Homework 5 due: 6/8 11:59 pm

Submission Instructions

You will need to submit the following materials:

1. **Questions and Answers** in PDF format. Remember to put your name in your report.
2. A folder containing all ipynb files with your results for the the problems. Please remember to keep all results and logs in the submitted ipynb files to get full credit.

All submitted files should be put into one single zip file named as HW#_xxx.zip, e.g. HW5_George_Clooney.zip**,** including all ipynb files with answers (both results and discussions), and the **Questions and Answers** report.

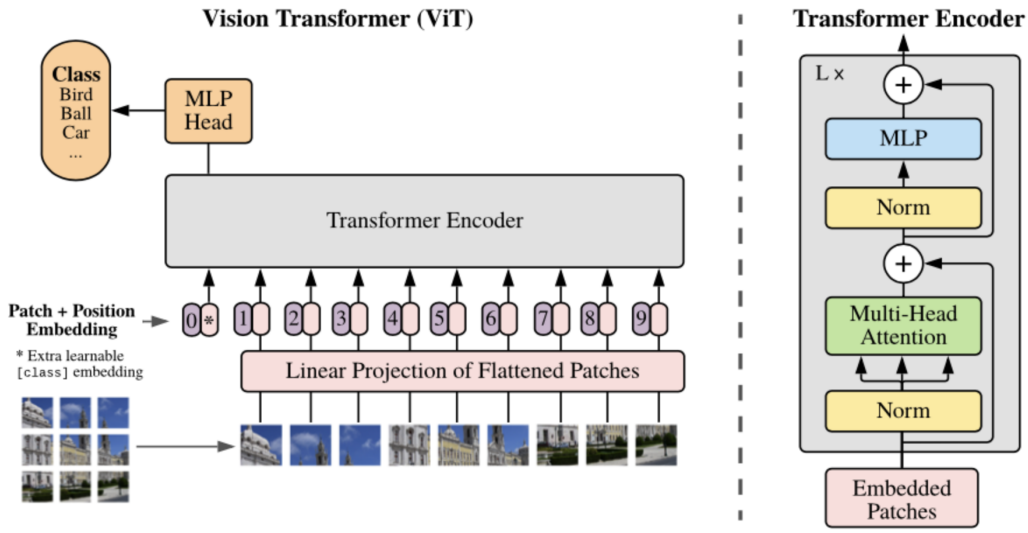
Problem 1: PointNet (30%)

You will implement PointNet (https://arxiv.org/abs/1612.00593) for point cloud object classification. Please find the details of each question in HW5.ipynb.
Questions

1. PointNet model implementation (15%)
2. PointNet training (10%)
3. Inference and visualization (5%)

Problem 2: Vision Transformer for Image Classification (45%)

The Vision Transformer (ViT) is basically BERT, but applied to images. It attains excellent results compared to state-of-the-art convolutional networks. In order to provide images to the model, each image is split into a sequence of fixed-size patches (typically of resolution 16x16 or 32x32), which are linearly embedded. One also adds a [CLS] token at the beginning of the sequence in order to classify images. Next, one adds absolute position embeddings and provides this sequence to the Transformer encoder.



- Paper: https://arxiv.org/abs/2010.11929
- Official repo (in JAX): https://github.com/google-research/vision_transformer

Task A: Performing Inference with the Vision Transformer for Image Classification
Let's simply load in an image to let the ViT model perform inference on it. Please follow the Notebook for the following steps:

(1) **Load Pre-trained Model**: Install the relevant libraries and load the pre-trained model
(2) **Pre-Processing**: As mentioned in the paper, the model accepts an input resolution of `224x224`. The sample we have here is (640, 480). Here we use `ViTFeatureExtractor` to take care of `resizing` + `normalization`.
(3) **Forward Pass**: Let's send the image through the ViT model, which consists of a BERT-like encoder and a linear classification head on top of the last hidden state of the [CLS] token.

