Offline Handwritten Mathematical Symbols Recognition using CNN

Image Processing WS: 2021/2022
Kristian Hildebrand

Hinakoushar Tatakoti
Hafiz M Amir Hussain

https://github.com/Hinakoushar-Tatakoti/Hand-written-Math-sysmbol-recognization

Outline

- Dataset
- Sample size
- Tools used
- Model1
- Evaluation
- Prediction
- Model2
- Evaluation
- Prediction
- Results

Dataset

There are 82 classes of mathematical symbols including characters and digits that makes 375974 images.

special chars ---> '!' '(' ')' '+' ',' '-' ,'=','[' ']','{','}'

numbers ---> '0' '1' '2' '3' '4' '5' '6' '7' '8' '9'

Capital Letters ---> 'A','C','G' 'H' 'M' 'N' 'R' 'S' 'T' 'X'

Small Letters ---> 'b' ,'d','e','f','j' 'k' 'l', 'o' ,'p','q','u', 'v', 'w','y','z'

Maths symbol---> 'alpha', 'ascii_124','beta' ,'cos','exists','forall','forward_slash','gamma', 'geq', 'gt' ,'in', 'infty', 'int','lambda' ,'ldots', 'leq', 'lim', 'log', 'lt', 'mu', 'neq','lambda' ,'ldots','leq', 'lim' ,'log', 'lt', 'mu' ,'neq','rightarrow', 'sigma', 'sin', 'sqrt', 'sum', 'tan', 'theta', 'times'

Sample size

forward_slash : 199
int : 2742
y : 9340
leq : 973
sqrt : 8908
gamma : 409
5 : 3545
sigma : 201
infty : 1783
+ : 25112
6 : 3118
ldots : 609
} : 377
q : 1230
X : 26594
lt : 477

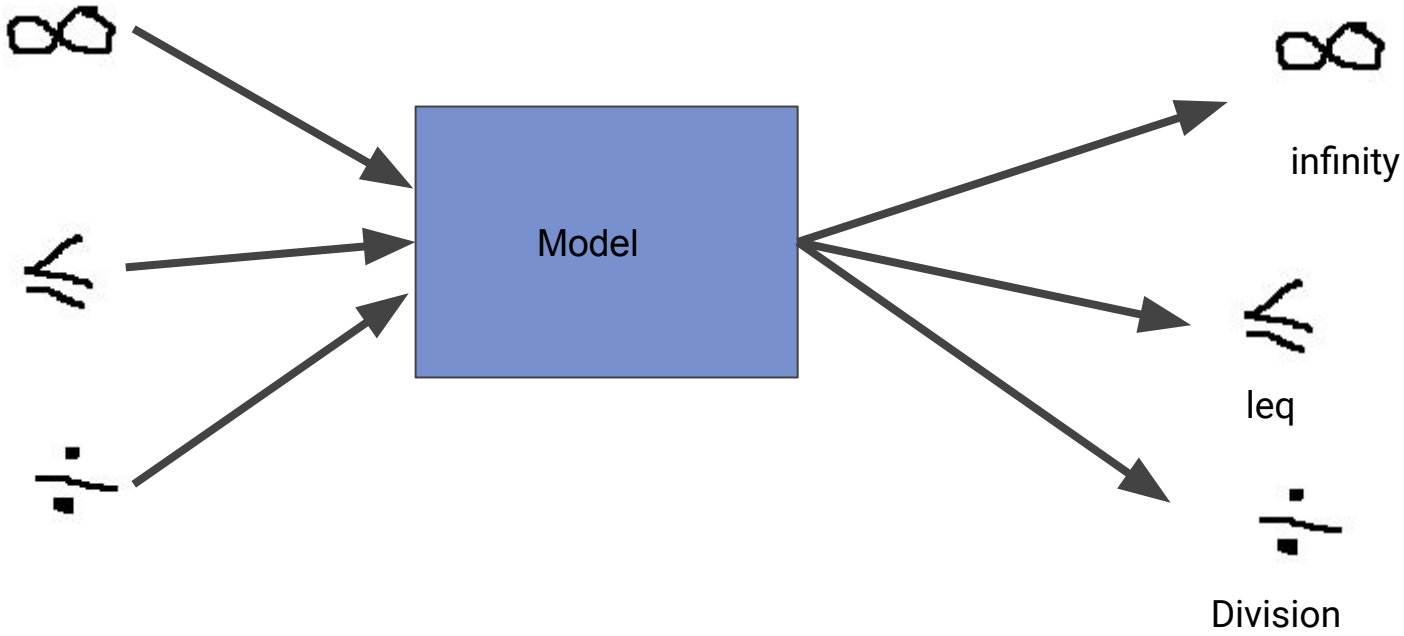
pi : 2332
ascii_124 : 1339
forall : 45
pm : 802
7 : 2909
gt : 258
{ : 376
cos : 2986
(: 14294
T : 3274
1 : 26520
o : 449
[: 778
C : 5802
sum : 2689
v : 1558
H : 1464

