COMPUTER VISION FINAL PROJECT “BANANA DETECTION”

Burla Nur Korkmaz

ICT For Internet and Multimedia

I. Introduction

The project aim is to develop software to detect each banana on images. There are 3 scripts that are named “CropTrainImages.cpp” , “training.py”, and “detection.py” in the project. “CropTrainImages.cpp” is developed with c++ and it is for crop the training images into 32x32 grids from the banana training dataset and label them. After cropping and labeling each cropped image, the script creates the “GridsFolder” folder, saves all the cropped images under the “GridsFolder” and saves all the labels as a .csv file into the current directory which is “BurlaNurKorkmaz_LAB5”. “training.py” is used to create neural network model and “detection.py” is used for detecting bananas.

```
\BurlaNurKorkmaz_LAB5
├───banana-detection
│       ├───bananas_test
│       │       ├───images
│       │       └───label.csv
│       └───bananas_train
│               ├───images
│               └───label.csv
├───GridsFolder -->(created after running CropTrainImages.cpp)
│
├───CropTrainImages.cpp
│
├───training.py
│
├───detection.py
│
├───gridslabel.csv -->(created after running CropTrainImages.cpp)
│
├───model.h5 -->(created after running training.py)
```

Fig.1. Project directory structure

II.Project Details

First of all, the “CropTrainImages.cpp” file must be run to crop train images and label them. The second “training.py” file must be run to train the neural network model with cropped train images and their labels. This file also saves the neural network model after training. Lastly, “detection.py” must be run to detect bananas on the test dataset.

II.I. How does “CropTrainImages.cpp” work

This script is written with c++ to crop train images into 32x32 grids and label each grid as 0 and 1 depending on they are banana or not. 0 means NOT BANANA while 1 means BANANA. This labelling depends on the banana coordinates of training images. There is a comparison between the banana coordinates of training images and cropped images to determine if the image is a banana or not banana. It means that if at least %25 of the cropped image intersect with the banana coordinates of training image, this cropped image is labelled as 1 which means BANANA.

- Reads “label.csv” under the “banana-detection/bananas_train”
- Calculates grids coordinates for future calculations
- Creates "gridslabel.csv" file
- Crops train images into 32x32 grids
- Create “GridsFolder” folder
- Saves the cropped images(32x32 grids) in “GridsFolder”
- Compares the coordinates in label.csv with grids coordinates
- Assign label each grids(cropped images), if at least %25 of the cropped images intersect with the banana coordinates of the training image, the label of cropped images are assigned as BANANA. Else the label of cropped images are assigned as NOT BANANA.
- Saves the labels of the cropped images with the names of them into "gridslabel.csv"
- Ends the program

