CARD GRADER REPORT

A Machine Learning system for the grading of trading cards

Abstract

By
Thomas Muldoon
Ed Musi
Wei Ong
Rohit Rokka
Jabin Shrestha

Title Page

Table of Contents

Contents

Acknowledgment

The information presented in the final report is of our own work and are referenced using credible resources, all of which are stated in the bibliography. We would like to also acknowledge Mel Razmjoo and Patrick Davis-Desmond for providing us with insightful comments on our project's draft during the first presentation which greatly aided us in completing it. We would also like to thank the TAFE library for providing us with access to useful information on the technologies we used in this project.

Abstract

Collectible card games are a growing industry with cards being sold for large amounts of money with a sports trading card depicting Honus Wagner being sold for $6.606 million US dollars in 2021 (Hajducky, 2021). In order to ease the selling of cards sellers and buyers have turned to the use of grading agencies that provide a metric for the physical condition of the card.

These grading agencies cost money to be used and are reliant on human graders who can be biased or and take time to grade a card. This has created the need for a system that can be made more objective, and this can be achieved with machine learning.

We attempted to solve this problem using a machine learning system and had some success, but we were not totally successful.

Team

Thomas Muldoon – Project Manager, App, Backend

Ed Musi – Machine Learning System

Wei Ong – Machine Learning, Subject Matter Expert

Rohit Rokka – Website

Jabin Shrestha - Website

Introduction

This report details our research into the feasibility of creating a machine learning system that can be used to grade cards in minutes for users as opposed to the days and weeks it currently takes. The current state of card grading is that grading cards is time-consuming, costly and can be inaccurate because it is reliant on human graders. we believe that through the use of machine learning we can create a system that is far faster and more objective.

The system consisted of several components. To function as a frontend, we created a website and for mobile users we created an Android app, to facilitate communication between the frontend and the machine learning system that is used to grade the cards. The backend comprised of the following: A machine learning model, which is trained for object detection, using a labelled dataset of pre-graded Pokémon cards with apparent imperfections.

*PLEASE ADD ANY OTHER COMPONENTS THAT WERE MADE IN THE BACKEND.

Problem

The problem that we are attempting to solve as a part of this research project is the problem of grading cards being subjective in that a human is needed to grade the cards, costly due to the prices these companies charge and time consuming due to the need to mail the cards to the company. The need to mail the cards to a grading company also creates risk that the cards may be lost or stolen in transit.

Research Questions

The question we were answering for this project was whether it is possible to create an online system using machine learning and the cloud to find the grade of a card without the need of a human grader.

Aims

Our aim for this project was to create a cloud-based machine learning system that will find the grade of a card that a user could use on their phone through an app or on their computer through a web browser

Hypotheses

Before commencing this project, we hypothesised that it was possible for a machine learning system to find the grade of a card.

Literature Review

Prior to starting this project, we conducted a literature review. Such reviews are crucial in research because they allow us to

Grading Experiments

The main grading methods involve human graders, machines and hybrid version which combines the two.

Richard A. Bassett writes about experiments that were conducted to determine the effectiveness of grading when carried out by machines, humans, and hybrid systems.

With the machine grading experiment, a systems effectiveness is determined through its capabilities on performing visual recognition of the collectibles. The results were then compared with human graders to compare their differences.

Some of the problems encountered while carrying out the experiment included:

- Cannot perform grading on subjective features.
- Some counterfeits may not be detected.

In the human grading experiment, 10 human graders received 25 different images over the web and were asked to provide their grades on each. All these individuals had a minimum of three years of experience in collectible grading. The results of each grader were compared to help determine an average grade.

The main challenges experienced are:

- Lack of accuracy and consistency.
- Wide grade variations by the grader despite their massive experience.

A third experiment involving both graders and machine was conducted and provided a platform to analyse both critical and subjective features. It combines human visual recognition with computers expansive memories and produced more acceptable results with less variance.

This hybrid method enables graders to inspect machine grade outcomes while applying subjective interpretations such as colour, planchet quality, toning, defects, and strike quality to arrive at a final grade.

This informs the purpose of our project in using machine learning algorithms to provide alternative methods of grading. By doing this, collectors can start the grading of the cards using new algorithms then only proceed to human graders when the scaling turns out to be higher. This in turn saves them costs of only using a human grader and ending up with a low-grade collectible card.

Technologies

For every project it is essential to carefully examine the technology that will be used to accomplish it to ensure that the technologies are appropriate.

AWS Rekognition

Amazon Web Services provides a service called AWS ‘Rekognition’ for use in image recognition which my group will be using, partly due to our previous familiarity with Amazon Web Services. Evaluating the accuracy will be crucial for this project.

AWS ‘Rekognition’ has seen great use in facial recognition especially in the recognition of criminals in instances where they try to hide their identities or for their identification using CCTV cameras.

S Jung, J An, et al evaluated four facial recognition tools for their accuracy in identifying Gender, Age and Race of people in images in the paper (Jung, An, Kwak, Salminen, & Jansen, 2018). To test these tools, they used a pre-existing dataset sourced from IMDB that was created by a prior study and consisted of 52,478 images of actors. They also used a dataset of 100 pictures of celebrities as well as a manually created dataset of Twitter profile pictures. Their findings were that in the case of detecting the presence of a face Amazon Rekognition was the second most accurate with IBM Blumix Visual Recognition being the most accurate. When all datasets were used Rekognition was never the most accurate of all the systems, but it was only in the case of the dataset for detecting gender from Twitter profile pictures, that it was not second in accuracy; in that case it was third. In the testing of detecting the subject's gender, Rekognition was also a close second or third for all datasets with Microsoft Azure Face API being the most accurate. In these tests, despite being second or third in accuracy Rekognition was still a very close second or third with the difference between it and first place being less than 8 percent except in the case of deriving age, where it had was far less successful and was correct in only 30 percent of cases (whereas Microsoft's tool had an accuracy of 45 percent}. The only tool used for race detection was Face++ and thus the results are not relevant.

There has also been recent controversy regarding the accuracy of AWS Rekognition facial recognition in identification of criminals. The American Civil Liberties Union (ACLU) (Snow, 2018) tested the system and found that it erroneously identified 28 members of the US Congress as people who had been arrested for a crime. The ACLU also released a user manual (Washington County Sheriff, 2018) used by the Washington County Sheriff for the use of their AWS Rekognition based facial recognition system for identifying prisoners which (on page 40), alerted users that even a 93.53% match should not be trusted and used an example wherein a Caucasian man was identified as the African American former footballer and actor, O J Simpson.

This indicates that while Rekognition, performs at least competitively with its competitors its accuracy still leaves much to be desired and as a result we must take great care to ensure that our dataset is of an appropriate size and that our reward function is well designed. The ACLU report also shows that while our project is of far lower stakes than a system for identifying criminals, we must take great care to ensure that our system is accurate and cannot simply assume that it would be useful without extensive testing. There is also the possibility that we may need to use a different system for our project, but our familiarity with AWS remains a compelling reason to use Rekognition despite its flaws.

Machine Learning

Healthcare:

ML is becoming more ubiquitous today and the demand for such technology is also ever growing. Pivotal industries such as Healthcare requires decision making, logical thinking and critical assessing. If a patient is incorrectly prescribed by any of these factors, undesirable consequences will occur which can be life altering. By implementing AI ML, you are effectively taking this logical step out of the equation which will mitigate the risk for human error (misdiagnoses, wrong prescriptions, etc.) and in result, will create a more accurate and safer diagnoses for patients.

