ECE285 Assignment 1: Neural Network in NumPy

Use this notebook to build your neural network by implementing the following functions in the python files under `ece285/algorithms` directory:

1. `linear.py`
2. `relu.py`
3. `softmax.py`
4. `loss_func.py`

You will be testing your 2 layer neural network implementation on a toy dataset.

TO SUBMIT: PDF of this notebook with all the required outputs and answers.

In [ ]:
```python
# Setup
import matplotlib.pyplot as plt
import numpy as np

from ece285.layers.sequential import Sequential
from ece285.layers.linear import Linear
from ece285.layers.relu import ReLU
from ece285.layers.softmax import Softmax
from ece285.layers.loss_func import CrossEntropyLoss
from ece285.utils.optimizer import SGD

%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0)  # set default size of plots

# For auto-reloading external modules
# See http://stackoverflow.com/questions/1907993/autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2
```

We will use the class Sequential as implemented in the file assignment2/layers/sequential.py to build a layer by layer model of our neural network. Below we initialize the toy model and the toy random data that you will use to develop your implementation.

In [ ]:
```python
# Create a small net and some toy data to check your implementations.
# Note that we set the random seed for repeatable experiments.

input_size = 4
hidden_size = 10
num_classes = 3  # Output
num_inputs = 10  # N


def init_toy_model():
    np.random.seed(0)
    l1 = Linear(input_size, hidden_size)
    l2 = Linear(hidden_size, num_classes)

    r1 = ReLU()
    softmax = Softmax()
    return Sequential([l1, r1, l2, softmax])


def init_toy_data():
    np.random.seed(0)
    X = 10 * np.random.randn(num_inputs, input_size)
    y = np.random.randint(num_classes, size=num_inputs)
    # y = np.array([0, 1, 2, 2, 1])
    return X, y


net = init_toy_model()
X, y = init_toy_data()
```

Forward Pass: Compute Scores (20%)

Implement the forward functions in Linear, Relu and Softmax layers and get the output by passing our toy data X The output must match the given output scores

In [ ]:
```
scores = net.forward(X)
print("Your scores:")
print(scores)
print()
print("correct scores:")
correct_scores = np.asarray(
    [
        [0.33333514, 0.33333826, 0.33332661],
        [0.3333351, 0.33333828, 0.33332661],
        [0.3333351, 0.33333828, 0.33332662],
        [0.3333351, 0.33333828, 0.33332662],
        [0.33333509, 0.33333829, 0.33332662],
        [0.33333508, 0.33333829, 0.33332662],
        [0.33333511, 0.33333828, 0.33332661],
        [0.33333512, 0.33333827, 0.33332661],
        [0.33333508, 0.33333829, 0.33332662],
        [0.33333511, 0.33333828, 0.33332662],
    ]
)
print(correct_scores)

# The difference should be very small. We get < 1e-7
print("Difference between your scores and correct scores:")
print(np.sum(np.abs(scores - correct_scores)))
```

```
Your scores:
[[0.33333514 0.33333826 0.33332661]
 [0.3333351  0.33333828 0.33332661]
 [0.3333351  0.33333828 0.33332662]
 [0.3333351  0.33333828 0.33332662]
 [0.33333509 0.33333829 0.33332662]
 [0.33333508 0.33333829 0.33332662]
 [0.33333511 0.33333828 0.33332661]
 [0.33333512 0.33333827 0.33332661]
 [0.33333508 0.33333829 0.33332662]
 [0.33333511 0.33333828 0.33332662]]

correct scores:
[[0.33333514 0.33333826 0.33332661]
 [0.3333351  0.33333828 0.33332661]
 [0.3333351  0.33333828 0.33332662]
 [0.3333351  0.33333828 0.33332662]
 [0.33333509 0.33333829 0.33332662]
 [0.33333508 0.33333829 0.33332662]
 [0.33333511 0.33333828 0.33332661]
 [0.33333512 0.33333827 0.33332661]
 [0.33333508 0.33333829 0.33332662]
 [0.33333511 0.33333828 0.33332662]]
Difference between your scores and correct scores:
8.799388540037256e-08
```

Forward Pass: Compute loss given the output scores from the previous step (10%)

Implement the forward function in the loss_func.py file, and output the loss value. The loss value must match the given loss value.

