COMPX326 Assignment 4
Computer vision applications

Abstract

In this assignment, you will use a DINO-pretrained ViT to produce attention-based saliency maps.

1 Tasks

There are two tasks you need to perform for this assignment. We will continue using the `dog.jpg` file from Assignment 2, so ensure that you still have this file. Download the provided `assignment4.py` file from Moodle and move it into the `COMPX326` directory. This file contains the skeleton code for this assignment. You can follow the comments in this code to complete the assignment. Test code is provided in the lower half of the file. After completing all tasks, uncomment the test code by removing the `'''` marks around it to test it in your run. Do not modify the names of classes, functions, or variables provided in the file. This is the file you will submit for marking. It is permitted to import additional modules required by your implementation, but you should not need to.

1.1 Task 1

Your first task is to complete the `forward_wrapper()` function. We will be using ViTs implemented in the `timm` library, and to get the attention scores we need to replace their attention mechanism's `forward()` function. Within this wrapper function, we will define our own `forward()` function and return it at the end.

Our `forward()` function should be a modified version of the original. See lines 86-107 of `https://github.com/huggingface/pytorch-image-models/blob/main/timm/models/vision_transformer.py` for the original function. Use it as your template.

The `if` statement in the original function can be simplified. As we will perform our experiments on the CPU, fused attention will not be used. Replace the whole `if` routine with only the code under the `else` statement.

As our goal is to examine the attention scores, after the `attn` variable is fully computed, save it as a variable of `self` named `attn_map`. The rest of the

code can remain unchanged. Return this `forward()` function at the end of the wrapper function.

Examine how the test code uses this wrapper function: It replaces the `forward()` function of the last attention block of the ViT, and the saved `attn_map` variable can be examined after forward propagation.

1.2 Task 2

Your second task is to complete the `get_attn_map_at_head_and_index()` function. Given an attention map, as well as an attention `head` number, a token `index`, a `height` and `width` of the attention map, it returns a formatted attention map ready to be displayed.

The input attention map is of dimensions (H, L, L), where H is the number of attention heads, and L is the sequence length. The returned attention map should be of dimensions $(height, width)$, by reorganising the sequence into a 2D map of patches. As a special [CLS] token is prepended to the sequence, L, $height$, and $width$ satisfy the following relation: $L = height * width + 1$.

The given `head` and `index` arguments should be used to index the two leading dimensions of the input attention map, obtaining a sequence of attention scores of length L. As we need to map the attention scores to the image, remove the first score (of the special [CLS] token) and reshape the remaining sequence into a 2D map of dimensions $(height, width)$.

In a special case, if `head` is given as -1, return the attention map averaged over all attention heads instead of from one head.

After completing all code, run the test code and examine what it does. The test code will evaluate two ViTs pretrained using DINO: one with patch size 16×16 and the other with patch size 8×8. For each of a ViT's six attention heads as well as their average, four images will be displayed: 1) an attention map of the [CLS] token, 2) this attention map overlaid with the image, 3) the full $L \times L$ attention map, and 4) the full attention map transformed to match the patches.