Seeing the Arrow of Time

Abstract

In this project we study the determination of time direction of video being played. A Siamese network is trained and tested on UCF-101 dataset. The network is then slightly modified to be capable of highlighting regions which activate class labels. We finally evaluate the models on YouTube dataset and achieve similar results to previous work.

1. Introduction

Often times we see on the internet fun videos with time reversed: a exploded balloon reassembles, poured beverage goes back to the cup, *etc.*. We are amused by such videos because the time reversal apparently violates the intrinsic chronological order of events. Therefore, there arises a natural question: *can we train machines to tell if a video clip is being played* forward *or* backward?

Clearly, it makes little sense to tell for example if a video of an oscillating pendulum is reversed or not. Therefore, we shall hereafter restrict our study on videos with intrinsic asymmetrical chronological order.

1.1. Related work

Pickup *et al*. [8] address the problem and use in particular *optical flows* to encode motions between consecutive frames. A visual vocabulary is learned on optical flows and a SIFT-like descriptor is produced and aggregated for each video. A binary linear SVM classifier is finally trained on the descriptor.

Misra *et al*. [7] study a similar problem which consists of telling if frames are shuffled. Formally, three frames a, b and c are considered unshuffled if either $t_a < t_b < t_c$ or $t_c < t_b < t_a$ holds and shuffled otherwise. A *Siamese* network[4] stacking three CaffeNet is trained to this end.

1.2. Contribution of the project

This project makes several contributions to the problem. Firstly, we review the optical flow approach adapted in [8] and proposes a more convincing measure, the *confusion matrix*, to evaluate models. A similar method using Convolutional neural network (CNN) instead of bag-of-words is developed as baseline. Secondly, we train a Siamese network similar to that of [7], but is self-contained and without external dependencies, to detect the time of arrow. Finally, we adapt *Global average pooling* to highlight the regions which activate class labels.

The code and instructions for reproducing results can be found at `https://github.com/BoeingX/seeing-the-arrow-of-time`.

2. Datasets and preprocessing

2.1. Datasets

Two datasets are used in the project. The first dataset is the YouTube dataset employed in [8] consisting 180 high-quality video downloaded from `YouTube`. The dataset contains 155 forward videos and 25 backward videos. The second dataset is the split 1 of the UCF-101 dataset[9]. This is a huge dataset consisting 13k videos from 101 action categories.

2.2. Preprocessing

2.2.1 YouTube dataset

For the YouTube dataset, it is split in three ways (note A, B and C) into training and validation sets. As in [8], the three partitions are such that each video appears exactly twice as training data and once as validation data. For validation set, the ratio between forward and backward videos are $51 : 9, 52 : 8$ and $52 : 8$ respectively. The dataset is then *bootstrapped* to four copies: (A) original video, (B) left-right flip of (A), (C) time reversal of (A) and (D) left-right flip of (C) as suggest in [8].

Dense optical flows are then calculated as in Figure 1. Given three consecutive frames, the first and third frame are *registered* with respect to the second in order to eliminate camera shake. Horizontal and vertical components of dense optical flow between registered frames are then calculated by Farneback algorithm[5] using `OpenCV`[3]. Horizontal and vertical flows are normalized between 0 and 255 and saved as gray-scale jpeg images.

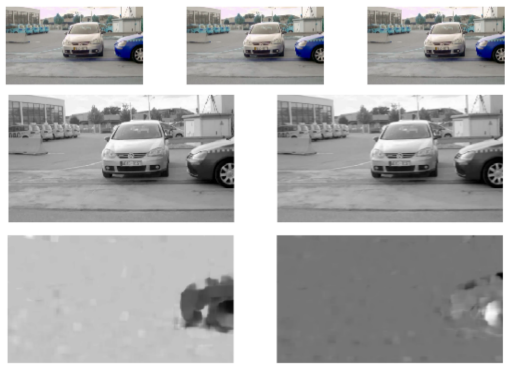

Figure 1: Generation of optical flow. Top: three consecutive frames. Middle: the first and third frame registered with respect to the second frame. Bottom: horizontal and vertical components of dense optical flow between registered frames.

