***Name**: Akansha Bhatia - **Student ID:** M00975443 – **Subject Code**: PDE4434*

***Demo Video Link:** https://youtu.be/_Hy-O0K-6BI*

***Public Git Hub:** https://github.com/Akansha-Robotics/IS-CW2*

Download for uno_colorG_model.keras:
https://drive.google.com/file/d/1zBiIp3FFiMFA7YPp5I3fae3EuUyTWF-s/view?usp=sharing

ASSESSMENT 2: COMPUTER VISION FOR UNO CARDS

Contents

Overview of Program

The program designed is used to conduct color and number recognition of UNO cards along with supporting code created to boost efficiency. To start off, for data collection, Create-dataset.py file is used to capture images to create the dataset and ensure that dataset is similar to frame collected in real time to make visual processing simple.

For color recognition, 2 main files are used which is Colour-ML.html and Color_RealTime.py to accomplish the task. Colour-ML.html holds a machine learning model that can conduct color predication of UNO cards from the images data set. From this code, a pre-trained model is downloaded called uno_colorG_model.keras and integrated with Color_RealTime.py to conduct real time recognition of color on UNO cards. Furthermore, Color_RealTime.py not used the pre-trained model but also has an internal function that check the most common hue of frame and corrects the predication accordingly to the hue range color set. Therefore, the color recognition utilizes an hybrid approach having the pre-trained ML model along with checking hue of frame.

For number recognition, the main file is Number_RealTime.py which conducts the recognition through template matching. The code starts with loading the image data set and creating templates to be matched with the video stream extracted in real time. Furthermore, the main function that processes the frame has 2 method of extracting ROI which is called normal processing and use_center_rule processing. The use_center_rule is quite similar to normal processing but differs when finding the centre in which is extracts it from the whole frame rather than the ROI in normal processing.

Create-dataset.py

Overview: The purpose was to simply automate the process of capturing the images of UNO cards. This python code enables the capturing of the UNO card images using the same method and expected image quality that would be present when conducting image recognition. Therefore, rather than capturing images from my phone or screenshots of webcam being broadcasted to my laptop that would have resulted in different image sizes and quality, it was more effective to create this simple code for data collection resulting in better performance.

capture_image is the main function that conducts the image capturing along with saving it in the right folder with name set by user. There are 2 prompts the user can do which is space or esc key. With pressing on space, the user can capture the image from webcam along with entering the name for file, through the terminal, that will be saved in images folder. Esc button press will simply allow the user to exit the program. The function also communicates these options when the code is first run along with printing further instructions to allow the user to name the file and confirmation that it is saved.

Colour-ML.html

Overview: The purpose was to create a machine learning model that can support in color recognition of UNO cards. The code starts with load_images_from_folder function that accesses the image data set to load images along with extract the labels to be prepared for model building and training. Next, find_roi_for_images work on extracting ROI and preparing the images loaded by adding the bounding box and standardizing the images. The cnn_model and transfer_learning_model function is created to build a hybrid model for conducting color predictions of UNO cards. Through an extensive trial and error process, the right layers are combined along with pre-trained VGG16 model being integrated to boost performance. The images are further processed through data

augmentation and labels being one-hot encoded along with data set being split to training and validation data set. Finally, the model is applied along with Adam optimizer and ReduceLROnPlateau to adjust learning rate according to performance. The final results are calculated of accuracy and loss along with confusion matrix displayed with Loss value over epochs graph.

