Machine Learning

Project Report

Indian Denomination Classification

Team Member 1: Aryan Tyagi (101816022)

Team Member 2: Mehul Singh Teya (101816021)

Batch: COSE 3rd Year

Group: 3CS1

Submitted to: Dr. Parteek Bhatia

Introduction

Our aim was to build a system that can classify the different denominations in an image, so we created a denomination classifier that can identify the different currency note in a given image.

This classifier can classify amongst the currency notes of Rs.10, Rs.20, Rs.100 & Rs.200. This classification can further be expanded and trained for other different kind of currency notes and for any currency, for demonstration purposes we took only 4 classes.

This Entire project is implemented in Python 3.8

TensorFlow, Keras are used for core implementation along with other modules and Google Collaboratory for faster computations is used.

Need of the System

Our system can be implemented at as many as places where there is a hard cash transaction involved. At Banks, Post Offices, Railway Stations, Bus Stations etc.

In rural areas, where there are a smaller number of workforce & manpower in post offices, this system can help in reducing workloads, by identifying the amount of currency notes.

This system can also help physically challenged (blind) people to verify the currency notes, as we can also combine this system with an audio feedback, enabling them to verify notes.

Problem Statement

Our main problem was to device a system that can differentiate between different currency denominations, to minimise the human efforts, and to automate & digitalize the identification of currency notes, to help differently abled people to handle money transactions in a better way

Speaking about the blind people, our Indian currency notes are not made in that way that a blind person can identify them, by adding an audio feedback, it can successfully help in identifying the note.

At post offices where, the manpower is not enough, this system can assist in identification of the denomination, similar kind of scenario can be solved at the banks also.

Data Collection

Data is collected from www.data.mendeley.com (source 1) which is having total 556 different images, having 139 images in each category for Rs.10, Rs.20, Rs.100 & Rs.200 notes in different orientations & with variable backgrounds. In some pictures even hands are there while clicking the images.

Due to less number of images, we also captured videos (source 2), featuring currency notes (shot in HD, for clearer images) & then screenshots were extracted from them, adding to our dataset.

Data was then segregated manually into different folders, for training purposes & simultaneously data cleaning & pre-processing, like removing blurred, duplicate, distorted images, was done.

Data Pre-processing

- All the images were mixed by default. Images were then segregated into different folders

- Duplicate images were removed from the dataset, as they could result in low accuracy of the model.

- Some images were having water marks on them, some notes were having something written on them, all that kind of images were dropped off from the dataset.

- Some images were having different extensions, jpeg & jpg. They all were converted into jpg format with the help of `pathlib` module.

Individual Contribution

Aryan:

- Gathered images for the dataset
- Resized the images to save computation time
- Removed unfit and watermarked images from the dataset

Mehul:

- Gathered images for the dataset
- Organized the images into class-wise folders
- Converted the images from `.png` to `.jpg` wherever required

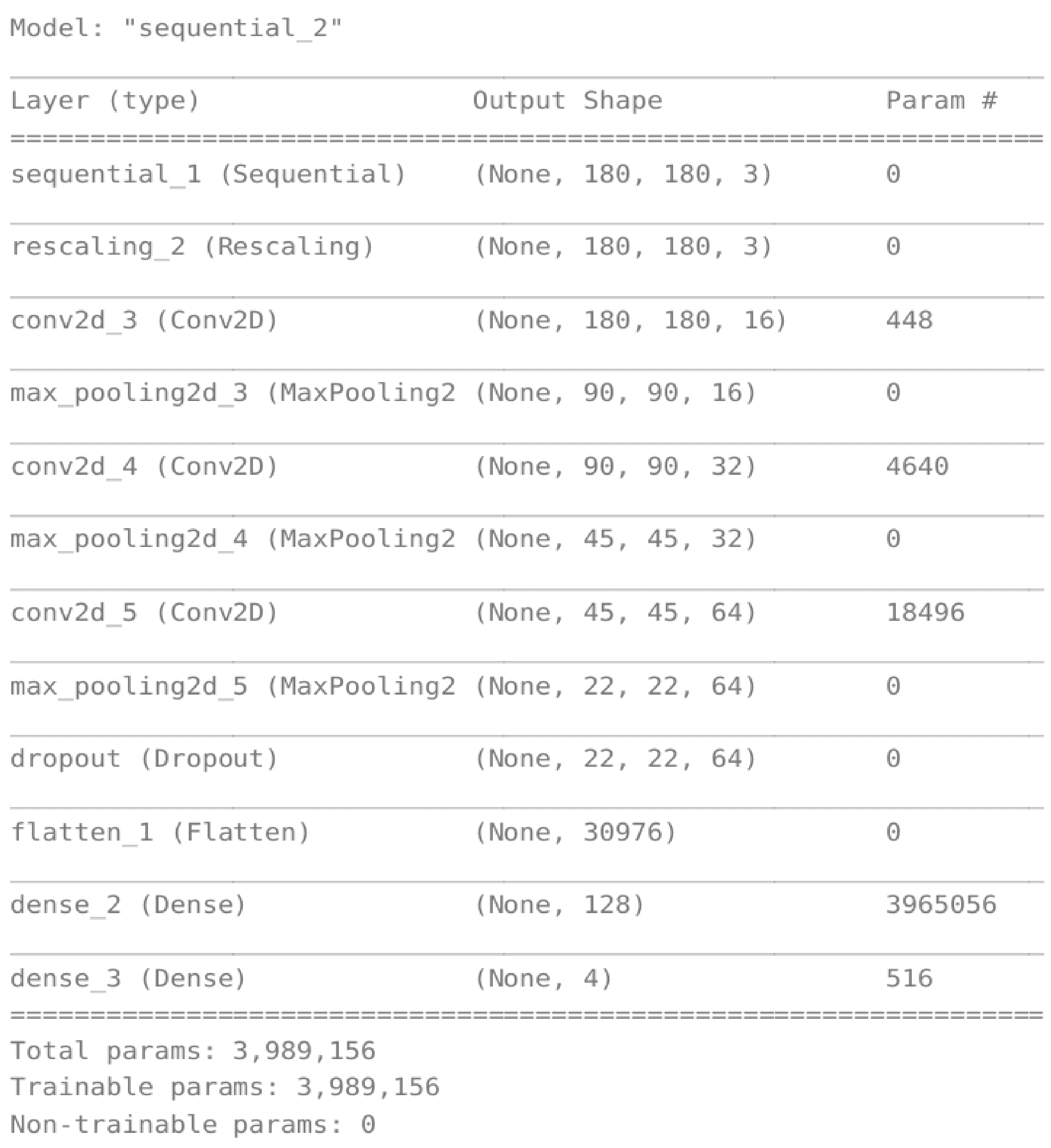
The coding and documentation work were done by both of us collaboratively.

