Professor – Vijayan Sugumaran

Project by – Vasudha Gulati, Shalini Sarathy Komandur Chakravarthy, Priya Sankar

Counting coins

MIS-4900 Deep Learning and Text Analytics



Table of Contents

[Introduction 2](#_Toc121572932)

[Problem statement 2](#_Toc121572933)

[Datasets 2](#_Toc121572934)

[Exploratory Data Analysis 3](#_Toc121572935)

[CNN Model 4](#_Toc121572936)

[Model 1: Base Model 4](#_Toc121572937)

[Data Transformation 4](#_Toc121572938)

[Model Building 5](#_Toc121572939)

[Model Application 6](#_Toc121572940)

[Model Predictions 8](#_Toc121572941)

[**Model 2: Augmented Model** 8](#_Toc121572942)

[Data Transformation 8](#_Toc121572943)

[Model Building 9](#_Toc121572944)

[Model Application 10](#_Toc121572945)

[Model Predictions 13](#_Toc121572946)

[Challenges 13](#_Toc121572947)

[Conclusion 13](#_Toc121572948)

[Fast AI Model 14](#_Toc121572949)

[Data Transformation 14](#_Toc121572950)

[Model Building 15](#_Toc121572951)

[Model Application 16](#_Toc121572952)

[Model Dissemination 16](#_Toc121572953)

[Model Predictions 16](#_Toc121572954)

[Challenges 18](#_Toc121572955)

[Conclusion 18](#_Toc121572956)

[Team Contribution 18](#_Toc121572957)

[Future Scope 18](#_Toc121572958)

[Acknowledgements 19](#_Toc121572959)

[References 19](#_Toc121572960)

## Introduction

A coin is a small, flat, (usually, depending on the country or value) round piece of metal or plastic used primarily as a medium of exchange or legal tender. They are standardized in weight and produced in large quantities at a mint in order to facilitate trade. They are most often issued by a government. Coins often have images, numerals, or text on them. Obverse and its opposite, reverse, refer to the two flat faces of coins and medals. In this usage, obverse means the front face of the object and reverse means the back face. The obverse of a coin is commonly called heads because it often depicts the head of a prominent person, and the reverse tails. Coins are usually made of metal or an alloy, or sometimes of man-made materials. They are usually disc shaped. Coins made of valuable metal are stored in large quantities as bullion coins.

## Problem statement

We have 1444 images of coins from various countries. Our aim here is to find out the number of coins found in each of the images. We will have to create a model that is able to read the images and figure out the shape and number of coins. We also have a csv file which contains the count of coins in each image, with which we can train and verify our model.

This dataset contains the coin images of various currencies, as listed below:

1. US coins
2. Chinese coins
3. Yen coins
4. Euro coins
5. Indian Rupee coins
6. Peso coins

## Datasets

Our dataset can be found [here](https://www.kaggle.com/datasets/balabaskar/count-coins-image-dataset?select=coins_count_values.csv).

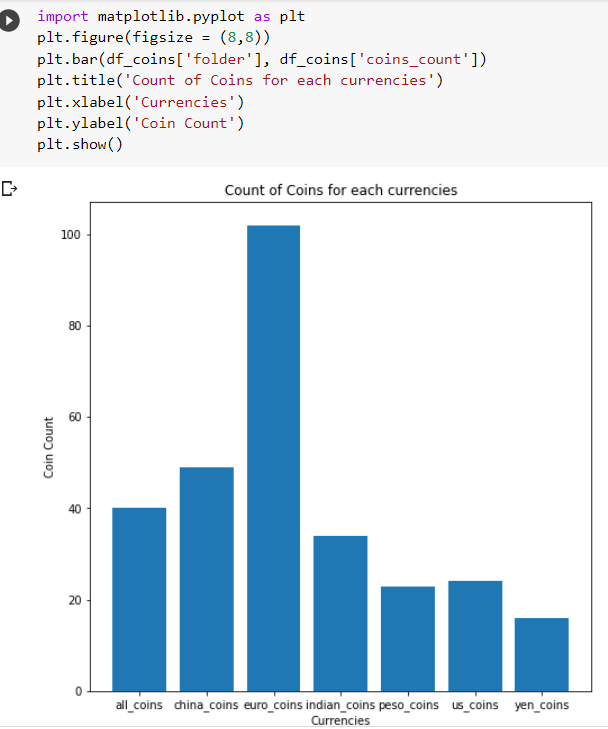
1. We have a csv file that contains
   1. Folder Name
   2. Image Name
   3. Coin Counts
2. 1440 images belonging to 7 currencies but different denominations.
   1. All coins – 215 images
   2. China coins – 210 images
   3. Euro coins – 231 images
   4. Indian coins – 220 images
   5. Peso coins (Mexican) – 184 images
   6. US coins – 173 images
   7. Yen coins (Japan) – 211 images

## Exploratory Data Analysis

We are trying to classify our data based on the coin counts. In the below code we are reading in the csv file and are looking at the unique coin count values.