In [ ]:
```
Loss = CrossEntropyLoss()
loss = Loss.forward(scores, y)
correct_loss = 1.098612723362578
print(loss)
# should be very small, we get < 1e-12
print("Difference between your loss and correct loss:")
print(np.sum(np.abs(loss - correct_loss)))
```

```
1.098612723362578
Difference between your loss and correct loss:
0.0
```

Backward Pass (40%)

Implement the rest of the functions in the given files. Specifically, implement the backward function in all the 4 files as mentioned in the files. Note: No backward function in the softmax file, the gradient for softmax is jointly calculated with the cross entropy loss in the loss_func.backward function.

You will use the chain rule to calculate gradient individually for each layer. You can assume that this calculated gradeint then is passed to the next layers in a reversed manner due to the Sequential implementation. So all you need to worry about is implementing the gradient for the current layer and multiply it will the incoming gradient (passed to the backward function as dout) to calculate the total gradient for the parameters of that layer.

We check the values for these gradients by calculating the difference, it is expected to get difference < 1e-8.

In [ ]:
```
# No need to edit anything in this block ( 20% of the above 40% )
net.backward(Loss.backward())

gradients = []
for module in net._modules:
    # print(module)
    for para, grad in zip(module.parameters, module.grads):
        assert grad is not None, "No Gradient"
        # Print gradients of the linear layer
        # print(grad)
        print(grad.shape)
        gradients.append(grad)

# Check shapes of your gradient. Note that only the linear layer has parameters
# (4, 10) -> Layer 1 W
# (10,)   -> Layer 1 b
# (10, 3) -> Layer 2 W
# (3,)    -> Layer 2 b
```

```
(4, 10)
(10,)
(10, 3)
(3,)
```

In [ ]:
```
# No need to edit anything in this block ( 20% of the above 40% )
grad_w1 = np.array(
    [
        [
            -6.24320917e-05,
            3.41037180e-06,
            -1.69125969e-05,
            2.41514079e-05,
            3.88697976e-06,
            7.63842314e-05,
            -8.88925758e-05,
            3.34909890e-05,
            -1.42758303e-05,
            -4.74748560e-06,
        ],
        [
            -7.16182867e-05,
            4.63270039e-06,
            -2.20344270e-05,
            -2.72027034e-06,
            6.52903437e-07,
            8.97294847e-05,
            -1.05981609e-04,
            4.15825391e-05,
            -2.12210745e-05,
            3.06061658e-05,
        ],
        [
            -1.69074923e-05,
            -8.83185056e-06,
            3.10730840e-05,
            1.23010428e-05,
            5.25830316e-05,
            -7.82980115e-06,
            3.02117990e-05,
            -3.37645284e-05,
            6.17276346e-05,
            -1.10735656e-05,
```

```
        ],
        [
            -4.35902272e-05,
            3.71512704e-06,
            -1.66837877e-05,
            2.54069557e-06,
            -4.33258099e-06,
            5.72310022e-05,
            -6.94881762e-05,
            2.92408329e-05,
            -1.89369767e-05,
            2.01692516e-05,
        ],
    ]
)
grad_b1 = np.array(
    [
        -2.27150209e-06,
        5.14674340e-07,
        -2.04284403e-06,
        6.08849787e-07,
        -1.92177796e-06,
        3.92085824e-06,
        -5.40772636e-06,
        2.93354593e-06,
        -3.14568138e-06,
        5.27501592e-11,
    ]
)

grad_w2 = np.array(
    [
        [1.28932983e-04, 1.19946731e-04, -2.48879714e-04],
        [1.08784150e-04, 1.55140199e-04, -2.63924349e-04],
        [6.96017544e-05, 1.42748410e-04, -2.12350164e-04],
        [9.92512487e-05, 1.73257611e-04, -2.72508860e-04],
        [2.05484895e-05, 4.96161144e-05, -7.01646039e-05],
        [8.20539510e-05, 9.37063861e-05, -1.75760337e-04],
        [2.45831715e-05, 8.74369112e-05, -1.12020083e-04],
        [1.34073379e-04, 1.86253064e-04, -3.20326443e-04],
        [8.86473128e-05, 2.35554414e-04, -3.24201726e-04],
        [3.57433149e-05, 1.91164061e-04, -2.26907376e-04],
    ]
)

grad_b2 = np.array([-0.1666649, 0.13333828, 0.03332662])

difference = (
    np.sum(np.abs(gradients[0] - grad_w1))
    + np.sum(np.abs(gradients[1] - grad_b1))
    + np.sum(np.abs(gradients[2] - grad_w2))
    + np.sum(np.abs(gradients[3] - grad_b2))
)
print("Difference in Gradient values", difference)
```

```
Difference in Gradient values 7.70191643436727e-09
```

