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.

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

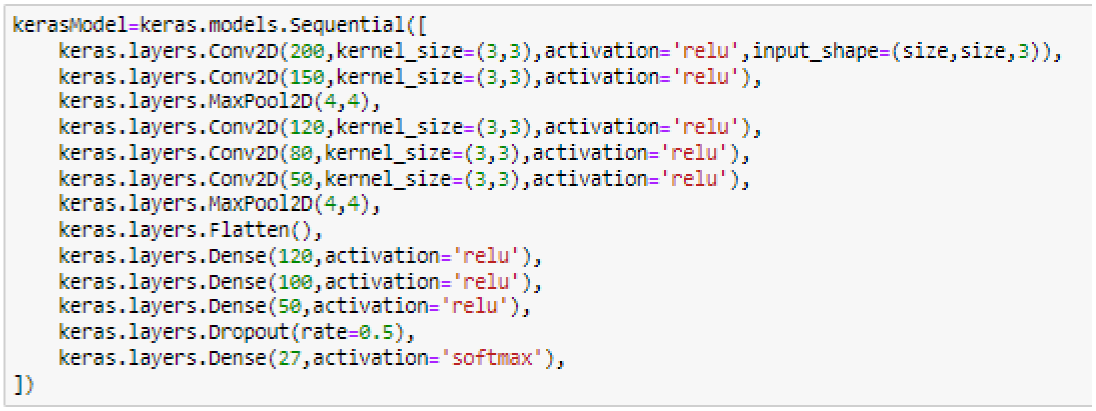
- Dimension of images (150 , 150)

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

- The ratio used for training, and testing : Training (86.84% of the training dataset) = 86,400 images testing dataset (13.51 % of the testing dataset) = 13,500 images

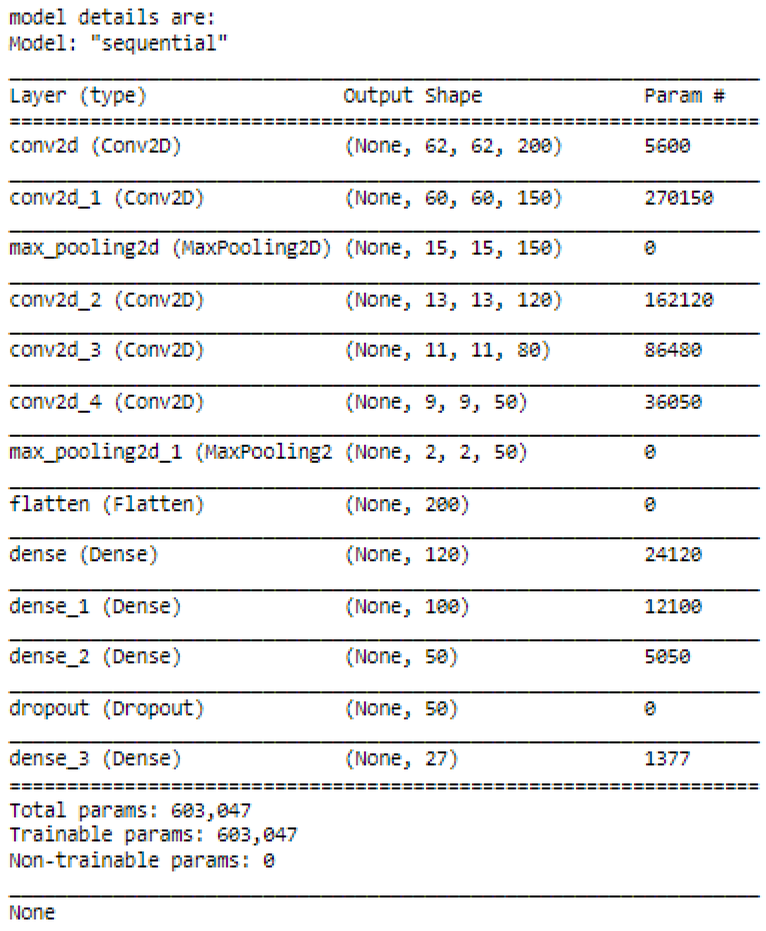
Implementation details

- The hyperparameters used in the model



```
kerasModel=keras.models.Sequential([
    keras.layers.Conv2D(200,kernel_size=(3,3),activation='relu',input_shape=(size,size,3)),
    keras.layers.Conv2D(150,kernel_size=(3,3),activation='relu'),
    keras.layers.MaxPool2D(4,4),
    keras.layers.Conv2D(120,kernel_size=(3,3),activation='relu'),
    keras.layers.Conv2D(80,kernel_size=(3,3),activation='relu'),
    keras.layers.Conv2D(50,kernel_size=(3,3),activation='relu'),
    keras.layers.MaxPool2D(4,4),
    keras.layers.Flatten(),
    keras.layers.Dense(120,activation='relu'),
    keras.layers.Dense(100,activation='relu'),
    keras.layers.Dense(50,activation='relu'),
    keras.layers.Dropout(rate=0.5),
    keras.layers.Dense(27,activation='softmax'),
])
```

