DLCV HW3 Report

電機四 b07901039 劉知穎

Collaborator: B07303024 林品樺

Problem 1: Image Classification with ViT

1. Accuracy on the validation dataset: 0.9513

   Setting:

   a. Use "B_16_imagenet1k" ViT model and pretrained weight from [1].
   b. Resize images to (384, 384) which is suggested by the model configuration. Add data augmentation, including RandomHorizontalFlip, RandomRotation, and ColorJitter.
   c. Set batch size 8, because larger batch size cause CUDA out of memory error.
   d. Train 40 epochs.
   e. Use small learning rate and noam scheduler to update the model. For noam scheduler, linearly increase the learning rate until warm up step, then exponentially decrease the learning rate. The actual learning rate for each step is shown in Figure 1.

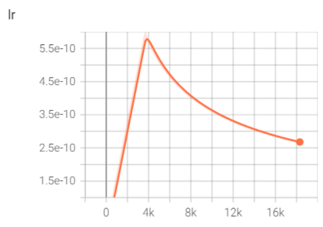


Figure 1: Learning rate scheduling

   Discussion:

   a. Because ViT is a very large pretrained model, the finetuning learning rate must be small enough. When learning rate was 1e-4, the loss function fluctuated severely, and the validation accuracy was only 0.8027. However, when learning rate was 1e-6, the loss function is more stable, and after adding data augmenation the validation accuracy improved to 0.9393.
   b. Scheduling the learning rate by the Noam scheduler, the accuracy improved from 0.9393 to 0.9513.

2. Visualize positional embedding

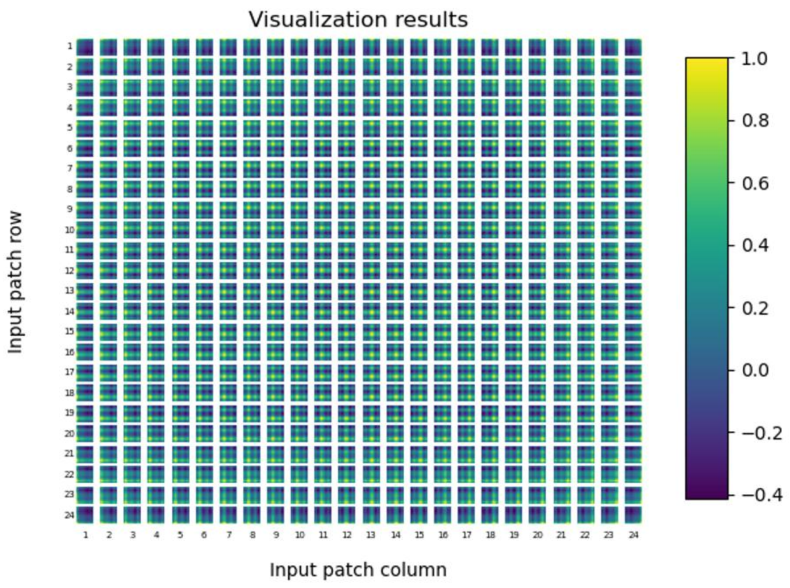


Figure 2: Visualization of positional embeddings

Discussion:

My model has input image size (384, 384), patch size (16, 16), hidden dimension 768. Therefore, there are 24 * 24 768-dim positional embeddings. Get the weights of positional embeddings from the state_dict of the model, normalize them, and do inner product by matrix multiplication. Resize each row of the resulting matrix into (24, 24) and plot them as Figure 2.

The (i, j) figure means the cosine similarity of the (i, j)th patch with all other patches. For example, the (1,1) figure is the cosine similarity of the (1, 1) patch (the most upper left patch) with all other patches. We can see that each cosine similarity figure has the brightest pixel corresponding to its patch index, which is reasonable because it is the cosine similarity of two same vectors.

