Unsupervised Learning: Clasifying Unlabeled Video frames

By

ROBERTO CHAVEZ

Jack Baskin School of Engineering
UNIVERSITY OF CALIFORNIA, SANTA CRUZ

A dissertation submitted to the University of California Santa Cruz in accordance with the requirements of the degree of BACHELORS OF COMPUTER ENGINEERING in Jack Baskin School of Engineering.

JUNE 2017
Abstract:

The proposal for my thesis will develop a classification Deep Learning algorithm accurately assigning video-level labels using the new and improved YT-8M V2 dataset from Google (Youtube). Not only the algorithm will help the consumer find their video labels quicker, it can classify copy righted videos and have them removed. The YT-8M V2 dataset contains 8 million video URLs, 0.5 million hours of video stream, 1.9 billion frame features with audio, 4800 classes, and 1.8 average labels/video. The data will be stored in the Google Cloud ML that is free for participants in the competition. This algorithm will most likely be a combination of Convolutional & Restricted Boltzmann Machine, making it the perfect opportunity to utilize tensorflow, a deep learning api from Google for both research and practical applications. Right now in the Kaggle competition, the best predictive model is a 83.2% success rate so there still proposals we need to consider. Below is a model of a fully connected layer that inspired me to replicate a similar approach for this thesis.

Figure 1: Fully Convolutional Localization Network from Andrej Karpathy [1]

Acknowledgement:

I would like to acknowledge Dimitris Achlioptas and Roberto Manduchi for giving me the opportunity to do my thesis on Deep Learning. Google for initiating the kaggle competition and distributing their Youtube-8M dataset. Lastly I like to thank Hugo Larochelle for teaching his online Neural Network course.
# Table of Contents

Abstract: \hspace{1cm} ................................................................. i  
Acknowledgement: \hspace{1cm} ................................................... i

1 Introduction: Looking at the Dataset  \hspace{1cm} 1  
1.1 YT8M V2 dataset \hspace{1cm} .................................................. 1  
1.2 Video level RGB features \hspace{1cm} ...................................... 2  
1.3 Frame Level Features \hspace{1cm} ........................................ 2

2 Background Research  \hspace{1cm} 5  
2.1 General Approaches \hspace{1cm} ............................................. 5  
2.2 Multilayer perceptron (Feedforward Neural Net): \hspace{1cm} .... 5  
2.3 Convolutional Neural Network (CNN):  \hspace{1cm} .................... 6  
2.3.1 Filters: \hspace{1cm} ......................................................... 6  
2.3.2 Rectified Layer Unit: \hspace{1cm} ..................................... 7  
2.3.3 Local Contrast Normalization Layer: \hspace{1cm} .................. 7  
2.3.4 Pooling: \hspace{1cm} ....................................................... 8  
2.4 Feature Extraction \hspace{1cm} .............................................. 9  
2.5 Feature Aggregation \hspace{1cm} .......................................... 9  
2.6 Restricted Boltzmann Machine (RBM):  \hspace{1cm} .................. 10

3 The Approach  \hspace{1cm} 11  
3.1 My Architecture \hspace{1cm} ..................................................... 11  
3.1.1 Feature Extraction \hspace{1cm} ........................................... 12  
3.1.2 Pooling: \hspace{1cm} ....................................................... 12  
3.1.3 Convolutional Restricted Boltzmann Machine (CRBM): \hspace{1cm} 12  
3.1.4 Fully Connected Layer: \hspace{1cm} .................................. 13  
3.1.5 Activation Label: \hspace{1cm} ........................................ 13  
3.2 Experiment: \hspace{1cm} ....................................................... 13  
3.3 Conclusion: \hspace{1cm} ..................................................... 14
There were two important tasks Google researchers [6] developed with labeled video dataset in a large scale.

1. **YT8M V2 dataset**

   1. Scaling time-consuming videos to annotate manually than images. This was resolved by identifying relevant knowledge graph topics for all public YouTube videos. Any URL site with more than 1000 views, seeking diverse vocabulary of entities, and 24 top-level verticals popular on YouTube were considered relevant knowledge.

