Anna Bielawska
Indeks: 1187688
In cooperation with GMUM

Individual project

Vision Transformers with Early Exits

The aim of this Python project was to understand the architecture of Vision Transformers (ViTs), experiment with parameter freezing (weights and biases) and finally add Early Exits to ViT's architecture. We used pretrained weights and base model from torchvision:
https://github.com/pytorch/vision/blob/main/torchvision/models/vision_transformer.py

Dataset used for the project purposes – Oxford Flowers102:
https://pytorch.org/vision/stable/generated/torchvision.datasets.Flowers102.html

Note: this dataset has a predefined dataset division – a training dataset contains 1020 pictures of flowers and a testing dataset – 6149 of them. We used Google Colab for visualization.

In the project directory there is an **src** folder which contains three files: **utils.py**, **basic_model.py** and **early_exits.py**. The first file contains functions useful for data preparation, visualization, parameter grid search for fitting best hyperparameters and model training. The second file contains a base level ViT class with freezed pretrained weights and appropriate classification output head. The third file contains custom ViT and Encoder classes with early exit classification heads.

There are two jupyter notebooks: **ViT_Experiments.ipynb** and **ViT_Early_Exits.ipynb**. In the first one we visualize Flowers102 dataset, check the parameter freezing function and output messages. Moreover, we show a short example of conducting a parameter grid search – which model hyperparameters should be chosen to obtain the most accurate results. Then, after choosing the best parameters, we experiment with unfreezing the pretrained weights for one epoch (either at the beginning or the end of training). We compare our results with the ones obtained for a model trained with freezed weights.

```
best_params = find_best_params(param_grid, max_num_sets, criterion, datasets, unfreezed=True,\
                               ViT_path = "BASIC_MODEL.pt")
```

```
Model with parameters: {'lr': 0.0001, 'epochs_num': 4, 'batch_size': 64}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with unfreezed params, last epoch
Model: 0 trained, valid accuracy: 0.4559
Model with parameters: {'lr': 0.0001, 'epochs_num': 3, 'batch_size': 32}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with unfreezed params, last epoch
Model: 1 trained, valid accuracy: 0.5392
Model with parameters: {'lr': 0.001, 'epochs_num': 4, 'batch_size': 64}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with unfreezed params, last epoch
Model: 2 trained, valid accuracy: 0.7696
Model with parameters: {'lr': 0.001, 'epochs_num': 3, 'batch_size': 32}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with unfreezed params, last epoch
Model: 3 trained, valid accuracy: 0.8431
Model with parameters: {'lr': 0.0005, 'epochs_num': 3, 'batch_size': 32}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with unfreezed params, last epoch
Model: 4 trained, valid accuracy: 0.7353
Best params: {'lr': 0.001, 'epochs_num': 3, 'batch_size': 32}, best validation accuracy: 0.8431
Test accuracy: 0.8148
```

From our experiments we concluded that unfreezing the pretrained weights for one epoch to tune them is beneficial to the model accuracy:

```
Model with parameters: {'lr': 0.001, 'epochs_num': 5, 'batch_size': 32}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with freezed params, epoch = 4
Training with unfreezed params, last epoch
Model trained, valid accuracy: 0.8186
Test accuracy: 0.8623

Training with unfreezed params, first epoch
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with freezed params, epoch = 4
Training with freezed params, epoch = 5
Model trained, valid accuracy: 0.8088
Test accuracy: 0.8312

Model with parameters: {'lr': 0.001, 'epochs_num': 5, 'batch_size': 32}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with freezed params, epoch = 4
Training with freezed params, epoch = 5
Model trained, valid accuracy: 0.7549
Test accuracy: 0.7718
```

