Artificial Intelligence

Assignment 2

Dahuin Jung

School of Computer Science and Engineering

Soongsil University

2024

Assignment2-1 Objective (Transformer)

- Problem: Implementing Transformer from scratch
  - To understand Transformer architecture
  - Implement 4 components of the Transformer
    - Positional Encoding
    - Multi-head attention
    - Encoder
    - Decoder

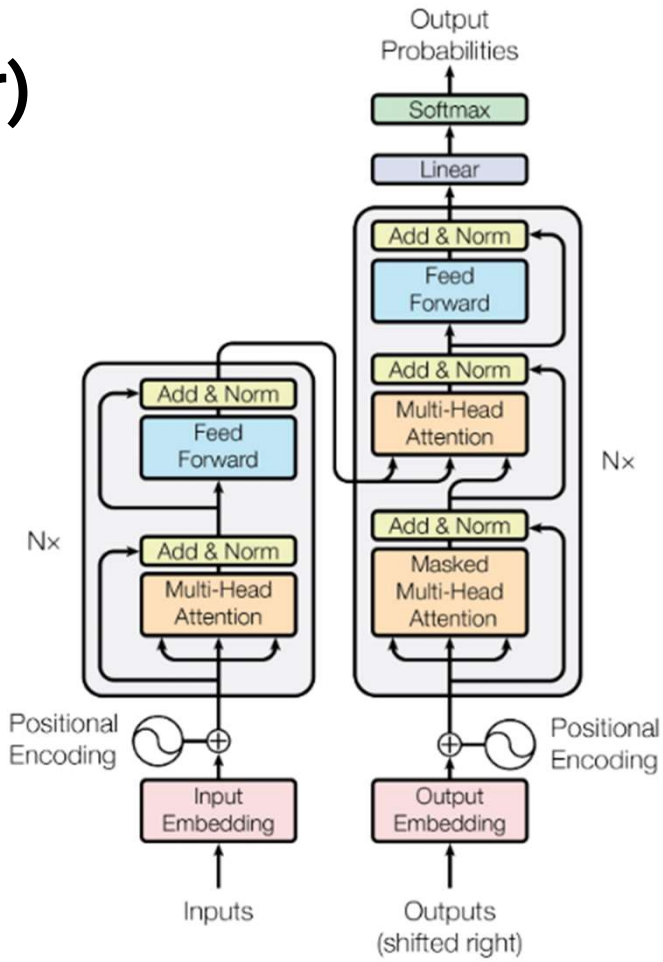


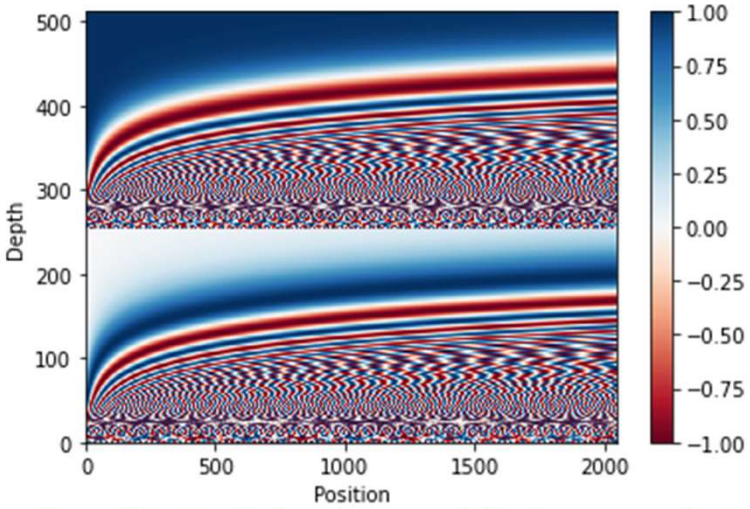
Figure 1: The Transformer - model architecture.

