Video Classification using 'tilted' batch

Name: Koo In Yong Student ID: 20160042

1. Introduction

In time-domain feature extraction, we used two approaches of convolution method; independent-dimension approach and additional-dimension approach. We can either extract temporal features and spatial feature independently, or we can add two dimensions and conduct spatio-temporal convolution multiple times. A spatio-temporal convolution method has advantage that we can consider spatial and temporal features simultaneously, and thus we may find some new interrelated features.

For spatio-temporal convolution, we implement a 3D (height × weight × frame) convolution filter (or batch). It is common to make a cuboid patch. That is, we usually set a 2D filter and extend it towards the frame axis. In a common convolution batch, we investigate same position for certain frames. My idea starts from here. What if I 'tilt' this batch?

The modification is simple. Select spatial 2D filter, and shift a certain pixels every frame to make a 'tilted' convolution batch.

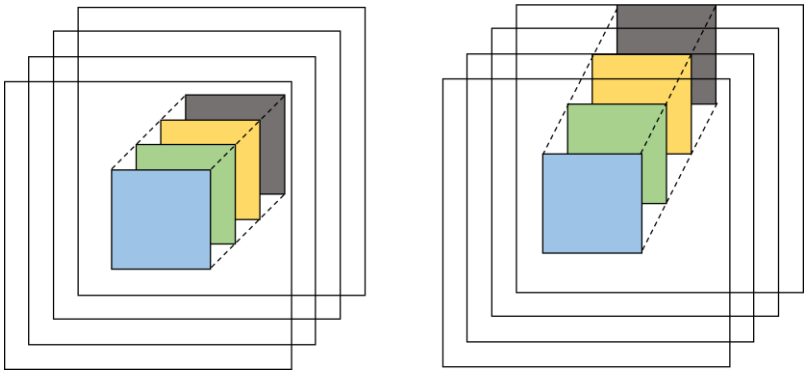

Figure 1 A common batch and a 'tilted' batch (towards y-axis)

I inferred that selecting a 'tilted' convolution batch can enhance CNN's classifying performance. I will elaborate my hypothesis and validate my theory on this report.

2. Theory/Methods

Selecting batch size for a network is a very important stage in network construction. The number of convolution layers and pooling layers can be determined by the network. It is an essential component of network configuration, and network's performance depends highly on this criterium.

However, we've never considered the shape of a convolution batch. My assumption is that a tilted batch can be a better option than a common cuboid batch in certain circumstances.

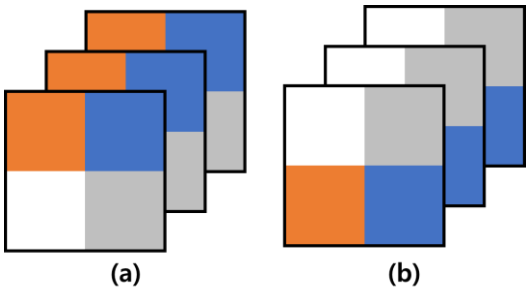

Figure 2 Example of video sequence

Let's say we'd like to classify two video sequence (a) and (b), which is a continuation of a same frame shown as Figure 2.

If we can classify (a) and (b) by a selected batch, (i.e., we cannot select a common batch from two video) then we will say it is distinguishable by the batch. Now we can compare distinguishability of the common cuboid batch and the tilted batch as in Figure 3.

