

Introduction to Computer Vision (Spring 2026)

Assignment 4

Release date: May 23, due date: June 5, 2026 11:59 PM

This assignment includes 3 tasks: implementing and training a PointNet for classification and segmentation, a self-attention captioning model, and a vanilla RNN for image caption. They sum up to 100 points and will be counted as 10 points toward your final score for this course.

Submission: To submit your homework, please compress your code and results using our provided script *pack.py*, and submit to course.pku.edu.cn. Feel free to post in the discussion panel for any questions and we encourage everyone to report the potential improvements of this assignment with a bonus of up to 5 points.

1. PointNet (35 points):

PointNet is the most widely adopted neural network for point cloud learning. In this question, you are requested to implement the pipeline of PointNet for both classification and segmentation tasks on ShapeNetPart¹ and then visualize the features. Please read and follow the README to prepare the environment and dataset.

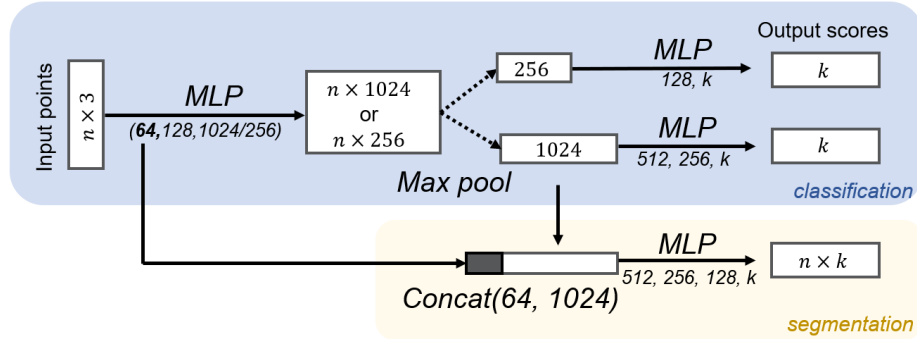


Figure 1: PointNet architecture for both classification and segmentation tasks.

a) Points Classification and Segmentation.

First of all, you need to build the PointNet by following the architecture in Fig.1. In implementation, you will build separate networks for different tasks with different feature dimensions. We provide the off-the-shelf data-loaders of ShapeNetPart for both classification and segmentation tasks. The training process may take you about 20 minutes. The content of this question can be found in *model.py*.

¹Yi L et al. A scalable active framework for region annotation in 3d shape collections[J]. ACM Transactions on Graphics (ToG), 2016

a.1)[10 points] **For segmentation task**, your network should predict the part labels of the given point cloud. Specifically, we consider the "airplane" category. **Follow the instruction in *model.py* to complete the PointNetfeat, PointNetSeg classes.** Please refer to *train_segmentation.py* for training details. You will receive a full score if your training and testing curve looks normal and the best test accuracy is above 0.6.

a.2)[15 points] **For classification task**, your network should predict the category of given point clouds. **Follow the instruction in *model.py* to complete the PointNetCls256D, PointNetCls1024D classes.** You are expected to investigate the effect of the dimensions of the global feature. So in this classification part, you are requested to train two separate PointNets. One is the original PointNet with 1024D global feature. And another one uses 256D global feature.

You may find that a bigger network with more capacity non-necessarily outperforms a smaller network. Please check *train_classification.py* for more details. You will receive a full score if your training and testing curve looks normal and the best test accuracy is above 0.85.