Source: Attention is all you need (NeurIPS 2017)

Implementing Transformer from scratch

- Four key components of the Transformer
  - Positional Encoding
  - Multi-head attention
  - Encoder
  - Decoder

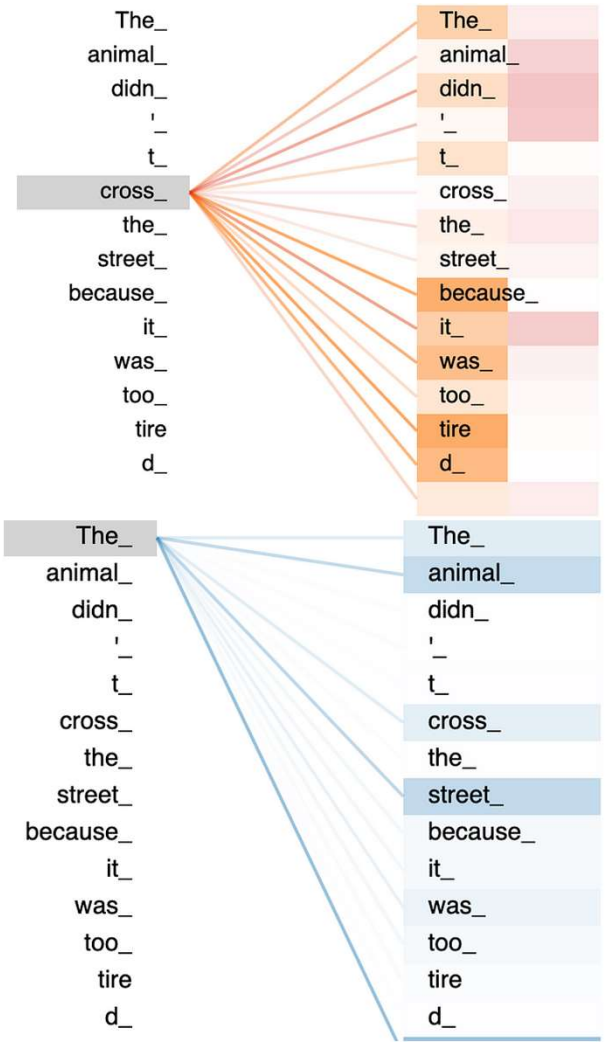
- Positional Encoding
  - Means to convey order information
    - $PE_{(pos,2i)} = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$
    - $PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$
  - If we draw the above equation, it looks like the left image



Source: TensorFlow tutorial: Transformer model for language understanding

Implementing Transformer from scratch

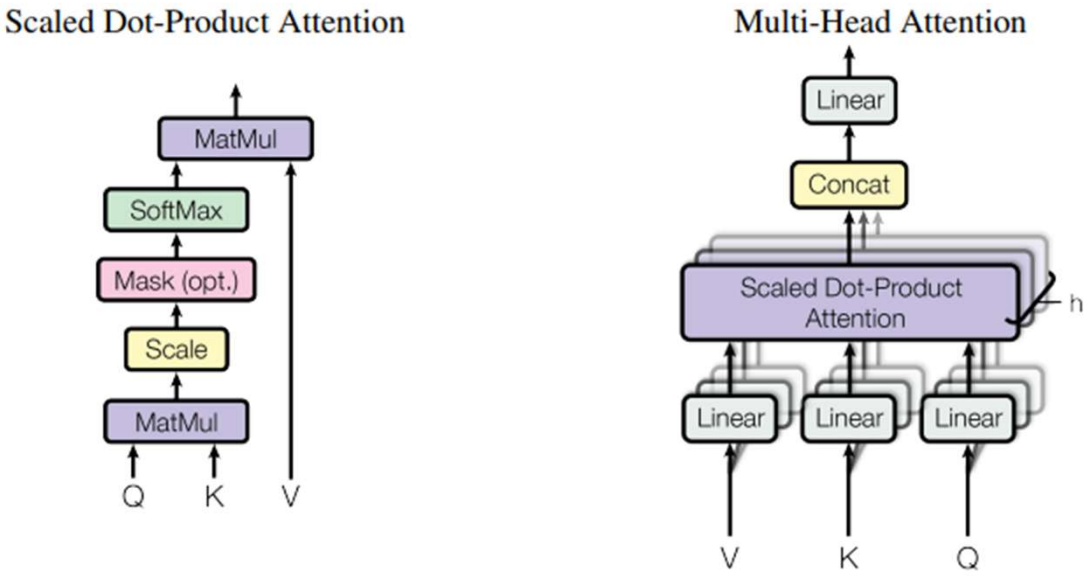
- Multi-head attention layers learn various features of a sequence
  - Which do you like better, coffee or tea? Sentence type
  - Which do you like better, coffee or tea? Noun
  - Which do you like better, coffee or tea? Relationship
  - Which do you like better, coffee or tea? Emphasis