The presented use-case for ML in Healthcare further explores this idea. A research study from the University of Iowa, United States (Abramoff et al., 2016), necessitates and optimizes the use of automation and Image Recognition via Deep Learning to detect early stages of Diabetic Retinopathy (**DR**). By utilizing the Messidor-2 dataset comprised of fundus images of patients (874 of the 1748 images were subjects with diabetes), it was found that when applying it with Deep Learning (**CNN**) and an enhanced algorithm (Implementation of CNN detectors), the percentages for sensitivity,

specificity, negative predictive values were more accurate than algorithms that did not utilize Deep learning. This result was achieved due to the way that the CNN detectors were deployed. Unlike other tests, this method utilized automation and had all CNN-based detectors trained with a sample size ranging from 10,000 to 1,250,000 images which included distinguishable imperfections such as lesions, haemorrhages and other conditions typically found in DR. This suggests that the implementation of automation with ML is warranted in order to improve the accuracy of a DR Screening which will help facilitate early intervention and treatment.

Dataset Testing:

Another relevant research was conducted with the primary objective of testing the accuracy of Image Recognition using several datasets (Alfayez, 2016). Each dataset contained the same set of images (dogs & cats) but with different resolutions (32x32, 64x64,128x128, and 256x256). The goal was to see how many epochs (iterations) that it would take for the image recognition software to run in order to achieve a success rate of 99.25%. All datasets were processed with the same set-up (sample size of 128 images) to make the CNN and the results showed the following:

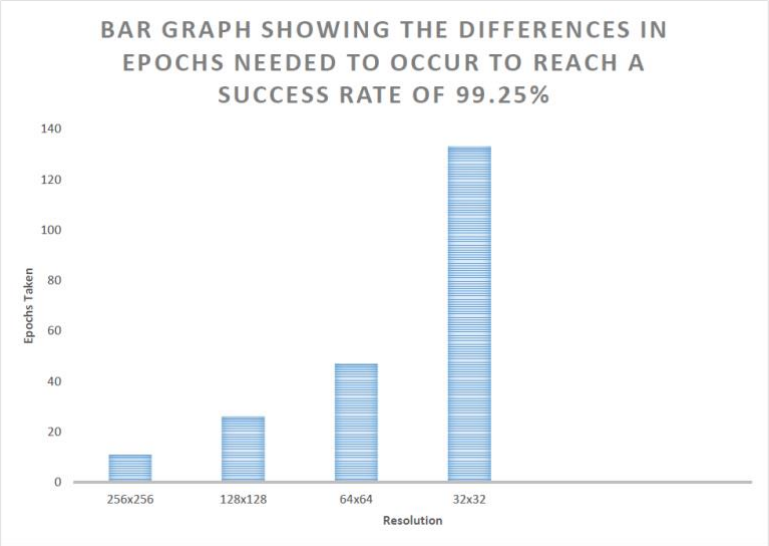


Figure 1. Bar Graph of epochs needed for each dataset

From the graph, it was concluded that images with a pixel size of 256 x 256 deemed to be the quickest with the least epochs needed to reach 99.25%, which was only 8% of the 32 x 32 dataset. This indicates that the dataset with the highest resolution will be the fastest for calibrating the CNN model which in turn would result in a more accurate prediction application.

The presented scholar papers can draw parallels to our project in some respects. As our Project will be dabbling with Machine Learning and Image Recognition, a Convolution Neural Network (used in both research) will be needed to bridge all these elements together. Programs such as TensorFlow and VGG16 are viable solutions that can be adopted and used to create the CNN needed to train our model. A large dataset will also be required. From the scholar research, it was concluded that images with a higher resolution will allow for a more accurate image recognition tool. This idea will be adopted going into our practical implementation of the project. Higher quality images will be prioritized and collected over lower quality ones.

Given the scope of our project (time-frame, resources, competency/knowledge in using the necessary programs, applications and so on), our group is not expected to create such an in-depth CNN model with a dataset of a large proportion like the healthcare case but still are able to create a

working model with a reasonably sized dataset that is able to achieve the primary objective of the project: “to accurately predict the condition of a card by identifying its imperfections.”

Security

The project we are working on will include a web dashboard that will need to be secure for users and, at the current point in the project, it appears most likely that the website will be implemented in AWS. The use of a cloud system presents new challenges for web security because there is no way to physically access the servers and thus any secure connections to the server must be done over the internet.

The book ‘Website Hosting and Migration with Amazon Web Services’ (Nadon, 2017) contains guidance on how to secure an AWS based web app. To ensure that the communications to the Web app are secure it advises the use of a security certificate to enable the use of SSL so that communications from users cannot be listened in on by others and that all their communications to the server remain confidential which will be essential to ensuring that the account system the web app will need remains secure. AWS provides a service called AWS Certificate manager to aid in managing security certificates as well as procuring them.

Securing the loud account is also a concern and Amazon provides tools to ensure that access to the servers rung the web app is secure. Nadon further advises the use of AWS Identity and Access Management which allows other accounts to be given limited access to AWS resources so that other members of the administration team do not have to use the root account and thus the damage an account breach causes can be significantly lessened.

Other tools that will be useful for securing the project include AWS Trusted Advisor and AWS Inspector which both serve to advise the user on how to ensure that all parts of the project are secure. AWS Trusted Advisor analyses accounts and resources and alerts the Administrator whether they are following best practices for security as well as performance which will also be important to the project. AWS Inspector is also regarded as an invaluable security tool that tests EC2 instances (which are the actual servers) for security vulnerabilities and informs the Administrator about how to remediate them which will ease the time-consuming process that patching all software on a server is.

These tools will be invaluable for ensuring that the project remains secure for both users and our group.

Similar Projects

When starting any project, it is vital to examine the work of others on similar projects to gain an insight into how they completed their projects. This insight is invaluable because it allows us a measure of foresight regarding future risks and can guide us towards the best route to take to ensure success. We can avoid taking routes that may at first appear wise but only with close examination reveal themselves to be unwise.

Konami Card Identification Project

In 2018, Konami staff developed powerful image recognition technology that can distinguish over 9000 different Yugioh cards. However, the main problem was that most cards only contain one card art (and other cards with multiple card arts have card arts significantly different from each other). In a machine learning sense, there are over 10,000 classes where each of those classes contains only one image each. They used Semi-transparent complex images were two slightly semi-transparent cards which are then rendered and merged, this image is then sent for ML processing. This process

was so efficient that ML processing time was reduced from 20 days to 4-5 days only, with 100% accuracy (Arkadia, 2018). They used a specific deck referred to as an Exodia deck for their main analysis which had many distinctions. They used greedy algorithm which will provide them instant output with a strategy for optimal solution. They used a system technique that contains interpreting two marginally different transparent images and combined them together as a traditional learning method. As it will not execute well on datasets that has class which has less than 100 trading cards.

Card Price Prediction

The paper *'Card Price Prediction of Trading Cards'* (Sakaji, Kobayashi, Kohana, Takano, & Izumi, 2019) was an attempt at finding the value of a trading card using ML. The experiment described in this paper was the closest we could find to our project and thus is likely to be the most informative about pitfalls we may face.

The experiment they performed was an attempt to find the price of a card using a variety of ML methods and compared these ML methods to see what the most accurate one was. They inputted all the features of the card (text, cost to play, etc) into a ML system and then tried to predict how that would change the selling price. Their project contrasts with ours in that they totally ignore the physical condition of the card while we focus almost exclusively on it.

The ML techniques they used were: Linear Regression, Random Forest Regressor, Support Vector Regression and Multi-Layer Perceptron Regressor. They found that Linear Regression was the least accurate of the methods they used whereas Multi-Layer Perceptron Regressor (MPLR) was the most accurate.

