Recurrent Neural Networks and Transfer Learning for Action Recognition

Andrew Giel
Stanford University
agiel@stanford.edu

Ryan Diaz
Stanford University
ryandiaz@stanford.edu

Abstract

We have taken on the task of action recognition, the classification of videos depicting human action from the UCF-101 dataset. In this paper, we present our experiments with a variety of recurrent neural networks trained on sequences of CNN 'codes' generated by AlexNet. Our results with these models indicate similar performance enhacements over baseline methods as found in modern literature, as well as demonstrate the representational power allotted by transfer learning from large, deep nets trained on massive datasets.

Introduction

Image classification via convolutional neural networks (CNNs) has been the focus of much of the academic landscape in the field of computer vision, yet video classification has been less developed. The task of action classification is notoriously difficult - even with advances in image classification, videos pose a unique challenge as one frame is often insufficient information for proper classification. The best way to incorporate temporal features into the CNN framework has been unclear, with research into fusion techniques as well as distinguishing the spatial-temporal streams using optical flow.

Recurrent neural networks (RNNs) have proven to be powerful tools for tasks with a focus on temporal sequences, such as natural language processing and speech recognition. The research into using these networks for video classification is limited at best, and is an exciting marriage of spatial and temporal information extraction.

Related Work

Our research is inspired by similar work in the task of video classification via convolutional neural nets and by advances in state-of-the-art image classification.

Karpathy, *et al*. [5] present video classification taken to the extreme scale, with over 1 million videos and 487 classes. The most interesting aspect of this paper from our perspective is the notion of fusion. Karpathy, *et al*. experiment with a variety of architectures for 'fusing' network representations of frames over time. This is achieved by extending the convolutional filters over time and by concatenating separate convolutional outputs. The results of the paper show that combining these two temporal techniques into an architecture known as 'Slow Fusion' consistently performs better than either one of these individually and better than the single-frame classifiers. Additionally, this paper presents a transfer-learning result on UCF-101, the dataset we are working with, claiming that a transfer learned model performs better on UCF-101 than a standalone model. The more significant conclusion here is that there SPORTS-1M trained network has significant representational power, but for our purposes at the very least it shows that good results can be obtained on UCF-101 by transfer learning from a deep network trained on large amounts of data. This paper motivated our interest in video classification by emphasizing the need for practices to incorporate temporal information into the CNN framework so successful at image representation.

Simonyan, *et al*. [7] present a two-stream network for video classification on UCF-101. This two-stream approach focused on separating the notion of the spatial and temporal information presented in video by creating two networks each with separate classification scores which are combined to produce the final output. The spatial network took a single frame as input, similar to many CNNs used for image classification. The more interesting component in terms of our research is the temporal stream network. The temporal stream was made by creating 2 channels for each frame, one for horizontal optical flow between consecutive frames and one for the vertical optical flow. Although this utilizing of hand-made features to represent temporal information may be contradictory to many of the goals of 'deep-learning', this paper's results speak for themselves. The authors claim this two-stream network outperforms the Slow Fusion from [5]. This paper serves as a alternate way to represent temporal information in CNNs as opposed to [5] and encourages us to experiment with

new ways to do so.

Recent publications show promising results with the use of recurrent nets on video. Much of this project is based on a model architecture known as LRCN, demonstrated by Donahue, *et al*. in his paper *Long-term Recurrent Convolutional Networks for Visual Recognition and Description* [2]. This paper outlines a model which first performs feature extraction over a sequence of T inputs then passes the extracted features to a sequencing model. In particular, for the task of Action Recognition, Donahue presents feature extraction in the form of large, deep CNNs and sequence models in the form of two-layer LSTM models. In experiments on UCF-101, the LCRN models perform very well, giving state of the art results for that dataset. In addition to Action Recognition, Donahue presents similar models created for the task of both Image Description and Video Description, demonstrating that this framework of extraction and then sequencing can be generalized to many other tasks. This paper is the main inspiration for our project, as it demonstrated not only the flexibility and representational power of CNN 'percepts' but also the capabilities of RNNs and RNN variants for sequence modeling simultaneously.

