DLCV Homework 4

張緣彩 R07922141

18/12/2019

Collaborators

1. 林子淵 R07921100
2. 黃孟霖 R07922170
3. 楊之郡 R08922050

Problem 1. Trimmed action recognition w/o RNN.

Q1. Describe your strategies of extracting CNN-based video features, training the model and other implementation details (which pretrained model) and plot your learning curve (The loss curve of training set is needed, others are optional).

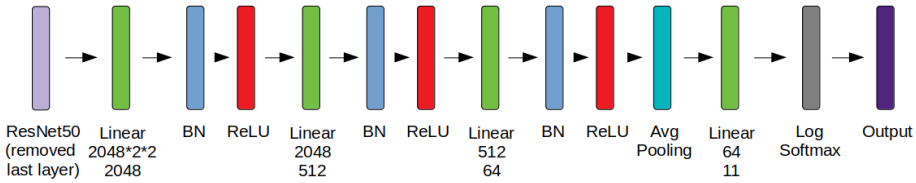

Figure 1: CNN-based video features classification model w/o RNN.

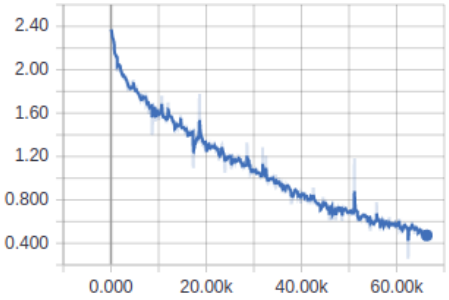

Figure 2: Learning curve (avg. training loss per iteration) of CNN-based video features classification model w/o RNN.

For data preprocessing, I normalized every frames with mean and standard deviation of ImageNet. After that, I used normalized all-zero for padding. I

also sort the batch with frames length for using *pack padded sequence* and *pad packed sequence* in model. The maximum length for a batch is 30, if the video's frame is longer than 30 frames, I truncated it into 30 frames with first 30 frames.

For the model part, backbone (pretrained) I used for features extraction is ResNet50. I removed ResNet50 the last classification layer. After few layers of feed-forward, I do average pooling (mean) for every time-steps and finally pass to last classification layer.

For the training, I used Adam with learning rate of 0.0001 and weight decay of 0.00005. Batch size of 4 with gradient accumulation of 8, that is accumulate 8 times *loss.backward()* before update the weights (*optimizer.step()*). With this trick, I am able to train the model on small memory's GPU with large batch size.

Q2. Report your video recognition performance (valid) using CNN-based video features and make your code reproduce this result.

	Validation set
Accuracy	0.427828

Table 1: Accuracy of CNN-based action recognition model w/o RNN.

Q3. Visualize CNN-based video features to 2D space (with tSNE) in your report.

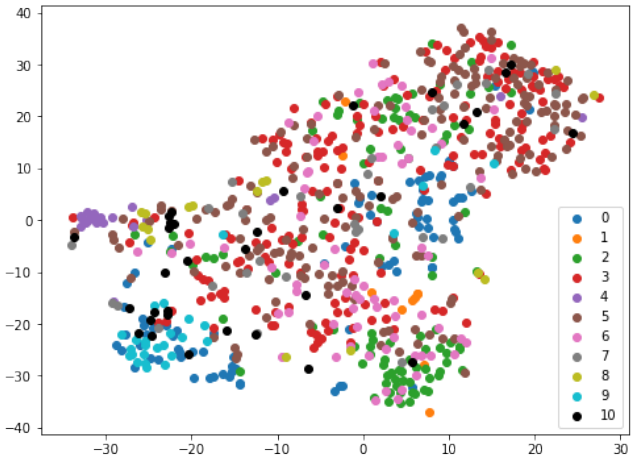

Figure 3: CNN-based video features 2D visualization w/o RNN.

Problem 2. Trimmed action recognition w/ RNN.

Q1. Describe your RNN models and implementation details for action recognition and plot the learning curve of your model (The loss curve of training set is needed, others are optional).

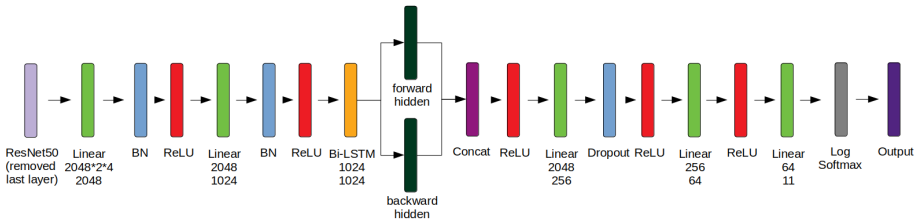

Figure 4: CNN-based video features classification model with RNN. Bidirectional LSTM is used, and forward hidden and backward hidden were concatenated.