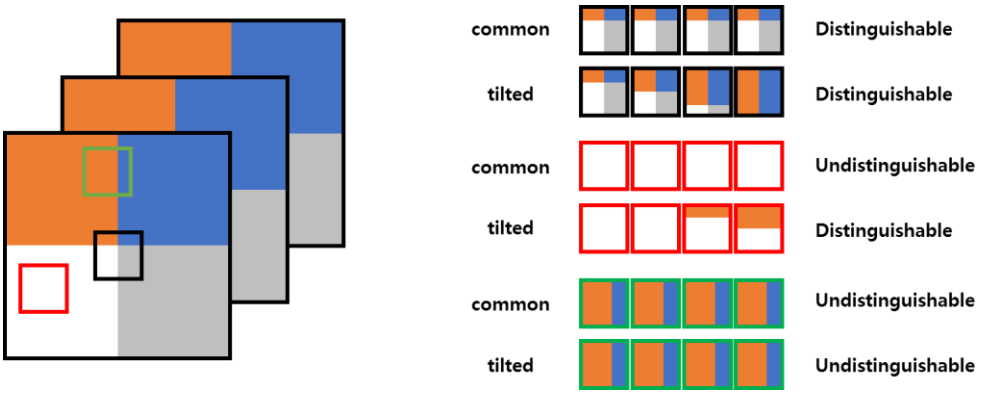

Figure 3 Distinguishability of the common batch and tilted batch (toward y-axis)

As we select red, black, and green frame in video (a), we can see that usage of common batch can only distinguish video (a) and (b) if the batch is set as black frame. However, using tilted (toward y-axis; selected filter window shift towards +y axis every frame) batch can distinguish video (a) and (b) even when it is selected on red frame. Since we randomly crop batches and convolute them to our network, this implies that we are more likely to classify (a) and (b) better if we use tilted batch.

This is because we were able to acquire more information from a tilted batch. We can understand more features of a video by shifting window every frame. One may protest that this is an extraordinary circumstance, since pixel value does not have any temporal difference. One may say this is only applicable in occasions when pixel values do not change much through frame axis. However, note that this is not at all 'unusual' situation. In real life, a video consists a certain **locality** due to continuity. A pixel value of a frame does not differ significant amount from the previous frame. Using a fixed angle camera often make background pixels to consist its value though frames. Therefore, we can reasonably assume that using a tilted batch is often better than a common cuboid batch for classification.

Now I'd like to validate my inference by comparing the performance of classifier using common batch and tilted batch. I used the CNN model I used in homework 3, and this is the architecture of my network.

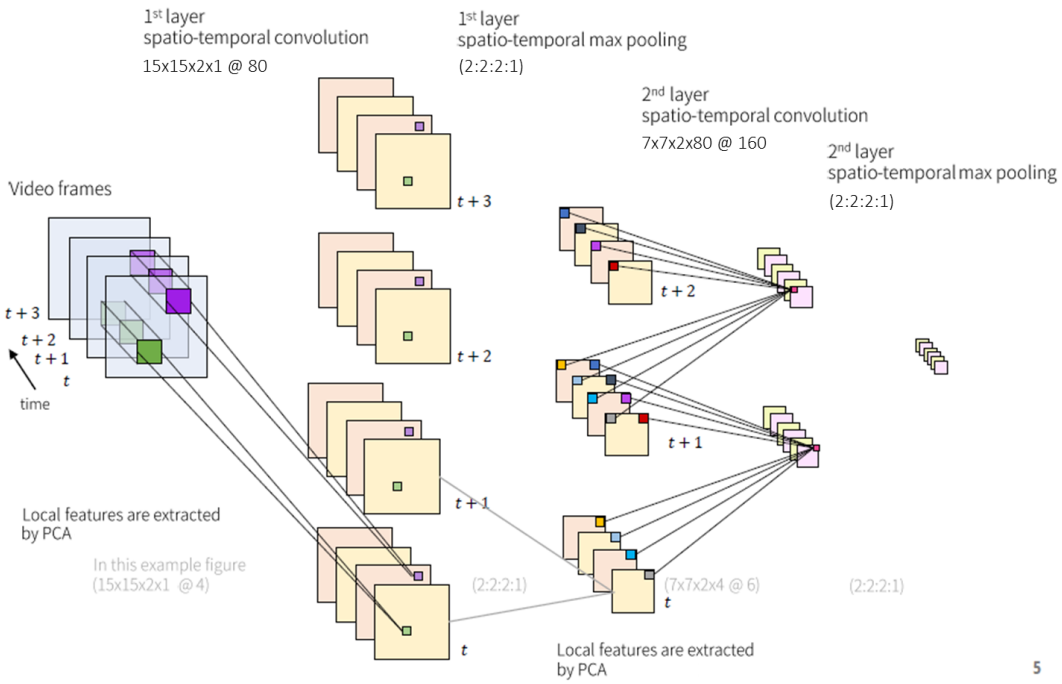

Figure 4 CNN architecture used in this project

Note that I used PCA feature extraction method, and that I modified temporal convolution/pooling window size to 2.