In an extremely recent publication, Srivastava, *et al*. [9] present an unsupervised technique for creating video representations using LSTMs. Using an LSTM Encoder-Decoder scheme given 'percepts' from previously trained CNNs, the authors were able to make qualitative assessments of their model by having it predict either the future sequence given an input sequence or an input sequence given an encoded representation. They found that their model seems to contain sufficient information regarding appearance and motion from these experiments. Additionally, they were able to qualitatively assess their model on UCF-101. This was attempted with both 'percepts' from CNNs trained on RGB and temporal flow data. The authors were able to report significant improvements over baseline models, especially when training on less data per action class. Additionally they were able to compare their results to other state-of-the-art classifiers in the RGB, temporal, and combined space and found that their model was competitive if not better across these input spaces. This paper helped showed us that the use of specialized RNNs of many varieties can be effective for use with video.

Problem Statement

Our task is to take a short video clip and classify the action taking place within the video. More specifically, our task is to classify 10 consecutive frames of size $(320 \times 240 \times 3)$ into one of 101 classes ranging from 'Applying Eye Makeup' to 'Kayaking' to 'Yo Yo'. As such, we are training some function $h(x) : X \mapsto y$ where $X \in \mathbb{R}^{10 \times 320 \times 240 \times 3}$ and $y \in [1, 101]$.

Figure 1. Sample images taken from classes of UCF-101

Data Processing

One of the biggest challenges we faced while classifying videos was simply processing the data from .avi format to something we could feed to our neural networks. Converting an .avi to a tensor of floats results in incredible amounts of memory storage, to the point where we couldn't even store the entire dataset on our personal laptops or cloud instances. As a result, we heavily sub-sampled the data, giving us a representative window into each video. We sampled 10 frames evenly distributed over the first 2 seconds of each video, allowing us to have a tractable amount of tensor data for us to process and utilize effectively.

We locally processed the .avi videos and then uploaded the numpy tensor representations to our cloud instance. Resizing each frame to run using the pretrained CaffeNet model we were able to extract the last fully connected hidden layer to represent each frame, storing the new representation of each video to disk. This 4096 dimensional representation drastically downscaled our data and allowed us to efficiently train our models.

	Dimensions	**Storage**
Original Video Frames	$300 \times 320 \times 240 \times 3$	275MB
Subsampled Frames	$10 \times 320 \times 240 \times 3$	9MB
CNN Codes	$10 \times 4096 \times 1 \times 1$	160kB

In order to transfer learn from a pretrained Convolutional Neural Network to our video classification task we fed frames of each video through the BVLC Caffe Net Model