This indicates that it may be best to use MPLR as a ML method for our project to increase accuracy.

Machine learning for image-based species identification

In ML, for image-based species identification (Jana Wäldchen, 2018) it is established that "Biologists are asking for more efficient methods to meet the identification demand. Smart mobile devices, digital cameras as well as the mass digitisation of natural history collections led to an explosion of openly available image data depicting living organisms. This rapid increase in biological image data in combination with modern ML methods, such as deep learning, offers tremendous opportunities for automated species identification." (Jana Wäldchen, 2018), this is scenario is very similar to the problem that we face today with a high-cost card grading system, there is no proper and efficient service available that allows an individual to give information whether the card should be sent to grade professionally. ML is one of the most invested and researched areas in computer science had led to availability and confluence (Jana Wäldchen, 2018) on three key areas.

- Powerful Computing Hardware.
- Algorithms are capable to take advantage of powerful hardware.
- Ubiquitous training data such as images, social media posts and documents (Jana Wäldchen, 2018).

(ERIC MJOLSNESS, 2001) explains that since ML is a form of Artificial Intelligence, AI, it does not require extensive and specific programming to solve the issue, with the help of AI it can learn from previously fed datasets of relevant images, also known as training, and can read new data in a process called inference. This process has shown that you can explore and extracts data from growing dataset and can be used in application where data are difficult to analyse like image and video analysation (Jana Wäldchen, 2018).

(Jana Wäldchen, 2018) has mentioned the use of computer vision which allows extracting great detailed information from digital images or video. This allows the user to automate the task that a

human visual would attempt to analyse the image condition (Milan Sonka, 2014). Feature extraction and classification are two phases of computer vision. Feature extraction allows extracting meaningful data from the raw data i.e., images and videos, these images are composed of millions of pixels. The extraction process gives relevant information for the classification problem such as colour information, shape, texture and so on. Traditionally, this was all done manually using humans which could give you results that may vary from person to person. (Lowe, 2004) has well described the SIFT, Scale Invariant Feature Transform, where the approach to object detection and image comparison that analyses the image efficiently ever than before and describes the images key characteristics and invariant key points, this is a huge benefit that would have not to be achievable with human analysis. The output of feature extraction is typically a vector file, which can be used to map the score using the classifier, (Jana Wäldchen, 2018).

(Jana Wäldchen, 2018) mentions the network class from deep learning of images which are known as CNN, Convolutional Neural Network, which is programmed to take full advantage of the 2D image supplied and its variability, this program outperformed any other techniques available (Y. Lecun, 1998). To cater for CNN all recent frameworks are compatible with CNN network architectures, supports Graphics Processing Units for advanced training process.

Package	Interface Languages	Platform	GitHub Stars[a]
TensorFlow	Python (Keras), C/C++, Java, Go, R	Linux, macOS, Windows	91,107
Caffe2	Python, MATLAB	Linux, macOS, Windows	30,471
Keras	Python, R	on top of MXNet, TensorFlow, CNTK	26,213
Microsoft Cognitive Toolkit	Python (Keras), C++, command line, BrainScript	Linux, Windows	13,951
Apache MXNet	C++, Python, Julia, MATLAB, JavaScript, Go, R, Scala, Perl	Linux, macOS, Windows, AWS, Android, iOS, JS	13,234

Figure: Top Software Packages used for training CNN (Jana Wäldchen, 2018)

Development in the cloud platform industry has led to services like Amazon Sage Maker from AWS enabling developers to use and deploy ML models regardless of their ML expertise.

Computer Vision-Based Fruit Grading System for Quality Evaluation of Tomato in Agriculture industry

(Megha.P. Arakeri, 2016) have two parts on their methodology Fruit handling and Image processing module. Their image processing module consists of image pre-processing, segmentation, feature extraction and selection, classification and they get their result in two parts i.e., defectiveness; defective or non-defective, and ripeness; ripe or unripe.

(Megha.P. Arakeri, 2016) explains that in the image pre-processing phase the captured image might consist of noise and specular reflection which can affect the grading of the tomato, therefore they have used the median filter to reduce noise and reflections. Otsu's method has been used in the

segmentation phase to convert the image to binary, this allows them to extract the tomato region from the image, the results are partitioned into background and tomato. To determine the defective or non-defective tomato, in their feature extraction and selection phase, they have extracted the colour statistical, colour texture for Red, Green, and Blue.

Colour mean, Standard Deviation, and Skewness were extracted from the colour statistical feature. Contrast, Correlation, Energy, and Homogeneity were extracted from each colour channel using grey level occurrence matric (GLCM) of the image for obtaining colour texture. For determining the ripeness of the tomato, they extracted the colour feature since ripeness is closely related to the colour of the fruit, (Megha.P. Arakeri, 2016). The Red, Green, Blue (RGB) values of the image were used to get the mean of each RGB colour, were greater the mean than the threshold the tomato is ripe otherwise it is unripe.

In the classification phase, the feature extracted were fed as training to a multilayer neural network. This multilayer neural network consists of three layers, Input Layer, Hidden Layer and Output Layer, which indicate whether the tomato is defective or non-defective (Megha.P. Arakeri, 2016). Their dataset consisted of 520 images from the University of Agriculture Sciences, Bangalore. The dataset had a sufficient picture of ripe, unripe, defective, and non-defective tomatoes, their image processing module was implemented using MATLAB (Megha.P. Arakeri, 2016).

Methodology

As we had found a lack of similar projects in our literature review, we decided that the best way to truly answer our hypotheses and find out if they were correct was to attempt to do the project and evaluate the result after that attempt. attempt. We would decide the success of the project based on the accuracy with which it could find the grade of a card by having it grade a card that had already been graded and comparing the similarity of the results.

Research Design Justification

Strengths and Weaknesses

Proceeding with the presented methodology, an assessment was created addressing its strengths and weaknesses:

Strengths:

- In the following Flow Chart, our project was achievable at first. Sufficient resources and tasks were allocated to each team member and a Gantt chart was provided to ensure tasks were done in a timely manner.

Weaknesses:

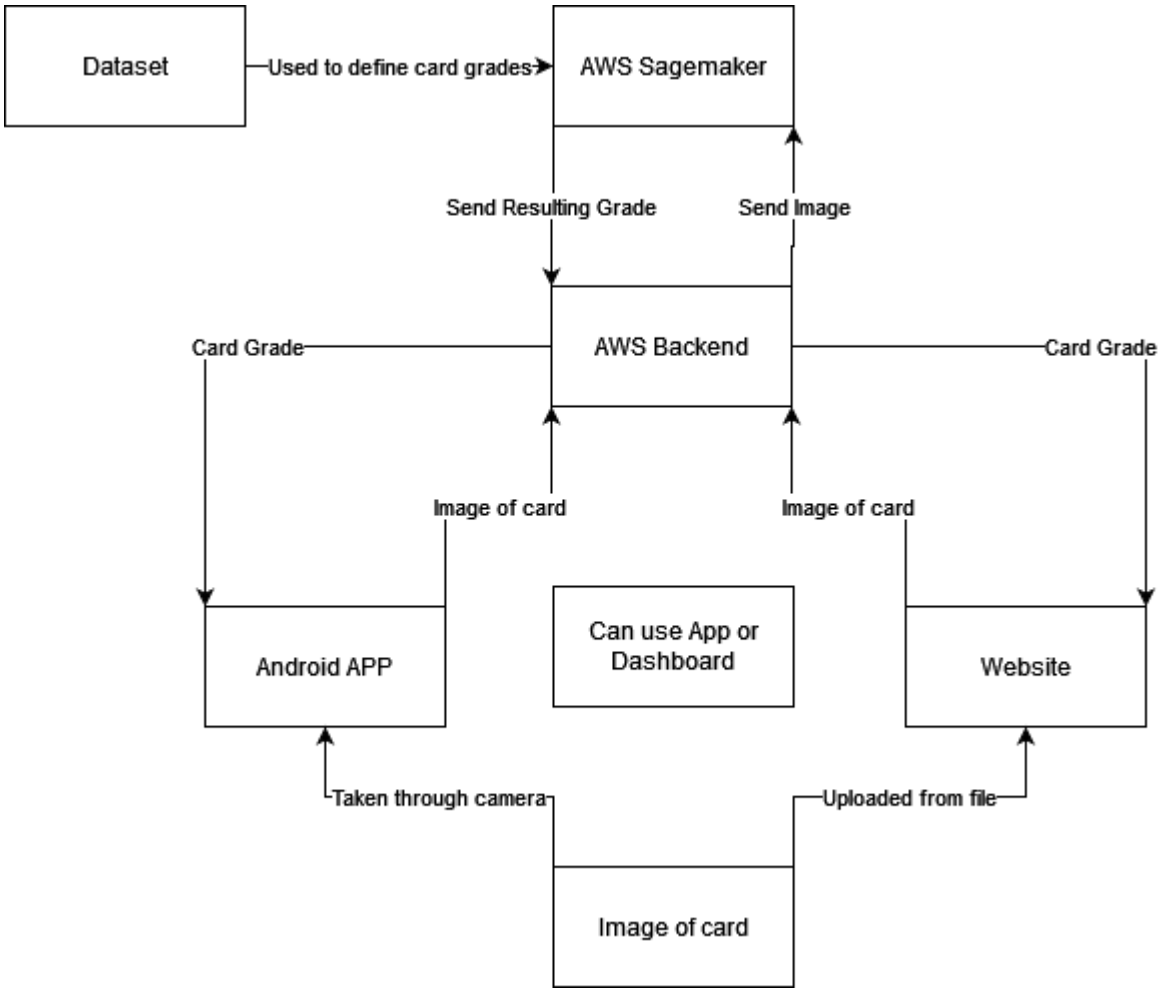
- There was clearly a lack of understanding and experience on the fields of Card Grading and Machine Learning from most team members. Members were found to have not been previously exposed to such subjects and thus, adequate amount of researching and planning on the background of card grading as well as the programs and services that were needed was required to successfully complete the project.

- Lack of Communication among team members. Due to logistical circumstances (COVID restrictions), members had only used Microsoft Teams as the means of communication throughout the semester and did not meet in-person. This limits the effectiveness in our collaboration and reduces the motivation to put in the effort into completing the project.

Research Design

System Process

The following flow chart depicts how our project was originally intended to function and how data was intended to flow between the parts of it.



App

The app was intended to function as a frontend for mobile users who would not be able to use the website due to the size of their screens. Thomas Muldoon was in charge of the app and chose to create an android app because Apple iOS apps require the ownership of a Mac for testing, and He did not own a Mac and the procuring of one was outside of the available budget. The app was

created the Android Studio IDE because that is the official IDE made by Google and it was believed that that would result in a higher level of compatibility with real world devices as well as ensure that the app will be able to work on the latest versions of Android. A plugin called Flutter was also used to make app development easier and to ensure that the app could be later transferred to iOS.

The intended function of the app was that it would take a picture of a card the user had, then send the picture to the machine learning system and then receive and display the resulting grade to the user

Website

For our website development we have chosen php using Laravel framework. We used web-central for hosting our website because it is reputed, trustworthy and provides efficient support. It also provides us with internet domain registration. We also used meta tags to make our website Search Engine Optimization visible and compatible to direct as much internet traffic to the site as possible.

Backend

As the backend existed to serve the Machine Learning system the original choice to use Sagemaker (and the early concept to use AWS Rekognition) was a significant influence on the choice to create the backend in AWS. AWS was chosen early on because we could ensure that all parts of the project would interconnect without issues as well as ease security because the different parts of an AWS solution can connect without being public facing to the internet.

The backend existed to pass information from the APP and Website frontends to the machine learning system and vice versa.

The technologies used in the backend were Amazon API Gateway and AWS Lambda. API Gateway allows us to create an API without the need of a server and therefore reduce costs and complexity. AWS Lambda allows us to run code without a server which was very useful for our project because the backend only existed to pass data between the machine learning system and back to the frontend and thus did not need a large amount of processing power because it would only need to encode the data and decode it for the transfer.

Machine Learning/ Dataset

As previously mentioned, Amazon Sagemaker and SageMaker Ground Truth were to be used for the machine learning model and the dataset, respectively. This would enable a seamless connection with the rest of the backend which would have been the ideal outcome. However, due to an unexpected fee cost that was incurred during the labelling process (*See Appendix X*), we had to find an alternative method to ensure the completion of the project. This resulted in the use of Roboflow (Dataset creation) and the YOLO V5 algorithm (Train the Machine Learning model).

Roboflow was used as the alternative for the labelling job. It a computer vision framework which can substitute most of SageMaker's capabilities. However, the downside of using it was that it was found to be incompatible with the rest of the system meaning that the machine learning part of the project had to branch off to its own, effectively created two different projects. After the completion of the labelling job, the dataset was then exported to a Google collab instance where it was trained with the YOLOv5 algorithm.

YOLO V5, which stands for you only look once, is a machine learning algorithm used for object detection. It helps divide images into a grid system after which the cells are responsible for detecting the objects.

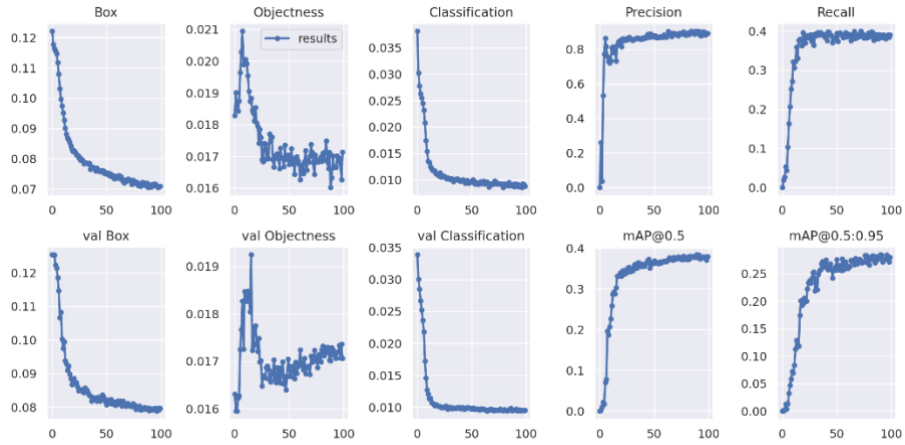
After running the necessary code (*See Appendix Y*) to train the model using 100 epochs, we were able to attain the following results:

Different graphs define the results of the training process

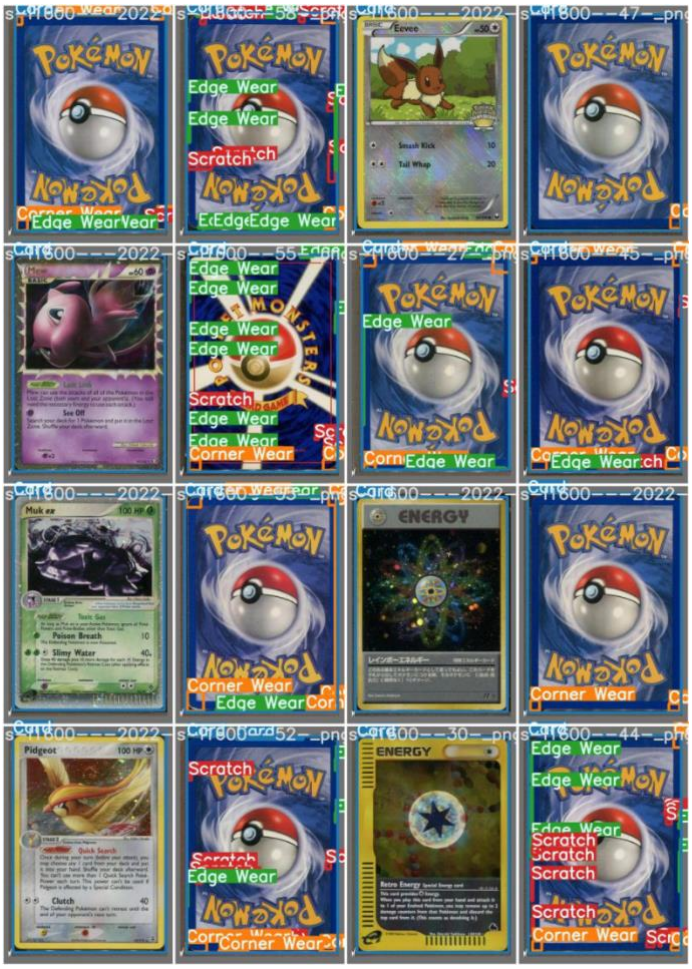
Objectness – Measures that indicate the likelihood of an image window containing an object

Precision – Measure accuracy of positive identifications.

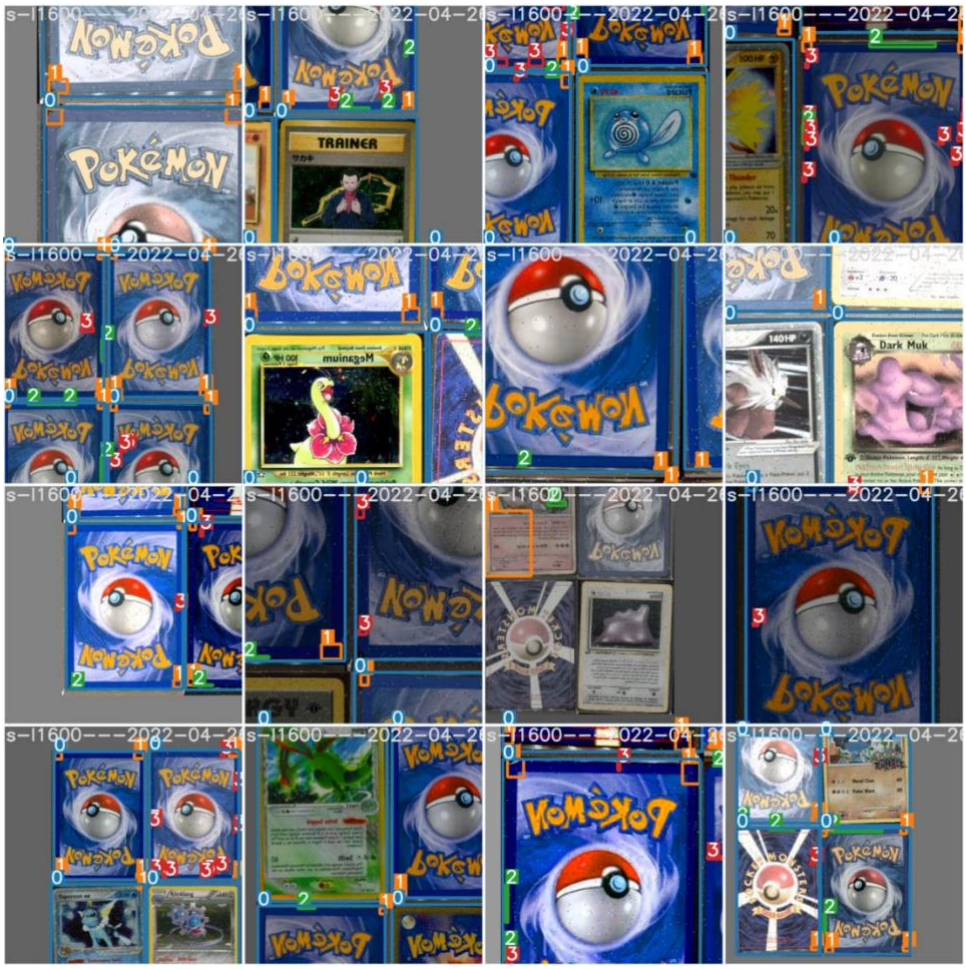
Recall – Number of true positives that are present.



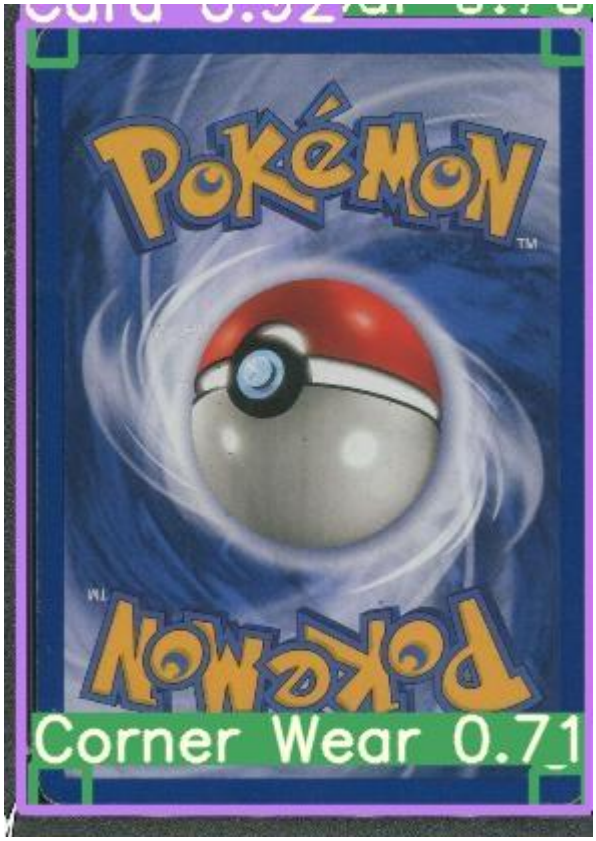
The trained image with the tags described in the labelling process. The images display the different labels assigned to them.



