Report - Human Activity Recognition

Mohit Singhal
2018eeb1165
Harshit Sakhuja
2018eeb1044
Indian Institute Of Technology - Ropar

1 Introduction To Problem Statement

Action recognition task involves the identification of different actions from video clips (a sequence of 2D frames) where the action may or may not be performed throughout the entire duration of the video. This seems like a natural extension of image classification tasks to multiple frames and then aggregating the predictions from each frame. Despite the stratospheric success of deep learning architectures in image classification (ImageNet), progress in architectures for video classification and representation learning has been slower.

Why is this comparatively tough ?

1:- **Huge Computational Cost** :- A simple convolution 2D net for classifying 101 classes has just 5M parameters whereas the same architecture when inflated to a 3D structure results in 33M parameters. It takes 3 to 4 days to train a 3DConvNet on UCF101 and about two months on Sports-1M, which makes extensive architecture search difficult and overfitting likely.

2:-**Capturing long context** :- Action recognition involves capturing spatiotemporal context across frames. Additionally, the spatial information captured has to be compensated for camera movement.

3:-**No standard benchmark** :-The most popular and benchmark datasets have been UCF101 and Sports1M for a long time. Searching for reasonable architecture on Sports1M can be extremely expensive.

2 Literature Survey

Before deep learning came along, most of the traditional CV algorithm variants for action recognition can be broken down into the following 3 broad steps:

1:-Local high-dimensional visual features that describe a region of the video are extracted either densely as in this paper(Action recognition by dense trajectories by Wang et. al.) or at a sparse set of interest points as in the papers(On space-time interest points by Laptev and

Behavior recognition via sparse spatio-temporal features by Dollar et al)

2:-The extracted features get combined into a fixed-sized video level description. One popular variant to the step is the bag of visual words (derived using hierarchical or k-means clustering) for encoding features at video-level.

3:-A classifier, like SVM or RF, is trained on bag of visual words for final prediction.

Of these algorithms that use shallow hand-crafted features in Step 1, improved Dense Trajectories which used densely sampled trajectory features used to be **state-of-the-art** (Action Recognition with Improved Trajectories). The average accuracy over all classes obtained using this algorithm were :- 64.3% on Hollywood 2 dataset, 57.2% on HMDB51 dataset, 91.1% on Olympic sports dataset and 91.2% on UCF50 dataset.

Two breakthrough research papers were released in around 2014 which form the backbone for all the papers as their approaches are variants of the approaches discussed in these two papers.

Approach 1: Single Stream Network

In this work [Large-scale Video Classification with Convolutional Neural Networks], the authors - Karpathy et al. use a baseline single-frame pre-trained CNN and its extensions in time according to different types of fusion.

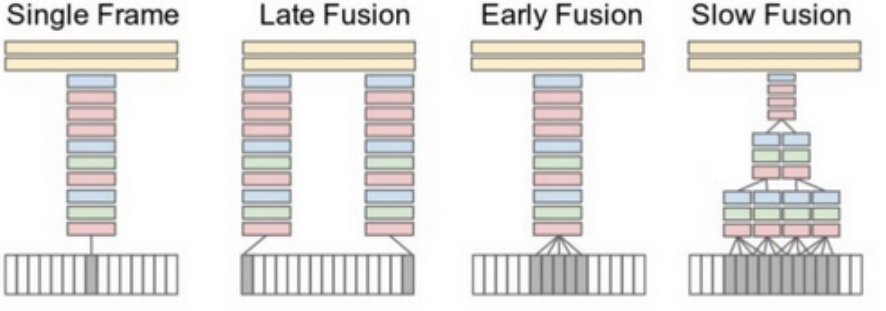

Fig 1: Fusion Ideas

As can be seen in Fig 1, the consecutive frames of the video are presented as input in all setups. Single frame uses single architecture that fuses information from all frames at the last stage in the following three ways.

The Early Fusion extension combines information across an entire time window immediately on the pixel level by convolving over 10 frames. The early and direct connectivity to pixel data allows the network to precisely detect local motion direction and speed.

The Late fusion extension uses two nets with shared parameters, spaced 15 frames apart, and then merges the two streams in the first fully connected layer. In this approach, neither

single frame tower alone can detect any motion, but the first fully connected layer can compute global motion characteristics by comparing outputs of both towers.

The Slow Fusion model is a balanced mix between the two approaches that slowly fuses temporal information throughout the network such that higher layers get access to progressively more global information in both spatial and temporal dimensions.

