CS F215 Data Mining Presentation

Yash Bhagat	2017A7PS0063P
Sujeet Srivastava	2017A4PS0503P
Yash Gupta	2019A7PS1138P
Nevin Thomas	2017A7PS1175P

2018 4th IEEE International Conference on Information Management

Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation

Ali A. Alani
Aboozar Taherkhani
Georgina Cosma
T.M McGinnity

Introduction

- Hand Gestures present a promising a tool for interactions between humans and computers through sign language.
- Imagine the situation where the lives of human passengers and drivers is significantly eased due to automatic recognition of hand gestures by the car.

Introduction

- However, there are several challenges in hand gesture recognition - variable illumination, rotation, observability of hand, and hand size, to name a few.
- Traditional machine learning techniques heavily rely on digital image processing to eliminate noise and get an even illumination. Further, extensive domain knowledge is needed for feature extraction.

Introduction

- Deep learning algorithms, particularly **Convolutional Neural Networks (CNNs)** have displayed superior performance in image classification.
- Therefore, it is only natural to apply CNNs to hand gesture recognition, as attempted in this paper.

Introduction

- The authors have used the sign language of Peru (LSP) dataset to train and test their adapted CNN (ADCNN) model.
- It demonstrated a better performance compared to baseline CNN models run on the same dataset. **[99.73% vs 95.73%]**

Dataset: LSP

- The LSP dataset contains hand gestures used in the sign language of Peru.
- It was developed by Flores et al for their research (2017).
- It contains 3750 colored (rgb) hand gesture images from 25 different people.
- Each gesture belongs to one of 6 distinct classes.

TABLE II. DATASET CHARACTERISTICS

Dataset	Classes	Input Dimension	No. of samples	
			Training	*Testing*
LSP	6	(12288, 1)	2625	1125

Dataset: LSP

- Challenges are: **Non-regular illumination, Rotation of hands, Uneven hand sizes, Translation of the key components and Noise.**
- These challenges make the classification task complex.

Image Preprocessing

- To reduce the number of parameters and to reduce the computational requirements, the images will be preprocessed before feeding them to the CNN.
- RGB images are converted to black-only grayscale, achieving the aforementioned objectives.

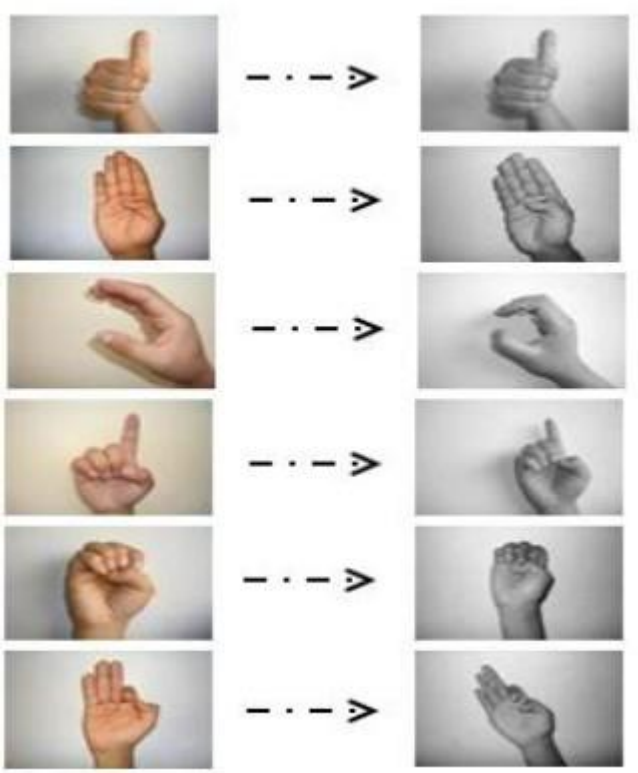


Figure 5. Image Pre-processing

Baseline Convolutional Neural Network

- Before delving into the specially tweaked version of CNN by the authors, let us see the basic plan for CNN to solve this problem.
- CNN will have three types of layers: convolutional layers, pooling layers, and fully-connected layers.