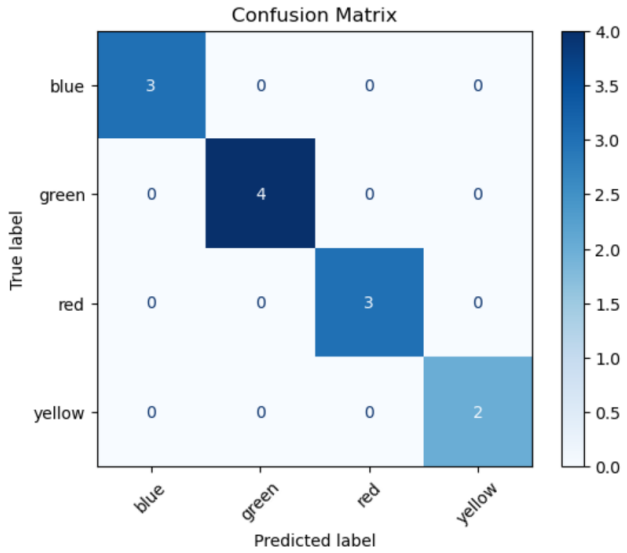
load_images_from_folder function stores the images with their color labels to be used for the machine learning model. To start off, it creates empty variables of images and labels so that images in the folder path is stored in it. Next, the if loop verifies if the image is in .png format to convert it to RGB format along with collect the label.

find_roi_for_images function finds and extracted the ROI of images so it can be used to have the machine learning model applied to it to make predictions. To start off, the function converts it to grayscale along with applying gaussian blur and adaptive thresholding to prepare image. Next, the contours are found, and bounding box calculated to capture the ROI. Finally, perspective transformation matrix is calculated so the image can be adjusted to ensure camera angle would not affect it along with ROI getting resize to be standardized.

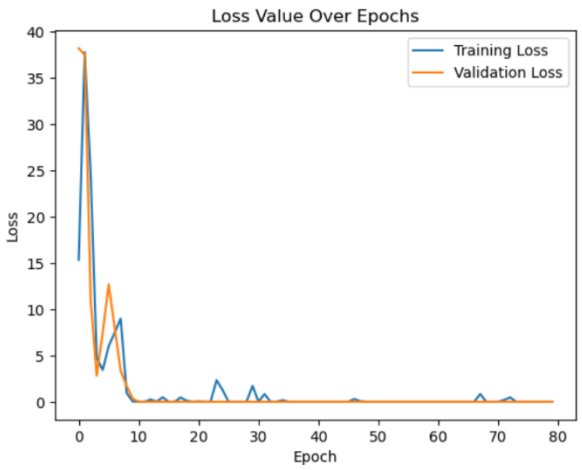
cnn_model function building the convolutional neural network (CNN) model to predict the color of UNO cards. The model starts 3 convolution layers which increase in size by each layer to extract features along with ReLU activation added to extract more complex patterns and relationships. However, to ensure overfitting does not occur, a max pooling layer is added after each convolution layer to reduce features extracted. Flatten layer is also added to convert them to 1D feature vectors and prepare them for dense layers below. Next, 2 dense layers are added to connect the features learned along with ReLU activation to handle complex patterns. Both these dense layers are followed by batch normalization layer, to standardized output, along with dropout layer, to randomly remove neurons, ensuring that overfitting does not occur. Finally, the last layer is an output layer which predicts the UNO card color along with softmax activation for probabilities.

transfer_learning_model function used the pre-trained VGG16 and combines it with additional layers to allow for the model to be adapted for our use case. The function starts with setting up the VGG16 as the base model along with freezing the training for the layers. Next, flatten layers is added to standardize the output while also a dense layer is added with relu activation to combine the features and learn complex pattern. Finally, the output layer is a dense layer with softmax activation allowing for the model to make prediction and activation function to calculate the probabilities.

```
1/1 [==============================] - 0s 441ms/step
Validation Accuracy: 1.0
<Figure size 800x800 with 0 Axes>
```



Results: The models have been applied with factor set to boost performance, such as Adam optimizer which adjust the learning rate during the training along with ReduceLROnPlateau which reduces the learning based on the performance of validation loss. As a result, the model was able to accurately predict all 4 color classes of the UNO cards.

