Assignment 2

Donghyeok Lee

Electrical and Computer Engineering
Seoul National University

http://dsail.snu.ac.kr

Assignment Objectives

- Assignment 2-1-1: Implementing GRU
  - Understand the recurrent neural network (RNN)
  - Implement the forward and backward propagation for a GRU

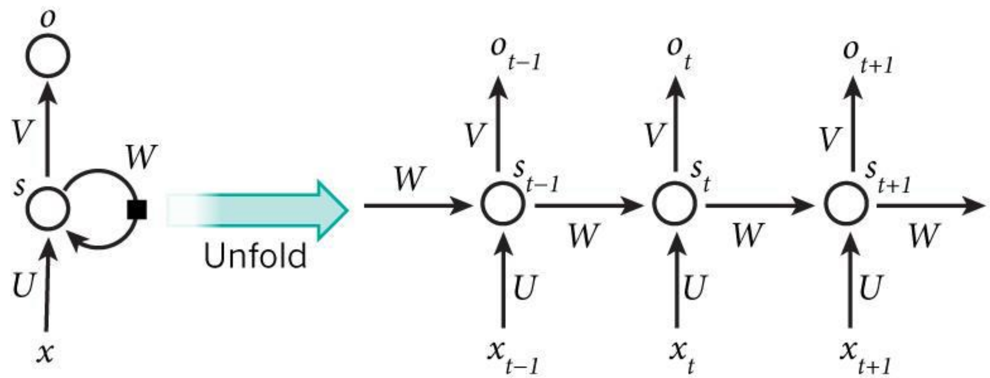
- Assignment 2-1-2: Training Multi-Layer RNN
  - Understand the roles of hyperparameter
  - Train text sentiment analysis by using IMDB dataset

- Assignment 2-2-1 : Understand Vision Transformer
  - Understand Vision transformer
  - Implement Swin-Transformer

- Assignment 2-2-2 : Vision Transformer for image classification
  - Introduction to Huggingface
  - Compare ViT and Swin-ViT
  - Explore hyper-parameters and pick the best

Recurrent Neural Network

- RNN (Recurrent Neural Networks)

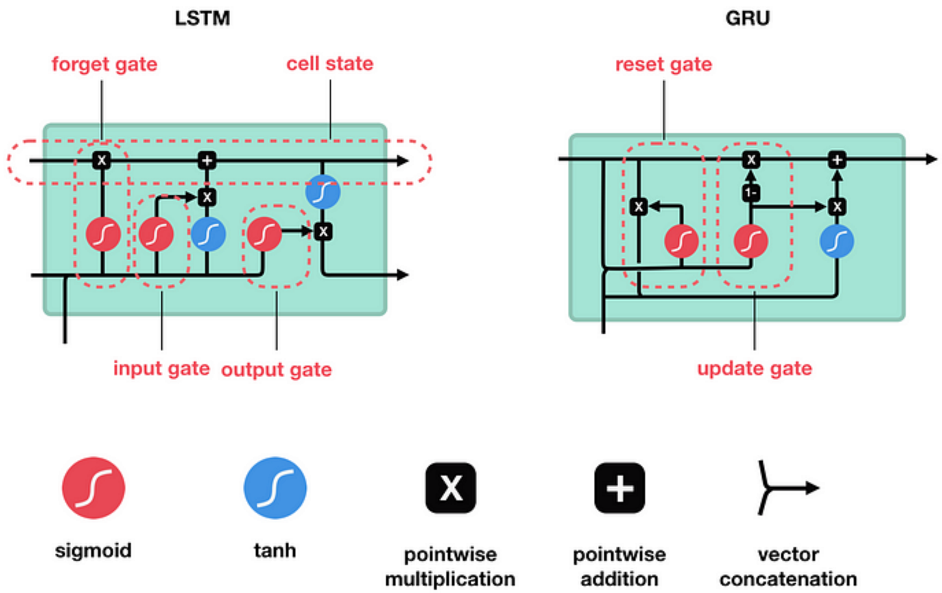


  - RNN perform the same task for every element of a sequence
  - Output depending on the previous computations, “memory”
  - $s_t = f(Ux_t + Ws_{t-1}), o_t = softmax(Vs_t)$
    - ✓ f is nonlinearity function such as tanh
    - ✓ The same parameters (U,V,W)

Source: https://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html

LSTM / GRU

- LSTM (Long short-term memory) and GRU (Gated recurrent unit)

