DATS 6203 Final Project Report - Group
Trading Card Authenticator

Ashwin Dixit, Jose Garcia
The George Washington University

24th April 2022

Table of Contents

Introduction:

The goal of our project is to identify if Pokémon trading cards are real or counterfeit based on their images.

Rare Pokémon cards are highly sought after by collectors, with some even valued over hundreds of thousands of dollars. It would be important for collectors to tell a genuine card from fake card to avoid getting scammed.

Our project aims to train a model on images of real and counterfeit trading cards and develop an interface to identify if a card is real or not.

Dataset Description:

For our project we used the 'Real and Fake Pokémon Cards Dataset'. This dataset was taken from Kaggle. The dataset contained predefined train and test split (80:20).

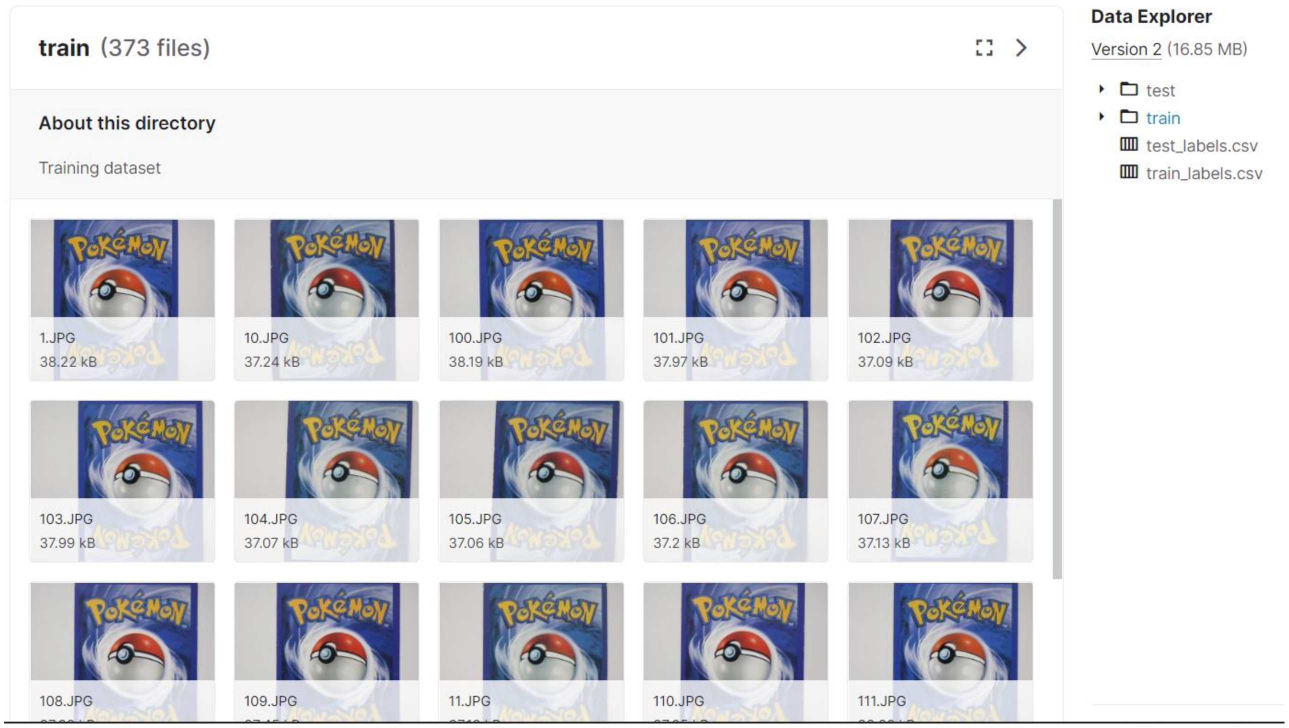


Figure 1: Dataset

The Train folder contains 451 images, Train folder contains 373 images and test folder contains 78 images. This was given as training data for Custom CNN model.

