Artificial Intelligence Final Report Assignment 問題 1 (Problem 1)

レポート解答用紙（Report Answer Sheet）

Group Leader

学生証番号（Student ID): 19522238

名前(Name): Nguyễn Lê Thanh

Group Members

学生証番号（Student ID): 19522145

名前(Name): Đinh Thị Diễm Sương

学生証番号（Student ID): 19522310

名前(Name): Phạm Hoàng Thư

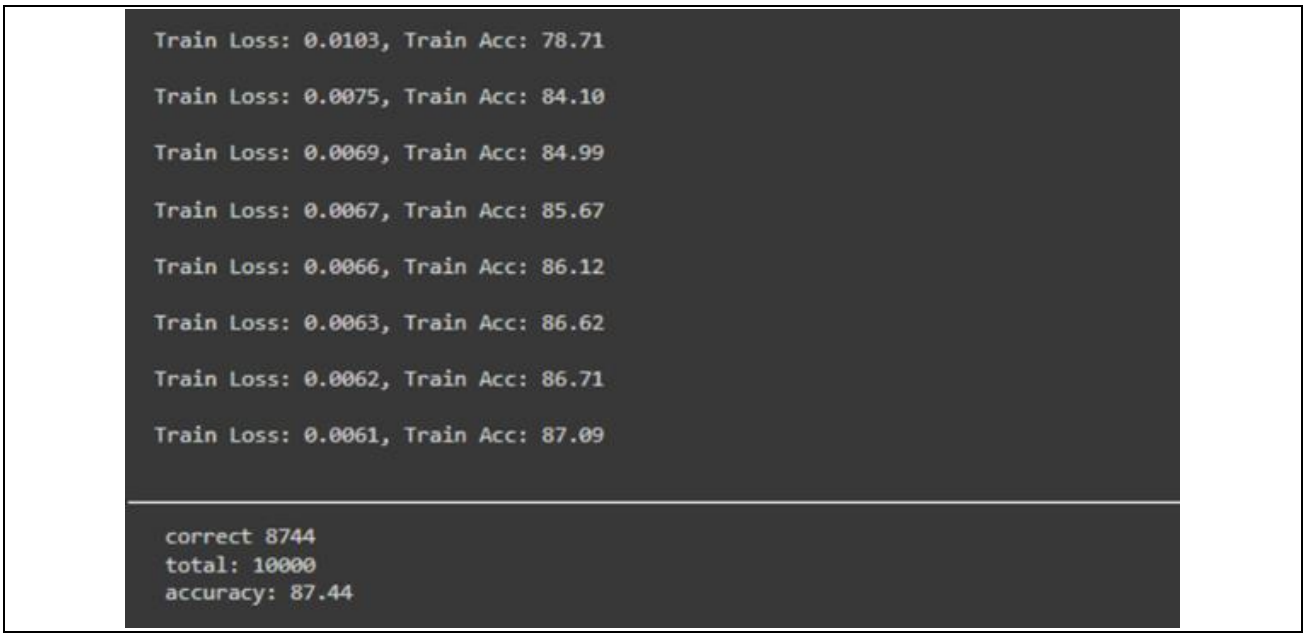
問題 1　(Problem 1)のレポート

Program

★ Link Colab (Problem 1):　Problem1_Labwork4.ipynb

★ Link GitHub: Artificial-Intelligence-IE229.M21.CNCL

Execution Results



Explanation

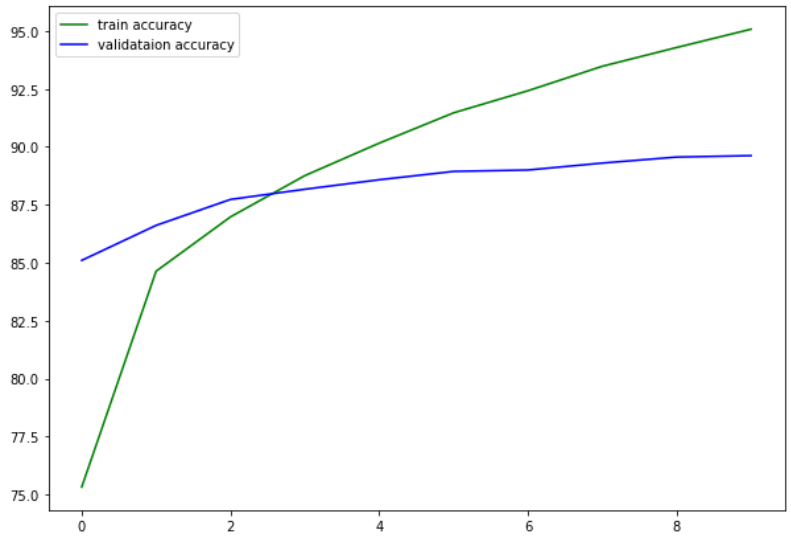
First, we used to pre-train model vgg16.

