CS535 HW3

933271081
Chih-Hsiang Wang

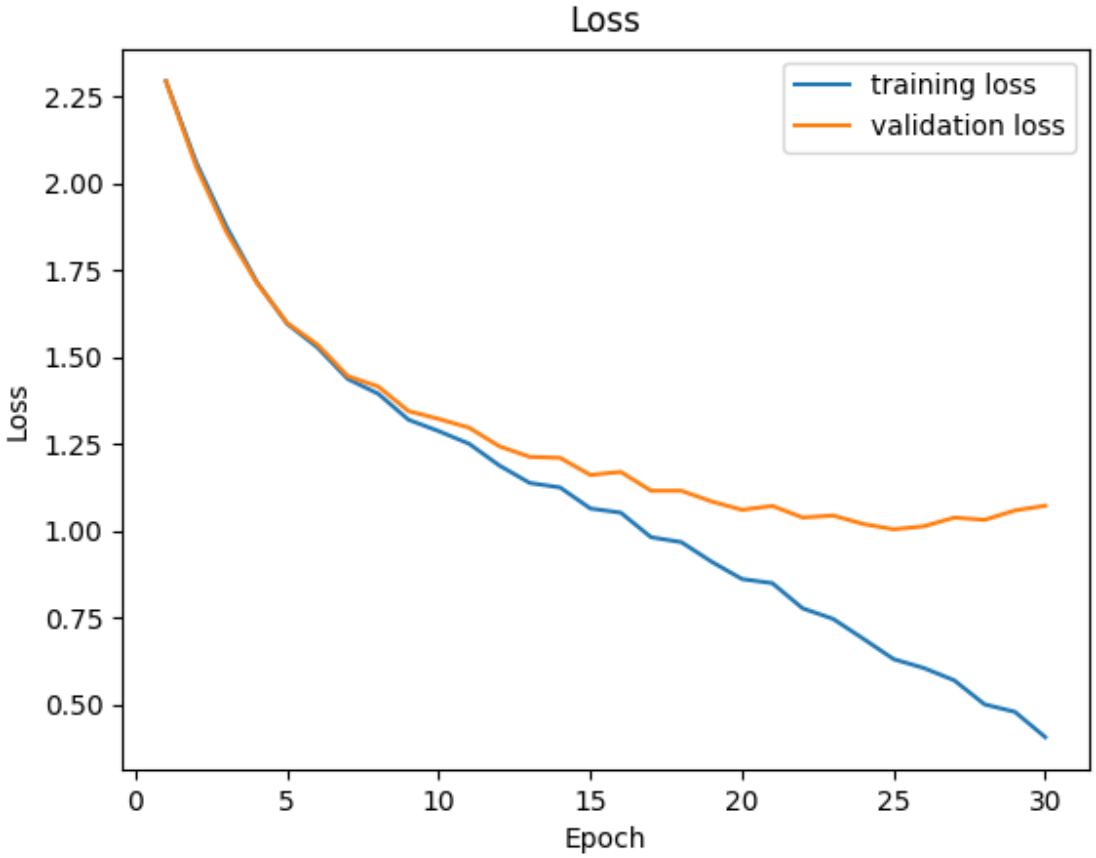
1) Add a batch normalization layer after the first fully-connected layer(fc1) (10 points). Save the model after training(Checkout our tutorial on how to save your model). Becareful that batch normalization layer performs differently between training and evalation process, make sure you understand how to convert your model between training mode and evaluation mode(you can find hints in my code). Observe the difference of final training/testing accuracy with/without batch normalization layer.

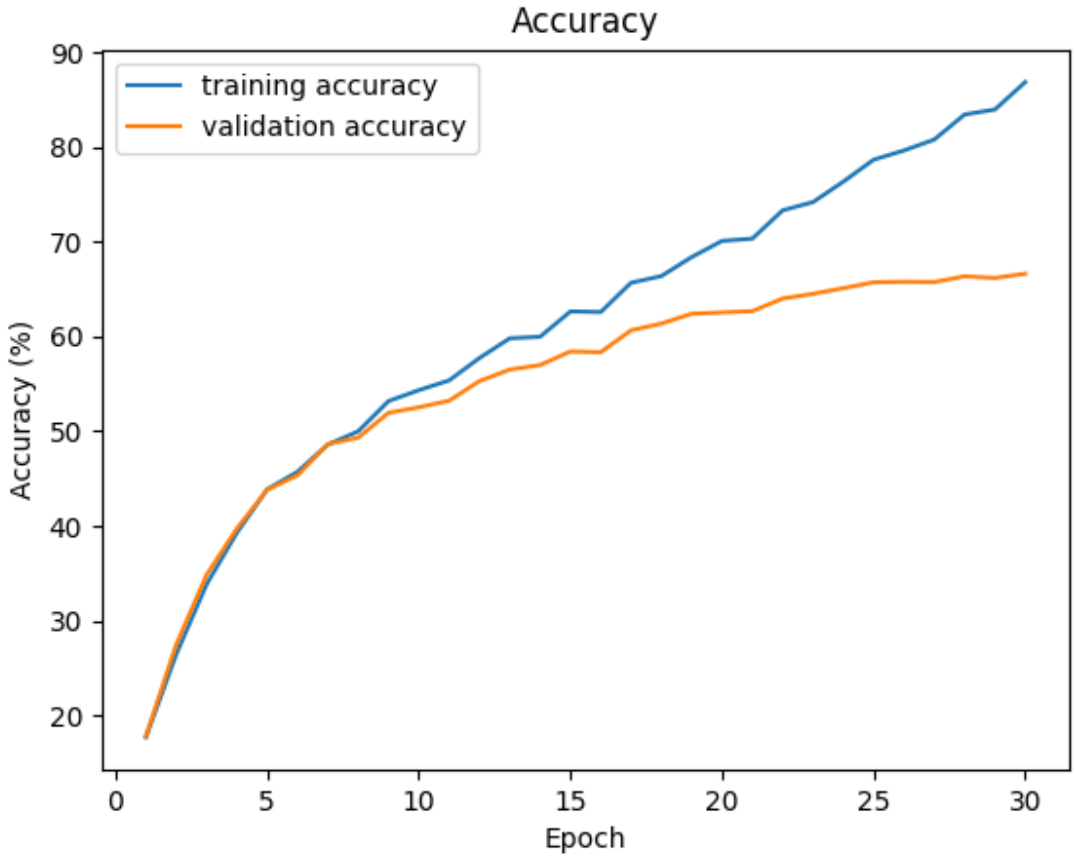
optimizer = optim.SGD(net.parameters(), lr=0.0005, momentum=0.8)

without batch normalization layer:

EPOCH: 1 train_loss: 2.29421 train_acc: 0.17684 test_loss: 2.29404 test_acc 0.17820
EPOCH: 2 train_loss: 2.05996 train_acc: 0.26456 test_loss: 2.04789 test_acc 0.27480
EPOCH: 3 train_loss: 1.87450 train_acc: 0.33844 test_loss: 1.85878 test_acc 0.34840
EPOCH: 4 train_loss: 1.71571 train_acc: 0.39284 test_loss: 1.71317 test_acc 0.39730
EPOCH: 5 train_loss: 1.59544 train_acc: 0.43870 test_loss: 1.59769 test_acc 0.43830
EPOCH: 6 train_loss: 1.52727 train_acc: 0.45722 test_loss: 1.53558 test_acc 0.45350
EPOCH: 7 train_loss: 1.43653 train_acc: 0.48592 test_loss: 1.44477 test_acc 0.48650
EPOCH: 8 train_loss: 1.39418 train_acc: 0.49994 test_loss: 1.41490 test_acc 0.49320
EPOCH: 9 train_loss: 1.31933 train_acc: 0.53186 test_loss: 1.34417 test_acc 0.51950
EPOCH: 10 train_loss: 1.28633 train_acc: 0.54340 test_loss: 1.32159 test_acc 0.52530
EPOCH: 11 train_loss: 1.25007 train_acc: 0.55370 test_loss: 1.29639 test_acc 0.53230
EPOCH: 12 train_loss: 1.18733 train_acc: 0.57734 test_loss: 1.24278 test_acc 0.55300
EPOCH: 13 train_loss: 1.13718 train_acc: 0.59814 test_loss: 1.21241 test_acc 0.56510
EPOCH: 14 train_loss: 1.12428 train_acc: 0.59978 test_loss: 1.20953 test_acc 0.56990
EPOCH: 15 train_loss: 1.06367 train_acc: 0.62654 test_loss: 1.16024 test_acc 0.58430
EPOCH: 16 train_loss: 1.05170 train_acc: 0.62590 test_loss: 1.16914 test_acc 0.58350
EPOCH: 17 train_loss: 0.98094 train_acc: 0.65682 test_loss: 1.11477 test_acc 0.60660
EPOCH: 18 train_loss: 0.96676 train_acc: 0.66388 test_loss: 1.11472 test_acc 0.61370
EPOCH: 19 train_loss: 0.91018 train_acc: 0.68394 test_loss: 1.08385 test_acc 0.62410
EPOCH: 20 train_loss: 0.86053 train_acc: 0.70100 test_loss: 1.05985 test_acc 0.62550
EPOCH: 21 train_loss: 0.84867 train_acc: 0.70342 test_loss: 1.07105 test_acc 0.62680
EPOCH: 22 train_loss: 0.77613 train_acc: 0.73328 test_loss: 1.03765 test_acc 0.64020
EPOCH: 23 train_loss: 0.74594 train_acc: 0.74198 test_loss: 1.04345 test_acc 0.64510
EPOCH: 24 train_loss: 0.68859 train_acc: 0.76354 test_loss: 1.01909 test_acc 0.65110
EPOCH: 25 train_loss: 0.62989 train_acc: 0.78678 test_loss: 1.00382 test_acc 0.65730
EPOCH: 26 train_loss: 0.60413 train_acc: 0.79650 test_loss: 1.01247 test_acc 0.65790
EPOCH: 27 train_loss: 0.56875 train_acc: 0.80802 test_loss: 1.03746 test_acc 0.65760
EPOCH: 28 train_loss: 0.49956 train_acc: 0.83448 test_loss: 1.03100 test_acc 0.66360
EPOCH: 29 train_loss: 0.47802 train_acc: 0.83972 test_loss: 1.05799 test_acc 0.66190
EPOCH: 30 train_loss: 0.40525 train_acc: 0.86874 test_loss: 1.07175 test_acc 0.66640

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
90
80
70
60
50
40
30
20
0
5
10
15
20
25
30

CS535 HW3

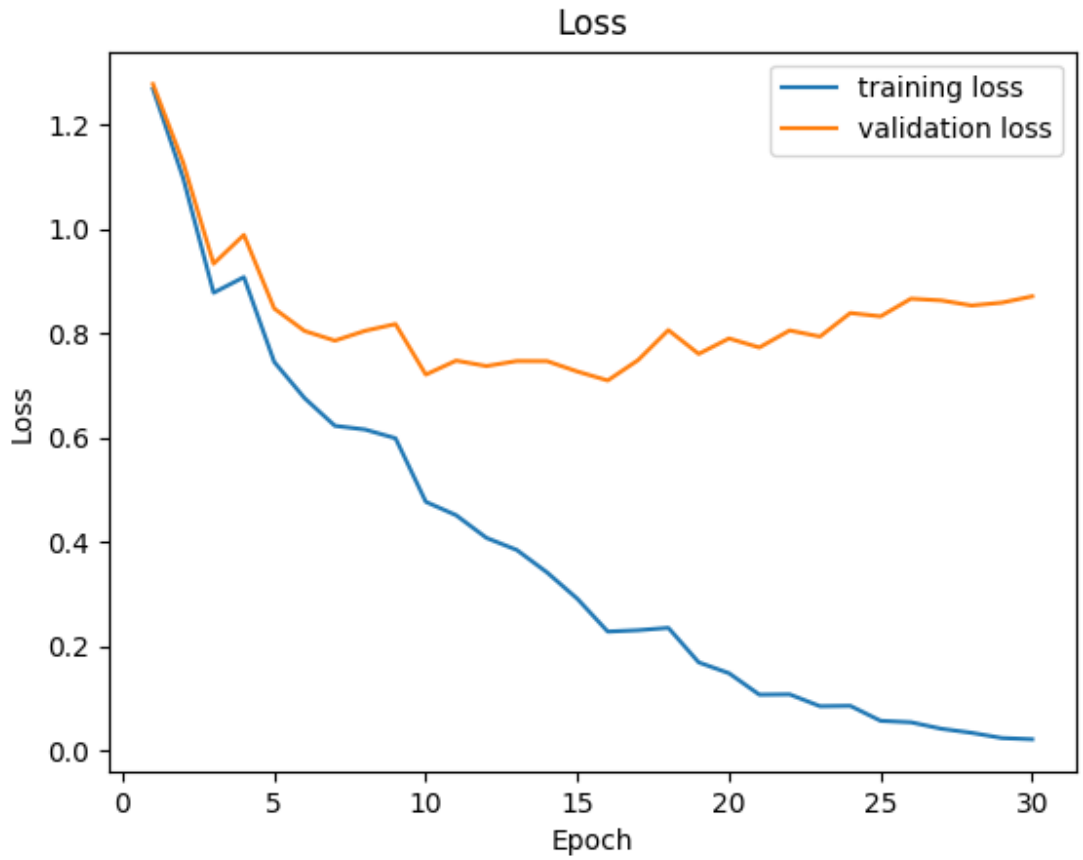
optimizer = optim.SGD(net.parameters(), lr=0.0005, momentum=0.8)

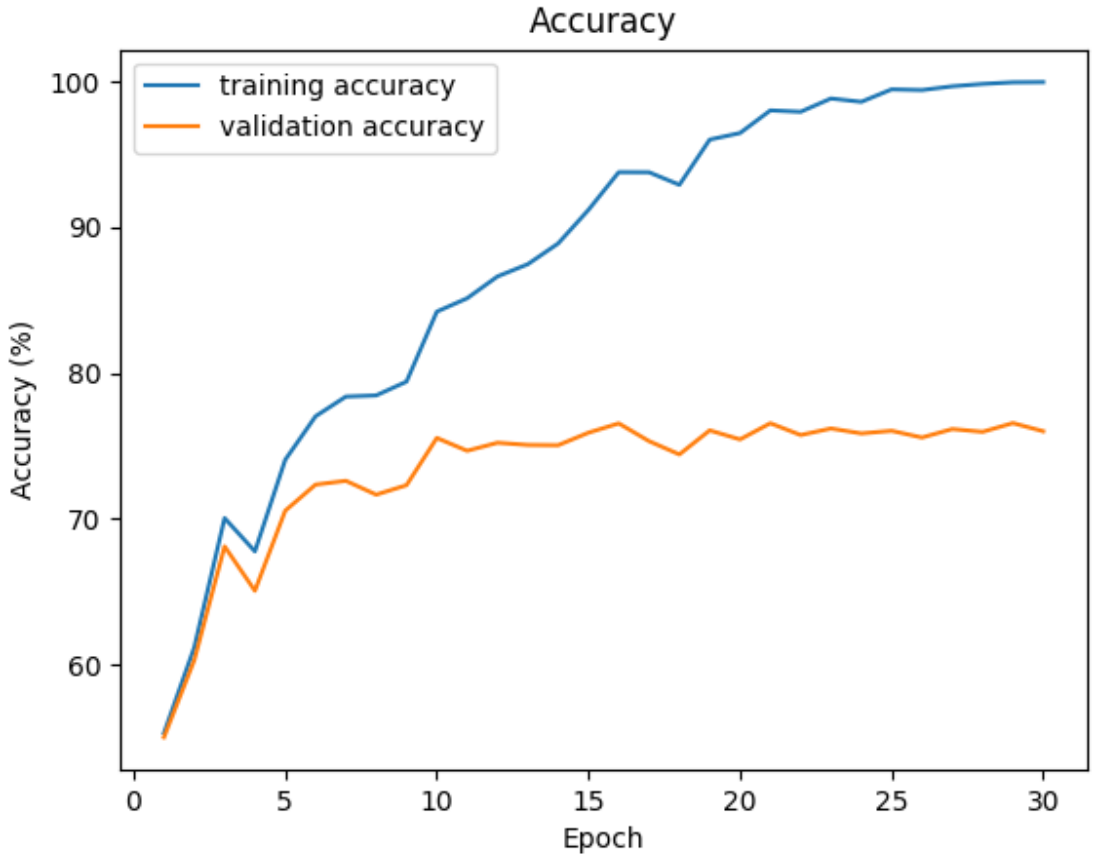
with batch normalization layer:

EPOCH: 1 train_loss: 1.26953 train_acc: 0.55312 test_loss: 1.27857 test_acc 0.55030
EPOCH: 2 train_loss: 1.09750 train_acc: 0.61194 test_loss: 1.12732 test_acc 0.60320
EPOCH: 3 train_loss: 0.87812 train_acc: 0.70062 test_loss: 0.93380 test_acc 0.68120
EPOCH: 4 train_loss: 0.90808 train_acc: 0.67782 test_loss: 0.98914 test_acc 0.65080
EPOCH: 5 train_loss: 0.74535 train_acc: 0.74062 test_loss: 0.84785 test_acc 0.70570
EPOCH: 6 train_loss: 0.67593 train_acc: 0.77046 test_loss: 0.80480 test_acc 0.72360
EPOCH: 7 train_loss: 0.62278 train_acc: 0.78398 test_loss: 0.78621 test_acc 0.72620
EPOCH: 8 train_loss: 0.61557 train_acc: 0.78478 test_loss: 0.80515 test_acc 0.71670
EPOCH: 9 train_loss: 0.59888 train_acc: 0.79424 test_loss: 0.81806 test_acc 0.72320
EPOCH: 10 train_loss: 0.47696 train_acc: 0.84224 test_loss: 0.72105 test_acc 0.75560
EPOCH: 11 train_loss: 0.45117 train_acc: 0.85140 test_loss: 0.74791 test_acc 0.74680
EPOCH: 12 train_loss: 0.40769 train_acc: 0.86638 test_loss: 0.73743 test_acc 0.75230
EPOCH: 13 train_loss: 0.38450 train_acc: 0.87476 test_loss: 0.74707 test_acc 0.75080
EPOCH: 14 train_loss: 0.34132 train_acc: 0.88898 test_loss: 0.74689 test_acc 0.75060
EPOCH: 15 train_loss: 0.29072 train_acc: 0.91200 test_loss: 0.72691 test_acc 0.75920
EPOCH: 16 train_loss: 0.22789 train_acc: 0.93790 test_loss: 0.70977 test_acc 0.76560
EPOCH: 17 train_loss: 0.23049 train_acc: 0.93782 test_loss: 0.74883 test_acc 0.75340
EPOCH: 18 train_loss: 0.23507 train_acc: 0.92922 test_loss: 0.80641 test_acc 0.74430
EPOCH: 19 train_loss: 0.16881 train_acc: 0.96024 test_loss: 0.76077 test_acc 0.76080
EPOCH: 20 train_loss: 0.14797 train_acc: 0.96474 test_loss: 0.79045 test_acc 0.75460
EPOCH: 21 train_loss: 0.10685 train_acc: 0.98026 test_loss: 0.77310 test_acc 0.76570
EPOCH: 22 train_loss: 0.10738 train_acc: 0.97932 test_loss: 0.80575 test_acc 0.75760
EPOCH: 23 train_loss: 0.08469 train_acc: 0.98842 test_loss: 0.79404 test_acc 0.76210
EPOCH: 24 train_loss: 0.08538 train_acc: 0.98614 test_loss: 0.83925 test_acc 0.75870
EPOCH: 25 train_loss: 0.05667 train_acc: 0.99470 test_loss: 0.83286 test_acc 0.76050
EPOCH: 26 train_loss: 0.05385 train_acc: 0.99422 test_loss: 0.86668 test_acc 0.75590
EPOCH: 27 train_loss: 0.04144 train_acc: 0.99680 test_loss: 0.86340 test_acc 0.76170
EPOCH: 28 train_loss: 0.03363 train_acc: 0.99842 test_loss: 0.85365 test_acc 0.75980
EPOCH: 29 train_loss: 0.02359 train_acc: 0.99954 test_loss: 0.85912 test_acc 0.76580
EPOCH: 30 train_loss: 0.02124 train_acc: 0.99970 test_loss: 0.87126 test_acc 0.76020

observation:
The final train/testing accuracy of the model with batch normalization is much higher than the model without batch normalization.

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
60
70
80
90
100
0
5
10
15
20
25
30

