

Machine Learning Homework 3 Report

R06221012 數學所 李岳洲

April 31, 2020

1. 請說明你實作的 *CNN* 模型，其模型架構、訓練參數量和準確率為何？

optimizer: Adam, learning rate: 0.0001, batchsize: 512, epoch: 300, early stopping patient: 30

	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy
CNN	0.971923	0.000707	0.829697	0.001128	0.85475

Table 1: CNN results

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 128, 128]	1,792
BatchNorm2d-2	[-1, 64, 128, 128]	128
ReLU-3	[-1, 64, 128, 128]	0
Conv2d-4	[-1, 64, 128, 128]	36,928
BatchNorm2d-5	[-1, 64, 128, 128]	128
ReLU-6	[-1, 64, 128, 128]	0
MaxPool2d-7	[-1, 64, 64, 64]	0
Conv2d-8	[-1, 128, 64, 64]	73,856
BatchNorm2d-9	[-1, 128, 64, 64]	256
ReLU-10	[-1, 128, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	147,584
BatchNorm2d-12	[-1, 128, 64, 64]	256
ReLU-13	[-1, 128, 64, 64]	0
MaxPool2d-14	[-1, 128, 32, 32]	0
Conv2d-15	[-1, 256, 32, 32]	295,168
BatchNorm2d-16	[-1, 256, 32, 32]	512
ReLU-17	[-1, 256, 32, 32]	0
Conv2d-18	[-1, 256, 32, 32]	590,080
BatchNorm2d-19	[-1, 256, 32, 32]	512
ReLU-20	[-1, 256, 32, 32]	0
Conv2d-21	[-1, 256, 32, 32]	590,080
BatchNorm2d-22	[-1, 256, 32, 32]	512
ReLU-23	[-1, 256, 32, 32]	0
MaxPool2d-24	[-1, 256, 16, 16]	0
Conv2d-25	[-1, 512, 16, 16]	1,180,160
BatchNorm2d-26	[-1, 512, 16, 16]	1,024
ReLU-27	[-1, 512, 16, 16]	0
Conv2d-28	[-1, 512, 16, 16]	2,359,808
BatchNorm2d-29	[-1, 512, 16, 16]	1,024
ReLU-30	[-1, 512, 16, 16]	0
Conv2d-31	[-1, 512, 16, 16]	2,359,808
BatchNorm2d-32	[-1, 512, 16, 16]	1,024
ReLU-33	[-1, 512, 16, 16]	0
MaxPool2d-34	[-1, 512, 8, 8]	0
Conv2d-35	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-36	[-1, 512, 8, 8]	1,024
ReLU-37	[-1, 512, 8, 8]	0
Conv2d-38	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-39	[-1, 512, 8, 8]	1,024
ReLU-40	[-1, 512, 8, 8]	0
Conv2d-41	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-42	[-1, 512, 8, 8]	1,024
ReLU-43	[-1, 512, 8, 8]	0
MaxPool2d-44	[-1, 512, 4, 4]	0
Linear-45	[-1, 4096]	33,558,528
Dropout-46	[-1, 4096]	0
ReLU-47	[-1, 4096]	0
Linear-48	[-1, 4096]	16,781,312
Dropout-49	[-1, 4096]	0
ReLU-50	[-1, 4096]	0
Linear-51	[-1, 1000]	4,097,000
Dropout-52	[-1, 1000]	0
ReLU-53	[-1, 1000]	0
Linear-54	[-1, 11]	11,011
Total params: 69,170,987		
Trainable params: 69,170,987		
Non-trainable params: 0		
Input size (MB): 0.19		
Forward/backward pass size (MB): 105.27		

Figure 1: CNN model architecture

2. 請實作與第一題接近的參數量，但 *CNN* 深度 (*CNN* 層數) 減半的模型，並說明其模型架構、訓練參數量和準確率為何？

optimizer: Adam, learning rate: 0.0001, batchsize: 512, epoch: 300, early stopping patient: 30

	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy
CNN (half layers)	0.758362	0.001420	0.662767	0.002215	—