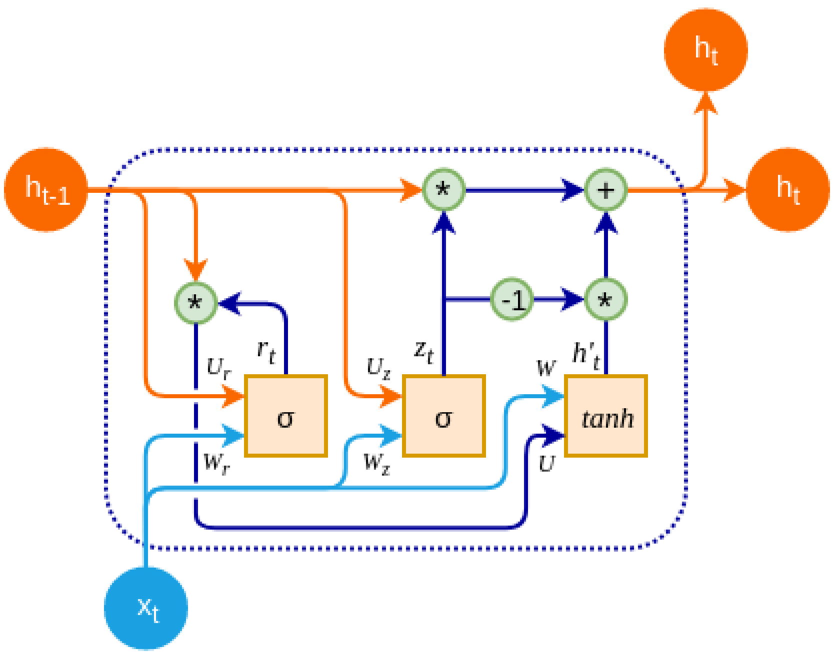


  - LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are advanced variations of the RNN architecture designed to address the problem of long-term dependencies and vanishing gradients.
  - LSTM can remember important information over long periods by maintaining and adjusting the cell state through the gates.
  - GRU simplifies the LSTM by combining the **forget** and **input gates** into a **single gate**, thus reducing complexity.

Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Assignment 2-1-1 Objective

- Assignment: Implement the forward and backward propagation for a GRU(Gated Recurrent Unit)



Key equations of the Gated Recurrent Unit (GRU):

1. Update Gate:

$$z_t = \sigma((U_z \cdot h_{t-1} + b_{zh}) + (W_z \cdot x_t + b_{zw}))$$

2. Reset Gate:

$$r_t = \sigma((U_r \cdot h_{t-1} + b_{rh}) + (W_r \cdot x_t + b_{rw}))$$

3. Current Memory Content:

$$\tilde{h}_t = \tanh((W \cdot x_t + b_w) + r_t * (U \cdot h_{t-1} + b_h))$$

4. Final Memory:

$$h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h}_t$$

Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Assignment 2-1-1 Objective

- Assignment: Implement the forward and backward propagation for a GRU(Gated Recurrent Unit)

```python
import torch


def gru_forward(
    input: torch.Tensor,  # (batch_size, input_size), dtype=torch.double
    hidden: torch.Tensor,  # (batch_size, hidden_size), dtype=torch.double
    weight_ih: torch.Tensor,  # (3 * hidden_size, input_size), dtype=torch.double
    weight_hh: torch.Tensor,  # (3 * hidden_size, hidden_size), dtype=torch.double
    bias_ih: torch.Tensor,  # (3 * hidden_size), dtype=torch.double
    bias_hh: torch.Tensor,  # (3 * hidden_size), dtype=torch.double
):
    ##############################################################################
    #                          IMPLEMENT YOUR CODE                               #
    ##############################################################################

    ##############################################################################
    #                          END OF YOUR CODE                                  #
    ##############################################################################
    output: torch.Tensor  # (batch_size, hidden_size)
    return output


def gru_backward(
    grad_output: torch.Tensor,  # (batch_size, hidden_size), dtype=torch.double
    #
    input: torch.Tensor,  # (batch_size, input_size), dtype=torch.double
    hidden: torch.Tensor,  # (batch_size, hidden_size), dtype=torch.double
    weight_ih: torch.Tensor,  # (3 * hidden_size, input_size), dtype=torch.double
    weight_hh: torch.Tensor,  # (3 * hidden_size, hidden_size), dtype=torch.double
    bias_ih: torch.Tensor,  # (3 * hidden_size), dtype=torch.double
    bias_hh: torch.Tensor,  # (3 * hidden_size), dtype=torch.double
    # IMPORTANT!
    # Thhe order of weight_ih, weight_hh, bias_ih, bias_hh (3 hidden_size, input_size)
    # is reset, update, new (current)"
):
    ##############################################################################
    #                          IMPLEMENT YOUR CODE                               #
    ##############################################################################

    ##############################################################################
    #                          END OF YOUR CODE                                  #
    ##############################################################################
    grad_hidden: torch.Tensor  # (batch_size, hidden_size)
    grad_weight_ih: torch.Tensor  # (3 * hidden_size, input_size)
    grad_weight_hh: torch.Tensor  # (3 * hidden_size, hidden_size)
    grad_bias_ih: torch.Tensor  # (3 * hidden_size)
    grad_bias_hh: torch.Tensor  # (3 * hidden_size)
    return grad_hidden, grad_weight_ih, grad_weight_hh, grad_bias_ih, grad_bias_hh
```