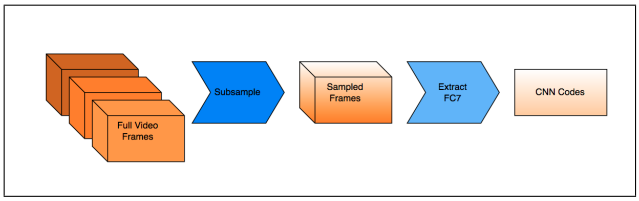

Figure 2. Data Processing Pipeline

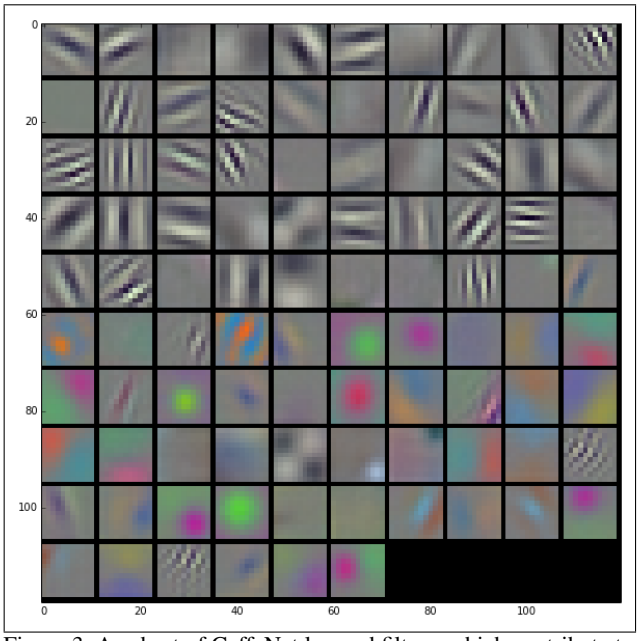

Figure 3. A subset of CaffeNet learned filters, which contribute to the generalizable representational power of large CNNs and allow for transfer learning

using the Caffe deep learning framework. We used the codes from the second to last fully connected layer (FC7) to get a 4096 dimensional representation of each frame. This is standard practice for many transfer learning tasks.

Given our limited data, training our models from scratch was not likely to produce the best possible results. As demonstrated in Yosinki, *et al*. [4] transfer learning from deep convolutional network trained on large amounts of data has been shown to be effective across image datasets. This is due to the fact that the image features extracted by convolutional networks which have been exposed to large image datasets generalize to many other domains. We used CaffeNet, an implementation of AlexNet[6], to extract features, allowing us to take advantage of features learned on the massive ImageNet dataset.

Models

We experimented with a variety of models, specifically hoping to encapsulate the temporal change between frames

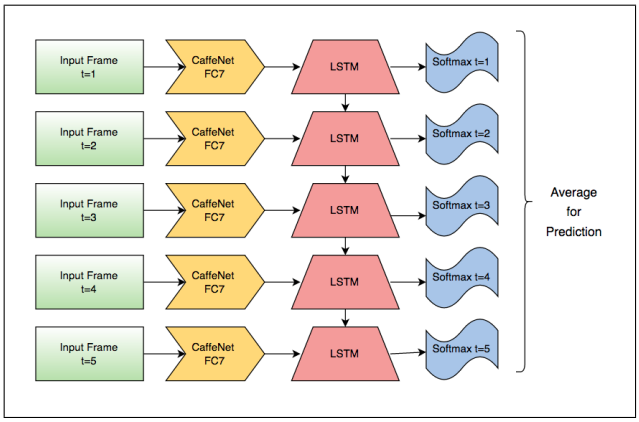

Figure 4. Our transfer-learned LSTM model

by using Recurrent Neural Networks.

Our first model, which we will refer to as the 'Average' Model, was a simple model which took the CNN codes of all 10 frames, averaged them, and fed them to a 2-layer fully connected neural network, utilizing the ReLu activation function and dropout. The intuition behind this model is that the CNN codes have enough representability alone to get a good idea of what is going on in the video clip, and averaging them incorporates all the frames evenly. This intuition also comes as a function of the dataset; many of the activities are spatially distinct enough that no temporal information is necessary to make decent estimates of activity.

Our second model we will refer to as the 'RNN' Model. Recurrent Neural Networks are run over sequences, with hidden layer activation at time t defined as a function of both the input at time t and the hidden layer from the previous time $t-1$. More formally, given an input sequence $x = \{x_1, ..., x_T\}$, the hidden layer activation at time t, denoted h_t, is defined as

$$h_t = a(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$

where W_{ih} is the weight matrix from input to hidden layer, W_{hh} is the weight matrix between hidden layers, b_h is the bias for the hidden layer, and a is the activation function (in this case ReLU). Once h_t is computed, the output for time t, denoted y_t, is defined as

$$y_t = W_{ho}h_t + b_o$$

where W_{ho} is the weight matrix from hidden layer to output and b_o is the bias for the output of the recurrent layer.

