Video Label Classification

Objective

Aim of the project is to develop classification algorithms which accurately classify multiple labels of videos.

Problem

There are billions of videos available on Internet and billions of videos are added every year on Internet. These videos don't necessarily have labels embedded with them. Therefore, there is no easy way for websites like YouTube, Dailymotion, Hulu, etc. can recommend these videos to User's based on their interest or based on similar video criteria. It is therefore necessary to classify these videos beforehand (single or multiple labels) based on its type and content. Thus, classifying videos is important and presents unique challenge for machine learning models.

Brief Introduction

In this report, we first review the related work of existing benchmarks for image and video classification. We then present the details of our dataset. Then we focus on current approaches and our approach in this project. Further, we evaluate the approaches on the models. Finally, we discuss regarding our future work.

YouTube-8M Dataset

YouTube-8M is a large-scale labeled video dataset that consists of millions of YouTube video IDs and associated labels from a diverse vocabulary of 4716 visual entities(classes).

The dataset contains videos and labels that are publicly available on YouTube, each of which must satisfy the following:

- Must be public and have at least 1000 views
- Must be between 120 and 500 seconds long
- Must be associated with at least one entity from YouTube target vocabulary

Some of the key highlights about the dataset are:

- This dataset contains over 7 million YouTube videos
- Video Files are stored in tfrecord format files
- Each tfrecord file has approximately 1200 videos
- There are total of 4716 labels (each video with multiple labels)
- Each video can have up to 10 labels

The following are the top 10 labels in the Dataset

1) Games
2) Arts and Entertainment
3) Autos and Vehicles
4) Computer and Electronics
5) Food and Drink
6) Business and Industrial
7) Sports
8) Miscellaneous
9) Pets and Animals
10) Science

Current Approaches

One of the most recent approaches by Google uses a Deep Convolutional Neural Network(CNN) pre-trained on ImageNet to extract the hidden representation immediately prior to the classification layer. Some researchers at Stanford have used temporal feature pooling (TFP) for pooling in convolutional blocks.

Our Approaches

The approaches that we followed in our project are as follows:

1) Approach A: Classifying one frame at a time with a CNN (Single Frame Model)
2) Approach B: Long term Recurrent Convolutional Networks (LRCN)

Approach A: Classifying one frame at a time with a CNN (Single Frame Model)

- For the 2D CNN we have ignored temporal features of videos and attempted to classify each clip by looking at a single frame
- As part of demo and experiments we have only used approximately 720,000 training examples(videos) (approximately 16% of the available training data)
- We have used two types of optimizers namely: “rmsprop” and “adam”
- RmsProp is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients whereas Adam is an optimizer which instead of adapting the parameter learning rates based on the average first moment (the mean) as in RmsProp, it makes use of the average of the second moments of the gradients (the uncentered variance).
- For pooling we have used max pooling which uses the maximum value from each of a cluster of neurons at the prior layer.
- We have used ReLU (Rectified linear unit) as the activation function for each convolutional block.
- Global Average Precision score on the validation set obtained is 0.5924987350077567