Source: https://towardsdatascience.com/self-attention-5b95ea164f61

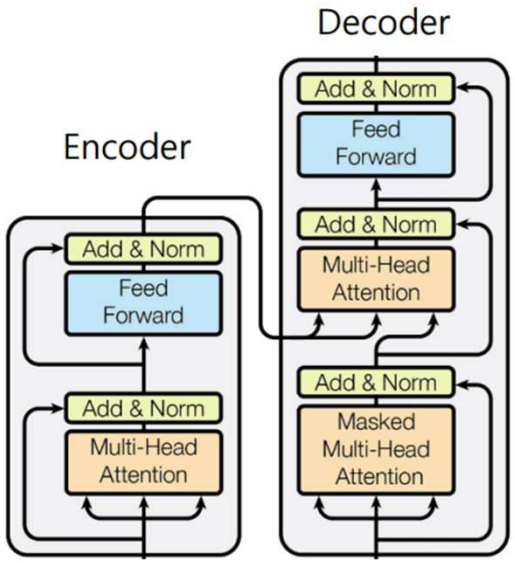
Implementing Transformer from scratch

- Specific equation of multi-head attention layer
  - $Q = X * W_q$
  - $K = X * W_k$
  - $V = X * W_v$
  - $scores = \frac{QK^T}{\sqrt{word\_dim}}$
  - $masked\_scores = mask(\frac{QK^T}{\sqrt{word\_dim}})$
  - $probs = softmax(masked\_scores)$
  - $heads = probsV$
  - $output = heads * W_o$



Source: Attention is all you need (NeurIPS 2017)

Implementing Transformer from scratch



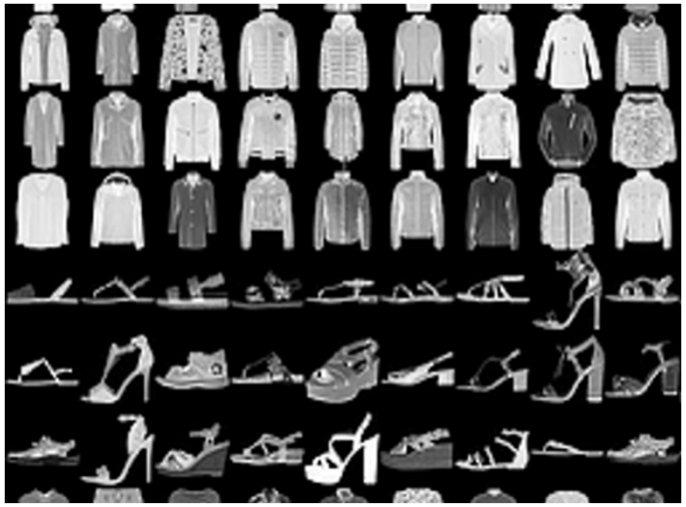
Source: Attention is all you need (NeurIPS 2017)

- Encoder
  - One Multi-Head Attention layer, one FFN layer, two Normalization layers
- Decoder
  - Two Multi-Head Attention layers, one FFN layer, three Normalization layers

Assignment2-2 Objective (Vision Transformer)

- Problem 1: Implement the code for the Vision Transformer (ViT) backbone, referring to the Transformer implemented in Part 1.
  - Considering the characteristics of using images as input, directly implement the key modules of the ViT structure according to the description.
- Problem 2: Train the ViT model using the implemented code (FashionMNIST).
  - Train the ViT model on the FashionMNIST dataset by changing hyperparameters
  - Train a model that achieves at least 90% performance.
  - Conduct experiments with at least 5 different hyperparameter settings and provide an analysis for each experiment.