For our implementation of the RNN model, we fed the 10 CNN codes from the 10 frames to a 'vanilla' recurrent layer, calculated the probability of each class using Softmax at each layer, and then averaged the probabilities over each

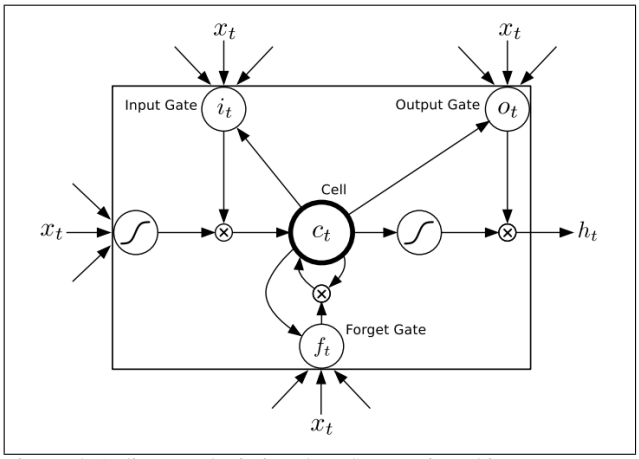

Figure 5. A diagram depicting the LSTM unit and its gates.

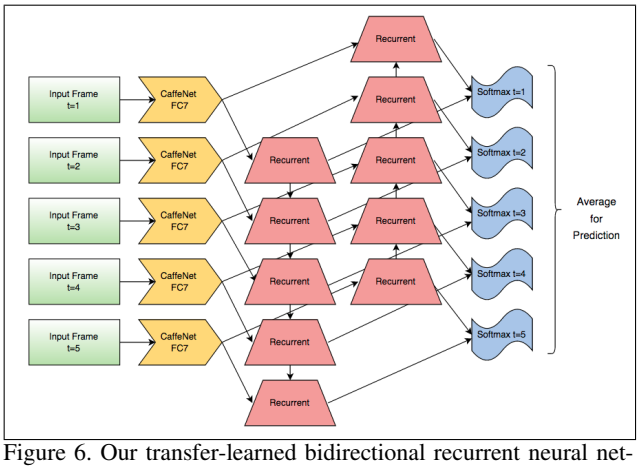

Figure 6. Our transfer-learned bidirectional recurrent neural network

time step. More formally, our prediction p ws defined as

$$p = \arg\max_i \frac{1}{T} \sum_{i=1}^{T} g(W_s y_i + b_s)$$

where W_s is our weight matrix from recurrent layer output to 101 class scores, b_s is the bias, and $g(z)$ is the softmax function. Thus we weight each timestep's class prediction equally, as opposed to alternatives such as only taking the last frame's prediction.

Our third model we will refer to as 'LSTM' Model. This model is exactly the same as the RNN model, but replacing the 'vanilla' RNN with a Long Short-Term Memory Layer. LSTMs have shown promise and are preferred to 'vanilla' RNNs for many tasks. LSTM units consist of three gates, known as the *input gate*, *forget gate*, and *output gate*. The inclusion of these gates is motivated by the regulation of information flow through the computational unit, allowing for more stable gradients and long-term sequence dependencies. The inner workings of LSTM is not the focus of this paper, yet many resources such as Grave's paper *Generating Sequences With Recurrent Neural Networks* [3] serve as good introductions to the computational unit and its usages. The hope for this model was that the complex internal mechanisms would outperform the 'vanilla' recurrent neural networks.

Our last model, which we will refer to as 'Bidirectional RNN', was a bidirectional recurrent neural network. Whereas our 'RNN' model only incorporated temporal information using the previous frame, our bidirectional network had recurrent layers which examined the next frame as well. Our CNN codes were fed to two layers, one working through the sequence forward in time and another layer which ran the sequence back in time. The forward layer's hidden layer activation h_t^f, was identical to our 'RNN' model's activation, while the backward layer's hidden layer activation h_t^b, was defined as

$$h_t^b = a(W_{ih}^b x_t + W_{hh}^b h_{t+1} + b_h)$$

The output of the recurrent layers, y_t, was simply the sum of y_t^f and y_t^b. Predictions p were calculated in the same way described for other models.

