Title: "Integrating Card Recognition for Blackjack Gameplay without QR Code Assistance"

Syed Ali Hasany

Abstract:

This project focuses on incorporating a card recognition system into Cozmo, enabling it to play the game of blackjack without relying on QR codes. The integration involves training a machine learning (ML) model to recognize and classify playing cards, followed by implementing the model within Cozmo's decision-making logic script.

Introduction:

This project aims to enhance the realism of the Cozmo robot playing a game of blackjack by integrating a card recognition model, allowing it to recognize cards without the aid of QR codes. Cozmo will recognize cards in a deck, using a trained machine learning (ML) model and display the suit and value on its LCD. This project combines elements of programming Cozmo in Python, developing a machine learning model for card recognition and later incorporating that in an actual game of Blackjack.

Baseline and Strech Goals:

Card Recognition Model (C):

Generate ground truth to train a machine learning model to recognize and classify playing cards. Implement the model in Python, leveraging computer vision libraries such as OpenCV, Tensorflow.

Cozmo Integration in gameplay(B):

Integrate the card recognition model in Cozmo's decision-making logic for playing blackjack based on the recognized cards (suit and value).

Cozmo feedback(A):

Provide real-time feedback from Cozmo, via using its LCD display to show the card value and suit.

Methodology:

Project Overview:

The project consists of four steps:

1. Data Collection and Model Labeling: Taking enough pictures using Cozmo to train a ML model.
2. Machine Leaning Model Overview: Making a ML model to recognize suit and card using cozmo’s camera in real time.
3. Blackjack Gameplay: Integrating the model in the gameplay of blackjack.
4. LCD Display: Displaying the suit and value on the LCD of Cozmo.

Data Collection and Labeling:

I developed two scripts in python to help me collect images for making a data set of images of the 52 playing cards and the joker. The first script uses slower processing to give the user ample time to change the location and angle of cozmo significantly for more diverse images. The second much faster script can

be used by the user to take images significantly quickly, a specific use case could be in a situation where you do not want to vary the information in subsequent images greatly. I used both the scripts to take 400 images of each card.

Machine Leaning Model Overview:

Model Architecture

The model is built using a convolutional neural network (CNN) architecture, designed to recognize card images.

Data Preprocessing

Data augmentation techniques are applied to the images in the training dataset to increase diversity and improve model generalization. Techniques include random horizontal flipping, rotation, and zooming. The dataset consists of card images organized in directories, and inferred labels are categorized using a categorical label mode.

Model Layers

The architecture comprises several convolutional layers (Conv2D) and max-pooling layers (MaxPooling2D) to extract features from the input card images.

Convolutional layers with increasing filter sizes (16, 32, 64, 128) are stacked, each followed by max-pooling layers to reduce spatial dimensions.

After the final max-pooling layer, a flatten layer is applied to convert the 2D output into a 1D vector.

Dense layers are added for further feature extraction and mapping, leading to the final output layer.

The output layer consists of 53 nodes (assuming a 52-card deck plus the Joker since I wanted to make sure that people can use this code for games involving the joker), using softmax activation for multi-class classification.

The convolutional layers learn visual patterns, while max-pooling layers downsample features, focusing on crucial information. The depth, four layers in my case, facilitates learning intricate patterns and hierarchical representations present in the card images. The utilization of Rectified Linear Unit (ReLU) activation function introduces non-linearity, aiding in learning complex image relationships. The Softmax activation function in the output layer generates a probability distribution across card categories, making the model suitable for multi-class classification by providing probabilistic predictions for each card category.

Model Compilation

The model is compiled with categorical cross-entropy loss, Adam optimizer with a learning rate of 1e-3.

Training Configuration

Early Stopping callback is defined to monitor validation accuracy and restore the best weights.

Model Checkpoint callback is set up to save the best-performing model during training.