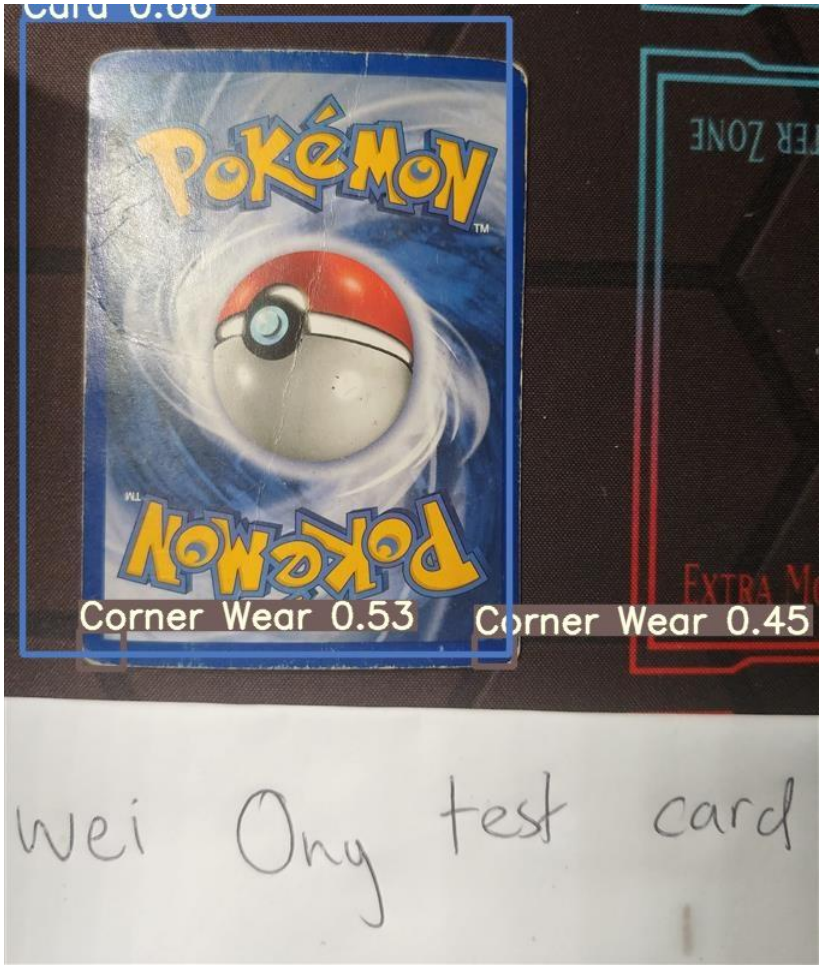
The trained image with augmented tags. The numbers represent the different labels assigned.



The image below is part of the test images from the dataset. The test images were not part of the training, and the model was able to identify the card and corner wear.



The image below represents a physical card that was captured by a smartphone camera. The model was able to detect the card and indicate the corner wear present.



Ethical Considerations:

Data collection for the dataset must be compliant and abide with relevant laws associated the country's origin of the creation of the project (Kublik). Data that was collected for the dataset were under a CC (Creative Commons) license and are used for a non-profitable purpose thus, avoiding any copyright infringements.

We did not retain any information from users and therefore we did not see any privacy problems for users. To ensure that data sent to the website would be secure we used HTTPS to ensure that the images that users sent and the grades they received were not viewable by others.

How study will fill knowledge gap

Our study will fill the knowledge gap seen in the literature review by finding whether it is possible to use machine learning to derive the grade of a card.

Planning and Timeline

Week	Task	Notes
1	• Find Group • Discuss possible research topics	

2	• Decide research topic	
3	• Start researching tools	Submitted project proposal
4	• Continue researching tools	
5	• Begin Researching programming languages	
6	• Continue Researching programming languages • Prototyping	Submitted Literature Review
7	• Continue Researching programming languages • Early prototyping	
8	• Begin all aspects of project in earnest.	Submitted practical project plan and performed early stage presentation
9 (including holidays)	• Finish Camera for App • Start Networking for App • Backend created. • Finished collecting images for the Dataset	
10	• Example output for backend created due to delays with machine learning system. • Finished working on the labelling jobs	
11	• Complete Networking for app • Display output from backend on app. • Encountered an unexpected budget issue in the invoice of SageMaker which forced us to change the entirety of the Backend of the Machine Learning System. • Had to Re-create the labelling jobs. • Re-trained the model using a different algorithm (YOLO v5).	
12	• Completed the training of the model and tested it using sample images (as shown above). • Collaborated on the final presentation	As the project had to be presented on Tuesday, we only had Monday to work on the presentation of the project.

Conclusion

Our findings were that it is possible to create the system that the project proposes and with the assistance of this research another group could do so with far less trouble than we had encountered and could avoid many pitfalls that we encountered. We found that the choice to use Sagemaker was an incorrect choice due to problems we encountered with it, and we should instead have Roboflow from the beginning. The choice of a different backend would have been required by a change in machine learning system.

At this current state, the machine learning system is only capable of identifying flaws (corner/edge wear, scratches) on cards that might affect the grade which would aid a card grader in their job, but it is not currently capable of finding the grade entirely by itself which was the intention of the model. This was the compromise our group had to make due to the cost reasons as mentioned previously.

The choice to do an app as well as a website proved to be unwise, and it would in retrospect be better to have merged the two parts by creating a mobile website for the website to reduce the need for labour on the frontend and therefore free up man-hours for the machine learning system.

Appendix

Website

Machine Learning/Dataset

Appendix X: Total Cost of the Labelling Job in SageMaker Ground Truth

US East (N. Virginia)		**$645.97**
Amazon SageMaker CreateLabelingJob		$645.96
$0.840 per public workforce task	769.000 Tasks	$645.96
Free Tier - First 1,000 objects	892.000 Objects	$0.00