0 : 6914
= : 13104
in : 47
u : 1269
A : 12367
e : 3003
times : 3251
beta : 2025
2 : 26141
k : 3074
alpha : 2546
R : 2671
S : 1413
G : 1692
4 : 7396
, : 1906

rightarrow : 1703
) : 14355
i : 5140
! : 1300
exists : 21
z : 5870
prime : 329
w : 556
8 : 3068
M : 2476
lambda : 109
mu : 177
div : 868
theta : 2796
sin : 4293
Delta : 137
b : 8651

l : 1017
] : 780
lim : 1675
N : 10862
log : 2001
f : 3712
geq : 693
d : 4852
9 : 3737
neq : 558
p : 2680
phi : 355
j : 1536
- : 33997
3 : 10909
tan : 2450

Recognition



Tools used

Google colab + kaggle + github

Model 1- Training from scratch

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 44, 44, 32)        416

 activation (Activation)     (None, 44, 44, 32)        0

 max_pooling2d (MaxPooling2D  (None, 22, 22, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 21, 21, 32)        4128

 activation_1 (Activation)   (None, 21, 21, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 9, 9, 64)          8256

 activation_2 (Activation)   (None, 9, 9, 64)          0

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 activation_3 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 82)                5330

 activation_4 (Activation)   (None, 82)                0

=================================================================
