Final Project

Deep learning in computer vision

Nada Labib	40-0876
Nadine Moustafa	40-8087
Youssr Abou-Youssif	40-9369
Yomna El-Shazly	40-2724
Aya Bedwi	40-0731

➢ **Overview:**

In this project, we choose vgg16 deep learning model to perform the proposed topic which is CNN for action recognition. It recognizes different actions from multiple images that have been taken from a video sequence. So that, it is an image classification task by making the classification labels are the different actions that people perform.

➢ **Input:**

The input to the model is sequence of images for each action. In this project we have 11 different actions that can be classified from images which are:

- Driving a car
- Sleeping
- Dancing
- Walking
- Sitting and talking
- Going downwards
- Swimming
- Photography
- Camping
- Driving a Motor
- Order food

These 11 different actions are recognized from a video film that have been taken its frames for each second in the film corresponds to 25 frames/ images. This film results in 160,000 image and then we choose the most accurate images for each action that describes this action specifically and finally we take 250 different images for each action or class and start to create this dataset.

This have been done using python code that, initially it reads the video file then, looping over the video frames until they are done taking each frame and save it as jpg file so, we have 160,000 different images for this film and then start taking 250 images for each action or class in order to create this dataset results in 2750 different image in total.

➢ Model:

To create the model, we must first organize the dataset that have been gathered through the video film.

- **Create_dataset:**

  In this file we create a csv file with number of rows equals to the number of images so, it includes 2750 row. Then, for each 250 rows we give them a specific label corresponds to the image action type, so we give it a string label describes the action. Moreover, a number also indicates a label for this specific action.

  This have been done using 11 loops that are looping in the specific positions corresponding to each action type in order to maintain its correct labels and save all this information in a csv file.

- **Classifier:**

  In this file it includes the model itself and the training phase of the model over the dataset created and then start to discover the accuracy and test the model over different action types.

  The model used was vgg16 model to test the action recognition from multiple images.

➢ VGG16 Model:

VGG16 is a convolution neural net (CNN) architecture which was used to win (ImageNet) competition. Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2.

o Classifier:

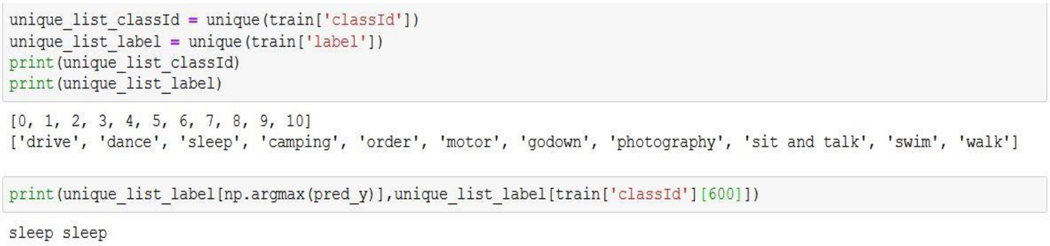
1. Starting by importing the needed libraries to implement vgg16 model which are sequential which means that all the layers of the model will be arranged in sequence. Moreover, importing different layers needed which are: Dense, convolition2D, maxpooling2D, averagepooling2D, zeropadding2D, dropout, flatten, reshape, merge and activation.
2. Then looping over the csv file created to take the labels and save them together with the images in a train folder for the sake of the model to train over these images.
3. Method that creates vgg16 model which starts by initializing the model by make it pretrained over ImageNet and include the 3 fully connected layers on the top of the network. Then, getting the output of the model layers.
4. Then, use the softmax activation function over the model output layers.
5. Then, setting the first 8 layers of the model to non-trainable which means that the weights are not updated.
6. Using the SGD “Stochastic gradient descent” optimizer which is an iterative method to optimize objective functions.

7. Using categorical crossentropy is a loss function an example can only belong to one out of many possible categories, and the model must decide which one.
8. Then start training the model over 3 channels, 16 batch size and 10 epochs.
9. Do the fine tuning, making the predictions and calculate the loss entropy score.

- **Testing:**

Predict the image that have been used for testing to predict the labels. Then, having a function to retrieve the unique values.

➢ **Output:**

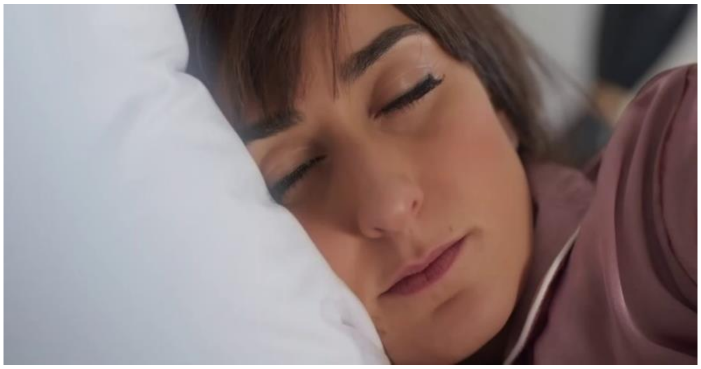


```
unique_list_classId = unique(train['classId'])
unique_list_label = unique(train['label'])
print(unique_list_classId)
print(unique_list_label)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
['drive', 'dance', 'sleep', 'camping', 'order', 'motor', 'godown', 'photography', 'sit and talk', 'swim', 'walk']
```

