DeepLogo : A Deep Learning Architecture for Brand Recognition

Venkat Neehar Kurukunda
A53206917
vkurukun@ucsd.edu

Ranti Dev Sharma
A53223197
rds004@eng.ucsd.edu

Manu Seth
A53219535
mseth@eng.ucsd.edu

Francesco Ferrari
A53204320
fferrari@eng.ucsd.edu

Ayush Jasuja
A53103338
ayjasuja@eng.ucsd.edu

Abstract

Logo detection implemented using computer vision algorithms has attracted a lot of attention in the past. It is over past couple of years that researchers have tried to solve the problem of logo recognition and localization using Convolutional Neural Nets. This problem particularly interests companies which are not only interested in measuring their social media infusion but it also helps in monitoring the social settings in which audiences see the brand. In our work we use VGG16 model to learn and extend logo classification for videos.

1 Introduction

Logo detection in social media images is a very important metric to measure brand infusion and is one of the keys to brand management. Classification of digital content based on brand logos has a huge number of applications, viz. designing advertising campaigns, automating vehicle traffic control system, video metadata validation and many more. A lot of work which has been done for this problem has focused on just images. The majority of approaches have been standard computer vision techniques based on keypoint-based detectors and descriptors. An example of such an approach is extraction of spatial layout of local features (edges, triangles etc) in logo containing images, and then coming up with representation of logo containing regions in images [1].
Bianco et al. [2] used a deep convolutional neural network for classification of logos in images. In their approach they localise the logo using a recall-oriented logo region proposal [3], and further train a CNN for logo classification even if localization was imprecise. Details of their approach are described in the next section. Our project is an extension of their idea for a video dataset. We use augmented Flickr-27 [4] dataset on a pre-trained VGG16 and apply transfer learning to train the model for image classification(softmax). For the RNN input, we took a window of ten frames from the video, passed every frame from the learned CNN we got from image classification and trained it to put a softmax classification of the video. For testing the video classification part we use Youtube-8M dataset which provides access to a wide class of pre-tagged videos. Particularly for our project we do a five way classification of videos.

2 Background

The work of Bianco et al on logo classification is a two step process on the high level. In their proposed algorithm they approach the problem by first identifying possible image regions which can contain logo (as a single image can contain multiple logos). The second step is about processing the data obtained by augmenting, normalization etc. and training the convolutional net using the processed dataset.

2.1 Logo region annotation

For all the images in the training they mark rectangular regions which contain logo. They also design a object-proposal logo annotation algorithm over both the test and train images. For the training images they mark the regions returned by the algorithm with the ground truth class, if it contains any logo (completely or partially above some threshold). Otherwise they mark the region as background. The regions are cropped to a common size to match the input dimension of the network. For training the CNN all the regions which contain logos above a threshold percentage were used in this approach. Using regions with partial logos make their algorithm more robust.

Fig. 1, is an image of an high level logo classification pipeline used in this work (from the paper).

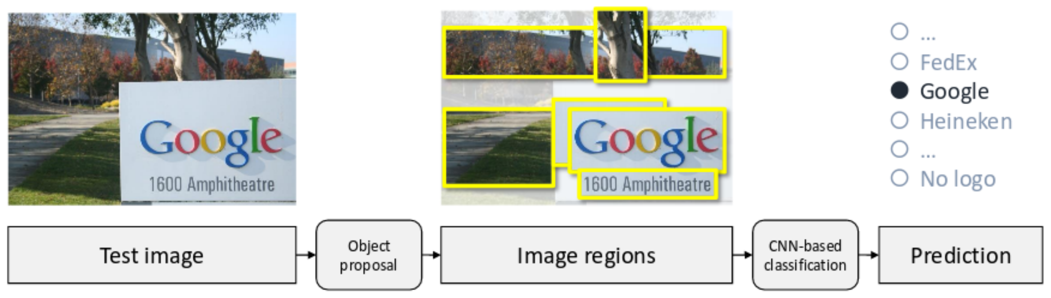

Figure 1: An overview of Bianco et al's algorithm of CNN

2.2 Data Processing

For all the possible image regions used for training, authors apply following methods to enhance data quality and quantity:

- **Class Balancing**: Replicating the samples for classes with less cardinality.
- **Data Augmentation**: Shifting the logo inside the regions to increase the quantity and to make logo detection more robust.
- **Contrast Normalization** : They subtracted the mean of pixel values from all the images.
- **Sample Weighing**: Positive samples are weighted based on their overlap with logo region. This makes the CNN learning more confident about its prediction for total overlap cases.
- **Background Class**: In their learning approach authors also added a background class apart from the logos. For this class they trained the network with regions returned from the test images which had no overlap with logos(not a separate dataset). So, this made the algorithm more robust in detecting regions in the image, which are not of interest.

