CS550 Project Proposal – Turkish Lira Classification

Cengiz Emre Dedeağaç - Doğa Yılmaz

January 8, 2022

1 Introduction

Even after the pandemic came out, cash is still an essential part of transactions in today's daily life. People exchange various types of cash when shopping depending on the price and how much money they have in their pocket. It can be challenging for visually impaired people to distinguish easily and use the correct amount of cash trustfully during these transactions. To help people with these kinds of struggles, we want to create a tool where people can scan their banknotes and learn how much it is.

In this project, we prepared an application that satisfies the groundwork for such service. For the purposes of this course, we utilized cloud services in our project. Our final product can recognize all Turkish banknotes of different values and differentiate a particular set of fakes.

2 Utilized Tools and Services

2.1 Amazon Rekognition

Amazon Rekognition is a service for training machine learning models for image and video processing. [1] It accepts labeled images and trains a model to categorize any input image. It is effortless to use but might need some readjustments on the training data for better performance. This service acts as the core of our decision mechanism.

2.2 AWS S3

S3 is a cloud storage service provided by Amazon. It can easily integrate with other services provided by Amazon. In this project, we use it to store training/test data.

2.3 Python

Python is one of the most popular programmings languages of recent years. It has a wide range of libraries and easy to type syntax for faster prototyping.

2.3.1 Django

Django is a prevalent Python framework that allows users to quickly create scalable websites, which was a good choice because it is easy to test and host it to be used by other people.

3 Solution

In this section, we will elaborate on the datasets we used, the AWS Rekognition Project, our Django client, and the pricing of our solution.

3.1 Dataset

Throughout this project, we created three different datasets by using several methods. In this section, we will elaborate on each dataset in detail.

3.1.1 Dataset 1: Web Scraped Images Combined with Self Taken Images

Our initial approach was collecting Turkish Lira images from the web and using them as our dataset. However, when we searched for high-quality images, we could not find a diverse range of images. Some images are available, but most are taken from the same angle or have poor quality. Thus, to make our dataset better, we took images of each TL banknote and combined our images with web scraped images. The sample count for each class can be found in the table below.

Class Name	Sample Count
5 TL	28
10 TL	26
20 TL	23
50 TL	26
100 TL	32
200 TL	27

As seen from the table above, we tried to pick sample counts close to each other. Also, for a traditional ML model, sample counts are extremely low. However, AWS Rekognition does not require large datasets to train a sufficient performance model using transfer learning.

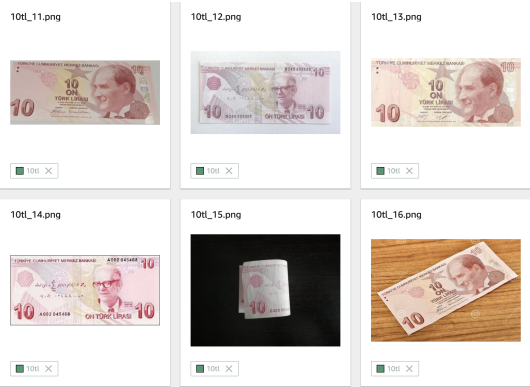

Figure 1: Sample images from dataset 1

3.1.2 Dataset 2: Self Taken Images

Like the first dataset, we have created another dataset by only using the images taken by us. We aimed to decrease the number of very similar data samples (as shown in figure 1) and increase the image quality. Also, we added an extra class for detecting fake banknotes, as shown in figure 2. The additional class adds functionality to our model by supplying data to the AWS Rekognition model for fake banknote instances. The sample count for each class in dataset two can be found in the table below.

Class Name	Sample Count
5 TL	22
10 TL	19
20 TL	20
50 TL	20
100 TL	21
200 TL	25
Fake	37

3.1.3 Dataset 3: Self Taken Images with Bounding Box Data

AWS Rekognition Data Management tool has a feature which we can draw bounding boxes around the object we want to recognize. Thus we used this tool to draw and add bounding box information to our second dataset. One sample with a bounding box can be seen in figure 4. By providing extra information about the whereabouts of the banknote in the given image, we aim to improve the performance of our AWS Rekognition model in the real world.