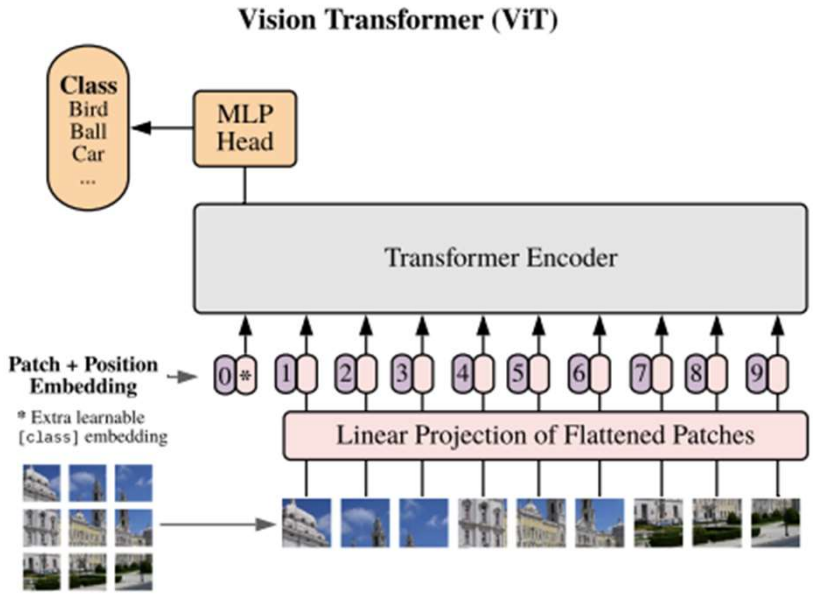
FashionMNIST dataset



Source: https://www.kaggle.com/datasets/zalando-research/fashionmnist

- Each image in the FashionMNIST dataset is associated with a specific label,indicating the type of clothing item depicted in the image.
- Consists of 70,000 28x28 grayscale images in 10 classes, with 6,000 images per class
- There are 60,000 training images and 10,000 test images.

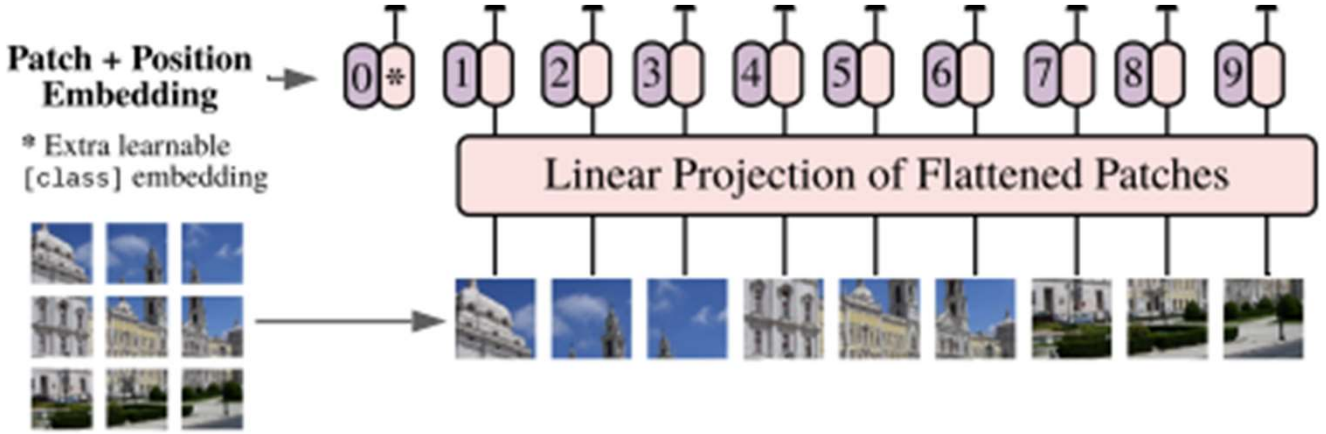
Vision Transformer (ViT)



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021)

- ViT leverages the transformer architecture, which was originally proposed for natural language processing tasks.
  - Patchify image into patch embeddings for encoder input
  - Multi-Head self-Attention mechanism learns global relationships between image patches
  - MLP head performs target downstream task, e.g., classification.