`Predicted class: Egyptian cat`

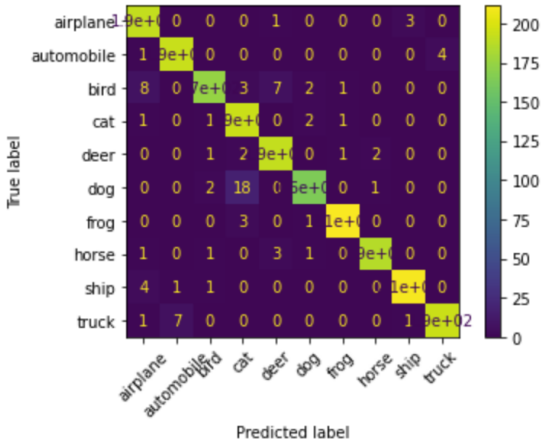
Questions:
1. (5%) Please find another image from the web (or you can upload them from local as well, any image that is good for classification is fine), and use the ViT-B model to get the predictions.
2. (5%) As we know, pre-training a ViT model often requires lots of data and computation resources and that's why we use the pre-trained model weights provided by the official. Please visit the HuggingFace model hub for ViT model, and choose one to replace the `google/vit-base-patch16-224` in the following codes. Run the inference using the new pre-trained model checkpoints again and get the predictions.
(NOTE: Some of the pre-trained weights may be too large to fit for your GPU, choose the one that works.)
3. (5%) **Clearly** explain why the [CLS] logits have a shape of [1, 1000]. What if we train the model on CIFAR-10 from scratch, will this shape of logits change? (Yes/No), How? (The shape).
4. (5%) What if I already had a trained model on ImageNet1k, in this case, a `vit-base-patch16-224` checkpoint , and now I have a new dataset, e.g., CIFAR-10, coming with multiple new classes. What do you think is the best strategy to adapt my existing models to the new datasets? (NOTE: Here is the link fine-tuning for transfer learning for your reference. You can read it first and then come back to answer this question.)

Task B: Fine-tune the Vision Transformer on CIFAR-10
In this section, we are going to fine-tune a pre-trained Vision Transformer on the CIFAR-10 dataset. This dataset is a collection of 60,000 32x32 colour images in 10 classes, with 6000 images per class, which we have been familiar with in the past.

We will prepare the data using HuggingFace datasets, and train the model using Pytorch. Please follow the Notebook for the following steps:

(1) **Preprocessing the data**: Here we import a small portion of CIFAR-10, and prepare it for the model using `ViTFeatureExtractor`. This feature extractor will resize every image to the resolution that the model expects (let's use the default 224), and normalize the channels.
Note that in the ViT paper, the best results were obtained when fine-tuning at a higher resolution. For this, one interpolates the pre-trained absolute position embeddings.
(2) **Define the model** The model itself uses a linear layer on top of a pre-trained `ViTModel`. We place a linear layer on top of the last hidden state of the [CLS] token, which serves as a good representation of an entire image. We also add dropout for regularization.
(3) **Train (Fine-tune) the model**

Questions:
5. (5%) Fine-tune the model based on CIFAR-10 dataset using given `vit-base-patch16-224-in21k`, which is a **ViT-B model pre-trained on ImageNet-21k** (14 million images, 21,843 classes) at resolution 224x22, for 3 epochs with **end-to-end setting** (it might take ~20 mins on Colab). Report the accuracy on the test set, show the whole tensorboard training logs and the confusion matrix from sklearn.



6. (10%) Fine-tune the model based on CIFAR-10 dataset using `vit-base-patch16-224-in21k`, which is a **ViT-B model pre-trained on ImageNet-21k**, for 3 epochs with **backbone (encoder) fixed setting** (see torch.no_grad() for details). Please also report the accuracy on the test set, show the whole tensorboard training logs and the confusion matrix from sklearn.

7. (5%) Think about the difference between these two settings (end-to-end vs. backbone (encoder) fixed). Why we need the second setting, especially when we are facing even larger models (e.g., ViT-L `vit-large-patch16-384-in21k`)?

Problem 4: Questions and Answers (25%)

1. (15%) Can you illustratively describe the evolutions of three widely used RCNN families, from Faster RCNN for object detections, mask RCNN for instance segmentation and mask RCNN for human pose estimation (i.e., Keypoint RCNN). In each evolution, what are the new additions of components and their corresponding training and inference strategies.

2. (10%) Why do we always need a simple symmetric function in the PointNet and PointNet++ to perform the classification of point cloud data? On the other hand, why we do not need this simple symmetric function for segmentation of point cloud data? Can you also verbally describe what is the necessity of introducing T-Net in the PointNet and PointNet++, what is effect of using T-Nets and the evidence of performance improvement.