Appendix Y: Python Code/Output for Training the Object Detection Model with 100 epochs

```
# train yolov5s on custom data for 100 epochs
 # time its performance
 %%time
 %cd /content/yolov5/
 !python train.py --img 416 --batch 16 --epochs 100 --data
{dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights
'' --name yolov5s_results  --cache
```

```
/content/yolov5
 github: ⚠ WARNING: code is out of date by 1050 commits. Use 'git pull'
to update or 'git clone https://github.com/ultralytics/yolov5' to download
latest.
 YOLOv5 v4.0-126-g886f1c0 torch 1.11.0+cu113 CUDA:0 (Tesla K80,
11441.1875MB)

Namespace(adam=False, batch_size=16, bucket='', cache_images=True,
cfg='./models/custom_yolov5s.yaml', data='/content/yolov5/Card-Grader-
5/data.yaml', device='', entity=None, epochs=100, evolve=False,
exist_ok=False, global_rank=-1, hyp='data/hyp.scratch.yaml',
image_weights=False, img_size=[416, 416], linear_lr=False, local_rank=-1,
log_artifacts=False, log_imgs=16, multi_scale=False,
name='yolov5s_results', noautoanchor=False, nosave=False, notest=False,
project='runs/train', quad=False, rect=False, resume=False,
save_dir='runs/train/yolov5s_results', single_cls=False, sync_bn=False,
total_batch_size=16, weights='', workers=8, world_size=1)
 wandb: Install Weights & Biases for YOLOv5 logging with 'pip install
wandb' (recommended)
```

```
 Start Tensorboard with "tensorboard --logdir runs/train", view at
http://localhost:6006/
 hyperparameters: lr0=0.01, lrf=0.2, momentum=0.937, weight_decay=0.0005,
warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05,
cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0,
fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0,
translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0,
fliplr=0.5, mosaic=1.0, mixup=0.0

                 from  n    params  module
arguments
  0                -1  1      3520  models.common.Focus
[3, 32, 3]
  1                -1  1     18560  models.common.Conv
[32, 64, 3, 2]
  2                -1  1     19904  models.common.BottleneckCSP
[64, 64, 1]
  3                -1  1     73984  models.common.Conv
[64, 128, 3, 2]
  4                -1  1    161152  models.common.BottleneckCSP
[128, 128, 3]
  5                -1  1    295424  models.common.Conv
[128, 256, 3, 2]
  6                -1  1    641792  models.common.BottleneckCSP
[256, 256, 3]
  7                -1  1   1180672  models.common.Conv
[256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP
[512, 512, [5, 9, 13]]
  9                -1  1   1248768  models.common.BottleneckCSP
[512, 512, 1, False]
 10                -1  1    131584  models.common.Conv
[512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat
[1]
 13                -1  1    378624  models.common.BottleneckCSP
[512, 256, 1, False]
 14                -1  1     33024  models.common.Conv
[256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat
[1]
 17                -1  1     95104  models.common.BottleneckCSP
[256, 128, 1, False]
 18                -1  1    147712  models.common.Conv
[128, 128, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat
```

```
[1]
  20                -1  1    313088  models.common.BottleneckCSP
[256, 256, 1, False]
 21                -1  1    590336  models.common.Conv
[256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat
[1]
 23                -1  1   1248768  models.common.BottleneckCSP
[512, 512, 1, False]
 24      [17, 20, 23]  1     24273  models.yolo.Detect
[4, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156,
198, 373, 326]], [128, 256, 512]]
 /usr/local/lib/python3.7/dist-packages/torch/functional.py:568:
UserWarning: torch.meshgrid: in an upcoming release, it will be required
to pass the indexing argument. (Triggered internally at
../aten/src/ATen/native/TensorShape.cpp:2228.)
   return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
 Model Summary: 283 layers, 7263185 parameters, 7263185 gradients, 16.8
GFLOPS

Scaled weight_decay = 0.0005
 Optimizer groups: 62 .bias, 70 conv.weight, 59 other
 train: Scanning 'Card-Grader-5/train/labels' for images and labels...
1311 found, 0 missing, 0 empty, 0 corrupted: 100% 1311/1311 [00:00<00:00,
2228.81it/s]
 train: New cache created: Card-Grader-5/train/labels.cache
 train: Caching images (0.5GB): 100% 1311/1311 [00:02<00:00, 531.56it/s]
 val: Scanning 'Card-Grader-5/valid/labels' for images and labels... 122
found, 0 missing, 0 empty, 0 corrupted: 100% 122/122 [00:00<00:00,
1117.16it/s]
 val: New cache created: Card-Grader-5/valid/labels.cache
 val: Caching images (0.0GB): 100% 122/122 [00:00<00:00, 361.02it/s]
 Plotting labels...

autoanchor: Analyzing anchors... anchors/target = 1.83, Best Possible
Recall (BPR) = 0.7100. Attempting to improve anchors, please wait...
 autoanchor: WARNING: Extremely small objects found. 2311 of 10030 labels
are < 3 pixels in size.
 autoanchor: Running kmeans for 9 anchors on 10018 points...
 autoanchor: thr=0.25: 0.9496 best possible recall, 1.91 anchors past thr
 autoanchor: n=9, img_size=416, metric_all=0.166/0.633-mean/best,
past_thr=0.502-mean: 6,8,  4,48,  54,5,  21,16,  5,108,  116,5,  223,14,
19,260,  285,405
 autoanchor: Evolving anchors with Genetic Algorithm: fitness = 0.6734:
100% 1000/1000 [00:02<00:00, 355.28it/s]
 autoanchor: thr=0.25: 0.9903 best possible recall, 1.99 anchors past thr
 autoanchor: n=9, img_size=416, metric_all=0.174/0.675-mean/best,
past_thr=0.510-mean: 6,6,  3,24,  23,3,  4,77,  19,18,  126,6,  166,8,
12,245,  281,401
 autoanchor: New anchors saved to model. Update model *.yaml to use these
```

```
anchors in the future.

Image sizes 416 train, 416 test
 Using 2 dataloader workers
 Logging results to runs/train/yolov5s_results
 Starting training for 100 epochs...

     Epoch   gpu_mem       box       obj       cls     total   targets
img_size
       0/99     1.78G    0.1222   0.01829   0.03816    0.1786       152
416: 100% 82/82 [00:41<00:00,  1.99it/s]
               Class      Images     Targets           P           R
mAP@.5  mAP@.5:.95: 100% 4/4 [00:03<00:00,  1.09it/s]
                 all         122           0           0           0
0           0

     Epoch   gpu_mem       box       obj       cls     total   targets
img_size
       1/99      1.8G    0.1178   0.01901   0.03025     0.167       136
416: 100% 82/82 [00:36<00:00,  2.27it/s]
               Class      Images     Targets           P           R
mAP@.5  mAP@.5:.95: 100% 4/4 [00:01<00:00,  2.73it/s]
                 all         122         948       0.261      0.0186
0.00167    0.000318


     Epoch   gpu_mem       box       obj       cls     total   targets
img_size
      99/99      1.8G   0.07089   0.01714  0.008795   0.09683       143
416: 100% 82/82 [00:34<00:00,  2.36it/s]
               Class      Images     Targets           P           R
mAP@.5  mAP@.5:.95: 100% 4/4 [00:03<00:00,  1.29it/s]
                 all         122         948       0.894        0.39
0.38        0.28
                Card         122         121       0.979       0.992
0.987       0.945
          Corner Wear         122         248       0.597       0.569
0.529       0.174
           Edge Wear         122         327           1           0
0.00285    0.000422
             Scratch         122         252           1           0
5.84e-05    9.54e-06
 Optimizer stripped from runs/train/yolov5s_results/weights/last.pt,
14.8MB
 Optimizer stripped from runs/train/yolov5s_results/weights/best.pt,
14.8MB
 100 epochs completed in 1.028 hours.

CPU times: user 42.2 s, sys: 6.43 s, total: 48.6 s
 Wall time: 1h 2min 12s
```

App

main.dart

```
import 'dart:async';

import 'dart:io';

import 'dart:typed_data';

import 'package:http/http.dart' as http;

import 'package:camera/camera.dart';

import 'package:flutter/material.dart';

import 'dart:convert';

import 'package:path/path.dart';


class MyButton extends StatelessWidget {

 const MyButton({Key? key}) : super(key: key);


 @override

 Widget build(BuildContext context) {

  return GestureDetector(

   onTap: () {

    print('Go to Camera');

   },

   child: Container(

    height: 50.0,

    padding: const EdgeInsets.all(8.0),

    margin: const EdgeInsets.symmetric(horizontal: 8.0),

    decoration: BoxDecoration(

     borderRadius: BorderRadius.circular(5.0),

     color: Colors.lightGreen[500],

    ),

    child: const Center(

     child: Text('Take Picture'),

    ),
```

```
    ),
   );
  }
}

class CameraWindow extends StatefulWidget {
  const CameraWindow({Key? key, required this.camera}) : super(key: key);
  final CameraDescription camera;

  @override
  CameraWindowScreenState createState() => CameraWindowScreenState();
}

class CameraWindowScreenState extends State<CameraWindow> {
  late CameraController _controller;
  late Future<void> _initializeControllerFuture;

  @override
  void initState() {
    super.initState();
    _controller = CameraController(widget.camera, ResolutionPreset.max);
    _initializeControllerFuture = _controller.initialize();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(title: const Text('Take a picture')),
      body: FutureBuilder<void>(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            return CameraPreview(_controller);
          } else {
            return const Center(child: CircularProgressIndicator());
          }
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          try {
            await _initializeControllerFuture;
            final image = await _controller.takePicture();
            print('picture taken');

            Navigator.push(context,
              MaterialPageRoute(builder: (context) =>  SendQuery(imagePath: image.path)));
          } catch (e) {
              print(e);
          }
        },
      ),
    );
  }
}

// Sourced from https://docs.flutter.dev/cookbook/plugins/picture-using-camera
```

```
class SendQuery extends StatelessWidget {
  final String imagePath;
  const SendQuery({Key? key, required this.imagePath});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: const Text('Do you want to send that photo?')),
      body: Center(
        child: ElevatedButton(
            onPressed: () => {
              uploadFileToServer(imagePath)


            },
            child: const Text('Yes')
            )
      ),
    );
  }
}


Future uploadFileToServer(String imagePath) async {
  Map TestData = { 'testkey' : "test"};
  String body = json.encode(TestData);
  var response = await http.post(Uri.parse('https://x65k5b8f46.execute-api.us-east-
1.amazonaws.com/RestV1'),
  headers: <String, String>{
    'Content-Type' : 'application/json; charset=UTF-8'
  },
  body: body,);
```

```
  print('Response status: ${response.statusCode}');
  print('Response body: ${response.body}');
  var grade = response.body;
  return grade;
/*
   File file = File(imagePath);
  List<int> imageBytes = file.readAsBytesSync();
  String base64Image = base64.encode(imageBytes);

  var request = http.MultipartRequest(
      "POST", Uri.parse('https://x65k5b8f46.execute-api.us-east-1.amazonaws.com/RestV1'));

  request.fields['title'] = 'Card Image';
  request.files.add(await http.MultipartFile.fromPath('Card Image', imagePath));
  var response = await request.send();
  print(response.statusCode);
  response.stream.transform(utf8.decoder).listen((value) {
    print(value);
  });*/
}



Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  final cameras = await availableCameras();
  final firstCamera = cameras.first;

  runApp(
     MaterialApp(
      home: CameraWindow(camera: firstCamera),
```

```
    ),
  );
}
```