For final predictions, multiple clips were sampled from the entire video and prediction scores from them were averaged for final prediction. The best results on UCF-101 for various Transfer Learning approaches using the Slow Fusion network were obtained by fine tuning the top 3 layers to obtain an average accuracy of 65.4 %. The results were significantly worse as compared to state-of-the-art hand-crafted feature based algorithms. There were multiple reasons attributed for this failure:

1:-The learnt spatiotemporal features didn't capture motion features.

2:-The dataset used by them being less diverse, learning such detailed features was tough

Approach 2: Two Stream Networks

In this pioneering work [June 2014] by Simmoyan and Zisserman, the authors build on the failures of the previous work by Karpathy et al. Given the toughness of deep architectures to learn motion features, authors explicitly modeled motion features in the form of stacked optical flow vectors. So instead of single network for spatial context, this architecture has two separate networks - one for spatial context (pre-trained), one for motion context. The input to the spatial net is a single frame of the video. Authors experimented with the input to the temporal net and found bi-directional optical flow stacked across for 10 successive frames was performing best. The two streams were trained separately and combined using SVM. Final prediction was same as previous paper, i.e. averaging across sampled frames. The Mean accuracy (over three splits) observed was 88% on UCF-101 and 59.4% on HMDB-51 .

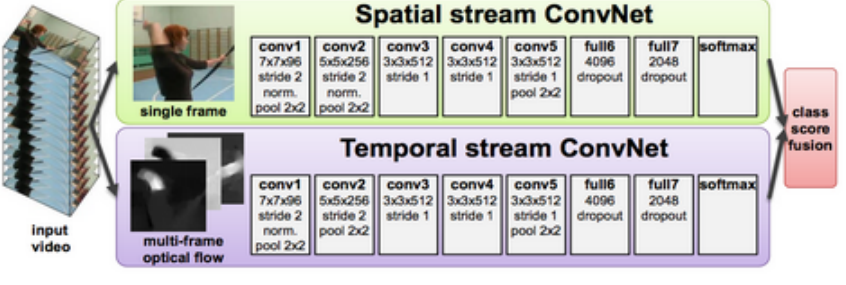

Fig 2: Two Stream Architecture

Though this method improved the performance of single stream method by explicitly capturing local temporal movement, there were still a few drawbacks:

Because the video level predictions were obtained from averaging predictions over sampled clips, the long range temporal information was still missing in learnt features.

Since training clips are sampled uniformly from videos, they suffer from a problem of false label assignment. The ground truth of each of these clips are assumed to be the same as ground truth of the video which may not be the case if the action just happens for a small duration within the entire video.

The method involved pre-computing optical flow vectors and storing them separately. Also, the training for both the streams was separate implying end-to-end training on-the-go is still a long road.

Following papers which are, in a way, evolutions from the two papers (single stream and two stream) are listed below:-
1:- LRCN
2:- C3D
3:- Conv3D Attention
4:- Two stream Fusion
5:- TSN
6:- ActionVlad
7:- Hidden Two Stream
8:- I3D
9:- T3D

The recurrent theme around these papers can be summarized as follows. All of the papers are improvisations on top of these basic ideas.

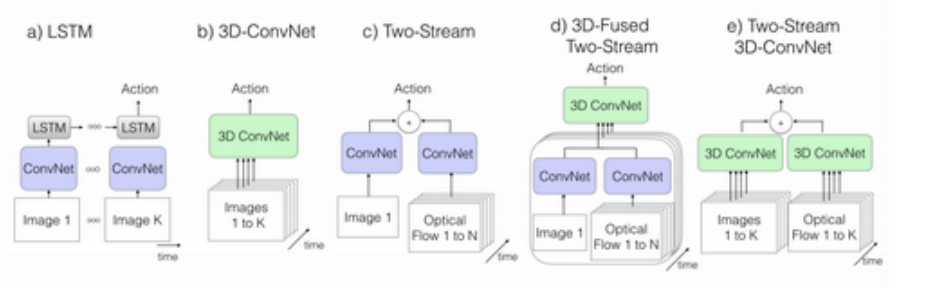

Fig 3: Recurrent Theme Across Papers

For each of these papers, we list down their key contributions.