Since performance may be different due to video characteristic, I used 3 types of tilted batch; horizontal, vertical, and diagonal.

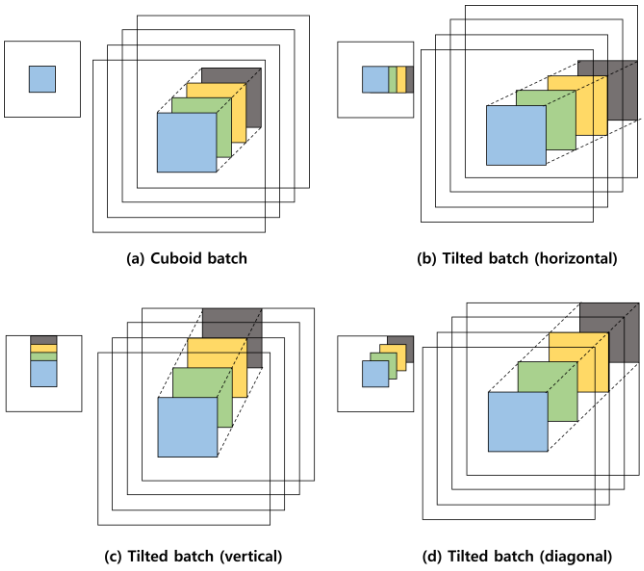

Figure 5 different shape of batches

Note that unlike in figure 5, we used (15×15×2) batch for the first convolution layer, and (7×7×2) batch for the second convolution layer.

MATLAB implementation

Number of pixels to shift can be designated by network configuration as in following manner.

```
config{1,1}.shift_vertical = 2;
config{1,1}.shift_horizontal = 0;
. . .
config{2,1}.shift_vertical = 2;
config{2,1}.shift_horizontal = 0;
```

I first trained 4 PCA models with tilted data.

```
tilted_data = zeros(config{layer,1}.filter_height, config{layer,1}.filter_width, config{layer,1}.filter_frame,
config{layer,1}.in_feat_maps);

for k = 1: config{layer,1}.in_feat_maps
        for t = 0 : config{layer,1}.filter_frame -1
                tilted_data(:,:,t+1,k) = data(height + t*config{layer,1}.shift_vertical: height + t*config{layer,1}.shift_vertical +
config{layer,1}.filter_height -1, ...
                                                        width + t*config{layer,1}.shift_horizontal: width +
t*config{layer,1}.shift_horizontal + config{layer,1}.filter_width -1 , ...
                                                        frame + t, k);
        end
end
              data_video(:,idx) = reshape(tilted_data, ...
                    config{layer,1}.filter_height * config{layer,1}.filter_width * config{layer,1}.filter_frame *
config{layer,1}.in_feat_maps, 1);
```

Also, I tilted data before calculating activation_c (which is resized) as following.

```
activation_c = zeros(config{layer,1}.in_height - config{layer,1}.filter_height - (config{layer,1}.shift_vertical *
(config{layer,1}.filter_frame - 1)) + 1, ...
                        config{layer,1}.in_width - config{layer,1}.filter_width - (config{layer,1}.shift_horizontal *
(config{layer,1}.filter_frame - 1)) + 1, ...
                        size(data,3) - config{layer,1}.filter_frame + 1 , ...
                        config{layer,1}.out_feat_maps); % convolution activation

…

for k=1:config{layer,1}.out_feat_maps
        for t=1 : size(data,3) - config{layer,1}.filter_frame + 1
                for tilt = 0 : config{layer,1}.filter_frame - 1
                        for y = 1 : config{layer,1}.in_height - config{layer,1}.filter_height - (config{layer,1}.shift_vertical *
(config{layer,1}.filter_frame -1 )) + 1
```

```
                    for x = 1 : config{layer,1}.in_width - config{layer,1}.filter_width -
(config{layer,1}.shift_horizontal * (config{layer,1}.filter_frame -1 )) + 1
                        tilted_data(:,:,:,:) = data(y + tilt*config{layer,1}.shift_vertical : y +
tilt*config{layer,1}.shift_vertical + config{layer,1}.filter_height -1, ...
                            x + tilt*config{layer,1}.shift_horizontal : x +
tilt*config{layer,1}.shift_horizontal + config{layer,1}.filter_width -1 , ...
                            t : t + config{layer,1}.filter_frame - 1, :);
                    end
                end
            end
        end
end
```

