NYCU Introduction to Machine Learning, Final Project

110550108 施柏江

Note:

1. **First, download the folder from 'https://github.com/brianshih95/Introduction-to-Machine-Learning.git', and navigate the current path to 'Introduction-to-Machine-Learning/final_project'.**
2. **The path to the trained model is stored in the variable 'model_path', default = 'training/model.pt'.**
3. **The path to the test data is stored in the variable 'test_folder', default = 'training/data/test'.**
4. **The path to the generated prediction results is '110550108_predictions.csv'.**

Environment details (5%)

Python version:

3.10.2

Framwork:

PyTorch 2.1.0

Hardware:

GPU: NVIDIA Tesla T4

RAM: 15GB

Implementation details (15%)

Model architecture:

I utilized a pre-trained RegNetY-32GF model in PyTorch. It is a specific variant of the RegNet architecture. The RegNet architecture is characterized by using a regular pattern of building blocks and scaling factors to achieve a balance between model performance and computational efficiency.

Hyperparameters:

I use a batch size of 16 due to limited RAM capacity. For the optimizer, I've chosen Adam with a learning rate of 5e-5. I experimented with learning rates ranging from 1e-4 to 1e-5 and found that 5e-5 yields the highest accuracy. Additionally, I've set the weight decay to 1e-5. I've opted for 5 epochs as, beyond this point, there is no significant increase in accuracy. Considering the training time, each epoch takes approximately 30 minutes. This decision strikes a balance between achieving satisfactory accuracy and optimizing the overall training process.

Training strategy:

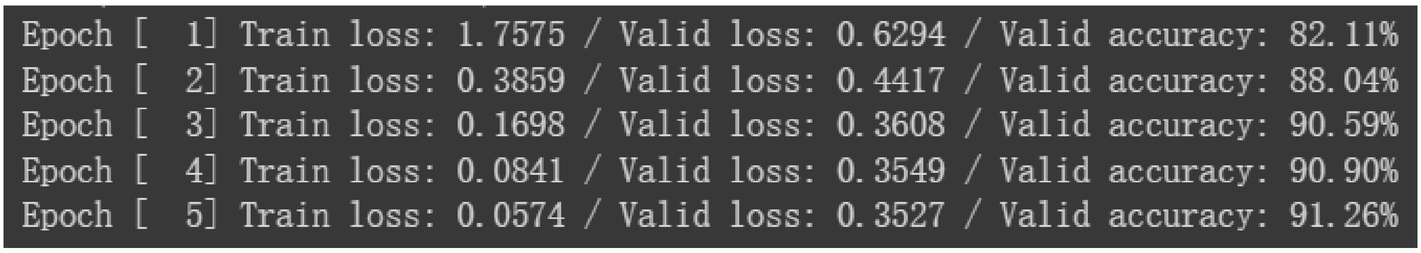
I have applied transfer learning by training a model using the regnet_y_32gf architecture with weights initialized from "IMAGENET1K_SWAG_E2E_V1." I set the img_width and img_height according to the pretrained model I am going to use. The input size for regnet_y_32gf is 384 by 384 by 3. For the training dataset, which constitutes 80% of the original train folder, I employed data augmentations such as random horizontal flip and random rotation to enhance model generalization. The validation dataset comprises 20% of the original train folder.

To adapt the model to the specific classification task, I adjusted the final fully connected layer to match the number of output features with the classes in the dataset. Additionally, I calculated class weights, assigning equal weight to each class, and incorporated them into the training process using CrossEntropyLoss. In terms of optimization, I implemented a learning rate scheduler (lr_scheduler.ExponentialLR) that progressively reduces the learning rate by a factor of 0.5.

Experimental results (15%)

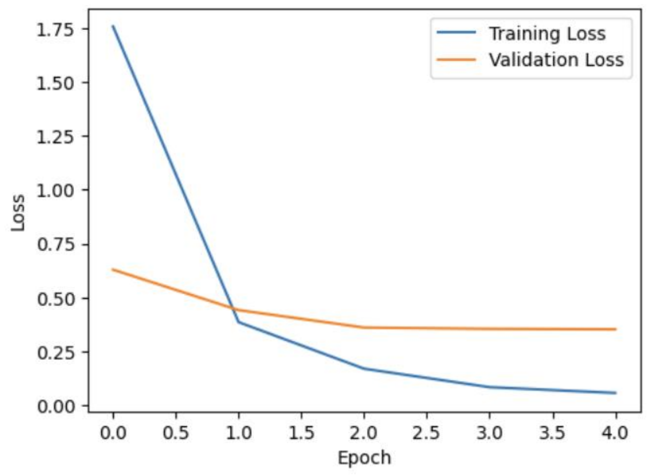
Evaluation metrics:

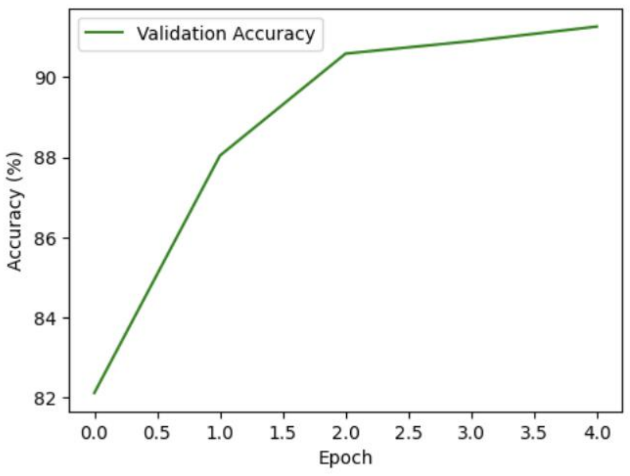
I evaluate the model's performance using accuracy as the primary metric. This metric is well-suited for classification tasks, offering a clear indication of the model's ability to make correct predictions across all classes. I chose accuracy as it aligns with the objective of achieving high overall correctness in the classification task.



```
Epoch [  1] Train loss: 1.7575 / Valid loss: 0.6294 / Valid accuracy: 82.11%
Epoch [  2] Train loss: 0.3859 / Valid loss: 0.4417 / Valid accuracy: 88.04%
Epoch [  3] Train loss: 0.1698 / Valid loss: 0.3608 / Valid accuracy: 90.59%
Epoch [  4] Train loss: 0.0841 / Valid loss: 0.3549 / Valid accuracy: 90.90%
Epoch [  5] Train loss: 0.0574 / Valid loss: 0.3527 / Valid accuracy: 91.26%
```

