INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM, 600127

Project Report

on

English Alphabet classification from hand signs using CNN

Submitted by

(Group - 23)

KRISHNA KUMAR SUTAR (CED17I003)

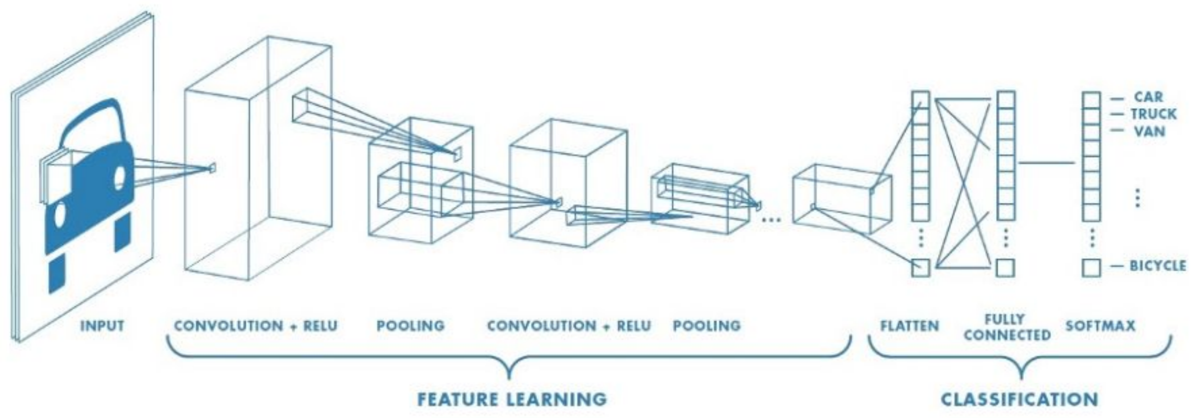
BAZIF RASOOL (CED17I015)

AMAR KUMAR (CED17I029)

TO

Dr. Umarani J.

Introduction



A **Convolutional Neural Network (ConvNet/CNN)** is a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.

Dataset ,Libraries,Preprocessing:

The dataset we are using is “American Sign Language Dataset” and we are using Keras with tensorflow backend as libraries in python to build our CNN model.We had to recognize alphabets “a - z” from the dataset.We used the “dataset_seperation.py” to separate the above dataset into “train”,”test”, “val” folders with a specified amount of original images divided among them.

CNN Architecture

Our Architecture has the following layers in order:

1. Convolutional Layer with 32 , 3x3 filters , relu activation
2. Max Pooling Layer with 2x2 Poolsize
3. Convolutional Layer with 32 , 3x3 filters, relu activation
4. Max Pooling Layer with 2x2 Poolsize
5. Convolutional Layer with 32 , 3x3 filters, relu activation
6. Max Pooling Layer with 2x2 Poolsize
7. Convolutional Layer with 32 , 3x3 filters, relu activation
8. Max Pooling Layer with 2x2 Poolsize
9. Flatten Layer
10. Dense Layer with 32 neurons, relu activation
11. Dense Layer with 32 neurons, relu activation
12. Dense Layer with 32 neurons, relu activation

13. Dropout Layer with 0.2 Dropout (for Regularization)
14. Dense Layer with 32 neurons, relu activation
15. Output Dense Layer with 36 neurons, softmax activation

Optimizer is chosen to be “adam” with “categorical_crossentropy” as loss function, for metric we have chosen “accuracy”

Input

We fed the above network images through the “flow_from_directory” method of ImageDataGenerator class provided by keras which does dataset augmentation by producing additional images by slightly modifying (shear , zoom,flip etc) images, this is used to train the model , validate it and then test.
Keras infers labels from directories so that is done automatically.

Parameters , System Specs :

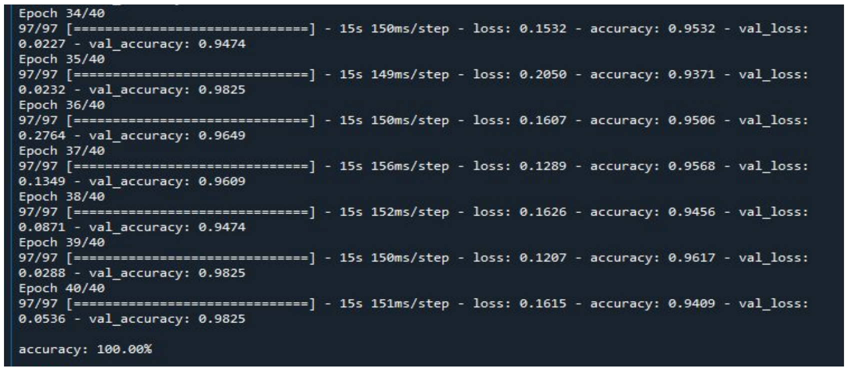
Batch size is chosen to be 16 and epochs are 40.
The model is being trained on a system with 8GB RAM, Nvidia GTX 1050 4GB, using Tensorflow-Gpu

Output

We achieved the best accuracy of 100% on the test set by using the above Network architecture and 99% accuracy on the validation set.

```
Found 1555 images belonging to 26 classes.
Found 130 images belonging to 26 classes.
Found 130 images belonging to 26 classes.
Epoch 1/40
97/97 [==============================] - 17s 175ms/step - loss: 3.1409 - accuracy: 0.0708 - val_loss: 2.4747 -
val_accuracy: 0.1875
..
.,
Epoch 11/40
97/97 [==============================] - 15s 153ms/step - loss: 0.5761 - accuracy: 0.7966 - val_loss: 0.3607 -
val_accuracy: 0.8684
Epoch 12/40
97/97 [==============================] - 15s 151ms/step - loss: 0.4934 - accuracy: 0.8330 - val_loss: 0.2986 -
val_accuracy: 0.8947
Epoch 13/40
97/97 [==============================] - 15s 151ms/step - loss: 0.4614 - accuracy: 0.8395 - val_loss: 0.2470 -
val_accuracy: 0.9035
```

```
Epoch 14/40
97/97 [==============================] - 15s 151ms/step - loss: 0.4454 - accuracy: 0.8473 - val_loss: 0.0821 -
val_accuracy: 0.9123
...
...
Epoch 40/40
97/97 [==============================] - 15s 151ms/step - loss: 0.1615 - accuracy: 0.9409 - val_loss: 0.0536 -
val_accuracy: 0.9825
accuracy: 100.00%
```