Counterfeit and Real Cards:

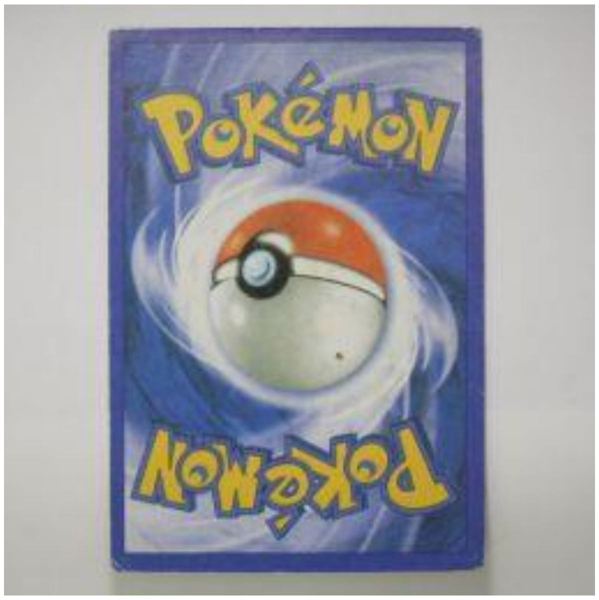


Figure 2: Counterfeit Card

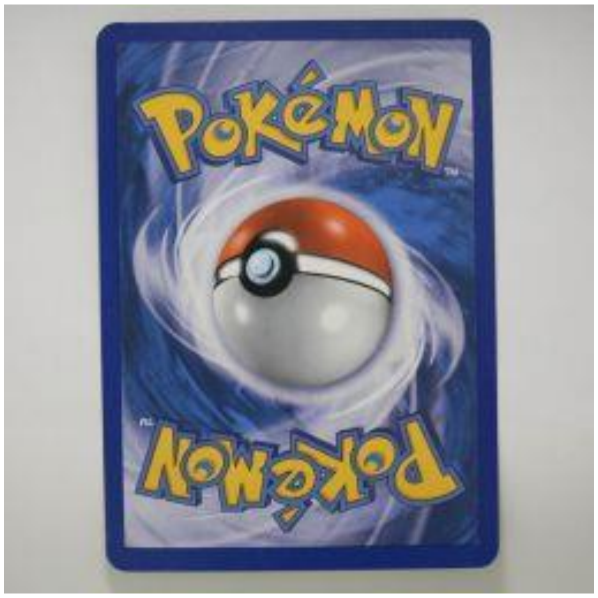


Figure 3: Real Card

Deep Learning Network:

In this project, we built a custom CNN model. We trained the build model with 200 epochs and selected the model which gave the best accuracy and f1-score.

Convolution Networks:

Convolution Neural Network -- [S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.]

A convolution network is a multilayer feedforward network that has two or three-dimensional inputs. It has weight functions that are not generally viewed as matrix multiplication operations.