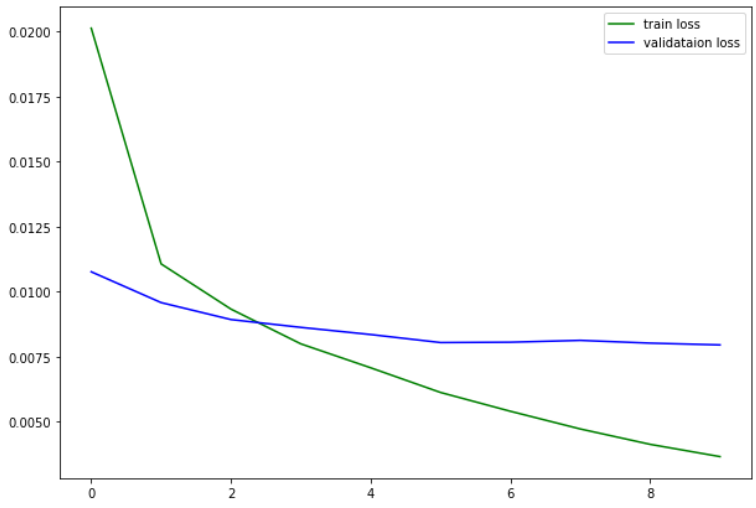
★ On the first training, keep the feature layers and classifier layers the same, learning rate = 0.001, optimizer = SGD and epochs = 10. After plot accuracy and plot loss => the model is heavily overfit (acc train ~ 98% and acc test ~ 87%).

★ On the second training, add 1 Dropout layer with p = 0.3, learning rate = 0.001, optimizer = SGD, epochs = 10 , out_features = 10=> overfit not fixed yet

Classifier class after adding
 	
class dropout
	`Sequential(`
	`  (0): Dropout(p=0.3, inplace=True)`
	`  (1): Linear(in_features=25088, out_features=4096, bias=True)`
	`  (2): ReLU(inplace=True)`
	`  (3): Dropout(p=0.5, inplace=False)`
	`  (4): Linear(in_features=4096, out_features=4096, bias=True)`
	`  (5): ReLU(inplace=True)`
	`  (6): Dropout(p=0.5, inplace=False)`
	`  (7): Linear(in_features=4096, out_features=10, bias=True)`
	`)`

Output	`Train Loss: 0.0201, Train Acc: 75.32`
	`Train Loss: 0.0111, Train Acc: 84.63`
	`Train Loss: 0.0093, Train Acc: 86.98`
	`Train Loss: 0.0080, Train Acc: 88.76`
	`Train Loss: 0.0071, Train Acc: 90.16`
	`Train Loss: 0.0061, Train Acc: 91.47`
	`Train Loss: 0.0054, Train Acc: 92.42`
	`Train Loss: 0.0047, Train Acc: 93.48`
	`Train Loss: 0.0041, Train Acc: 94.29`
	`Train Loss: 0.0037, Train Acc: 95.08`
	
	`correct 8965`
	`total: 10000`
	`accuracy: 89.65`

Plot acc + loss	

★ On the third training, change the classifier layer of vgg16 (move the dropout layer at position [5] -> to position [0]), optimizer = SGD, learning rate = 0.001, weight_decay =5e-4, epochs=10, out_features = 10 = > no overfit, but test acc reduced to about 82%.

Change the classifier layer	`Sequential(`
	`  (0): Dropout(p=0.5, inplace=True)`
	`  (1): Linear(in_features=25088, out_features=4096, bias=True)`
	`  (2): ReLU(inplace=True)`
	`  (3): Dropout(p=0.5, inplace=False)`
	`  (4): Linear(in_features=4096, out_features=4096, bias=True)`
	`  (5): ReLU(inplace=True)`
	`  (6): Linear(in_features=4096, out_features=10, bias=True)`
	`)`

Output	`Train Loss: 0.0132, Train Acc: 71.57`
	`Train Loss: 0.0098, Train Acc: 78.16`
	`Train Loss: 0.0091, Train Acc: 79.44`
	`Train Loss: 0.0086, Train Acc: 80.59`
	`Train Loss: 0.0081, Train Acc: 81.52`
	`Train Loss: 0.0079, Train Acc: 82.11`
	`Train Loss: 0.0075, Train Acc: 83.24`
	`Train Loss: 0.0072, Train Acc: 83.71`
	`Train Loss: 0.0069, Train Acc: 84.38`
	`Train Loss: 0.0066, Train Acc: 85.00`
	