Assignment 2-1-2 Objective

- Assignment: Design and train a GRU model using the PyTorch module (IMDB dataset)
  - Try to achieve the best performance with a multi-layer GRU model

text string · *lengths* 52 13.7k	label class label 2 classes
I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was first released in 1967. I also heard that at first it was seized by U.S. customs if it ever tried to enter this country, therefore being a fan of…	0 neg
"I Am Curious: Yellow" is a risible and pretentious steaming pile. It doesn't matter what one's political views are because this film can hardly be taken seriously on any level. As for the claim that frontal male nudity is an automatic NC-17, that isn't true.…	0 neg
If only to avoid making this type of film in the future. This film is interesting as an experiment but tells no cogent story. One might feel virtuous for sitting thru it because it touches on so many IMPORTANT issues but it does so without any…	0 neg
This film was probably inspired by Godard's Masculin, féminin and I urge you to see that film instead. The film has two strong elements and those are, (1) the realistic acting (2) the impressive, undeservedly good, photo. Apart from that, what strikes…	0 neg
Oh, brother...after hearing about this ridiculous film for umpteen years all I can think of is that old Peggy Lee song.. "Is that all there is??" ...I was just an early teen when this smoked fish hit the U.S. I was too young to get in the theater…	0 neg

- IMDB dataset
  - The IMDB dataset is a large movie review dataset for binary sentiment classification.
  - It contains 50,000 highly polarized reviews from the Internet Movie Database (IMDB), split evenly into 25,000 training and 25,000 test samples.
  - Each review labeled as either positive or negative sentiment.

Hyperparameters

- Hyperparameters
  - Learning rate
  - Mini-batch size
  - Number of training iterations
  - …
- Choosing a set of optimal hyperparameters
  - Is challenging and heuristic

```
OptimizerClass = torch.optim.Adam  # < ----- set this parameter
optimizer_params = {"lr": 1e-3}  # < ----- set this parameter

model_config = ExperimentConfig(
    config_name="gru",  # <---- will be used for saving model
    #
    seed=1,  # < ----- set this parameter
    #
    batch_size=32,  # < ----- set this parameter
    #
    embed_dim=128,  # < ----- set this parameter
    hidden_dim=128,  # < ----- set this parameter
    num_layers=2,  # < ----- set this parameter
    is_bidirectional=False,  # < ----- set this parameter
    dropout=0.2,  # < ----- set this parameter
    optimizer_params=optimizer_params,
    #
    epochs=10,  # < ----- set this parameter
    #
    output_dim=1,
)
```

Experiment with changing the parameters and submit best model

Train at least 5 different models with varying parameter combinations and report the results.
This instruction is asking you to:

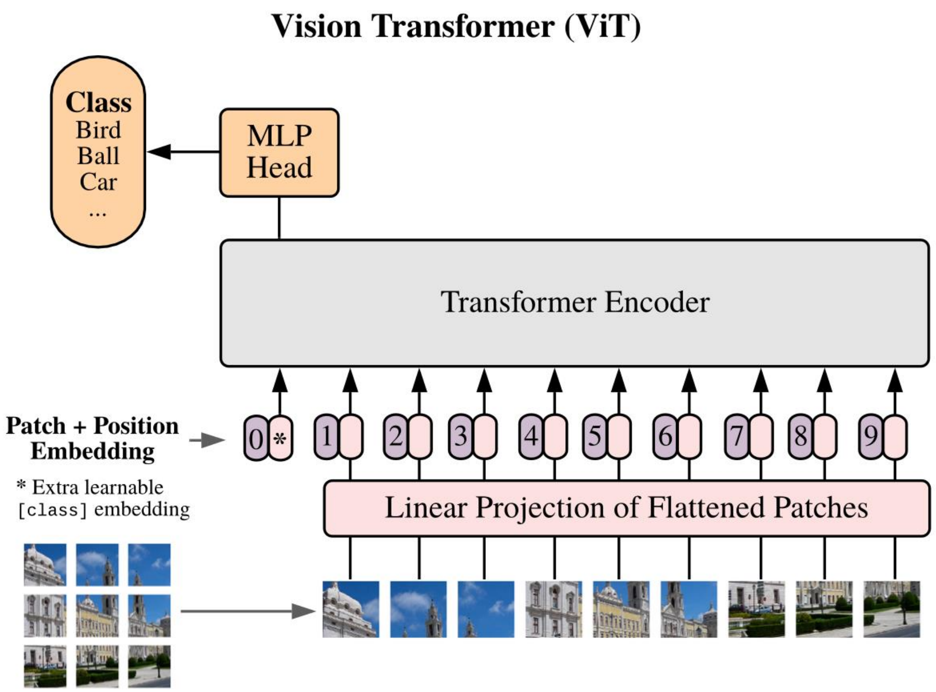
1. Modify the given parameters to create different model configurations.
2. Train at least 5 distinct models, each with a unique combination of these parameters.
3. Run experiments with these different models.
4. Collect and analyze the results from each experiment.
5. Prepare a short report that compares and contrasts the performance of these different model configurations at `model_checkpoints/assignment2-1-2/report.md`
   !!Tip. Write it briefly. Length and content are not part of the grading score.!!
6. **Submit all trained models and configs, including the best-performing one**
   The scores will be assigned in order based on the highest score, and a perfect score will be given for accuracy of 88% or above

Assignment 2-2 Objective

- Assignment 2-2-1 : Understand Vision Transformer
  - Understand Vision transformer
  - Implement Swin-Transformer

- Assignment 2-2-2 : Vision Transformer for image classification
  - Introduction to Huggingface
  - Compare ViT and Swin-ViT
  - Explore hyper-parameters and pick the best

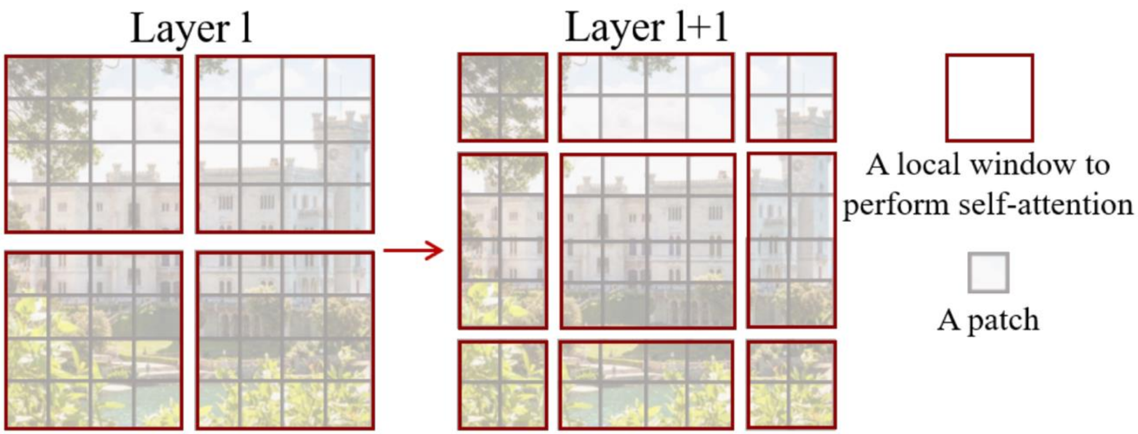
Assignment 2-2-1 Understand Vision Transformer



- Recently, Transformers for images (i.e. ViT) are shown to encode the meaningful information of images, that can be utilized in computer vision tasks.
  (structures are almost same with Transformers for NLP)