Learning curve:

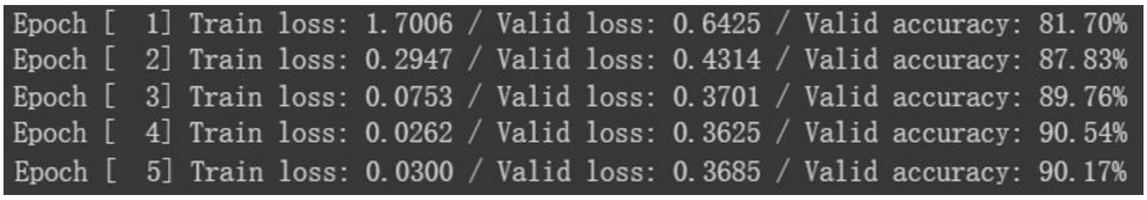




Ablation study:

1. Data Augmentation Ablation

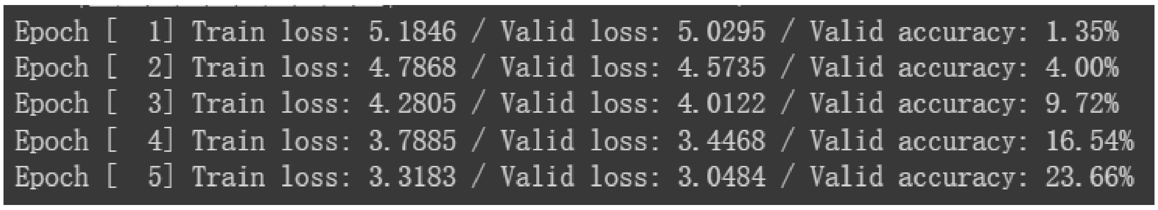
   I train the model without any data augmentation.



```
Epoch [  1] Train loss: 1.7006 / Valid loss: 0.6425 / Valid accuracy: 81.70%
Epoch [  2] Train loss: 0.2947 / Valid loss: 0.4314 / Valid accuracy: 87.83%
Epoch [  3] Train loss: 0.0753 / Valid loss: 0.3701 / Valid accuracy: 89.76%
Epoch [  4] Train loss: 0.0262 / Valid loss: 0.3625 / Valid accuracy: 90.54%
Epoch [  5] Train loss: 0.0300 / Valid loss: 0.3685 / Valid accuracy: 90.17%
```

2. Learning Rate Ablation

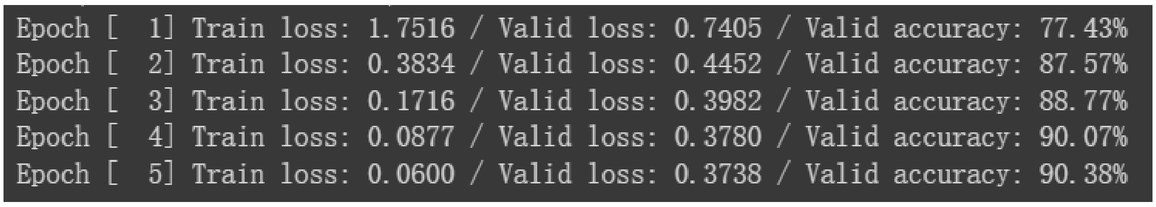
   I use 10 times learning rate during training.



```
Epoch [  1] Train loss: 5.1846 / Valid loss: 5.0295 / Valid accuracy: 1.35%
Epoch [  2] Train loss: 4.7868 / Valid loss: 4.5735 / Valid accuracy: 4.00%
Epoch [  3] Train loss: 4.2805 / Valid loss: 4.0122 / Valid accuracy: 9.72%
Epoch [  4] Train loss: 3.7885 / Valid loss: 3.4468 / Valid accuracy: 16.54%
Epoch [  5] Train loss: 3.3183 / Valid loss: 3.0484 / Valid accuracy: 23.66%
```

3. Normalization Ablation

   I train the model without normalization.



```
Epoch [  1] Train loss: 1.7516 / Valid loss: 0.7405 / Valid accuracy: 77.43%
Epoch [  2] Train loss: 0.3834 / Valid loss: 0.4452 / Valid accuracy: 87.57%
Epoch [  3] Train loss: 0.1716 / Valid loss: 0.3982 / Valid accuracy: 88.77%
Epoch [  4] Train loss: 0.0877 / Valid loss: 0.3780 / Valid accuracy: 90.07%
Epoch [  5] Train loss: 0.0600 / Valid loss: 0.3738 / Valid accuracy: 90.38%
```

Bonus (5%)

Starting Learning Rate	Learning Rate Decay	Data Augmentation	Normalization	Accuracy After 5 Epochs
5e-4	0.5	V	V	23.66%
1e-4	0.7	V	V	88.82%
5e-5	0.7	V	V	89.86%
5e-5	0.5	X	V	90.17%
5e-5	0.5	V	X	90.38%
5e-5	0.5	V	V	**91.26%**