Total params: 83,730
Trainable params: 83,730
Non-trainable params: 0
_________________________________________________________________
Number of layers of our model :  15
```

Accuracy :

loss: 0.3292 - accuracy: 0.8944 - val_loss: 0.1657 - val_accuracy: 0.9512

EPOCHS = 50

BS = 100 #Batch size

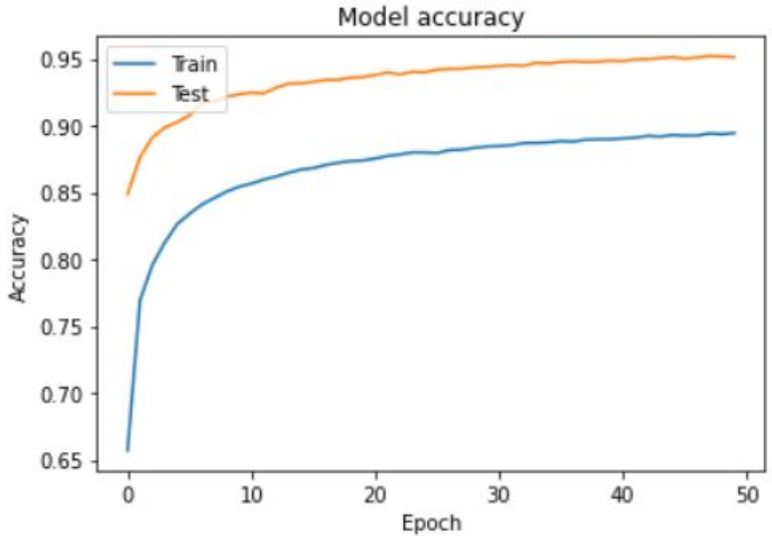
LR = 1e-3 #Learning rate 0.001

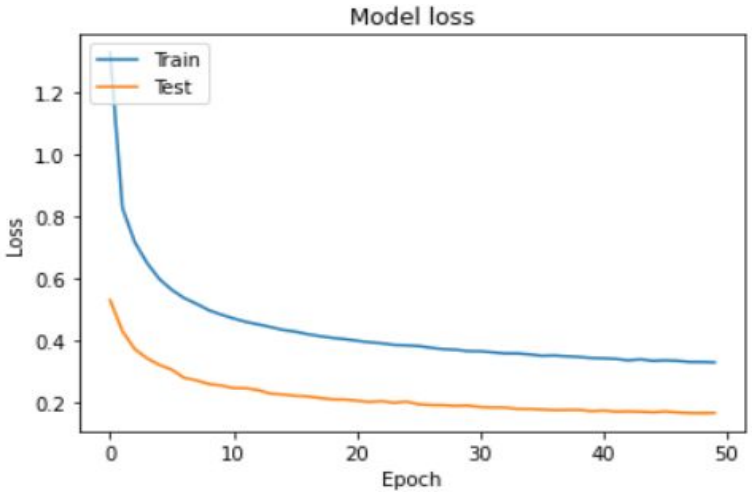
img_dim = (45,45,3)

Time taken : 130 mins total

Evaluation

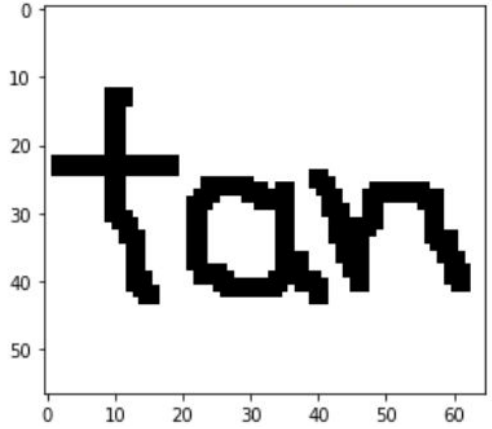
```
1129/1129 [==============================] - 40s 35ms/step - loss: 0.1657 - accuracy: 0.9512
Accuracy =  0.9511570334434509
```



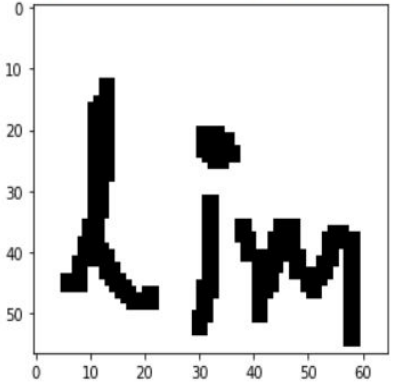


Prediction

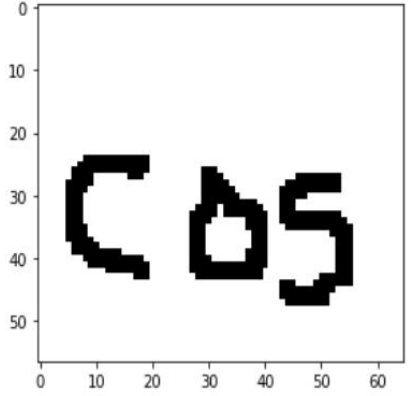
```
This Symbol is : t a n with probability match  97.76%
<matplotlib.image.AxesImage at 0x7f56817d7810>
```



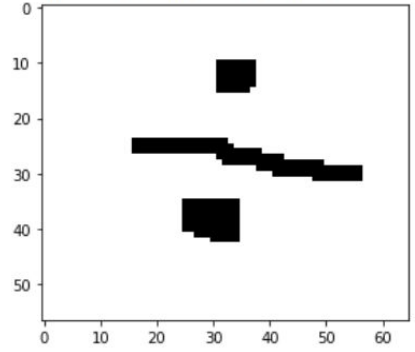
```
This Symbol is : l i m with probability match  99.99%
<matplotlib.image.AxesImage at 0x7f146f9ce2d0>
```



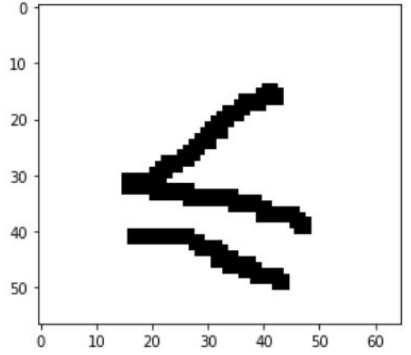
```
This Symbol is : t a n with probability match  80.35
<matplotlib.image.AxesImage at 0x7f56faa29ed0>
```



```
This Symbol is : d i v with probability match
<matplotlib.image.AxesImage at 0x7f5613e76990>
```