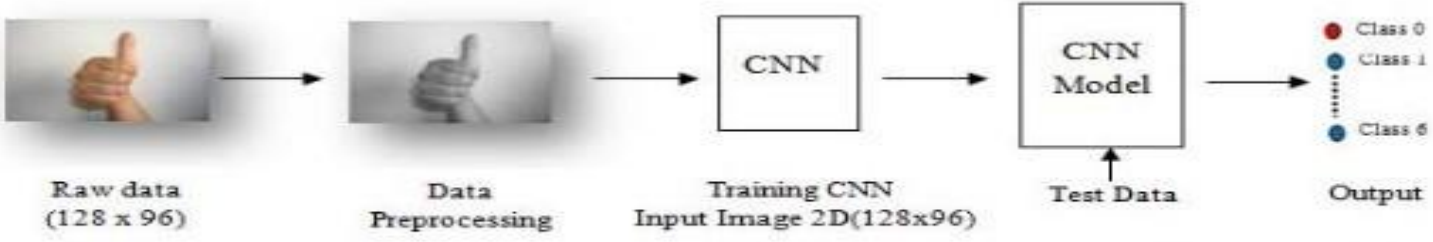


Figure 1. Overview of the proposed hand gesture recognition model.

Baseline Convolutional Neural Network

- Bottom layers collect information about low level features. As we move to deeper and deeper layers, the features become more high-level and abstract.
- A convolution layer investigates the spatially-local correlation of its input, mapping it into the next layer as a feature map.
- The max-pooling layer reduces the spatial size of representation to reduce number of parameters and computation in the network.

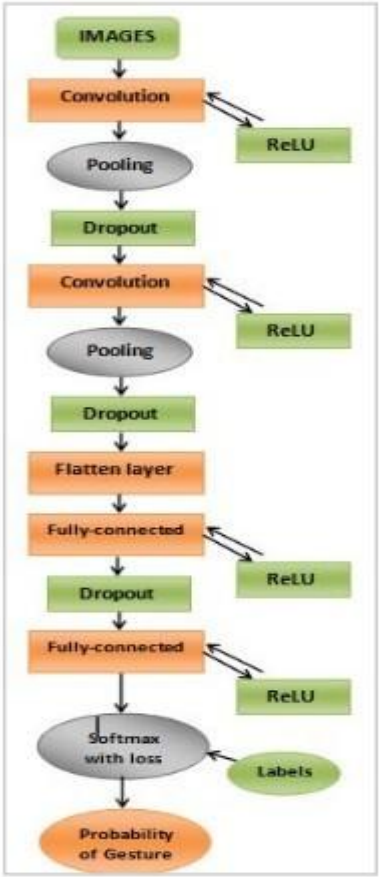
Baseline Convolutional Neural Network

- A flatten layer converts the two-dimensional matrix data to a vector for the fully-connected layer.
- Fully connected layer with ReLU activation function contains neurons for classification (here 128).
- Dropout mechanism drops outputs of random neurons (say 20%) at assigned stage to reduce overfitting.
- Finally Softmax activation function with 6 neurons, one for each distinct class of gesture gives the final output or prediction.

Baseline Convolutional Neural Network

TABLE I. BASELINE CNN CONFIGURATION

Layers Operation	Layers Configuration
Convolution	32 filters, 5x5 kernel and ReLU
Max-Pooling	2x2 kernel
Dropout	20%
Convolution	32 filters, 3x3 kernel and ReLU
Max-Pooling	2x2 kernel
Dropout	20%
Flatten layer	800 Neurons
Fully connected	128 Neurons
Dropout	20%
Fully connected	64 Neurons
Output layer	Softmax 6 classes



Adapted Convolutional Neural Network

- To improve the performance of the previously mentioned baseline CNN model, the authors used the following techniques.
- **Data Augmentation:** It is a well known fact in Data Science that the more data a learning algorithm has access to the more effective it can be.
- Transformations can be applied to the original images to obtain more images which increase the number of examples and also increase robustness of the model.