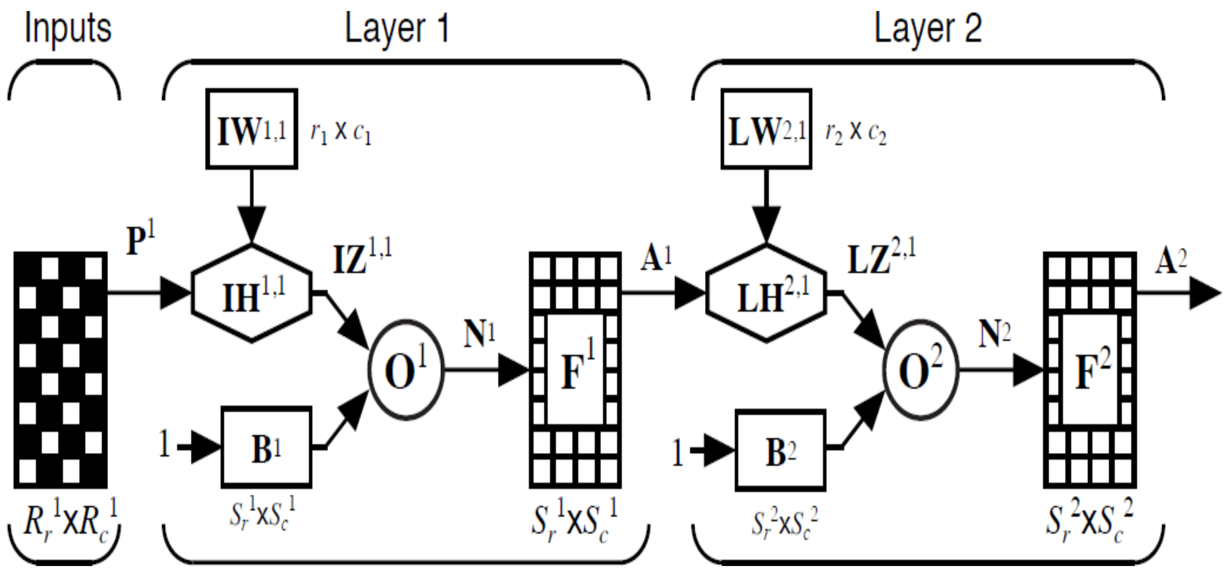


Figure 4: Convolution Network

Let the input image be represented by the Rr x Rc matrix V. The weight function for this layer performs a convolution kernel that is represented by the r x c matrix W.

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1} \cdot$$

Figure 5: Convolution Operation

Model Metrics:

We looked at the following metrics in deciding how effective our models were:

F1-score

F1-score weights precision and recall [blog.floydhub.com].

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Figure 7: F1-score Calculation

Accuracy

Accuracy is the percentage of observations that are classified correctly[blog.floydhub.com].