We experimented with other models that were either ineffective or not able to be experimented with given time and resource constraints. Specifically we attempted to do a one dimensional convolution on the collection of CNN codes (10 x 4096), this model was extremely unstable in training with validation accuracy fluctuating drastically regardless of parameters. It also achieved a much lower classification accuracy compared to our final models. We were not able to implement more complex models like bidirectional LSTM due to problems with hardware and limited computation time available.

Experiments

We ran our experiments on the UCF-101 Dataset for Action Recognition [8]. With our extracted CNN codes obtained via CaffeNet remaining architecture as implemented and trained in Theano and Lasagne. As a result, during training time, there was no finetuning of the CaffeNet. More explicitly, we only backpropagated into the layers found after the CNN codes, and did not alter the CNN codes nor the convolutional neural net which generated them. All our experiments were run on a Terminal.com GPU instance with 10GB of RAM.

Results

Our results can be found below. It should be noted that while validating our hyperparameters we found that the best validation set accuracy came when using hyperparameters

that massively overfit to our training set, often with training accuracies >95%.

Model	Validation Accuracy	Test Accuracy
Average	50.5%	50.0%
RNN	52.2%	49.1%
LSTM	46.7%	45.9%
Bidirectional RNN	51.1%	50.5%

Our results are not state of the art for the dataset nor the architecture employed. Nonetheless, we are happy with our results. Our RNN model and our Bidirectional RNN models were able to outperform the Average model. In fact, our results indicate similar performance enhancement over the Average model as found in the LRCN paper [2]. In the LRCN paper, when using only RGB inputs, Donahue, *et al*. found only 2% increase over their average model, which is similar to what our results indicate.

Conclusions

We found that our slight improvement in classification accuracy from using a recurrent architecture was due to taking advantage of differences between frames. The bidirectional recurrent architecture was able to take advantage of information from frames before and after it in the video. These differences resulted in different softmax scores at each time step and a stronger classification accuracy overall.

Our CNN code averaging model was a strong baseline, achieving nearly as good classification accuracy as the recurrent model. This was due to the dissimilarities between classes. The UCF-101 data set contains extremely diverse classes, from "ApplyingEyeMakeup" to "Surfing". Looking at a single frame of the video between very different classes is sufficient to make an accurate classification. As Figure 7 shows, the average CNN codes for 10 randomly selected classes are somewhat clustered showing the average CNN representation has strong classification power.

Much of our work process was impeded by the effectiveness and availability of hardware. We used Terminal.com to train our models to take advantage of a cloud instance with a GPU allowing us to drastically speed up computations. However, GPU instances were often not available when we required them for development. Peak hours of demand were not handled well and hindered us from implementing more complex models including bidirectional LSTM or double-layer RNNs. Additionally, the lack of cloud access prohibited us from more extensively cross-validating our current models. For example, our results with the Bidirectional RNN model are from a limited number of experiments searching near the parameters of previous models, less Terminal.com outages would have allowed for more extensive cross validation.

