# Video Genre Classification using Convolutional Recurrent Neural Networks

Dr K Prasanna Lakshmi[1]
Professor and Head, Information Technology Department, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, India

Mihir Solanki[2], Jyothi Swaroop Dara[3]
Information Technology Department
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India

Avinash Bhargav Kompalli[4]
Department of CSE
SRM University,
Chennai, India

***Abstract*—A wide amount of media in the internet is in the form of video files which have different formats and encodings. Easy identification and sorting of videos becomes a mammoth task if done manually. With an ever-increasing demand for video streaming and download, the Video Classification problem is brought into foresight for managing such large and unstructured data over the internet and locally. We present a solution for classifying videos into genres and locality by training a Convolutional Recurrent Neural Network. It involves feature extraction from video files in the form of frames and audio. The Neural Networks makes a suitable prediction. The final output layer will place the video in a certain genre. This problem could be applied to a vast number of applications including but not limited to search optimization, grouping, critic reviews, piracy detection, targeted advertisements, etc. We expect our fully trained model to identify, with acceptable accuracy, any video or video clip over the internet and thus eliminate the cumbersome problem of manual video classification.**

***Keywords*—Convolutional recurrent neural networks; video classification; temporal and spatial aspects; machine learning; computer vision; images classification; audio classification***

## I. Introduction

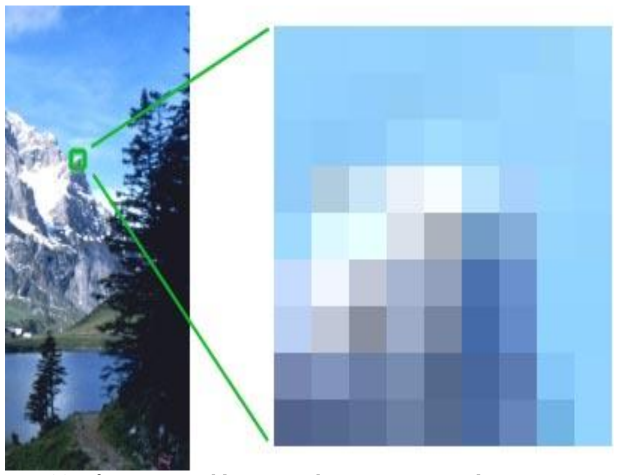
By and large all techniques used in video classification have been image based, with little consideration going into the background audio and annotations. CNN-LSTMs [1] have shown great strides in recognizing image-based video inputs and classifying them into output categories. As humans though, we not only recognize a video by its visual features, but also by the perceived audio it generates. To teach a machine to take similar features into consideration would make a lot of sense because audio plays a large role in classifying videos too. For example, an action scene in a movie will have a fast-paced audio accompanying it, a serious dialogue session will have a lot of voices and weak music notes. Also, a lot of video shot in the internet could be amateurish, with blurry images and weird camera angles.

Giving the context of audio will help the Neural Network more features to rely on while making a classification.

To a human a video is a ray of different colors striking the eyes, but computers perceive video in a completely different way from us. At the lowest level, it is a series of 1's and 0's which makes no sense to the processor except to light up a certain pixel in certain color. When we teach a Neural Network to identify videos, we are asking it to identify certain patterns in those numbers based on mathematical calculations. An image, therefore may be viewed by a machine as in Fig. 1(a) and 1(b).

The problem inherent in computer vision, in fact, the very purpose of the field, is to recover information about the world from sensory input. This can be thought about as a formula:

$$S = f(W) \tag{1}$$



(a) Information visible to a Machine in Gaussian Blur Format.

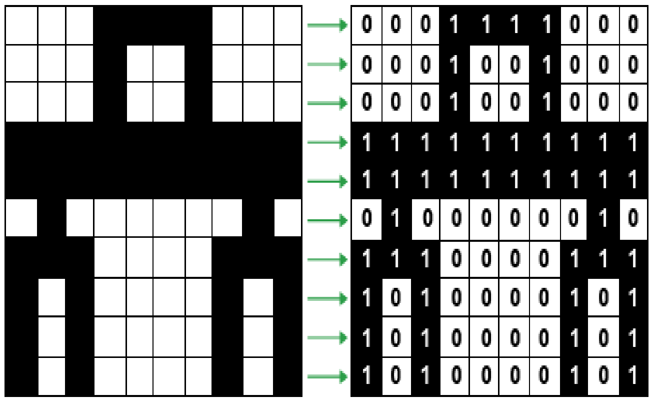


Fig 1. (b): Information visible to a Machine in Bitmap Format.

