Distracted Driver Detection

Using Computer vision to classify the behavior of the driver

By-
Shivansh Singla
UIET

Explaining the problem

We are given driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc).

Our goal is to predict the likelihood of what the driver is doing in each picture.

We will be given 10 different classes in which we can classify the activity being done by the driver.

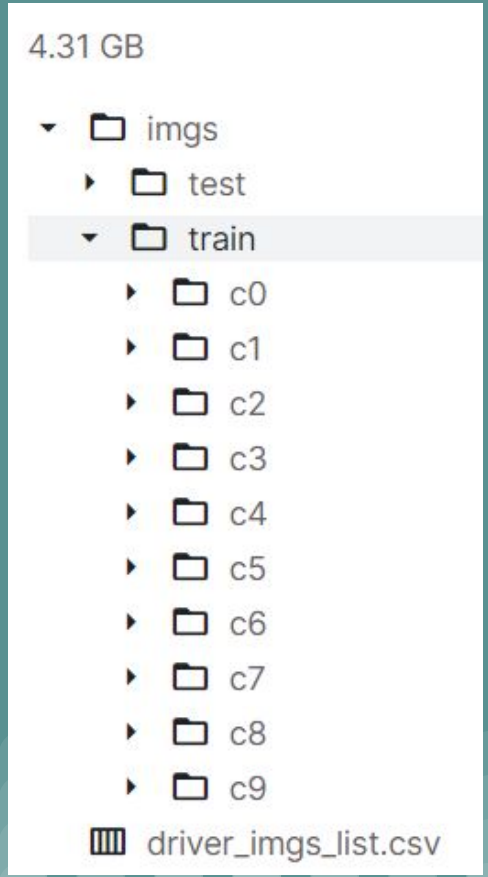
The 10 classes to predict are:

- c0: safe driving
- c1: texting - right
- c2: talking on the phone - right
- c3: texting - left
- c4: talking on the phone - left
- c5: operating the radio
- c6: drinking
- c7: reaching behind
- c8: hair and makeup
- c9: talking to passenger

Dataset description

Our dataset will have one folder and one csv file

1. Folder called ‘imgs’ contain 2 directories - train and test.
    a. Train contains 10 directories each labelled by class i.e c0,c1,c2,... where each directory contains the images of the class given by directory label
    b. Total images in train set are 22424 where each class contributes approximately same number of images
    c. Test folder contains approx 80000 unlabelled images on which we have to make predictions
2. The csv is driver_imgs_list.csv It contains data about train images like which driver_id is corresponding to which image as there are many photos of the same driver in the dataset



Grading and Submission details

As the predictions will be graded on kaggle we need to make our predictions submittable in a csv format which are in the format defined by kaggle.

Submissions are evaluated using the multi-class logarithmic loss which is also called **Categorical cross entropy**. Each image has been labeled with one true class.

For each image, you must submit a set of predicted probabilities (one for every image).

Function for grading

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Submission format ->

The order of the rows does not matter. The file must have a header and should look like the following:

```
img,c0,c1,c2,c3,c4,c5,c6,c7,c8,c9
img_0.jpg,1,0,0,0,0,...,0
img_1.jpg,0.3,0.1,0.6,0,...,0
...
```

How to download the dataset

We will be using Kaggle API to download the dataset

1. Install kaggle package in python using !pip install kaggle.You can check if its already present using !pip show kaggle.
2. Upload the kaggle.json file which contains the api key to root directory
   `~/.kaggle/kaggle.json`
3. Use !kaggle datasets download -d ashkhagan/brain-tumor-dataset-with-saliency to download it in zip format

Loading/Analysing/Cleaning the data

1. The dataset is in zip folder so first we need to unzip the dataset which can simply be done by
   a. `!unzip /content/brain-tumor-dataset-with-saliency.zip`
2. So now we will load the driver_imgs_list.csv using pandas and describe it
   a. `driver_index = pd.read_csv('driver_imgs_list.csv')`
   b. `print(driver_index.describe(),end ='\n\n')`
3. We get to know that there are only 26 drivers which contribute to total 22424 images
4. We also realize most of the classes have nearly equal number of images and each subject also has classes divided nearly equally b/w their actions
5. So our model won't be biased towards a certain class
6. We will pick the train(X_train1,Y_train1) and validation test(X_train2,Y_train2) subjects alternatively starting from the most frequently occurring subject to the least occurring subject
7. Reasons for using this instead of simple train_test_split will be explained in later slides
8. We get names of every image belonging to each unique (subject,class) pair using two loops and dataframe slicing
9. After preprocessing input(reason later),We load the image and class to
   a. X_train1,Y_train1 if index is even
   b. X_train2,Y_train2 if index is odd
10. All the subjects are sorted by number of images belonging to them
11. We convert these lists to np array because most functions of tensorflow require np arrays
12. We one hot encode the outputs