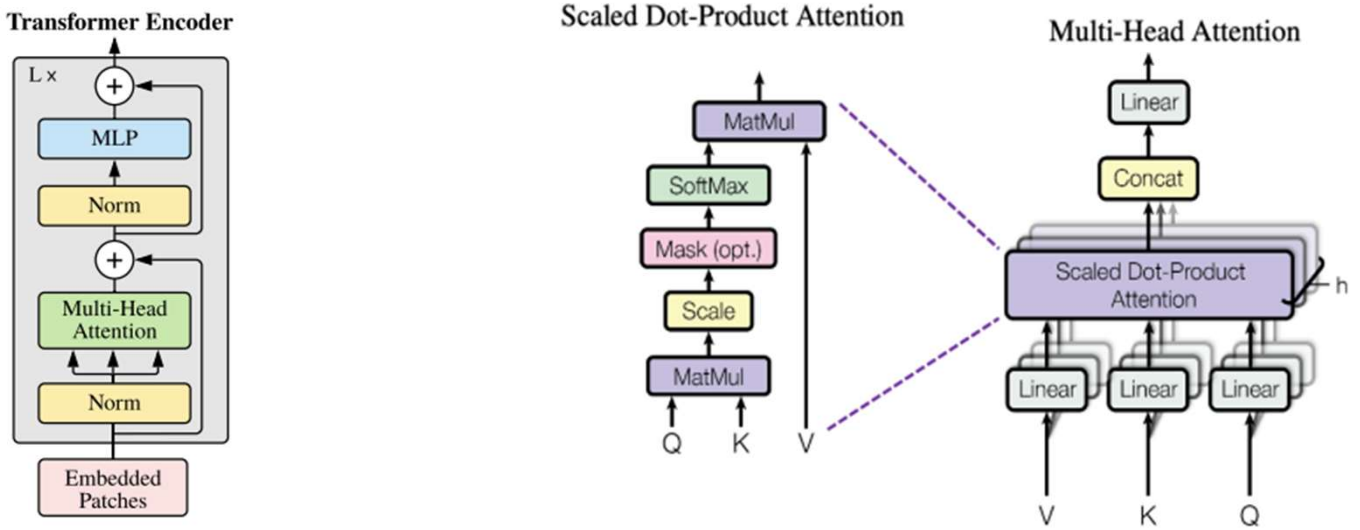
Patch Embed



Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021)

- **Patchify**:
  - Split the input image into small patches.
  - Flatten the patches and linearly project them into embedding dimension.
- **Class Token**: A special learnable token that is added to the sequence of patch embeddings.
- **Learnable positional Encoding**: Instead of using fixed functions to compute positional encodings, ViT learns these encodings as part of the training process.

Encoder (Multi-Head Attention & MLP)

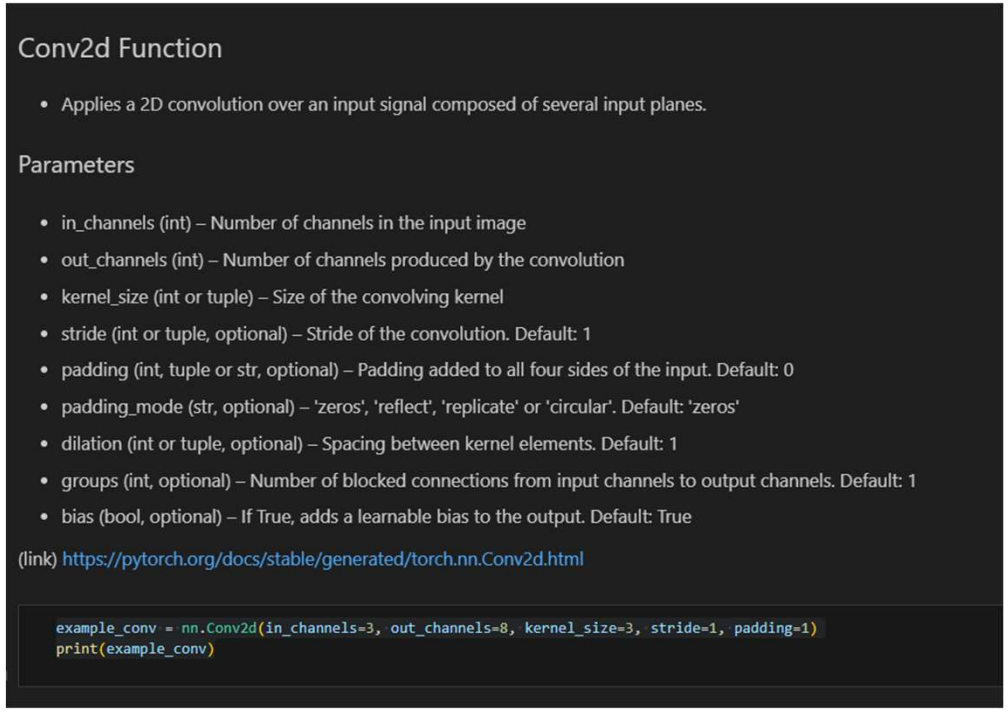


Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ICLR 2021), Attention is All You Need (NeurIPS 2017)

- **Query, Key, and Value Representations:** For each patch embedding, three vectors are derived: the Query (Q), Key (K), and Value (V) vectors.
- **Self-Attention Computation:** For each patch, the self-attention mechanism computes a set of Attention scores, indicating the importance of other patches concerning the current patch.
- **Weighted Sum (Attention Output):** The Attention scores obtained from the Softmax operation are used to compute a weighted sum of the Value vectors (V) of all patches.

