Low-Complexity Video Classification using Recurrent Neural Networks
Extended Abstract

Ifat Abramovich*, Tomer Ben-Yehuda* and Rami Cohen
Signal and Image Processing Lab (SIPL)
Andrew and Erna Viterbi Faculty of Electrical Engineering
Technion - Israel Institute of Technology
Technion City, Haifa 3200003, Israel
Email: ifatabr@gmail.com, tomerby2@gmail.com, rc@technion.ac.il

I. Introduction

YouTube-8M [1] is a large-scale video dataset containing more than 8 million labeled YouTube videos. Announced by Google in September 2016, it is the first video classification dataset whose size and diversity are comparable to existing image datasets. The labels of the videos in the dataset are machine-generated, from a vocabulary of 3862 categories. A video may have multiple labels, where on average there are 3 labels per video. Although YouTube-8M dataset has a great potential and had already advanced research on video classification, there are still difficulties arising from computational challenges related to the size of the dataset, from noisy and incomplete labels, and from higher complexity compared to image datasets due to the temporal dimension.

In this work, we train several recurrent neural networks for video classification using a subset of YouTube-8M dataset. We focus on computational complexity analysis, and provide experimental results aimed at enabling the efficient development of low-complexity network architectures for both fast training and inference. Similarly to [1], our approach is based on extracting frame-level features using the powerful Inception-V3 architecture. We use recurrent neural networks with either long short-term memory (LSTM) or bidirectional LSTM layers to capture temporal information. Finally, we evaluate the performance of a transfer-learning approach using an architecture trained on only a part of our dataset. We demonstrate that transfer learning offers a good complexity-performance tradeoff for the video classification task.

II. Methods

A. Dataset and preprocessing

We selected 10 categories from YouTube-8M and downloaded 1,000 randomly-selected videos for each category (resulting in a dataset of 10,000 videos). The categories are: *Athlete, Car, Cartoon, Musical ensemble, Pet, Aircraft, Cosmetics, Food, Nature* and *Toy*. For reduced complexity, the videos were down-sampled to 1FPS. Up to the first 360 frames were extracted from each video, with average number of 246 extracted frames and average length of 4.33 minutes.

Following a similar approach taken by the creators of YouTube-8M, we process each frame using an Inception-V3 network pretrained on ImageNet (with 82.8% accuracy) to obtain a 2048-dimensional feature vector. The features are extracted from the activations of the last hidden layer. Inception-V3 consists of *inception* modules that apply multiple convolutions of varying kernel sizes in parallel to the input. This way, features at multiple scales are extracted at each layer, resulting in an efficient and compact representation of the frames. For each video, we concatenate the extracted feature vectors in chronological order to obtain a frame-level representation in the feature space. In this representation, each video is a matrix with feature vectors as columns, where the number of columns corresponds to the number of frames.

B. Network architecture

Figure 1 shows the architecture of our video classification network. The input to the network are feature sequences, extracted from video frames as described in Section II-A. We consider different kinds of recurrent neural network (RNN) architectures, which are based on either LSTM or Bidirectional LSTM layers. For reduced computational complexity and for a low-delay inference, we use at most 3 LSTM/BiLSTM layers in our experiments. The output of the last LSTM/BiLSTM layer is mapped to a probability distribution over the categories using a fully-connected layer followed by softmax.

The use of recurrent networks is motivated by their ability to capture temporal information, which is highly important for the task of video classification. Compared to a possible approach of using feed-forward networks with 3D convolutions (i.e., operating on the temporal axis as well), RNNs with 2D convolutions offer a more efficient and less complex approach for learning spatiotemporal features. As a category of a video may depend on long-term temporal dependencies among frames, we use LSTM/BiLSTM layers that are better suited for capturing such dependencies. We examine as well

*Contributed equally.

