***All the models are not assigned to a fixed seed.

***Graphs are attached in Appendix.

***Original excel file is available in Github and Canvas submission.

The following excel shows the **model from scratch**, with each specification.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	model#	training sample#	validation sample#	test sample#	total epochs#	batch size	training accuracy	validation accuracy	training loss	validation loss	best epochs	test accuracy(PERFORMANCE)	if data augmentation	if regularization		
2	scratch	each class, total sample= #*2													model build from scratch	pretrain model
3	part1	1000	500	500	150	32	~0.97	~0.86	~0.084	~0.4647	~25	77.20%	yes(flip, rotation and zoom)	yes(dropout) 0.4	yes	
4		1000	500	500	150	32	~0.94(epoch 120)	~0.829(epoch 100)	~0.194(epoch 144)	0.5305(epoch 19)	~19	77.20%	not apply		yes	
5	part2	4000	500	500	150	32	~0.8553 (epoch 22)	~0.8950 (epoch 30)	~0.3510(epoch 30)	~0.2510 (epoch 38)	~38	78.90%	yes(flip, rotation and zoom)	yes(dropout) 0.5	yes	
6	part3	1500	500	500	100	32	~0.9243 (epoch 71)	~0.8580 (epoch 93)	~0.2138 (epoch 96)	~0.3966 (epoch 51)	~51	78.60%	yes(flip, rotation and zoom)	yes(dropout) 0.4	yes	
7		4000	2000	1000	40	32	~0.8533 epoch 29)	~0.8835(epoch 25)	~0.356(epoch 25)	~0.3081 (epoch 25)	~25	78.80%	yes(flip, rotation and zoom)	yes(dropout) 0.4	yes	
8		4000	2000	1000	40	128, increase batch size will extremely increase the training time, before for one epoch around 30s now, it around 90s	~0.9079(epoch 39)	~0.8707(epoch 31)	0.2432(epoch 34)	~0.2759 (epoch 39)	~39	78.80%	yes(flip, rotation and zoom)	yes(dropout) 0.4	Yes	
9	add more layers	1000	500	500	40	32	~0.8070(epoch 39)	~0.7950(epoch 39)	~0.4639(epoch 39)	~0.5188 (epoch39)	~39	77.20%	yes(flip, rotation and zoom)	yes(dropout) 0.4	yes	

Row4 shows the model from scratch for the most naïve one, and the test accuracy is 77.2%. Then I tried to **add data augmentation, layers, and changing batch size and sample size for other models**.

Initially, I expected a significant improvement in the performance (10%+), but the results turned out not to be this much. My best model is the one in row 5, which is based on a training/validation/test size as 4000 500 500 and with data augmentation and drop out. The performance reached 78.90%.

Conclusion and takeaways for scratch model:

Data augmentation, regularization, enrichment of layers and large sample size help with increase the test performance but the tuning process takes a lot of effort and computer needs more time to compute compared to pretrain model. Even though I tuned 7 models, I still cannot reach the performance as 83.5% as the test book. Data augmentation will mitigate overfitting and should be included in all models. Besides, I found batch size does matter a lot in terms of calculation time and performance; **Sometimes the training/validation graph behaves abnormally as we discussed last time, and this issue could be mitigated by tuning batch size and choose training/validation/test size properly. Sometimes, the validation accuracy will be above the training accuracy, which I also check the first edition of the textbook, and this happened in their writing as well (page 114), but not all the time. One last thing is that we don't have to train too many epochs(over 100).**

The following excel shows the **model from pretrain**, with each specification.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	model#	training sample#	validation sample#	test sample#	total epochs#	batch size	training accuracy	validation accuracy	training loss	validation loss	best epochs	test accuracy(PERFORMANCE)	if data augmentation	if regularization			
11	pretrain	each class, total sample= #*2													model build from scratch	pretrain model	
12	model1	1000	500	500	20		1 (epoch 20)	0.978(epoch 20)	0	4.8347(epoch 5)	5	77.20%	not apply	yes(dropout)0.5		yes	
13	model2	1000	500	500	50		~0.9935(epoch 50)	~0.984(epoch 3)	~2.873 （epoch50）	~2.8149 （epoch30）	30	97.30%	yes(flip, rotation and zoom)	yes(dropout)0.5		yes	
14	model3(fine-tuning) **overfit comes more later**	1000	500	500	30		~0.9970 （epoch 24）	~0.9800(epoch 30)	~0.0532(epoch 24)	~1.76539(epoch 16)	16	97.50%	yes(flip, rotation and zoom)	yes(dropout)		yes	
15	model4(fit-tuning)	3000	750	750	30		~0.9995(epoch 30)	~0.984(epoch 14)	~0.0057	~1.7508 (epoch 21)	21	97.5%	yes(flip, rotation and zoom)	yes(dropout)		yes	
16	model5(fit-tuning)	5000	1000	1000	30		~1 （epoch 26）	~0.9850(epoch 4)	0(epoch 26)	~1.5412 (epoch 29)		97.80%	yes(flip, rotation and zoom)	yes(dropout)		yes	
17																	