Pytorch Function Examples

- Before writing example code, for those who find using PyTorch difficult, please make sure to refer to the following content when writing your code(hint!).



Conv2d Function

- Applies a 2D convolution over an input signal composed of several input planes.

Parameters

- in_channels (int) – Number of channels in the input image
- out_channels (int) – Number of channels produced by the convolution
- kernel_size (int or tuple) – Size of the convolving kernel
- stride (int or tuple, optional) – Stride of the convolution. Default: 1
- padding (int, tuple or str, optional) – Padding added to all four sides of the input. Default: 0
- padding_mode (str, optional) – 'zeros', 'reflect', 'replicate' or 'circular'. Default: 'zeros'
- dilation (int or tuple, optional) – Spacing between kernel elements. Default: 1
- groups (int, optional) – Number of blocked connections from input channels to output channels. Default: 1
- bias (bool, optional) – If True, adds a learnable bias to the output. Default: True

(link) https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

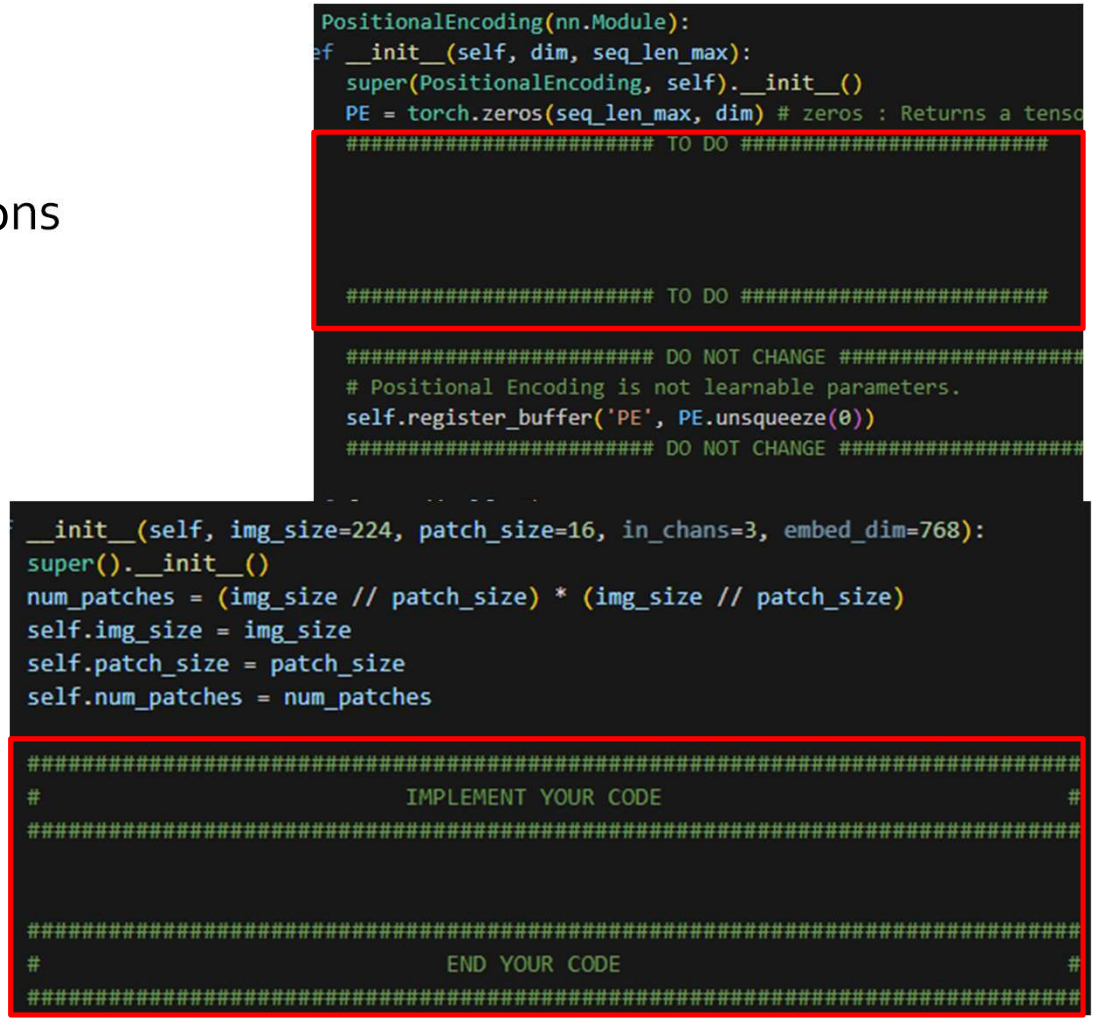
```
example_conv = nn.Conv2d(in_channels=3, out_channels=8, kernel_size=3, stride=1, padding=1)
print(example_conv)
```

