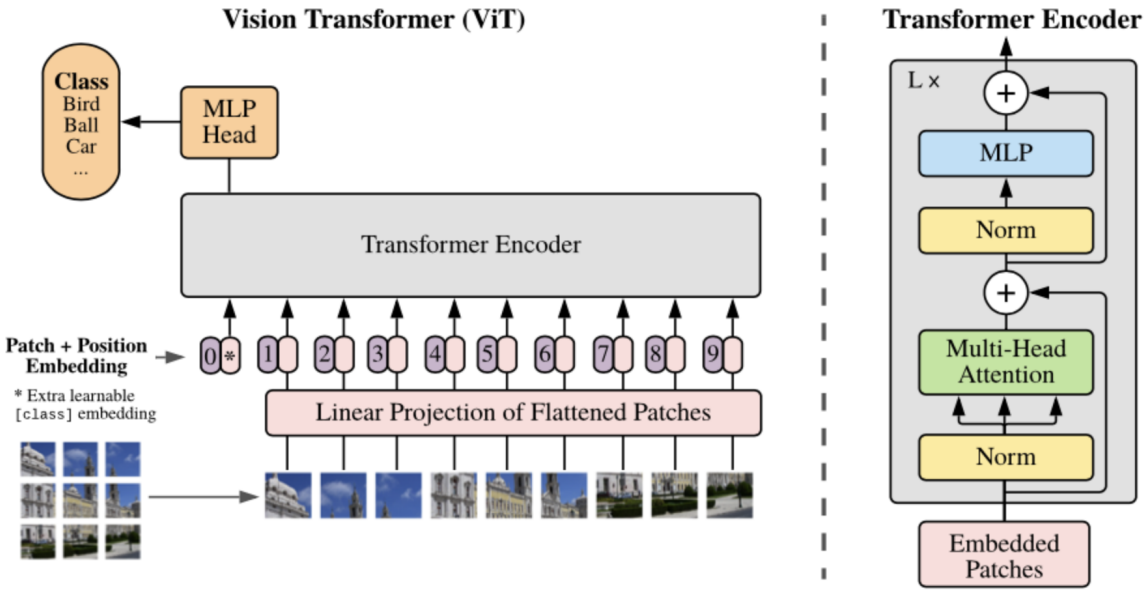
PMP596 Homework 5

Due December 16, Thursday, 5:59 PM

Problem 1: Vision Transformer for Image Classification (40%)

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

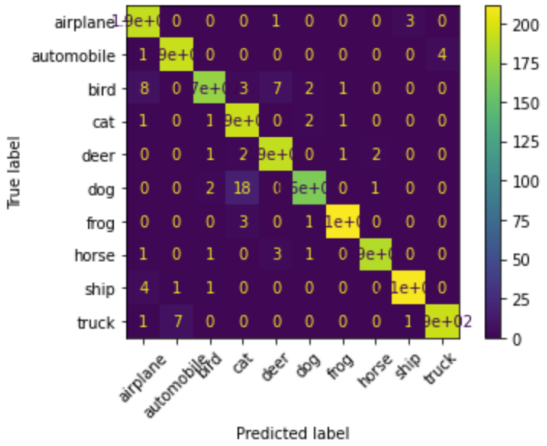
In this section, we are going to fine-tune a pre-trained Vision Transformer on the CIFAR-10 dataset. This dataset is a collection of 60,000 32x32 colour images in 10 classes, with 6000 images per class, which we have been familiar with in the past.

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:

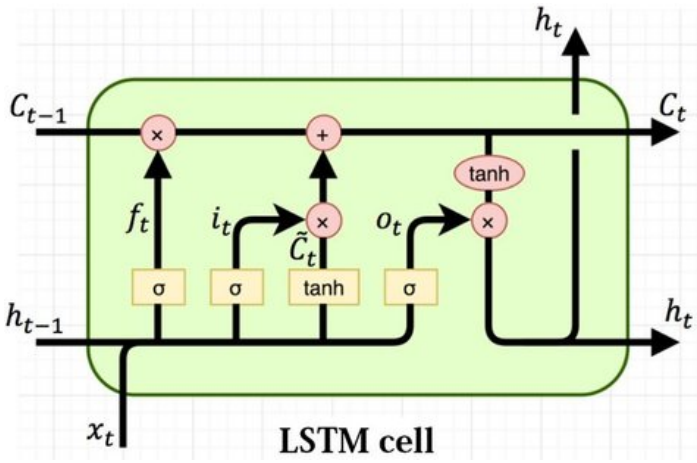
5. (5%) Fine-tune the model based on CIFAR-10 dataset using given `vit-base-patch16-224-in21k`, which is a **ViT-B model pre-trained on ImageNet-21k** (14 million images, 21,843 classes) at resolution 224x22, for 3 epochs with **end-to-end setting** (it might take ~20 mins on Colab). Report the accuracy 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 2: LSTM for Music Genre Classification (40%)

In this problem, we will implement a long short-term memory (LSTM) network for music genre classification on the GTZAN dataset.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure: Illustration for an LSTM cell.

(a) Prepare Dataset

Music experts have been trying for a long time to understand sound and what differentiates one song from another. How to visualize sound. What makes a tone different from another. The GTZAN dataset hopefully can give the opportunity to do just that.

In this problem, we split the dataset into training, validation, and testing sets:

- Training: 420 sequences
- Validation: 120 sequences
- Testing: 60 sequences

There are eight classes included:

```
genre_list = [
    "classical", "country", "disco", "hiphop",
    "jazz", "metal", "pop", "reggae",
]
```

Download GTZAN dataset from the link on Canvas. Put the zip file in your Google drive and mount it to your Google Colab notebook. Unzip the files using the provided scripts.

(b) Audio Feature Extraction (10%)

In HW1, we used MFCC for audio feature extraction. Now, we are going to try more feature extraction methods for the later genre classification.

The following four feature extraction methods should be considered in your answers:

- MFCC: `librosa.feature.mfcc`
- Spectral centroid: `librosa.feature.spectral_centroid`
- Chromagram: `librosa.feature.chroma_stft`
- Spectral contrast: `librosa.feature.spectral_contrast`

The feature dimensions can be determined by yourself. But please identify the feature dimensions clearly in your submissions. Visualize MFCC and Chromagram for some sample data. (See `librosa` documentation).

(c) LSTM Implementation (15%)

Define an LSTM network with the following parameters:

- Number of layers: 2
- Hidden layer dimension: 128
- Loss function: cross entropy loss
- Optimizer: Adam

Note that we only use the output of the **last step** of the sequence for classification purposes. This output is then sent into a fully connected layer to accomplish the final classification. For other parameters that are not mentioned, you can define them yourself, and finetune them during the training.

Remember to keep the losses to visualize the training loss during the training.

(d) Evaluation (10%)

- Use your trained model to do inference on the testing set. Report your classification accuracy.
- Now let's do an ablation study. Train another LSTM model using **MFCC features ONLY**. What classification accuracy can you achieve on the testing set now?

(e) Try a customized music (5%)

Find a piece of customized music, e.g., download from the internet. Do audio feature extraction and pass through your trained LSTM. How is the prediction result?

Problem 3: Paper Reading (20%)

Title: Attention Is All You Need
Authors: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin
Abstract: The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.