Our sensory information(S) is a function of the world (W) around us (1). What humans take for granted, and what the field of Computer Vision struggles to make machines do, is the reverse:

$$W = f^{-1}(S) \quad (2)$$

That is, to understand the world from sensory information (2).

Audio is different scene altogether. A common way to input audio to Machine Learning algorithms is by using a Mel spectrogram. A mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

MFC coefficients are commonly derived as follows:

*1)* Take the Fourier transform of (a windowed excerpt of) a signal.

*2)* Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.

*3)* Take the logs of the powers at each of the mel frequencies.

*4)* Take the discrete cosine transform of the list of mel log powers, as if it were a signal.

*5)* The MFCCs are the amplitudes of the resulting spectrum.

A popular formula to convert *f* hertz into *m* mels is in Fig. 2. Fig. 3 shows the generated Mel spectrogram of an audio file using Audacity.

Most state-of-the-art algorithms use this technique as a baseline for their inputs. Hence, we've chosen the same techniques for the inputs to our model. Combining the best of both audio and video classification techniques, we present a unique solution for the video genre classification problem using a Convolutional Recurrent Neural Network or a Convolutional Long Short-Term Memory Network.

Convolutional Neural Networks have been the best at spatial feature extraction and classification problems for images. Some popular examples are ImageNet [2], MobileNet [3], Inception [3], and Google's WaveNet [4]. Feature extraction from a single frame may be straightforward, however a video is a sequence of frames, and every frame is important. For example, we cannot recognize an action of say eating a bowl of cereal, until we have seen a person putting a spoon into the bowl and then into his mouth. Similarly, we must teach a machine to not only look at one frame, but a sequence of frames, to grasp the context of the video. The same analogy can be applied to audio. Hearing a single beat will not help us identify the genre of a song. Only when we hear it for a few seconds are we be able to identify its tempo, the instruments used and its theme. This is where Recurrent Neural Networks come into play. A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. So RNNs can not only understand features at a single timestep, but also remember features from previous timesteps, making them best suited for solving temporal region problems. Long short-term memory (LSTM) [20] follow the RNN architecture and have shown great promise in the video classification problem.

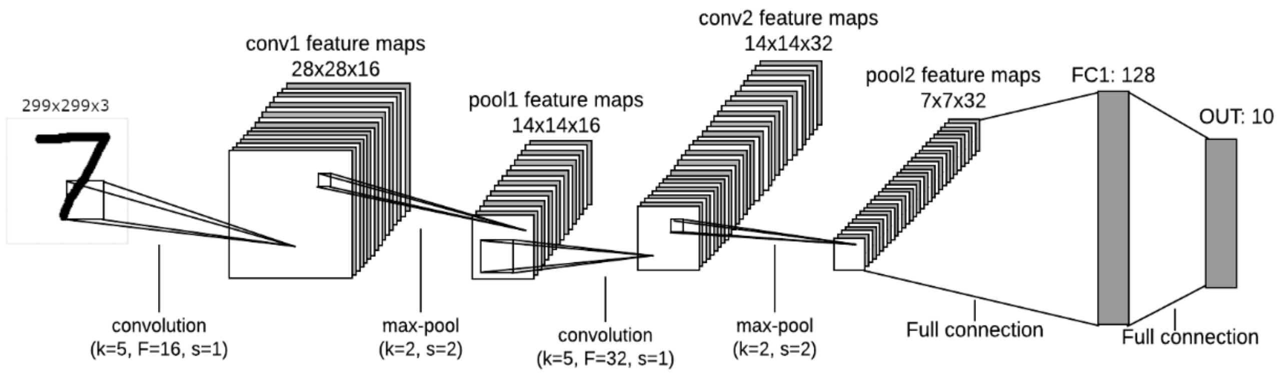


Fig 2. The Convolutional Neural Network Architecture.