Table

Description automatically generated



Table

Description automatically generated

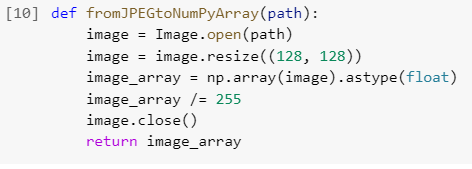
The data transformation is different for the both the models that we have tried, hence we will discuss it separately with each model.

# CNN Model

## Model 1: Base Model

### **Data Transformation**

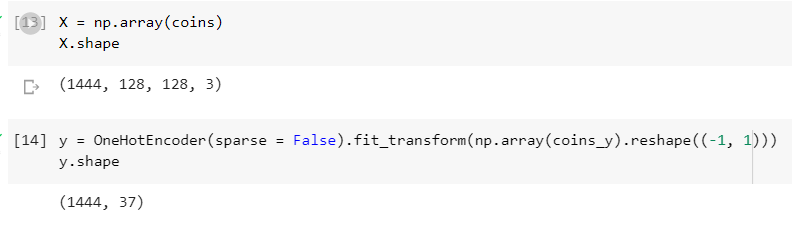
Each image is transformed into 128x128 pixels, and we are converting the jpeg image into numpy array using Keras library.



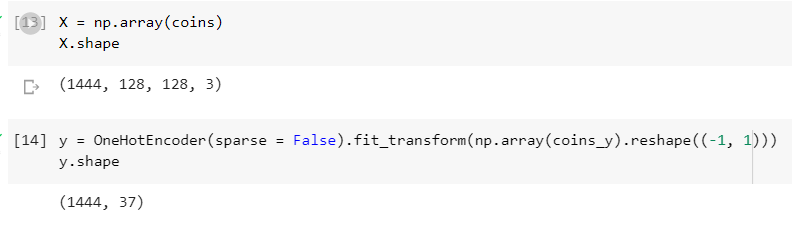
We then use the above fromJPEGtoNumPyArray function to change all the images in the coins\_images folder.



Now X contains the coin images. X.shape has (1444,128,128,3). Meaning we have 1444 images which are 128 x 128 pixels.

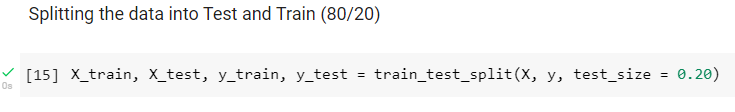


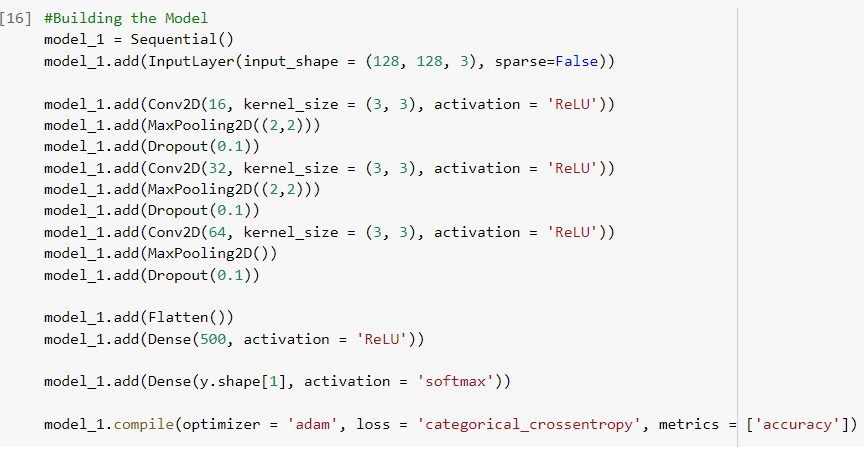
We are applying **OneHotEncoder** on y to convert the different coin count into categorical variables which will help in better prediction. y.shape shows (1444,37) which means there are total 1444 images with 37 different coin counts.



### Model Building

We have taken 80/20 data for test-train split.





We decided to implement convolution neural network CNN for our images. CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data.

