Machine Learning II: Final Project
Temporal Shift Module Evaluation
Sarvesh Bhagat

Introduction

This problem falls under the domain of action-recognition. In this field the goal is to classify actions being performed by a human in a video. This has applications in video surveillance, and video storage and retrieval. Most accurate methods of accomplishing this task are large, and take a significant amount of processing power. The Temporal Shift Module, suggested in the paper, ""TSM: Temporal Shift Module for Efficient Video Understanding" [ICCV 2019]" suggests a novel method for creating a CNN capable of efficiently combining both spatial and temporal information from videos. This method of performing classification was implemented, and assessed using the something-something v2 dataset. For this project pre-trained models available on the MIT Han LAbs github were implemented and assessed at varying levels of fidelity. A number of hyperparameters were tuned, and used to train new models in order to attempt to improve the performance of the models.

Dataset

The '20BN-SOMETHING-SOMETHING-DATASET' contains 220,847 videos, divided into a training set of 168,193 videos, a validation set of 24,777 videos, and an unlabeled test set of 27,157 videos. There are a total of 174 class labels, generally in the form of verb - noun - preposition - noun, e.g. "Covering something with something," though they can be more complex. Crucially, this challenging dataset has adequate samples to train a deep network. All videos are at a resolution of 240x240 pixels. Most videos are only a few seconds long. This dataset is significantly larger than most similar datasets. This was allowed by the use of crowdsourcing to generate the labels.

Train	Validation	Test	Total
168,193	24,777	27,157	220,847

Methods

The temporal shift module (TSM) allows for an increase in the ability to model temporal information, at no computational cost greater than that of a 2D CNN. The only required change to a model with 2D kernels is the ability to shuffle the feature map created from the input video clip. This happens inside a residual branch, rather than before a convolutional layer. Shifting the feature map before a residual block causes the spatial learning ability to be harmed. Shifting all

channels causes what the authors of the paper describe as the Naive Shift. It actually results in degradation of both performance and accuracy. While no additional computations are performed, the effect of shifting the data in memory causes a slowdown. The model was tested with no shift, which is the same as a standard 2D CNN, shifting all the channels, which is the naive shift, and then shifting ⅛, ¼, and ½ of the channels. For the partial shift, the first fraction of the feature map is shifted. The partial shift results in lower latency, which is crucial for close-to-real-time performance. All models were pre-trained on ImageNet for the vision component.

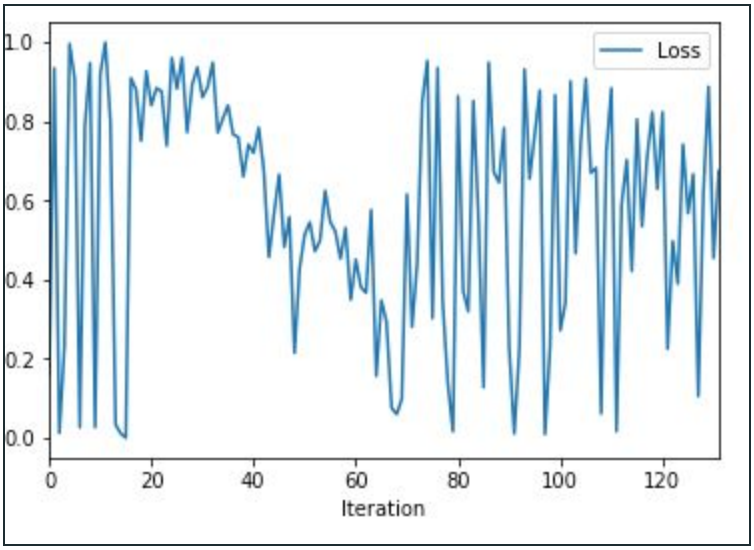
Both bi-directional and uni-directional shifts are possible, depending on the application scenario. **Experimental Setup**. Describe how you are going to use the data to train and test thenetwork. Explain how you will implement the network in the chosen framework and howyou will judge the performance. Will you use minibatches? How will you determine thesize of the minibatches? How will you determine training parameters (e.g., learning rate)?How will you detect/prevent overfitting and extrapolation?

Both newly trained and pre-trained networks were utilized for this project. All testing and training was conducted in pytorch. The MIT Han Labs github provided the code necessary to implement the TSM.