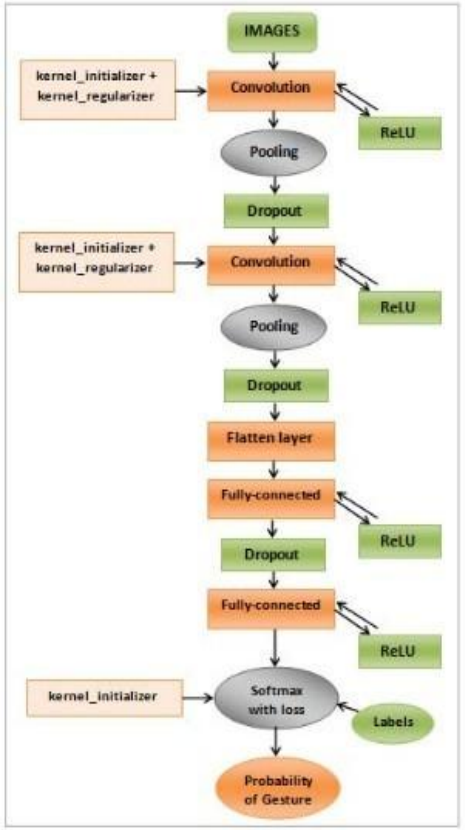
Adapted Convolutional Neural Network

- The authors applied vertical and horizontal shift operations to the images to an extent of 20% of the original dimensions.
- **Network Initialization:** Initializing weights can affect the ease with which the network learns from the training set.
- **L2 Regularization:** To decrease complexity of model, parameters with large weight magnitudes are penalized using L2 norm with hyperparameter $\lambda = 0.0001$.

Adapted Convolutional Neural Network

TABLE IV. CNN AND ADCNN PARAMETERS

Network	Baseline CNN (CNN)	Proposed CNN (ADCNN)
Number of training samples	2625	2625
Activation function	ReLU-Softmax	ReLU-Softmax
Learning rate	0.01	0.01
Iterations	10	10
Cost function	categorical crossentropy	categorical crossentropy
Optimization	Adam	Adam
Data augmentation	None	yes
Network initialization	None	ReUL(he_uniform) Softmax(glorot_uniform)
L2 Regularization	None	L2 Regularization



Evaluation Metrics

The common evaluation metrics for judging performance of CNNs are precision, recall, accuracy, and F1 score. Higher these metrics the better the model.
Here TP = true positive, FP = false positive
TN = true negative, FN = false negative

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

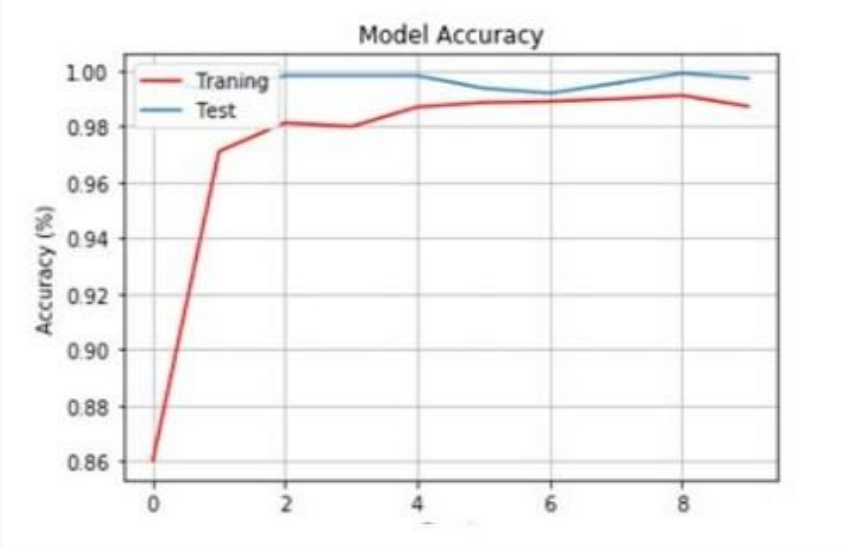
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

Results

Authors' ADCNN model achieved a 99.73% accuracy compared to 95.73% accuracy of basic CNN.

TABLE V. CLASSIFICATION RESULTS

Methods	No. Epoch	Precision	Recall	F1 Score	Accuracy (%)
Baseline CNN	10	0.96	0.96	0.96	95.73
Proposed ADCNN	10	1	1	1	99.73



Implementation

Since the Peruvian Sign language dataset wasn't publically available, the implementation of this paper has been done on a sample of American Sign language dataset.

The first 6 characters (A to F) have been used.

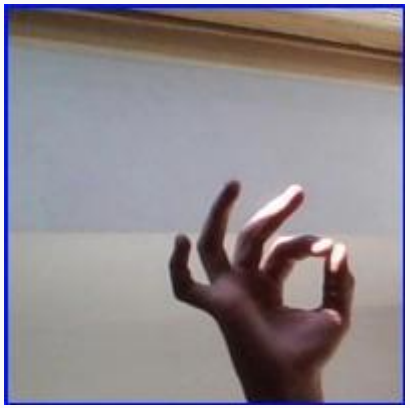
No. of Training Samples	3000
No. of Test Samples	600