3. Data/Results

4 PCA were trained with 15 video data and 3000 patches.

This is the PCA filter for the first convolution layer, with original (cuboid) batch.

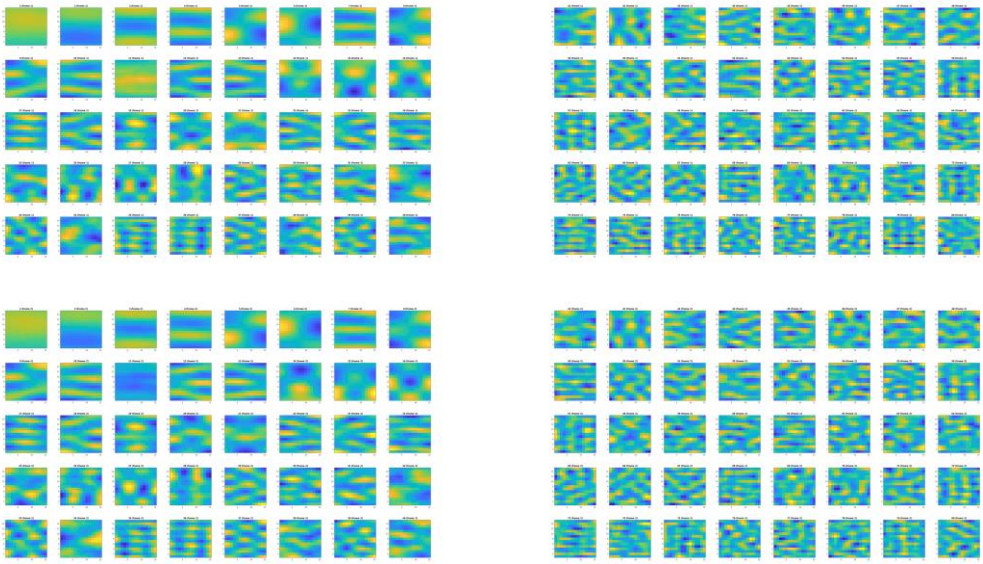

Figure 6 PCA filter (cuboid batch)

This is the PCA filter for the first convolution layer, with tilted batch (horizontal; 2 pixels per frame)

