1- Architecture used in the Paper

Each image is multiplied with a filter matrix, which is used to extract the feature from it. Initially the filer contains random values and it is a 3x3 matrix. The result matrix will be 32x32 and if we use more than one filter, the result will change. For example if we use 4 filters, the output will be 32x32x4.

The pooling process is to reduce the dimension of the image as well as the depth of image. This process is used to flattened the image, finally the array will be 1x1x1024. This array and the label is feed in to the training process with set of some parameters.

The training process will start with set of hyperparameters like no of epochs, number of images per epoch and the most important parameter, the training rate. The training rate must be very low so that the model can get the best fit. For each epochs, the loss value is calculated. Initially, the loss value will be larger, and in time it will comes to nearly zero. After the training is stopped, some value will be generated for the filters. We need to store those values. And we are using one-hot encoding, so if the detect object is correct, then there will be one in the object position in the label and other nine values will be zero.

2- Dataset details

- Name of the dataset used: American Sign Language

- The link of dataset: https://www.kaggle.com/datasets/kapillondhe/american-sign-language

- The total number of samples in the dataset : 121,608 samples

- Dimension of images (150 , 150)

- Number of classes & their labels : 27 classes with labels (from a to z and space sign)

- The ratio used for training, validation, and testing : Training (90% of the training dataset) = 97,200 images Validation (10 % of the training dataset) = 10,800 images ,

Testing (100% of the testing dataset) = 13608 images

3- Implementation details

- The Hyperparameters Used in The Model

```
my_model=Sequential()
my_model.add(Conv2D(64, kernel_size=7, strides=1, activation='relu', input_shape=[img_size,img_size,1]))
my_model.add(MaxPooling2D(pool_size=(2, 2)))
my_model.add(Conv2D(64, kernel_size=7, strides=2, activation='relu'))
my_model.add(Dropout(0.5))
my_model.add(Conv2D(256, kernel_size=3, strides=1, activation='relu'))
my_model.add(MaxPooling2D(pool_size=(2, 2)))
my_model.add(Conv2D(256, kernel_size=3, strides=2, activation='relu'))
my_model.add(MaxPooling2D(pool_size=(2, 2)))
my_model.add(Flatten())
my_model.add(Dropout(0.5))
my_model.add(Dense(512, activation='relu'))
my_model.add(Dense(27, activation='softmax'))
my_model.summary()
```

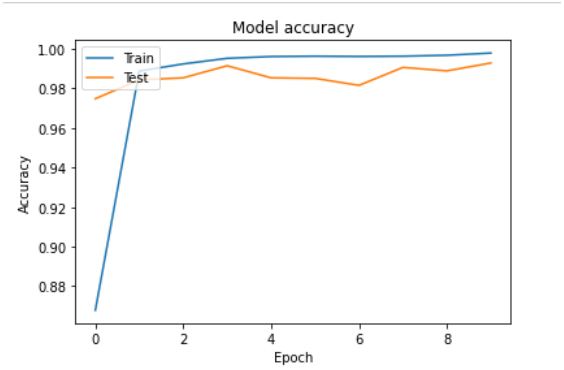
- Model Summery

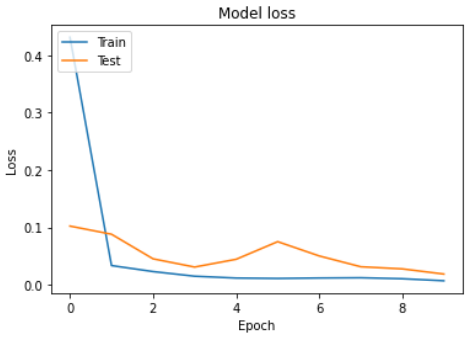
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 144, 144, 64)      3200
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 72, 72, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 33, 33, 64)        200768
_________________________________________________________________
dropout_1 (Dropout)          (None, 33, 33, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 31, 31, 256)       147712
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1180160
_________________________________________________________________
dense_1 (Dense)              (None, 27)                13851
=================================================================
Total params: 2,135,771
Trainable params: 2,135,771
Non-trainable params: 0
```

4- Results and visualizations

- The Accuracy : 99.29%

- The loss : 1.88%





- The Confusion Matrix