Once the image regions are generated after applying these techniques, they are used along with the original manually annotated training images for training the convolution net. For training the CNN, the authors of the paper used the framework shown in the Fig. 2.

Finally, for testing they used the framework shown in Fig. 3.

3 Model

3.1 Augmenting the FlickrLogo27 dataset

One of the first issues that we faced was the lack of initial data to train the first layer of our architecture, the convolutiona l neural network. In fact the Flickr Logo-27 dataset contains approximately 30 images for brand, not enough to reach high levels of accuracy. We approached the problem

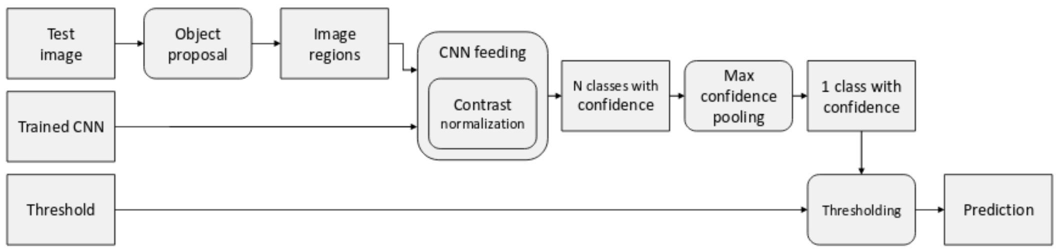

Figure 2: Training framework used by the authors of Bianco et al

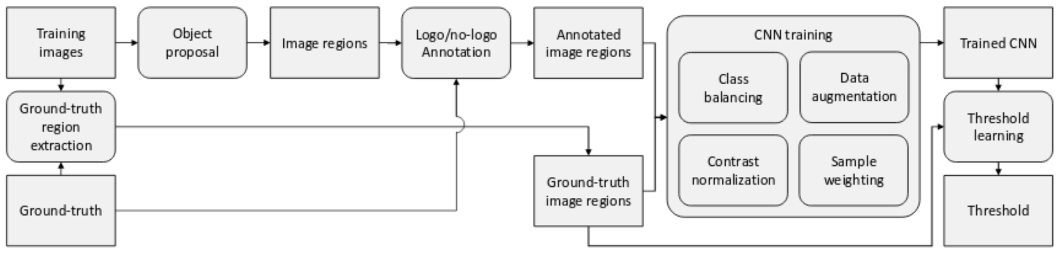

Figure 3: Testing framework used by the authors of Bianco et al for testing

in two ways. Firstly, we selected a subset of five logos which are chromatically and aesthetically different from one another. The brands we selected were Coca Cola, Pepsi, Starbucks, Apple and Nike. We reduced the range of brands also to facilitate the dataset collection of the videos that we used for the RNN. Furthermore some brands, such as Ferrari, are often hard to detect in videos (especially the ones provided by Youtube 8 Millions dataset) because often their product (the car in this specific scenario) is either associated with other brands (other supercars) or because it does not display the logo in a very visible way. Secondly, we worked on augmenting the Flickr-Logo dataset. We firstly cropped each logo using the x-y coordinates provided by the Flickr-Logo dataset. Secondly we applied a series of transformations, including cropping, rotation, scaling and shifting. We excluded mirroring since logos are not normally mirrored. By doing this the final dataset was greatly expanded to approximately 60,000 images for the five selected logos. We did not use the full augmented dataset though, but a subsample of it containing 20,000 images. This was due to some memory issues that we encountered when running the code on the GPU. We would like to highlight though that the average number of images per label in ImageNet is 1000, hence 4000 images is considered enough to achieve a robust accuracy.

3.2 Convnet

Differently from our initial project proposal, we later decided to use the VGG16 architecture for our convolutional neural network. VGG16 is in fact more precise than AlexNet and Keras offers a pre-trained model that can be used for transfer learning. Transfer learning can be helpful in order to prevent overfitting of the training data, if the training data is limited in size. In our case the dataset was not limited in size but it was augmented by a limited number of images, increasing the possibility that we might overfit.