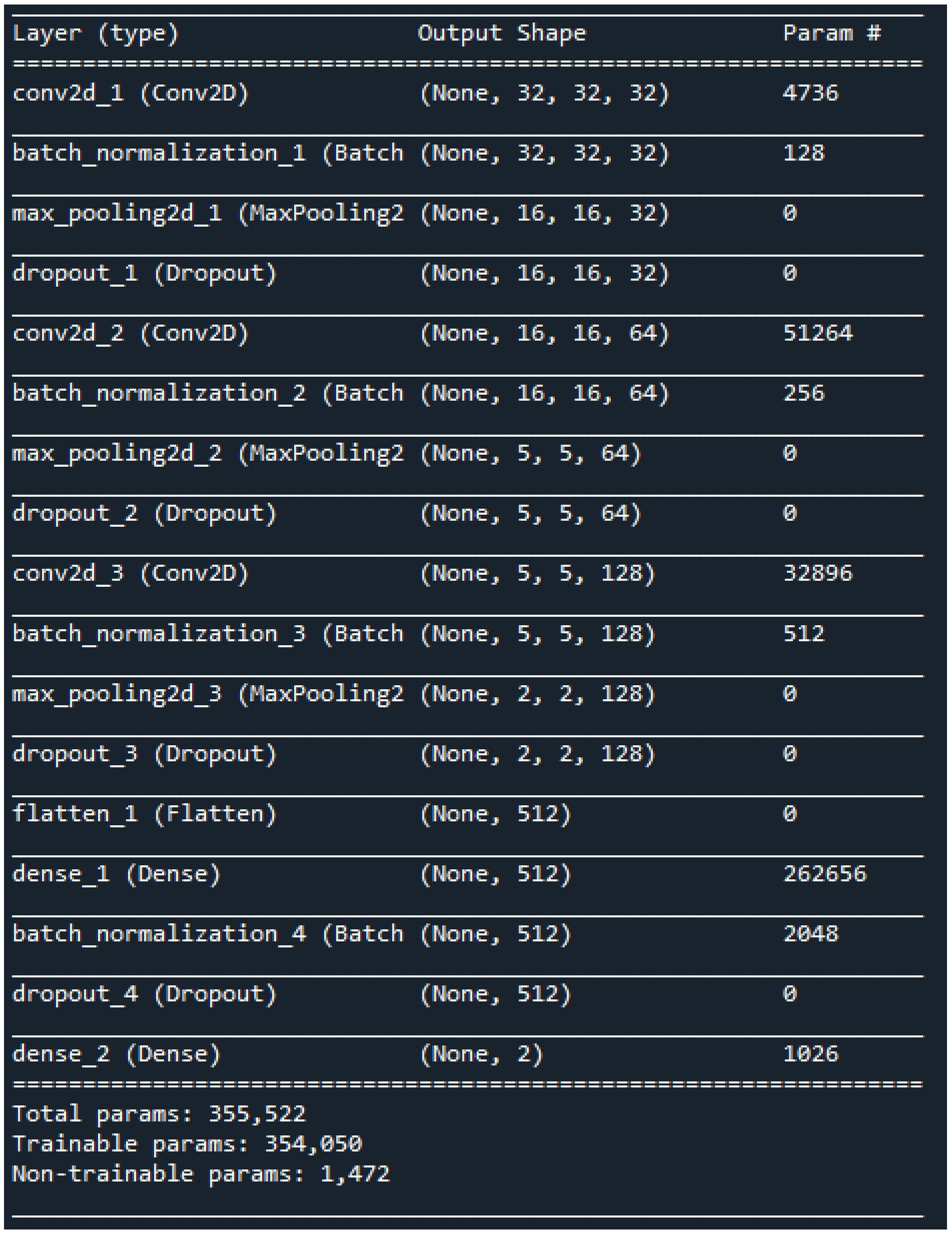
II.II. How does “training.py” work

This script is written by Python to build a neural network model and train it with labeled cropped images.

- Reads the “gridslabel.csv”
- Reads the cropped images from “GridsFolder”
- Merges the labels and the images into a list
- Shuffles the list
- Calculates the amount of NOT BANANA and BANANA
- Checks if there are more NOT BANANA than BANANA
- If there are more NOT BANANA than BANANA, deletes NOT BANANAS from the list until the amount of NOT BANANA and BANANA becomes equal.
- Then separates the list into train_label and train_images
- Turns train_label and train_images into arrays
- Creates the neural network model
- Trains the model
- Saves the trained model

II.II.I. Model Architecture

The model architecture(*Fig.2*) includes three convolutional neural network layer with kernel size (7,7), (5,5), (2,2) respectively, three batch normalization layer, 3 max pooling layer with pool size (2,2), three dropout layer with the 0.25 fraction of the input units to drop, one dense layer with 512 neurons and one dense layer as an output layer with softmax activation function which converts a vector of numbers into a vector of probabilities. This feature will be used to determine which image is a better banana than other banana images. As an optimizer, Adam optimization algorithm is chosen with 0.001 learning rate. As a loss function, categorical crossentropy is chosen. Model has been trained 30 epochs with batch size 64.



