BBM 418 – Programming Assignment 4

Erhan Kabaoğlu - 21627389
Hacettepe University
Computer Science
erhankabaoğlu@hacettepe.edu.tr

1. Introduction

Video classification is challenging problem in computer vision. In this assignment, I will implement two stream CNNs to recognize actions in videos suggesting by Simmoyan and Zisserman. This architecture consist of two parts which are Spatial and Temporal. Spatial part is similar to image classification. We choose a frame from video. But videos different from images because of time space. So, Simmoyan and Zisserman suggest Temporal CNNs to capture motions. Finally, We merge spatial and temporal CNNs to classify videos. In temporal, We calculate optical flow from given sequance frames and stack them together. In spatial, We just capture a single rgb frame.

2.Implementation Details

2.1. HDF5 file

I preprocess videos and extract spatial and temporal frames from videos and resize into 256x256. After that, I store this structure into HDF5 file format to speed up training time.

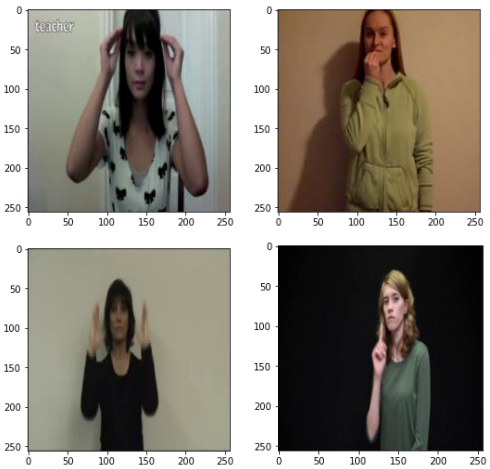
2.2. Optical Flow

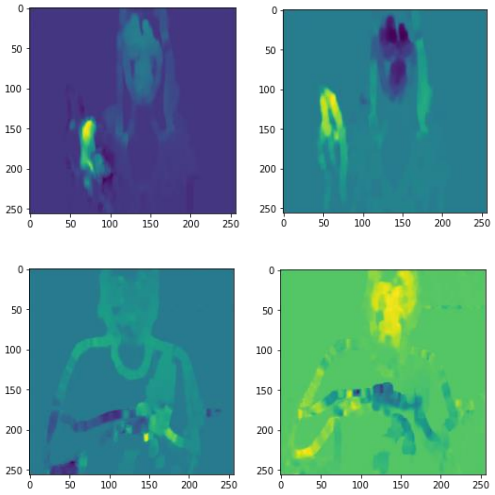
I used OpenCV FlowFarneback algorithm to calculate dense optical flow. In more details, I give start and end frame number and video name in the json file to the function.
I calculate motion size by substracting start frame number from end frame number and find step size with dividing by optical flow size (in our case 10). First part, I find middle frame for spatial input. Second part, I iterate all frames starting with start frame and calculate optical flow with this sequence of frames. I advance by the step size because I want to capture all motions between start and end frames.To calculate Dense Optical Flow, I find x and y component of optical flow suggested by Simmoyan and Zisserman and stack them together.Finally I return spatial and temporal frames.

2.3. Dataset

I inherit pytorch.utils.data.Dataset. In this function, I get spatial and temporal frames and labels from HDF5 file. After that, I apply some transformations to spatial and temporal frames.In more details, I convert into PIL image, apply random crop and random flip, Also apply color jitter to spatial frames. I normalize all frames and return these after converting tensor in train dataset. In validation and test dataset, I just apply center crop different from train dataset. Finally, I return spatial, temporal frames and labels by given index.

Here are some random examples from spatial and temporal frames:



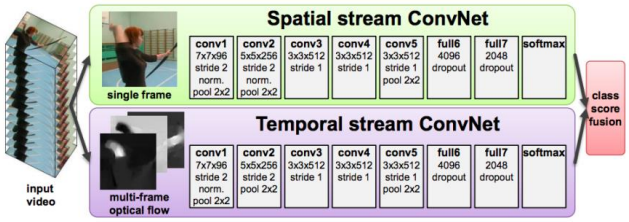


2.4. Dataloader

I just use torch.utils.data.Dataloader for train, validation and test datasets.

2.5. Architecture

As Simmoyan and Zisserman suggest, I use architecture shown below.



I implement this architecture using pytorch nn modul. I implement another modul to apply early fusion. This modul takes two architecture which are spatial and temporal as input. I removed last fc layer of this input moduls and rebuild a joint fc layer. In forward part, I calculate spatial and temporal parts by given input and concatenate output of spatial and temporal moduls.Finally, I apply common fc layer and get output.

