**Machine Learning II: Final Project**

**Temporal Shift Module Evaluation - Individual Report**

**Peter Shagnea**

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.

My portion of this project included pre-processing of the dataset, and determining how various parameters affected the classification accuracy. In order to read the video files in to a model, they needed to be converted to an image format. Each frame of every video was converted to a jpg file. The effect of compression on the classification was not considered. This resulted in a large increase in the required disk space, and created a need for a new GCE instance. Training was very computationally expensive, and monetarily expensive. I created a machine with multiple GPUs, but the limitation of $100 maximum student credit made it impossible for me to train the models on my machine, as the funds would run out before training was complete. As such, I let Sarvesh handle this aspect of the project.

The primary application of the TSM is to efficiently classify videos. With this goal in mind, I examined the classification time and accuracy of the models at varying degrees of resolution. Ion order to do this, I wrote scripts that iterated through the cartesian product of the test parameters. First, the resolution and cropping of the video was varied. A center crop, full resolution, random crop, and random crop with random flipping augmentation were tested. Methods on the MIT GitHub made it easy to implement these augmentations. Each of these increased processing time by a factor of two over the baseline center crop method. When a single clip is fed into the model, which is not standard practice, reasonable accuracy can be achieved. Using two clips, which is standard practice, performance is moderately improved but at the cost of doubling inference time.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Test Crops** | **Twice Sample** | **Top5 Acc** | **Top1 Acc** | **Average Time/Video (s)** |
| TSM ResNet50 | 1 | 0 | 85.37 | 58.72 | 0.013 |
| TSM ResNet50 | 3 | 0 | 86.27 | 60.20 | 0.037 |
| TSM ResNet50 | 5 | 0 | 86.21 | 59.92 | 0.061 |
| TSM ResNet50 | 10 | 0 | 86.21 | 58.93 | 0.121 |
| TSM ResNet101 | 1 | 0 | 86.60 | 60.98 | 0.02 |
| TSM ResNet101 | 3 | 0 | 87.45 | 61.95 | 0.057 |
| TSM ResNet50 | 1 | 1 | 86.30 | 60.21 | 0.025 |
| TSM ResNet50 | 3 | 1 | 87.08 | 61.21 | 0.073 |
| TSM ResNet50 | 5 | 1 | 87.002 | 61.101 | 0.12 |
| TSM ResNet101 | 1 | 1 | 87.467 | 62.15 | 0.04 |
| TSM ResNet101 | 3 | 1 | 87.94 | 62.76 | 0.11 |
| TSM ResNet101 | 5 | 1 | 87.97 | 62.65 | 0.19 |

I would estimate I wrote ~10% of the code used in this project. The MIT GitHub was heavily leveraged to train and run these models, and tweaked by us in order to meet our needs. The framework used was Pytorch.