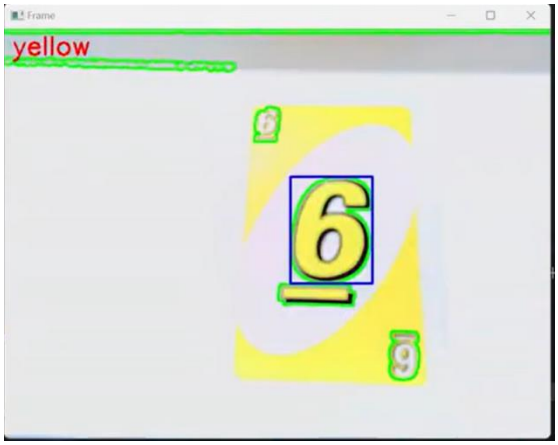


In addition, both the training and validation accuracy is at 100% while both loss metrics are low value and nearly. Furthermore, in Loss value over epochs graph displayed, it can be observed that the loss starts off extremely high and decreases when reaching 10 epochs along with stabilizing at a low value for the rest of the epochs. Therefore, this model is well suited to conduct prediction and can be integrated with Color_RealTime.py.

```
Epoch 80/80
1/1 [==============================] - 1s 1s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.
0000e+00 - val_accuracy: 1.0000 - lr: 1.0000e-04
```

Color_RealTime.py

Overview: The purpose is to conduct real time color recognition for UNO cards. The code uses a pre-trained model (Color-ML.ipynb) along with hue detection to guide it towards an accurate predication. The flow of program starts with finding ROI and contours with process_frame function which conducts basic steps, such as converting to gray scale and gaussian blur, along with techniques to boost effectiveness of finding ROI, such as dilation and adaptive thresholding. The next step would be conducting color prediction with check_prediction_confidence function which starts with applying the pre-trained model to get the first prediction. Once that it is received, it conducts a verification by converting the image to HSV and checking the dominant hue. This dominant hue is matched with hue range values set in the function to make final color prediction. In addition, get_most_common_hue functions help to get the most common and dominant hue supporting the check_prediction_confidence function to do verification.



process_frame function processes the frame to find the ROI of the UNO card. It starts with converting to gray along with gaussian blurring which are the basic steps for finding the contour. Next, adaptive thresholding and dilation is conducted to boost performance of finding ROI as it helps in clearly identifying hard to capture ROI. One key learning point was found when I was facing an issue that yellow and blue color were not being defected which was due to the ROI not being captured properly. Once dilation was added, it boosted performance and resulted in accurate color predication. The final part would be detecting the contours, which is set to find external and edge contours, along with drawing these contours on frame allowing for user to understand what is being processed.

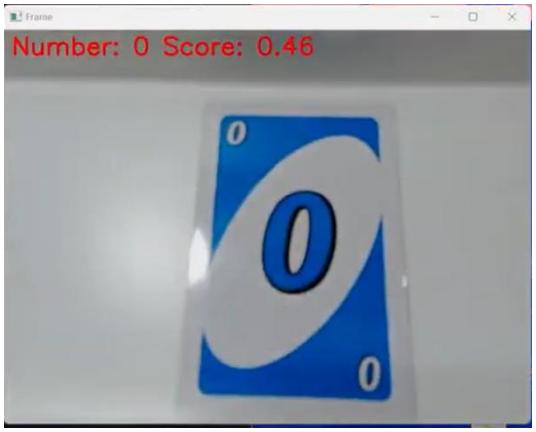
get_most_common_hue function is a simply one for finding the dominant hue by either calculating the most common hue using mode or getting the last hue. This function supports the

check_prediction_confidence function by providing the dominant hue so that it can be matched to hue range in check_prediction_confidence function.