Methodology

- First, we procured dataset
- Defined parameters for the loader, set the `batch_size`, set the input `image_size` as `180x180`
- Split the dataset in 80-20 ratio for training and validation datasets
- Then each class was represented properly, visualized the data using `matplotlib.pyplot`
- The RGB channels were also standardized to fit in the [0, 1] range from the [0, 255] as neural networks prefer that range. This was done by using a Rescaling layer.
- We then created a sequential model and used `relu` as the activation function
- We used data augmentation to make up for the rather small size of our dataset.
- We also used `Dropout` to reduce the overfitting
- Then compiled the model, with optimizer set to ‘`adam`’ & loss function as ‘`SparseCategoricalCrossentropy`’
- We then trained the model for `100 Epochs`
- We then tested the model on new data
- Finally, we saved the model using `Keras` for embedding it in our python script

Summary of the model:

```
model.summary()
```



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_1 (Sequential)    (None, 180, 180, 3)       0
_________________________________________________________________
rescaling_2 (Rescaling)      (None, 180, 180, 3)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 30976)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               3965056
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 516
=================================================================
Total params: 3,989,156
Trainable params: 3,989,156
Non-trainable params: 0
```

ML Techniques used

- Deep Learning (CNN)
- Classification
- Rescaling
- Data Augmentation
- Dropout

Experimentation

First, we tried to train the model for 10 Epochs, but we only achieved 80% accuracy, over the validation set.

Then, we realised that our model was overfitting, so we used Dropout to reduce the overfitting

We also realised that our dataset was small in size, so we used data augmentation to increase its size

Conclusion

We trained our model using TensorFlow and we were able to get an accuracy of 92%

Previously, you have to manually enter the image path in the script but then we modified the script to get the input from the web cam. Just run the script, it will prompt you to input the image through the web cam, and after clicking the picture (like holding the note with your hand(s)), it will predict the denomination from the image.

The values of the parameters are as follows:

```
Epochs = 200

optimizer = ‘adam’

loss function = ‘SparseCategoricalCrossentropy’
```

Limitations

- This system can't read multiple notes in a single image. The image should only contain a single note of currency.

- Till now, it can not detect the location of the note in image. It only predicts amount of the note

- It cannot detect multiple notes of same currency, of same denomination.

Future Scope

- We can make it self-learning, so that whenever anyone gives an image to our model, it only classifies the note, but also add that image into the dataset and get trained on it. So, it will keep on adapting and improving itself as we use it to predict.

- We can extend it to any kind of currency with any number of classifications for denomination.

- We can also install support with audio feedback, or turn it into a fully fledged software, to make it more interactive with the outer world. Audio feedback can help blind people to use this system.

- We can also train this software detect any kind of cracks, or is the note is in whole shape or not.

- With the help of object detection, we can also detect multiple notes in a single image, having same notes of single/multiple currency.

References

The documentations and references that were helpful in completion of this project

[1] www.keras.io

[2] www.kaggle.com

[3] www.tensorflow.org

[4] www.data.mendeley.com

THANK YOU