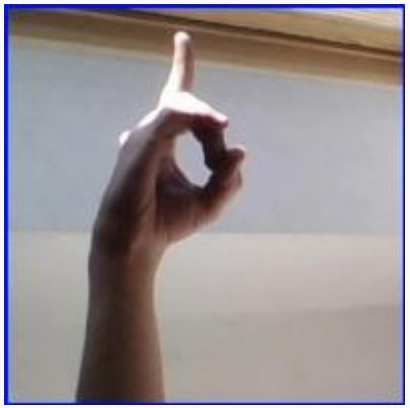
Methods	**No. of Epochs**	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Baseline CNN	30	0.92	0.92	0.92	92.00%
ADCNN	30	0.96	0.96	0.96	95.67%

Implementation

Classes a - f used in the dataset in that order [clockwise from top left]



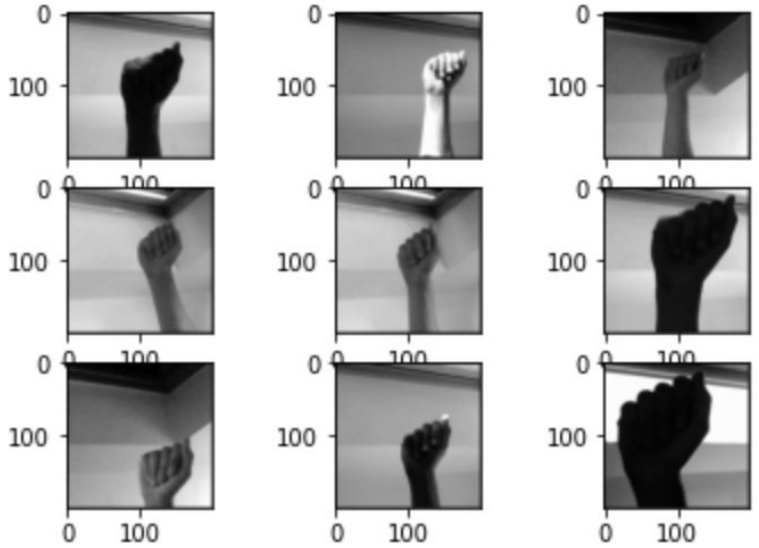





Implementation

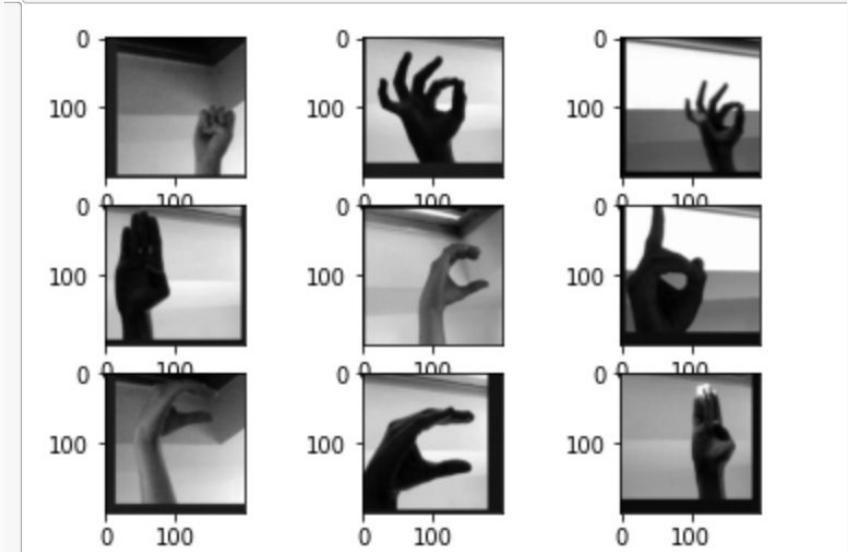
Pre- processed image

```
for i in range(9):
    plt.subplot(330 + 1 + i)
    image = images[i].astype('uint8')
    plt.imshow(image, cmap='gray')
```



Data Augmentation : After applying horizontal and Vertical shifts

```
for i in range(9):
    plt.subplot(330 + 1 + i)
    batch = it.next()
    image = batch[0].astype('uint8')
    plt.imshow(image, cmap='gray')
```