Figure 2: Original 20 TL banknote vs Fake 20 TL Banknote

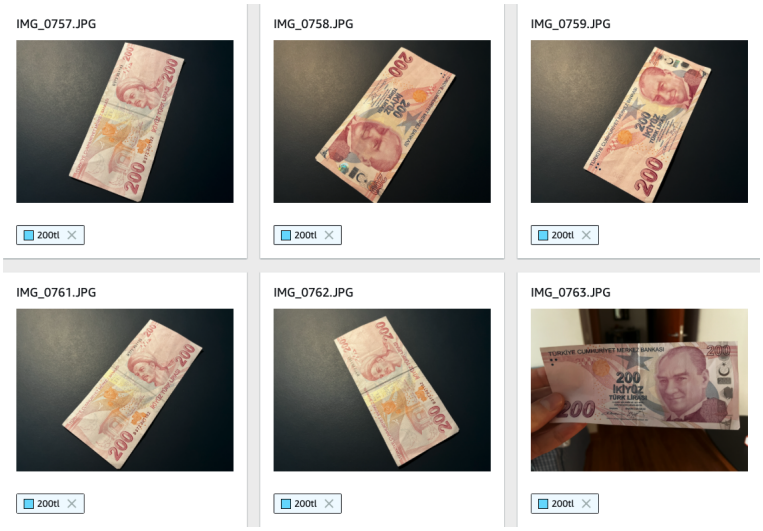

Figure 3: Sample images from dataset 2

Figure 4: Image sample from dataset 3

3.1.4 Managing Datasets

Throughout this project, we have created a lot of experimental datasets. At some point, dataset management becomes an issue. That is due to the AWS dataset management system. When we started this project in the mid-semester, the system did not allow us to delete any dataset from the Web interface. Even though we delete the actual data in the s3 bucket, the dataset remains visible in the AWS console. In order to solve this issue, we headed to the AWS documentation; however, we could not find any solution there. Following that, we have posted a question to StackOverflow regarding this issue.

Around the late semester, an update was made by AWS, which changed the management of the datasets. Currently, datasets are managed under AWS Rekognition projects, and they can be

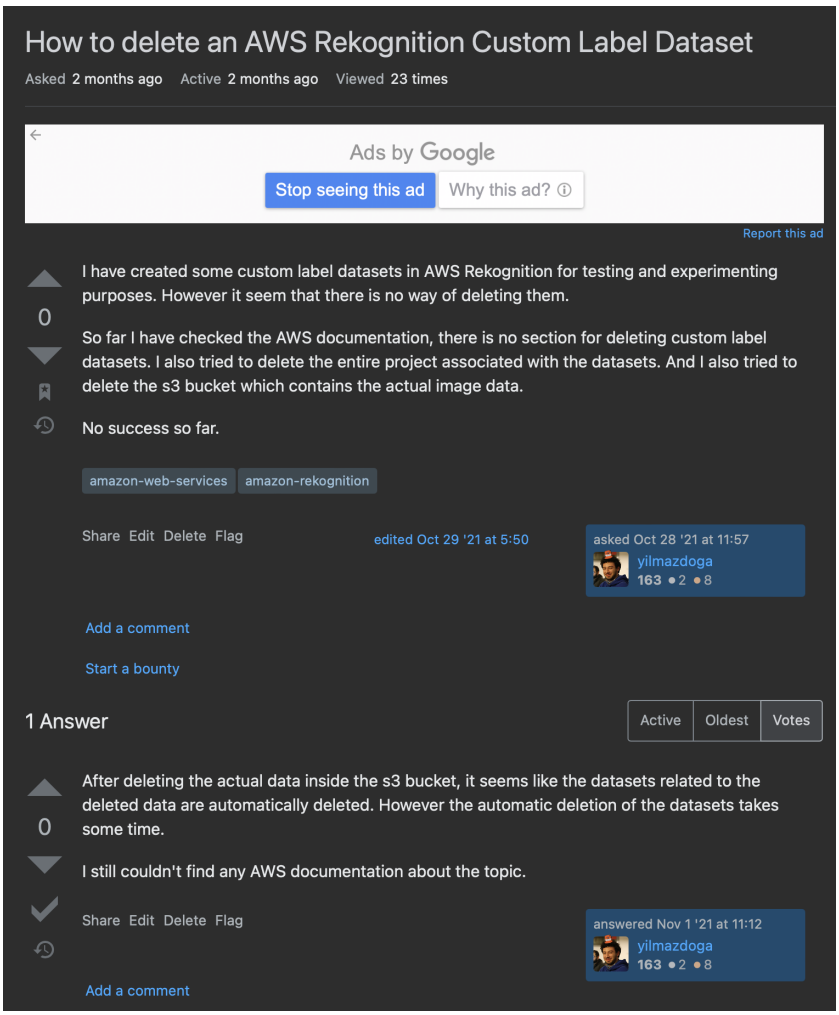

Figure 5: Our StackOverflow question.

deleted or modified from the AWS console. We think that this is an excellent example of how cloud products change according to users' needs, and we thought that the situation is worth reporting.

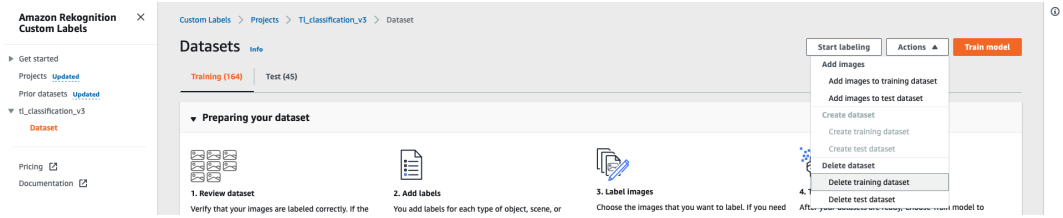

Figure 6: Updated AWS Rekognition console

3.2 AWS Rekognition Project and Training Progress

We have trained three different models with different datasets throughout this project, as seen in figure 7. The training process takes around 1 hour.