LRCN
Building on previous work by using RNN as opposed to stream based designs Extension of encoder-decoder architecture for video representations End-to-end trainable architecture proposed for action recognition

C3D
Repurposing 3D convolutional networks as feature extractors Extensive search for best 3D

convolutional kernel and architecture Using deconvolutional layers to interpret model decision

Conv3D and Attention

Novel 3D CNN-RNN encoder-decoder architecture which captures local spatiotemporal information Use of an attention mechanism within a CNN-RNN encoder-decoder framework to capture global context

TwoStreamFusion

Long range temporal modeling through better long range losses Novel multi-level fused architecture

TSN

Effective solution aimed at long range temporal modeling Establishing the usage of batch normalization, dropout and pre-training as good practices

ActionVLAD

Learnable video-level aggregation of features End-to-end trainable model with video-level aggregated features to capture long term dependency

HiddenTwoStream

Novel architecture for generating optical flow input on-the-fly using a separate network

I3D

Combining 3D based models into two stream architecture leveraging pre-training Kinetics dataset for future benchmarking and improved diversity of action datasets

T3D

Architecture to combine temporal information across variable depth Novel training architecture technique to supervise transfer learning between 2D pre-trained net to 3D net

Available datasets

In recent years, more and more datasets dedicated to human action and activity recognition have been created. The use of these datasets allows us to compare different recognition systems with the same input data. The various datasets used (used either for training/ fine-tuning or as a benchmark) in the before mentioned approaches for action recognition are:-

1) HMDB51: A Large Video Database for Human Motion Recognition with 6766 videos and 51 action classes
2) UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild with 13320 videos and 101 action classes
3) The YouTube Sports-1M Dataset with 1100000 videos and 487 action classes
4) Kinetics is a large-scale, high-quality dataset of YouTube video URLs which include a diverse range of human focused actions . The total number of videos are 500000 with 600 action classes

3 Preliminary Study

We studied the architecture and the underlying concepts in the paper Long-term Recurrent Convolutional Networks for Visual Recognition and Description by Donahue et al., authors used the idea of using LSTM blocks (decoder) after convolution blocks(encoder) and using end-to-end training of entire architecture. They also compared RGB and optical flow as input choice and found that a weighted scoring of predictions based on both inputs was the best.

During training, 16 frame clips are sampled from video. The architecture is trained end-to-end with input as RGB or optical flow of 16 frame clips. Final prediction for each clip is the average of predictions across each time step. The final prediction at video level is average of predictions from each clip.

	Single Input Type		Weighted Average	
Model	RGB	Flow	$1/2, 1/2$	$1/3, 2/3$
Single frame	67.37	74.37	75.46	78.94
LRCN-fc_6	**68.20**	**77.28**	80.90	**82.34**

Fig 4: Comparing single frame models to LRCN networks for activity recognition on the UCF101 dataset, with RGB and flow inputs.

Even though the authors suggested end-to-end training frameworks, there were still a few drawbacks

1)False label assignment as video was broken to clips
2)Inability to capture long range temporal information
3)Using optical flow meant pre-computing flow features separately

4 Our Approach

The above LSTM based approach is **extremely resource hungry and time consuming**. Also, we require a lot of video data to train this network.
But if we look at the architecture, they are using CNN for generating a representation of every image and then passing these representations to a LSTM to take into the account the temporal nature of a video i.e subsequent frames in a video generally have similar semantic contents.
What if we remove these LSTM's ? We can just classify every frame on the basis of spatial information only. But this way we will loose all the temporal information and will also observe a great amount of flickering in the predictions i.e. a lot of noisy predictions .
Inspiration
Taking inspiration from Adam optimizer where we do not update in the direction of the current gradient as it is very noisy because that gradient is just computed on a mini-batch . So,what we do there is take an **exponentially decaying weighted average** of previous gradients and combine them to make our weight update , this makes the weight updates smooth .

Here also we will classify on a single frame and then will take an average of the predictions of the last few frames to give our final prediction on this particular frame . There is a slight difference in terms of averaging method used in the Adam optimizer and that we are using. Unlike Adam, we will just take the "simple" average of last few predictions of our classifier neural network. **We are not making any assumption that there is strong correlation between the immediate next frames thus refraining from using exponentially decaying weighted average**.Also, If we use exponentially decaying weighted average our predictions can fail when there are two or more activities to classify in a single video.

Thus, our rolling average technique will help us to reduce flickering and according to our results this approach works fine on a decent number of videos.
How many previous frames should we consider ? Well that's a hyper-parameter for which we will do a coarse to fine search . We tried three different values of this hyperparameter (1, 5 and 10) and observed the least flickering in the case when it was set to 10.