2.6. Train and Test

I implement two different functions to train two stream CNNs. First of these train for spatial, temporal and early fusion. It takes one CNN, one criterion and one optimizer. Second of this train for late fusion. It takes two CNN and two optimezer. It calculate two different losses and sum these together. I set epoch size to 100 for each traning function. I use CrossEntropyLoss, Adam optimizer and StepLR learning scheduler. There is an another function to test. It calculate test accuracy and plot confusion matrix.

3.Experimantal Results

For all experiments, I use 100 epoch because its enough to train and I use StepLR scheduler with starting lr 0.0001, gamma 0.1 and step size 40.

Here is validation accuracies and plots for all setups:

Spatial CNN

Batch Size	Dropout ratio	
	0.1	0.4
8	35.6%	33.3%
16	33.3%	31.0%

Temporal CNN

Batch Size	Dropout ratio	
	0.1	0.4
8	16.0%	14.9%
16	17.2%	20.6%

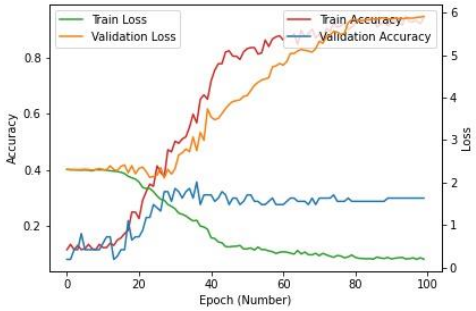
Early Fusion (S=Spatial T=Temporal)

Batch Size	Dropout ratio			
	S	T	S	T
	0.2	0.5	0.4	0.8
8	21.8%		22.9%	
16	21.8%		32.1%	

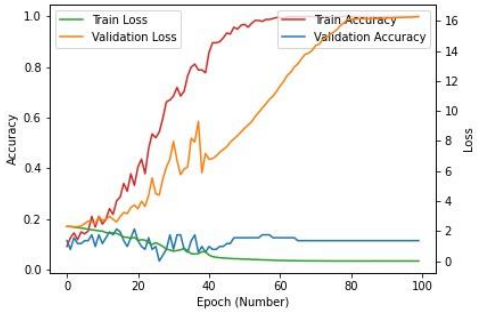
.

Late Fusion (S=Spatial T=Temporal)

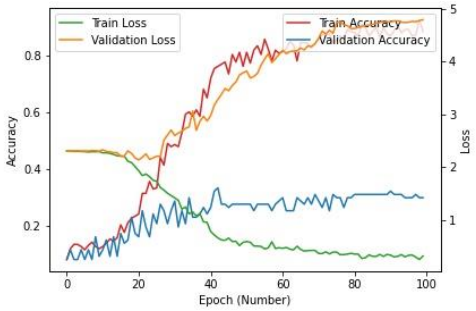
Batch Size	Dropout ratio			
	S	T	S	T
	0.2	0.5	0.4	0.8
8	29.8%		27.5%	
16	27.5%		24.1%	