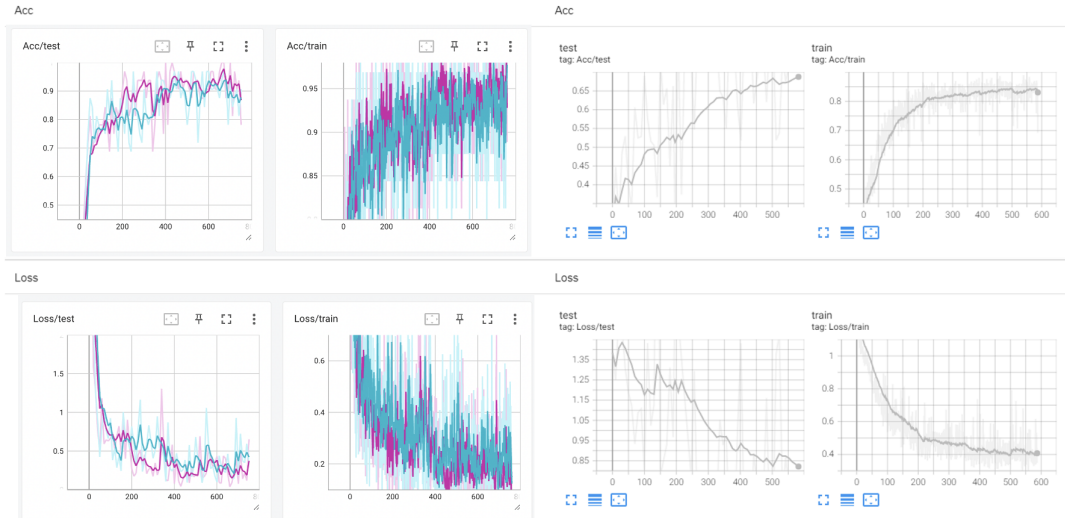


Figure 2: Classification task with 1024D (pink) and 256D (blue) global feature. Segmentation task with 1024D global feature (gray)

For submission, put the 3 screenshots of TensorBoard corresponding to the three experiments under the folder 'PointNet/results'. Name them as 'segmentation.png', 'classification_256.png', 'classification_1024.png'

b) [10 points] Point Feature Visualization

We have already trained some networks and draw lots of curves. But sometimes, we want a more comprehensive way to understand what the network actually learned. So in this question, you are required to visualize the *cruciality* of $n \times 1024$ points feature before max-pooling in the classification network(1024D). And the *cruciality* is simply defined as the maximum value along the point feature dimensions. The colormap and points-to-ply functions are provided and you can obtain similar colored point clouds as Fig.3. please see *feature_vis.py* for more details. Choose 4 among the 16 samples that best show the visual results.

For submission, put the 4 feature visualization screenshots under the folder 'PointNet/results' and name them as '0.png', '1.png', '2.png', '3.png'.

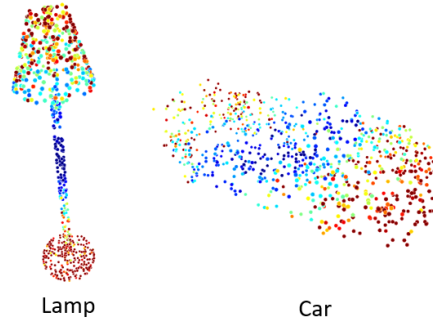


Figure 3: Feature visualization with colormap. Your results may not be exactly the same.

2. Self-Attention Captioning (30 points):

Self-attention is a core component in modern vision-language models. In this question, you are requested to implement causal self-attention and use it in a small Transformer captioning model. The model uses the same COCO image features and captions as the RNN task. The content of this question can be found in the folder *Attention*.

a)[8 points] Scaled dot-product attention

Please complete the *scaled_dot_product_attention* function in *Attention/attention.py*. Given query Q , key K , value V , and an optional additive mask, compute

$$S = \frac{QK^T}{\sqrt{d}}, \quad A = \text{softmax}(S), \quad O = AV. \quad (1)$$

If a mask is provided, add it to S before softmax. Valid positions should have value 0 and masked positions should use a large negative value such as -10^9 .

b)[8 points] Causal mask and multi-head self-attention

Please complete *make_causal_mask* and *MultiHeadSelfAttention* in *Attention/attention.py*. The causal mask should prevent each token from attending to future tokens. In the Transformer captioning model, the image feature is treated as the first token and every word token can attend to the image token and previous word tokens.