Source: https://arxiv.org/pdf/2010.11929

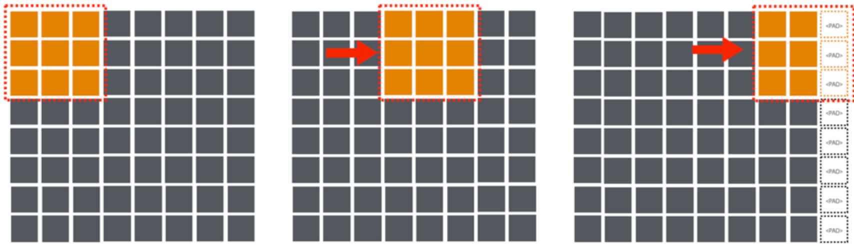
Assignment 2-2-1 Swin Transformer



- Swin Transformer is a variation of ViT designed for better performance in vision tasks by introducing a **hierarchical structure** and **shifting windows**.

Source: https://arxiv.org/pdf/2103.14030

Assignment 2-2-1 Swin Transformer

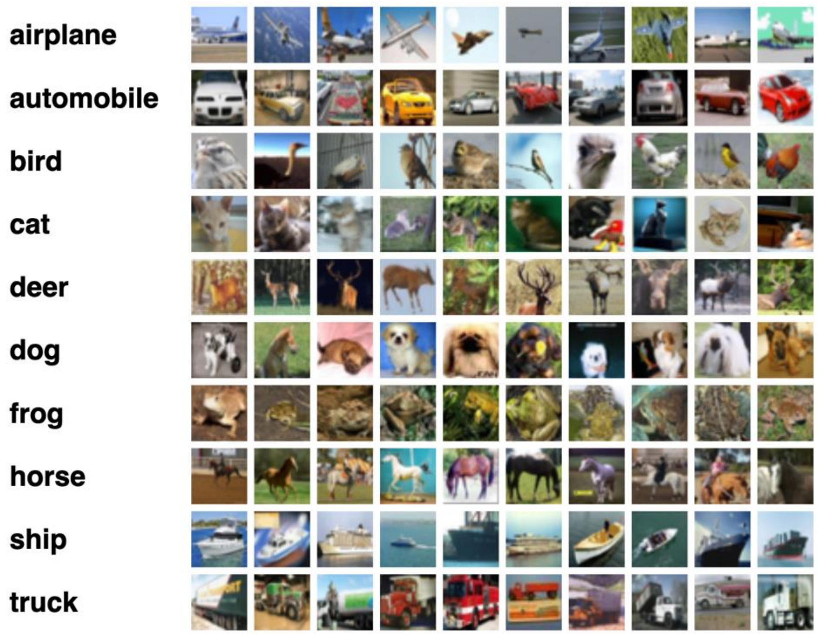


- **Assignment** is to implement the Attention module with the shifted window used in the Swin Transformer.

```python
# use identity mlp when calculating q, k, v
class IdentityMLP(nn.Module):
    def __init__(self, dim: int):
        super().__init__()
        self.mlp = nn.Linear(dim, dim)
        self.initialize_weights_ones()

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return self.mlp(x)

    def initialize_weights_ones(self):
        nn.init.ones_(self.mlp.weight)
        nn.init.ones_(self.mlp.bias)


# use HeaderConcatMLP when merging heads
class HeaderConcatMLP(nn.Module):
    def __init__(self, in_dim: int, out_dim: int):
        super().__init__()
        self.mlp = nn.Linear(in_dim, out_dim)
        self.initialize_weights_ones()

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return self.mlp(x)

    def initialize_weights_ones(self):
        nn.init.ones_(self.mlp.weight)
        nn.init.ones_(self.mlp.bias)


def cal_window_transformer_block(
    x,
    #
    window_size: Tuple[int, int],
    num_heads: int,
):
    B, H, W, C = x.shape
    assert num_heads > 0, "num_heads must be greater than 0"
    assert (
        window_size[0] > 0 and window_size[1] < H and window_size[1] < W
    ), "window_size must be less than image size"

    ################################################################################
    #                              IMPLEMENT YOUR CODE                             #
    ################################################################################

    ################################################################################
    #                              END OF YOUR CODE                                #
    ################################################################################

    output: torch.Tensor = x
    B_, H_, W_, C_ = output.shape
    assert (B_, H_, W_, C_) == (
        B,
        H,
        W,
        C,
    ), "output shape should be same as input shape"
    return output
```

Source: https://arxiv.org/pdf/2103.14030

Assignment 2-2-2 Vision Transformer for image classification



- CIFAR-10 contains 60,000 color images, each sized 32x32 pixels, divided into 10 different classes. These classes include objects like airplanes, cars, birds, cats, and more.
- CIFAR-10 is commonly used for benchmarking computer vision models due to its small image size and manageable dataset size.
- Divided into 50,000 training images and 10,000 test images.