In order to retrain the VGG16 model we removed all the layers after the last max pooling layer and added a new fully connected layer and a new softmax with five output. We also used a callback that would stop the classifier if the cross-entropy loss in the validation set would start increasing again for four times in a row.

The final validation accuracy obtained by the classifier was 99.4%.

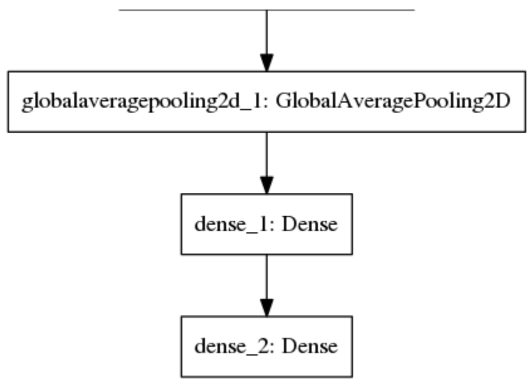

Figure 4: Last three layers of our modified VGG16 Architecture

3.3 Building the video dataset

We are using YouTube-8M [5] dataset to get videos for all the brands for this project. Every video in the dataset is about 2 to 5 minutes long and on average has 3.4 labels. The labels in the dataset are not just brands but can also be activities (dance, singing), natural or artificial phenomenon (snowfall), Sport (Tennis) etc, with a total of 4716 classes. The dataset provides both frame level and video level features for every video. Video level features contain averaged values of image pixels whereas frame level features contain individual frame features for the video. Every data entity also contains fields for youtube video link and list of labels. We downloaded the video level features of every entity in the dataset and discarded the ones which have irrelevant labels as per our project.

3.3.1 Selecting videos and formatting them

After getting about hundred videos for each brand, we used eighty percent of the data for training and the remaining for validation. We sampled each video into frames, and got two frames per second with a maximum limit of thousand frames per video. To ensure uniform quality across all the videos, we resized the frames to 224x224 pixels jpeg images. For all of this processing we use ffmpeg [6] which allows to process and encode video data into required format and size.

Nike	Starbucks	Apple	CocaCola	Pepsi

Figure 5: Example of processed frames from the five brands we chose for our project

3.3.2 Pruning the videos

Since we wanted to use the image frames to do video classification and not every frame is useful in predicting the video class (because not every frame supports the video metadata), we decided to run a sliding window of size 10 over all the frames. For training the RNN, we chose the window which has the maximum average probability of the corresponding brand being present in those frames. For predicting the probability of brand present in a single frame, we used the VGG16 model described in the previous section. Once we got the best frame window for a video, we discarded all other frames for a video as those frames are never used to train the RNN.

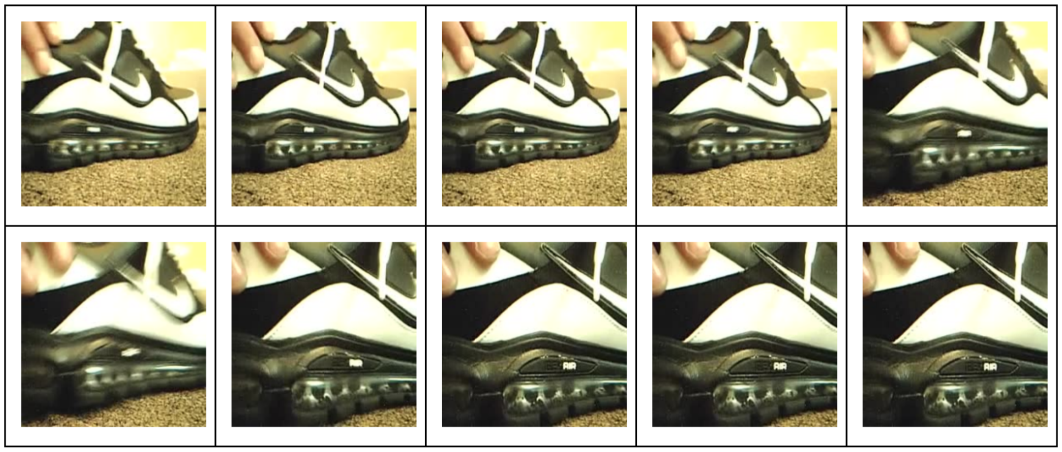

Figure 6: Example of the best window from one of our sample videos of class $Nike$

3.4 RNN architecture

For video sequence classification we decided to use GRU-RNN (Gated Recurrent Unit - Recurrent Neural Network). GRU-RNN is a class of feed forward neural network in which weights are shared across time and connections between units form a directed acyclic graph. Backpropagation through time is used to update its weights. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. GRU-RNNs can use their internal memory to process arbitrary sequences of inputs. Our GRU-RNN is two layer RNN with 100 hidden units for first layer and 50 hidden units for second layer. The input to our RNN can be either softmax layer of previous ConvNet or FC layer of previous ConvNet. We experimented with both. We decided not to use Conv layer as direct input to RNN as Conv layer is high dimensional and Conv layer as input is used for spatial information which is not needed in our case. While training RNN we fixed all ConvLayers weights to reduce training time.

