Deep Learning With PyTorch: Project Final Report

## Diwanshu Jindal

# Abstract

Creation and working of a Coin Classifier using PyTorch and pretrained model EfficientNet B3

# Introduction

In this project, I have developed a machine learning model designed to classify coins from a diverse global dataset. The dataset, sourced from Kaggle, contains images of coins from 70 different countries, across 211 distinct classes. These classes vary in characteristics such as size, shape, color, and design, reflecting the rich diversity of world currencies.

The primary challenge of this project lies in the varying formats of the coin images. Some images capture only the front side of the coin, while others show the back side, and in certain cases, the images are not clear which represent a clear case in the real world. However, the model is able to accurately identify the class of each coin based on the features.

Using data augmentation and pretrained model, I was able to create a model that is able to correctly predict the input image out of the 211 possible outcomes more than 70% of the times.

This model can be applied in various fields, such as Sorting Machines, as Coin fraud detector, Coin authenticator for uncommon coins, for coin recycling for coins made of recyclable materials. One can make small changes to the model and use in various other fields if necessary.ba

.

# Background

As the process of training a model from the beginning and have accurate results is quite expensive, takes a lot of time and demands high number of training dataset which is a problem for the dataset that I had so I choose to use a pretrained model to work as a base model and tweak its parameters so it will be more suitable for the current dataset.

For this project, I utilized **EfficientNet B3**, a model known for its efficiency and high performance in image classification tasks. EfficientNet is based on a novel architecture that balances depth, width, and resolution, suited for image recognition. By using a pre-trained model, the system benefits from the extensive feature learning achieved from large-scale datasets such as ImageNet, making it well-suited for transfer learning on a specific domain, in this case, coin classification.

To complete the fine tuning I have added a dense fully connected layer from the 100 neurons that are the output layer of the EfficientNet B3 to 211 neurons that represent the number of unique classes that the training data have. As we are using both the architecture of the model and the calculated parameters so I have frozen the parameters till the last layer and then calculated the last layer using the new training data

# Experiments

I have used different methods to make the model first I tried to build a model from the scratch it wasn’t able to learn the different features of the images and the model was only able to give accuracy of 30% on the testing data. This model was based on CNN and had 5 hidden layers.

Then I tried to add more layers, but the training time kept increasing while led me to use a pretrained model EfficientNet B3 is a suitable pre-model for the coin-based data as it is trained on the similar images and the parameters can be used as it is.

While using the pretrained model I first try to train the model on the train data without any data augmentation which resulted into accuracy of 58% then push me to add augmented images into the training set and then train the model on the balanced dataset which gave the 71% accuracy on the testing data.

The model could have been better but due to the time restrains and large training time I could not train the model on a highly augmented data where each class would have around 100 images in the training set which is 72 images more than the current image count per class.

# Results/Discussion

The performance of the Coin Classifier model was evaluated using several approaches, starting with a basic CNN-based model and progressing to the final implementation using a pretrained EfficientNet B3 model. The results varied significantly based on the different stages of the experimentation.

Initially, I trained the model without any data augmentation, which resulted in a test accuracy of 58%. Although this was a significant improvement over the previous models, the performance was still not optimal. The lack of data augmentation led to some overfitting, as the model was unable to generalize well to unseen images. To enhance the model's generalization capability, I introduced data augmentation, which involved rotating, flipping, and scaling the coin images. This helped to increase the diversity of the training data and reduce overfitting. After incorporating the augmented data, the model's accuracy improved to 71% on the test set. This performance demonstrated that the model was able to better handle the variability in the images, improving its ability to classify coins from different countries and with varying sizes, shapes, and designs.

# Future Work

# While the model performed reasonably well with a 71% accuracy, there are several factors that could further enhance its performance. One significant limitation was the relatively small size of the training dataset for some classes. Some coin categories had fewer images than others, which made it harder for the model to learn distinctive features for those classes. Ideally, each class should have around 100 images to improve the model's robustness and accuracy. However, due to time constraints and the large amount of training time required for each iteration, this level of data augmentation could not be fully realized within the scope of this project.

# References

1. Kaggle Coin Recognin Model, https://www.kaggle.com/code/zaranoor9907/coin-recognition-model

PTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg

[2] Torch :An Open Source Deep Learning Platform, https://pytorch.org/vision/stable/index.html