4.1 CNN model

So our video classification task can now be treated as a simple image classification task. This can be done by training a reasonably deep convolution neural network. But rather than training the whole neural network (which requires a lot of image data and is quite time consuming) we use the concept of transfer learning. We are not using the feature extraction based transfer learning where a pretrained convolutional neural network will be used as a feature extractor and on top of that we train a very simple dense neural network. Instead, we are fine-tuning the Resnet50 pre-trained on ImageNet. We are removing the fully connected nodes at the end of the network (where the actual class label predictions are made) and then replacing them with our own freshly initialized layers. To ensure that none of the robust features previously learned by the ResNet are destroyed we freeze all the previous convolution layers and just train the fully connected heads added by us.

average_pooling2d_1 (AveragePoo	(None, 1, 1, 2048)	0	activation_49[0][0]
flatten (Flatten)	(None, 2048)	0	average_pooling2d_1[0][0]
dense_1 (Dense)	(None, 512)	1049088	flatten[0][0]
dropout_1 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 8)	4104	dropout_1[0][0]

Total params: 24,640,904
Trainable params: 1,053,192
Non-trainable params: 23,587,712

Fig 4:Summary of the layers that we added on the base model from Resnet50 (Information of trainable and non-trainable parameters refers to the whole model(Base$_{model} + HeadModel$)

4.2 Dataset Used:-

Event Dataset
Since we are just using a CNN we took a sports images dataset so that we can classify various sports actvities performed by humans .

Specifics of Dataset

This dataset contains 8 sports event categories: rowing (250 images), badminton (200 images), polo (182 images), bocce (137 images), snowboarding (190 images), croquet (236 images), sailing (190 images), and rock climbing (194 images). Images are divided into easy and medium according to the human subject judgement. Information of the distance of the foreground objects is also provided for each image.

How to obtain the image information? E.g. Easy-Close-Rowing-504.jpg

"Easy" indicates the challenge level.
"Close" indicates the distance of the foreground objects
. "Rowing" indicates the event class.
"504" indicates the image index in the raw dataset, you can ignore it.

4.3 Results

After fine-tuning the ResNet50 on the sports dataset we attain about 90 percent on the test data set which can be seen from the below plot.

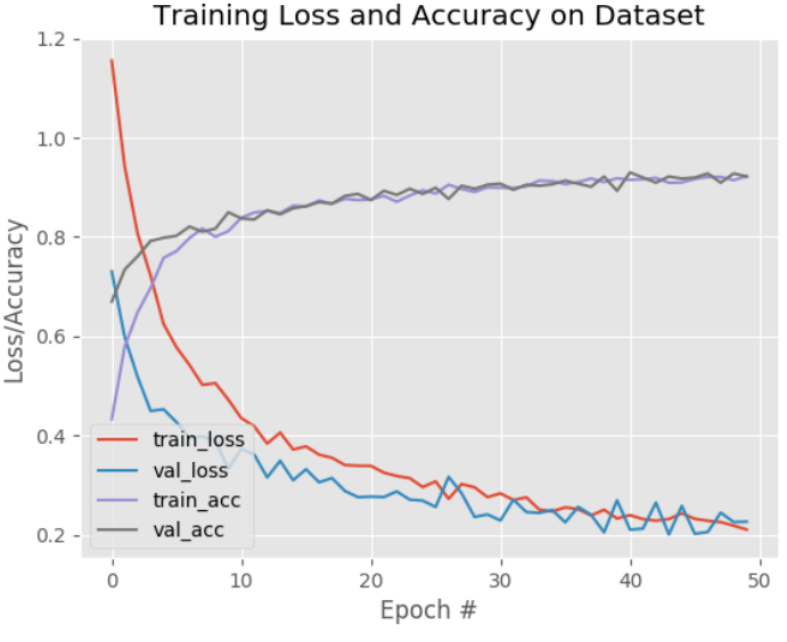

Fig 5: Train and validatation accuracy on sports dataset

Due to our inability to download huge video data-sets, we have used some videos of our own to test our method. Also , since we don't have any standard benchmark data-sets we cannot use any confusion matrix or classification report to compare our algorithm with the existing ones.

Fig 6: Sample predictions on two frames of different videos. Both the videos have been recorded using the cameras of our mobile phones For seeing performance of model on other videos refer repo1 or repo2

5 Conclusion

We got an overview of the existing approaches after the literature survey , we also studied the algorithm of the paper (Long Term Recurrent Convolutional Networks for Visual Recognition and Description). We used an approach where we replaced LSTM with temporal averaging , lockdown inspired us to come up with an approach which requires less computation and still performs reasonably well on some videos. As part of the result , due to unavailability of benchmark datasets and the existing datasets (Kinetics and UCF) were too big to work as per the resources we were having thus we didn't made any comparisons rather showed some frames from the videos on which we ran our model .