Table 2: CNN (half layers) results

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 128, 128]	448
BatchNorm2d-2	[-1, 16, 128, 128]	32
ReLU-3	[-1, 16, 128, 128]	0
Conv2d-4	[-1, 16, 128, 128]	2,320
BatchNorm2d-5	[-1, 16, 128, 128]	32
ReLU-6	[-1, 16, 128, 128]	0
MaxPool2d-7	[-1, 16, 64, 64]	0
Conv2d-8	[-1, 32, 64, 64]	4,640
BatchNorm2d-9	[-1, 32, 64, 64]	64
ReLU-10	[-1, 32, 64, 64]	0
Conv2d-11	[-1, 32, 64, 64]	9,248
BatchNorm2d-12	[-1, 32, 64, 64]	64
ReLU-13	[-1, 32, 64, 64]	0
MaxPool2d-14	[-1, 32, 32, 32]	0
Conv2d-15	[-1, 64, 32, 32]	18,496
BatchNorm2d-16	[-1, 64, 32, 32]	128
ReLU-17	[-1, 64, 32, 32]	0
Conv2d-18	[-1, 64, 32, 32]	36,928
BatchNorm2d-19	[-1, 64, 32, 32]	128
ReLU-20	[-1, 64, 32, 32]	0
MaxPool2d-21	[-1, 64, 16, 16]	0
Linear-22	[-1, 4096]	67,112,960
Dropout-23	[-1, 4096]	0
ReLU-24	[-1, 4096]	0
Linear-25	[-1, 4096]	16,781,312
Dropout-26	[-1, 4096]	0
ReLU-27	[-1, 4096]	0
Linear-28	[-1, 1000]	4,097,000
Dropout-29	[-1, 1000]	0
ReLU-30	[-1, 1000]	0
Linear-31	[-1, 11]	11,011
Total params: 88,074,811		
Trainable params: 88,074,811		
Non-trainable params: 0		
Input size (MB): 0.19		
Forward/backward pass size (MB): 22.09		
Params size (MB): 335.98		
Estimated Total Size (MB): 358.25		

Figure 2: CNN model architecture (half layers)

3. 請實作與第一題接近的參數量，簡單的 *DNN* 模型，同時也說明其模型架構、訓練參數和準確率為何？

optimizer: Adam, learning rate: 0.0001, batchsize: 256, epoch: 300, early stopping patient: 30

	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy
DNN	0.417784	0.006590	0.390837	0.007002	—

Table 3: DNN results

Layer (type)	Output Shape	Param #
Linear-1	[-1, 3, 128, 256]	33,024
Dropout-2	[-1, 3, 128, 256]	0
ReLU-3	[-1, 3, 128, 256]	0
Linear-4	[-1, 3, 128, 512]	131,584
Dropout-5	[-1, 3, 128, 512]	0
ReLU-6	[-1, 3, 128, 512]	0
Linear-7	[-1, 3, 128, 1024]	525,312
Dropout-8	[-1, 3, 128, 1024]	0
ReLU-9	[-1, 3, 128, 1024]	0
Linear-10	[-1, 3, 128, 2048]	2,099,200
Dropout-11	[-1, 3, 128, 2048]	0
ReLU-12	[-1, 3, 128, 2048]	0
Linear-13	[-1, 3, 128, 4096]	8,392,704
Dropout-14	[-1, 3, 128, 4096]	0
ReLU-15	[-1, 3, 128, 4096]	0
Linear-16	[-1, 3, 128, 6120]	25,073,640
Dropout-17	[-1, 3, 128, 6120]	0
ReLU-18	[-1, 3, 128, 6120]	0
Linear-19	[-1, 3, 128, 3072]	18,803,712
Dropout-20	[-1, 3, 128, 3072]	0
ReLU-21	[-1, 3, 128, 3072]	0
Linear-22	[-1, 3, 128, 2048]	6,293,504
Dropout-23	[-1, 3, 128, 2048]	0
ReLU-24	[-1, 3, 128, 2048]	0
Linear-25	[-1, 3, 128, 1024]	2,098,176
Dropout-26	[-1, 3, 128, 1024]	0
ReLU-27	[-1, 3, 128, 1024]	0
Linear-28	[-1, 3, 128, 512]	524,800
Dropout-29	[-1, 3, 128, 512]	0
ReLU-30	[-1, 3, 128, 512]	0
Linear-31	[-1, 3, 128, 256]	131,328
Dropout-32	[-1, 3, 128, 256]	0
ReLU-33	[-1, 3, 128, 256]	0
Linear-34	[-1, 3, 128, 128]	32,896
Dropout-35	[-1, 3, 128, 128]	0
ReLU-36	[-1, 3, 128, 128]	0
Linear-37	[-1, 3, 128, 64]	8,256
Dropout-38	[-1, 3, 128, 64]	0
ReLU-39	[-1, 3, 128, 64]	0
Linear-40	[-1, 3, 128, 32]	2,080
Dropout-41	[-1, 3, 128, 32]	0
ReLU-42	[-1, 3, 128, 32]	0
Linear-43	[-1, 3, 128, 16]	528
Dropout-44	[-1, 3, 128, 16]	0
ReLU-45	[-1, 3, 128, 16]	0
Linear-46	[-1, 3, 128, 8]	136
Dropout-47	[-1, 3, 128, 8]	0
ReLU-48	[-1, 3, 128, 8]	0
Flatten-49	[-1, 3072]	0
Linear-50	[-1, 384]	1,180,032
Dropout-51	[-1, 384]	0
ReLU-52	[-1, 384]	0
Linear-53	[-1, 48]	18,480
Dropout-54	[-1, 48]	0
ReLU-55	[-1, 48]	0
Linear-56	[-1, 11]	539
Total params: 65,349,931		
Trainable params: 65,349,931		
Non-trainable params: 0		