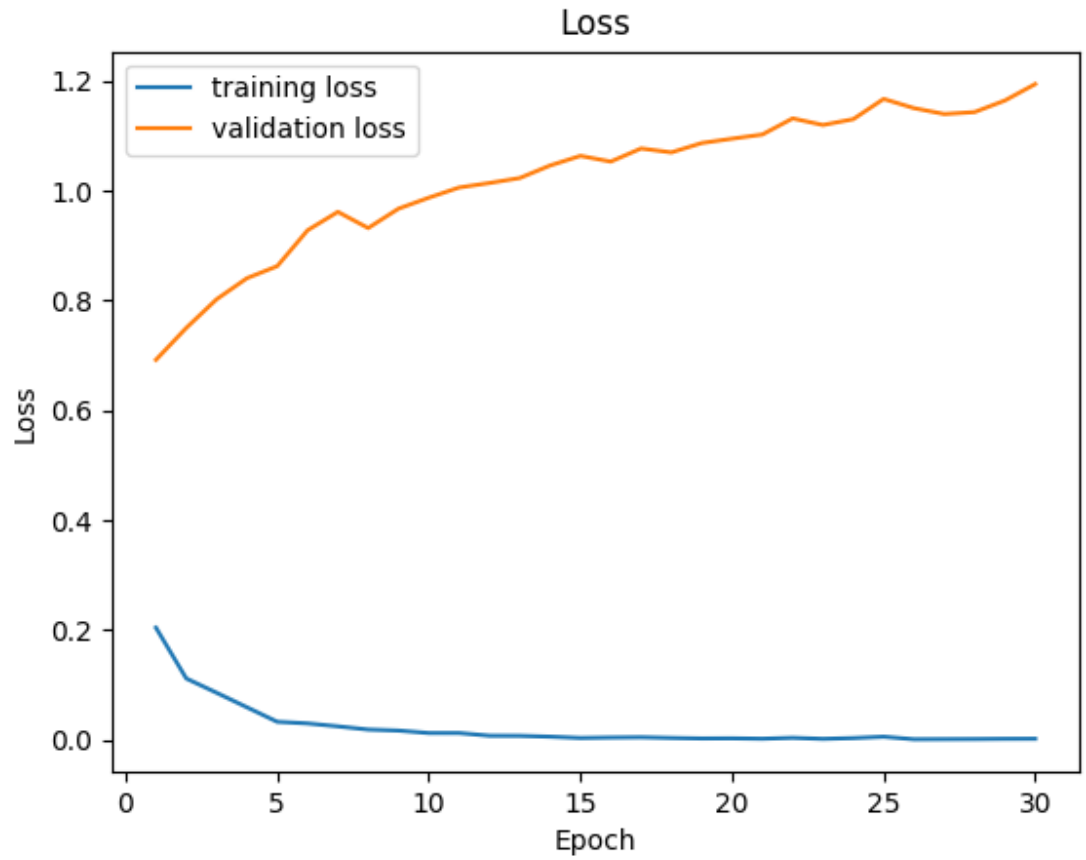
CS535 HW3

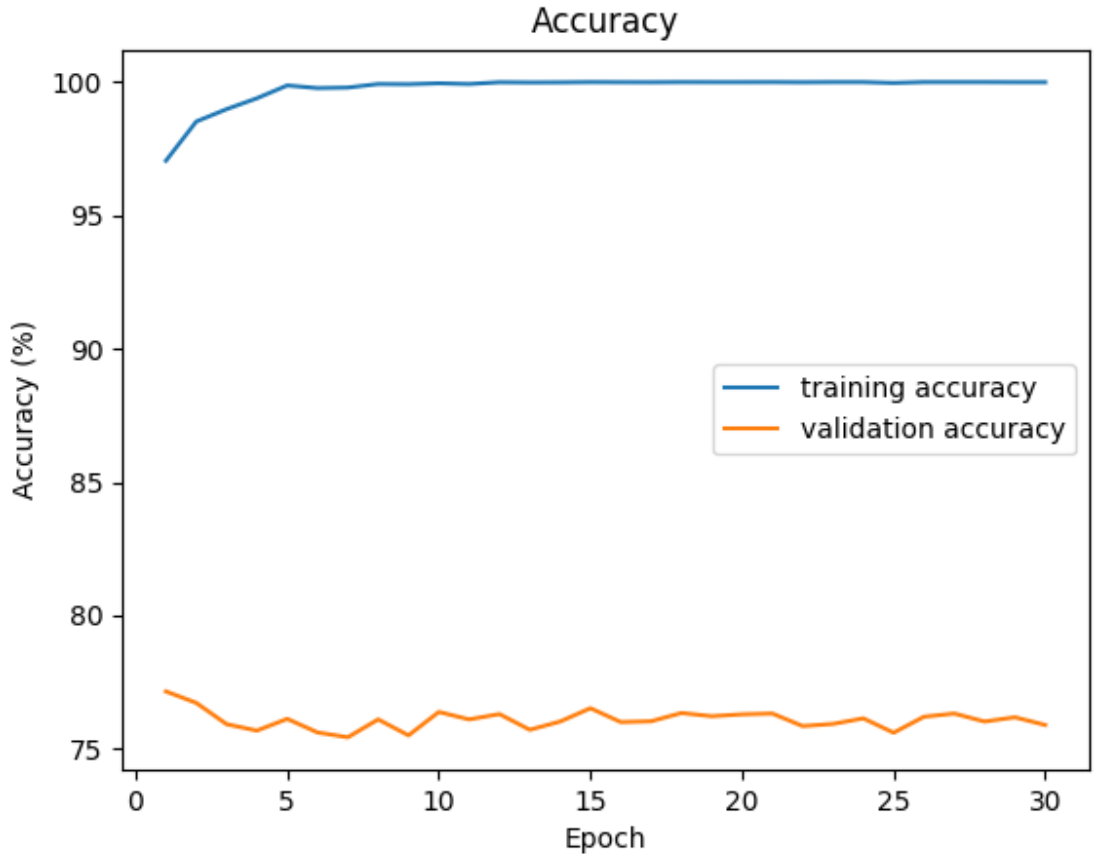
2) Modify our model by adding another fully connected layer with 512 nodes at the second-to-last layer (before the fc2 layer) (10 points).
Apply the model weights you saved at step 1 to initialize to the new model(only up to fc2 layer since after that all layers are newly created) before training. Train and save the model (Hint: check the end of the assignment description to see how to partially restore weights from a pretrained weights file).

optimizer = optim.SGD(net.parameters(), lr=0.0005, momentum=0.8)

EPOCH: 1 train_loss: 0.20439 train_acc: 0.97050 test_loss: 0.69216 test_acc 0.77150
EPOCH: 2 train_loss: 0.11190 train_acc: 0.98520 test_loss: 0.75029 test_acc 0.76720
EPOCH: 3 train_loss: 0.08588 train_acc: 0.98984 test_loss: 0.80242 test_acc 0.75920
EPOCH: 4 train_loss: 0.05944 train_acc: 0.99392 test_loss: 0.84080 test_acc 0.75680
EPOCH: 5 train_loss: 0.03300 train_acc: 0.99874 test_loss: 0.86273 test_acc 0.76120
EPOCH: 6 train_loss: 0.03008 train_acc: 0.99774 test_loss: 0.92804 test_acc 0.75610
EPOCH: 7 train_loss: 0.02471 train_acc: 0.99794 test_loss: 0.96162 test_acc 0.75430
EPOCH: 8 train_loss: 0.01888 train_acc: 0.99928 test_loss: 0.93214 test_acc 0.76100
EPOCH: 9 train_loss: 0.01701 train_acc: 0.99920 test_loss: 0.96745 test_acc 0.75500
EPOCH: 10 train_loss: 0.01253 train_acc: 0.99954 test_loss: 0.98742 test_acc 0.76380
EPOCH: 11 train_loss: 0.01251 train_acc: 0.99928 test_loss: 1.00598 test_acc 0.76100
EPOCH: 12 train_loss: 0.00765 train_acc: 0.99994 test_loss: 1.01417 test_acc 0.76300
EPOCH: 13 train_loss: 0.00745 train_acc: 0.99986 test_loss: 1.02324 test_acc 0.75710
EPOCH: 14 train_loss: 0.00588 train_acc: 0.99990 test_loss: 1.04609 test_acc 0.76020
EPOCH: 15 train_loss: 0.00357 train_acc: 1.00000 test_loss: 1.06360 test_acc 0.76520
EPOCH: 16 train_loss: 0.00434 train_acc: 0.99996 test_loss: 1.05318 test_acc 0.76000
EPOCH: 17 train_loss: 0.00481 train_acc: 0.99994 test_loss: 1.07706 test_acc 0.76030
EPOCH: 18 train_loss: 0.00359 train_acc: 1.00000 test_loss: 1.06991 test_acc 0.76340
EPOCH: 19 train_loss: 0.00267 train_acc: 0.99998 test_loss: 1.08694 test_acc 0.76220
EPOCH: 20 train_loss: 0.00287 train_acc: 0.99998 test_loss: 1.09487 test_acc 0.76290
EPOCH: 21 train_loss: 0.00213 train_acc: 1.00000 test_loss: 1.10246 test_acc 0.76320
EPOCH: 22 train_loss: 0.00382 train_acc: 0.99994 test_loss: 1.13182 test_acc 0.75850
EPOCH: 23 train_loss: 0.00188 train_acc: 1.00000 test_loss: 1.11980 test_acc 0.75930
EPOCH: 24 train_loss: 0.00336 train_acc: 1.00000 test_loss: 1.13043 test_acc 0.76140
EPOCH: 25 train_loss: 0.00579 train_acc: 0.99968 test_loss: 1.16729 test_acc 0.75600
EPOCH: 26 train_loss: 0.00118 train_acc: 1.00000 test_loss: 1.15053 test_acc 0.76200
EPOCH: 27 train_loss: 0.00137 train_acc: 1.00000 test_loss: 1.13976 test_acc 0.76320
EPOCH: 28 train_loss: 0.00155 train_acc: 1.00000 test_loss: 1.14330 test_acc 0.76020
EPOCH: 29 train_loss: 0.00195 train_acc: 0.99996 test_loss: 1.16441 test_acc 0.76180
EPOCH: 30 train_loss: 0.00209 train_acc: 0.99996 test_loss: 1.19435 test_acc 0.75890

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
75
80
85
90
95
100
0
5
10
15
20
25
30