Source: https://www.cs.toronto.edu/~kriz/cifar.html

Assignment 2-2-2 Introduction to Huggingface

- Hugging Face is an open-source platform that provides state-of-the-art machine learning models.
- It offers a variety of pre-trained models and tools, making it easy for developers to fine-tune and deploy models for tasks like text classification, translation, and question-answering.
- The platform also fosters a large community, making cutting-edge research and resources accessible to everyone.

Source: https://huggingface.co/brand

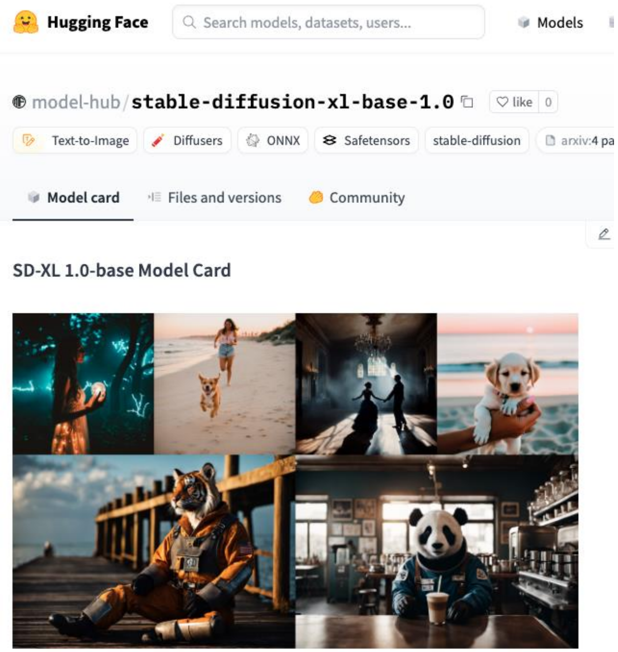
Assignment 2-2-2 Introduction to Huggingface

Example usage:

```
from transformers import SwinModel, SwinConfig

# Creating a Swin Transformer model
configuration = SwinConfig()
model = SwinModel(configuration)

# For a pre-trained model
model = SwinModel.from_pretrained("microsoft/swin-tiny-patch4-window7-224")
```

Example usage:

```
from transformers import ViTModel, ViTConfig

# Creating a ViT model
configuration = ViTConfig()
model = ViTModel(configuration)

# For a pre-trained model
model = ViTModel.from_pretrained("google/vit-base-patch16-224-in21k")
```



- Hugging Face abstracts much of the complexity, allowing models to be modified and built in just a few lines of code.
- Recently, it has become a central hub for the deep learning community, with many teams and researchers uploading models to the Hugging Face Hub for easy access and use.

Source: https://huggingface.co/brand

Assignment 2-2-2 Vision Transformer for image classification

```python
from transformers import ViTConfig
from transformers import ViTForImageClassification

vit_config = ViTConfig(
    image_size=224,
    num_labels=num_classes,
    hidden_size=768,
    num_hidden_layers=12,
    num_attention_heads=12,
    patch_size=16,
    intermediate_size=3072,
    hidden_act="gelu",
    classifier_dropout=0.1,
)

vit_model = ViTForImageClassification(vit_config)
```

```python
from transformers import SwinConfig
from transformers import SwinForImageClassification

swin_config = SwinConfig(
    image_size=image_size[0],
    num_labels=num_classes,
    embed_dim=32,
    depths=[2],
    num_heads=[4],
    window_size=5,
    drop_path_rate=0.1,
)

swin_model = SwinForImageClassification(swin_config)
```

For each model (ViT and Swin-ViT), create and submit the best version in terms of performance.