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

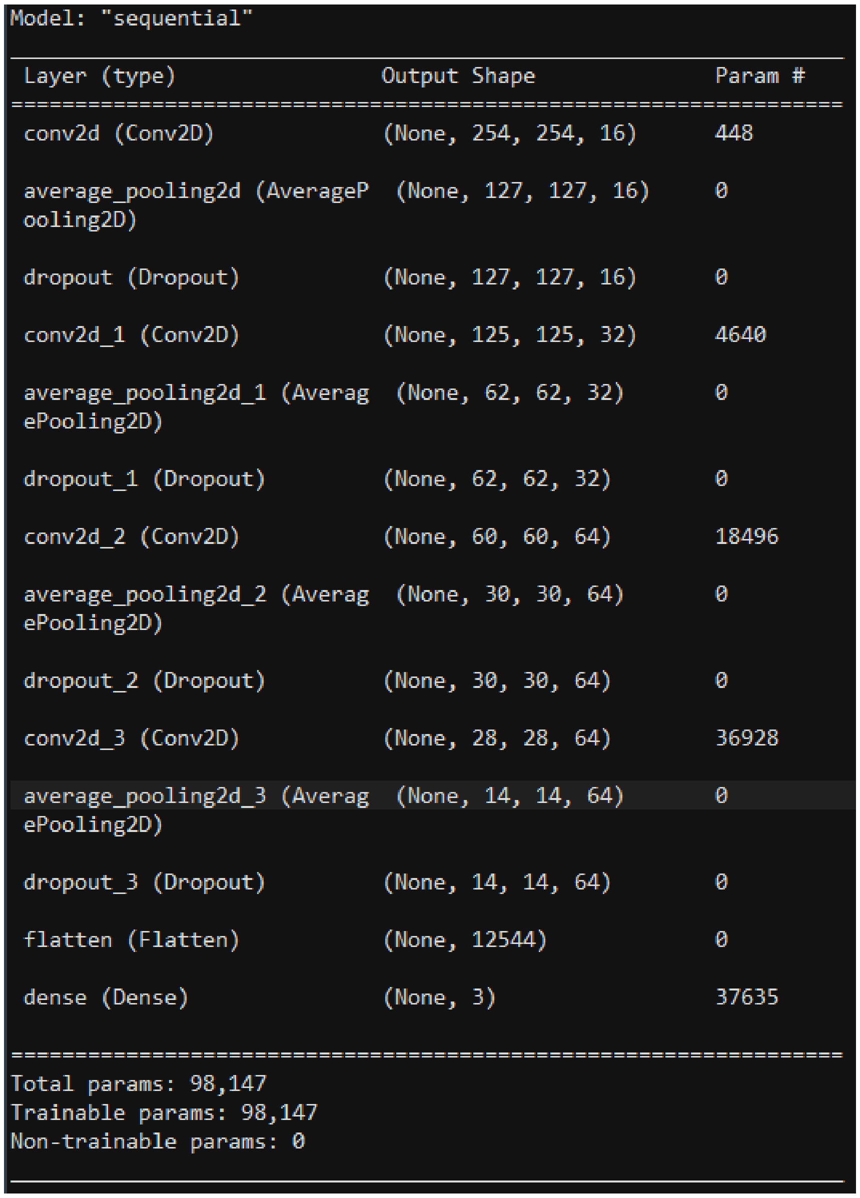
Figure 8: Accuracy Calculation

Experimental Setup:

The dataset contained predefined train and test split (80:20). In addition to this, we derive a stratified validation dataset from the train dataset to judge model performance (90:10).

We also added a third class of random images so that our Trading Card Authentication Application could predict invalid input images.

Custom CNN Model:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 average_pooling2d (AverageP  (None, 127, 127, 16)     0
 ooling2D)

 dropout (Dropout)           (None, 127, 127, 16)      0

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 average_pooling2d_1 (Averag  (None, 62, 62, 32)       0
 ePooling2D)

 dropout_1 (Dropout)         (None, 62, 62, 32)        0

 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496

 average_pooling2d_2 (Averag  (None, 30, 30, 64)       0
 ePooling2D)

 dropout_2 (Dropout)         (None, 30, 30, 64)        0

 conv2d_3 (Conv2D)           (None, 28, 28, 64)        36928

 average_pooling2d_3 (Averag  (None, 14, 14, 64)       0
 ePooling2D)

 dropout_3 (Dropout)         (None, 14, 14, 64)        0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 3)                 37635

=================================================================
Total params: 98,147
Trainable params: 98,147
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Custom CNN Model Summary

Hyperparameters:

Learning Rate - 0.0001

Batch Size - 32

Optimizer - Adam

Epochs - 200

Patience - 40 epochs

Dropout - 0.2

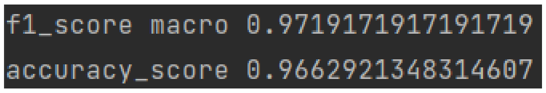
Kernel Size - (3, 3)

Pool Size - (2, 2)

Results:

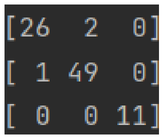
The performance of the model was judged based on the accuracy and loss values for train, validation and test sets.

Final Performance of Model:



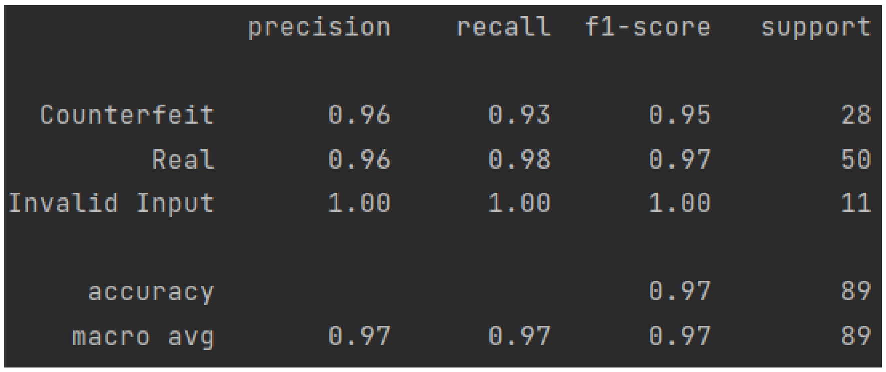
```
f1_score macro 0.9719171917191719
accuracy_score 0.9662921348314607
```