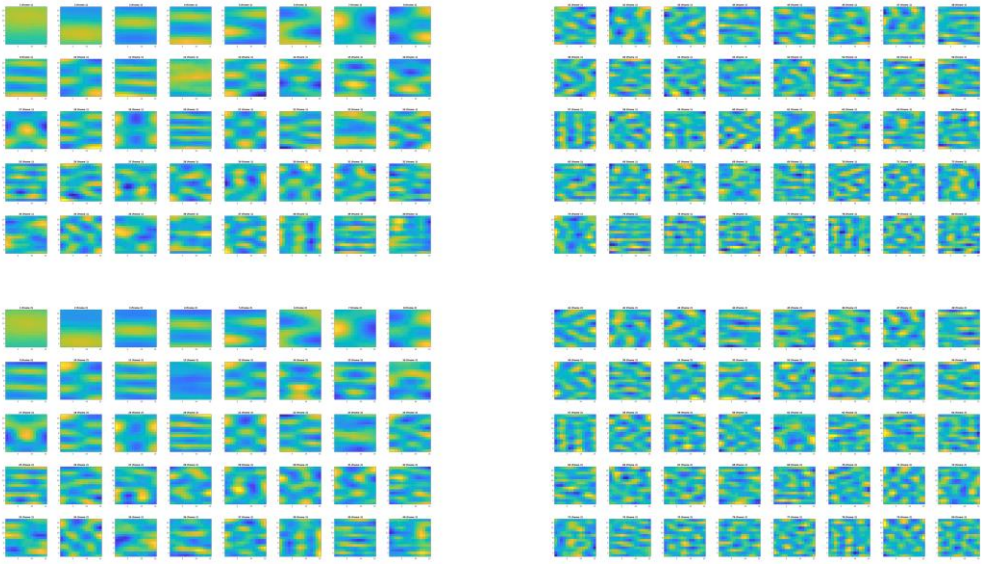

Figure 7 PCA filter (tilted batch - horizontal)

This is the PCA filter for the first convolution layer, with tilted batch (vertical; 2 pixels per frame)

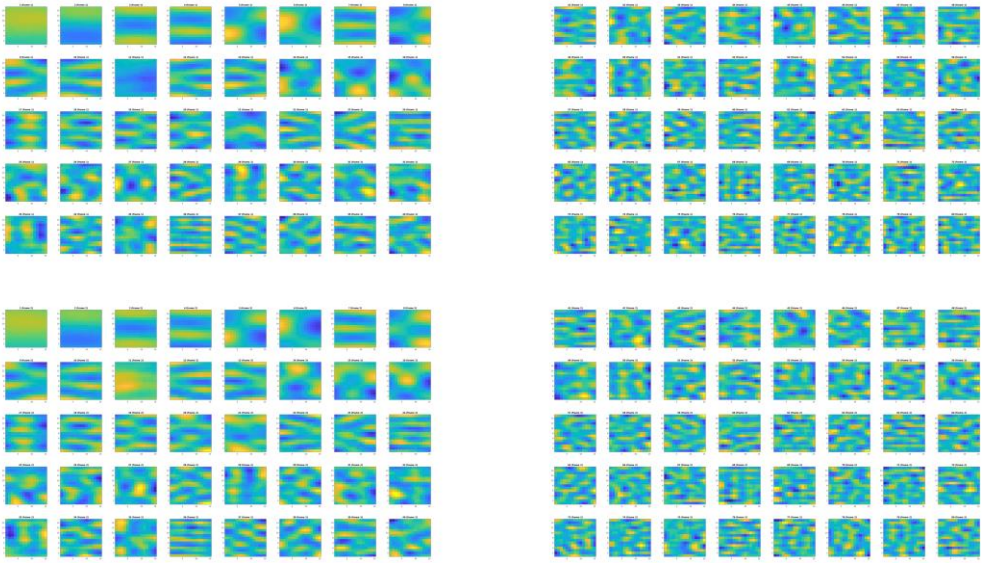

Figure 8 PCA filter (tilted batch – vertical)

This is the PCA filter for the first convolution layer, with tilted batch (diagonal; 2 pixels per frame)

