Adrian Marinovich
Springboard Data Science Career Track
Capstone Project #2

Classification of human actions in videos

Final Report Slide Set
February 20, 2018

Mentor: Hobson Lane

The problem

Classify human actions as seen in videos.

Why is this interesting?

Potential applications:

- Development of instructional tools for robots
- Accident prevention in industrial sites, pools, recreational facilities
- Monitoring of secure locations
- Predict intentions and possible outcomes of human actions for robots / self-driving cars safety

Description of data

UCF101 Action Recognition Data Set:

- 13320 videos collected from YouTube
    - Diverse set of realistic examples in varied settings
- 101 action categories, of 5 types:
    - human-object interaction
    - body-motion only
    - human-human interaction
    - playing musical instruments
    - sports
- 25 unique groups of videos per action
    - Each group consisting of 4-7 video clips segmented from larger video
- Frame rate: 25
- Length range: 1.06 to 71.04 seconds (mean: 7.21 seconds)
- 240 x 320 pixel dimensions, x 3 colors

Data wrangling

Data split into train and test sets using pre-made lists prepared by the UFC research group

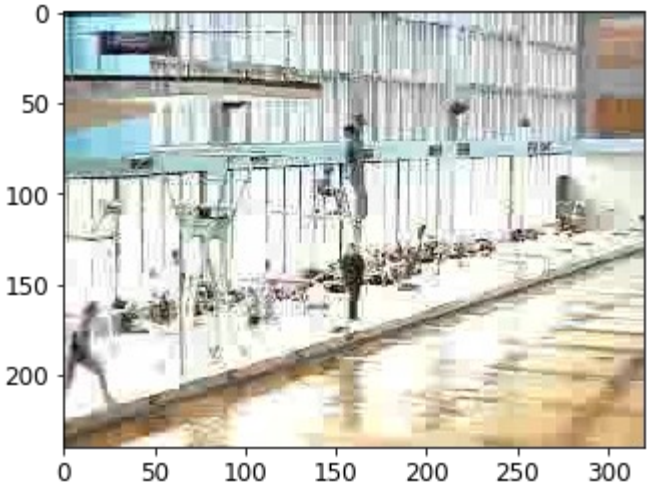
- Designed so groups of video clips obtained from the same original video not split between sets
- Train set: 9537 videos (~72% of total)
- Test set: 3783 videos (~28%)

Extracted frame-by-frame JPEG still image file sequences from each video using FFmpeg

Example images

Biking

Diving



Modeling

Video classification using single-stream methods:

- Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) recurrent neural network (RNN) models
- Fed 2048 features extracted from the image sequences using a pretrained convolutional neural network (CNN) InceptionV3 model
- To reduce computational load, fixed sequence lengths were used during hyperparameter tuning
    - Downsampling and interpolation of frames to get sequence length desired
    - Range of sequence lengths tested for effect on model accuracy
    - Range of model architectures and batch sizes tested
- Top-performing model then fed features extracted using variable sequence lengths

Machine learning was performed using Python, primarily in Keras with a TensorFlow backend, on a stand-alone server equipped with an NVIDIA GTX 1060 GPU with 6 GB memory.

Model tuning

Fixed-sequence-length LSTM and GRU models gave top-1 validation accuracies between 64% and 75%

- GPU memory constraints limited development of LSTM models
- GRU models used for most hyperparameter tuning

Highest fixed-sequence-length validation accuracy: 75%

- Attained using model with 3 GRU layers with 2048 nodes each
- 50-frame sequence length and batch size of 8.
- 50-frame sequence showed improved validation accuracy with smaller batch sizes down to 8
- 80-frame sequence did not show a pronounced association with batch size:
    - 74% accuracy at batch sizes of both 8 and 32
- Adding a fourth GRU layer did not improve validation accuracy
- Number of trainable parameters not clearly associated with validation accuracy

Top-performing model with features extracted from variable sequence lengths from original video image sets, resulted in 80% validation accuracy

- batch size of 1

Model template:

Layer (type)	Output Shape
RNN input layer	(None, sequence length, 2048)
Additional RNN layers	
dense_1 (Dense)	(None, 512)
dropout_1 (Dropout)	(None, 512)
dense_2 (Dense)	(None, 101)

Top performing model:

Layer (type)	Output Shape	Param #
gru_1 (GRU)	(None, seq, 2048)	25171968
gru_2 (GRU)	(None, seq, 2048)	25171968
gru_3 (GRU)	(None, 2048)	25171968
dense_1 (Dense)	(None, 512)	1049088
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 101)	51813

Trainable params: 76,616,805

Table 1. Hyperparameter tuning results.