**Input layer** – 128 pixels by 128 pixels and 3(colored images)

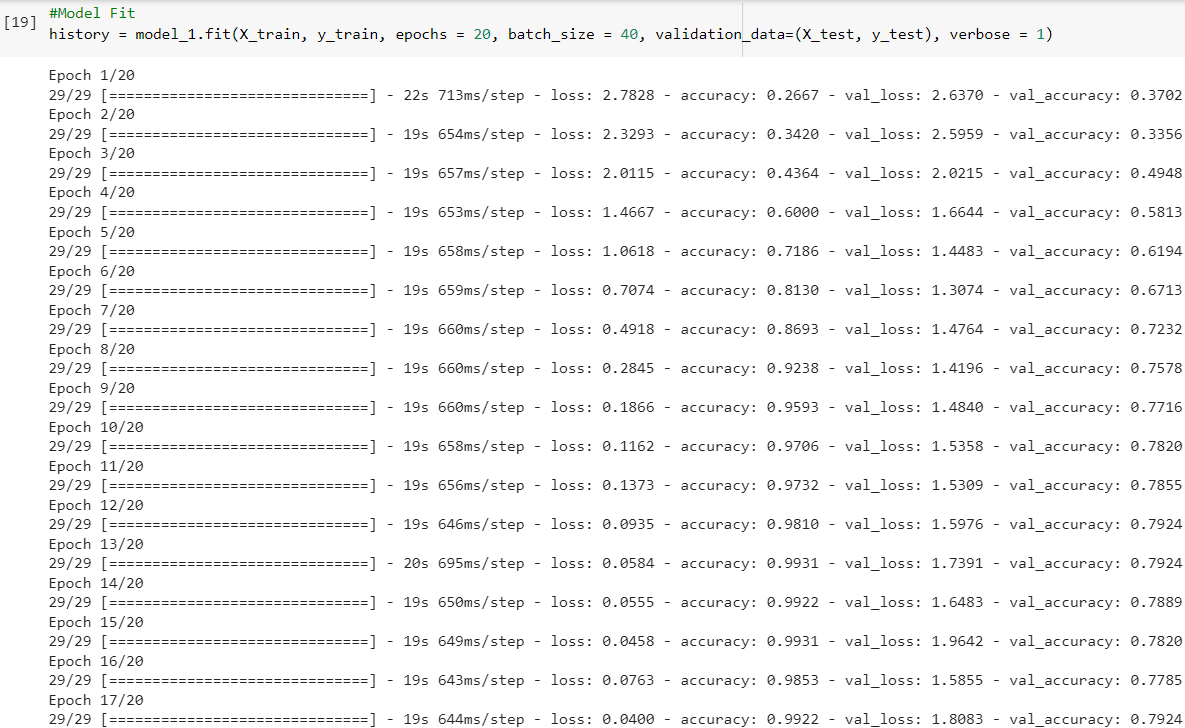
**Convolution2D** – Our model contains 3 convolution layers first with 16 neurons, second with 32 neurons, and third with 64 neurons.

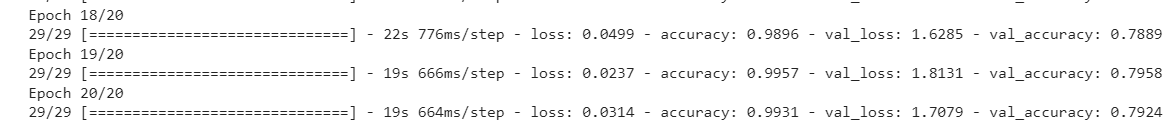
**Activation function** – We are using ReLU activation function for all convolutional layers. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. For the output layer, we have used Softmax activation function.

**Dropout** – we have used 0.1 as our dropout value.

### Model Application

We are fitting our base model based on 20 epochs.



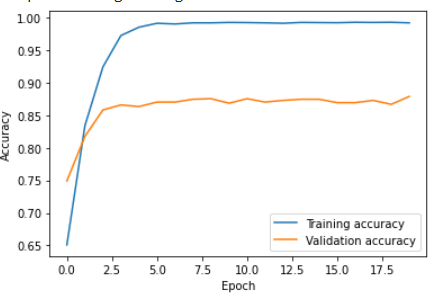


**Accuracy for Training and Test Set:**

**Training accuracy = 99%**

**Test accuracy = 79%**

For training set, model predicted about 99% of the images correctly. For test set, model predicted about 79% of the images correctly.