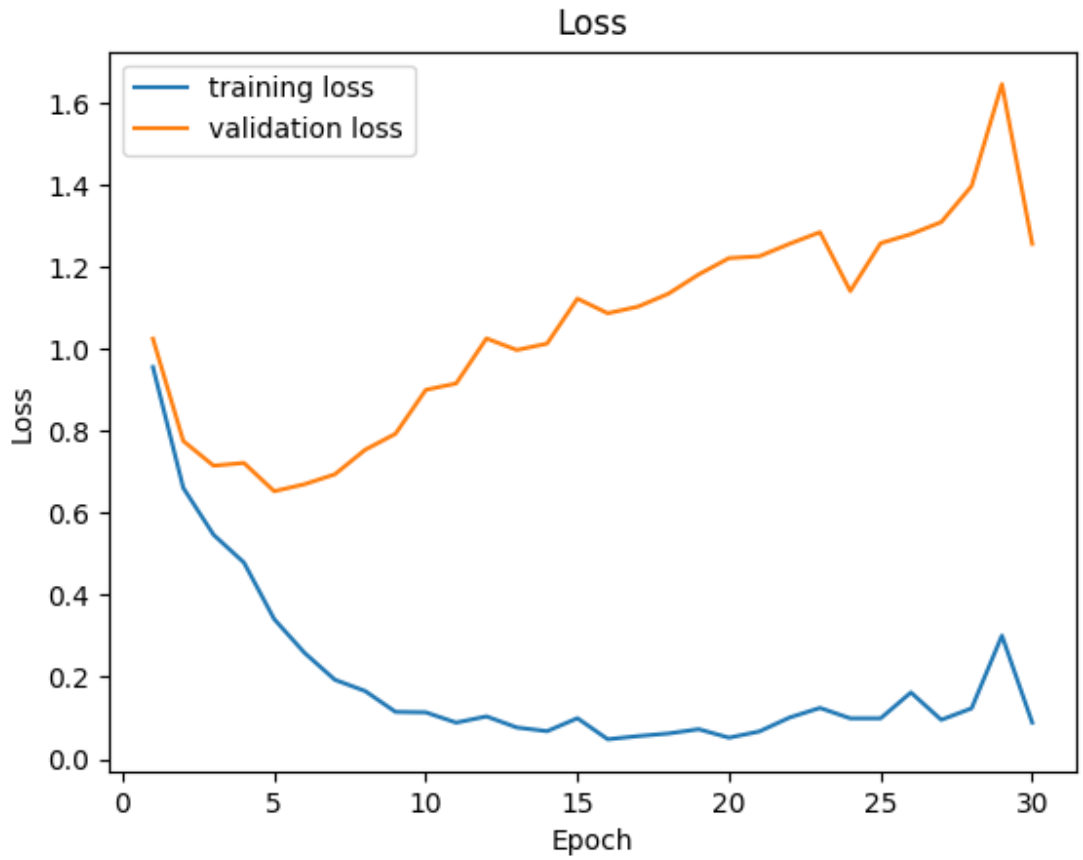
CS535 HW3

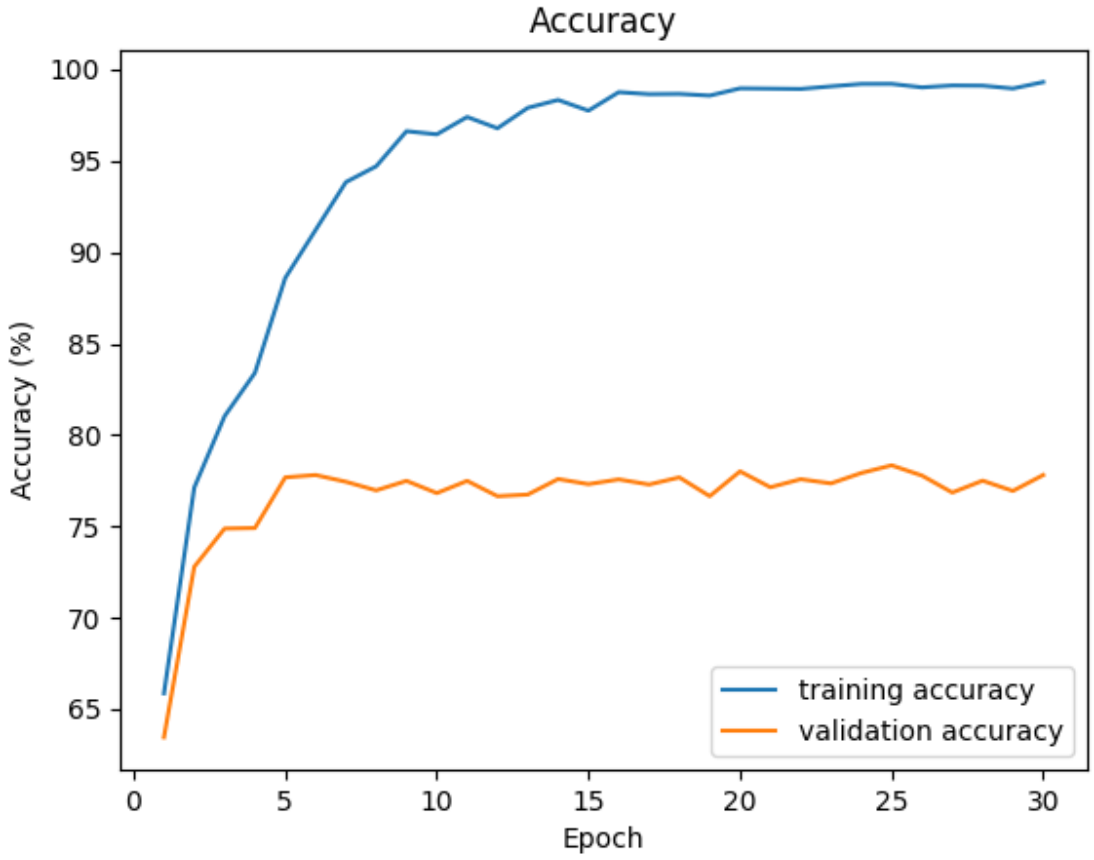
3) Try to use an adaptive schedule to tune the learning rate, you can choose from RMSprop, Adagrad and Adam (Hint: you don't need to implement any of these, look at Pytorch documentation please) (10 points).

optimizer = optim.Adam(net.parameters(), lr=0.0005)

EPOCH: 1 train_loss: 0.95639 train_acc: 0.65864 test_loss: 1.02547 test_acc 0.63460
EPOCH: 2 train_loss: 0.66152 train_acc: 0.77126 test_loss: 0.77597 test_acc 0.72790
EPOCH: 3 train_loss: 0.54705 train_acc: 0.81030 test_loss: 0.71572 test_acc 0.74880
EPOCH: 4 train_loss: 0.47939 train_acc: 0.83398 test_loss: 0.72233 test_acc 0.74910
EPOCH: 5 train_loss: 0.34120 train_acc: 0.88590 test_loss: 0.65336 test_acc 0.77680
EPOCH: 6 train_loss: 0.25891 train_acc: 0.91214 test_loss: 0.67045 test_acc 0.77810
EPOCH: 7 train_loss: 0.19334 train_acc: 0.93834 test_loss: 0.69468 test_acc 0.77440
EPOCH: 8 train_loss: 0.16574 train_acc: 0.94704 test_loss: 0.75456 test_acc 0.76970
EPOCH: 9 train_loss: 0.11499 train_acc: 0.96622 test_loss: 0.79360 test_acc 0.77490
EPOCH: 10 train_loss: 0.11380 train_acc: 0.96450 test_loss: 0.90076 test_acc 0.76820
EPOCH: 11 train_loss: 0.08869 train_acc: 0.97402 test_loss: 0.91630 test_acc 0.77490
EPOCH: 12 train_loss: 0.10389 train_acc: 0.96782 test_loss: 1.02608 test_acc 0.76650
EPOCH: 13 train_loss: 0.07684 train_acc: 0.97900 test_loss: 0.99794 test_acc 0.76740
EPOCH: 14 train_loss: 0.06785 train_acc: 0.98332 test_loss: 1.01358 test_acc 0.77590
EPOCH: 15 train_loss: 0.09948 train_acc: 0.97746 test_loss: 1.12333 test_acc 0.77310
EPOCH: 16 train_loss: 0.04811 train_acc: 0.98758 test_loss: 1.08761 test_acc 0.77570
EPOCH: 17 train_loss: 0.05584 train_acc: 0.98648 test_loss: 1.10383 test_acc 0.77290
EPOCH: 18 train_loss: 0.06225 train_acc: 0.98662 test_loss: 1.13529 test_acc 0.77680
EPOCH: 19 train_loss: 0.07261 train_acc: 0.98570 test_loss: 1.18239 test_acc 0.76650
EPOCH: 20 train_loss: 0.05193 train_acc: 0.98966 test_loss: 1.22214 test_acc 0.78010
EPOCH: 21 train_loss: 0.06718 train_acc: 0.98952 test_loss: 1.22659 test_acc 0.77140
EPOCH: 22 train_loss: 0.10125 train_acc: 0.98934 test_loss: 1.25723 test_acc 0.77580
EPOCH: 23 train_loss: 0.12461 train_acc: 0.99076 test_loss: 1.28546 test_acc 0.77350
EPOCH: 24 train_loss: 0.09918 train_acc: 0.99208 test_loss: 1.14177 test_acc 0.77920
EPOCH: 25 train_loss: 0.09920 train_acc: 0.99212 test_loss: 1.25853 test_acc 0.78340
EPOCH: 26 train_loss: 0.16229 train_acc: 0.99020 test_loss: 1.28066 test_acc 0.77780
EPOCH: 27 train_loss: 0.09559 train_acc: 0.99124 test_loss: 1.31012 test_acc 0.76850
EPOCH: 28 train_loss: 0.12357 train_acc: 0.99114 test_loss: 1.39792 test_acc 0.77500
EPOCH: 29 train_loss: 0.30105 train_acc: 0.98958 test_loss: 1.64686 test_acc 0.76940
EPOCH: 30 train_loss: 0.08877 train_acc: 0.99310 test_loss: 1.25696 test_acc 0.77810

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
65
70
75
80
85
90
95
100
0
5
10
15
20
25
30

CS535 HW3

4) Try to tune your network in two other ways (10 points) (e.g. add/remove a layer, change the activation function, add/remove regularizer, change the number of hidden units, more batch normalization layers) not described in the previous four. You can start from random initialization or previous results as you wish.

