Brand Recognition using Deep Learning

By:

Ahmad Nadeem Saigol

Contents

Dataset:

The dataset that was provided consisted of approximately 1000 videos of various brands and products. The way we have set up the problem, it required the dataset to be in the following structure:

```
Main Folder → brand 1 → video 1
                      → video 2
            → brand 2 → video 1
                      → video 2
               & so on.
```

Thus, the dataset was changed to meet this configuration. Also, our model required to have at least two videos for each brand during training phase. Thus, all those brands were excluded from the dataset that did not meet this criterion. Furthermore, all videos were different formats and for some reason OpenCV was not able to read the .mp4 files. Thus, to cater for it a script was written (code provided) to convert all files to '.avi' format which was compatible with OpenCV. This also insured the consistency of formats in all videos.
As the number of the videos were large, a custom class video data generator was written using Keras API. The API required to implement __len__ and __getitem__ method which returns the steps to take and batch of the videos, respectively. Apart from that, some helper functions were included. In summary, it reads the frame by frame, converts it into RGB color space, resize it to specified shape (112,112), maps the pixel values to [0,1] and stores only specified number frames of video (NBFRAME = 1000). These things are done to ensure faster computation and manage limited available memory. Also, for each batch the labels were converted to categorical values since (positive, anchor, negative) values were important. We will talk about this more in the following sections. Also, the dataset was split in ratio (60,20,20).

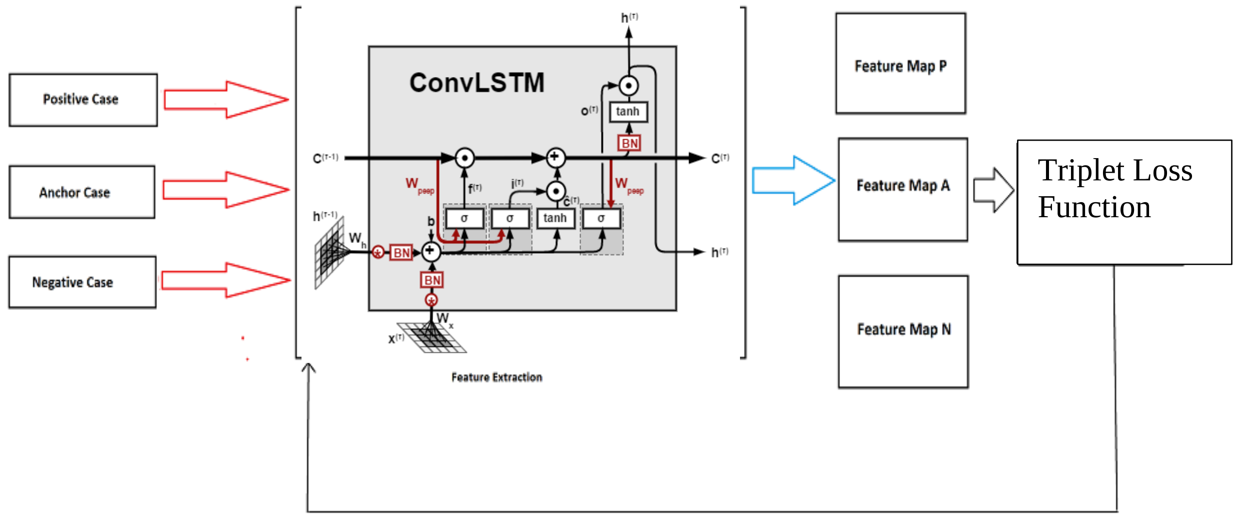
Model:

The problem statement was as such that we had to cater for the unknown brand at the inference time. This made the modelling a little tricky and we could not just take it as a classification problem. Inspired by FaceNet[1], we employed a similar approach for the brand recognition. Our model tries to learn how to generate feature vector for a brand i.e it maps a given video to 256 x 1 feature vector. The model is as follows:

1. Since advertisements have temporal information associated with them, we had to incorporate that. To do this, we are using ConvLSTM2D layer which is basically a LSTM but uses convolutional operation for both input and recurrent transformations:
    a. Number of Filters; 64 (with bias)
    b. Kernel_size: 3 x 3
    c. Stride: 1
    d. Padding: no
    e. Activation: tangent hyperbolic
    f. Recurrent Activation: sigmoid
    g. Input shape: (batch, seq, width, height, channel)
2. The output of the previous layer is flattened
3. A Dense layer follows it which has 256 neurons with no activation
4. As suggested in the paper, L2 normalization is performed.