   2. Scaling videos that are computationally more expensive to process. Initially dealing with petabyte of videos storage and dozens of CPU-Layers worth of processing may seem impractical for students. Google researchers successfully pre-processed the videos and extracted frame-level features using Deep Learning in Image Recognition. Features were extracted 1 frame per second from 1.9 billion videos and was able to compressed petabyte content down to less than 1.5 TB. Thus making it practical to train the data from a tensorflow model in less than a day (1 CPU).

This dataset will spark new opportunities in research, designing new video modeling architectures and representation learning to adapt to a less noise and unknown video labels.
1.2 Video level RGB features

Now it’s time to look at the RGB features. We need to compare the distributions of the average video level features in order to prevent usage of video level features that are not important for the predictive classifier. The distributions is in a compressed reduction.

![Histogram of video level RGB features](image)

Above diagram shows the distributions don’t differ a lot by just looking at them.

1.3 Frame Level Features

Now we examine the audio frame level features from a finite set of videos. The frame feature were extracted from taking a sample each second through a video [6].

![Heatmaps of frame level features](image)
1.3. FRAME LEVEL FEATURES

Above we have a two-dimensional array for the audio frame level feature of one video. These data arrays represent a spectrogram where each row represents frequency whereas each column is a timestep. The color represents the magnitude of the audio frequency. Now that we have a good understanding of the provided dataset, now we can theorize machine learning algorithms that'll help classify YouTube labels as accurate as possible.
2.1 General Approaches

Most of the research classifying video labels initially started with Multilayer perceptron or Convolutional Neural Network. Since half of the reading committee doesn’t have machine learning background, this chapter will go in depth behind Feedfoward Net and CNN to introduce Convolutional Restricted Boltzman Machine CRBM into a deep belief net in the next chapter for the youtube dataset. Industrial applications require large amount of labled data to have an accurate classifer. Unsupervised or semi-supervised learning algorithms will help reduce the large amount of labeled data necessary for existing models to work well across broader range of tasks. CRBM is an unsupervised algorithm that’ll help support this issue and we’ll exam which features from the youtube video frame will be utilized in the next chapter.

2.2 Multilayer perceptron (Feedforward Neural Net):

It’s roughly a mathematical replication of the human brain where the model are a number of learning units called the neuron. These neurons convert input signals (pre-activation) into corresponding output signals to the next possible layer which learns more about the data. The next layer of units is more knowledgable compared to the previous units because there were a series of activation units (sigmoid) it recieved and combine them all into one new unit. The new unit also contains different times the bias and weights were initialized from all the previous units. This process continues until we reach the final activation unit which is the output (hypothesis function). For example, from a 3-D graph If one signal sends a signal that was shaped half of a convex and another send the other half, the new activation combines the two and it’ll be a complete convex graph. Most basic Multilayer perceptron accuracy falls down under 92% if we’re
looking at labeled MNIST data set which is really bad. Feedforward Neural Net is still important concept to know before going into CNN and later CRBM.

2.3 Convolutional Neural Network (CNN):

CNN is a deep learning model [1] used to classify objects from an image. CNN is an upgrade from a regular feed-forward for four reasons. (1) Filters (2) Rectified Layer Unit (3) Local Contrast Normalization Layer (4) Pooling

2.3.1 Filters:

This is a patch (weight) with lower dimensions compared to the input image. The filter is an array of numbers that will be utilized for a mathematical linear operator called convolution, multiplying the filter with the original values of the given image. The filter doesn’t compute every value of the given image instantly. It has a smaller depth than the image so it needs to start computing values from the top-left corner of the input, send the output to the feature map (mention later), and have the filter slide to the next part of the image that hasn’t been computed.

\[
(1) \quad x_{ij}^l = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} x(i+a)(j+b)^{l-1}
\]

\[
(2) \quad y_i = g_i \tanh \sum_i k_{ij} * x_i
\]

1. For the convolution: (2)
   
   - $x_i$ is the $i$th channel of input
   - $k_{(i,j)}$ is the convolution kernel
   - $g_j$ is the scaling factor
   - $y_j$ is the hidden layer
2.3. CONVOLUTIONAL NEURAL NETWORK (CNN):

If you look at the image above, the top left corner of the input neurons (32x32x3) is being computed from a (5x5x3) filter through element wise multiplication. These numbers are summed up (75 multiplications total since were multiplying the same dimension from the filter with the top left corner of the input neuron) and output of 24x24x1 array of numbers which we liked to call the feature map /activation map. These filters are actually screenshot portion of the image and we compare how similar the filter is with a portion of the image. When the filter is placed on top of the image where we initially screenshoted, the computation will be high since that generally resembles the curve this filter is representing. If we perform convolution of the filter on a location on the image the two don’t resemble, the computation will be low. The feature map is an output, utilized to help us detect low level features such as edges and curves and will be incoming inputs for the next layer. Each feature map that’ll be the next input has different values since each one represent their own respected filter that was patched from the image. Now we can start worrying about higher level features such as eyes and ears if were given an image of a face.