Evaluation of the ML model:

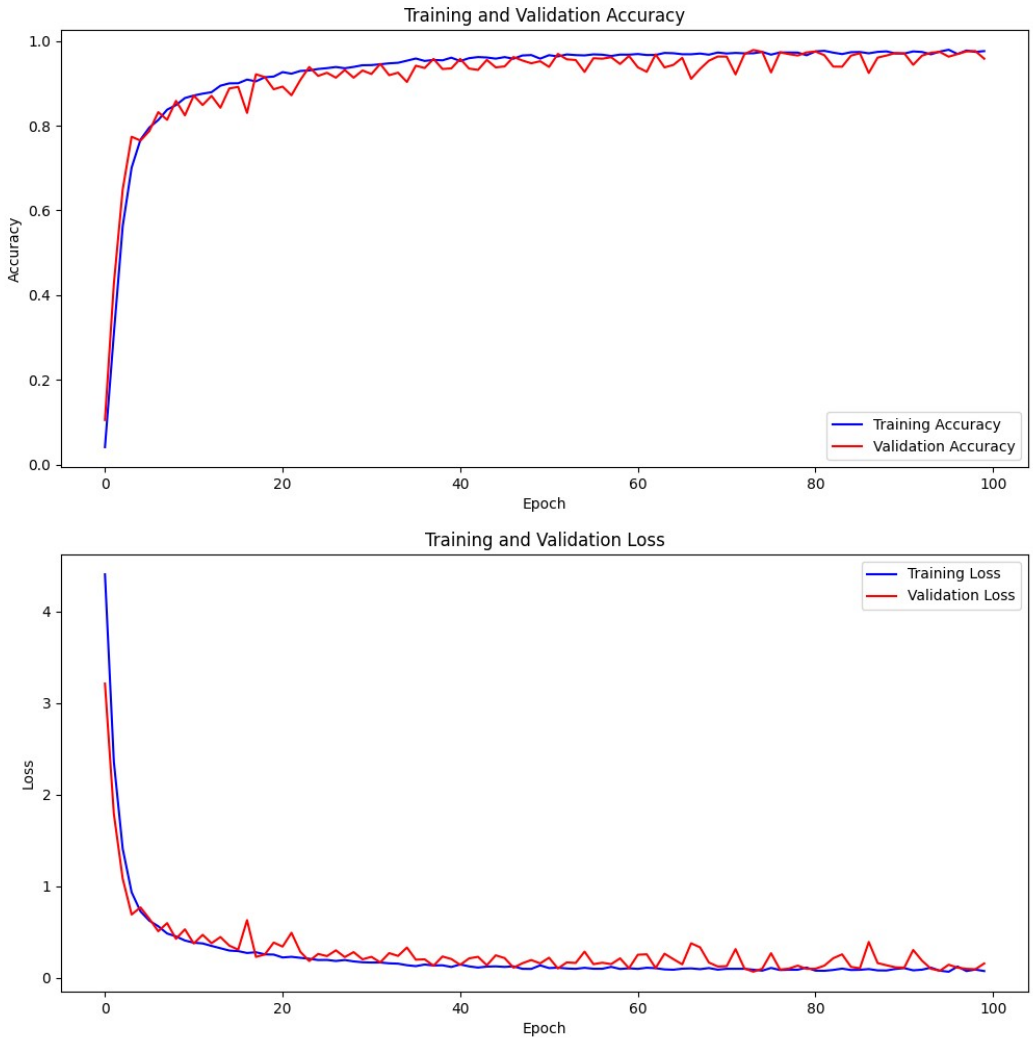

Figure 1: Training and Validation Accuracy/ Loss curves

With reference to Figure 1 we can deduce that the model has successfully learned representations and hierarchical features from the input card images during the training phase. The reported training loss of 0.0746 indicates that the model efficiently minimizes errors and discrepancies between predicted and actual values during the training process. The validation loss of 0.1576, slightly higher than the training loss, demonstrates the model's ability to generalize well to unseen data, indicating good performance.

Blackjack Gameplay:

I wrote a script called cozmo_as_player.py which sets up the gameplay with all the factors e.g., number of players, number of decks, making a shoe etc. The script then calls the appropriate model for card recognition where the capability of card recognition is required, and it also incorporates and invokes the model needed for decision-making which was developed in the previous paper. Finally, the script incorporates the function needed to display the card suit and card value on the LCD of cozmo (this script is discussed below). Therefore, the complete gameplay was successfully implemented.