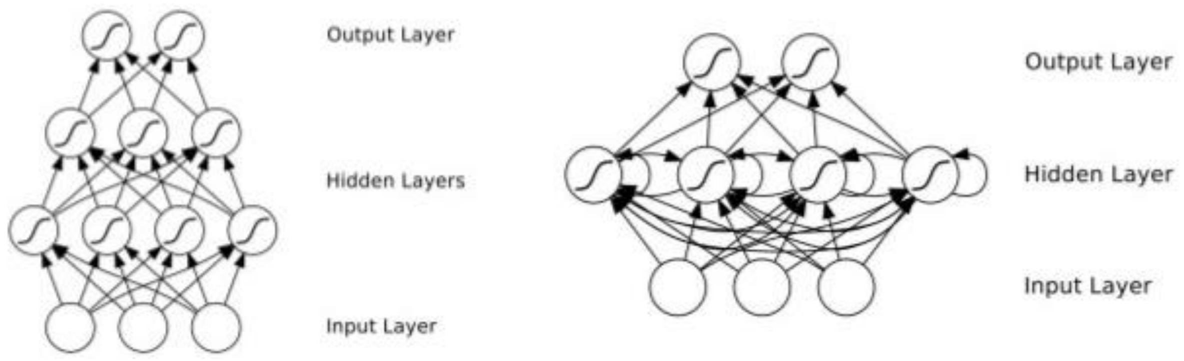


Fig 3. Simplified Representation of a Convolutional Neural Network and a Recurrent Neural Network.

The CNN output can be taken in two methods. One is we take the output from the SoftMax layer, which includes the predictions the CNN has made. The other method is to use the output from the pool layer, which leaves the output prediction to the RNN. We have tried both methods for this paper and they are explained in detail later.

## II. Genre in Videos

A genre for a video specifies a certain expectation about the video. Genres in real world videos are neither specific nor implicit but tend to be overlapping. Also, they vary from person to person as perspective matters. In such a case, defining specific boundaries for genres tends to become difficult. For example, there is a very thin line between the genres Drama and Thriller, and many film critics argue for the same. To define audio into genres has a different shortcoming. Audio Classification is usually multi-label, because they tend to be a mixture of multiple tastes and themes. Background music in modern movies tend to be a mixture of both classical and contemporary, two very different genres if seen separately. Period movies and biographies today are examples for the above.

Therefore, to define the genres for a classifier, we must ensure that we remove the maximum conflicts that occur in genre identification. Hence, we have chosen 6 genres which we can safely say are non-overlapping and mutually exclusive, even if based on different perspectives. The genres we chose are: Action, Animation, Horror, Romance, Sports and Science Fiction. This ensures that our model does not form any tight assumptions about one genre and is also flexible and open for new genres in the future.

## III. Related Work in Video Classification

### A. *Truly Multi-modal YouTube-8M Video Classification with Video, Audio, and Text [5]*

Zhe Wang, Kingsley Kuan, Mathieu Ravaut and others[5] present a novel way in Video classification by using multi-modal features from audio, video and text.Their algorithm classifies the YouTube 8M dataset, which is a collection of over 0.7 million YouTube videos , each labelled automatically, without human curation. The challenge involves classifying an imbalanced dataset based on user generated video content on YouTube. They used TextCNN for titles and Random Forest and max pooling for frame classification. Their research showed that the inclusion of text yielded state-of-the-art results, e.g. 86.7% GAP on the YouTube-8M-Text validation dataset.

### B. *Large Scale Video Classification using both visual and audio Features on YouTube-8M Dataset [6]*

Emma An, Anqi Ji and Edward Ng. [6] presented a solution for the YouTube-8M challenge by considering both audio and video features. They used a Convolutional Neural Network to classify videos into their 4716 classes. Their model used a Mixture of experts (MoE) to receive 3 inputs, video level features only, audio level features only and a concatenation of both audio and video features. They applied a dense layer, followed by a rule activation layer in their model. Taking the softmax function output, they achieve an AvgHit of 0.84, Avg PERR (average precision at equal recall rate) of 0.709, and mAP (mean average precision) of 0.415 compared to the best performing baseline.