1)
optimizer = optim.Adam(net.parameters(), lr=0.0003)

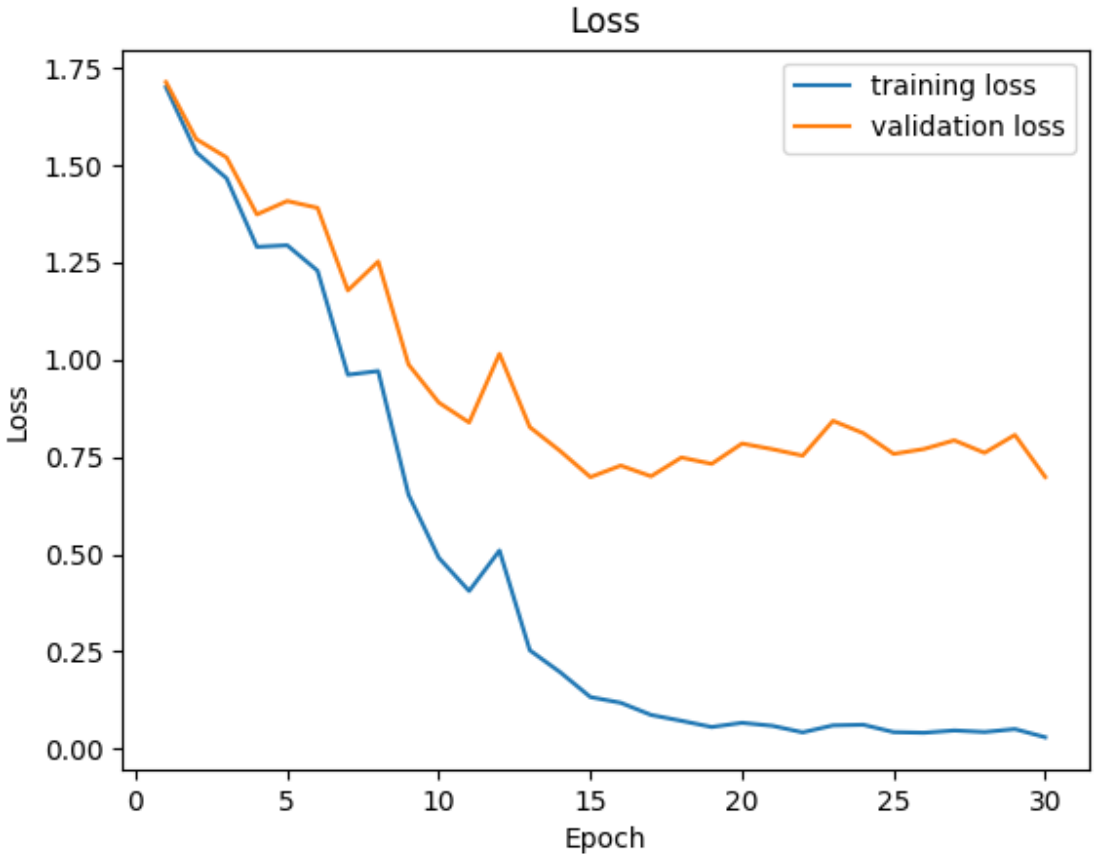
```
# for Q4-1
def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(3, 64, 3, padding=1)
    self.conv2 = nn.Conv2d(64, 64, 3, padding=1)
    self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
    self.conv4 = nn.Conv2d(128, 128, 3, padding=1)
    self.conv5 = nn.Conv2d(128, 256, 3, padding=1)
    self.conv6 = nn.Conv2d(256, 256, 3, padding=1)
    self.pool = nn.MaxPool2d(2, 2)
    self.bNorm = nn.BatchNorm2d(512)
    self.fc1 = nn.Linear(256 * 4 * 4, 512)
    self.fcm = nn.Linear(512, 512)
    self.fc2 = nn.Linear(512, 10)
```

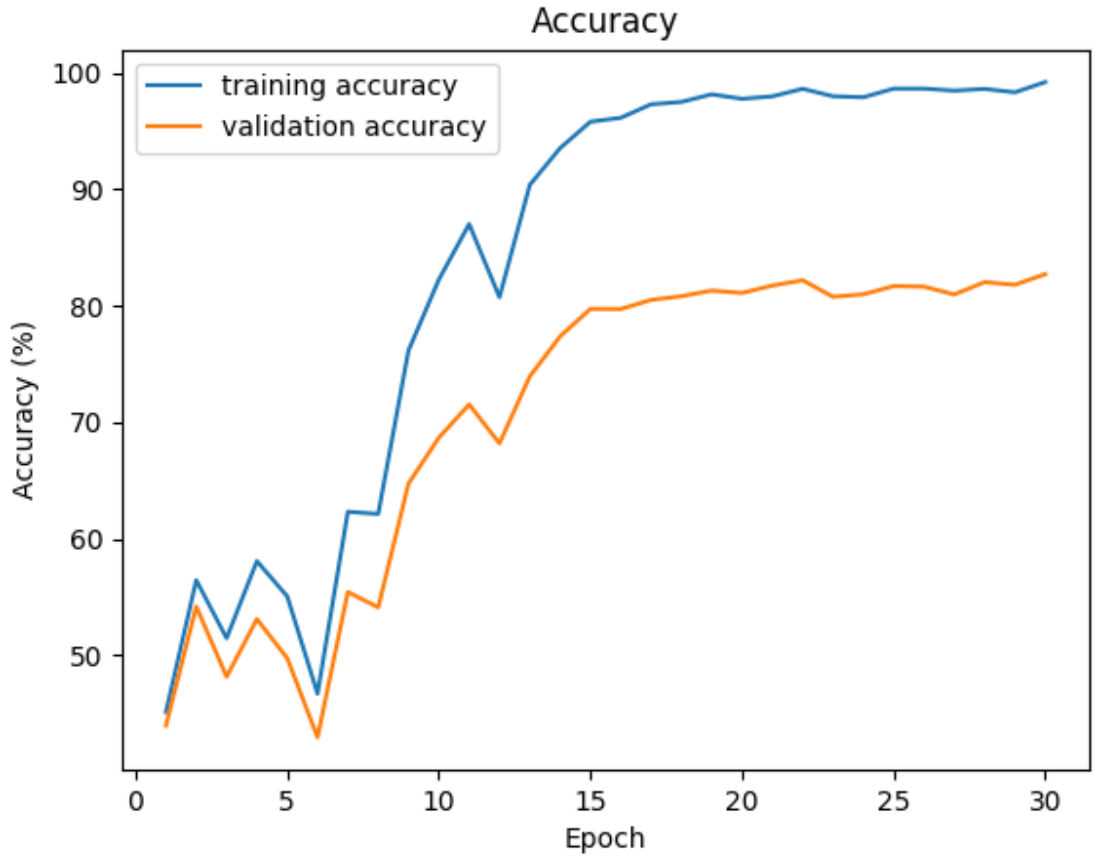
```
def forward(self, x):
    x = F.leaky_relu(self.conv1(x))
    x = F.leaky_relu(self.conv2(x))
    x = self.pool(x)
    x = F.leaky_relu(self.conv3(x))
    x = F.leaky_relu(self.conv4(x))
    x = self.pool(x)
    x = F.leaky_relu(self.conv5(x))
    x = F.leaky_relu(self.conv6(x))
    x = self.pool(x)
    x = x.view(-1, self.num_flat_features(x))
    x = F.leaky_relu(self.fc1(x))
    x = self.bNorm(x)
    x = F.leaky_relu(self.fcm(x))
    x = self.bNorm(x)
    x = self.fc2(x)
    return x
```