How to install assignment files

- Files included: 5 in total
  - Assignment2-1_Transformer_from_scratch.ipynb
  - Assignment2-2_Vit.ipynb
  - CollectSubmission.sh
  - imgs and data folders (data folder is empty)

Code writing space

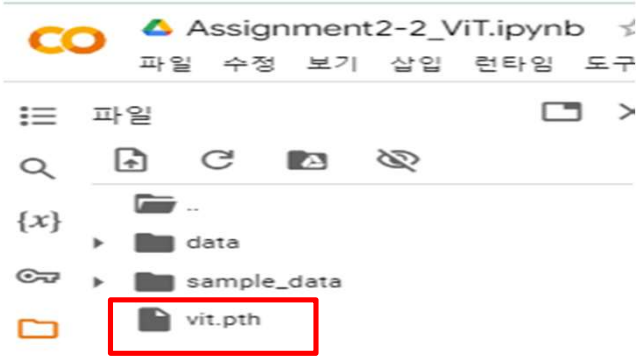
- Please write the code only in the sections marked with 'TO DO' in red boxes and 'IMPLEMENT YOUR CODE'.



```
PositionalEncoding(nn.Module):
f __init__(self, dim, seq_len_max):
  super(PositionalEncoding, self).__init__()
  PE = torch.zeros(seq_len_max, dim) # zeros : Returns a tens
  ######################### TO DO ########################
  ######################### TO DO ########################
  ######################### DO NOT CHANGE ###################
  # Positional Encoding is not learnable parameters.
  self.register_buffer('PE', PE.unsqueeze(0))
  ######################### DO NOT CHANGE ###################
__init__(self, img_size=224, patch_size=16, in_chans=3, embed_dim=768):
super().__init__()
num_patches = (img_size // patch_size) * (img_size // patch_size)
self.img_size = img_size
self.patch_size = patch_size
self.num_patches = num_patches
##############################################################################
#                          IMPLEMENT YOUR CODE                               #
##############################################################################
##############################################################################
#                            END YOUR CODE                                   #
##############################################################################
```

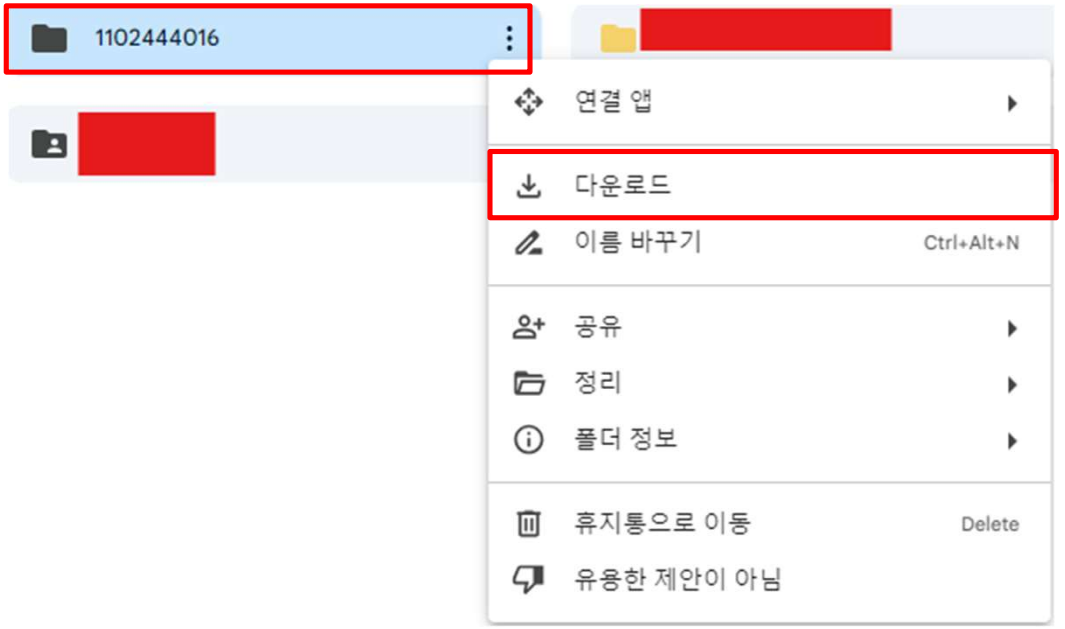
Submitting your work(Colab)