Training (Experimental Setup)

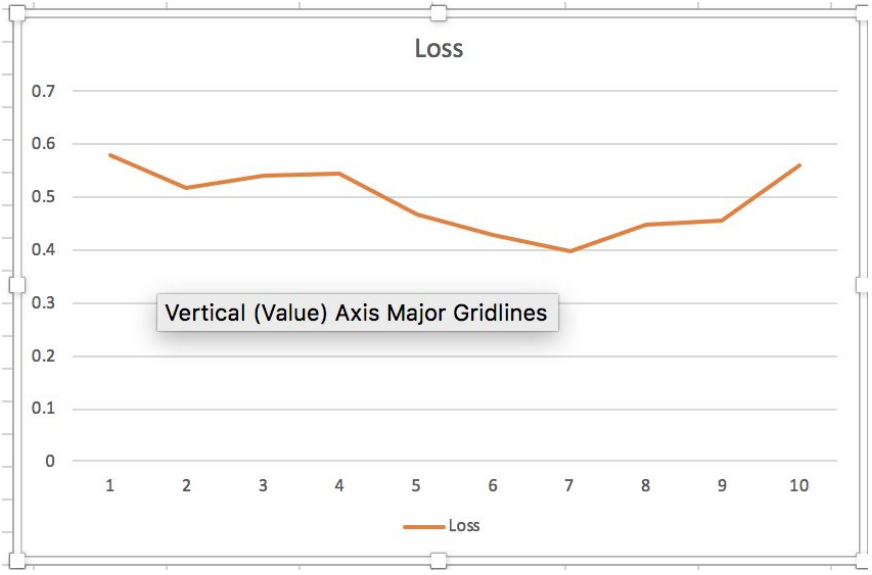
- **Steps for one epoch training**
    - We used Amazon web Services (AWS) cloud for training model using Resnet-50 backbone. Initial training was done for only one epoch.
    - It took about 2 hours.
    - Initial batch size was 128. Due to GPU memory constraints that training failed.
    - To overcome this problem we changed the batch size to 64 and training successfully completed.
    - One epoch took almost 2 hours.
    - Below figure shows loss and number of iterations for just one epoch.

■



Number of epochs = 1
Figure 1

- **Steps for 10 epoch training**
  - Based on lessons learned from the previous training. We trained the network for 10 epch with the same batch size.
  - This took almost 12 hours to train the result.
  - Loss per epoch is plotted in the figure 2.



Number of epochs = 10
Figure 2

- **Training command/script**
  - Below is link for the shell script to run a training job.
  - https://github.com/sarveshbhagat/20BN-something-something/blob/master/Code/train.sh

- **Training Architecture**

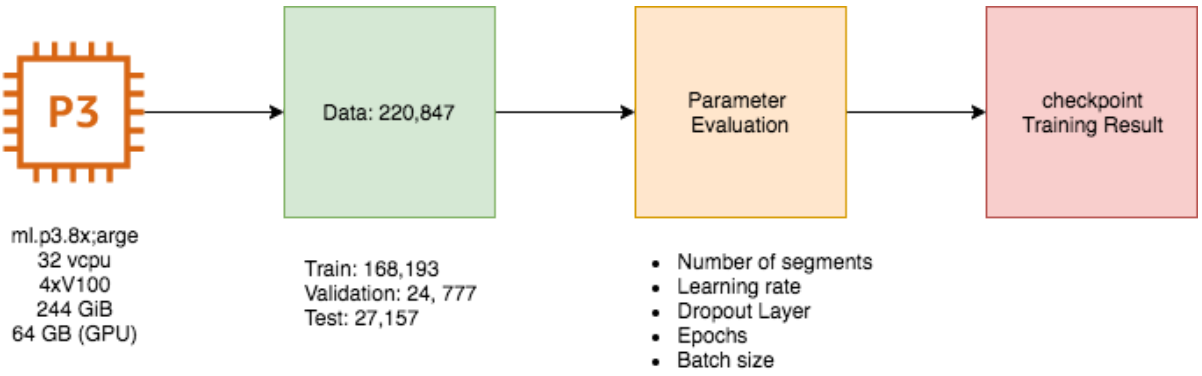


Figure 3

Code contribution:
Create shell scripts to run to code repeadilty. Wrote a python script to download images from source.

Conclusion:

- We propose Temporal Shift Module for hardware-efficient video recognition. It can be inserted into 2D CNN backbone to enable joint spatial-temporal modeling at no additional cost. The module shifts part of the channels along temporal dimension to exchange information with neighboring frames. Our framework is both efficient and accurate, enabling lowlatency video recognition on edge devices.