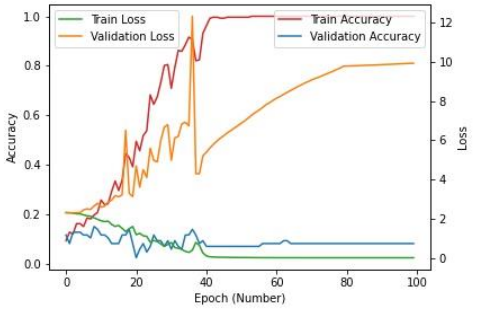
Spatial CNN Bath size=8 Drop ratio=0.1



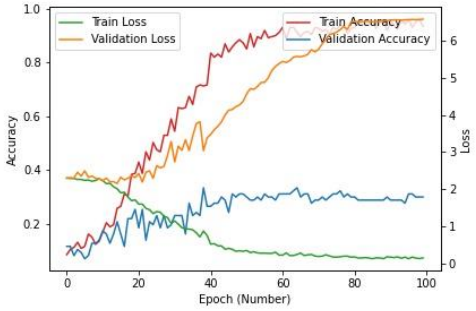
Temporal CNN Bath size=8 Drop ratio=0.1



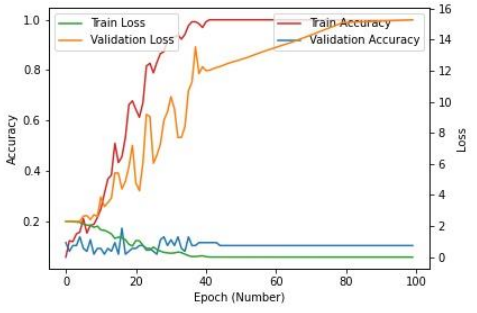
Spatial CNN Bath size=8 Drop ratio=0.4



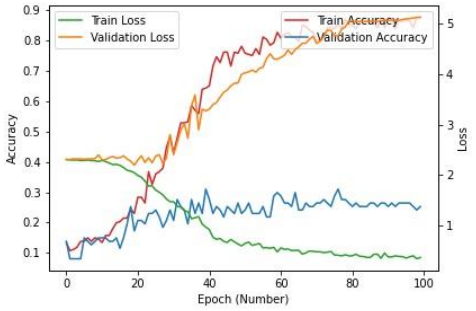
Temporal CNN Bath size=8 Drop ratio=0.4



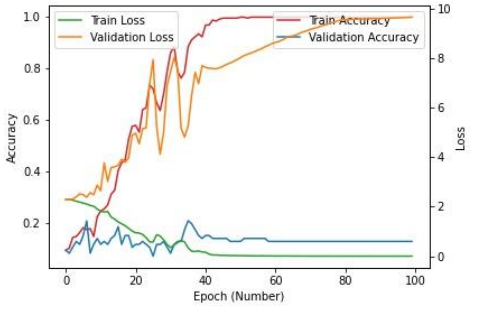
Spatial CNN Bath size=16 Drop ratio=0.1



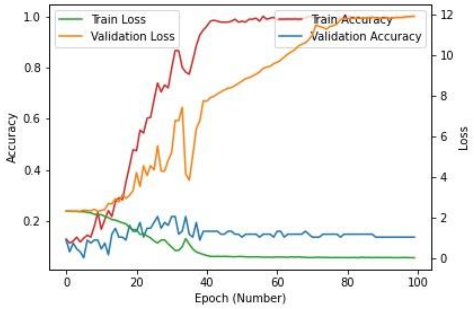
Temporal CNN Bath size=16 Drop ratio=0.1



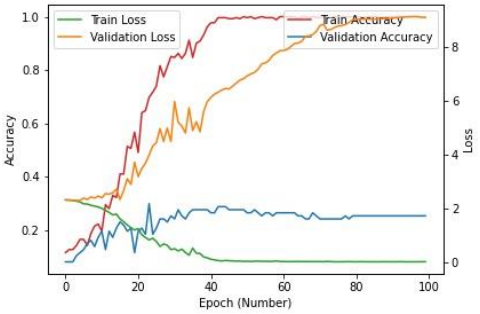
Spatial CNN Bath size=16 Drop ratio=0.4



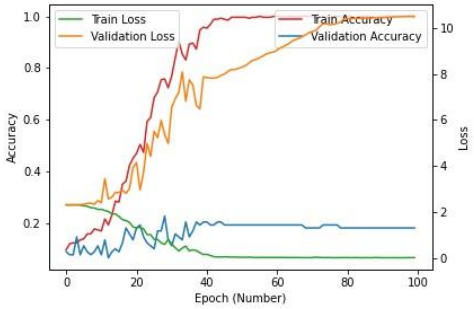
Temporal CNN Batch size=16 Drop ratio=0.4



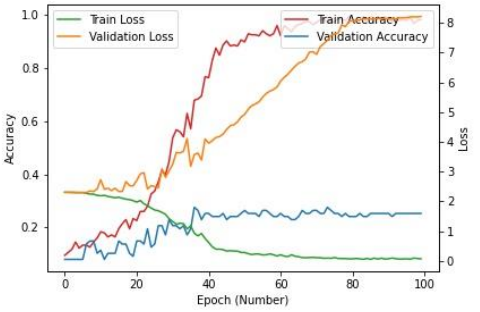
Early Fusion Batch size=8 Drop ratio=0.2, 0.5



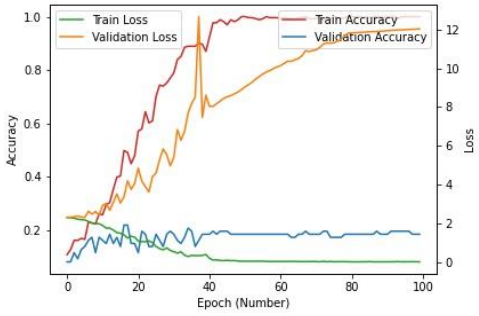
Late Fusion Batch size=8 Drop ratio=0.2, 0.5