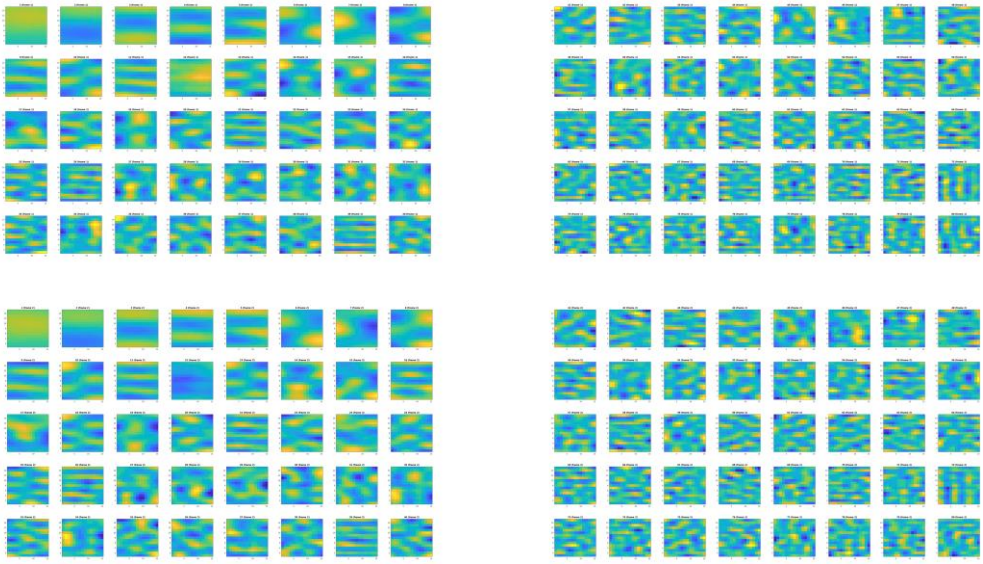

Figure 9 PCA filter (tilted batch - diagonal)

I applied these filters to classify videos as in Homework 4. I used ‘max’ pooling method, and used 50 samples for training, 10 samples for validation and 10 samples for testing.

Four classifying results were identical.

Cross entropy error converged to 36.72 (0.734 for validation) as shown in Figure 10 (on next page).

Validation accuracy shifted between 0% and 50% as shown in Figure 11 (on next page).

Test error was 0.73, with accuracy 50%.