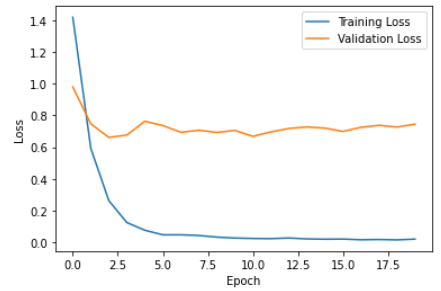


**Loss for Training and Test Set:**

**Training loss = 3%**

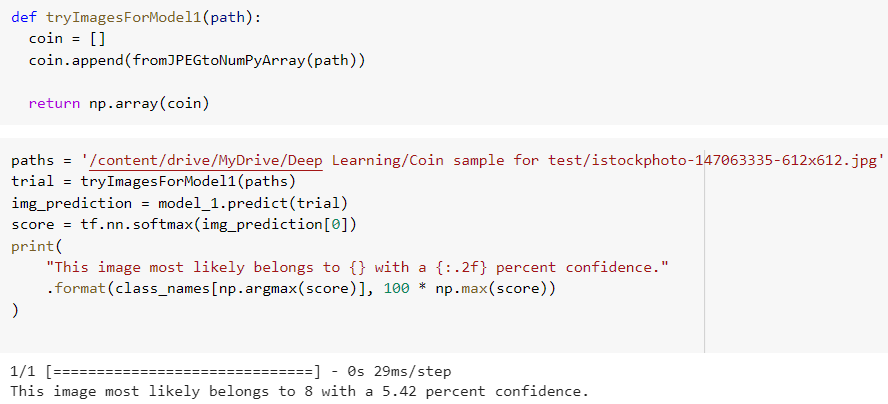
**Test loss = 170%**

The loss is 3% for the training set which means that the prediction error is very less. The loss is 170% for the test set which means that the prediction error is much higher than expected. The result shows that there are very high chances of overfitting of the data.



### Model Predictions

We predicted google coin images, and our base model predicted the coin count really well. We can clearly see 8 US coins in the image and our model predicted 8 with 5.42 confidence.

## Model 2: Augmented Model

### **Data Transformation**

Each image is being transformed in the “rotateImage” function. We are resizing the images to 128 x 128 pixels. We then rotate the image and normalize the image array to ensure that each image has a similar data distribution.

Graphical user interface, text, application

Description automatically generated

We then use the above function to change all the images in the coins\_images folder. For better prediction, we are increase the number of images to train our model we are adding degree rotation to each image between 0 to 360 with steps of 90. Now X.shape has (5776,128,128,3). Meaning we have 5776 images which are 128 x 128 pixels.

Text

Description automatically generated

### Model Building

We have taken 80/20 data for test-train split.





**Input layer** – 128 pixels by 128 pixels and 3(color images)

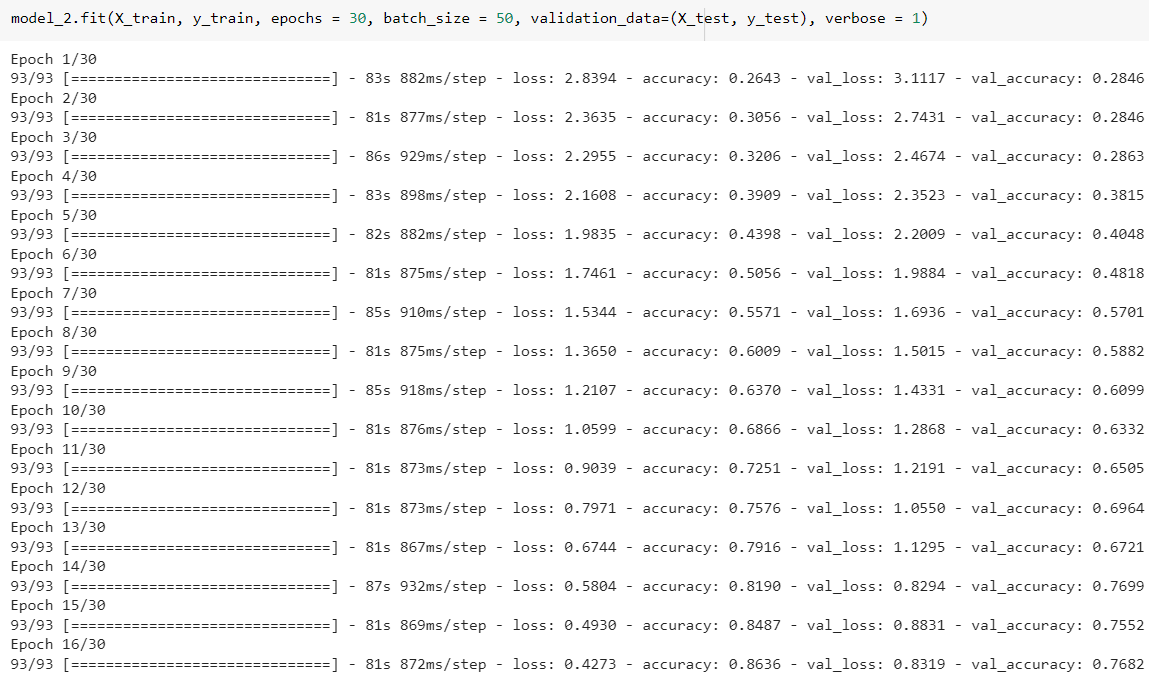
**Convolution2D** – Our model contains 4 convolution layers first with 16 neurons, second with 32 neurons, third with 64 neurons, and fourth with 128 neurons.