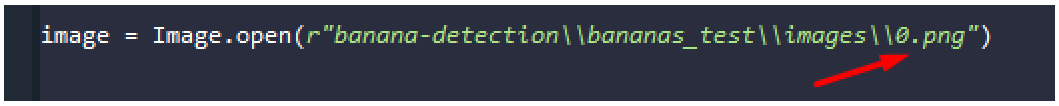
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        4736      
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        51264     
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 5, 128)         32896     
_________________________________________________________________
batch_normalization_3 (Batch (None, 5, 5, 128)         512       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656    
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048      
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026      
=================================================================
Total params: 355,522
Trainable params: 354,050
Non-trainable params: 1,472
_________________________________________________________________
```

Fig.2. Model summary

II.III. How does “detection.py” work

This script is written by Python to detect each banana on the test dataset. It crops 10000 images from a given test image randomly and assigns cropped images into a list. Then the trained model predicts on 10000 cropped images and stores predictions result in a list. Later, the prediction that is higher BANANA score is selected and the features of the image with the highest BANANA score are assigned to new variables to be used in future calculations. In short, generate 10000 32x32 images randomly from a given 256x256 test image and find the randomly generated image that has the highest banana score.

The starting coordinates of the best banana is used to crop new randomly generated images that is around the best banana. The variable that is called “number” is used to determine how far pixels from the starting point of the best banana will be selected to generate random images. After cropping 100 new random images that are close to the best banana, again prediction is done to eliminate not banana images among them. For the final step starting point of all new cropped images are merged to determine the bounding box of the banana.



```
image = Image.open(r"banana-detection\\bananas_test\\images\\0.png")
```

Fig.3. Banana test images path

As shown on the *Fig.3.* , in order to detect banana on different test images, image name must be changed manually.

“number”, the count of randomly cropped images for both the first step and second step can be change to get different results.

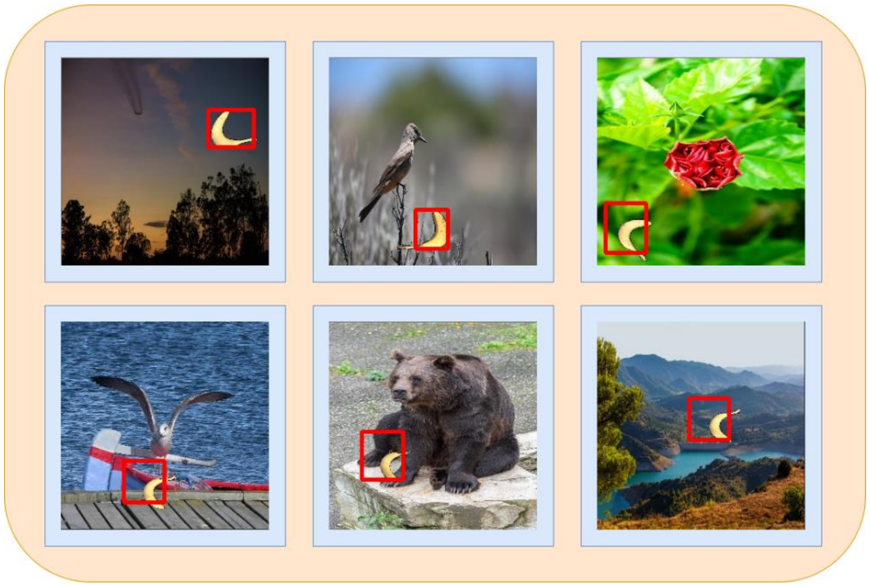


Fig.4. banana detection with different test images. “number” is taken as 32.

III. Conclusion

In order to get better results, the deep neural network model could be fine-tuned, hyperparameters could be changed. In addition to the fine-tuning of the model, the parameters which randomly crop images and the variable that is called “number” is used to determine how far pixels from the starting point of the best banana on the “detection.py” could be changed. Also, more elimination steps can be used to generate better banana images before merging them. Furthermore, each pixel of the images can be analyzed and a different elimination can be done by depending of the colors of the pixels.