LCD Display i.e., Cozmo's face:

I downloaded multiple png images for each suit and card value. I wrote two scripts the first was convert_png_to_bmp_resize.py which converts the png image to bmp, since cozmo cannot display in color, and resizes it to fit cozmo's face. The second script display_card_on_LCD.py uses a function to loop through all the images on cozmo's face to see if it displays all the combinations correctly. The function displaying the images was later incorporated into the cozmo_as_player.py (the gameplay script) to display the relevant suit and value in actual gameplay.

Results and Discussion:

As could be observed from the presentation on last Tuesday it was evident that cozmo was able to successfully identify the suit and value of certain cards but struggled on others. It sometimes got the suit right and at other times messed up the value. In particular, it was good at recognizing spades and clubs but misjudged diamonds and heats for one another. It also had problems with recognizing the difference between '8' and '9' and between 'J' and 'K'.

The gameplay script worked perfectly without any hiccups and so did the displaying function which accurately displayed the suit and value of the card following recognition.

I believe that I was able to complete both my baseline and stretch goals although there is room for improvement which I will discuss in the next section.

Limitations and future work:

I noticed that there were a few key limiting factors the primary one was Cozmo's camera resolution which is not high enough to capture all the intricate details that are found on a playing card which are a big factor in determining the suit and value of the card which is why cozmo was mixing up the suits sometimes.

I also noticed that lighting had a huge impact on recognition if the lighting was adequate then cozmo performed significantly better and even got both the suit and value right!

Also, the background contributed to the recognition sometimes having a white background helped cozmo recognize some cards better and I had almost a hundred photos of each card with white background from various angles.

Given the two points above I think cozmo's card recognition ability will be greatly enhanced if we can take more pictures to increase the size of our dataset. Moreover, in the future a rig could be made that automatically adjusts cosmos location and angle to completely automate the task of image generation.

Another future work could be to employ the use of two different ML models. One to recognize the card suit and the other to recognize the card value thereby reducing the burden on a single ML model.

Conclusion:

This project successfully enhanced Cozmo's capabilities by integrating a card recognition system for blackjack gameplay. It encompassed the development of an ML model to recognize cards, integration of the model into Cozmo's decision-making process for blackjack, and the display of recognized card values and suits on Cozmo's LCD. The methodology involved collecting card images, creating an ML model, and implementing it into a blackjack gameplay script. Evaluation of the ML model showcased promising learning capabilities and generalization to unseen data. During gameplay, Cozmo demonstrated the ability to recognize certain cards accurately while facing challenges with others.

Limitations included Cozmo's camera resolution affecting recognition accuracy, lighting impacting recognition performance, and background interference. Future work involves expanding the dataset for better recognition, potentially automating image collection, employing distinct ML models for suit and value recognition, and exploring methods to enhance Cozmo's card recognition abilities.

Implementation:

I am assuming that the user has a working Cozmo SDK installed in their machine if not they can refer to the Requirements.txt file and "pip install package_name" all the required packages and dependencies. Furthermore, detailed instructions are given in the README file on the github page which provides all the necessary step by step guide to make your cozmo play a game of blackjack flawlessly.

Checkout the master branch in the following Github link:

https://github.com/CozmoRobots/Ali_Hasany-repo

Credit:

I would like to give credit to Professor Denise (basically she provided me with the idea!) she has been one of the most supportive and passionate people I have met in life. She is always ready to help along with inspiring people like me to learn ML and this is what I feel makes the US education system standout. I am yet to see professors so dedicated to their work. Moreover, I would like to thank one of my classmates AKM who is an encyclopedia of Machine Learning, whose opinions were invaluable in tweaking the ML model.

References:

1. For card suit and value png images that were utilized to display on cozmo's LCD: https://images.google.com/
2. https://github.com/anki/cozmo-python-sdk
3. https://github.com/anki/cozmo-python-sdk/tree/master/examples

4. https://towardsdatascience.com/building-a-toy-detector-with-tensorflow-object-detection-api-63c0fdf2ac95