### C. *Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification [7]*

Ali Diba, Mohsen Fayyaz, Vivek Sharma and others [7] introduced new 3D convolutional neural network architectures for video classification named DenseNet3D and T3D.They introduced a new temporal layer that models variable temporal convolution kernel depths, embedding this new temporal layer in their proposed 3D CNN, thus extend the DenseNet architecture - which normally is 2D - with 3D filters and pooling kernels. Their research mainly dealt with action recognition in videos, using the Sports-1M, HMDB and UCF101 datasets. They beat algorithms trained in multi-GPU setup for days by removing bottlenecks in the knowledge gained by 2D ConvNets.

### D. *Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks [8]*

Pouya Bashivan, Irina Rish, M. Yeasin, and Noel Codella [8], applied Deep Recurrent-Convolutional Neural Networks in classifying electroencephalogram data. Electroencephalography (EEG) is an electrophysiological monitoring method to record electrical activity of the brain. It is typically non-invasive, with the electrodes placed along the scalp, although invasive electrodes are sometimes used, as in electrocorticography. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain. By training their model, they were successful in demonstrating significant improvements in classification accuracy over current state-of-the-art approaches in this field. A similar CNN-LSTM [21] is used in this paper.

Although a hot topic in Computer Vision, surprisingly less research has been done in the category of genre identification in videos. Most of the state-of-the-art research has been done on image recognition and on solely visual features. The challenges posed for such a classification are noisy data, huge computational costs, large size of datasets and locality/copyright of videos. Some of the limitations of the above papers are:

- They classify videos into categories of fixed actions, which are very specific.
- Most researchers use the YouTube-8M [9] dataset, which is a highly imbalanced dataset and contains very generic categories.
- They rely solely on visual features, ignoring a large amount of audio data.
- They rarely consider temporal space, relying on only spatial features, which bottlenecks most classification attempts.

Through this paper, we attempt to outline and demonstrate methods to improve video classification by fixing most of the limitation mentioned above. Our research is solely academic and is meant to spark interest into the genre-classification problem and its current limitations.

## IV. Data Gathering

There are certain limitations while using existing popular video datasets like YouTube-8M, HMDB [10], UCF101 [11] for genre identification problems, mainly because their labels are not categorized into movie genres. For examples categories like playing sports are placed into human actions, which should instead be classified into a sports genre by our model. Hence, we had to do a lot of manual data cleaning to get our training set.

This paper presents the work which are used in parts or in their entirety as follows:

- The UCF101 dataset
- The Hollywood2 [12] dataset
- The HMDB dataset
- The YouTube-8M dataset

We choose video from these pre-labelled datasets and classified them into folders representing our six genres. For example, videos of punching, fighting and explosions went into the action folder, cases of hauntings and paranormal scenes went into the horror folder and so on. For animation however, we had to take an entirely different approach since there is a dearth of freely available animation videos for research purposes on the internet. We resorted to manually downloading clips from public domain websites [13] [14].

Most of the animated videos found online were old hand-drawn ones, but we were able to secure some modern 3D animation from the blender.org foundation and other open sources.

The compiled dataset now consisted of 39GB of videos, each separated into folders whose names displayed their labels.

## V. Data Cleaning and Preprocessing

A movie video file is usually run at a constant 24 frames per second. If we convert an entire video file into frames, we would get 24 images for a second, which when scaled for a 2-minute video amounts to 2,880 frames. This data is a lot for a model to process and therefore we had to cut down on frame count by taking only 4 frames for each second. This limit was decided after a simple test conducted on human subjects. We split different videos into 2, 4, 6 and 8 frames per second and asked the subjects to cycle through the images and guess the action to be performed. We found that the human mind could perceive any action taking place optimally in 4 frames every second, where 2 would be difficult for slow actions and 6 and 8 would be too easy to guess. Hence, we concluded that an average of 4 frames per second is enough information for a model to recognize what is going on in a frame of time as depicted in Fig. 4. The conversion of video to frames was done with the well-known library OpenCV2 [13] written in Python3.The frames were arranged in similar folders as the videos, with folder names specifying the label.