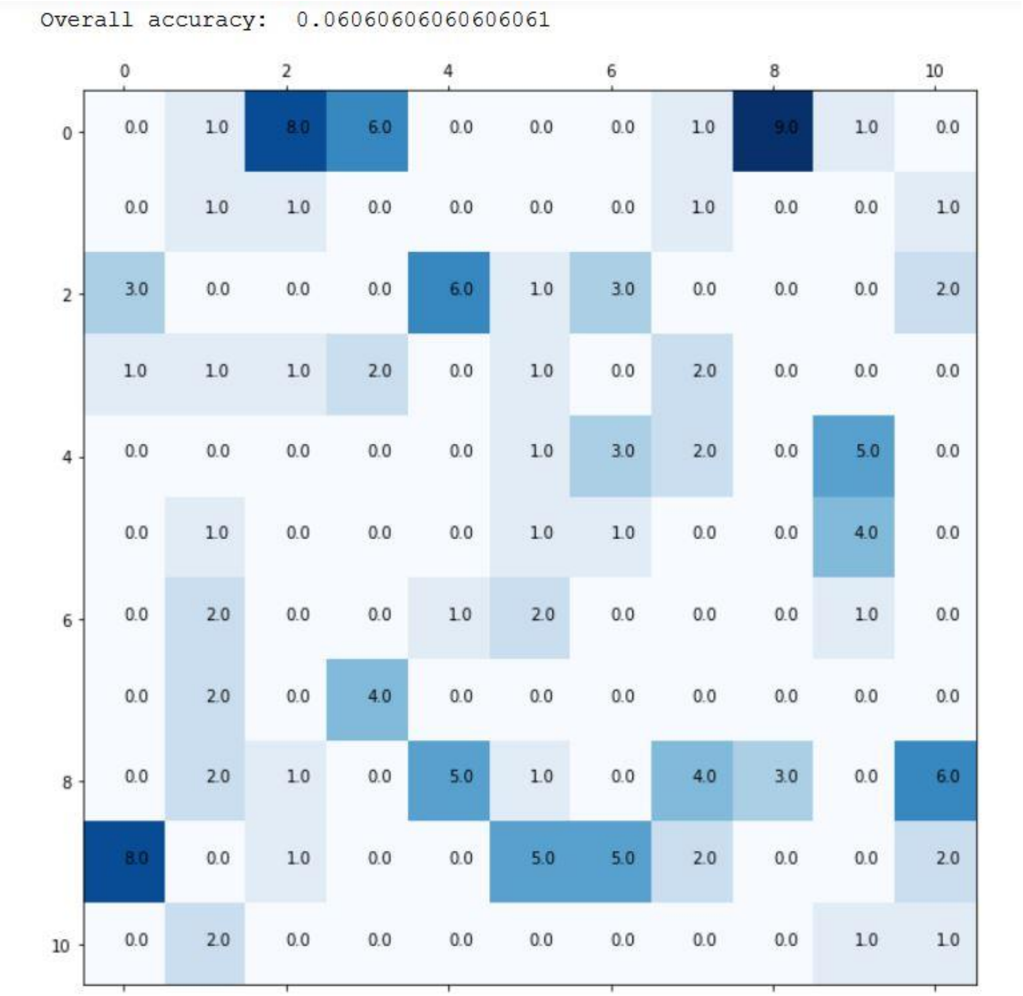
```
print(unique_list_label[np.argmax(pred_y)],unique_list_label[train['classId'][600]])
```

```
sleep sleep
```



Here is a test image of sleeping action and the model detects that the action is sleeping.

- Here is the confusion matrix over the test images together with their predicted values and actual values.



```
Overall accuracy:  0.06060606060606061
```

	0		2		4		6		8		10
0	0.0	1.0	8.0	6.0	0.0	0.0	0.0	1.0	9.0	1.0	0.0
	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	3.0	0.0	0.0	0.0	6.0	1.0	3.0	0.0	0.0	0.0	2.0
	1.0	1.0	1.0	2.0	0.0	1.0	0.0	2.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	1.0	3.0	2.0	0.0	5.0	0.0
	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	4.0	0.0
6	0.0	2.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0
	0.0	2.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	2.0	1.0	0.0	5.0	1.0	0.0	4.0	3.0	0.0	6.0
	8.0	0.0	1.0	0.0	0.0	5.0	5.0	2.0	0.0	0.0	2.0
10	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

<u>Note:</u>

If the model is going to be trained again, it is going to take few hours (3-5 hours) depending on the processor capabilities.