Figure 7: Classification Metrics



```
[26  2  0]
[ 1 49  0]
[ 0  0 11]
```

Figure 8: Confusion Matrix



```
               precision    recall  f1-score   support

  Counterfeit       0.96      0.93      0.95        28
         Real       0.96      0.98      0.97        50
Invalid Input       1.00      1.00      1.00        11

     accuracy                           0.97        89
    macro avg       0.97      0.97      0.97        89
```

Figure 9: Classification Report

Interpretability and Explainability: Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM), uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting important regions in the image for predicting the concept.

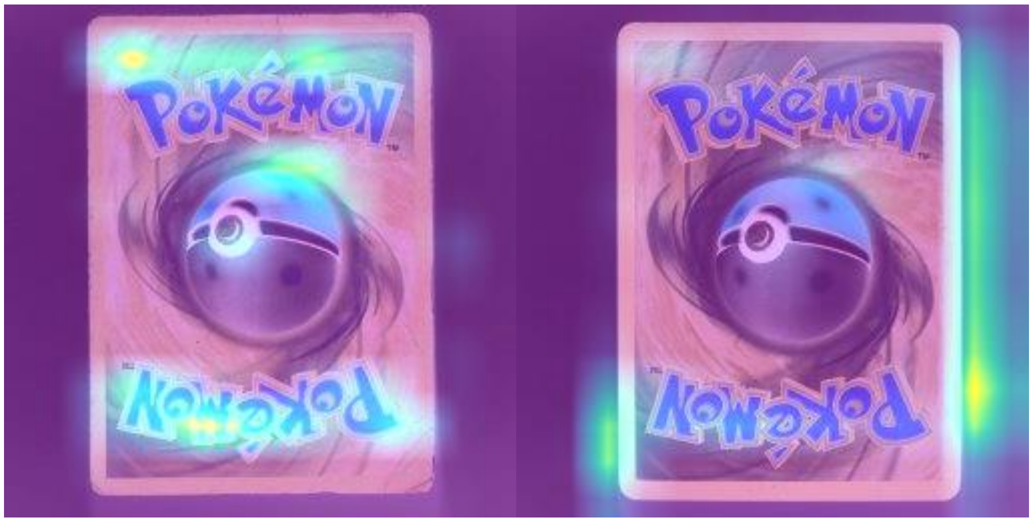


Figure 9: Grad-CAM for Counterfeit and Real Image respectively

We can see that the model discriminates between classes based on details and borders on the cards.

Note: Grad-CAM has a limitation and sometimes highlights locations the model did not actually use as a side effect of the gradient averaging step