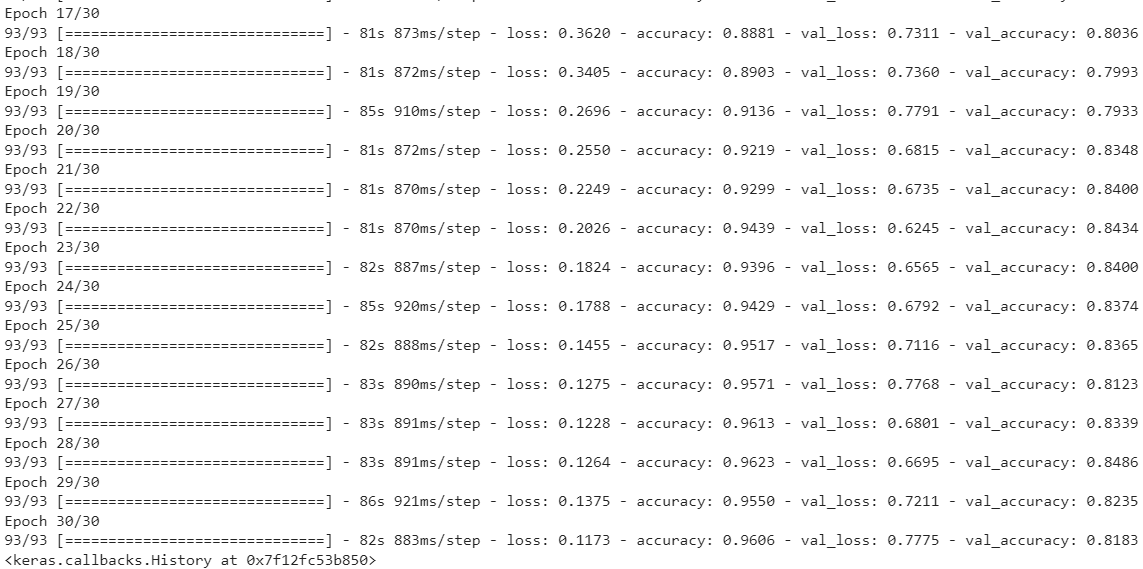
**Activation function** – We are using ReLU activation function for all convolutional layers. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. For the output layer, we have used Softmax activation function.

**Dropout** – we have increased the dropout to 0.5.

### Model Application

We are fitting the new model based on 30 epochs.



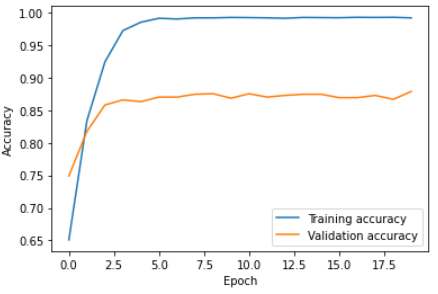


**Accuracy for Training and Test Set:**

**Training accuracy = 96%**

**Test accuracy = 82%**

For training set, model predicted about 96% of the images correctly. For test set, model predicted about 82% of the images correctly which is high as compared to base model.