Block Diagram

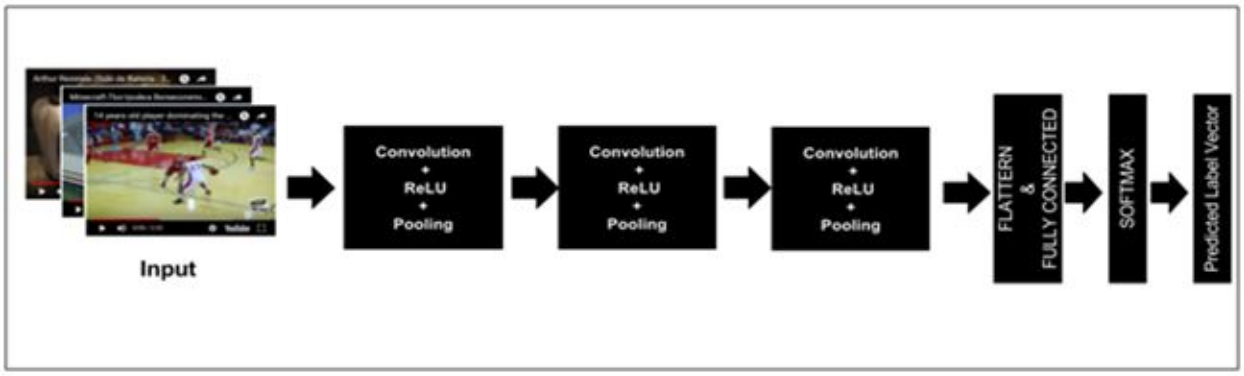

Figure: High Level Architecture Diagram Approach A

Detailed Architecture

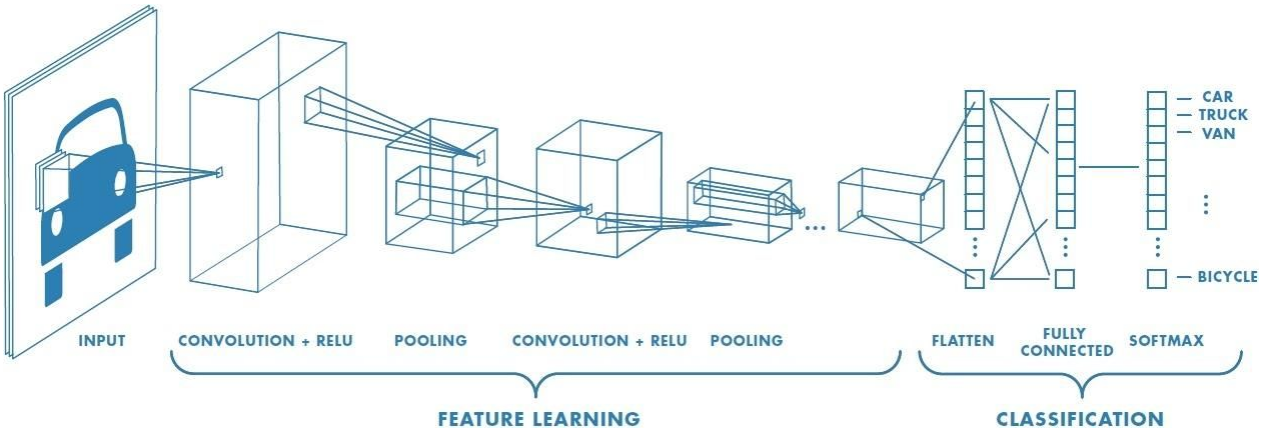

Figure: Feature learning and Classification using CNN

- Convolution - Convolutional layers apply a convolution operation to the input, passing the result to the next layer.
- ReLU - a stack of images becomes a stack of images with no negative values.
- Pooling - Pooling reduces the spatial dimensions of the Input Volume for the next Convolutional Layer. It does not affect the depth dimension of the Volume.
- Regularization - Dropout forces the neural network to learn multiple independent representations of the same data by alternately randomly disabling neurons in the learning phase.
- Probability Conversion - converts the outputs to probability values for each class

Code Snippet of the model using *Keras* Python Library

```
#Model
model = Sequential()
model.add(Conv2D(32,(3,3), input_shape = (32, 32, 1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(32,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))

#Dropout to prevent overfitting
model.add(Flatten()) #flatten feature map into 1 dimension
model.add(Dense(64))
model.add(Activation("relu"))
model.add(Dropout(0.2))

#final layer to predict probabilities
model.add(Dense(4716))
```

Following are few examples for approach: Actual labels vs predicted labels

- **Example 1**

Actual Labels	Predicted Labels
Game	Game
Athlete	Basketball
Basketball moves	Basketball moves
Point guard	Highlight film
School	Association football
Nan	Slam dunk
Highlight film	Athlete
Basketball	Wrestling
Slam dunk	Stadium
Number of labels found common: 6 out of 9	