pubspec.yaml

```
name: card_grader

description: A new Flutter project.


# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish_to: 'none' # Remove this line if you wish to publish to pub.dev


# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number used as
CFBundleVersion.

# Read more about iOS versioning at

#
https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

version: 1.0.0+1


environment:

  sdk: ">=2.16.2 <3.0.0"


# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,
```

```
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
  camera: ^0.9.5
  path_provider: ^2.0.10
  path: ^1.8.0
  http: ^0.13.4
  english_words: ^4.0.0
  typed_data: ^1.3.0



  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.4

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^1.0.4

# For information on the generic Dart part of this file, see the
```

```
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware.

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
```

```
#        style: italic
#    - family: Trajan Pro
#      fonts:
#        - asset: fonts/TrajanPro.ttf
#        - asset: fonts/TrajanPro_Bold.ttf
#          weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

Backend

Blogs

Bibliography

Bibliography

Arkadia, N. (2018). *Konami Staff Develop Powerful Image Recognition Technology*. Retrieved March 26, 2022, from https://ygorganization.com/konami-staff-develop-powerful-image-recognition-technology/#:~:text=%E2%80%9CKonami%20develops%20image%20recognition%20technology,the%20CEDEC%202018%20game%20developer's

ERIC MJOLSNESS, D. D. (2001, September 14). *Machine Learning for Science: State of the Art and Future Prospects.* Retrieved March 20, 2022, from https://www.science.org/doi/10.1126/science.293.5537.2051

Hajducky, D. (2021, August 16). *T206 Honus Wagner baseball card sells for $6.606 million, shattering previous record*. Retrieved May 20, 2022, from ESPN: https://www.espn.com/mlb/story/_/id/32031670/t206-honus-wagner-baseball-card-sells-6606-million-shattering-previous-record

Jana Wäldchen, P. M. (2018, August 13). *Machine learning for image based species identification.* Retrieved March 20, 2022, from https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.13075

Jung, S.-G., An, J., Kwak, H., Salminen, J., & Jansen, B. J. (2018). Assessing the Accuracy of Four Popular Face Recognition Tools for Inferring Gender, Age, and Race. *Twelfth International AAAI Conference on Web and Social Media*, (pp. 624 - 627). Retrieved March 16, 2020, from https://www.aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/viewFile/17839/17066

Lowe, D. G. (2004). *Distinctive Image Features from Scale-Invariant Keypoints*. Retrieved March 23, 2022, from https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

Megha.P. Arakeri, L. (2016). *Computer Vision Based Fruit Grading System for Quality Evaluation of Tomato in Agriculture industry*. Retrieved March 24, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050916001861?via%3Dihub

Milan Sonka, V. H. (2014). *Image Processing, Analysis, and Machine Vision* (Fourth Edition ed.). Cengage Learning.

Nadon, J. (2017). *Website Hosting and Migration with Amazon Web Services.* Kingsville, Ontario, Canada: Apress. Retrieved March 20, 2022, from https://link.springer.com/content/pdf/10.1007%2F978-1-4842-2589-9.pdf

Sakaji, H., Kobayashi, A., Kohana, M., Takano, Y., & Izumi, K. (2019). Card Price Prediction of Trading Cards Using Machine Learning Methods. (L. Barolli, H. Nishino, T. Enokido, & M. Takizawa, Eds.) *Advances In Networked-based Information Systems, 1036*, 705 - 714. Retrieved March 17, 2021, from https://link.springer.com/book/10.1007/978-3-030-29029-0

Snow, J. (2018, July 26). *Amazon's Face Recognition Falsely Matched 28 Members of Congress With Mugshots*. Retrieved March 24, 2022, from ACLU NorCal: https://www.aclunc.org/blog/amazon-s-face-recognition-falsely-matched-28-members-congress-mugshots

Washington County Sheriff. (2018, May 22). *PSWeb Facial Recognition.* Retrieved March 22, 2022, from ACLU Oregon: https://www.aclunc.org/docs/20180522_ARD.pdf

Y. Lecun, L. .. (1998). *Gradient-based learning applied to document recognition*. Retrieved March 23, 2022, from https://ieeexplore.ieee.org/document/726791/authors#authors

Abr`amoff, M., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. and Niemeijer, M., 2018. Improved Automated Detection of Diabetic Retinopathy on a Publicly Available Dataset Through Integration of Deep Learning. [ebook] Iowa: Retina, pp.5200,5201,5202,5203,5204. Available at: <https://iovs.arvojournals.org/article.aspx?articleid=2565719> [Accessed 25 March 2022].

Alfayez, M., 2019. The extent to which different resolutions affect the success rate of image recognition software implemented through Convoluted Neural Networks. [ebook] Methkal Alfayez, pp.4, 11, 12, 13, 14, 15, 19. Available at: <https://www.researchgate.net/publication/339130684> [Accessed 26 March 2022].

Bassett, R Western Connecticut State University 2003, 'Machine assisted visual grading of rare collectibles over the Internet', *School of Computer Science and Information Systems*. July, viewed 25 March 2022, <https://Bassett.PDF (psu.edu)>.

Kend, M Australian National University (ANU) 2007, 'The Impact of Product Grading: Adding Value to Online eBay Trading', *SSRN*. pp. 1-17, viewed 25 March 2022, <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=981649>.

McCole, P. (2002), "The role of trust for electronic commerce in services", *International Journal of Contemporary Hospitality Management*, Vol. 14 No. 2, pp. 81-87, viewed 25 March 2022, < https://doi.org/10.1108/09596110210419264>.

Arkadia, N., 2018. Konami Staff Develop Powerful Image Recognition Technology. [Online] Available at: https://ygorganization.com/konami-staff-develop-powerful-image-recognition-technology/#:~:text=%E2%80%9CKonami%20develops%20image%20recognition%20technology,the%20CEDEC%202018%20game%20developer's [Accessed 26 March 2022]

Kublik, V., 2022. *EU/US Copyright Law and Implications on ML Training Data*. [online] Valohai. Available at: <https://valohai.com/blog/copyright-laws-and-machine-learning/> [Accessed 28 May 2022].