4 Experiments & Results

4.1 Convnet Classification

Shown in Fig. 8, is the accuracy plot for VGG16 trained on the $FlickrLogo27$ data set.

We also experimented with the sampling rate of our videos. Since we were using just 10 frame sequences, the sampling rate determined the coverage of the extracted sequence. For small sampling rate ($\leq$ 1 fps), we got a varied sequence of frames with little correlation between them and hence not so useful for training. For large sample rates ($>$ 2 fps), we observed that most of the frames were same, hence not capturing the essence of video, which is our primary objective. After carefully trying different sampling rates, we found that 2 frames per second gives the best result on the validation set.

4.2 RNN Classification

We added two layers over the pre-trained VGG16, an FC layer followed by a softmax layer. We experimented by using the outputs of the both these layers as RNN inputs. Following plots (figure 9 and 10) show the accuracy plots for both cases. We can see that after a few epochs the model seems to overfit the data but we ensure that weights before over-fitting are taken for results and demo purposes.

4.3 Result

We observe that better results are obtained for the case when final Softmax layer is not bypassed i.e. final softmax probabilities are used for each brand instead of features extracted for the images.

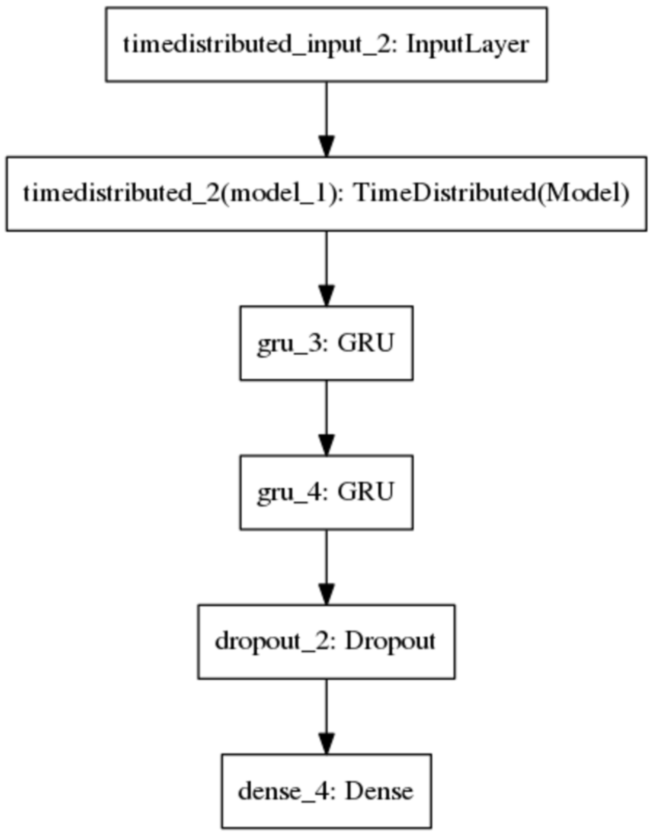
timedistributed_input_2: InputLayer
timedistributed_2(model_1): TimeDistributed(Model)
gru_3: GRU
gru_4: GRU
dropout_2: Dropout
dense_4: Dense

Figure 7: RNN architecture we used in our project

5 Discussion and Conclusions

We can draw some conclusions from the results reported in the section above. First of all we notice that we achieved a 99.4% accuracy on the validation dataset for the first stage of our architecture. This result was achievable after several improvements in our model. Augmenting the dataset and improving the architecture of the convolutional network were the two main factors that lead to this improvement. After augmenting the dataset we saw almost a 10% improvement in the classification. Furthermore adding a new densely connected layer before the softmax improves the results by an additional 2% (the previous accuracy was 97%). The densely connected layer learns in fact an extra internal representation of the logos, adjusting the weights specifically for our task.