3. Visualize attention maps

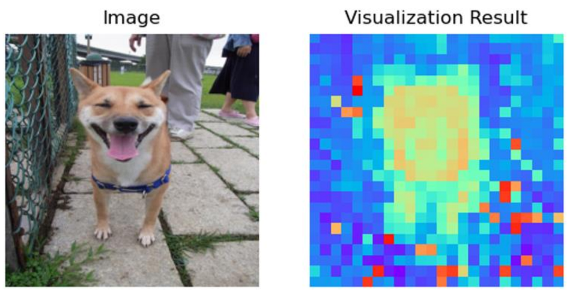


Figure 4: Attention map of 31_4838.jpg

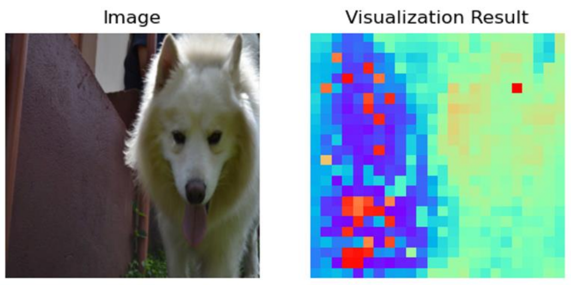


Figure 3: Attention map of 29_4718.jpg

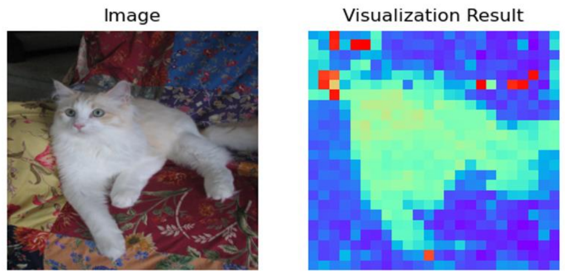
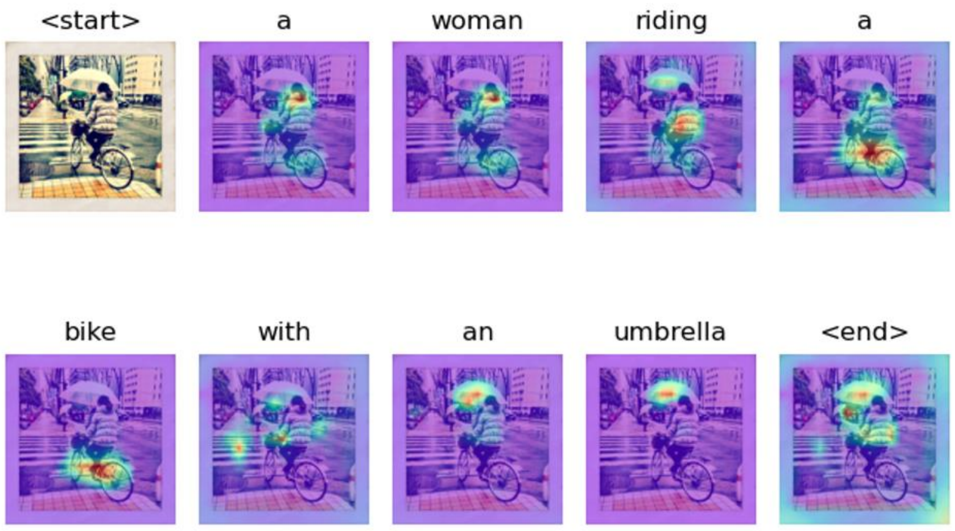


Figure 5: Attention map of 26_5064.jpg

Discussion:
Use activation hook to get the query and key vectors for each patch and each attention head, use matrix multiplication to compute the attention maps, and average over all attention maps to get the final visualization result.
Pixels of the objects have higher attention weights, which means that the model successfully focuses on the front ground object instead of the background. In addition, there are higher weights focusing on the face of the cat and the dogs. However, there are also a few sparse large attention weights distributing on the background.

Problem 2: Visualization in Image Captioning



1. Analysis
   The caption is reasonable. The attended region reflects corresponding caption. For example, the model knows to focus on the woman when the caption is “woman”, and same goes with “bike” and “umbrella”.
2. Discussion
   Resize each image into (640, 640). The Resnet backbone extracted features into (2048, 19, 19) vectors. The extracted feature is fed into the transformer. Modify

the TransformerDecoderLayer to get the attention weight of the last attention head, which is of size (128, 361), where 128 is the maximum sequence length. At last, resize the attention map at each time step back to (640, 640) for better resolution and overlap the attention map with the original image.
At first, I wrongly match the attention map with the corresponding output. I originally thought that the attention map is matched to the current time step input. For example, at the first time-step (index 0), the input of the transformer decoder is “<start>”, and its corresponding attention map is at index 0. Later, I realize that the attention map should be matched to the current time step output. For example, at the first time-step (index 0), the output of the transformer is “a”, so the index 0 attention map should belongs to “a”.

Reference:

1. https://github.com/lukemelas/PyTorch-Pretrained-ViT
2. https://github.com/saahiluppal/catr