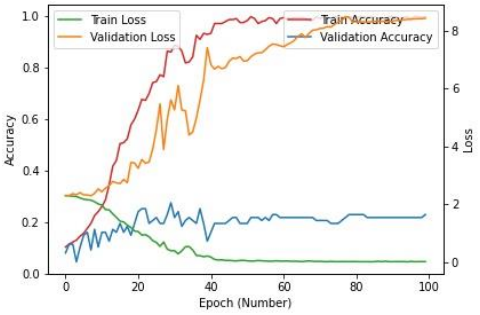
Early Fusion Batch size=8 Drop ratio=0.4, 0.8



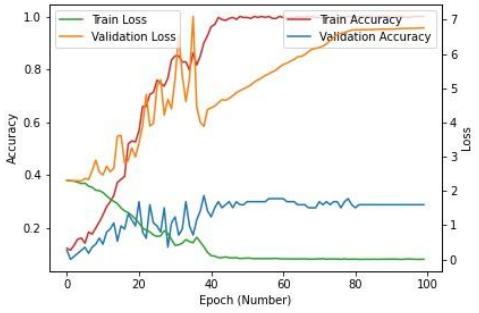
Late Fusion Batch size=8 Drop ratio=0.4, 0.8



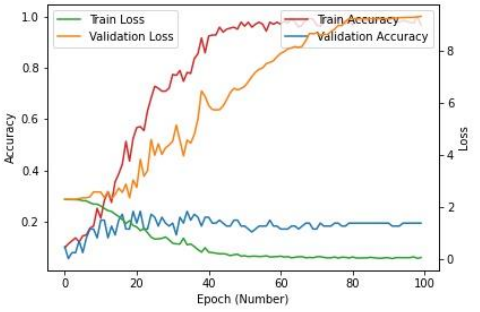
Early Fusion Batch size=16 Drop ratio=0.2, 0.5



Late Fusion Batch size=16 Drop ratio=0.2, 0.5

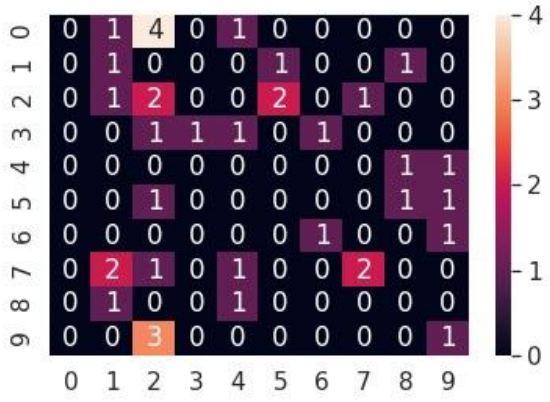


Early Fusion Batch size=16 Drop ratio=0.4, 0.8



Late Fusion Batch size=16 Drop ratio=0.4, 0.8

I choose Early Fusion batch size=16 drop ratio=0.4 0.8. I have better accuracy in Spatial CNNs experiments but I think Early Fusion is more convenient for this assignment.
Here is confusion matrix:



Test Accuracy = 21.0%

I reach approximately 30% validation accuracy in Spatial CNNs. We have small dataset in this assignment so our model overfitting. I reach less validation accuracy than Spatial CNNs in the Temporal CNNs. I think just learning Temporal CNNs doesn't make sense. İt just capture motions. As Spatial CNNs, Temporal CNNs overfit as well because of small dataset. In Early Fusion and Late Fusion, models overfitting faster because both Spatial and Temporal CNNs train together. Experimental accuracies less than Spatial CNNs and more than Temporal CNNs, The reason is that, We have small dataset and there is a lot of local optimum for this experiment. So Temporal CNNs need more data to get better results.
I select bath size's 8 and 16 because this is appropriate for size of our dataset.
I apply 0.1 and 0.4 drop out for Spatial and Temporal CNNs. I want to reach different local optimum.
I apply 0.2, 0.5 and 0.4, 0.8 drop out for Early and Late Fusion. These models are more complex than others. So I apply drop out more aggressively. The reason same as Spatial and Temporal CNNs.
I set lr to 0.0001. When I set bigger lr, My loss values doesn't decrease. I reduce lr at each 40 epoch.

4.Conclusion

Overfitting because of small dataset is the most obvious problem in the assignment. That's why I can't reach remarkable validation and test accuracies.

I notice that merging two models for different subproblems helps get better results in a specific problem.

I think pretrained models can help to impove results.

Also drop out factor is very important part of this experiment because dataset distribution is scattered. That is why there are a lot of local optimum.

To improve experiment, we can crop necessary part of frames.