May 13th, 2020

Johns Hopkins Final Report (2021)

Sander Schulhoff
University of Maryland

1 Intro

I worked on two main projects for Johns Hopkins University over the 2021 summer and into my Sophomore year of college. First, I learned C# and Microsoft \Psi (Bohus et al., 2021) and used it for two mini-projects. Then, I built a video classification model with Pytorch. In between these two projects, I engaged with the Telluride Neuromorphic Workshop.

2 Microsoft \Psi (MPSI)

I had never used C# before, so I started learning Unity and Visual Studio Code. I built some mini projects/functions to learn keywords and important functionality like LINQ. I found the MPSI Tutorial page to be very helpful, as well.

2.1 RealSense Mini-Project

Professor Andreou asked me to connect a RealSense camera to MPSI. To do so, I needed to build MPSI from source with some special installs. The instructions provided in the MPSI documentation were mostly thorough, however, I had a number of difficulties with building from source (documented here) and using Psi after building (documented here).

After completing the install, recording data from the RealSense camera was quite simple. I copied code from the Azure Kinect Sample and replaced the Kinect source with the MPSI RealSense source. This gave me a video stream similar to the Azure Kinect.

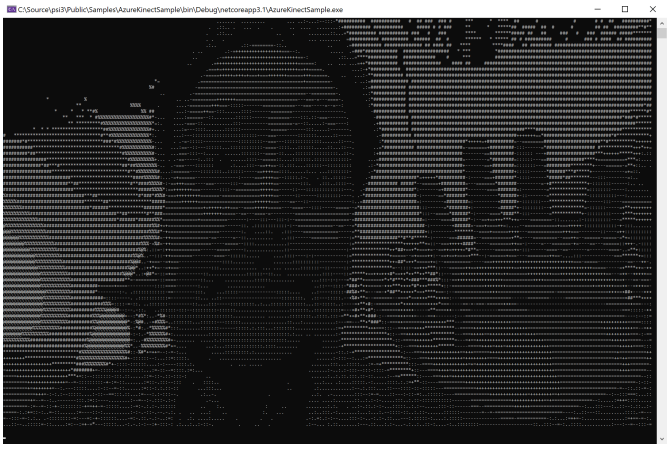


Figure 1: Sample Output from Azure Kinect sample.

2.2 Arduino Sound Data Mini-Project

Next, Professor Andreou asked me to send sound data from Arduino to MPSI. I executed this project with three scripts in different languages. First, I wrote a C++ script to instruct the Arduino to record sound then send it over the serial port. Secondly, I wrote an intermediary

Python script to receive data over the serial port and send it to a tcp address. Finally, I wrote a C# script using MPSI to read the data being sent.

2.2.1 Difficulties

I had a number of issues with the Arduino IDE such as the serial port I was targeting continually changing. Switching to PlatformIO resolved most of my issues, but there were still occasions when the Arduino would not connect to my computer properly. Other than this, my difficulties mainly lay in my general inexperience with the Arduino. Fortunately, I was able to complete the project despite these issues.

3 Neuromorphic Workshop

I watched a number of videos from LTC21 (Learning to Control). I downloaded and setup the l2race (learn to race) environment. I communicated frequently with the project heads while I was going through this process since I encountered a number of bugs which existed as the project was still under development. I played with some of their simple control (e.g. PID) and slightly more complicated RNN models. Most of the work on the project was devoted to using classic control algorithms or using deep learning to predict dynamics of the environment. Since I had recently learned how to use A2C (advantage actor critic), I was interested in pretraining a policy using classic control then fine tuning with reinforcement learning. I began developing a neural model and some training code for this idea, but the developers of l2race kept pushing breaking changes which made my development process difficult. Eventually, Professor Andreou asked me to work on the next project.

4 Video Classification With Event Data

My last project was building an event video classification model for the videos from the The Johns Hopkins University multimodal dataset for human action recognition (JHUMMA) (Murray et al., 2015). The JHUMMA dataset contains rgb video Kinect data of people performing certain actions. All of these actions take place in the same area with similar lighting. Professor Andreou asked me to build a video classification classification model for five of the actions represented in the dataset (walking in place, walking forwards and backwards, walking horizontally across the view of the camera, and walking from the back left corner of the camera view to the front right (and vice versa)).

4.1 Data

4.1.1 Main Dataset

Jonah Sengupta ran 64 videos from the JHUMMA dataset through a model he built which converts rbg video data to event data. After processing the JHUMMA video data, he gave me files which contained the raw events as well as 3-channel video files containing frames of events (dimensions are (400 frames, 3 channels, 420 (h), 560 (w))). Each frame contains events over 33 milliseconds. The events in the frame are organized as positive events in the g channel and negative events in the b channel. The r channel is dropped at training time. I used the 2-channel videos as my main dataset.