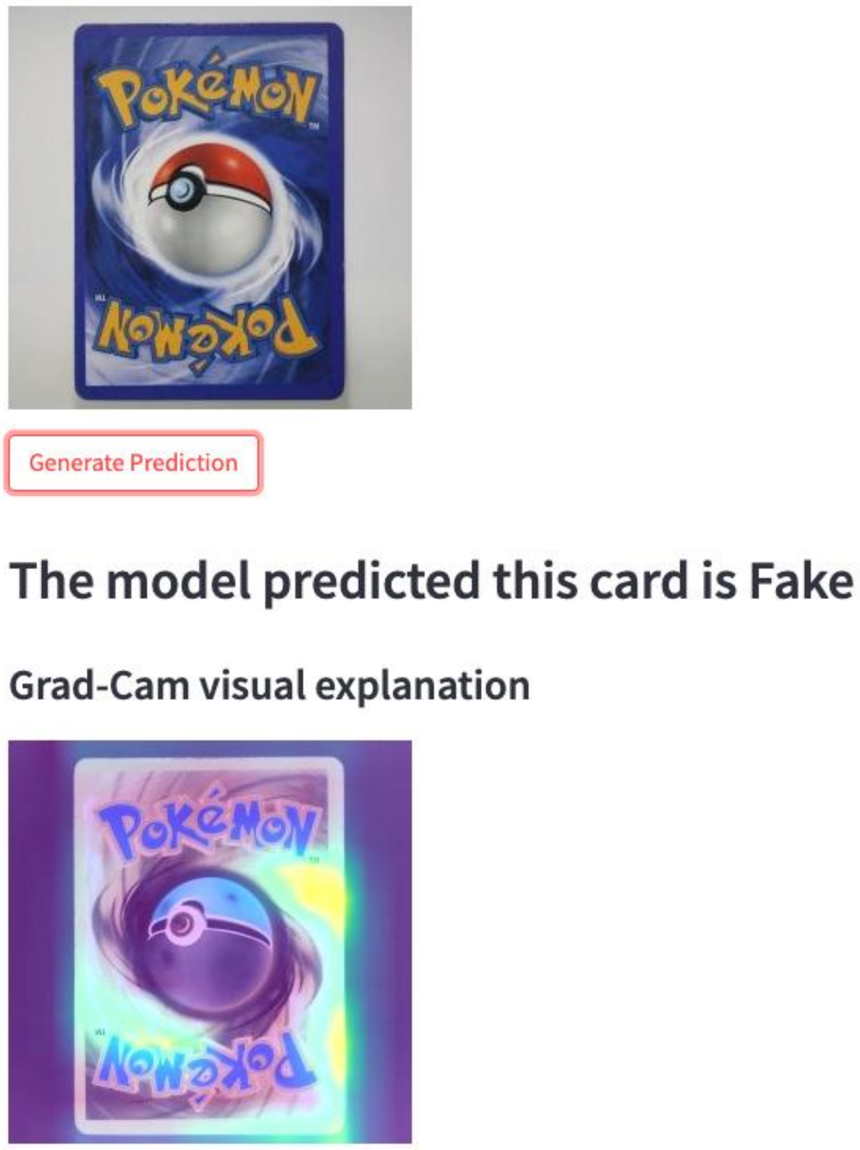
Summary:

1. Accuracy of the model = 96.62%.
2. Overall f1-score of the model = 0.9719
3. From the classification report of models:
    1. F1 score for Counterfeit Cards = 0.95,
       F1 score for Real Cards = 0.97,
       F1 score for Invalid Input = 1.00
    2. Precision for Counterfeit Cards = 0.96,
       Precision for Real Cards = 0.96,
       Precision for Invalid Input = 1.00
    3. Recall for Counterfeit Cards = 0.93,
       Recall for Real Cards = 0.98,
       Recall for Invalid Input = 1.00

Application:

We created an app where you can upload a photo and it predicts if it’s of a Real/ Counterfeit Pokémon Trading Card or is an Invalid Input.

We used Streamlit for the frontend of the application. Streamlit is an open-source app framework in Python language. It allowed us to deploy our model in a local host and demonstrate the performance of the model in real time. In the backend, the trained model is uploaded and performs predictions. The results of these predictions are mapped to the different classes. This application outputs the prediction label and the activation maps of the image to visualize how parts of the image affects the model’s output.



The model predicted this card is Fake

Grad-Cam visual explanation

Figure 10: Streamlit Application

Conclusion:

From the results we can see that the custom-built CNN model is able to perform well for the task of classifying inputs - Real, Counterfeit or Invalid Images. We can see that it makes decisions based on image details like quality of text and illustrations, border thickness and edge sharpness.

Future scope for this project could be building an ensemble model with front-side images of cards so that the model can decide based on both front and backside of the cards. Ideas from this project could also be used in detecting other counterfeit items like currency notes and have a sizeable real-world impact.

References:

1. Datasets - https://www.kaggle.com/datasets/ongshujian/real-and-fake-pokemon-cards, https://www.kaggle.com/datasets/jamesmcguigan/playingcards

2. TensorFlow - https://www.tensorflow.org/api_docs/python/tf/keras

3. Grad CAM - https://arxiv.org/abs/1610.02391, https://arxiv.org/abs/2011.08891

4. Streamlit - https://docs.streamlit.io/