The naïve model for this part is model1, which is a pretrain model with VGG but without data augmentation. This again indicates that data augmentation is very important when doing image recognition. The naïve model, even pretrained but without data augmentation, its performance is bad (accuracy 77.20%, no difference from model from scratch).

Once data augmentation is deployed in the pretrain model, the accuracy improves significantly from around 80% to around 97%. The reason for this is that this model structure is very sophisticated in terms of complexity, target-oriented (for image). We could also mimic this structure by building it from scratch, but it will take some time to reach the perform as the pretrain model. My models also turn out that fit-tuning model should be mainly used, because the VGG model performs good in image, then we don't have to tune around with our modification too much. It is trustful to do fit tunning.

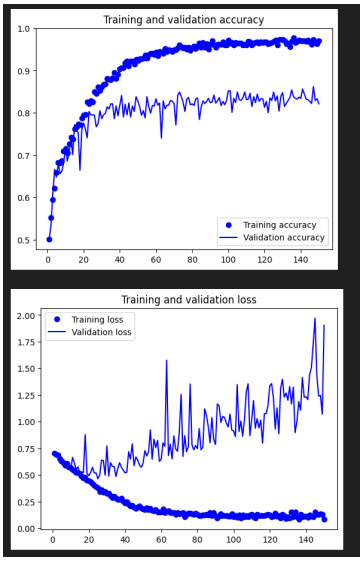
Conclusion and takeaways for pretrain model:

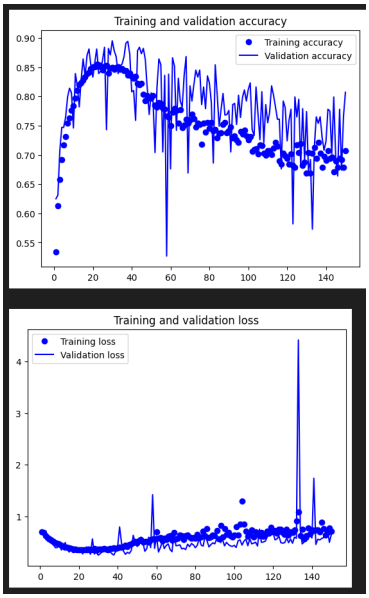
Aligned with the model from scratch that I talked aboded, data augmentation, regularization, rich sample all help with building up better pretrain model and mitigate overfitting issue.

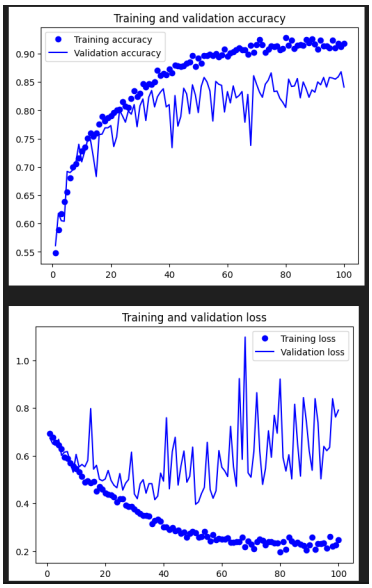
One takeaway from pretrain model, at least from my model results, is that we should prefer fit-tuning model among pretrain models, because it give me the highest accuracy as 97.80% with proper sample size (5000,1000,1000). And it saves us some effort as well.