2.2.2 UCF-101 dataset

`ffmpeg` is used to extract frames from videos and save them as jpeg images. Misra *et al*. [7] provide a list of image triples with maximum motion (in terms of optical flow norm) and their corresponding labels (0 for shuffled and 1 for unshuffled). We then remove all triples with label 0 (since we are not interested in unshuffled images) and assign remaining triples 0 and 1 according to the temporal order (0 for backward and 1 for forward). In total, we have 151760 triples. We keep 150000 triples as training data and 1760 as validation data. Since for a given triple (a, b, c), the triple (c, b, a) appear also in the list, we have a balanced dataset. The triples are reshaped to 256×256 and stored to a `lmdb` database.

3. Baseline method

It is known [7] that the higher-level layers of a CNN are very effective generic features. We thus propose a simple baseline method employing a pretrained CNN as feature extractor and evaluate the method on YouTube dataset.

3.1. Setup

We use CaffeNet, a slight modification of AlexNet[6], pretrained on ImageNet, as primary convolutional neural network. For each video, the horizontal and vertical optical flow with maximum magnitude is taken as input to CNN. Since the CNN is trained on 3-channel image, we stack the horizontal and vertical flow as well as the orientation of optical flow (normalized between 0 and 255) as a “RGB” image. It is then centered by subtracting 128 at each channel.

The outputs of `fc7` layer are fed to a Gaussian-kernel SVM to produce a binary prediction. The hyper-parameter C (in our experiment 0.001) in SVM is determined by grid search using 3-fold cross-validation on training data. Since each video is bootstrapped to four copies, we ensemble the result as sign$(A + B - C - D)$ as the final prediction.

3.2. Results

For the validation set A, B and C, the baseline method achieves an accuracy 75.00%, 76.67% and 76.67% respectively. The simple baseline method thus results comparable accuracy to the bag-of-words approach adapted in [8].

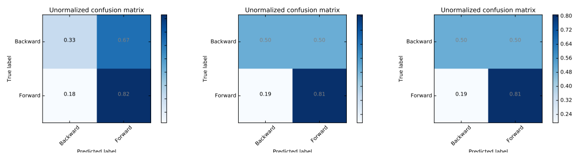

Figure 2: Confusion matrix on validation set A, B and C of baseline method.

However, keeping in mind that the YouTube dataset is highly imbalanced, the accuracy is not convincing. To shed light on this issue, just consider a classifier producing always *forward*, we would have accuracy 85%, 86%, 86% on validation sets. Therefore, it makes more sense to consider the *confusion matrix*. Figure 2 shows the confusion matrix of baseline method. While it performs well on forward videos, it behaves randomly on backward videos.

4. Siamese network

In this section, we propose a more elaborate method by directly training a Siamese CNN producing binary labels.

4.1. Setup

We use the same Siamese network as proposed by Misra *et al*. [7]. Their source code being open source, we choose however to implement the network from scratch. The reason is that the existing code has many external dependencies in particular *Python layers* which makes the network structure unclear and the modification difficult.

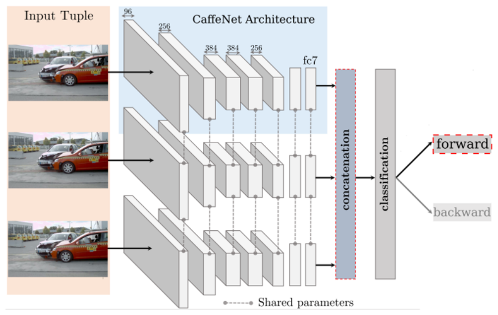

Figure 3: Siamese network consisting three CaffeNet sharing parameters. Diagram adapted from [7].

Figure 3 illustrates the structure of Siamese network consisting of three CaffeNet sharing parameters. The output of `fc7` layers are *concatenated* onto a $4,096 \times 3$ vector and a full connected layer `classification` produces binary label.

The network is initialized by a CaffeNet trained on ImageNet and is then trained on UCF-101 dataset. The network is trained during two days on one `g2.2xlarge` instance of AWS by batches of 64 for $100,000$ iterations (about 40 epochs).

4.2. Results

The Siamese network achieves 94.90% test accuracy[1] on $1,760$ validation videos.

Since YouTube dataset is rather different from UCF-101, it does not make sense to use the trained model out-of-box. Note however that it is unlikely to perform fine-tuning because of too few training samples ($n \times 120$ where n is number of triples per video is largely insufficient). We therefore take the `concat` layer, which concatenates three `fc7` layers, as input to a linear SVM as before. We achieve accuracy $71.67\%, 63.33, 63.33\%$ on validation set A, B and C. Figure 4 shows the confusion matrices. Surprisingly, we see that the method does not outperform baseline method. The reason might be the fact that the feature dimension is three times bigger than the baseline method but the number of samples remain the same.