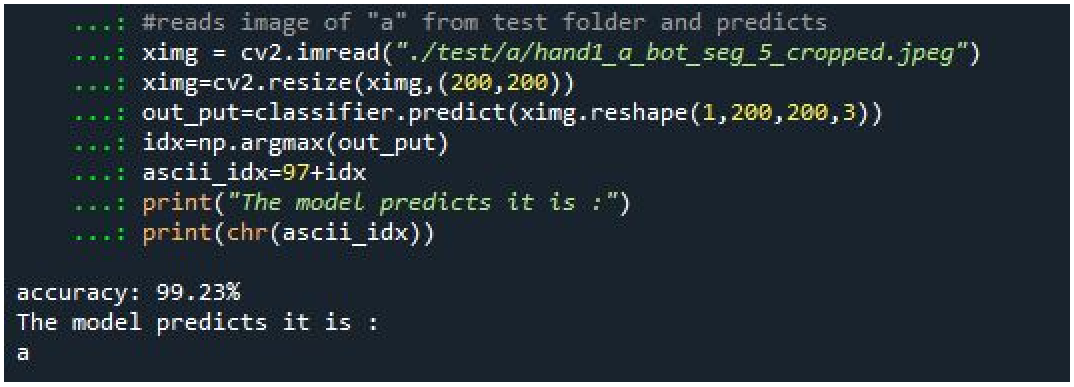


```
Epoch 34/40
97/97 [==============================] - 15s 150ms/step - loss: 0.1532 - accuracy: 0.9532 - val_loss:
0.0227 - val_accuracy: 0.9474
Epoch 35/40
97/97 [==============================] - 15s 149ms/step - loss: 0.2050 - accuracy: 0.9371 - val_loss:
0.0232 - val_accuracy: 0.9825
Epoch 36/40
97/97 [==============================] - 15s 150ms/step - loss: 0.1607 - accuracy: 0.9506 - val_loss:
0.2764 - val_accuracy: 0.9649
Epoch 37/40
97/97 [==============================] - 15s 156ms/step - loss: 0.1289 - accuracy: 0.9568 - val_loss:
0.1349 - val_accuracy: 0.9609
Epoch 38/40
97/97 [==============================] - 15s 152ms/step - loss: 0.1626 - accuracy: 0.9456 - val_loss:
0.0871 - val_accuracy: 0.9474
Epoch 39/40
97/97 [==============================] - 15s 150ms/step - loss: 0.1207 - accuracy: 0.9617 - val_loss:
0.0288 - val_accuracy: 0.9825
Epoch 40/40
97/97 [==============================] - 15s 151ms/step - loss: 0.1615 - accuracy: 0.9409 - val_loss:
0.0536 - val_accuracy: 0.9825

accuracy: 100.00%
```

Over multiple runs we consistently got accuracy over 96% by changing seeds(random) .

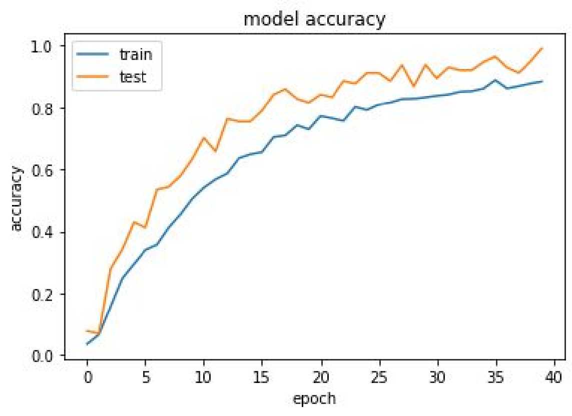
Below(Sample Prediction)

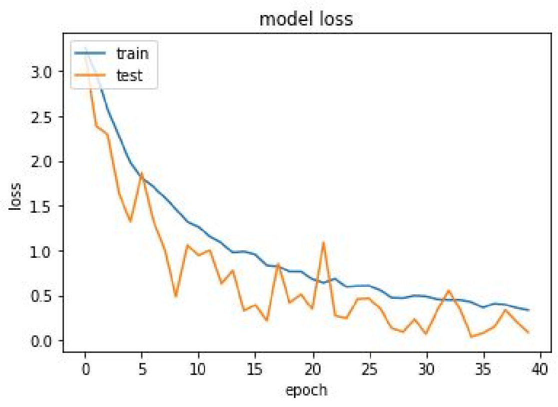


```
    ...: #reads image of "a" from test folder and predicts
    ...: ximg = cv2.imread("./test/a/hand1_a_bot_seg_5_cropped.jpeg")
    ...: ximg=cv2.resize(ximg,(200,200))
    ...: out_put=classifier.predict(ximg.reshape(1,200,200,3))
    ...: idx=np.argmax(out_put)
    ...: ascii_idx=97+idx
    ...: print("The model predicts it is :")
    ...: print(chr(ascii_idx))

accuracy: 99.23%
The model predicts it is :
a
```

Loss and Accuracy


model accuracy
train
test
accuracy
epoch


model loss
train
test
loss
epoch