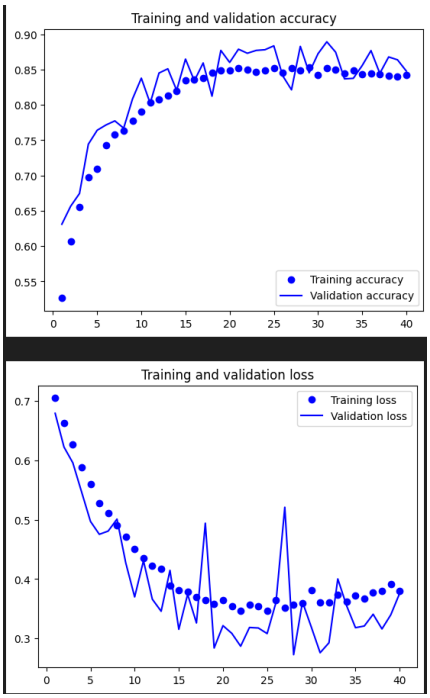
Appendix

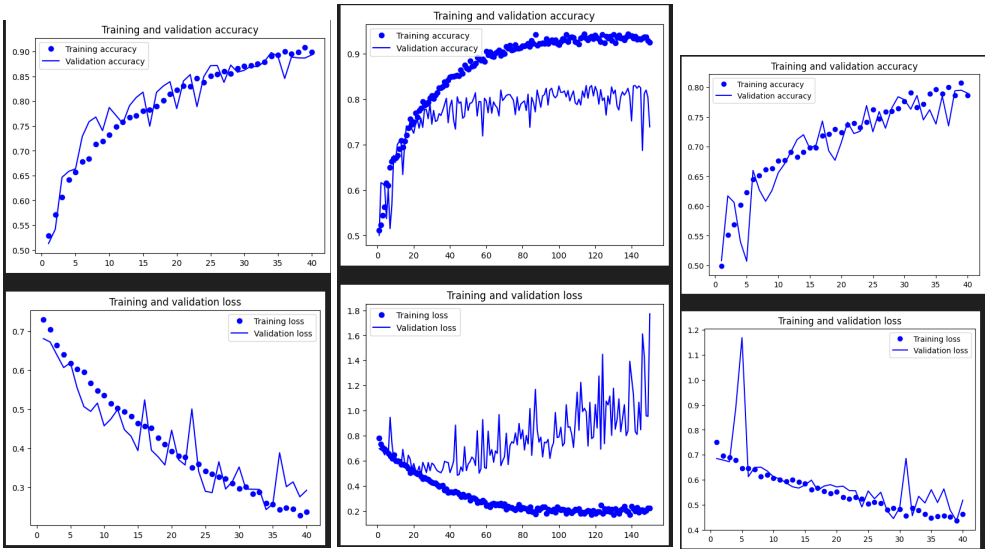
*** graphs is shown as the sequence of excel models.

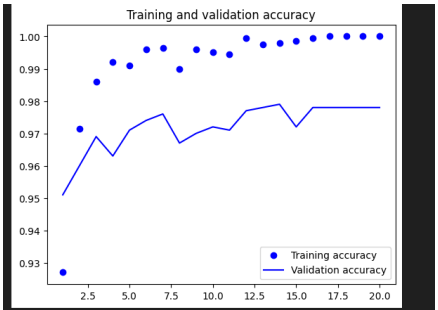


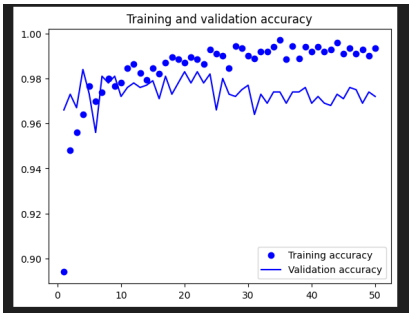


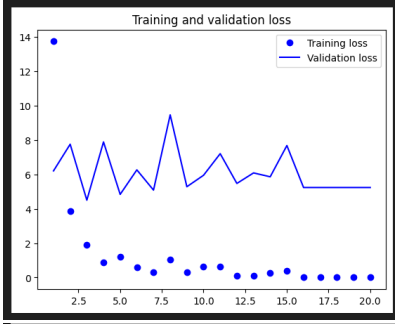

Training and validation accuracy
Training accuracy
Validation accuracy
Training and validation loss
Training loss
Validation loss

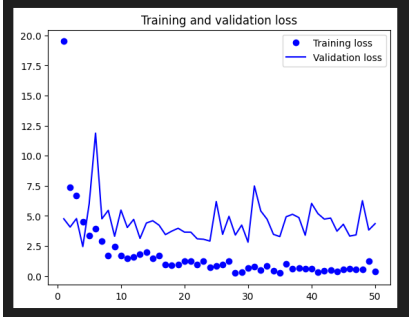

Training and validation accuracy
Training accuracy
Validation accuracy
Training and validation loss
Training loss
Validation loss