Figure 3: DNN model architecture

4. 請說明由 1 ~ 3 題的實驗中你觀察到了什麼？

We use data normalization, data augmentation and early stopping in the above 1 ~ 3. All of them are stopped before 300 epochs.

5. 請嘗試 *data normalization* 及 *data augmentation*，說明實作方法並且說明實行前後對準確率有什麼樣的影響？

(a) Data normalization:

Calculate the batch mean and standard deviation of training set, then apply this mean and standard deviation to validation set and testing set.

Usage:

torchvision.transforms.Normalize(mean, std)

(b) Data augmentation:

Input size: $3 \times 128 \times 128$

(i) We resized to the size $3 \times 150 \times 150$, then randomly crop center by the size $3 \times 128 \times 128$.

(ii) Randomly horizontal flip.

(iii) Randomly rotation by degree 15.

(iv) Double the original dataset by augmentation.

Usage:

(i) *torchvision.transforms.Resize(150)*,

torchvision.transforms.RandomResizedCrop(128)

(ii) *torchvision.transforms.RandomHorizontalFlip()*

(iii) *transforms.RandomRotation(15)*

(c) Comparison table:

	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy
original cnn	0.995231	0.000439	0.761623	—	0.78720
augmentation	0.928874	0.000812	0.776205	—	0.79438
augmentation + normalization	0.971923	0.000707	0.829697	0.001128	0.85475

Table 4: Results comparison table

6. 觀察答錯的圖片中，哪些 *class* 彼此間容易用混？

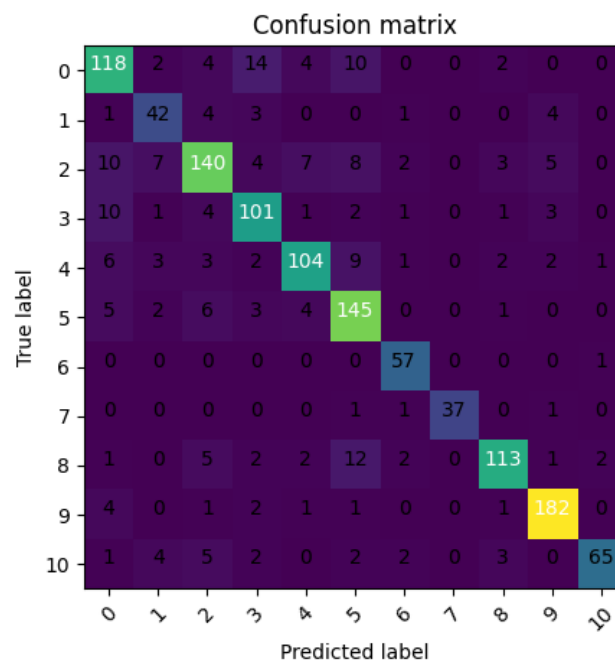
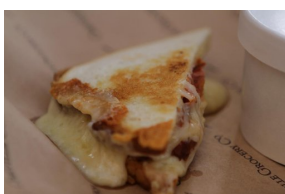


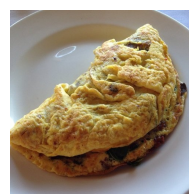
Figure 4: Confusion matrix



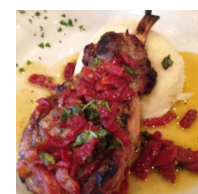
(a) Class 0



(b) Class 2



(c) Class 3



(d) Class 5

Figure 5: Confused classes