- How to save a .pth file in Colab
  - If the model trains successfully, first go to the respective part and check the .pth file.
  - Secondly, as shown in the image, select the corresponding .pth file and download it.
  - Finally, place the downloaded .pth file into the folder that was previously mentioned.



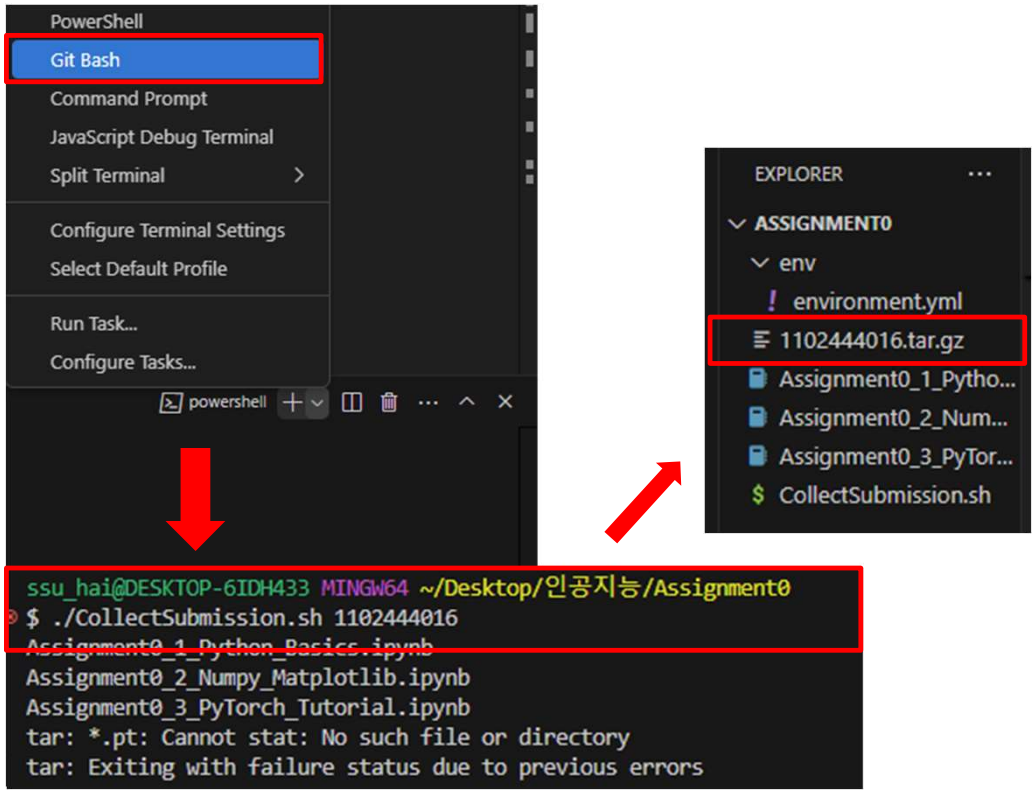
Submitting your work(Colab)

- Go into the Google Drive folder you created earlier
  - Files to be placed inside the folder: 2 ipynb files, 1 pth files
  - Folder name : student-id
  - Please send the downloaded zip file.
  - DO NOT clear the final outputs



Submitting your work(local)

- Submitting your work
  - After you are done
    - Open Git Bash in VSCode or download it locally.
    - $ ./CollectSubmission.sh 1102444016 (student-id)
    - Upload the (student-id).tar.gz on ETL
  - DO NOT clear the final outputs




Thank You!