Image preprocessing

1. Initially sizes of images is 480*640*3 which we change to 64*86*3(maintaining the aspect ratio) because images are so much in number that they can't be loaded in high resolution. I even had to make predictions in running instead of making an X_test then predicting at once because testing data has nearly 80000 images.
2. We also preprocess inputs using
    a. `from tensorflow.keras.applications.resnet import preprocess_input`
    b. Because Pretrained models have specific input conditions which need to be fulfilled for good results
3. The model also has a cropping layer to crop some part of image which doesn't contribute anything to the activity which we are trying to detect

Reason for the specific type of splitting

1. A big challenge in action recognition in still images is the lack of large enough datasets, which is problematic for training deep Convolutional Neural Networks (CNNs) due to the overfitting issue.
2. A simple train test split in proportion will just overfit the model and validation scores will also be good as subjects are same in both training and validation set.Also,because
    a. Instead of Learning the activity being performed by the driver
    b. It learns If a specific driver is doing this then activity being done is. So In Predictions it looks for cues similar to a particular set(train) of drivers performing actions instead of actions themselves
3. The only reason to crop was to focus more on the activity
4. THIS WAS A VERY IMPORTANT THING I LEARNED
5. Most of the notebooks you find on kaggle just simply use train test split to get fake impressive validation scores and never show their submission score
6. When i started out I was also getting amazing validation scores and bad submission scores took me a long time to figure out the reason because every notebook contained False results. I only realized it after analysing the data

Some improvements in data that can be made to improve training Process

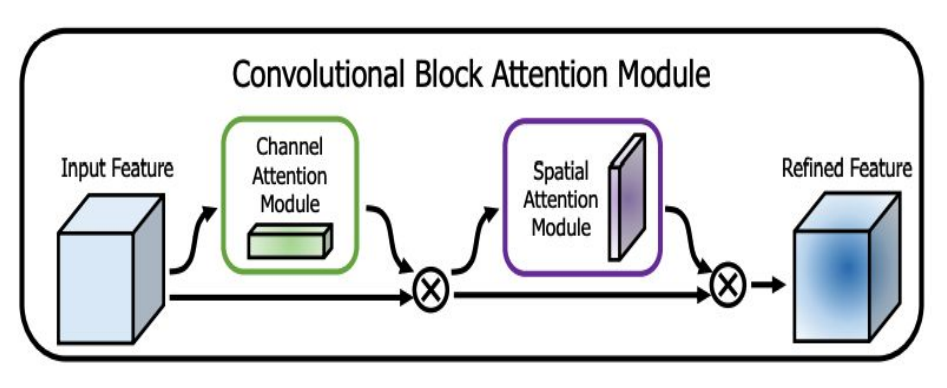
1. If Ram available is sufficient enough we can resize the image to higher resolutions like 256*256*3 which will help in better training
2. We can directly save the images in memory then predict at once instead of Running.The former method I found out is at least 7 to 8 times faster when i was working with smaller images(32*32*3) for testing the model.
3. Data Augmentation can be applied for better training and preventing overfitting to some extent
4. Hyper parameter tuning can also be applied
5. Ensemble learning is also possible
6. THIS MADE ME REALIZE GOOD RESULTS START AT DATA MANIPULATION AND NOT JUST AT THE MODEL ARCHITECTURE

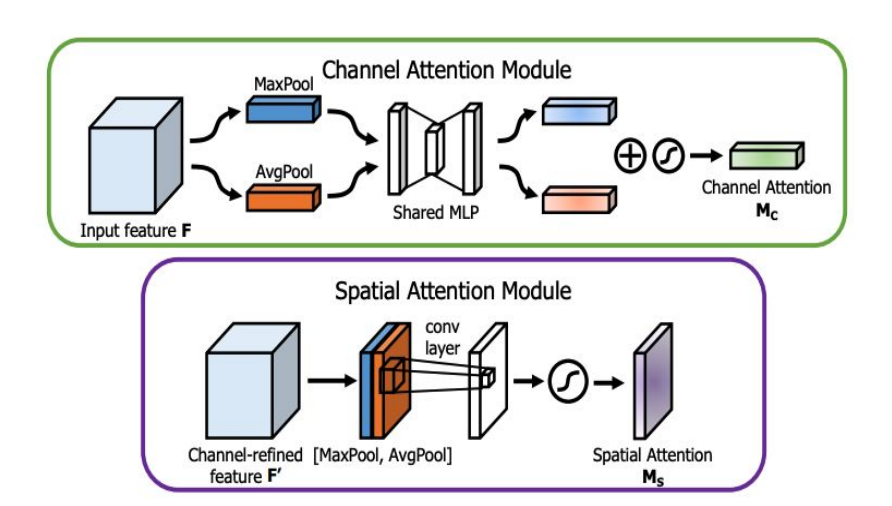
Transfer Learning

1. I will be using Resnet50 model as the pretrained model without its top layer which is used for classification to get some information about the images on which we work
2. Output of Resnet50 without top layer is Height * channels * width i.e data format is “channels last”
3. We have kept the pooling as none because that will be done by the CBAM block
4. Resnet50 model from keras need to have a specific input format which can be obtained by preprocessing images with help of the module https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/preprocess_input
5. We have to make this layer not trainable