	`correct 8207`
	`total: 10000`
	`accuracy: 82.07`

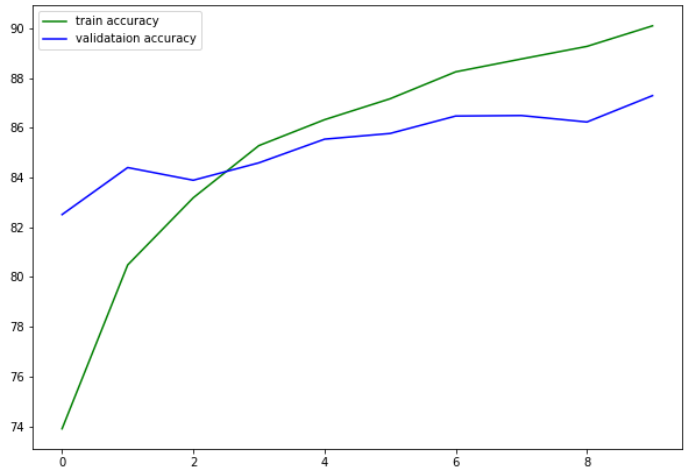
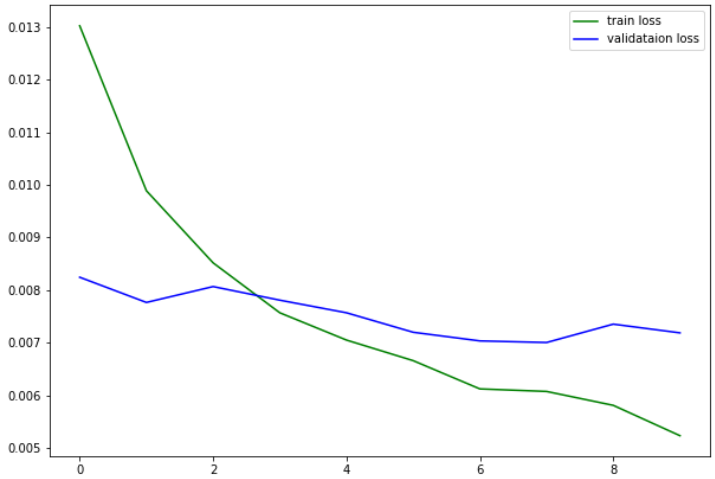
★ On the fourth training, keeping feature classes and classifiers unchanged, epochs = 10, optimizer = Adam, learning rate = 0.0005 and weight_decay=0.001, out_features = 10

Output	Train Loss: 0.0103, Train Acc: 78.96 Train Loss: 0.0073, Train Acc: 84.37 Train Loss: 0.0069, Train Acc: 85.29 Train Loss: 0.0066, Train Acc: 86.07 Train Loss: 0.0064, Train Acc: 86.41 Train Loss: 0.0062, Train Acc: 86.70 Train Loss: 0.0060, Train Acc: 87.12 Train Loss: 0.0059, Train Acc: 87.48 Train Loss: 0.0057, Train Acc: 87.82 Train Loss: 0.0057, Train Acc: 87.87 correct 8657 total: 10000 accuracy: 86.57

★ On fifth training, keep network layers intact + freeze feature layer + optimizer = Adam, learning rate = 0.0005, epochs=15, weight_decay=0.001, out_features = 10

Output	Train Loss: 0.0103, Train Acc: 78.89 Train Loss: 0.0073, Train Acc: 84.32 Train Loss: 0.0068, Train Acc: 85.13 Train Loss: 0.0066, Train Acc: 85.88 Train Loss: 0.0064, Train Acc: 86.35 Train Loss: 0.0061, Train Acc: 87.04 Train Loss: 0.0061, Train Acc: 87.27 Train Loss: 0.0059, Train Acc: 87.42 Train Loss: 0.0058, Train Acc: 87.60 Train Loss: 0.0058, Train Acc: 87.74 correct 8675 total: 10000 accuracy: 86.75

★ On the last training, add 1 dropout layer p = 0.3, optimizer = Adam, learning rate = 0.0005, weight_decay = 0.001 and epochs = 8, out_features = 10

Classifier class after adding *class dropout*	Sequential((0): Dropout(p=0.3, inplace=True) (1): Linear(in_features=25088, out_features=4096, bias=True) (2): ReLU(inplace=True) (3): Dropout(p=0.5, inplace=False) (4): Linear(in_features=4096, out_features=4096, bias=True) (5): ReLU(inplace=True) (6): Dropout(p=0.5, inplace=False) (7): Linear(in_features=4096, out_features=10, bias=True))
Output	Train Loss: 0.0103, Train Acc: 78.71 Train Loss: 0.0075, Train Acc: 84.10 Train Loss: 0.0069, Train Acc: 84.99 Train Loss: 0.0067, Train Acc: 85.67 Train Loss: 0.0066, Train Acc: 86.12 Train Loss: 0.0063, Train Acc: 86.62 Train Loss: 0.0062, Train Acc: 86.71 Train Loss: 0.0061, Train Acc: 87.09 correct 8744 total: 10000 accuracy: 87.44
Plot acc + loss	
	