2.3.2 Rectified Layer Unit:

We return the absolute of the previous layer which was convolutionized. The intention behind this is to eliminate non-polarity edges from the feature map that are black-to-white or white-to-black. This was inspired from [1] and [2]. Edges are considered confusing for the learning algorithm.

\[ y(i,j,k) = |x(i,j,k)| \] (2.1)

2.3.3 Local Contrast Normalization Layer:

Another technique used to improve the classification for CNN is Local contrast Normalization. From Hugo Larochelle’s work [3], we need to reduce some unit activation’s if neighbors are also active. This is a competition between each neighbor and feature maps to extract the highest active value, and reduce the other units. The first step we do is taking the previous input layer \((x(i,j,k))\) and then we subtract it from some local average computing some local neighborhood \(w(p,q)\) and we can also compute the average across the other channels (feature maps) \(\sum_{i,j,k} w(p,q)\) giving us the equation \(v(i,j,k) = x(i,j,k) - \sum_{(i,p,q)} w(p,q)x(i,j+p,k+q)\) where \(v(i,j,k)\) will preserve the difference. We
plan on dividing our value $v(i,j,k)$ by the local standard deviation $\sigma_{(j,k)} = \left( \sum_{(i,j,k)} w_{ij(pq)} v_{(i,j+p,k+q)}^2 \right)^{1/2}$ to give us our final result $y(i,j,k) = \frac{v(i,j,k)}{\max(c, \sigma_{(j,k)})}$. Image below show’s an example of us reducing a number of weights between each feature map. Max function is to create a low constant variable ‘c’ to prevent numerical problems if the local std dev becomes zero.

2.3.4 Pooling:

We still need to propagate to the next hidden layer with higher level features. We need to shrink each of the feature maps so we can start reducing the number of hidden units for the next hidden layer and introduce invariance to local translation. The method for this is called Pooling [1][2][5][7] where we pick a window size matrix, walk the window across your filtered image, and take the maximum within the window to be stored into the new shrink feature map.

$$y_{ijk} = \max_{p,q} x_{(i,j+p,k+q)}$$

or

$$y_{ijk} = \frac{1}{m^2} \sum_{p,q} x_{(i,j+p,k+q)}$$
2.4 Feature Extraction

1. \( x_{(i,j,k)} \) is the value of the \( i^{th} \) feature map at position \( j,k \)

   \( p \) is vertical index in local neighborhood

   \( q \) is horizontal index in the local neighborhood

   \( y_{(i,j,k)} \) is pooled and subsampled layer.

   \( m \) is the neighborhood height/width

Once were done pooling, we'll have a shrinked version of our new feature map with previous max values. they simply take some \( k \times k \) region and output a single value, which is the maximum in that region. For instance, if their input layer is a \( N \times N \) layer, they will then output a \( \frac{N}{k} \times \frac{N}{k} \) layer, as each \( k \times k \) block is reduced to just a single value via the max function.

2.4 Feature Extraction

Now that we discussed Feedforward and CNN, we can utilize this knowledge for extracting features from the youtube label dataset. Looking at image and motion gradient orientations, extracting are usually done through deep beliefs net activations computed from one frame at a time.

2.5 Feature Aggregation

Aggregating video features can be quite difficult for long videos since were looking at a series of video frames. This was sorted out through recurrent neural network technique Long short-term memory LSTM since the algorithm can handle long sequences of dataset. We would need to capture the distributions of features in the video by averaging/maximizing the pooling of features.
2.6 **Restricted Boltzmann Machine (RBM):**

The most important concept from this thesis is choosing a unsupervise algorithm [3][4] since the RGB frame contains a large amount of unlabeled data. I choose Restricted Boltzman Machine since it has shown a lot of promise from ImageNet competitions in the past 5 years [1][5]. This model can overcome vanashing gradients by automaticaly finding patterns in our data through reconstruction. Thus likely outputing a higher accuracy for a classifier. An RBM is a shallow two layer net where the first is the visible layer whereas the second is the hidden layer. The model is a two-way translater between the visible and hidden layer in order to eliminate which input features are essential. This saves time when performing back propagation inside the fully connected layer.