Implementation : Baseline CNN

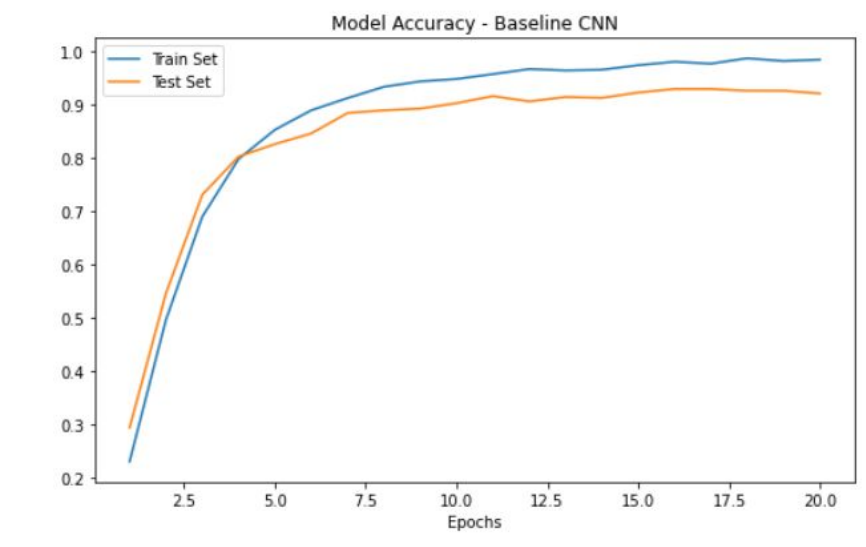
```
model = Sequential()

model.add(layers.Conv2D(32, (5, 5), padding='valid', input_shape=(200,200,1), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))

model.add(layers.Conv2D(32, 3, 3, padding='valid', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(0.2))#

model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dropout(0.2))#
model.add(layers.Dense(64, activation='relu'))#

model.add(layers.Dense(6, activation='softmax'))
```

Implementation : ADCNN

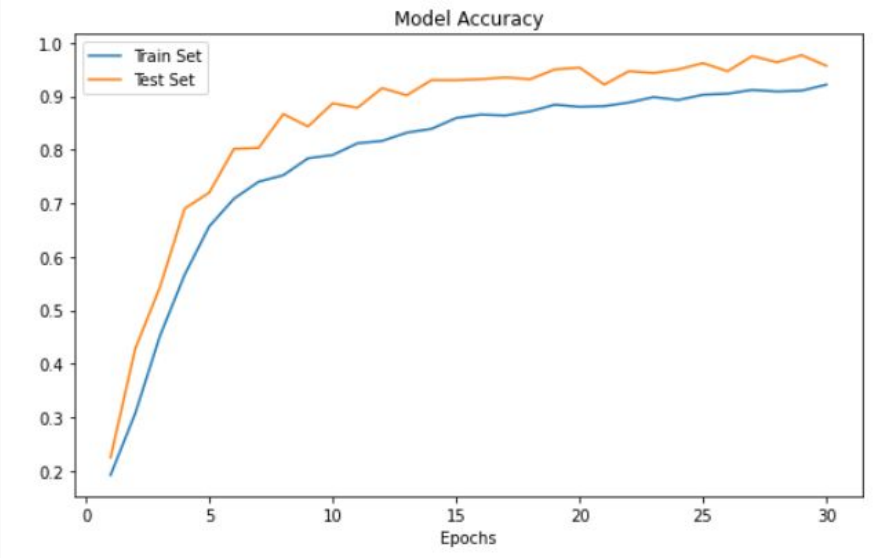
```
model = Sequential()
model.add(layers.Conv2D(32, (5, 5), padding='valid', input_shape=(200,200,1), kernel_regularizer=regularizers.l2(l2_lambda),
                        kernel_initializer='he_uniform'  ,activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))

model.add(layers.Conv2D(32, 3, 3, padding='valid', kernel_regularizer=regularizers.l2(l2_lambda),
                        kernel_initializer='he_uniform', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(0.2))

model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dropout(0.2))
model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(6, kernel_initializer='glorot_uniform', activation='softmax'))
```