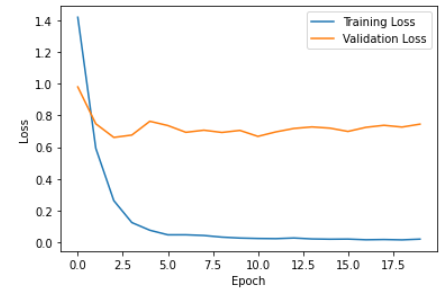


**Loss for Training and Test Set:**

**Training loss = 11%**

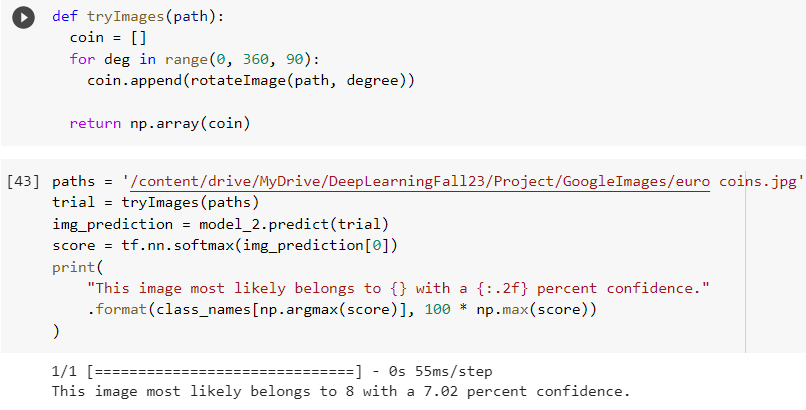
**Test loss = 78%**

The loss is 11% for the training set which means that the prediction error is very less. The loss is 78% for the test set which means that the prediction error is higher than expected and test loss for this model is lower compared to the base model. The new result shows also shows the chances of overfitting but very less as compared to base model.



### Model Predictions

We predicted google coin images, and our augmented model predicted the coin count really well. We can clearly see 8-euro coins in the image and our model predicted 8 with 7.02 confidence.

### Challenges

1. Even after adding more convolution layers to the neural network, it still shows chances of overfitting of the data.
2. The test accuracy and test loss improved only marginally after changing different parameters.
3. We tried increasing the dropout to 50% but that had a negative impact on our training and test accuracy and losses.

## Conclusion

|  |  |  |
| --- | --- | --- |
|  | **Base Model** | **Augmented Model** |
| **Training Accuracy** | **99%** | **96%** |
| **Test Accuracy** | **79%** | **82%** |
| **Training Loss** | **3%** | **12%** |
| **Test Loss** | **170%** | **78%** |

We have tried data augmentation techniques and also increased the dropout percentage and tried different loss functions for reducing the overfitting but both the models still show overfitting. **Augmented Model** is considered to be better model because it has better test accuracy and shows less Test Loss comparative to the base model.

# Fast AI Model

Fast.ai is a deep learning library built on top of Pytorch, one of the most popular deep learning frameworks. Fast.ai uses advanced methods and approaches in deep learning to generate state-of-the-art results. This approach enables us to train more accurate models, more quickly, with less data and in less time. Since the number of images in our original data set is only 1444 this method will help us reach better predicting capability.

### Data Transformation

We are creating a new data frame which has our image paths and the exact coin count in each image. We initially open the csv file and then map the image name to the absolute paths.

Graphical user interface, text, application, email

Description automatically generated

### Model Building

We are transforming our dataset into a DataBlock. DataBlock and DataLoader are Python Classes in the fastai library for data processing. They provide many useful methods to facilitate the handling of data.

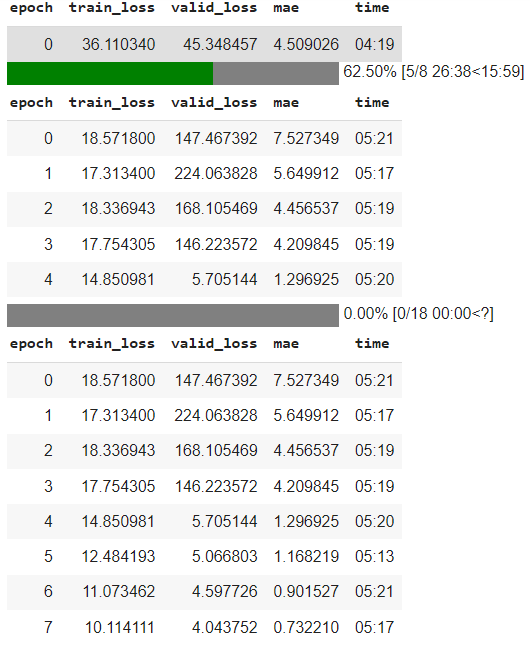
Text

Description automatically generated

We are using cnn\_learner to learn from our data. The cnn\_learner factory method helps you to automatically get a pretrained model from a given architecture with a custom head that is suitable for your data. ‘fine\_tune’ – we are specifying the number of epochs here.

our learner/model using the ‘resnet18’ architecture that is a pre-trained model, with accuracy rate as our metrics. We trained our model over 8 epochs and our model was able to get Mean absolute error of 0.73 in its 8th epoch compared to 4.50 in the first epoch. This is a good result. Validation loss is 4.04 very less than the training loss 10.11 which indicates it is a good fit model.





### Model Application

Our Model shows coin counts closer to the actual coin counts:



### Model Dissemination

Now let’s try our model to see if we are actually able to show the count of coins in images. For testing we have taken a random image from our images folder and will try to predict the number of coins in the image. We are using the ‘learn’ variable to predict this.