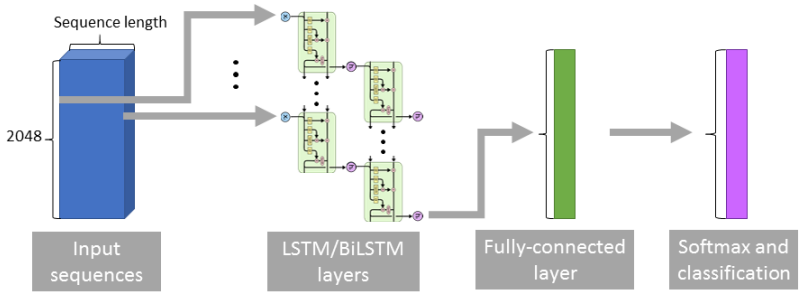

Fig. 1: Our network architecture. The LSTM/BiLSTM part is used with permission from C. Olah.

whether the additional complexity of BiLSTM layers results in a performance gain compared to LSTM.

Training deep networks on video datasets is typically highly time consuming. Thus, we consider a transfer learning setting where the network is initially trained using data from 5 categories and then fine-tuned using the data from the remaining 5 categories. In the fine-tuning stage, the fully-connected layer is trained. This results in a significant saving in training time.

C. Training

We partition the data into train and test sets, such that 80% of the videos in each category are used for training and the remaining 20% for testing. We limit the number of epochs in each training process to 500. During training of LSTM or BiLSTM layers, the sequences in each mini-batch are padded to the length of the longest sequence in the mini-batch. To reduce padding and thereby improve training performance, the sequences are sorted by sequence length, so that sequence lengths in each mini-batch are approximately the same. In the training process, we use stochastic gradient descent with momentum. We use learning rate value of 0.01, momentum value of 0.9 and mini-batch size of 50. The hardware used is Intel Core i7-7700K 4.20GHz CPU, NVIDIA GeForce GTX 1080 Ti GPU and 32GB RAM.

III. Performance Evaluation

We trained several network architectures using datasets of different sizes. We started with 5 categories (*Athlete, Car, Cartoon, Musical Ensemble* and *Pet*) and then used all 10 categories (adding: *Aircraft, Cosmetics, Food, Nature* and *Toy*). In addition, we performed transfer learning from 5 to 10 categories. The evaluation method we used is the *hit*@1 metric, defined as the fraction of test samples whose predicted top category concurs with the ground truth.

For performance evaluation, we used confusion matrices and precision-recall graphs. A confusion matrix visualizes the number of times each category was correctly labeled or mislabeled as another category. Precision is the fraction of examples labeled correctly as belonging to a certain category relative to the total number of examples labeled as this category. Recall is the fraction of examples labeled correctly as belonging to a certain category relative to the total number of examples taken from this category. A larger area under a precision-recall graph corresponds to better performance.

Architecture	***hit*@1**
2 LSTMs, 1024 units	89.43%
2 BiLSTMs, 1024 units	89.13%
Transfer learning, 2 BiLSTMs, 1024 units	89.68%
2 BiLSTMs, 512 units	86.79%
3 BiLSTMs, 682 units	87.49%
3 BiLSTMs, 512 units	87.74%
3 BiLSTMs, 256 units	87.84%

TABLE I: *hit*@1 results, 10 categories.

In our preliminary experiments, we evaluated the performance of an architecture with two LSTM/BiLSTM layers trained using 5 categories only. Similar *hit*@1 results were obtained: 96.92% and 96.62% for LSTM and BiLSTM, respectively. A major advantage of BiLSTM is its faster training convergence; the BiLSTM architecture converged in only 100 epochs, compared to more than 200 epochs for the LSTM architecture. In our hardware setting, 3.5 hours were required for the BiLSTM architecture compared to 6.5 hours for the LSTM architecture. In light of this significant saving, we moved to using BiLSTM exclusively in our remaining experiments.

Table I lists the results of our different trained architectures (based on BiLSTM layers) on 10 categories. It is shown that all the architectures achieved good results, even when the number of hidden units in the BiLSTM layers was decreased from 1024 to 256. In Figure 2, we show *hit*@1 on the test set as a function of the training epoch for various settings. There are two observations that we make. First, transfer learning generalizes faster and in a smoother manner compared to full training (i.e., training a network on 10 categories from scratch). In addition, increasing the number of BiLSTM layers to 3 while preserving the total number of units (i.e., 2 BiLSTMs with 1024 units compared to 3 BiLSTMs with 682