4.1.2 Secondary Dataset(s)

In addition to the primary dataset synthesized from Kinect videos, Jonah Sengupta, Michael Tomlinson, and I created videos of ourselves completing each of the five actions. We used a DVS event camera to record our actions. Jonah Sengupta also ran rgb video of our actions through his model. This yielded 15 videos of our actions created by the event camera and 15 videos of those same exact actions created by Jonah Sengupta's model.

4.1.3 Data augmentation

Since the size of the data set is very small, data augmentation was very important for testing performance, even when the test set was from the same distribution as the train set. I applied a number of data augmentation strategies such as:

- Scaling a video down (up to 50% of original size) then randomly padding back to original size.
- Reversing (some) videos
- Horizontally flipping videos and adjusting labels accordingly
- Removing every other frame from videos (even or odd indices at random)
- Perspective changes

All data augmentation is done at training time.

5 Model

Initially, Daniel Mendat asked me to create a CNN+LSTM model. I ended up using a simple single frame spacial temporal CNN model with a GRU for the temporal component. I started with a LSTM, but this project did not have extreme long term dependencies and GRUs cut the training time. Figure 2 contains the exact layer configuration of my model.

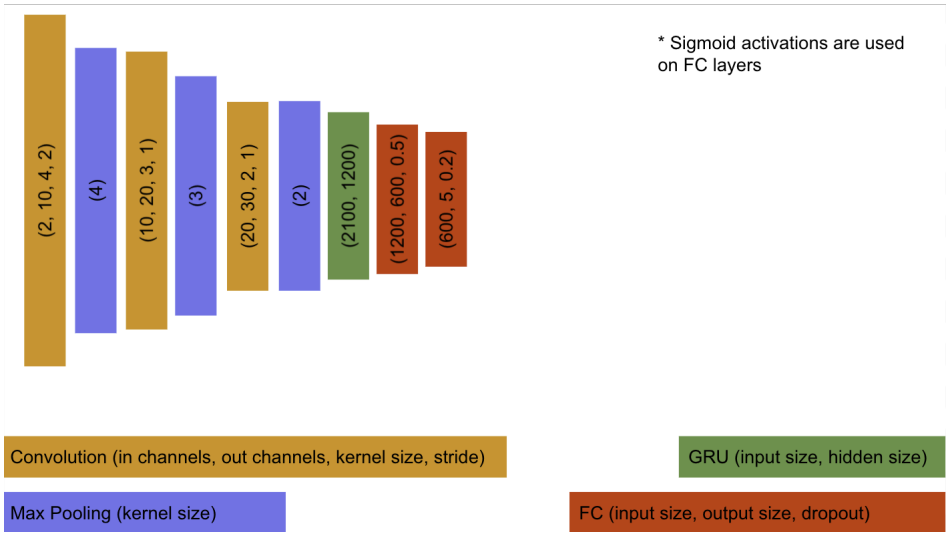


Figure 2: Model Architecture

5.1 Training

I used various training configurations to estimate the performance of the model. I wrote a script to test leave one out cross validation (LOOCV) in addition to regular train/test configurations.

I used a simple training loop, iterating through a Pytorch Data loader object on each epoch. I used a batch size of 1, the Adam optimizer (LR=1e-4), and cross entropy loss. # of epochs varied depending on the train/test configuration.

Please see training results in Section 6

5.1.1 Training Difficulties

- Training would not converge with LR=1e-3. Changing to 1e-4 resolved this.
- The model would always predict the same value for any input. After doing some research, I found that removing biases from the FC layers resolved this issue.
- I spent about 2 weeks debugging a CUDA memory issue where CUDA would consistently run out memory a couple batches into my LOOCV script. I eventually figured out the reason for this was that I was indexing the Pytorch Dataset object twice. However, this double indexing only happened once per model training, so I don't know why I was getting memory errors.

5.1.2 Miscellaneous Training Items

I had significantly large loss spikes that occurred consistently throughout training, when using data augmentation. They did not seem to affect the convergence of the model. I tried regular SGD to see if I could remove the spikes, but this resulted in lack of convergence.

5.2 Future Model Improvements

Although a model as simple as this one should be sufficient for this classification task provided sufficient data, many superior video classification models exist such as early, late, and slow fusion (Karpathy et al., 2014), as well as multi-resolution architectures which make use of a fovea stream (Karpathy et al., 2014).

The architecture I developed does not take advantage of the asynchronicity which event cameras provide as a data source. Research into applying spiking neural networks to video classification such as (Yao et al., 2021) is promising and might be useful for a future iteration of this model. Taking the average of predictions over the last n frames or stacking frames (Huang, 2020) may also provide improved accuracy.

