Individual Report

Aihan Liu

INTRODUCTION

The topic we chose is video classification, also called video recognition. It is probably the most straightforward computer vision task related to the video. Unlike video object detection, it is easier to understand, and we have enough time to understand its knowledge.

We work together most of the time.

We first took nearly one and half weeks to decide which problem we wanted to solve. Due to the time limitation and the understanding of our previous work, we decided to choose the video classification as our project topic.

Then for the data loading, we had the obstacle of using the build-in function in PyTorch. Deyu was training to figure out where the error is from, while I was training to discover how people previously loaded the video data into the model. Finally, he found that the problem was unsolvable, and I found a GitHub repository that we could rely on. He modified the data loader into a more straightforward format that we only need to change the path.

Based on the baseline model (C3D) of Conv3D, I generate a customized network called VC3D (which stands for Video Classification 3D). Then, We work on the evaluation file and demo file separately.

INDIVIDUAL WORK

My work for this project is based on model generation and raises questions. I also tidy the code for the training and evaluation files to the Exam2 format. Change the output in the demo from figures to a video (Thanks for my working experience).

Also, I kept critical thinking and raised many questions during this project, and we tried to solve these questions together. For example, the question I asked about the meaning of clip length leads us to get back to the data loader and figure out how the video has been extracted into the image sequences. This question had been asked after the presentation by the professor coincidently.

PORTION OF WORK

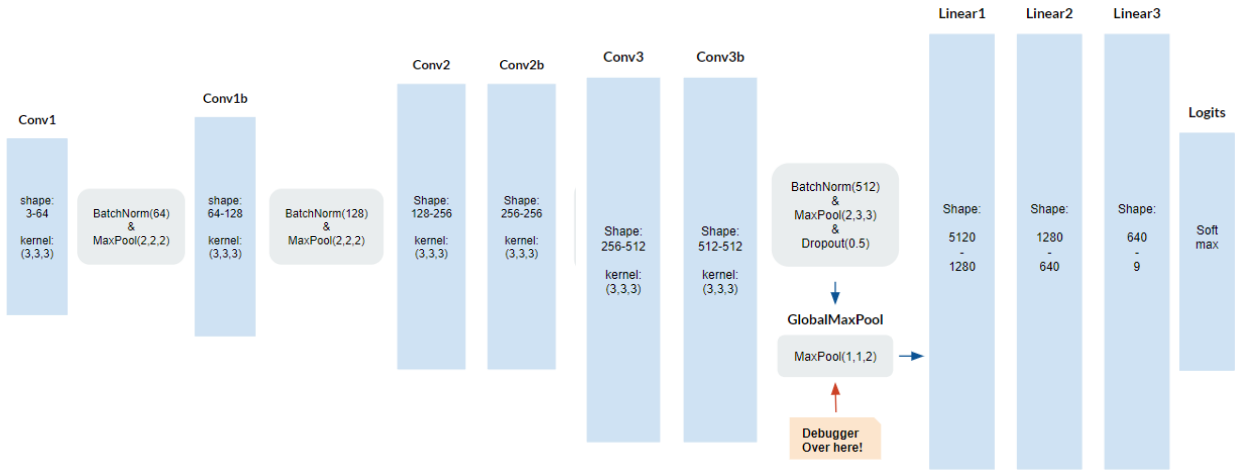

Fig.1 VC3D architecture

Fig1 is the network I generated. I had stacked on the dimension of the last layer for a while, but it is not the most challenging part, I feel. With the increasing of layers and number of neurons, the size of images and frames become smaller and smaller. Suppose we use too large kernel size or the stride size, the size of if frames will become one before the network reaches enough layers. I switched these two parameters several times and got this architecture finally. After the last day of the presentation, I realized that if we increased the number of frames, this architecture could actually go deeper.

RESULT and SUMMARY

```
Test acc 1.00000 Test hlm -0.00000 Test sum 1.00000
```

Fig.2 Testing Result

In this project, we proposed a model called VC3D to deal with the video classification task. It achieved 100% accuracy in the Playing Instruments group for the UCF101 dataset.