CS535 HW3

EPOCH: 1 train_loss: 1.70131 train_acc: 0.45150 test_loss: 1.71433 test_acc 0.43980
EPOCH: 2 train_loss: 1.53380 train_acc: 0.56460 test_loss: 1.56763 test_acc 0.54200
EPOCH: 3 train_loss: 1.46680 train_acc: 0.51484 test_loss: 1.52050 test_acc 0.48170
EPOCH: 4 train_loss: 1.29054 train_acc: 0.58092 test_loss: 1.37398 test_acc 0.53120
EPOCH: 5 train_loss: 1.29459 train_acc: 0.55076 test_loss: 1.40846 test_acc 0.49760
EPOCH: 6 train_loss: 1.22919 train_acc: 0.46718 test_loss: 1.39085 test_acc 0.42980
EPOCH: 7 train_loss: 0.96160 train_acc: 0.62334 test_loss: 1.17850 test_acc 0.55450
EPOCH: 8 train_loss: 0.97112 train_acc: 0.62146 test_loss: 1.25277 test_acc 0.54130
EPOCH: 9 train_loss: 0.65310 train_acc: 0.76172 test_loss: 0.98814 test_acc 0.64750
EPOCH: 10 train_loss: 0.49079 train_acc: 0.82254 test_loss: 0.89013 test_acc 0.68690
EPOCH: 11 train_loss: 0.40545 train_acc: 0.87022 test_loss: 0.83878 test_acc 0.71540
EPOCH: 12 train_loss: 0.50954 train_acc: 0.80746 test_loss: 1.01555 test_acc 0.68200
EPOCH: 13 train_loss: 0.25304 train_acc: 0.90384 test_loss: 0.82702 test_acc 0.73950
EPOCH: 14 train_loss: 0.19682 train_acc: 0.93538 test_loss: 0.76564 test_acc 0.77380
EPOCH: 15 train_loss: 0.13268 train_acc: 0.95804 test_loss: 0.69846 test_acc 0.79740
EPOCH: 16 train_loss: 0.11803 train_acc: 0.96138 test_loss: 0.72810 test_acc 0.79730
EPOCH: 17 train_loss: 0.08673 train_acc: 0.97296 test_loss: 0.70052 test_acc 0.80510
EPOCH: 18 train_loss: 0.07167 train_acc: 0.97502 test_loss: 0.74879 test_acc 0.80830
EPOCH: 19 train_loss: 0.05576 train_acc: 0.98154 test_loss: 0.73247 test_acc 0.81310
EPOCH: 20 train_loss: 0.06650 train_acc: 0.97768 test_loss: 0.78473 test_acc 0.81110
EPOCH: 21 train_loss: 0.05880 train_acc: 0.97980 test_loss: 0.77023 test_acc 0.81740
EPOCH: 22 train_loss: 0.04196 train_acc: 0.98630 test_loss: 0.75359 test_acc 0.82200
EPOCH: 23 train_loss: 0.05989 train_acc: 0.97988 test_loss: 0.84313 test_acc 0.80780
EPOCH: 24 train_loss: 0.06150 train_acc: 0.97902 test_loss: 0.81137 test_acc 0.80990
EPOCH: 25 train_loss: 0.04253 train_acc: 0.98642 test_loss: 0.75812 test_acc 0.81700
EPOCH: 26 train_loss: 0.04092 train_acc: 0.98644 test_loss: 0.77060 test_acc 0.81650
EPOCH: 27 train_loss: 0.04680 train_acc: 0.98470 test_loss: 0.79303 test_acc 0.80980
EPOCH: 28 train_loss: 0.04285 train_acc: 0.98620 test_loss: 0.76084 test_acc 0.82040
EPOCH: 29 train_loss: 0.05037 train_acc: 0.98328 test_loss: 0.80692 test_acc 0.81810
EPOCH: 30 train_loss: 0.02932 train_acc: 0.99206 test_loss: 0.69876 test_acc 0.82720

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
100
90
80
70
60
50
0
5
10
15
20
25
30

CS535 HW3

2)
optimizer = optim.Adam(net.parameters(), lr=0.0003)

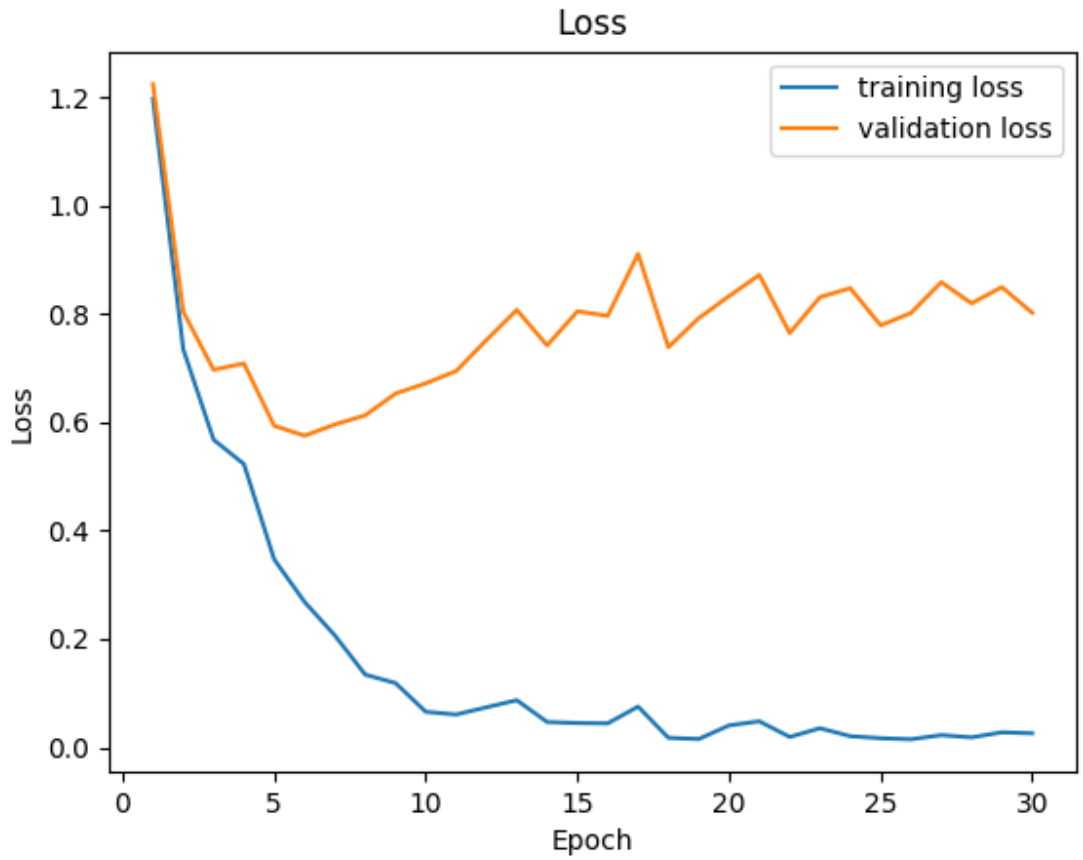
```
# for Q4-2
def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(3, 64, 3, padding=1)
    self.conv2 = nn.Conv2d(64, 64, 3, padding=1)
    self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
    self.conv4 = nn.Conv2d(128, 128, 3, padding=1)
    self.conv5 = nn.Conv2d(128, 256, 3, padding=1)
    self.conv6 = nn.Conv2d(256, 256, 3, padding=1)
    self.conv7 = nn.Conv2d(256, 512, 3, padding=1)
    self.conv8 = nn.Conv2d(512, 512, 3, padding=1)
    self.pool = nn.MaxPool2d(2, 2)
    self.bNorm = nn.BatchNorm2d(512)
    self.fc1 = nn.Linear(512 * 2 * 2, 512)
    self.fcm = nn.Linear(512, 512)
    self.fc2 = nn.Linear(512, 10)
```