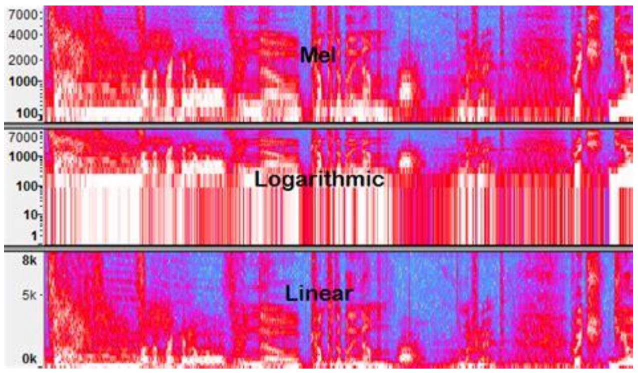


Fig 4. A visualization of different Audio (WAV File) to Frequency Graph Conversion Techniques.

For audio, we used the well-known codec FFMPEG [14]. It provides fast, efficient and lossless conversion of video files into wav files. Then each WAV file was converted into a mel Frequency Spectrogram using the Python 3 library matplotlib [15] and stored into a similar folder structure as above.

Overfitting happens when the model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set. For example, the model recognizes specific images in the training set instead of general patterns. The training accuracy will be higher than the accuracy on the validation/test set. To prevent overfitting, we needed regular validation checks as most of our dataset consisted of specific videos. Hence, we split the set into 80/10/10 for the training, testing and validation sets respectively.

## VI. The Convolutional Recurrent Neural Network Approach

A high-level architecture view of the model is shown in Fig. 3. Both audio and visual features were essentially treated as images, so they could be easily vectorized. This ensured us to categorize inputs easily as a TensorFlow/NumPy array to be given to the model.

### *A. The Convolutional Neural Network*

Pouring research into the availability of state-of-the-art open-source CNNs like ImageNet, VGGNet [16], InceptionV3 [19] and others, we were able to reduce the resource intensive and repetitive task of preparing a CNN model without a baseline. It would prove more time-consuming since we needed a network that was aware of what an image was first, before it could start finding patterns. Hence, we decided on training our dataset on the pre-created models as a baseline. Inception was selected as our base CNN due to its ability of transfer learning for new classes of data as well as better accuracy for home and amateur clips. The InceptionV3 is a neural network architecture for image classification, originally published by Christian Szegedy [19], Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna[17]. This model has already been trained on a similar task for thousands of images and thus comes with an intuition for feature extraction from images.

The input to this layer comes in the form of images of size 225 x 225 x 3(width x height x channels) which are scaled accordingly using NumPy [18].

We train the session for a total of 4000 steps with the default hyper-parameters. Training checkpoints are created every 400 steps. The output will give us a retrained graph in pb format and a text file containing labels. However, we are more interested in the output of the pool and softmax layers. The layer output was taken accordingly in code and then passed on to the RNN. The reason why softmax is useful is because it converts the output of the last layer in the neural network into what is essentially a probability distribution. This gives the RNN more data to work with rather than a single 2048 vector and a class label. The advantage here is that instead of just getting a predefined label as output, we are giving our next iteration the entire data that led to its prediction of a particular label. At the end of this process we have both the vector arrays containing the features as well as the prediction probability of each class label for that vector.

## B. The Recurrent Neural Network

We could choose to build our RNN either as a deeper network or as a wider network. Testing with both options, we discovered better training results while using a wider network. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. The RNN-LSTM [21] has 2 main layers, viz. LSTM layer and the regression layer. The LSTM layer provides the temporal feature extraction that we need for the video. Denoting $*$ as elementwise multiplication and ignore bias term, LSTM calculates a hidden state ht as:

$$i_t=\sigma(x_tU_i+h_{t-1}W_i)$$

$$f_t=\sigma(x_tU_f+h_{t-1}W_f)$$

$$o_t=\sigma(x_tU_o+h_{t-1}W_o)$$

$$\sim C_t=\tanh(x_tU_g+h_{t-1}W_g)$$

$$C_t=\sigma(f_t*C_{t-1}+i_t*\sim C_t)$$

$$h_t=\tanh(C_t)*o_t \qquad (3)$$

Here, *i*, *f*, *o* are called the input, forget and output gates, respectively. These gates have the exact same equations, just with different parameter matrices (*W* is the recurrent connection at the previous hidden layer and current hidden layer, *U* is the weight matrix connecting the inputs to the current hidden layer). They are called gates because the sigmoid function squashes the values of these vectors between 0 and 1, and by multiplying them element wise with another vector it defines the part of the other vector that is allowed to the next layer. The input gate defines how much of the newly computed state for the current input you want to allow to the next layer. The forget gate defines how much of the previous state you want to allow to the next layer. Finally, the output gate defines how much of the internal state you want to expose to the external network (higher layers and the next time step). All the gates have the same dimensions $d_h$, the size of your hidden state. ~*C* is a candidate hidden state that is computed based on the current input and the previous hidden state. *C* is the internal memory of the unit. It is a combination of the previous memory, multiplied by the forget gate, and the newly computed hidden state, multiplied by the input gate. Thus, intuitively it is a combination of how we want to combine previous memory and the new input. We could choose to ignore the old memory completely (forget gate all 0's) or ignore the newly computed state completely (input gate all 0's), but most likely we want something in between these two extremes. $h_t$ is output hidden state, computed by multiplying the memory with the output gate. Not all of the internal memory may be relevant to the hidden state used by other units in the network.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before. Mathematically, the carrying forward of memory is represented as:

$$h_t=\varphi(Wx_t + Uh_{t-1}) \qquad (4)$$

The hidden state at time step t is `h_t`. It is a function of the input at the same time step `x_t`, modified by a weight matrix `W` (like the one we used for feedforward nets) added to the hidden state of the previous time step `h_t-1` multiplied by its own hidden-state-to-hidden-state matrix `U`, otherwise known as a transition matrix and similar to a Markov chain. The weight matrices are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via backpropagation and be used to adjust their weights until error can't go any lower.

From the output of the CNN, we group the vector sequences into 40 frames, giving us 10 seconds of information to process. The RNN has 2056 nodes and gives the output as the six classes with their probabilities. The label with the most probability assumed as the predicted class for the current frame sequence. Fig. 5 shows the architecture of our RNN.

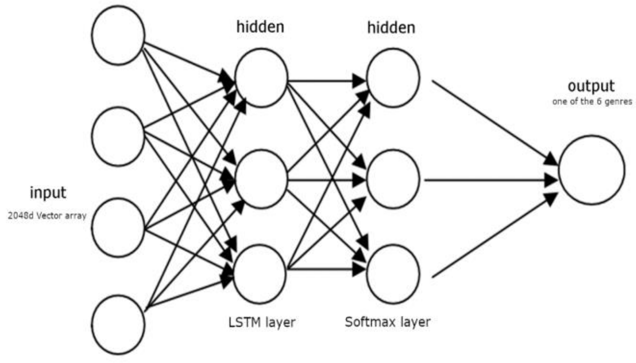


Fig 5. The Recurrent Neural Network Architecture and Layers.

## VII. Training Specifications

Training video classifiers require tremendous hardware capabilities due to the size and structure of data. We decided to use the Google Cloud Platform for training our model. A Deep Learning AMI by Google was deployed on the platform and its' specifications were:

- 4x Intel XEON vCPUs
- 1x NVIDIA Tesla K80 with 12GB VRAM
- 10GB of RAM
- 100GB of fast SSD
- Debian OS

This provided us a fast, reliable, on-the-go and cost-effective solution for cloud training.

## VIII. Experimental Results

With the dataset and model ready, the training took us 4 hours for the CNN part and 4 hours for the RNN part, running on the machine specified above.

The accuracy mark when we used the output from the softmax layer method, that is taking the output from the second layer, yielded 85.4%. This method gave raw data from CNN to the RNN, hence the RNN had an upper hand in making a decision. The TensorFlow log is attached in Fig. 6.

To further improve this, we used the pool layer method which took output from the third layer. This gave more computational power to the CNN and the predictions were narrowed down. This brought the accuracy mark up to 90.3%.