### Model Predictions

**Example1**:

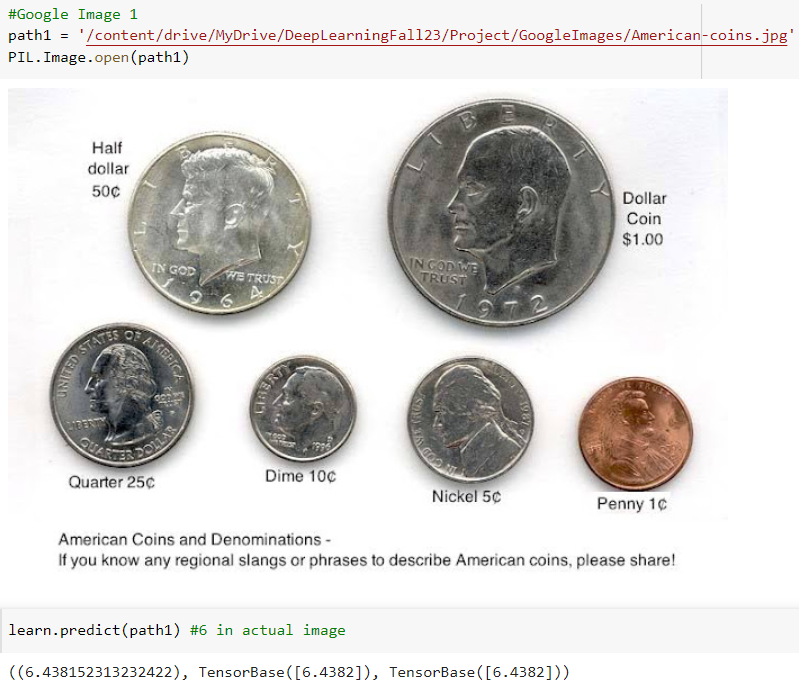
Graphical user interface, text, application

Description automatically generated

As seen in the above image we have 2 coins, and our model predicted 1.958 which is very close to the actual count of coins.

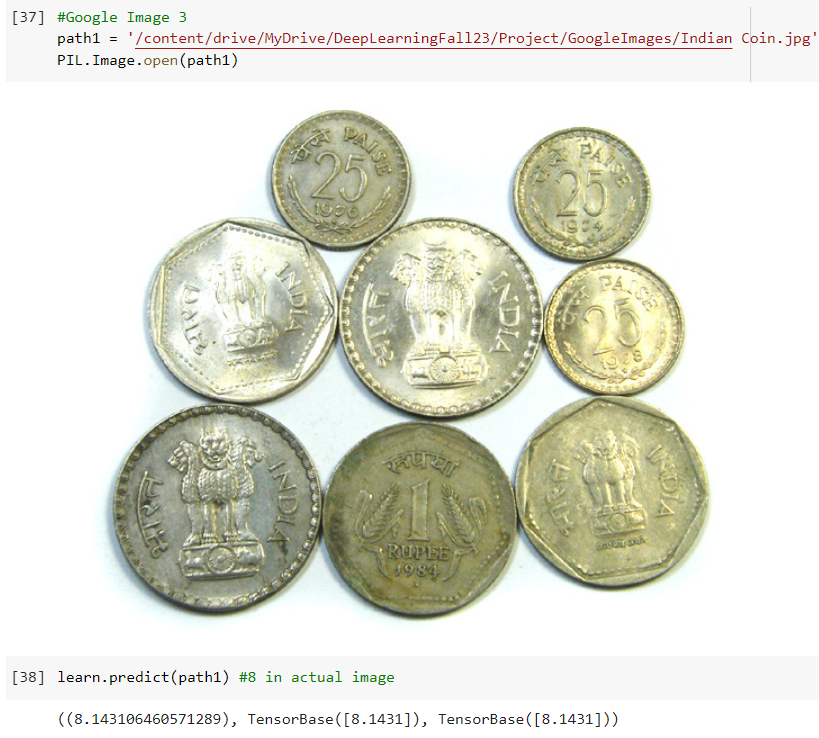
**Example2:** Now let’s take some examples from google images

The actual count was 6 but our model has predicted 5.77, which is very close.



**Example3:**

The actual count was 8 but our model has predicted 8.14, which is very close.



### Challenges

* All coins are not round shaped.
* The background and color of the coins is not the same.
* If we have overlapping coins then the count is slightly lower than the actual count.
* If we have many coins in a single image, then also the predicted count tends to be lesser than the actual count.