![Diagram of Restricted Boltzmann Machine](image)

Next chapter will bring up feature extraction and feature aggregation again with my own unique approach.
3.1 My Architecture

There are four main components in my architecture to classify video labels. First component will contain the video and audio inputs separately. Both will be extracted separately into the next state pooling in order to compact the max/avg features. Next will follow up with Convolutional Restricted Boltzman Machine (CRBM) [7], eliminating the least important features from the image datasets. Inside the same stage will be compute inside a fully connected layer. The last stage will be an activation unit outputting the video label.
3.1.1 Feature Extraction

Instead of performing LSTM to extract an audio feature, the data set features already provides activation units that were trained from ImageNet. The dimensions for both audio and video were also reduced to a PCA whitening.

3.1.2 Pooling:

With Pooling, we reduce the audio and video features into a single representation vector. This allows us to avoid tedious computation for unnecessary features with the lowest scores. More info behind pooling is discussed on chapter 2.

3.1.3 Convolutional Restricted Boltzmann Machine (CRBM):

Once were done pooling our inputs, concept mentioned in chapter 2, we propagate into Convolutional Restricted Boltzmann Machine (CRBM). Both CNN and RBM were taught separately from the previous chapter so CRBM is a combination of the two. It’s a convolutional model that’ll support unsupervised datasets from an RGB dataset. We apply the similar ideas we’ve discussed in the convolutional neural net in the feed-forward supervised case and adapted for the unsupervised case. Having hidden units locally connect to shared parameters will divide into feature maps, which introduces a notion of pooling units. This design was inspired from Lee et and his college’s back in 2009.

Above is a layout how convolutional RBM works. From using adapted conditionals, we can perform contrastive divergence. In other words, have energy gradients (function) involve convolutions, similar to the backprop gradients in convolutional network. This provides a pretraining procedure with no extraction of patches but the youtube dataset already done the work for us. With hidden units (detection later), they interact with some visible units in the visible layer then they
interact through some connections that are weighted. We also have some biases for the visible and hidden units.

### 3.1.4 Fully Connected Layer:

We perform a simple forward and backward propagation with little optimization since pooling and Convolutional RBM avoided vanishing gradients and features that weren't important to classify.

### 3.1.5 Activation Label:

This outputs a multi-label classifying the youtube video content from our prediction through a non-linear sigmoid activation.

### 3.2 Experiment:

The dataset contain 8 million videos [6] which is too much for my computer to handle. I decided to download 1/800 of the content and same applied for the audio features. I divided the dataset into 70% training, 20% validation and 10% for testing. Using tensorflow to replicate my architecture, my accuracy gave me 71% which is disappointing since the highest in the kaggle competition is 85%. At the moment I can’t think of any other methods to improve my model other than have more data inside my prediction.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pooling for the Audio features</td>
<td>45%</td>
</tr>
<tr>
<td>Pooling for the Video features</td>
<td>42%</td>
</tr>
<tr>
<td>Convolutional RBM for Audio</td>
<td>65%</td>
</tr>
<tr>
<td>Convolutional RBM for Video</td>
<td>61%</td>
</tr>
<tr>
<td>Fully Connected Layer</td>
<td>71%</td>
</tr>
</tbody>
</table>

Above is the accuracy I received from utilizing each of the components separately. We can see the accuracy increase when we integrate into the next component in the architecture. Even though I didn’t get the best accuracy, using CRBM in comparison to a regular CNN has improved my accuracy significantly.
If you look at the chart above, the highest accuracy drop 11% in comparison to an CRBM model implementation. From this competition, I can validate using an unsupervised learning algorithm for images is an improvement compared to a learning model that is only supervised.

3.3 Conclusion:

Right now the computer vision community are finding improvements to classify image recognition. Using unsupervised algorithms in Deep Learning will help us in the long run to have higher accuracy for these issues. The kaggle site has more machine learning competitions and hope to pursue more deep learning techniques.
Bibliography