We then tested the model on completely unknown videos from the internet. They consisted of movies, science fiction documentaries, live sport matches and TV series. Our algorithm was able to safely classify videos by observing the temporal space in most of the cases. The shortcomings are discussed later.

To set a benchmark for our method, we trained the naïve model on the UCF101 dataset. The model was able to beat the average accuracy benchmark set on the dataset after just 3 hours of training. Table I lists the comparison of accuracies for different Video Classification methods applied on the dataset.

TABLE I. A Comparison of Various Video Classification Techniques used in the Paper

| ***Sno*** | ***Name*** | ***Accuracy*** |
|---|---|---|
| 1 | ConvNet[22] | 65% |
| 2 | Time distributed CNN [23] | 41% |
| 3 | 3D convolutional Network[24] | 52.8% |
| 4 | CNN-RNN | 74% |
| 5 | **CNN-LSTM – soft-max** | **85.4%** |
| 6 | **CNN-LSTM – pool** | **90.3%** |

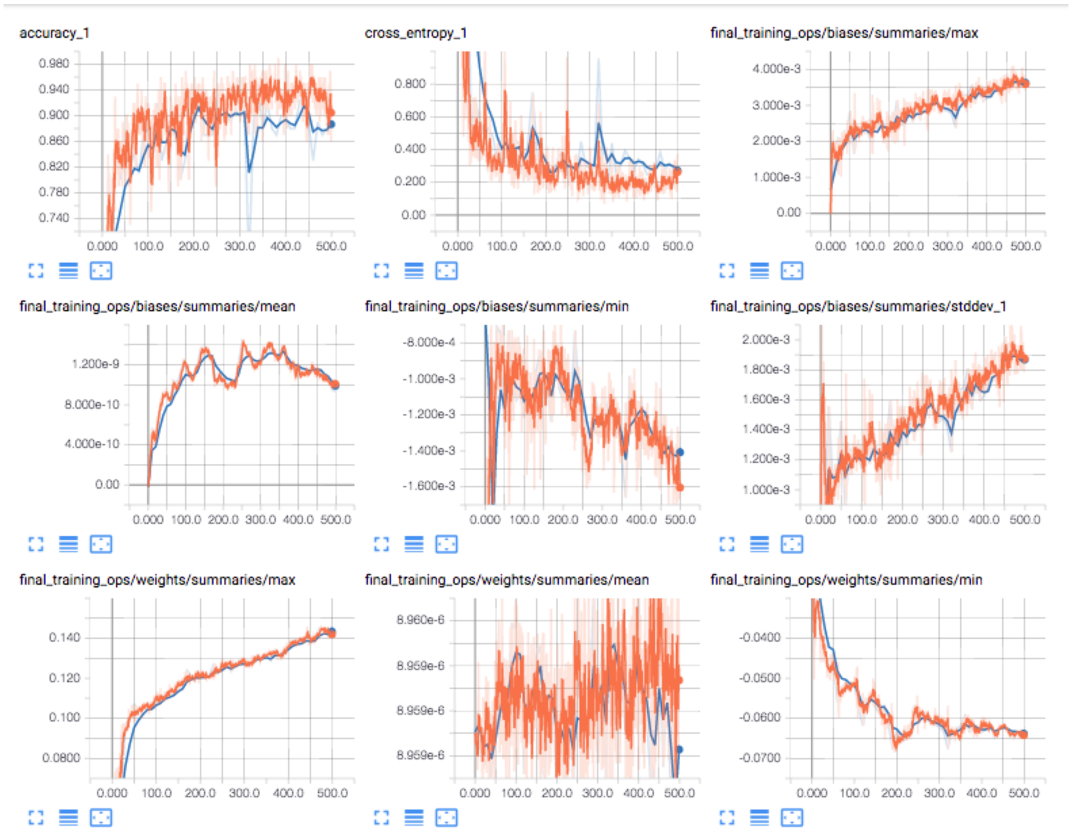


Fig 6. Tensor board Training Graphs for the CNN-LSTM Network.

