Data was scaled using standscalar separately on train and test data
    - Training was performed without regularization
    - Class imbalance was checked
    - Accuracy was 0.86
- CNN
    - Data was scaled, reshaped for inputting. Target variable was transformed to a categorical variable
    - Initially 3 hidden layers with relu activation were constructed with an accuracy of 0.87 on validation
    - 2 more layers were added with relu activation with improved the accuracy by 0.98 on validation
    - Test accuracy was around 0.90 indicating some overfitting which was addressed using drop out, batch normalization, increasing units under the dense layer
- Visualizing Data
        - Seaborn was used to plot confusion matrix
        - Matplotlib for comparing actual and predicted pictures

Tools

- Scikit-learn for modeling
- Matplotlib and Seaborn for plotting

Communication

Presentation to the class