In the second jupyter notebook we aspired to incorporate Early Exits in the ViT's architecture. We used some of the torchvision classes as a basis and applied changes to them. We added a layer with 12 classification heads (after each encoder layer) and incorporated their outputs in the forward method of the VisionTransformer class. Unfortunately, after trying out a few different approaches, the training of this new layer was unsuccessful.

```
acc, trained_model = train_with_params(params=params, criterion=criterion, datasets=datasets)

Training with freezed params, epoch = 0
Early exit nr 0, last picture - max probability = 0.01278 for class = 43, actual class = 87
Early exit nr 1, last picture - max probability = 0.01612 for class = 36, actual class = 87
Early exit nr 2, last picture - max probability = 0.01374 for class = 17, actual class = 87
Early exit nr 3, last picture - max probability = 0.01620 for class = 65, actual class = 87
Early exit nr 4, last picture - max probability = 0.01582 for class = 96, actual class = 87
Early exit nr 5, last picture - max probability = 0.01636 for class = 22, actual class = 87
Early exit nr 6, last picture - max probability = 0.01440 for class = 93, actual class = 87
Early exit nr 7, last picture - max probability = 0.01605 for class = 28, actual class = 87
Early exit nr 8, last picture - max probability = 0.01396 for class = 82, actual class = 87
Early exit nr 9, last picture - max probability = 0.01304 for class = 21, actual class = 87
Early exit nr 10, last picture - max probability = 0.01310 for class = 61, actual class = 87
Early exit nr 11, last picture - max probability = 0.01449 for class = 39, actual class = 87
Training with freezed params, epoch = 1
Early exit nr 0, last picture - max probability = 0.01272 for class = 43, actual class = 7
Early exit nr 1, last picture - max probability = 0.01483 for class = 36, actual class = 7
Early exit nr 2, last picture - max probability = 0.01374 for class = 39, actual class = 7
Early exit nr 3, last picture - max probability = 0.01582 for class = 65, actual class = 7
Early exit nr 4, last picture - max probability = 0.01631 for class = 96, actual class = 7
Early exit nr 5, last picture - max probability = 0.01612 for class = 22, actual class = 7
Early exit nr 6, last picture - max probability = 0.01402 for class = 19, actual class = 7
Early exit nr 7, last picture - max probability = 0.01513 for class = 1, actual class = 7
Early exit nr 8, last picture - max probability = 0.01329 for class = 65, actual class = 7
Early exit nr 9, last picture - max probability = 0.01346 for class = 47, actual class = 7
Early exit nr 10, last picture - max probability = 0.01452 for class = 90, actual class = 7
Early exit nr 11, last picture - max probability = 0.01334 for class = 57, actual class = 7
Training with freezed params, epoch = 2
Early exit nr 0, last picture - max probability = 0.01279 for class = 43, actual class = 92
Early exit nr 1, last picture - max probability = 0.01650 for class = 36, actual class = 92
Early exit nr 2, last picture - max probability = 0.01392 for class = 39, actual class = 92
Early exit nr 3, last picture - max probability = 0.01635 for class = 65, actual class = 92
Early exit nr 4, last picture - max probability = 0.01612 for class = 96, actual class = 92
Early exit nr 5, last picture - max probability = 0.01688 for class = 22, actual class = 92
```

Note: we chose here to switch the testing and training datasets to see how much the accuracy of our model would change:

```
Accuracy on the validation dataset: 0.92113817
```

Another file, a python script **train.py**, can be used to train a ViT with specific training parameters and to save the trained model weights in a given directory. If the directory doesn't exist, it will be created. The script loads the Flowers102 dataset, allows a user to define the data split fraction (validation and train datasets), checks labels and trains the ViT. A model can also be trained with either freezed or unfreezed weights, as in the experiments.

Help: `python "train.py" -h`

```
parser = argparse.ArgumentParser(add_help=True, description="Python script to run Basic ViT model training")
parser.add_argument('-f', metavar='f', type = float, help='fraction of the training set', default = 0.8)
parser.add_argument('-lr', metavar='lr', type = float, help='learning rate', default = 0.001)
parser.add_argument('-b_s', metavar='b_s', type = int , help='batch size', default = 32)
parser.add_argument('-e_n', metavar='e_n', type = int, help='number of training epochs', default = 1)
parser.add_argument('-u', metavar='u', type = bool , help='unfreeze weights for one epoch', default = False)
parser.add_argument('-a_b', metavar='a_b', type = bool, help='train with unfreezed weights during the first epoch', default = False)
parser.add_argument('-b', metavar='b', type = str , help='path to a file where base weights will be saved', default = "BASIC_MODEL.pt")
parser.add_argument('-t', metavar='t', type = str, help='path to a file where training results should be saved', default = "TRAINED_MODEL.pt")
```

Example of usage:

```
!python "train.py" -lr 0.001 -e_n 4 -b_s 32 -u True
```

```
Sizes of datasets:  816 204 6149
Checking training dataset:
There's at least 1 instance of each class
Checking validation dataset:
At least one class is not represented in this set
Model with parameters: {'lr': 0.001, 'batch_size': 32, 'epochs_num': 4}
Training with freezed params, epoch = 1
Training with freezed params, epoch = 2
Training with freezed params, epoch = 3
Training with unfreezed params, last epoch

Model trained. Weights saved to the directory: TRAINED_MODEL.pt
Accuracy: 0.8235
```