- model summery



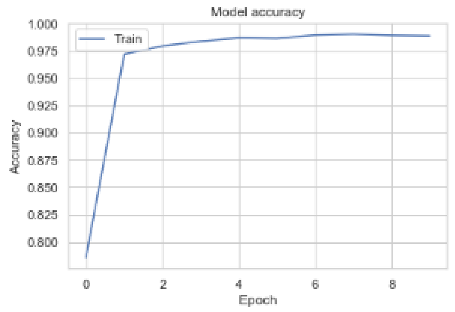
```
model details are:
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 200)       5600
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 150)       270150
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 150)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 120)       162120
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 80)        86480
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 9, 9, 50)          36050
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 50)          0
_________________________________________________________________
flatten (Flatten)            (None, 200)               0
_________________________________________________________________
dense (Dense)                (None, 120)               24120
_________________________________________________________________
dense_1 (Dense)              (None, 100)               12100
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 27)                1377
=================================================================
Total params: 603,047
Trainable params: 603,047
Non-trainable params: 0
_________________________________________________________________
None
```

- running epochs details

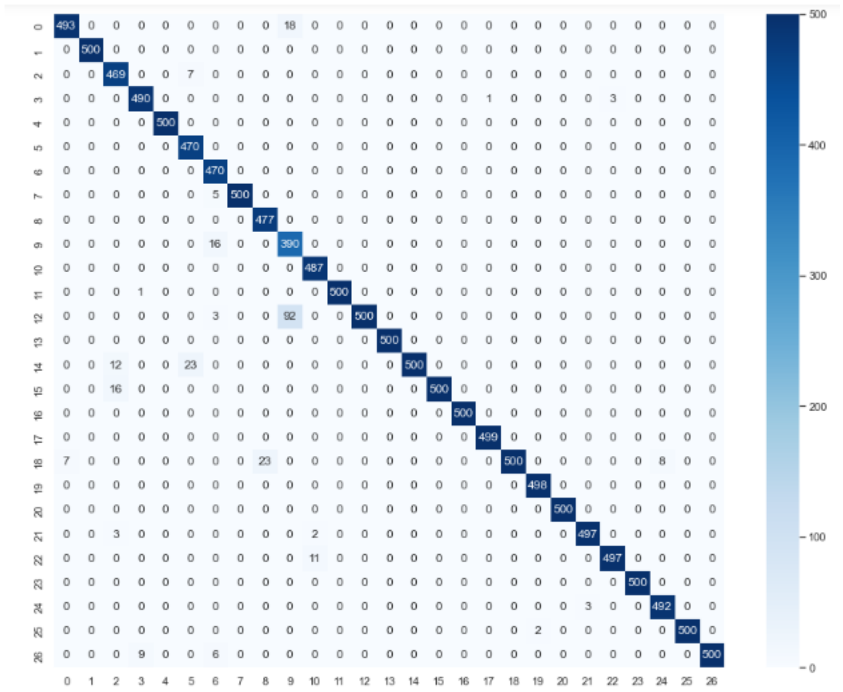
```
Epoch 1/10
2700/2700 [==============================] - 113s 39ms/step - loss: 0.6892 - accuracy: 0.7858
Epoch 2/10
2700/2700 [==============================] - 107s 40ms/step - loss: 0.0873 - accuracy: 0.9719
Epoch 3/10
2700/2700 [==============================] - 108s 40ms/step - loss: 0.0714 - accuracy: 0.9794
Epoch 4/10
2700/2700 [==============================] - 108s 40ms/step - loss: 0.0589 - accuracy: 0.9837
Epoch 5/10
2700/2700 [==============================] - 108s 40ms/step - loss: 0.0436 - accuracy: 0.9868
Epoch 6/10
2700/2700 [==============================] - 109s 40ms/step - loss: 0.0496 - accuracy: 0.9863
Epoch 7/10
2700/2700 [==============================] - 110s 41ms/step - loss: 0.0387 - accuracy: 0.9894
Epoch 8/10
2700/2700 [==============================] - 110s 41ms/step - loss: 0.0309 - accuracy: 0.9903
Epoch 9/10
2700/2700 [==============================] - 110s 41ms/step - loss: 0.0401 - accuracy: 0.9891
Epoch 10/10
2700/2700 [==============================] - 110s 41ms/step - loss: 0.0475 - accuracy: 0.9887
```

Results and visualizations

- The Accuracy : 97.99%

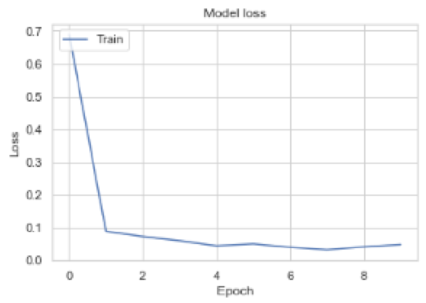


- The Confusion Matrix



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	493	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	469	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	490	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3	0	0	0	0
4	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	5	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	477	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	16	0	0	390	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	487	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	1	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	3	0	0	92	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	12	0	0	23	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	499	0	0	0	0	0	0	0	0	0
18	7	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	8	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	498	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0
21	0	0	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	497	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	497	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	492	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	500	0
26	0	0	0	9	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500

- The loss : 13.22%


Model loss
Train
Loss
Epoch