- **Example 2**

Actual Labels	Predicted Labels
Cymbal Snare drum Drum Drummer Drum Kit	Cymbal Snare drum Drummer Drum Kit Drum
Number of labels found common: 5 out of 5	

- **Example 3**

Actual Labels	Predicted Labels
Minecraft Castle	Game Video game
Number of labels found common: 0 out of 2	

Approach B: Long term recurrent convolutional networks (LRCN)

- First, we ran every frame from every video through inception, thus saving the output from the final pool layer of the network.
- Then, we converted the extracted features into sequence of extracted features and turned each video into 120 frame sequence.
- Finally, we passed on this sequence to train RNN model without needing to continuously pass our images through CNN every time we read the sample or when we trained a new network architecture.
- For the RNN, we used a single, 4096-wide LSTM layer, followed by a 1024-wide dense layer with some dropout in between.

Block Diagram

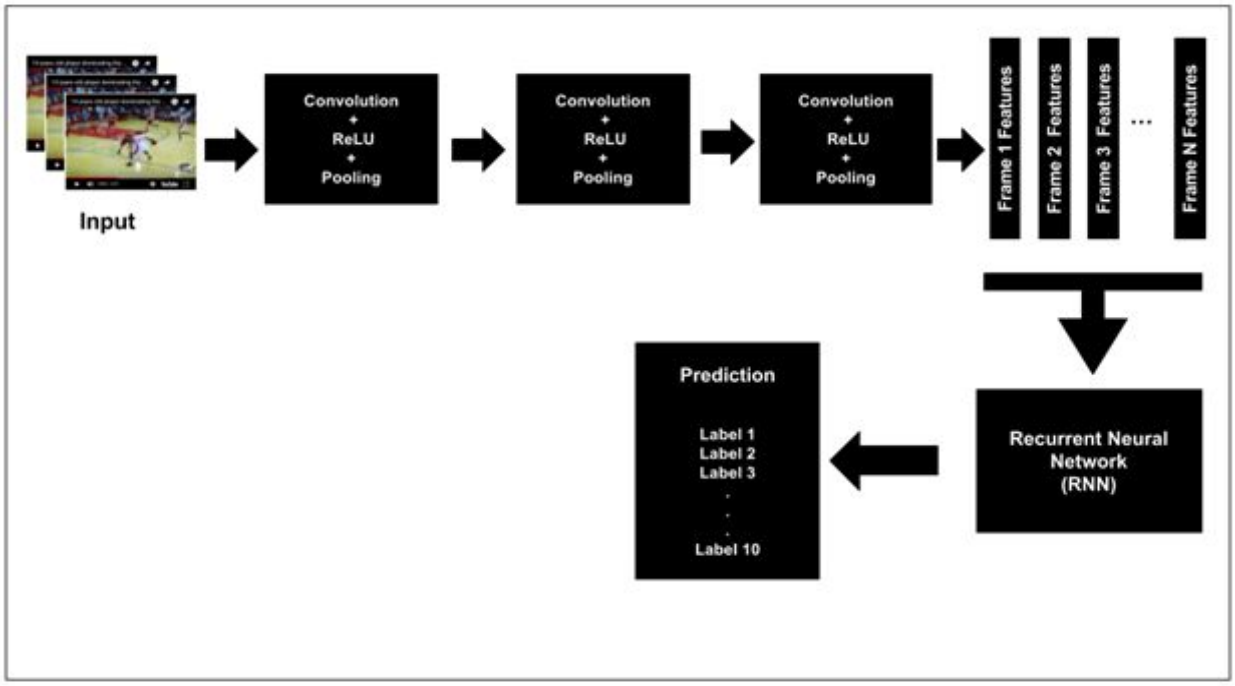

Figure: High Level Architecture Diagram Approach B

- Long-term Recurrent Convolutional Network (LRCN) model combines a deep hierarchical visual feature extractor (such as a CNN) with a model (such as a RNN) that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data (inputs or outputs), visual, linguistic, or otherwise.
- RNNs (Recurrent Neural Network) make use of the sequential information. They perform the same task for every element of a sequence, with the output being depended on the previous computations.
- To make things interesting, Convolutional networks with recurrent units are generally applicable to visual time-series modeling.
- LRCN model works by passing each visual input (an image in isolation, or a frame from a video) through a feature transformation to produce a fixed-length vector representation.
- With sequential inputs and scalar outputs, we take a fusion approach to merging the per-timestep predictions into a single prediction for the full sequence.
- Individual frames are input to convolutional networks which are then connected to a single LSTM layer with 256 hidden units.
