Ting-An Chen

Homework #3
Deep Learning for Computer Vision

Problem 1: Vision Transformer (ViT) (30% performance + 50% report results)

1. Report accuracy of your model on the validation set. (Result should be reproduced in error ±0.5%) (10%)
    a. Discuss and analyze the results with different settings (e.g. pretrain or not, model architecture, learning rate, etc.) (8%)
        - **Pretrain**: Yes (pretraned weights refers to the github)
            - I have tried to train from scratch. However, it did not work. It may be that the training data is not enough.
        - **Image resize**: 384x384
            - I have tried to resize to 224x224 which is often applied in many cases as what I have seen in github codes. However, 384x384 has a better performance. The reason may be that many of the training/testing images which sizes are over 300x300.
        - **Model architecture**
            - Here, I use the patch size = 16. I have tried size = 32. However, size 16 can achieve a better performance.
            - In addition, I use 12 transformer blocks as shown below.

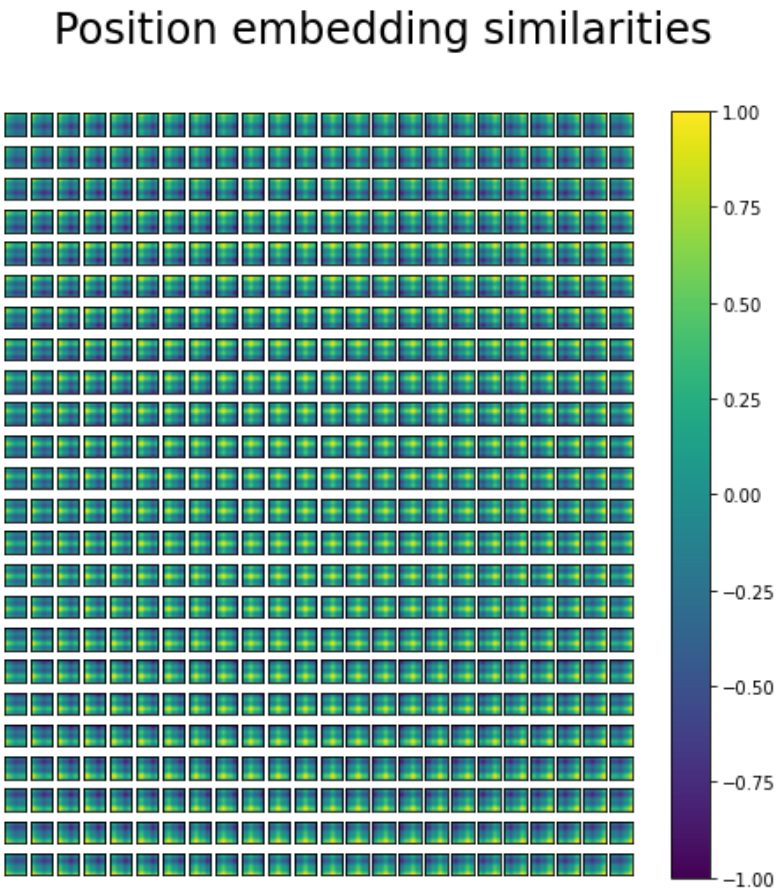
```
VisionTransformer(
  (patch_embed): PatchEmbed(
    (proj): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
    (norm): Identity()
  )
  (pos_drop): Dropout(p=0.0, inplace=False)
  (blocks): Sequential(
    (0): Block(
      (norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (attn): Attention(
        (qkv): Linear(in_features=768, out_features=2304, bias=True)
        (attn_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in_features=768, out_features=768, bias=True)
        (proj_drop): Dropout(p=0.0, inplace=False)
      )
      (drop_path): Identity()
      (norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
        (act): GELU()
        (fc2): Linear(in_features=3072, out_features=768, bias=True)
```

```
      (drop): Dropout(p=0.0, inplace=False)
    )
  )
  … Skipped. Block 1 to 10 are same as Block 0
  (11): Block(
    (norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
    (attn): Attention(
      (qkv): Linear(in_features=768, out_features=2304, bias=True)
      (attn_drop): Dropout(p=0.0, inplace=False)
      (proj): Linear(in_features=768, out_features=768, bias=True)
      (proj_drop): Dropout(p=0.0, inplace=False)
    )
    (drop_path): Identity()
    (norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
    (mlp): Mlp(
      (fc1): Linear(in_features=768, out_features=3072, bias=True)
      (act): GELU()
      (fc2): Linear(in_features=3072, out_features=768, bias=True)
      (drop): Dropout(p=0.0, inplace=False)
    )
  )
 )
 (norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
 (pre_logits): Identity()
 (head): Linear(in_features=768, out_features=1000, bias=True)
)
```

- **Learning rate**
  - LR = 0.0005, quite small since I used a pretrained model.