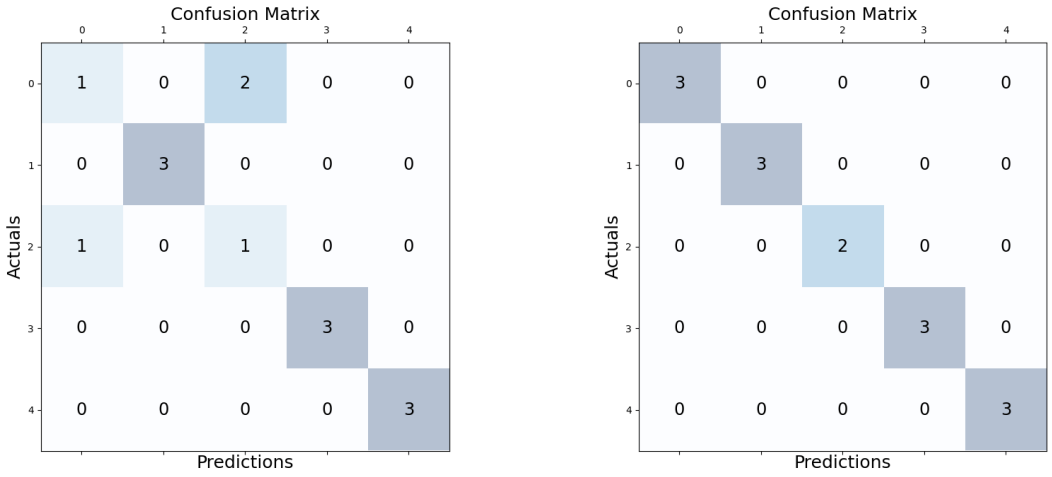
6 Results

I ran multiple training and testing configuration to determine the performance of the model (e.g. train on primary, test on secondary or train on primary and part of secondary and test on the rest). When training on a subset of the primary dataset, I could acheive 100% accuracy on the rest of the primary dataset. Testing performance on the secondary dataset was consistently around 50% accuracy, even when training on subsets of the secondary dataset as well as the main dataset. The performance decrease on the secondary dataset is unsurprising considering the significant covariate shift from training on model-produced videos produced in the same area with the same lighting to testing on both model-produced and event camera-produced videos created in a completely different location/lighting. LOOCV results when using the primary dataset showed approximately 97% accuracy. The high accuracy on the primary dataset and low accuracy on the smaller secondary dataset show the need for more data if a robust classification model is to be built.

7 Miscellaneous

I had to install/set up CUDA on the G7 Dell laptop Professor Andreou gave me to work with. I had difficulty doing so, mostly owing to the fact that the system has two different graphics cards (Intel and NVIDIA).

For ease of reproducibility, I wrote a configuration reading script which reads training configurations from a yaml file and runs them sequentially. Each yaml entry contains its name, what type of data augmentation should be applied, what datasets should be used for training (you can specify some samples from a folder by writing "last" or "first" N), how many EPOCHS to train, and what datasets should be used for testing (with the same "first" and "last" keywords allowed).



(a) No data augmentation

(b) Full data augmentation

Figure 3: Model trained on 50 videos from the main dataset and tested on the rest. [0-4] correspond to [walk_facing_forward_N_S, walk_facing_sideways_W_E, walk_in_place_N, walk_pivot_NE_SW, walk_pivot_NW_SE]

The GitHub repository for the classification model contains various other runs with confusion matrix images corresponding to their accuracies. As mentioned previously, those which test on subsets of the secondary data set tend to have approximately 50% accuracy.

References

Dan Bohus, Sean Andrist, Ashley Feniello, Nick Saw, Mihai Jalobeanu, Patrick Sweeney, Anne Loomis Thompson, and Eric Horvitz. 2021. Platform for situated intelligence. *CoRR*, abs/2103.15975.

Chaoxing Huang. 2020. Event-based action recognition using timestamp image encoding network. *CoRR*, abs/2009.13049.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Thomas S. Murray, Daniel R. Mendat, Philippe O. Pouliquen, and Andreas G. Andreou. 2015. The Johns Hopkins University multimodal dataset for human action recognition. In G. Charmaine Gilbreath, Chadwick Todd Hawley, Kenneth I. Ranney, and Armin Doerry, editors, *Radar Sensor Technology XIX; and Active and Passive Signatures VI*, volume 9461, pages 566 – 581. International Society for Optics and Photonics, SPIE.

Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhao-Xu Yang, and Guoqi Li. 2021. Temporal-wise attention spiking neural networks for event streams classification. *CoRR*, abs/2107.11711.