- Networks are trained with video clips of 120 frames. The LRCN predicts the video class for each frame and we average these predictions for final classification.

Code Snippet of the model using *Keras* Python Library

```
#Model
model = Sequential()

model.add(TimeDistributed(Conv2D(32, (7,7), strides=(1, 1), activation='relu',
                                 padding='same'), input_shape=(120, 32, 32, 1)))
model.add(TimeDistributed(Conv2D(32, (3,3), kernel_initializer="he_normal", activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))


model.add(TimeDistributed(Conv2D(64, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(Conv2D(64, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

model.add(TimeDistributed(Conv2D(128, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(Conv2D(128, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))


model.add(TimeDistributed(Conv2D(256, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(Conv2D(256, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(1, 1))))

model.add(TimeDistributed(Conv2D(512, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(Conv2D(512, (3,3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2), strides=(1, 1))))

model.add(TimeDistributed(Flatten())) #flatten feature map into 1 dimension

model.add(Dropout(0.2)) #Dropout to prevent overfitting
model.add(LSTM(256, return_sequences=False, dropout=0.2))
model.add(Dense(4716, activation='sigmoid'))
```

Following are few examples for LRCN model: Actual labels vs predicted labels

- **Example 1**

Actual Labels	Predicted Labels
Dish Cooking Food Cuisine Banh	Food Cooking Recipe Dish Cuisine Cooking Show Kitchen Animation Vegetable Eating
Number of labels found common: 4 out of 5	

- **Example 2**

Actual Labels	Predicted Labels
Vehicle Car Driving Digital Video Recorder Dashcam Traffic	Car Vehicle Driving Road Dashcam Sports Car Motorsport Racing Highway Race Track Traffic Supercar
Number of labels found common: 5 out of 6	