The results in the second task, even though they do not have the same level of accuracy, are nevertheless remarkable. We would like to highlight in fact that some of the frames in the videos used for training and testing do not necessarily contain the logos on which we trained the CNN on. We trained several different architectures, including one in which we inputted the densely connected layer from the CNN directly into the RNN. The best performing model was ultimately one that outputted the softmax for each frame and used it as input for the GRU unit. Using our architecture the RNN is able to make stable predictions that do not change from frame to frame. In the second task the model achieved an accuracy above 87%. The input to our model was a sequence of five frames sampled using a frequency of one frame per second. We tried to increase the length of the frames sequence but unfortunately we encountered memory issues with Keras when we tried to do so. Longer frames sequences will resemble more closely the length of actual videos. For similar memory issues we trained both models using mini-batches with just one row per mini-batch. On

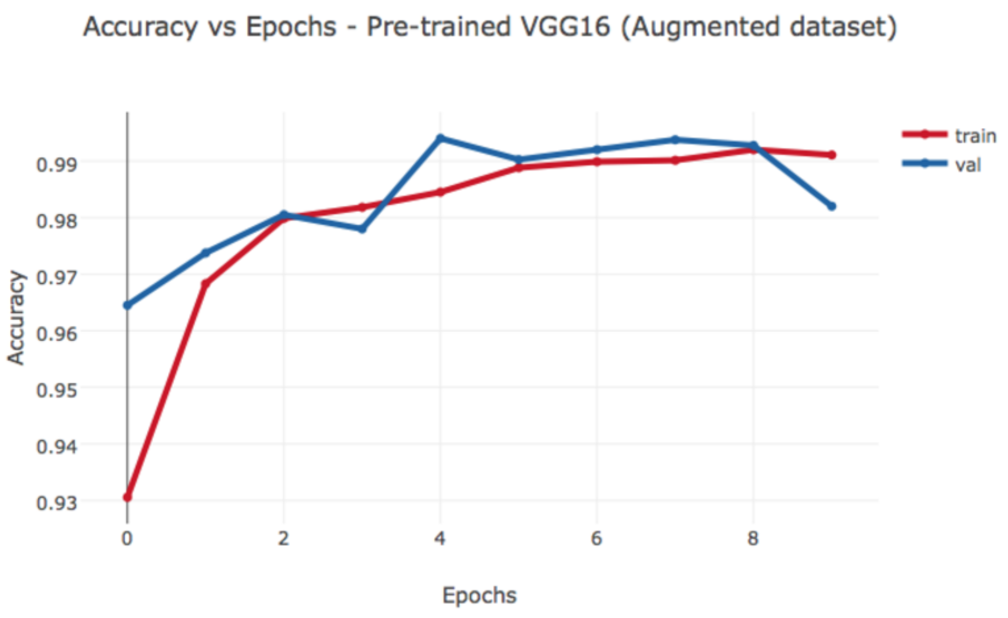

Figure 8: Accuracy plot for VGG16 trained on the $FlickrLogo27$ data set

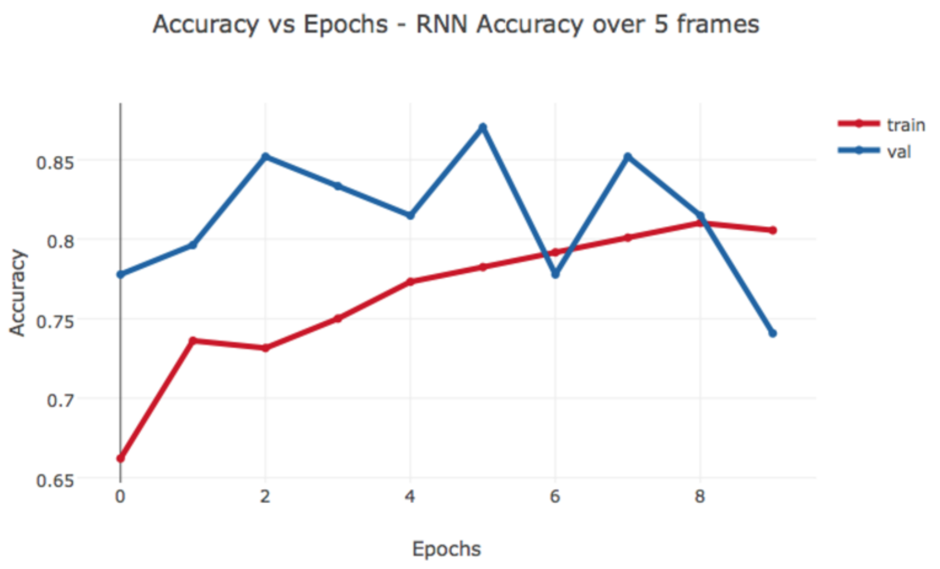

Figure 9: Case 1: When output of the final softmax layer was used as input to RNN

a final note its worth mentioning that the model so far is unable to determine between noise and brands and it labels random sequences of frames as a specific brand. We will overcome this issue by introducing random images in the training for the CNN and label them as *noise*.