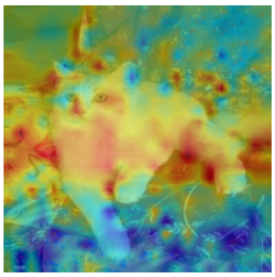
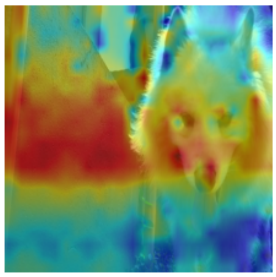
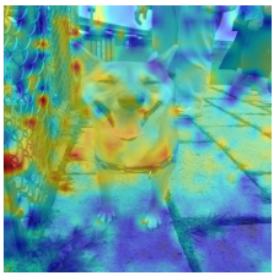
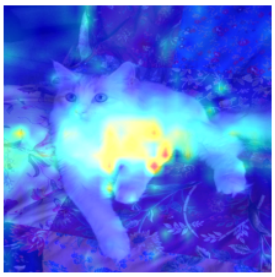
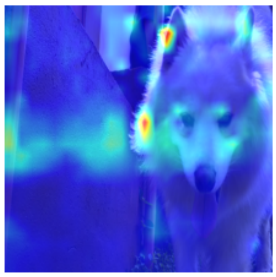
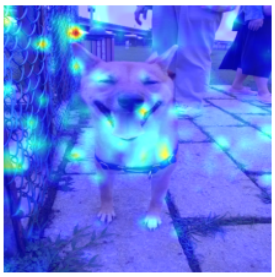
b. Clearly mark out a single final result to reproduce (2%)
Validation accuracy: **95.133%**

2. Visualize position embeddings of your model. (20%)
    a. Visualize cosine similarities from all positional embeddings (15%)



    b. Discuss or analyze the visualization results (5%)
        - Image size 384 / patch size 16 = number of patches 24 → 24x24 patches
        - Results: Correlations (each patch embedding, others)
        - Middle patches are generally highly correlated to more other patches since patterns often reside in the middle.
        - The near patches are generally tightly correlated.

3. Visualize attention map of 3 images. (p1_data/val/26_5064.jpg, p1_data/val/29_4718.jpg, p1_data/val/31_4838.jpg) (20%)
    a. Visualize the attention map between the [class] token (as query vector) and all patches (as key vectors) from the LAST multi-head attention layer. Note that you have to average the attention weights across all heads (15%)

	26_5064.jpg	29_4718.jpg	31_4838.jpg
Original images (resized to 384x384)			
Attention (weights)			
Attention (|weights|)			

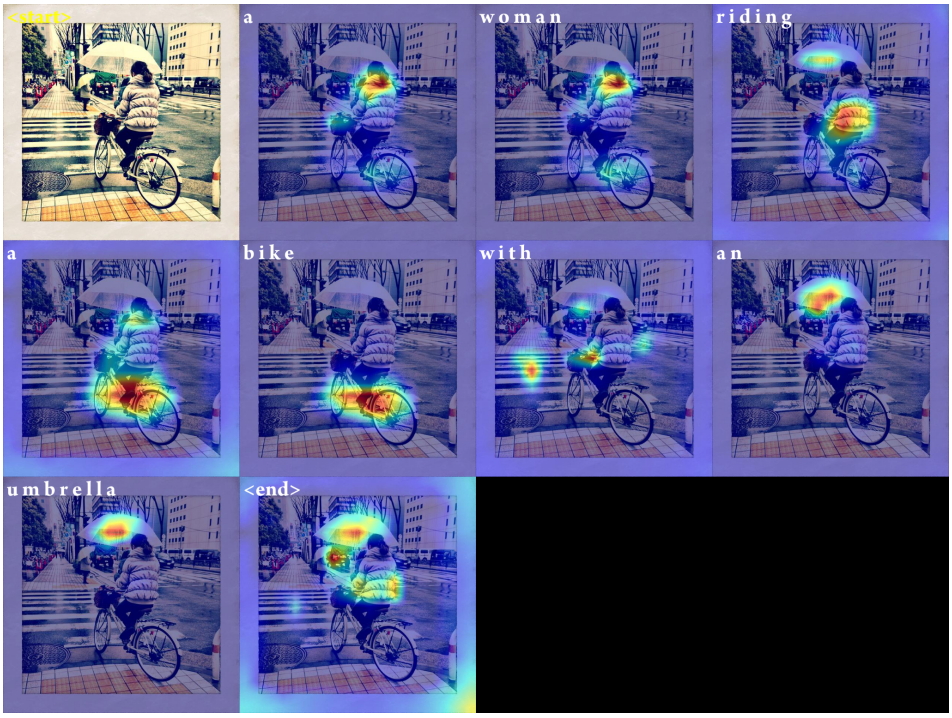
b. Discuss or analyze the visualization results (5%)
The model can well recognize the cat and wolf by focusing on the cat's belly and the wolf's face and fur. However, the attention in the dog figure is ambiguous. The reason may be that there is a net beside the dog. It may distort the recognition result. In a summary, the background is important in image classification.

Problem 2: Caption Transformer (20%)

1. For the five test images, please visualize the predicted caption and the corresponding series of attention maps in a single PNG output. (10%)
    a. Save the five visualization results (PNG images) in the specified folder directory.
    b. Name your output PNG images as follows (same as the input filename):
        ■ bike.png
        ■ girl.png
        ■ sheep.png
        ■ ski.png
        ■ umbrella.png

2. Choose one test image and show its visualization result in your report. (10%)
    a. Analyze the predicted caption and the attention maps for each word. Is the caption reasonable? Does the attended region reflect the corresponding word in the caption?

        ■ bike.png



The attention results are quite reasonable. The nouns, “woman”, “bike”, and “umbrella”, where the attention is in the correct region. The verb, “riding”, is also focused on in the reasonable region. On the other hand, the abstract terms such as “with” and “a” which attention can hardly show the meaning on the figure.

b. Discuss what you have learned or what difficulties you have encountered in this problem.
I consider that to finish the attention visualization, the first priority for me is to figure out and realize the meaning of the query and key vectors in this task. In the following, the meaning of the dimension of the vectors is also critical.

Actually, I have also taken a selfie to examine the prediction and the attention results. It can be observed that the caption results will depend on the words which have been trained before.