Results

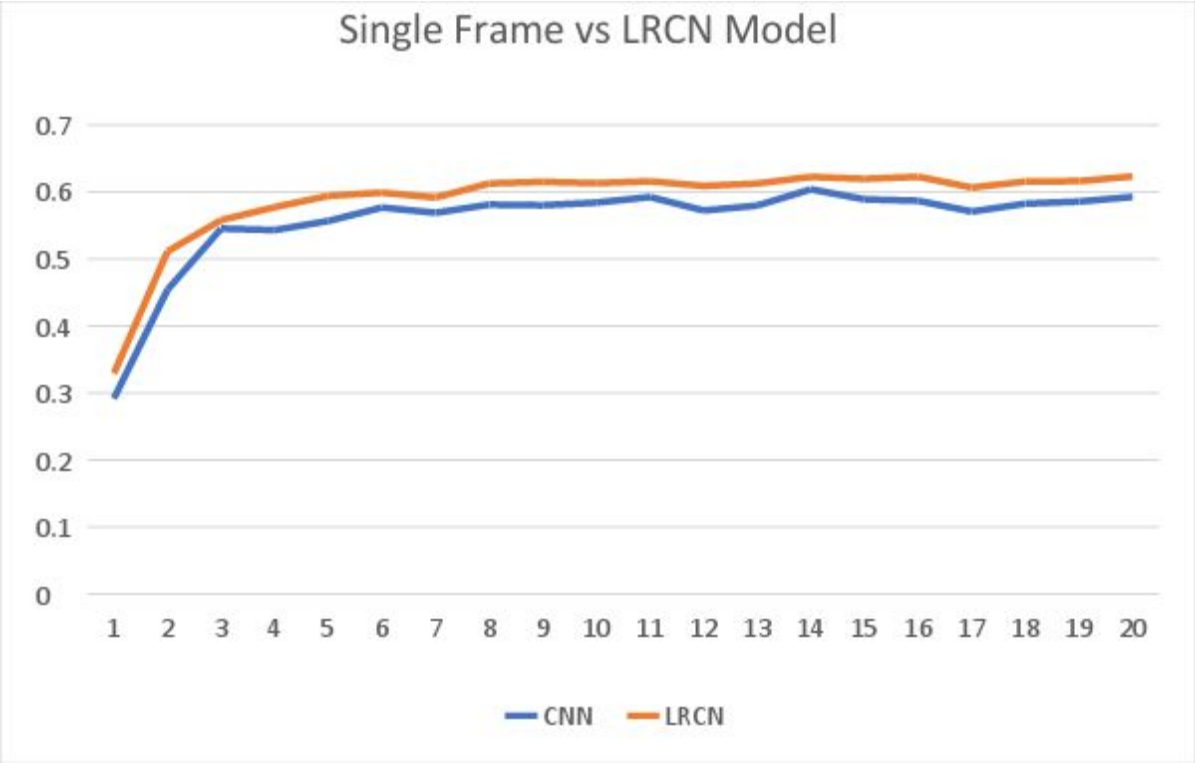

Figure: Comparing results of both approaches

Industrial Applications

- Video hosting websites can use trained models to better recommend their users
- Can prevent clickbait users from hosting attractive and incorrect description
- Video hosting websites can also auto-label and caption videos specially the videos that have no description or have description in different languages.

Validation and Testing

We have used Holdout Method for validation for the following reasons:

- We have an enormous amount of data to train on. So, we aren't losing out on the number of available examples.
- It takes huge amount of time for a single iteration of training and testing so using k-fold cross validation is not a good option.

Testing has been done on completely unseen dataset provided by Google.

Evaluation Metrics

Global Average Precision score is used as evaluation metrics for our project. Accuracy is not a good evaluation metric as we can have a lot false-positives and false-negatives. Moreover the number of labels to be predicted is not known.

Future Work

We plan to implement a few other things in the future to get better Global Average Precision scores:

1. Extracting audio features of the videos and using a separate LSTM layer to improve predictions over the LRCN (Approach B).
2. Building a 3D CNN (time as the third dimension) architecture

Cost Incurred

We have used Google Cloud Machine learning platform which gives $300 credit free for first 12 months and we used approximately $57 of instance hour credits to do the computation for both models.

Software and Hardware

We have used Python platform for implementation due to the availability of several popular Deep Learning libraries like Tensorflow to load the tfrecords and Keras that helped in creating CNN for the Single Frame Model and LRCN model. Also, we have used a 16GB RAM computer system to execute our implementation on the dataset.

Software: Python using libraries - numpy, pandas, tensorflow and keras
Hardware: ThinkPad T460p with i7-6820HQ processor and 16GB of RAM

Conclusion

Approach B using LRCN gives a better Global Average Precision score than Approach A using Single Frame Model but it is computationally more expensive and time consuming to train a model because it requires huge amount of memory to store all the frames of each video.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. CoRR, abs/1609.08675, 2016.

[2] Po-Yao Huang, Ye Yuan, Zhenzhong Lan, Lu Jiang, Alexander G. Hauptmann. Video Representation Learning and Latent Concept Mining for Large-scale Multi-Label Video Classification.

[3] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell. Long-term Recurrent Convolutional Networks for Visual Recognition and Description.

[4] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, George Toderici. Beyond Short Snippets: Deep Networks for Video Classification

[5] Christian Szegedy, Wei Liu, Chapel Hill, Yangqing Jia, Pierre Sermanet, Scott Reed,Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going deeper with convolutions.