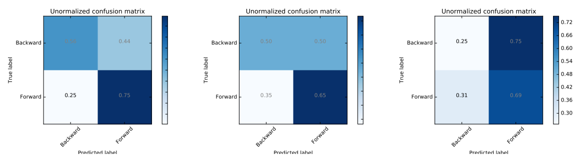

Figure 4: Confusion matrix on validation set A, B and C of Siamese approach.

5. Class activation mapping

In previous sections we developed methods to detect if a video is playing forward or backward. Can we do better? *Is it possible to highlight which part of image* activates *the label*?

Zhou *et al.* [10] study the *global average pooling*(GAP) and show that by removing all fully connected layer and using GAP followed by a single fully connected layer. Without modifying dramatically the existing network structure and degrading significantly the performance, it is possible to highlight regions which actually activate class labels.

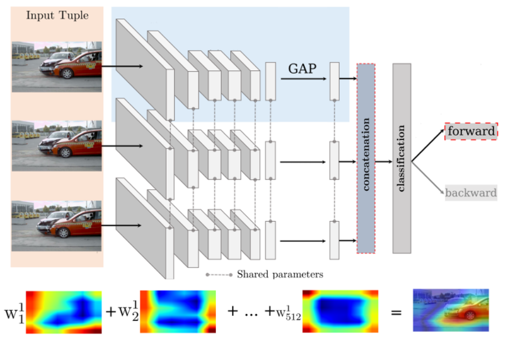

Figure 5: Class activation mapping. Adapted from [7].

5.1. Setup

We begin by taking the Siamese network shown in Figure 3 studied in previous section. Following [10], all layers after `pool5` are removed and two convolution layers `conv6` and `conv7`, followed by GAP layer `global_average_pool8` are added. They are then *concatenated* to form a long vector and a final fully connected layer `fc9` is used to produce binary labels.

The output of `fc9` layer is firstly sliced into 3 vectors (corresponding to 3 networks in Siamese network). These vectors are then used to generate class activation map as weighted sum of `conv7` layer[2] for 3 input images. Figure 5 shows the diagram of the network and the generation of (one) class active mapping.

The `caffemodel` fitted after $80,000$ iterations in previous section is used as initialization of the network. It is trained similarly to the original siamese network. However, the convergence is much faster and we stopped the fine-tuning after $18,000$ iterations.

5.2. Results

The Siamese network achieves 90.78% test accuracy on $1,760$ validation videos. The removal of fully connected layers degrade slightly the performance of network as expected.

Figure 6 shows examples of class activation maps from YouTube dataset. As we can see, the global average pooling is indeed capable of localizing discriminative region of each class.

6. Discussion

In this project we train a Siamese network for determining the direction of video being played and shows impressive accuracy on UCF-101 dataset. We also develop a class activation mapping to show regions which actually activate

[1]Recall that the validation set is balanced.

[2]A *upsampling* is used to result heatmap of same dimension to the input.

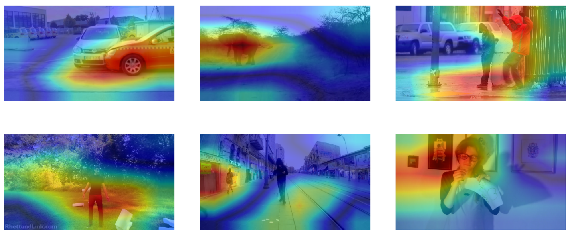

Figure 6: Examples of class activation mappings from YouTube dataset. Top: forward; bottom: backward.

a label. Though performing very well on UCF-101, it does not produce satisfactory results for YouTube dataset. In the future, on can adapt the two models and fine tune them on datasets similar to the YouTube dataset (but larger enough) to evaluate their performance.

Acknowledgments This project benefited from GitHub Education[2] and AWS Educate[1] program.

References

[1] Aws education. `https://aws.amazon.com/education/awseducate/`. Accessed: 2017-01-12.

[2] Github education. `https://education.github.com/`. Accessed: 2017-01-12.

[3] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[4] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *In NIPS Proc*, 1994.

[5] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016.

[8] L. C. Pickup, Z. Pan, D. Wei, Y. Shih, C. Zhang, A. Zisserman, B. Schölkopf, and W. T. Freeman. Seeing the arrow of time. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[9] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[10] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.