## IX. Conclusion and Future Enhancements

Video classification is a long open problem with tremendous possibilities for applications in the fields of medicine, entertainment, surveillance, search optimization and many others. Using only visual features has inputs leave a lot of gap for the classification methods to fill. By using audio features, we aim to fill this gap and also make a machine more intelligent while dealing with data. Video files take up a huge chunk of data stored on the internet and easy classification will always be a prime problem to be solved. The lack of proper datasets, copyright issues, video quality, etc. will always continue to be bottlenecks in the way of this problem. However, as more open-source research is made into this field, we can expect to see more efficient methods emerge which are not so computationally expensive. Our paper highlights the main shortcomings many video classifiers are plagued with, namely in using audio features and in the temporal space. Computer vision is and will be a booming field in the years to come as we move to autonomous machines and robots. Video feeds are the best input we can give to these intelligent machines.

However, there is still a long way to go before we can completely trust machines to make prediction on genres. In our testing we found two interesting cases where the model classified a certain genre wrong. In the first case; the input video we gave was from a horror movie, where a ghost is walking vertically on a tree trunk. The model continued to classify the video as action despite there being clear elements of horror present in the scene. Another case is highlighted in the genre of Romance, where due to the lack of lighting and the expressions of the actress, the model thinks the genre is horror. Such false classifications will always arrive as long as machines are ignorant about a lot of other features like human emotions and the technicalities involved in the direction of a movie. To bridge this gap would be a major step in building an AI critic, who could not only classify movies, but also judge their effectiveness and themes.

Every project is at any stage a work in progress, since we cannot achieve a perfect system. The scope for its future enhancement rests on the shoulders of its creators. Our work in this field will continue to grow and we have a roadmap for adding more features to this classifier. Some of our planned enhancements are:

- Subtitle and transcript generation
- Changing video speed based on the action going on
- Vocal narration for disabled
- Large Curations and Sorting of videos
- Medical Video Analysis.

## Acknowledgment

First and foremost, we would like to thank our Institute, Gokaraju Rangaraju Institute of Engineering and Technology, for providing us this opportunity and the environment to conduct the research for this paper. We would additionally like to thank the faculty members and peers of the college for their valuable feedback and assistance during our research process. Finally, we thank the UCF group, HMDB and Hollywood dataset creators for compiling the datasets and making them freely available to the public.

## References

[1] https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[2] http://www.image-net.org/

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv:1704.04861 [cs.CV]

[4] Oord, Aaron van den; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew; Kavukcuoglu, Koray (2016-09-12). "WaveNet: A Generative Model for Raw Audio". 1609. arXiv:1609.03499

[5] Zhe Wang, Kingsley Kuan and Mathieu Ravaut: Truly Multi-modal YouTube-8M Video Classification with Video, Audio, and Text, arXiv:1706.05461

[6] Emma An, Anqi Ji and Edward Ng : Large scale video classification using both visual and audio features on YouTube-8M dataset, unpublished.

[7] Ali Diba*, Mohsen Fayyaz*, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, Luc Van Gool : Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification, arXiv:1711.08200

[8] Bashivan, et al. "Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks." International conference on learning representations (2016).

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

[10] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A Large Video Database for Human Motion Recognition. ICCV, 2011.

[11] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild, CRCV-TR-12-01, November, 2012.

[12] Marcin Marsza{\l}ek and Ivan Laptev and Cordelia Schmid : Actions in Context, IEEE Conference on Computer Vision \& Pattern Recognition, 2009

[13] http://publicdomainmovie.net/

[14] http://publicdomainflix.com/

[15] https://opencv.org/

[16] https://ffmpeg.org/about.html

[17] https://matplotlib.org/

[18] Karen Simonyan∗ & Andrew Zisserman+ Visual Geometry Group, Department of Engineering Science, University of Oxford {karen,az}@robots.ox.ac.uk, arXiv:1409.1556v6 [cs.CV] 10 Apr 2015

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, Rethinking the Inception Architecture for Computer Vision, arXiv:1512.00567 [cs.CV]

[20] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[21] https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

[22] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", arXiv:1506.01497 [cs.CV].

[23] Hyeonwoo Noh, Seunghoon Hong, Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation", arXiv:1505.04366 [cs.CV]

[24] Shuiwang Ji ; Wei Xu ; Ming Yang ; Kai Yu, "3D Convolutional Neural Networks for Human Action Recognition", IEEE Explore.