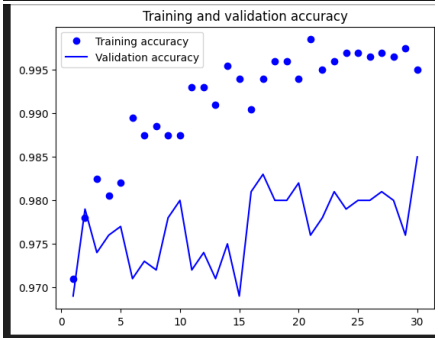

Training and validation accuracy
Training accuracy
Validation accuracy
Training and validation loss
Training loss
Validation loss
Training and validation accuracy
Training accuracy
Validation accuracy
Training and validation loss
Training loss
Validation loss
Training and validation accuracy
Training accuracy
Validation accuracy
Training and validation loss
Training loss
Validation loss

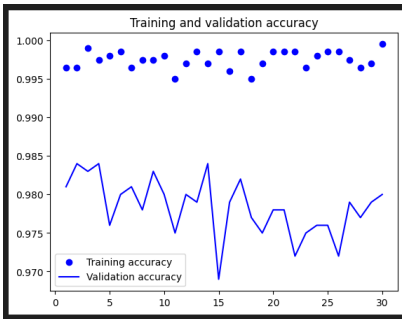

Training and validation accuracy
Training accuracy
Validation accuracy

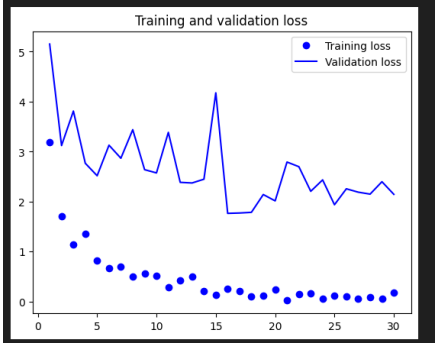

Training and validation accuracy
Training accuracy
Validation accuracy

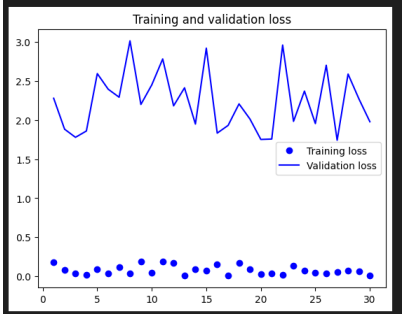

Training and validation loss
Training loss
Validation loss

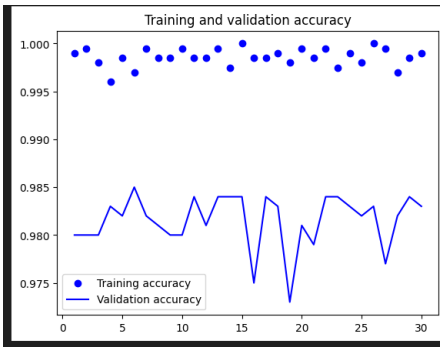

Training and validation loss
Training loss
Validation loss

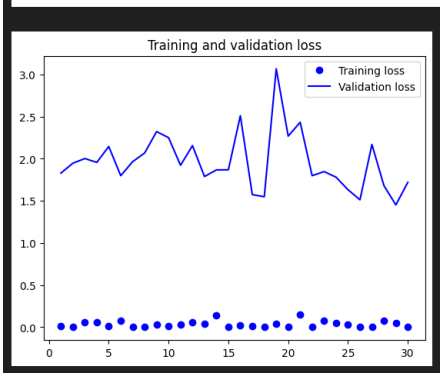

Training and validation accuracy
Training accuracy
Validation accuracy


Training and validation accuracy
Training accuracy
Validation accuracy


Training and validation loss
Training loss
Validation loss


Training and validation loss
Training loss
Validation loss


Training and validation accuracy
1.000
0.995
0.990
0.985
0.980
0.975
Training accuracy
Validation accuracy
0
5
10
15
20
25
30


Training and validation loss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Training loss
Validation loss
0
5
10
15
20
25
30