Specifically:

1. Performance Optimization:
    - Experiment with various hyperparameters and configurations for both ViT and Swin-ViT.
    - Aim to achieve the highest possible accuracy on the given task (e.g., CIFAR-10 classification).
2. Comparative Analysis:
    - Provide a short comparison between your ViT and Swin-ViT models.
    - Discuss the strengths and weaknesses of each in terms of performance and computational efficiency at **model_checkpoints/assignment2-2-2/report.md**
      !!Tip. Write it briefly. Length and content are not part of the grading score.!!
3. Best Model Selection:
    - **Submit the best model from either ViT or Swin-ViT.**
      The scores will be assigned in order based on the highest score, and a perfect score will be given for accuracy of 73% or above

Assignment 2-2-2 Vision Transformer for image classification

```
summary(swin_model, input_size=(1, 3, image_size[0], image_size[1]))
```

✓ 0.0s Python

```
==========================================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================================
SwinForImageClassification                              [1, 10]                   --
├─SwinModel: 1-1                                        [1, 32]                   --
│    └─SwinEmbeddings: 2-1                              [1, 1024, 32]             --
│    │    └─SwinPatchEmbeddings: 3-1                    [1, 1024, 32]             1,568
│    │    └─LayerNorm: 3-2                              [1, 1024, 32]             64
│    │    └─Dropout: 3-3                                [1, 1024, 32]             --
│    └─SwinEncoder: 2-2                                 [1, 1024, 32]             --
│    │    └─ModuleList: 3-4                             --                        26,056
│    └─LayerNorm: 2-3                                   [1, 1024, 32]             64
│    └─AdaptiveAvgPool1d: 2-4                           [1, 32, 1]                --
├─Linear: 1-2                                           [1, 10]                   330
==========================================================================================================
Total params: 28,082
Trainable params: 28,082
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 2.04
==========================================================================================================
Input size (MB): 0.20
Forward/backward pass size (MB): 6.97
Params size (MB): 0.11
Estimated Total Size (MB): 7.27
==========================================================================================================
```

- Additionally, detailed information about the model can be easily accessed using torchinfo.summary.

Source: https://arxiv.org/pdf/2010.11929

How to Install Assignment Files

- Assignment files
  - images/ (image file included for explanation)
  - data/ (empty)
  - model_checkpoints/
  - Assignment2-1_RNN.ipynb
  - Assignment2-2_Transformers.ipynb
  - CollectSubmission.sh
- Install assignment files
  - $ tar zxvf Assignment2.tar.gz (decompress tar gz file)
  - $ chmod 755 CollectSubmission.sh (get permission of script file)
- Open notebooks on your browser and get started!

Important Notes

- Due: 10/30 23:59
- PLEASE read the notes on the notebooks carefully
- Googling first before mailing TAs

- Submitting your work
  - DO NOT clear the final outputs
  - After you are done all three parts
    - ✓ $ ./CollectSubmission.sh 2000-00000(학번)
    - ✓ Upload the 2000-00000.tar.gz on ETL

- TA email: deeplearning.snu@gmail.com

FAQ (Assignment 2-2)

- Q : Batch size를 수정해도 되나요?
- A : 네. 모든 hyperparameter는 수정 가능합니다. 다만, 평가의 공정성을 위해 tokenizer 또는 max length의 수정은 불가능합니다.

FAQ (Assignment 2-1)

- Q : 실제 Swin Transformer 논문과 다르게 구현되는 건가요?
- A : Swin Transformer의 실제 구현은 일반적인 성능을 위해 복잡도가 높습니다. 이번 과제에서는 상대적으로 간단한 구현체를 목표로 출제하였습니다.

- Q : 실행 속도 및 메모리 사용량 등을 포함하여 평가하나요?
- A : 실행 속도 및 메모리 사용량 등의 지표는 평가에 사용되지 않고,
  오로지 결과의 정확도만 평가합니다.

FAQ (Assignment_2-2)

- Q : Huggingface(HF) 를 사용하여 ViT나 Swin-ViT의 config 변경하는 도중 에러가 발생하였습니다.
- A : HF의 config는 정의에 따라 모델을 만들 뿐, 타당성을 검증하지 않습니다.
(예를 들어, 연속된 두 레이어의 output과 input 형식이 동일하지 않는 경우 에러발생.)
따라서 모델을 학습하기 전에 간단한 손계산 등을 통해 config의 타당성을 먼저 확인한 후
학습하는 것을 추천드립니다.

- Q : window size를 작게 하였더니, Colab에서 에러가 발생합니다.
- A : ViT는 CNN에 비해 학습해야 할 파라미터가 매우 많습니다. 특히 window 크기가 작아질수록 필요한 파라미터 수가 지수적으로 증가합니다. 따라서 에러가 발생할 경우 window 크기를 늘리고, 다른 config를 조절하는 것을 추천드립니다.


Thank You!