```
This Symbol is : 6 with probability match  20.50%
<matplotlib.image.AxesImage at 0x7f56fbb54810>
```



Results

Recognized positive: 20
Recognized Negative: 62

Conclusion:

Balanced data
More training Data
More GPU power
more epochs to get best fit

Model 2 - Pretarined on MobileNetV2

base_model=tf.keras.applications.MobileNetV2(include_top=**False**, input_shape=(128,128,3), pooling='max', weights='imagenet')

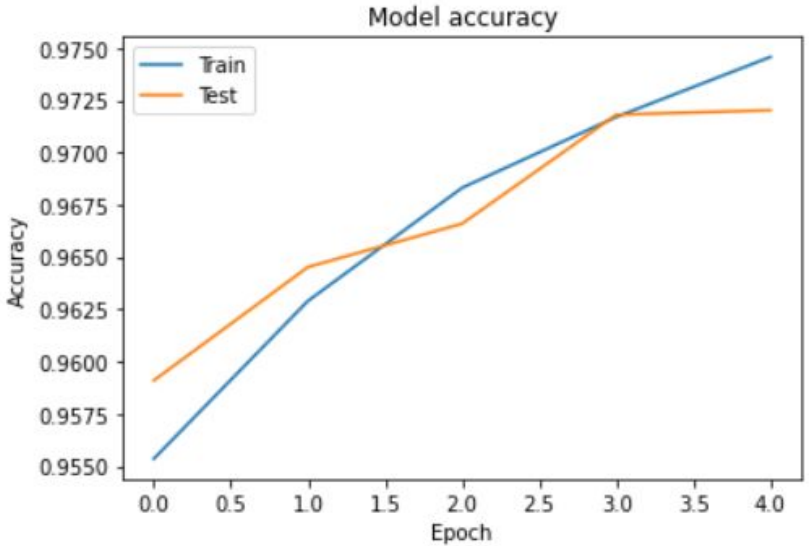
Last layer modified:
BatchNormalization
Dense
Dropout
Fully connected layer

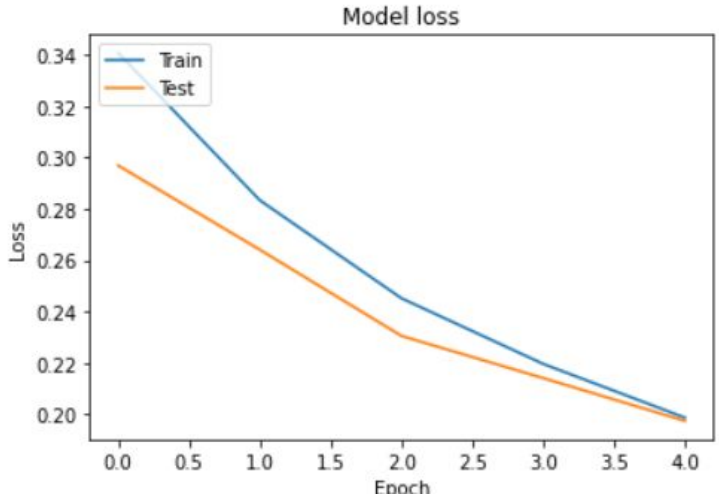
Epoch: 5
Time:60 mins for each epoch
learning_rate=.001

```
Epoch 1/5
10575/10575 [==============================] - 3539s 335ms/step - loss: 0.3406 - accuracy: 0.9553 - val_loss: 0.2968 - val_accuracy: 0.9591 - lr: 0.0010
Epoch 2/5
10575/10575 [==============================] - 3602s 341ms/step - loss: 0.2833 - accuracy: 0.9629 - val_loss: 0.2640 - val_accuracy: 0.9645 - lr: 0.0010
Epoch 3/5
10575/10575 [==============================] - 3574s 338ms/step - loss: 0.2451 - accuracy: 0.9683 - val_loss: 0.2305 - val_accuracy: 0.9666 - lr: 0.0010
Epoch 4/5
10575/10575 [==============================] - 3554s 336ms/step - loss: 0.2196 - accuracy: 0.9717 - val_loss: 0.2141 - val_accuracy: 0.9718 - lr: 0.0010
Epoch 5/5
10575/10575 [==============================] - 3641s 344ms/step - loss: 0.1987 - accuracy: 0.9746 - val_loss: 0.1974 - val_accuracy: 0.9720 - lr: 0.0010
```

Evaluation

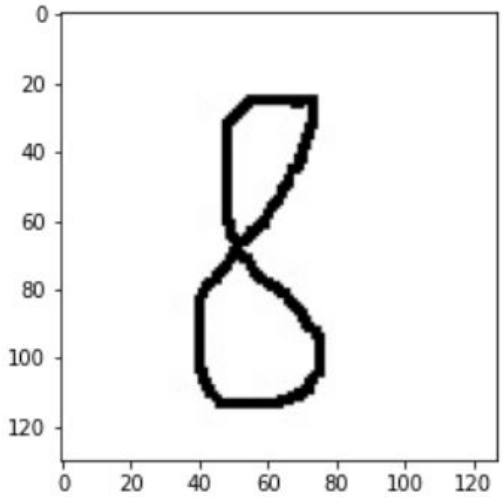
```
5875/5875 [==============================] - 340s 58ms/step - loss: 0.1819 - accuracy: 0.9752
0.18187175691127777 0.975179135799408
```





Prediction