Overview result table

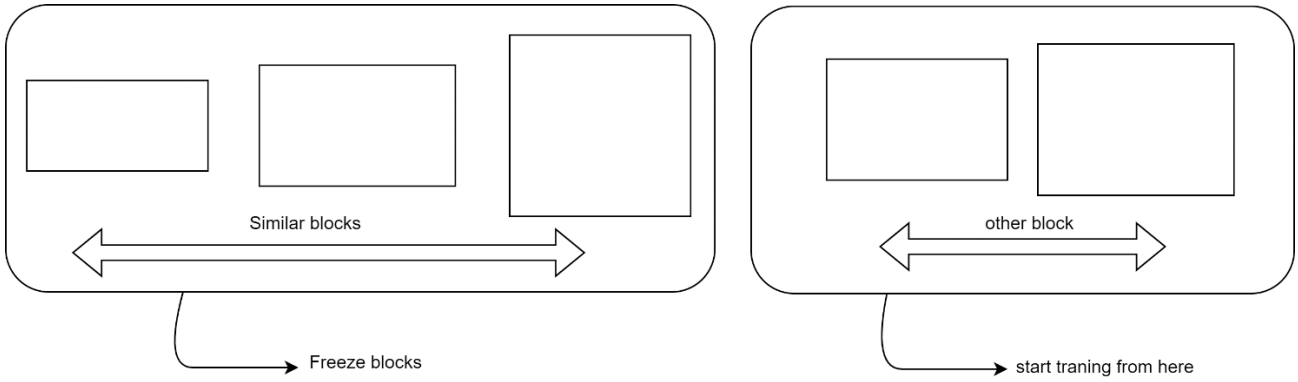
	Features Layer	Classifier layers	Learning rate/ weight_decay	Optimizer	epoch	Output
1	unchanged	unchanged	0.001	SGD	10	- Acc train ~ 98% - Acc test ~ 87%
2	unchanged	add Dropout with p=0.2 out_feature = 10	0.001	SGD	10	- Acc train ~ 95% - Acc test ~ 90%
3	unchanged	move Dropout at position [5] to position [0] out_feature = 10	lr = 0.001 w = 5e-4	SGD	10	- Acc train ~ 85% - Acc test ~ 82%
4	unchanged	out_feature = 10	lr = 0.0005 w = 0.001	Adam	10	-Acc train~87.87% - Acc test~86.57%
5	Freeze convolution weights	out_feature = 10	lr = 0.0005 w = 0.001	Adam	15	-Acc train~87.74% -Acc test~86.75%
6	unchanged	add Dropout with p=0.2 out_feature = 10	lr = 0.0005 w = 0.001	Adam	8	-Acc train~87.09% -Acc test~87.44%

Conclusion

The result of the labwork test is quite small (about accuracy 52,85%) so we used to VGG16, and we try adjusting the hyperparameters 6 times (optimizer, lr, weight_decay, epochs). Finnally, the model achieves the best results (accuracy ~87.44%) with hyperparameters: add 1 Dropout layer on classifier layer with p = 0.2, learning rate = 0.0005, weight decay = 0.001, optimizer = adam, epochs = 8.

Future work

- ★ Use other models such as Resnet50, …
- ★ Try adjusting other hyperparameters like epochs, lr, optimizer, ...
- ★ Change feature layers (after the Conv2d layer, add a batch norm 2d layer and at the end of the feature layer, add a layer AvgPool2d).
- ★ Visualize each block to see how that block describes the image and evaluate whether the output of the block is like your data to be transferred, then freeze the corresponding features and start training the model.



References

1. VGG16 Pre-trained Architecture (beginner) | Kaggle
2. Các phương pháp tránh Overfitting - Regularization, Dropout (viblo.asia) (Last visited: 21/06/2022)
3. Machine Learning cơ bản (machinelearningcoban.com) (Last visited: 21/06/2022)
4. Fine-Tuning Pre-trained Model VGG-16 | by Muriel Kosaka | Towards Data Science (Last visited: 22/06/2022)
5. Hyperparameter Optimization for Transfer Learning of VGG16 for Disease Identification in Corn Leaves Using Bayesian Optimization - PubMed (nih.gov) (Last visited: 29/06/2022)
6. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.