check_prediction_confidence function conducts the color predication based on the pre-trained model and matching dominant hue to hue range. It starts with resizing and preparing the image to allow the model to be applied on it to provide prediction. Once the predication is made by pre-trained model, it is stored in prediction_index variable along with confidence level for that predication is stored in confidence variable. Next, function prepared the image to match dominant hue by converting image to HSV along with calculating the histogram. In addition, the hue range for color are also defined in the function so it can be compared to and guide the predication. The final section of this function is the if loop that checks if the common hue matches the hue range of that color along with whether the predication is for that color. According to these conditions, it will correct the prediction to provide an accurate one. For example, if the common hue is in the blue hue range but the prediction by pre-trained model is not blue, then the function will correct it to blue. However, if the common hue range is in blue and prediction by pre-trained model is blue, then it will just keep the prediction by pre-trained model as it is accurate. One key point is that the yellow hue ranges and if loop condition is not set out in this function. This action is done on purpose as when I had set out the yellow hue range it was not predicating yellow accurately and even affecting prediction of other color despite trying multiple ranges. However, when the yellow hue range and if loop condition was removed, the color predication improved, as the pre-trained model was able to accurately predict yellow color and did not require verification with hue range.

Number_RealTime.py

Overview: The purpose is to conduct real time number recognition for the UNO cards. The flow of the program starts with process_frame function is extracting the ROI of video stream frame along with choosing to apply normal processing or use_center_rule processing. For normal processing, it will apply the steps converting to grayscale and thresholding along with finding the contour and center of the ROI. On the other hand, use_center_rule does the similar processing but instead take the center from the whole frame rather than from the ROI. This option was applied to overcome the issue with recognizing 6 and 9 number. To activate the use_center_rule processing, user has to press “c” button to indicate to the function to change processing type along with the function also displaying a message for user on terminal that this has been activated. Furthermore, process_frame function also uses process_extracted_roi function to enhance the extract ROI and get a clear image. Finally, the functions called load_templates and match_template are used to conduct template matching to make prediction of UNO card number in which load_templates creates and stores the templated based on the image dataset while match_template matches the loaded templated to ROI frame to make predication.



process_frame function processes the captured video frame to find the ROI which hold the UNO card number. The flow of the process depends on whether normal processing is or use_center_rule is selected in which the main difference is the method of calculating the center. The function starts with converting the frame into grayscale along with thresholding the image to highlight the UNO card

numbers. Furthermore, dilation is also conducted to enhance the numbers allowing for the contour to find the boundaries. Finally, the closest contour to center is extracted as the ROI along with process_extracted_roi function is applied to better capture the UNO card number.

A key challenge faced with number recognition was that number 6 and 9 were predicted as 0 due to the ROI extraction being too zoomed in in finding the UN O card number. However, when the code was adjusted to take a larger area, 6 and 9 were being predicated correctly but other number recognition was affected. Therefore, to solve this issue, process_frame function was divided to add use_center_rule which allowed for the function to take a larger area when it came to recognition of 6 and 9 number.

process_extracted_roi function conducts further extraction to better enhance the feature for UNO card number recognition. The function starts with setting a kernel size to 3 by 3 along with applying morphological closing to make the number structure more prominent by closing small gaps and holes. In addition, to further reduce the noise, erosion is conducted to remove abnormal or isolated pixels to clean up edges. However, application of erosion also shrinks the number, so dilation is applied to add pixels making the UNO card number more apparent.

load_templates function is accessing the dataset images for UNO cards and creates templates for the corresponding number on UNO card to be prepared for the match_template function. The function starts with finding the file with .png format along with splitting the name of the file to collect the UNO card number. Next, the image is stored as a template along with the associated UNO card number inside a dictionary.

match_template function used the loaded templates and compares them with the extracted ROI of real time processed frame to make a prediction of the UNO card number. The function starts with converting to grayscale to ensure that it is compatible and can matched to the loaded templates which are all grayscale formats. Next, template matching is conducted in which it calculates how well the ROI matches using cross-correlation coefficient method through TM_CCOEFF_NORMED. This method slides the template over the processed ROI frame to calculate correlation and find the best match.