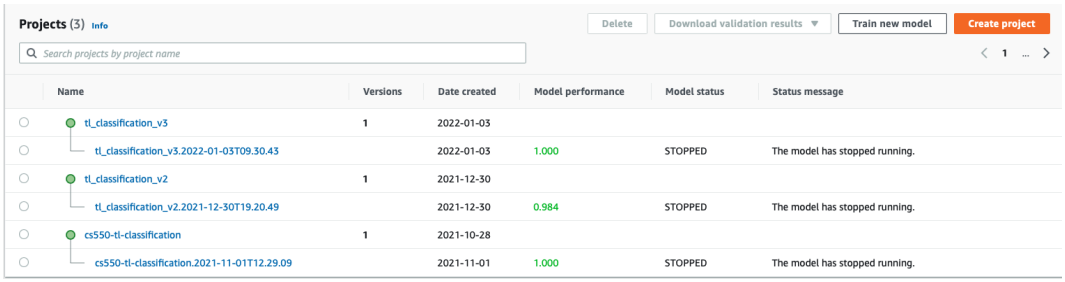

Figure 7: AWS Rekognition console screenshot of our models

With AWS Rekognition, we can train sophisticated models with the push of a button. The system uses transfer learning in order to train models with a minimal number of data.

3.3 Django Client

In order to use our trained models, we need software that uses a camera to take the image of a TL banknote and sends the captured image to the AWS Rekognition model. We used the Django framework and made a straightforward web app that opens up the system's webcam and captures images every 2 seconds to achieve this functionality. Then we used Boto3 (Python SDK for AWS) to send our captured images to our AWS Rekognition model. Once the response arrives at our client, we store it locally. If two consecutive frames are classified with the same label, the client shows the class label at the top left of the screen. Also, if the Rekognition service is offline, the client warns the user. A screenshot of our Django Client can be found in figure 8. Also, in order to show the real-world performance, we prepared a demo video in which we use the Django client with our AWS Rekognition model.

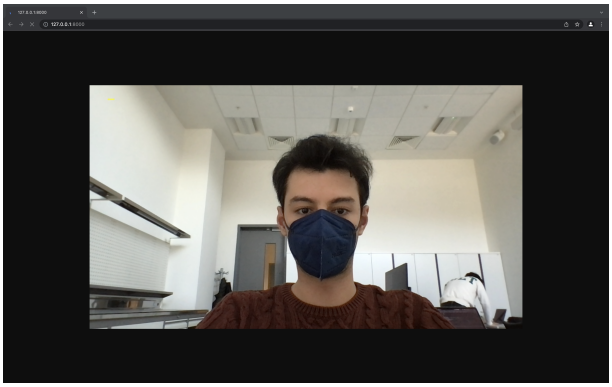

Figure 8: Screenshot of our Django Client

3.4 Pricing of Our Solution

When using cloud services, pricing is also an important matter. This section will provide information about the prices of each building block of our solution. The unit prices of our building blocks are listed below.

Service Name	Price
AWS S3 Storage	$0.023/GB
AWS S3 Data Read	$0.0004/1,000 request
AWS S3 Data Write	$0.005/1,000 request
AWS Rekognition Custom Label Training	$1.00/hr
AWS Rekognition Custom Label Inference	$4.00/hr

The table above shows that data storage, read and write costs are manageable even if we have an extensive dataset that we do not need with AWS Rekognition's transfer learning feature. However, the Rekognition service itself is an expensive one. It costs 1 USD per hour during the model training and 4 USD per hour while our prediction instance is up. However, the AWS Free tier provides a limited free-of-charge usage each month. The free tier provides 5GB of storage, 20000 read, and 2000 write requests for S3 service. It also provides 10 hours of training per month and 4 hours of inference time per month for the first three months. Using the AWS Free tier, we managed to stay at zero cost throughout the semester.

4 Results

We connected to the Rekognition service and detected different banknotes and fake versions using our trained model. 2 state confirmation makes sure that the detected object is not mistaken and eliminates a good portion of false positives. The demo video is accessible via the following link: https://youtu.be/JPPh7ovYRg4. Also, our Django Client implementation is accessible via the GitHub link: https://github.com/CengizEmreD/cs550project.

5 Conclusion and Future Work

Throughout this project, we evaluated different services. Even after deciding on Rekognition, we had to choose from different paths to connect our client and Rekognition service. We tried different languages, libraries, and platforms to base our product on. Overall, this was a perfect opportunity to see and experiment with different components and stages of developing a product with cloud services.

To create our minimum viable product, we had to choose some methods more suited for prototyping than creating a comprehensive product. After getting to this point, we can take a few paths to improve our product. We can change the client to an application instead of a website to provide better support for mobile phones, which we think will be the main platform used. We can add a voice-over option to make it more complete. Finally, we can improve the connection method to be more robust for mass usage by using different libraries and other services provided by Amazon.

References

[1] Aws rekognition faq. https://aws.amazon.com/rekognition/faqs/. Accessed: 2021-11-06.