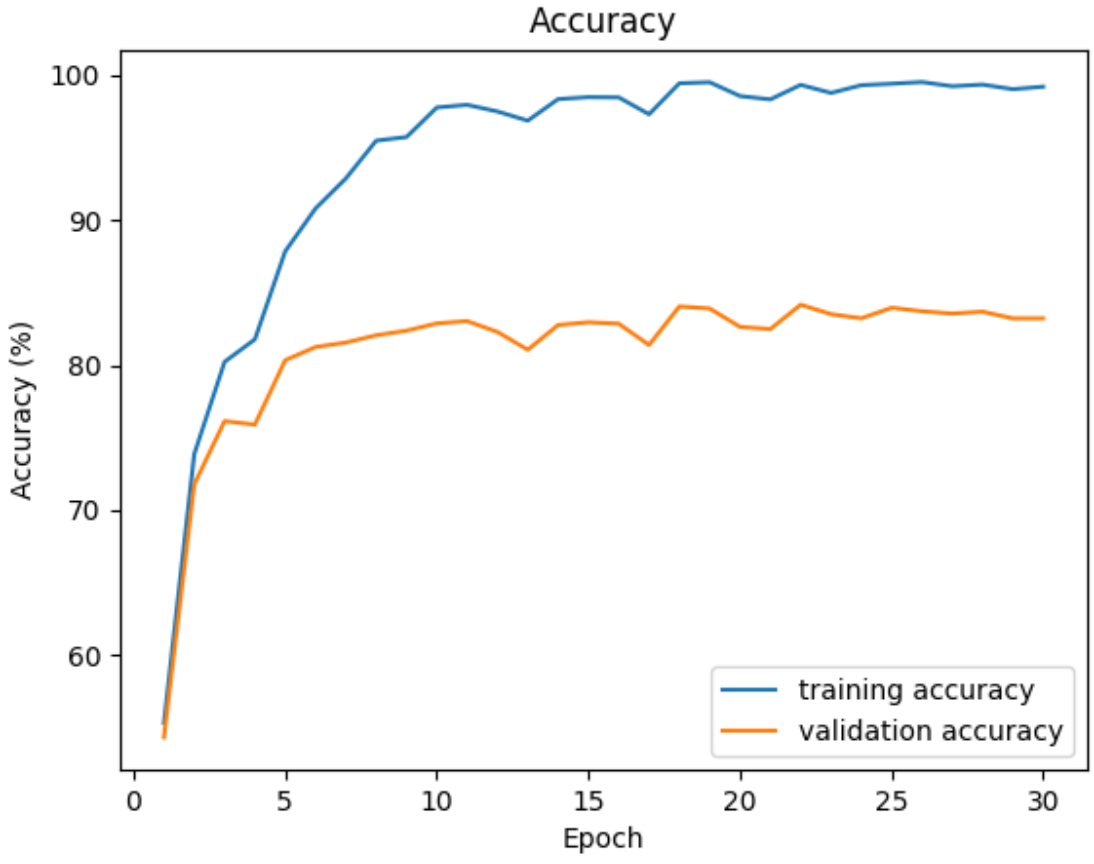
```
def forward(self, x):
    x = F.leaky_relu(self.conv1(x))
    x = F.leaky_relu(self.conv2(x))
    x = self.pool(x)
    x = F.leaky_relu(self.conv3(x))
    x = F.leaky_relu(self.conv4(x))
    x = self.pool(x)
    x = F.leaky_relu(self.conv5(x))
    x = F.leaky_relu(self.conv6(x))
    x = self.pool(x)
    x = F.leaky_relu(self.conv7(x))
    x = F.leaky_relu(self.conv8(x))
    x = self.pool(x)
    x = x.view(-1, self.num_flat_features(x))
    x = F.leaky_relu(self.fc1(x))
    x = self.bNorm(x)
    x = F.leaky_relu(self.fcm(x))
    x = self.fc2(x)
    return x
```

CS535 HW3

EPOCH: 1 train_loss: 1.19693 train_acc: 0.55322 test_loss: 1.22404 test_acc 0.54330
EPOCH: 2 train_loss: 0.73469 train_acc: 0.73866 test_loss: 0.80394 test_acc 0.71780
EPOCH: 3 train_loss: 0.56787 train_acc: 0.80230 test_loss: 0.69685 test_acc 0.76140
EPOCH: 4 train_loss: 0.52280 train_acc: 0.81780 test_loss: 0.70853 test_acc 0.75900
EPOCH: 5 train_loss: 0.34680 train_acc: 0.87854 test_loss: 0.59345 test_acc 0.80340
EPOCH: 6 train_loss: 0.26850 train_acc: 0.90808 test_loss: 0.57509 test_acc 0.81250
EPOCH: 7 train_loss: 0.20655 train_acc: 0.92880 test_loss: 0.59586 test_acc 0.81560
EPOCH: 8 train_loss: 0.13435 train_acc: 0.95486 test_loss: 0.61268 test_acc 0.82060
EPOCH: 9 train_loss: 0.11871 train_acc: 0.95736 test_loss: 0.65296 test_acc 0.82380
EPOCH: 10 train_loss: 0.06564 train_acc: 0.97776 test_loss: 0.67197 test_acc 0.82880
EPOCH: 11 train_loss: 0.06068 train_acc: 0.97956 test_loss: 0.69440 test_acc 0.83040
EPOCH: 12 train_loss: 0.07406 train_acc: 0.97498 test_loss: 0.75178 test_acc 0.82290
EPOCH: 13 train_loss: 0.08713 train_acc: 0.96866 test_loss: 0.80732 test_acc 0.81060
EPOCH: 14 train_loss: 0.04690 train_acc: 0.98352 test_loss: 0.74180 test_acc 0.82770
EPOCH: 15 train_loss: 0.04494 train_acc: 0.98488 test_loss: 0.80465 test_acc 0.82960
EPOCH: 16 train_loss: 0.04440 train_acc: 0.98470 test_loss: 0.79637 test_acc 0.82860
EPOCH: 17 train_loss: 0.07530 train_acc: 0.97306 test_loss: 0.91067 test_acc 0.81390
EPOCH: 18 train_loss: 0.01776 train_acc: 0.99444 test_loss: 0.73846 test_acc 0.84050
EPOCH: 19 train_loss: 0.01544 train_acc: 0.99516 test_loss: 0.79213 test_acc 0.83910
EPOCH: 20 train_loss: 0.04055 train_acc: 0.98562 test_loss: 0.83265 test_acc 0.82640
EPOCH: 21 train_loss: 0.04815 train_acc: 0.98332 test_loss: 0.87173 test_acc 0.82490
EPOCH: 22 train_loss: 0.01928 train_acc: 0.99334 test_loss: 0.76460 test_acc 0.84170
EPOCH: 23 train_loss: 0.03539 train_acc: 0.98778 test_loss: 0.83085 test_acc 0.83520
EPOCH: 24 train_loss: 0.02066 train_acc: 0.99312 test_loss: 0.84739 test_acc 0.83230
EPOCH: 25 train_loss: 0.01699 train_acc: 0.99414 test_loss: 0.77881 test_acc 0.83960
EPOCH: 26 train_loss: 0.01501 train_acc: 0.99528 test_loss: 0.80162 test_acc 0.83720
EPOCH: 27 train_loss: 0.02268 train_acc: 0.99240 test_loss: 0.85851 test_acc 0.83560
EPOCH: 28 train_loss: 0.01850 train_acc: 0.99354 test_loss: 0.81934 test_acc 0.83690
EPOCH: 29 train_loss: 0.02779 train_acc: 0.99036 test_loss: 0.84944 test_acc 0.83230
EPOCH: 30 train_loss: 0.02608 train_acc: 0.99210 test_loss: 0.80202 test_acc 0.83230

CS535 HW3


Loss
training loss
validation loss
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30


Accuracy
training accuracy
validation accuracy
Accuracy (%)
Epoch
60
70
80
90
100
0
5
10
15
20
25
30