## Conclusion

|  |  |  |
| --- | --- | --- |
|  | **FastAI Model** | **Augmented Model** |
| **Training Loss** | **10.11** | **0.12** |
| **Test Loss** | **4.04** | **0.78** |

For FastAI Model, the model predictions were close to the actual coin count. But Training Loss and Test loss for FastAI model is pretty bad compared to the Augmented Model.

We saw that the augmented model had a higher test accuracy of 82% with 78% of test loss.

We used the combination of right hyperparameters such as:

4 convolutional layers with 16, 32, 64, and 128 neurons

Batch size of 50, and

Dropout rate of 0.5

So, **Augmented Model** (Model2) is better in terms of prediction and test loss rate.

## Team Contribution

Shalini Sarathy Komandur Chakravarthy – 40%

Vasudha Gulati – 40%

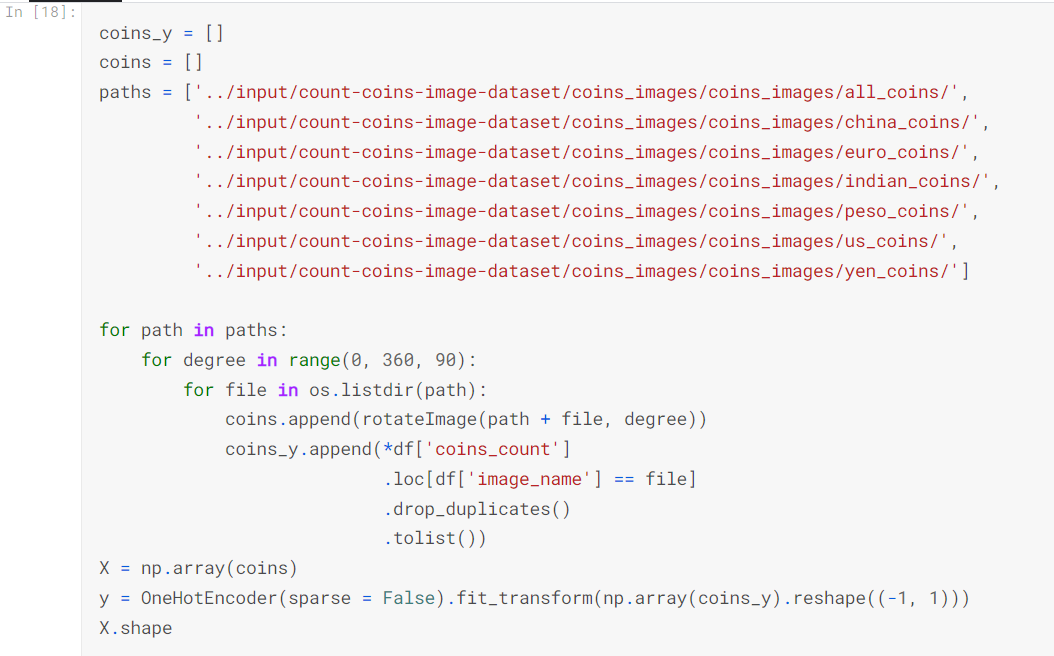
Priya Sankar – 20%

## Future Scope

* Building a model for images with different currencies which identify the type of currencies in the image and also give us count of coins.
* Sum of currency coins by reading text on image. We tried implementing it using **pytesseract** but it couldn’t work for us because we have circular text in some coin images.

## Acknowledgements

<https://www.kaggle.com/datasets/balabaskar/count-coins-image-dataset/code>

This link had helped in the data cleansing process. 

We then went further and build our CNN models based on this cleaned up data.

## References

* [**https://www.kaggle.com/code/jayitabhattacharyya/coins-classification**](https://www.kaggle.com/code/jayitabhattacharyya/coins-classification)
* [**https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/**](https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/)
* [**https://algoritmaonline.com/image-classification-cnn/**](https://algoritmaonline.com/image-classification-cnn/)
* [**https://www.freecodecamp.org/news/handling-overfitting-in-deep-learning-models/**](https://www.freecodecamp.org/news/handling-overfitting-in-deep-learning-models/)
* [**https://www.kaggle.com/code/ferasoughali/crowd-estimation-fastai-v2/notebook**](https://www.kaggle.com/code/ferasoughali/crowd-estimation-fastai-v2/notebook)
* [**https://docs.fast.ai/tutorial.vision.html**](https://docs.fast.ai/tutorial.vision.html)
* [**https://blog.griddynamics.com/how-to-recognize-coins-with-deep-learning-visual-model/**](https://blog.griddynamics.com/how-to-recognize-coins-with-deep-learning-visual-model/)