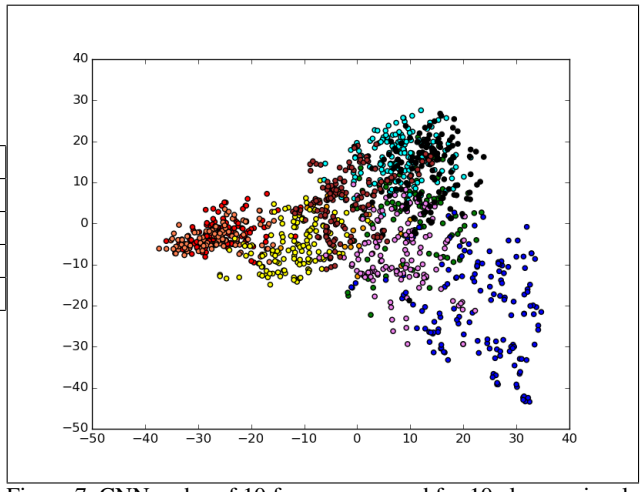

Figure 7. CNN codes of 10 frames averaged for 10 classes visualized with PCA

We feel positive about the techniques implemented in this paper and hope that our results could help to expand the field of video classification in the future.

Future Work

Expanding upon our current work we hope to have more extensive data augmentation. Our current feature extraction pipeline looks at the full resized image to extract its CNN code, but we hope to experiment with differing crops, flips, hue/contrast adjustment and the addition of random noise. In addition to augmenting our data we could improve our model by fine-tuning the convolutional network we use to extract our features. This would allow our frame representations to more accurately depict our classes.

We believe the techniques that we have implemented for video classification would show drastic improvement with larger datasets or ones with more similar classes. UCF-101 data has 100-140 videos per-class thus with 10000-14000 datapoints our models didn't have enough data to build an extensive representation of the space in which it was classifying over. The models with the highest test and validation accuracy suffered from high levels of overfitting caused by a relatively small training set compared to the number of parameters in our models. After 50-80 epochs the training accuracy was >90%. Training on a dataset like the Sport-1M[5] dataset could provide improved results.

Having a dataset which has a more focused domain could demonstrate a larger positive impact of our recurrent models. A dataset in which the CNN representation of a single frame is more similar between classes would be more dependent on the changes between frames to make a classification decision. Experimenting with a dataset such as the MICC-Soccer Actions 4[1] (no longer publicly avail-

able) could improve on previous action classification methods within that domain.

Including hand-engineered features such as optical flow could potentially increase the accuracy of our models. The CNN code representation of each frame captures much of the spatial information about a scene, but optical flow or other hand-engineered feature would give our models more information about the motion taking place. This could be incorporated as a dual stream model similar to the model described by Simonyan et. al.[7].

We used the final fully connected layer of CaffeNet (FC7) to get a representation of each frame. In order to extract features more relevant to our task we would like to experiment with feature extraction at different layers of the network. Lower level layers could have more spatially dependent values which could potentially help our classification accuracy. For example, Donahue, *et al*. found better results using FC6 in the LRCN paper [2].

Lastly, unsupervised techniques for video classification using RNNs, such as what is found in [9], seems promising. With unsupervised techniques, more training data becomes available as the need for labeling becomes unnecessary. Furthermore, unsupervised techniques can help develop generalizable temporal representations which can be used to further advance the performance of supervised models.

We hope to see the further usage of RNNs and similar sequencing models for video classification in the future academic landscape.

Acknowledgements

We would like to thank Andrej for his guidance and suggestions. We would also like to thank GitHub user *craffel*, whose open-source implementation of both Vanilla RNNs and LSTMs in Lasagne were modified for use in our experiments.

References

[1] M. Baccouche. Action classification in soccer videos with long short-term memory recurrent neural networks. 2010.

[2] J. Donahue. Long-term recurrent convolutional networks for visual recognition and description. 2015.

[3] A. Graves. Generating sequences with recurrent neural networks. 2014.

[4] J. C. J. Yosinski. How transferable are features in deep neural networks? 2014.

[5] A. Karpathy. Large-scale video classification with convolutional neural networks. 2014.

[6] A. Krizhevsky. Imagenet classification with deep convolutional neural networks. 2012.

[7] K. Simonyan. Two-stream convolutional networks for action recognition in videos. 2014.

[8] K. Soomro. Ucf101: A dataset of 101 human actions classes from videos in the wild. 2012.

[9] N. Srivastava. Unsupervised learning of video representations using lstms. 2015.