Convolutional Block attention Module Architecture

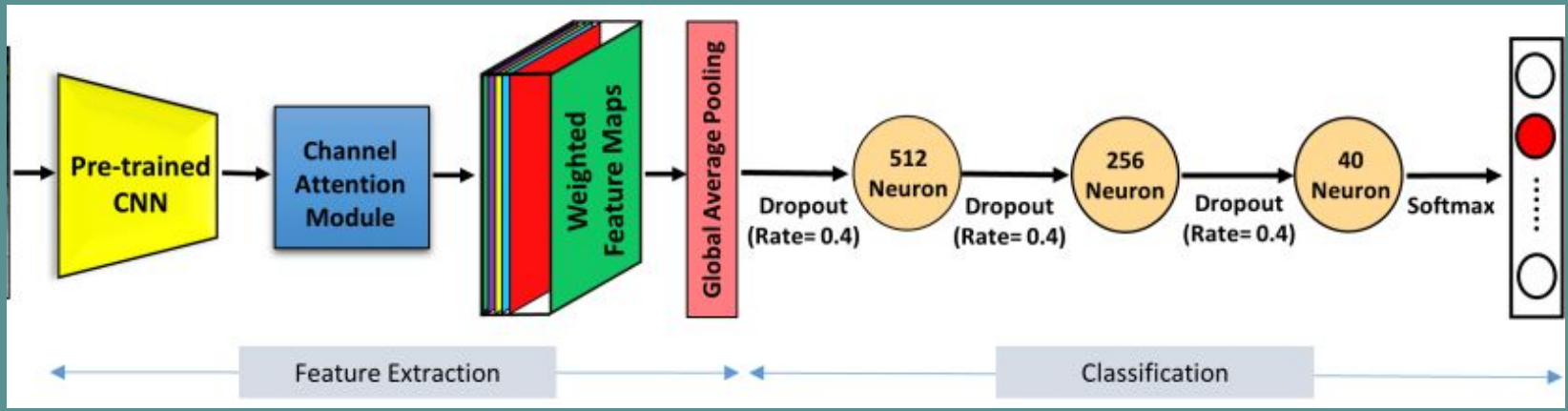
1. CBAM is an attention module which is used to make CNN learn and focus more on the important information, rather than learning non-useful background information. In the case of object detection, useful information is the objects or target class crop that we want to classify and localize in an image.
2. https://medium.com/visionwizard/understanding-attention-modules-cbam-and-bam-a-quick-read-ca8678d1c671 for explanation





$$\mathbf{F}' = \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F},$$
$$\mathbf{F}'' = \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}',$$

Final Neural network



SUBMISSION SCORES

1. The best score and the latest score is the one which was obtained with Transfer Learning + Attention Mechanism
2. The second most recent score was the one obtained when pretrained model was trained from scratch(option 2 model in the notebook)
3. The score 30 was a blunder I forgot to preprocess input
4. All the other ones were some brute force which i was trying on transfer learning by changing image sizes and epochs,etc.

fileName	date	description	status	publicScore	privateScore
csv_for_submission.csv	2022-08-10 17:18:32	submission from command line	complete	1.83507	1.79899
csv_for_submission.csv	2022-08-07 05:53:10	submission from command line	complete	4.14835	3.78386
csv_for_submission.csv	2022-08-06 19:18:52	submission from command line	complete	30.94361	31.06573
csv_for_submission.csv	2022-08-06 18:26:21	submission from command line	complete	5.72365	5.59883
csv_for_submission.csv	2022-08-06 12:27:35	submission from command line	complete	3.14899	3.26170
csv_for_submission.csv	2022-08-06 05:19:11	submission from command line	complete	3.88205	4.11758
csv_for_submission.csv	2022-08-05 16:16:44	submission from command line	complete	4.56226	4.57116
csv_for_submission.csv	2022-08-05 15:45:04	submission from command line	complete	4.17788	4.47560
csv_for_submission.csv	2022-08-05 15:21:33	submission from command line	complete	3.02924	3.30787
csv_for_submission.csv	2022-08-05 13:36:45	submission from command line	complete	6.59237	6.22998