Vision Transformer for Image Classification

Introduction

In this project, I have implemented a Vision Transformer(ViT) in PyTorch and used it for the classification of images. A custom Vision Transformer(ViT) network was created, consisting of multi-headed attention layers in an encoder network. The network was trained and validated using a subset of images present in the MS-COCO dataset(2014 version) which was downloaded with the help of COCO API, a library for managing the dataset. Finally, the performance of the network was compared by computing the confusion matrix to understand which images were correctly or incorrectly labeled for each of the classes. Some ideas related to the logic of downloading the COCO dataset, running the Transformer training and validation loops were taken from the source code of the DLStudio module for completing this assignment. Some help was also taken from the blog post on Vision Transformers.

Model Training

The optimizer used was Adam with beta values of (0.9,0.99), the learning rate was 0.0001 for 10 epochs and the batch size was 32. The loss function was Binary Cross Entropy Loss.

Model Architecture

A Vision Transformer consists of the following parts:

Patch Embedding

The following steps are involved in this:
(a) We divide the image into patches of fixed size (16×16 for this homework) and generate the embeddings for each patch. This is done by passing the image through a convolutional layer with stride and kernel size equal to the size of the patch and the number of output channels being equal to the embedding size (384

for this homework). We then permute this tensor so that its of shape (batch size, number of patches, embedding size).
(b) Then we append the same learnable class token of size (1,embedding size) for each image of the batch to the overall embeddings.
(c) Finally we create the learnable parameters for the position embedding of each patch. It is of shape (number of patches +1 , embedding size) for each image of the batch. We then add this to the elementwise to the overall embeddings for each batch.
Therefore the final output of this block is of shape (batch size, number of patches +1, embedding size). This block's learnable parameters help capture the image's short-range dependencies.

Transformer Master block

Each transformer master block consists of "d" transformer blocks within them (for this homework, I have taken it as 6). Each transformer block consists of Layer Normalization followed by a Multi-Head attention block and another Layer Normalization
followed by a MLP layer. There are skip connections between the layers.
In the Multi-headed attention block, we use learnable weights to compute the Query,
Key and Value for every patch. Then we split the embeddings for every patch into multiple heads (64 heads in this HW) so that multiple relationships can be learnt between the input patches. Next, we compute the relationship between all the patches
(which is called attention) using the Query and Key. This value is then normalized using softmax and the new Value is computed using the attention and the old Value.
Finally, the outputs Value embeddings of the multiple heads are reshaped together.

Classification head

The class embedding obtained from the transformer block is passed into the classification head, which gives out the 5-class logits.

Conclusion

In conclusion, a vision transformer can perform well in Image classification. However, the architecture of the transformer layers should be designed carefully to get better accuracy and faster training time. Although increasing the number of Basic Encoder blocks and the number of attention heads may result in higher classification accuracy, it may lead to slowness in training due to the increase in the number of trainable parameters and often leads to out-of-GPU memory. Also, some regularization in the form of dropout may control model overfitting to some extent. Hence, well-designed transformer architectures containing these practical layer elements like dropout or LayerNorm along with appropriate hyperparameters are essential in making the training much more efficient and stable and getting higher classification accuracy.