Action Recognition Report

March 2020

Action: Shouting

Josiah Coad

1. Background

Action recognition is the identification of actions, normally human actions, in the physical world through visual signals and audio signals. It would seem that action recognition is a natural extension of image recognition which deep learning has excelled at for years. Holver, there are some difficulties in making the extension. Large computational costs are a major challenge moving forward as videos have a temporal dimension which can greatly increase the computational complexity of learning. Most of the research in action recognition has been in how to effectively take advantage of the temporal aspect of videos.

Significant advances have been made through applying convolutional deep neural networks to action recognition. Much of the literature currently uses pre-trained CNN architectures for faster convergence. Notable advances include the following. In 2014, this foundational work came out that looked at fusing temporal approaches with traditional CNN. Namely, the authors compared various approaches of reading in several frames of the video at once to the CNN. Several feature engineering techniques proved to be useful including spatial flow in this work. Soon, several more works appeared that used 3D convolutions and RNNs to capture temporal information. These techniques have all been combined in various ways, finding improvements in accuracies on benchmark datasets. A more indepth background exposition can be found here.

2. Dataset

In this work, I use the Kinetics 700 dataset, released by DeepMind. Since the Kinetics 700 dataset covers 700 actions, and I are only interested in identifying shouting, I specifically take all the shouting videos and an equal number (651) of randomly sampled videos from the other 699 classes. I then descritize the videos to images with a sampling of 3 frames per second. In the end, this results in nearly even split of 20,094 images labeled as *shouting* and 19,913 images labeled as *other*.

A major difficulty of using the kinetics dataset *false labelling*. That is, in the Kinetics 700 dataset, an entire video has been labeled for an action that may only happen for a short time in the video. I plan to address this in future iterations of this project.

I perform a train and validation split of 2/3 and 1/3 respectively. This results in a training and validation set of 26,805 and 13,202 samples respectively.

In preprocessing, I converted each image to grayscale and reshaped the images to (144, 256).

3. DNN Model

3.1 Architecture

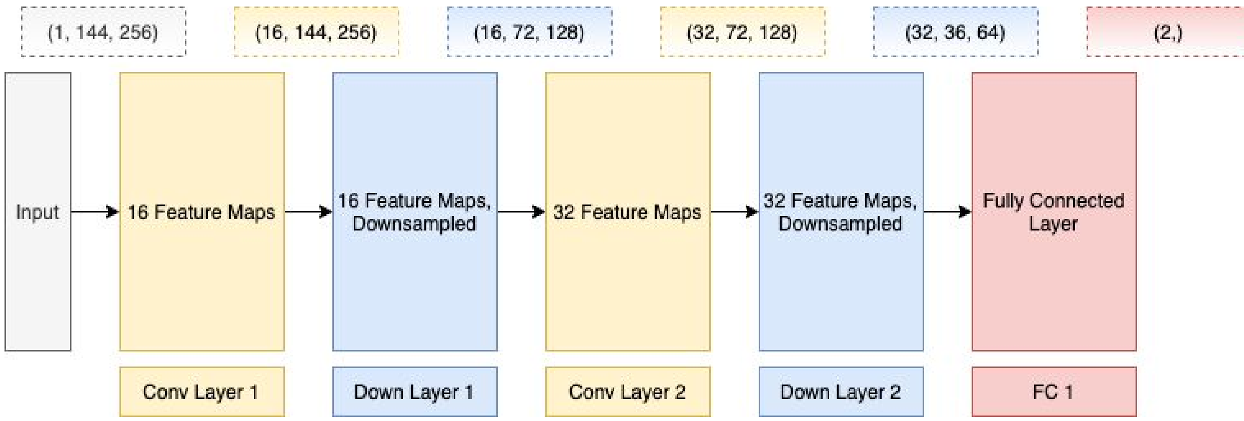

Image 1: Architecture of CNN used (adapted from this)

3.2 Input: Shape of Tensor

X_train shape: 26805 samples, 1 channel, pixel matrix: (144, 256)
X_test shape: 26805 samples, 1 channel, pixel matrix: (144, 256)

3.3 Output: Shape of Tensor

y_train shape: 26805 samples of binary {0, 1} label
y_test shape: 13202 samples of binary {0, 1} label

3.4 Shape of Output Tensor for Each Layer

Output of Conv Layer 1: (16, 144, 256), since same padding
Output of Maxpooling Layer 1: (16, 72, 128), since kernel size of 2
Output of Conv Layer 2: (32, 144, 256), since same padding
Output of Maxpooling Layer 2: (32, 24, 64), since kernel size of 2
Output of Fully Connected 1: (2,), for shout / not shout

4 Hyperparameters

4.1 List of Hyperparameters

In this project, there are three hyperparameters I tuned: batch size; epochs and dropout.

4.2 Range of Value of Hyperparameters Tried

- batch size: {8, 32, 64, 128}
- epochs: {2, 4, 8}

4.3 Optimal Hyperparameters Found

After testing, I chose the following hyperparameters worked best with my architecture:

- batch size = 32
- epochs = 2

5 Training and Testing Performance

```
Epoch: 2 Loss: 0.43146151304244995. Accuracy: 72, Time: 5 min
=> Saving a new best
Epoch: 3 Loss: 0.6871418952941895. Accuracy: 57, Time: 11 min
Epoch: 4 Loss: 0.49511557817459106. Accuracy: 57, Time: 16 min
Epoch: 5 Loss: 0.7898200154304504. Accuracy: 74, Time: 22 min
=> Saving a new best
Epoch: 6, Loss: 0.3489263653755188, Accuracy: 59, Time: 27 min
Epoch: 7, Loss: 0.3444364368915558, Accuracy: 70, Time: 32 min
Epoch: 8, Loss: 0.7275205850601196, Accuracy: 70, Time: 37 min
Epoch: 9, Loss: 0.23404459655284882, Accuracy: 73, Time: 42 min
```

6 Instruction on how to test the trained DNN and how to use the demo

6.1 Install Dependencies

```
cycler==0.10.0
kiwisolver==1.1.0
matplotlib==3.2.1
numpy==1.18.2
opencv-python==4.2.0.32
Pillow==7.0.0
pyparsing==2.4.6
python-dateutil==2.8.1
six==1.14.0
torch==1.4.0
torchvision==0.5.0
```

6.2 Execution

```
>>> cd demo
>>> pip install -r requirements
>>> python demo.py
```

or the following to define run on my own video

```
>>> python demo.py --vidpath ./videos/myvid.mp4
```

6.3 Code

See github here with demo and training colab here and inference colab here.

6.4 Video Link

See how to run code here.

7. Future Improvements

There are several advancements I'd like to address in the future iterations.

1. Architecture: I'd like to take advantage of temporal information by feeding in multiple frames at a time. I'd also like to use 3D convolutions.
2. Preprocessing: I'd like to use facial landmarks
3. Audio: Screaming can largely be detected by sound. I'd like to train an audio classifier and experiment with fusing the audio and visual features in the training.