However, there are some limitations to the model for the demo we made. It will misclassify some musical instruments when we apply other videos that are not included in the UCF101 dataset as a test set. To see the robustness of this model, I asked some of my friend to film a short clips of instrument playing, including piano, guitar, and flute. These clips will undergo the same preprocessing as were training data, then conduct inference by our trained model. Classification labels with their confidence were overlaid on the frames of the video clip output.

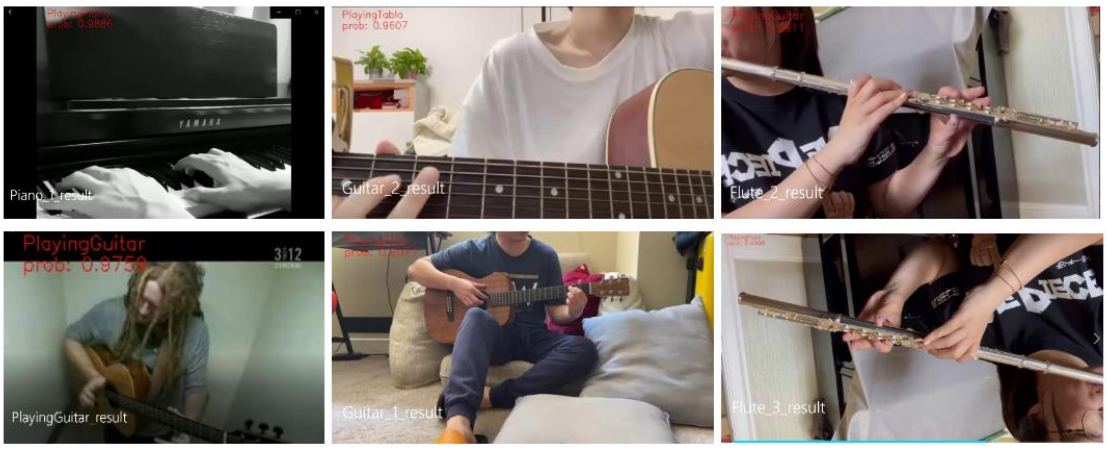

Fig.3 Inference on External Videos

The result shows that the model is not robust enough. Except the bottom left video which from the original testing data of the UCF101 dataset, the grayscale piano playing, and the upside down flute playing, other videos were misclassified.

The future work we could do to improve the result consists of data augmentation, applying more networks such as two-stream networks. Some other techniques

could be our potential improvements, such as attention mechanism and loss functions.

In this project, apart from learning the video classification methods, we also learned the number of neural changes in the neural network, the importance of kernel size, and stride size, and how to customize a network myself.

There are some details that we might ignore while training, such as the metrics differences and position of the softmax layer. The CrossEntropy loss has embedded the softmax layer already, therefore in the model architecture, it is not necessary to include it. Even Professor had mentioned it when we take Exam2, I didn't have a clear understanding of this at that time. While, when I built a model myself, I had a struggle time with this problem. Once I add a softmax layer for my output layer with CrossEntropy loss, the model is hard to converge. After consulting with Professor, I realized the difference between CrossEntropy loss and the negative log-likelihood loss (nn.NLLLoss). I am pretty sure I will not forget this point ever. This gives us a deeper understanding of what we know about neural networks.

CODE

We have several files related to this project:

Dataset.py/mypath.py: found online, based on the repository https://github.com/jfzhang95/pytorch-video-recognition/blob/master/dataloaders/dataset.py

Model_Definition.py: created model based on C3D, 50% copied online.

Train.py: based on the Exam2 code provided by Prof. Amir Jafari, rewrote the train() function, approximately 65% code remained the same.

Eval_CNN3D.py: based on the Exam2 code provided by Prof. Amir Jafari, rewrote the train() function, approximately 55% code remained the same.

Demo.py: change the output to video; approximately 90% code from my group mate's work

REFERENCES

[1] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 4489-4497).

[2] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, *27*.