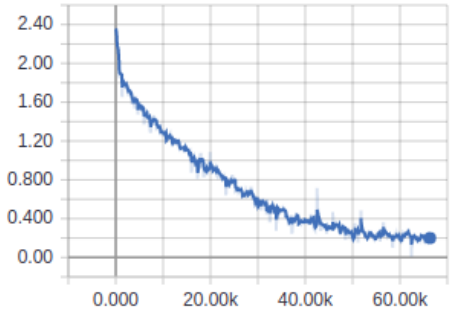

Figure 5: Learning curve (avg. training loss per iteration) of CNN-based video features classification model with RNN.

For data preprocessing, I normalized every frames with mean and standard deviation of ImageNet. After that, I used normalized all-zero for padding. I also sort the batch with frames length for using *pack padded sequence* and *pad packed sequence* in model. The maximum length for a batch is 30, if the video's frame is longer than 30 frames, I truncated it into 30 frames with first 30 frames.

For the model part, backbone (pretrained) I used for features extraction is ResNet50. I removed ResNet50 the last classification layer. After few layers of feed-forward and its dimension is lower from 2048*2*4 to 1024, I passed it to a bidirectional LSTM of 1024 dimension. Then I concatenated the forward hidden and the backward hidden and passed it to the classifier as shown in figure 4.

For the training, I used Adam with learning rate of 0.0002 and weight decay of 0.00005. Batch size of 4 with gradient accumulation of 8, that is accumulate 8 times *loss.backward()* before update the weights (*optimizer.step()*). With this trick, I am able to train the model on small memory's GPU with large batch size.

Q2. Report your video recognition performance (valid) using CNN-based video features with RNN and make your code reproduce this result.

	Validation set
Accuracy	0.496749

Table 2: Accuracy of CNN-based action recognition model with RNN.

Q3. Visualize RNN-based video features to 2D space (with tSNE) in your report. Do you see any improvement for action recognition compared to CNN-based video features ? Why? Please explain your observation

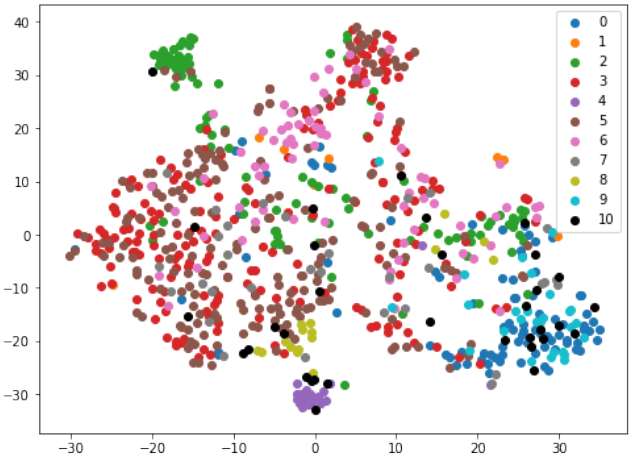

Figure 6: CNN-based video features 2D visualization with RNN.

Yes, by using LSTM (RNN) has slighly improved the classification accuracy. By comparing the tSNE visualization of CNN-based video features (figure 3) and Recurrent CNN-based video features (figure 6), we can see that class 2 (open) and class 4 (cut) have been clustered closer. I think because these classed

are continuous action, it is harder to classify these with only features without time information. And by using LSTM, the time information is added into the features.

Problem 3. Temporal action segmentation.

Q1. Describe any extension of your RNN models, training tricks, and post-processing techniques you used for temporal action segmentation.

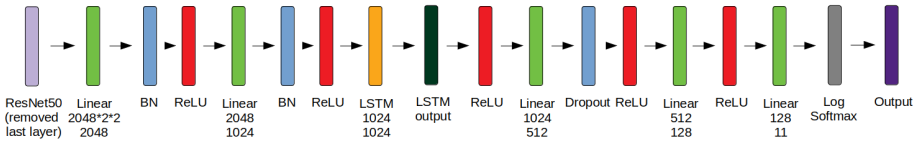

Figure 7: CNN-based video features classification model with RNN (Bidirectional LSTM).

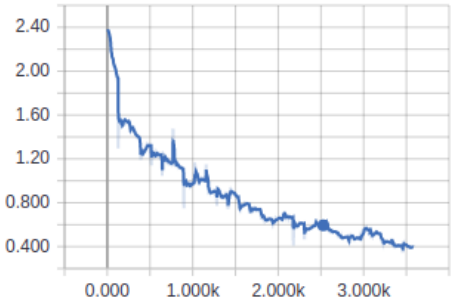

Figure 8: Learning curve (avg. training loss per iteration) of CNN-based video features temporal action segmentation model with RNN.