```
4.18%
This Symbol is : - with probability match  ('-', '4.18%')
<matplotlib.image.AxesImage at 0x7f2e5f92a090>
```



Quite low prediction probability
Not comparable
Not able to achieve a expected prediction

Model 3 - Training from Scratch

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 45, 45, 32)        832

 max_pooling2d (MaxPooling2D  (None, 22, 22, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 18, 18, 48)        38448

 max_pooling2d_1 (MaxPooling  (None, 9, 9, 48)         0
 2D)

 conv2d_2 (Conv2D)           (None, 5, 5, 64)          76864

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 84)                21588

 dense_2 (Dense)             (None, 82)                6970

=================================================================
Total params: 210,494
Trainable params: 210,494
Non-trainable params: 0
```

Training Accuracy :

loss: 0.3323 - accuracy: 0.8979 - val_loss: 0.3079 - val_accuracy: 0.9053

Model Evaluation:

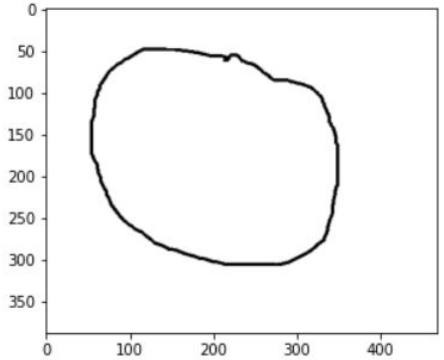
loss: 0.3079 - accuracy: 0.9058
0.3079117238521576
0.9057754278182983

Epoch: 3
Image dim: (45,45,1)
learning_rate = 5e-4 = 0.0005
Time taken: 30 mins for each epoch

Prediction

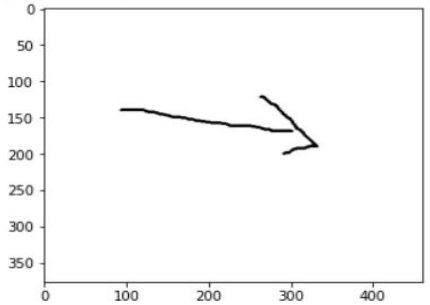
```
image1_path = '/content/sample_data/zero.png'
p1 = prediction(image1_path)
print(p1)
```

```
Prediction: 0
```



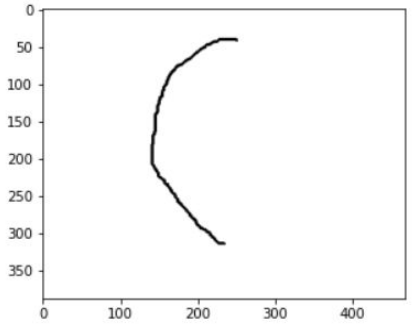

```
image3_path = '/content/sample_data/rightarrow.jpg'
p3 = prediction(image3_path)
print(p3)
```

```
Prediction: ldots
```



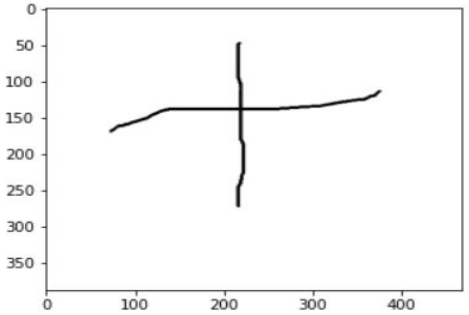

```
image3_path = '/content/sample_data/left_curve.png'
p3 = prediction(image3_path)
print(p3)
```

```
Prediction: (
```




```
image2_path = '/content/sample_data/plus.png'
p2 = prediction(image2_path)
print(p2)
```

```
Prediction: ldots
```



Summary:

- Trained from scratch model recognized better than pretrained model.
- Pretrained model using MobileNetV2 (weights imagnet) not quite good prediction.
- The higher dimension images not performed well during predictions.
- Model with 3 Conv layers worked better.
- More GPU power to try with different hyperparameters.
- Google colab often disconnects the session it’s problematic for training.
- Drawback faced, saving a model and weights directly into Github not possible.

Thank You

References

- https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch
- https://towardsdatascience.com/creating-a-trashy-model-from-scratch-with-tensorflow-2-an-example-with-an-anomaly-detection-27f0d1d7bd00
- https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342
- https://www.kaggle.com/xainano/handwrittenmathsymbols
- https://github.com/ThomasLech/CROHME_extractor
- https://stackoverflow.com/questions/67266161/how-to-train-amd-test-dataset-of-images-downloaded-from-kaggle
- https://github.com/RichmondAlake/tensorflow_2_tutorials/blob/master/13_lenet-5.ipynb