c)[8 points] Transformer captioning forward and sampling

We provide a small Transformer captioning model in *Attention/transformer_caption.py*. After your self-attention implementation is correct, this model should run a forward pass, compute the captioning loss, back-propagate with PyTorch autograd, and greedily sample captions. Run the public checks:

```
cd Attention
python check_attention.py
python check_transformer_caption.py
```

d)[6 points] Train on the same captioning task as RNN

You can train the Transformer captioning model on the same 50 COCO captioning samples as the RNN demo:

```
cd Attention
python train_transformer_caption.py
```

This script saves *transformer_loss_history.png* and four prediction images under *Attention/results*. We only expect a small overfitting demo; do not download or train real CLIP, LLM, VLM, or large Transformer models.

3. RNN (35 points):

Temporal analysis is a crucial part of computer vision. In this question, you are requested to implement a vanilla RNN for the image caption task on the COCO dataset. A sample of this dataset are shown in Fig.4. To cut down on processing time, we have extracted the image features using a pre-trained VGG-16 network and further processed them with Principal Component Analysis (PCA) to reduce the dimensionality to 512.

<START> a bowl of food that is sitting on a table <END>



Figure 4: A sample of COCO image caption.

a)[10 points] Implement a single vanilla RNN layer.

Specifically, you need to complete the *rnn_step_forward* and *rnn_step_backward* functions in the *rnn_layers.py* according to the following equation,

$$h_t = \tanh(h_{t-1}W_h + xW_x + b) \quad (2)$$

You should run ‘python check_single_rnn_layer.py’ to check your implementation and save your answer.

b)[10 points] Implement a RNN model for image captioning.

For simplicity, you are only required to complete the core functions *rnn_forward* and *rnn_backward* in the *rnn_layers.py* according to Fig.5. You should run ‘python check_rnn.py’ to check your implementation and save your answer. If you are interested about the complete implementation of the RNN model, you can refer to the *rnn.py* for more details.

c)[5 points] Train the RNN model on the COCO dataset.

You can run ‘python train_rnn.py’ to train the model. You will receive a full score if your loss curve looks normal.

d)[10 points] Visualize the results of the RNN model.

Please complete the sample function in *rnn.py* according to the comments in that file. Then uncomment the code in the *train_rnn.py* and run it again. Since we only overfit 50 training samples, the results on training dataset should be perfect while the results on validation dataset may not be that good. You will receive a full score if the training results are reasonable.

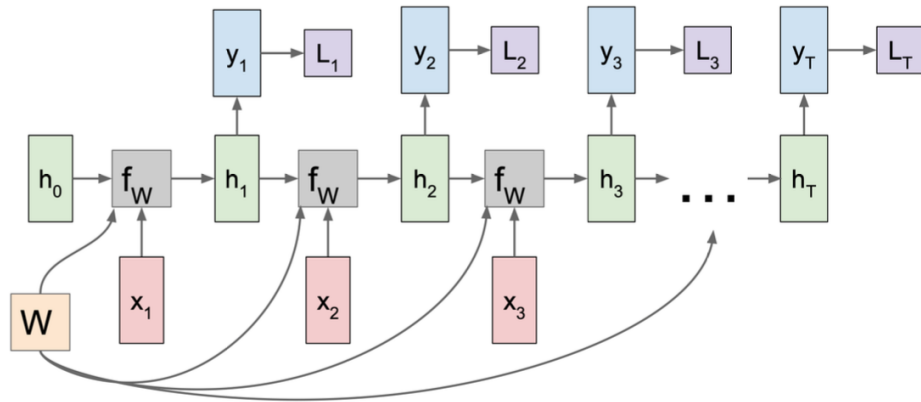


Figure 5: A computational graph of RNN for many-to-many task.