Implementation

Baseline CNN



ADCNN



Implementation

Baseline CNN

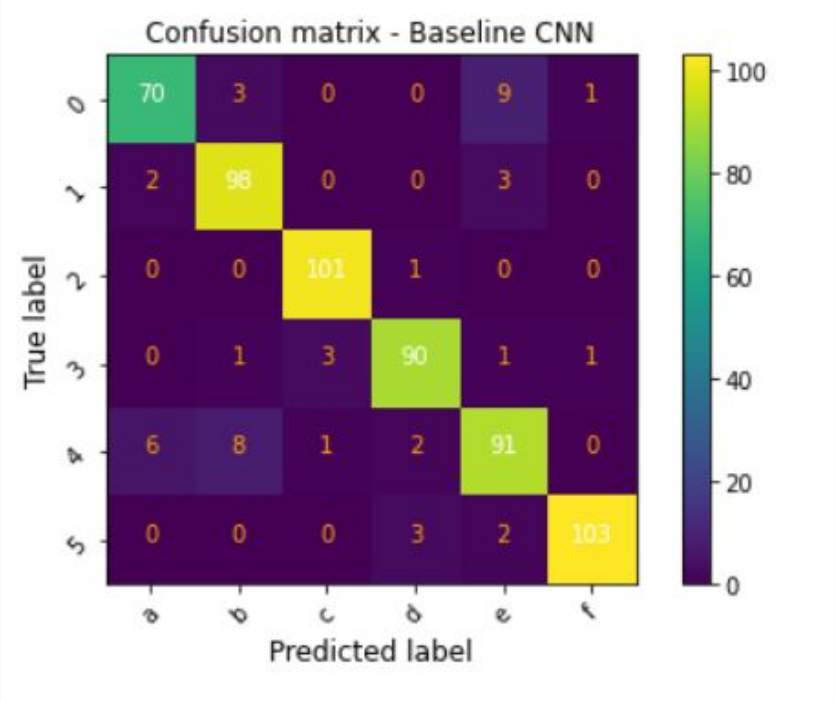
	precision	recall	f1-score	support
a	0.90	0.84	0.87	83
b	0.89	0.95	0.92	103
c	0.96	0.99	0.98	102
d	0.94	0.94	0.94	96
e	0.86	0.84	0.85	108
f	0.98	0.95	0.97	108
accuracy			0.92	600
macro avg	0.92	0.92	0.92	600
weighted avg	0.92	0.92	0.92	600

ADCNN

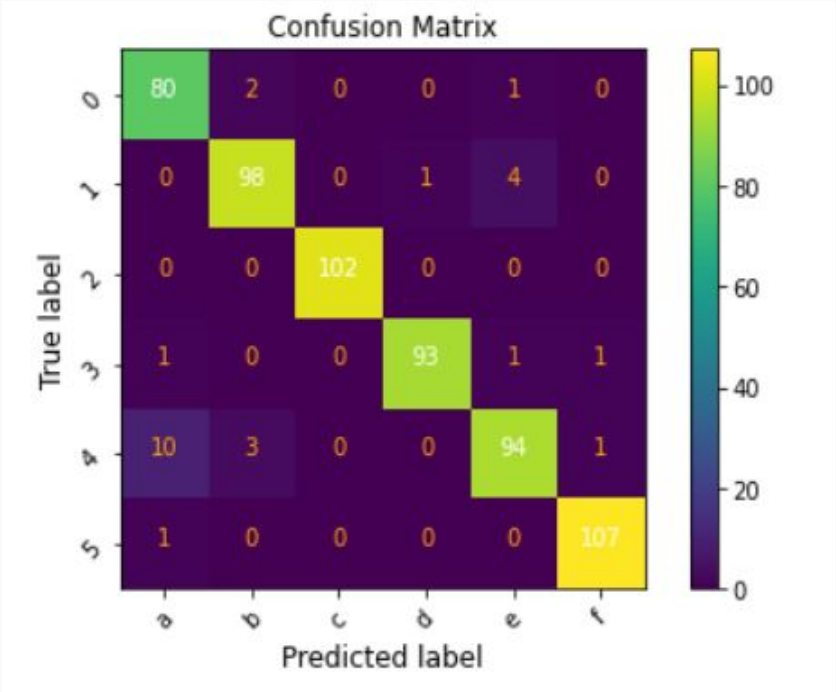
	precision	recall	f1-score	support
a	0.87	0.96	0.91	83
b	0.95	0.95	0.95	103
c	1.00	1.00	1.00	102
d	0.99	0.97	0.98	96
e	0.94	0.87	0.90	108
f	0.98	0.99	0.99	108
accuracy			0.96	600
macro avg	0.96	0.96	0.96	600
weighted avg	0.96	0.96	0.96	600

Implementation

Baseline CNN



ADCNN



Conclusion

- The implemented Baseline CNN and ADCNN algorithms work well on different datasets apart from the Peruvian sign language dataset.

- The number of epochs for which the training had to be given was increased to ensure that convergence occurs. It wasn't converging well on 10 epochs for American Sign language dataset.

- The accuracy obtained in ADCNN with data augmentation was more than that of Baseline CNN. This was because setting up the initialization and regularization parameters. Data augmentation also helped in increasing the size of dataset for training thereby improving the accuracy

Thank You.