For data preprocessing, first I trimmed every video into continuous short video of up to 100 frames. Because they are continuous short videos, I also overlap the start frame of the next video with the last n frames of the previous video to include the continuity. My maximum video length is 100 and overlap length is 20. Secondly, I normalized every frames with mean and standard deviation of ImageNet. After that, I used normalized all-zero for padding. I also sort the batch with frames length for using *pack padded sequence* and *pad packed sequence* in model.

For the model part, backbone (pretrained) I used for features extraction is ResNet50. I removed ResNet50 the last classification layer. After few layers of

feed-forward and its dimension is lower from 2048*2*4 to 1024, I passed it to a LSTM of 1024 dimension. Because this is a action segmentation task, so I have to classify each output (frame). I passed the LSTM's output to the classifier and calculate Cross Entropy Loss of each frame. I use *pack padded sequence* and *pad packed sequence* to ignore padding frame when updating the weights.

For the training, I used Adam with learning rate of 0.0002 and weight decay of 0.00005. Batch size of 2 with gradient accumulation of 16, that is accumulate 16 times *loss.backward()* before update the weights (*optimizer.step()*). With this trick, I am able to train the model on small memory's GPU with large batch size.

Q2. Report validation accuracy in your report and make your code reproduce this result.

	Validation set
OP01-R02-TurkeySandwich	0.514822
OP01-R04-ContinentalBreakfast	0.601833
OP01-R07-Pizza	**0.685148**
OP03-R04-ContinentalBreakfast	0.527559
OP04-R04-ContinentalBreakfast	0.648848
OP05-R04-ContinentalBreakfast	0.597046
OP06-R03-BaconAndEggs	0.678917
***Average**	**0.607739**
****Average**	**0.626203**

Table 3: Accuracy of temporal action segmentation per category and per action. ***Average = (sum of accuracy of each category ÷ number of category)**. ****Average = (sum of correct predicted actions ÷ sum of total actions)**.

Q3. Choose one video from the 7 validation videos to visualize the best prediction result in comparison with the ground-truth scores in your report. Please make your figure clear and explain your visualization results (You need to plot at least 500 continuous frames).

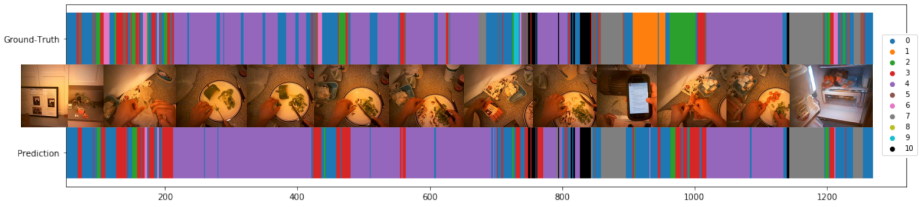

Figure 9: Temporal action segmentation visualization of **OP01-R07-Pizza** within frames 50 to 1320. X-axis is time stamp (t, no. of frame).

I choose **OP01-R07-Pizza** to visualize since my model predicted this category with highest accuracy. Then I picked frames from 50 to 1320 because these frames showed more activities.
From the first frame (t=50), the label is 'other' (blue), we can see in the figure 9 first frame, it do nothing.
For the second frame (t=200), the label is 'take' (red), my model also classified these few frames to 'take', we can see in the figure 9 second frame, the man is taking something out from the plastic bag.
For the third frame to sixth frame (t=300, 400, 500 and 600), most of the labels are 'cut' (purple), my model also classified these few frames to 'cut', we can see in the figure 9 third to sixth frame, the man is cutting green pepper and mushroom.
For the seventh frame (t=700), we can see in the figure 9 seventh frame, the man is moving around to take the sausage. From the ground-truth, 'take' (red) actions happened around t=700, but my model only predicted 'take' and mispredicted 'move around' to 'cut'.
For the eighth frame (t=800), the label if 'cut' (purple), from the eighth frame in the figure 9, the man is cutting again.
For the ninth frame (t=900), 'read' (orange) label appears and from the ninth frame in figure 9, we can clearly see that the man is reading cooking instruction from his phone, but unfortunately my model predicted labels are almost 'other' (blue).
For the tenth frame (t=1000), the labels are 'open' (green) and followed by 'take' (red), from the tenth frame in the figure 9, the man is opening the plastic bag to take the sausage out, and my model predict 'open' (green) and 'take' (take) in around t=1000.
For the eleventh frame (t=1100), the label is 'cut' (purple), from the figure 9, the man is cutting the sausage, my model successfully classified it to 'cut'.

For the last frame (t=1200), the labels are 'move around' (grey) followed by 'open' (green), from figure 9, we can guess that the man is moving around to the refrigerator and open the door as the last frame shown, my model is also successfully classified 'move around' (grey) followed by 'open' (green).

References

[1] `None`