Even though we worked on a limited set of brands the results obtained are meaningful. We proved that a reasonable accuracy can be achieved even with a limited dataset of training videos (270 videos, 1500 frames) by pre-training a convolutional layer tuned for those specific brands. Also,

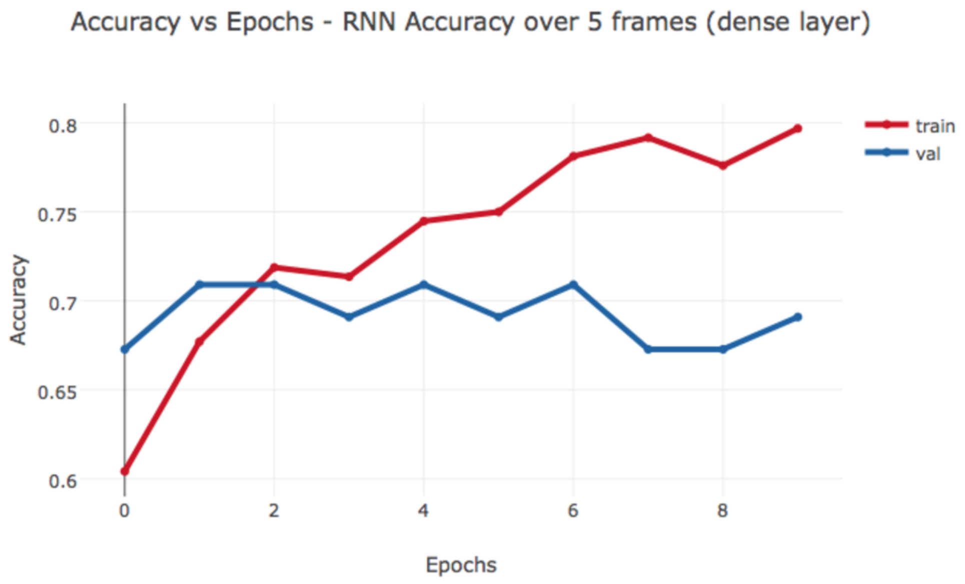

Figure 10: Case 2: When out of the penultimate FCC layer is used as input to RNN

we decided not to go with spatial attention models because we are not using the spatial location of the logos in the images. Hence, it made more sense to just go with the temporal sequence using an RNN.

6 Concept and/or innovation

For this project, the architecture consists of an RNN, the input to which comes from a CNN at every time step. We have not used the regular end-to-end training process, instead we have trained our network using a novel three-step process. Firstly, we have used a pretrained ConvNet (VGG16) trained on ImageNet dataset. Secondly, we have employed the concept of transfer learning to train the last FCC layer and softmax layer using FlickrLogo27 dataset. And lastly, we have trained the RNN using CNN outputs for a sequence of sampled frames. We have created our own training dataset by filtering and preprocessing the $FlickrLogo27$ and $YouTube8M$ dataset as explained in an earlier section. Some of the metaparameters specific to our architecture include number of nodes in the FCC added to VGG16, the number of hidden layers in RNN, and the number of nodes in each hidden layer of RNN. We used the process of validation to set them. Similar to the training process, validation was also a multi-step process, hence the mataparameters of CNN and RNN were set independently of each other. All other regular metaparameters like learning rate, dropout were also set in a similar way.

References

[1] Romberg, Stefan, et al. "Scalable logo recognition in real-world images." *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*. ACM, 2011.

[2] Bianco, Simone, et al. "Deep Learning for Logo Recognition." *arXiv preprint arXiv:1701.02620*. 2017.

[3] Girshick, Ross, et al. "Region-based convolutional networks for accurate object detection and segmentation." *IEEE transactions on pattern analysis and machine intelligence*. **38.1** (2016):142-158.

[4] `http://image.ntua.gr/iva/datasets/flickr_logos/`

[5] https://research.google.com/youtube8m/

[6] https://ffmpeg.org/