This model outputs a feature vector for the given video.

[1] Schroff, Kaleinchenkov, and Philbin, '' FaceNet: A Unified Embedding for Face Recognition and Clustering'' Google Inc



Loss Function:

The setting of our model was different from our usual settings so those loss functions couldn’t be used. Thus, after extensive search we decided to use the triplet loss function. It is given by

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ .$$

Where f(x) = Feature Vector,
a = Anchor point
p = Positive point
n = Negative point
alpha = margin

The Anchor Point is defined as the class sample we are using, while the Positive Point is a sample belonging to the same class (brand) as that of the anchor point. The Negative Point on the other hand is a sample of the class that does not belong to the class as that of former. Alpha determines by how margin we want classes to be set apart (or closer to positive point)

Choice of Triplet:

As we can saw, for our loss function we need three data points(videos). Anchor point is just the point on which we are working on while the selection of other two datapoints is important. There are different ways to chose them:

- Make all possible triplets. This is not feasible in any scenario.
- Choose randomly from the dataset. In this case the loss function will be almost always very small and thus the model will not learn anything.
- Another way would be to choose hard points from all datasets. It means select those data points in which distance of feature vector of anchor is largest from positive while smallest from negative value. However, this increases the training time and increase the complexity.
- The better way which we also have used in the model is to do something called ‘semihard mining’ which basically does the same thing described in the previous bullet but on batch of datapoints.

Optimizer:

Our algorithm makes use of Adam Optimizer with following Hyperparameters:

- Learning Rate = 0.001
- Beta 1 = 0.9
- Beta 2 = 0.999

This optimizer updates the weights using the following technique:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

Training:

The model was trained for 5 epochs and batch size 32.

Results:

Learning Curves:

Accuracy:

Database:

Since at inference time, we need to cater for the unknown brand and determine the brand of the advertisement, we create a database. This database contains feature vector of each brand (obtained by passing single random example of each brand through the network) in. npy format with brand name as filename.

Prediction Code:

To make predictions on unknown dataset, a separate file ‘AdNet_Prediction_Code.ipynb’ is provided. This separates the training code and makes it easier for the user to estimate the outputs on his data. This file has a function ‘prediction’ which takes two arguments:

- ‘test_video’ : path to the video for testing. The video must be in ‘.avi’ format.
- ‘unknown_data’: list of size two. The first element must be path to the video of unknown brand and second element must be name of the brand. (optional)

It gets the current working directory, and then looks for model “AdNet.h5” and folder “Database” in the directory. Once found, it checks whether unknown brand needs to be added in the database. If yes, it checks whether it already exist in the database or not. If no, it will read the video, pass through the model creating feature vector and saves it in the database as name_of_brand.npy. Afterwards, it obtains feature vector of test video and matches this vector one by one with each brand feature vector available in the database. For matching, we are using cosine similarity which is given below:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

The value ranges between -1 (complete opposite) and 1 (complete match). Further, the code will stop the search if it finds a match that has cosine similarity of more than 0.8 otherwise it will keep looking through the whole database and in the end it outputs the label with highest cosine similarity. More details on how to run this code is provided in Annex A.

Annexes:

Note: The Code was written using Google Colab and it is thus formatted in that way. Incase you want to re-run using other editor/environment, you would have to adjust the code accordingly to see the outputs or some outputs will not be visible because of the way Jupyter Notebook works. Further, although the complete codes are provided in the annexes, but it is recommended to read them through. ipynb file. Also, you may have to install few libraries when running the code. Furthermore, due to time and resources constraint, we were not able to complete the implementation especially the video data generator part since it had to be written from scratch in a way that it integrates with both our algorithm and Keras API. The algorithm described above in theory must work, although we were not able to test it.

Annex A: Instructions on running the code

- Open the file ‘CNN_Prediction_code.ipynb’ in a python environment and place the files named as ‘AdNet.npy’ and ‘Database’ in the same directory as that of .ipynb file.
- In the.ipynb file, run the libraries block and function prediction block. Then call the function by passing the argument:
    - ‘test_video’: path to your test video
    - ‘unknown_data’: [‘path to video’, ‘brand name’] (optional)

  A template is available in.ipynb file. You can reuse it by simply changing the directory according to your own requirement.
- Available brands can be found by going through the database folder.

Annex B: Training Code:

Annex C: Prediction Code