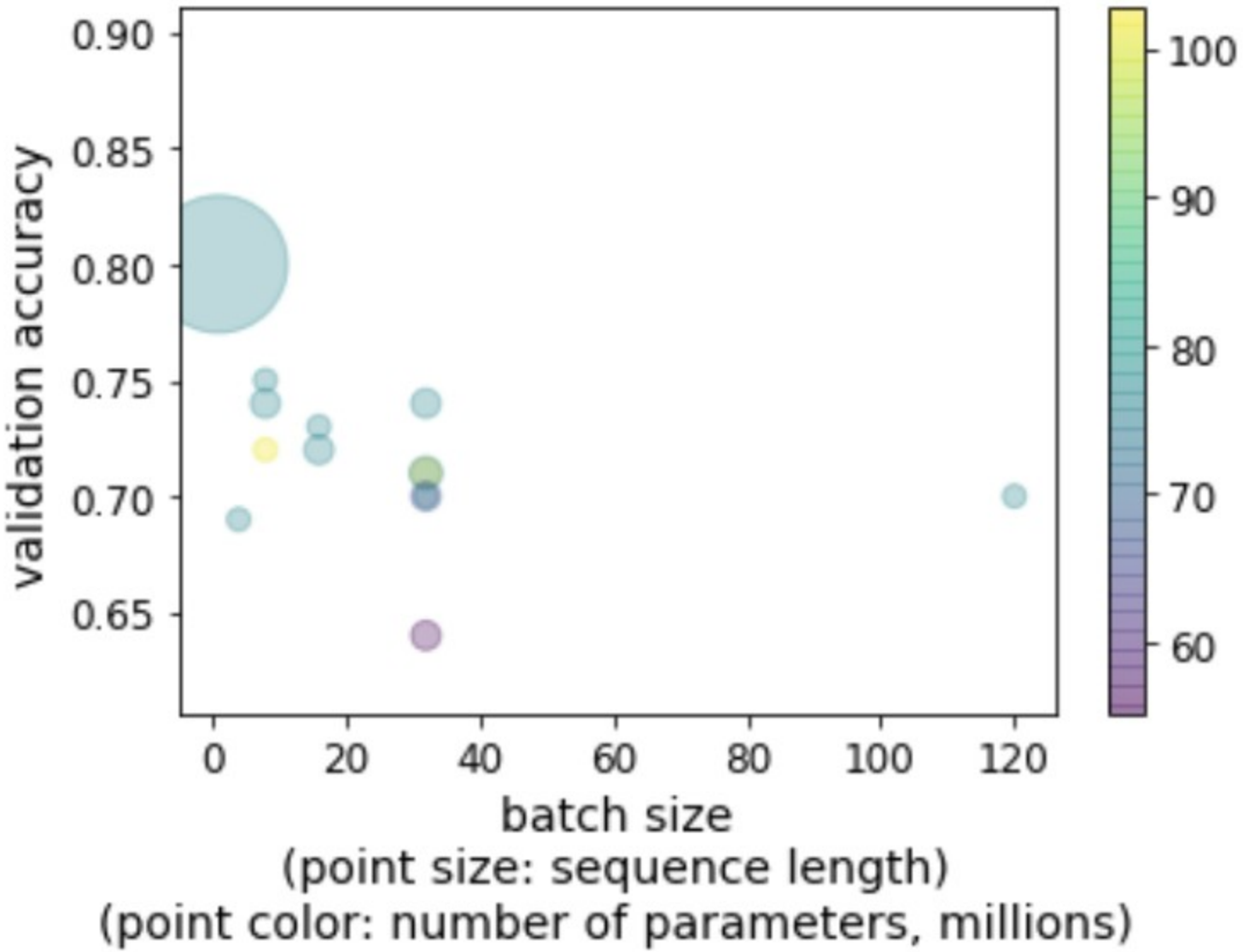
RNN type	Node number by RNN layer[1]				Trainable parameter total, in millions	Sequence length, in frames	Batch size	Epochs[2]	Validation accuracy
	1	2	3	4					
LSTM	2048	1024	1024		55.1	80	32	14	0.64
GRU	2048	4096			102.8	80	32	12	0.71
GRU	2048	2048	2048		76.6	50	4	4	0.69
GRU	2048	2048	2048		76.6	50	8	13	0.75
GRU	2048	2048	2048		76.6	50	16	10	0.73
GRU	2048	2048	2048		76.6	50	32	10	0.70
GRU	2048	2048	2048		76.6	50	120	26	0.70
GRU	2048	2048	2048		76.6	80	8	13	0.74
GRU	2048	2048	2048		76.6	80	16	16	0.72
GRU	2048	2048	2048		76.6	80	32	26	0.74
GRU	2048	2048	2048		76.6	100	32	17	0.71
GRU	2048	2048	2048		76.6	Variable (29-1776)	1	3	0.80
GRU	2048	2048	1024	1024	66.7	80	32	20	0.70
GRU	2048	2048	2048	2048	101.8	50	8	10	0.72

[1]Layer 1 is input layer

[2]Number of epochs until validation loss stopped improving

Figure 1. Plot of validation accuracy against selected tuning hyperparameters.

(Variable sequence length is represented by maximum length of 1776)



Future directions

Limitations of the above overall RNN approach:

- time-consuming to run variable-sequence-length models
- single stream training
- lack of readily available visualization tools for LSTM/GRU models

Further modeling approach to be considered:

- implement two-stream temporal-spatial neural network models
- end-to-end training framework that containing convolutional layers that can be visualized

Project code

https://github.com/adriatic13/springboard/tree/master/dsct_capstone2

References

https://crcv.ucf.edu/data/UCF101.php

https://github.com/fchollet/deep-learning-with-python-notebooks

https://github.com/harvitronix/five-video-classification-methods