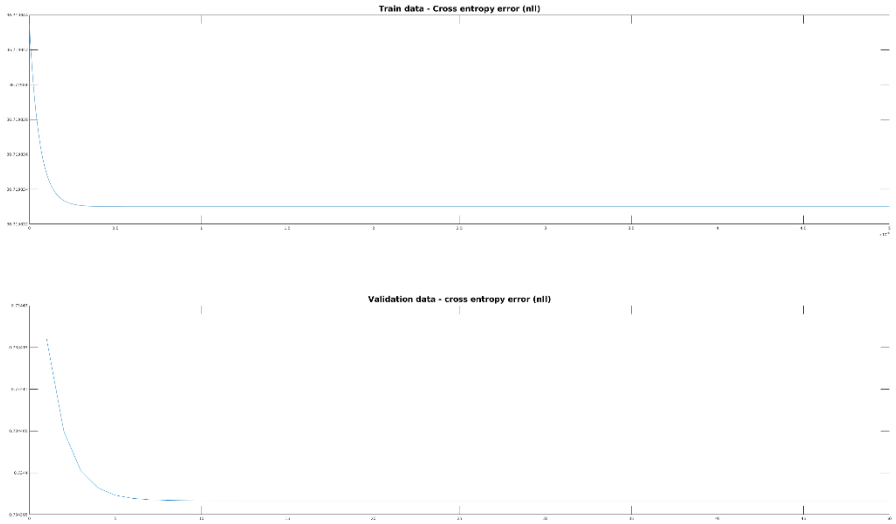

Figure 10 Cross entropy error convergence graph

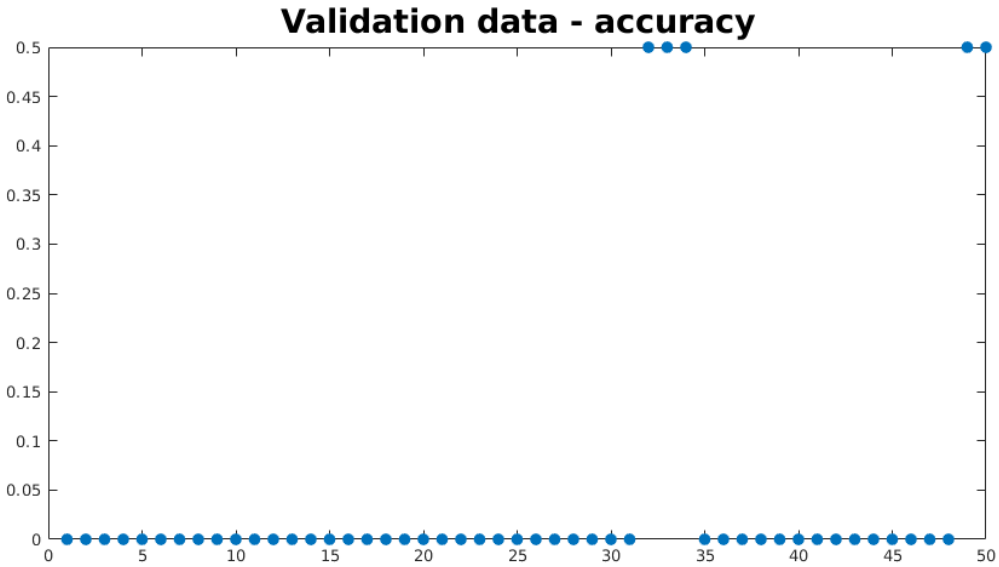

Figure 11 Validation data accuracy plot

4. Discussion/Conclusion

I was able to produce PCA filter using tilted batch, and convoluted them in appropriate way to classify a video. I expected classification with tilted batches to show better performance than classification with common batches, but the result shows that it did not.

My hypothesis did not turn out to be true from this validation example. Maybe, my assumption was wrong after all – we should use cuboid shape batch. But I don't think my hypothesis makes no sense, because classification using tilted batches showed that it is as good as the common cuboid shape batch. If we remind ourselves the result of homework 4, PCA and SAE feature extraction methods; 'max' and 'mean' pooling methods did not show any significant difference also. But it doesn't mean that those methods have no difference at all. Thus, I won't conclude my attempt of using tilted batches was non-sense failure, but rather make a discussion of possible causes of this result and setting objectives of future works.

The comparison between classification using tilted batches and cuboid batches may have shown same results for following reasons.

First, lack of data. I used 50 training cases, 10 validation cases and 10 test cases for my project. Though it is increased number compared to what I've used on homework 4, (I used 34 / 5 / 6 cases for homework 4. Test accuracy was 16.4%

back then.) It was still insufficient amount to train my data. I could not enlarge my data set due to limited memory (of the use of Haedong Lounge machine) and time. Considering the given data set involves 850 training cases and 150 test cases, I could try again on better environment with enlarged data.

Second, batch's temporal length was too small. I limited all temporal length of batches to 2 frames, since the data was too short. (some data were 7 frames long.) If the batch length is too short, additional-dimension approach loses its advantage over independent-dimension approach. A tilted batch would show no significant difference with a cuboid batch. Regarding that, I may have had to shift pixels more dramatically. I did not consider that my batch is only 2-frame-long, and just shifted 2 pixels each. I will make a prolonged tilt my batch more significantly to make significant difference from a cuboid batch.

Also, network could be too complex to be influenced by batch shape. My network consists 80 nodes for the first layer and 160 nodes for the second layer. If my theory is correct, a tilted batch shows advantage over cuboid batch in smaller network, because large number of nodes would overcome such 'locality' problem and allows cuboid batch to have sufficient information to classify data.

Maybe, the given data set did not match my preset circumstances. I expected a tilted batch is better than a cuboid batch when the pixel values does not change dramatically from the previous pixel. But the given video sequences were not acquired from the 'real-life' but produced by random function. A tilted batch shows no difference with a cuboid batch for such occasion.

Lastly, I would like to try using sparse-autoencoder (SAE) method. I did not afford to use SAE method, because I would then need to find appropriate learning rate and other configurations to let my SAE converge. Results using SAE to extract features can be different from the results using PCA.

In conclusion, to be honest, I doubted to get a significant result that matches my expectation. What I learned from homework 3 and 4 is that it is difficult to find an appropriate value to make a valid network. However, I am proud of my attempts, to elaborate the concept of 'tilted batch' and actually approves it work as fine as the common cuboid shape batch. I'd like to supplement limits and faults I listed above, and make future works to expand understanding on 'tilted' batch.

5. Reference

[1] All class materials and homework guidelines; reading materials of 2018 Spring EE476 Audio-Visual Perception Model

[2] Wikipedia, Principal component analysis, https://en.wikipedia.org/wiki/Principal_component_analysis

[3] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014.