Train the complete network on the toy data. (30%)

To train the network we will use stochastic gradient descent (SGD), we have implemented the optimizer for you. You do not implement any more functions in the python files. Below we implement the training procedure, you should get yourself familiar with the training process. Specifically looking at which functions to call and when.

Once you have implemented the method and tested various parts in the above blocks, run the code below to train a two-layer network on toy data. You should see your training loss decrease below 0.01.

In [ ]:

```
# Training Procedure
# Initialize the optimizer. DO NOT change any of the hyper-parameters here or above.
# We have implemented the SGD optimizer class for you here, which visits each layer sequentially to
# get the gradients and optimize the respective parameters.
# You should work with the given parameters and only edit your implementation in the .py files

epochs = 1000
```

```python
optim = SGD(net, lr=0.1, weight_decay=0.00001)

epoch_loss = []
for epoch in range(epochs):
    # Get output scores from the network
    output_x = net(X)
    # Calculate the loss for these output scores, given the true labels
    # print(output_x, y)
    loss = Loss.forward(output_x, y)
    # print(loss)
    # Initialize your gradients to None in each epoch
    optim.zero_grad()
    # Make a backward pass to update the internal gradients in the layers
    net.backward(Loss.backward())
    # call the step function in the optimizer to update the values of the params with the gradients
    optim.step()
    # Append the loss at each iteration
    epoch_loss.append(loss)
    print("Epoch {}, loss={:3f}".format(epoch + 1, epoch_loss[-1]))

    if (epoch + 1) % 50 == 0:
        print("Epoch {}, loss={:3f}".format(epoch + 1, epoch_loss[-1]))
```

```
Epoch 1, loss=1.098613
Epoch 2, loss=1.094024
Epoch 3, loss=1.089737
Epoch 4, loss=1.085733
Epoch 5, loss=1.081994
Epoch 6, loss=1.078505
Epoch 7, loss=1.075248
Epoch 8, loss=1.072208
Epoch 9, loss=1.069372
Epoch 10, loss=1.066726
Epoch 11, loss=1.064257
Epoch 12, loss=1.061952
Epoch 13, loss=1.059799
Epoch 14, loss=1.057785
Epoch 15, loss=1.055899
Epoch 16, loss=1.054124
Epoch 17, loss=1.052445
Epoch 18, loss=1.050839
Epoch 19, loss=1.049277
Epoch 20, loss=1.047713
Epoch 21, loss=1.046081
Epoch 22, loss=1.044279
Epoch 23, loss=1.042147
Epoch 24, loss=1.039435
Epoch 25, loss=1.035774
Epoch 26, loss=1.030637
Epoch 27, loss=1.023365
Epoch 28, loss=1.013337
Epoch 29, loss=1.000400
Epoch 30, loss=0.985502
Epoch 31, loss=0.970780
Epoch 32, loss=0.958224
Epoch 33, loss=0.948146
Epoch 34, loss=0.939687
Epoch 35, loss=0.932411
Epoch 36, loss=0.926597
Epoch 37, loss=0.920919
Epoch 38, loss=0.914630
Epoch 39, loss=0.908446
Epoch 40, loss=0.902708
Epoch 41, loss=0.895794
Epoch 42, loss=0.889273
Epoch 43, loss=0.882132
Epoch 44, loss=0.875647
Epoch 45, loss=0.870536
Epoch 46, loss=0.861775
Epoch 47, loss=0.855117
Epoch 48, loss=0.848626
Epoch 49, loss=0.839904
Epoch 50, loss=0.832706
Epoch 50, loss=0.832706
Epoch 51, loss=0.827365
Epoch 52, loss=0.815973
Epoch 53, loss=0.810844
Epoch 54, loss=0.802194
Epoch 55, loss=0.790601
Epoch 56, loss=0.783502
Epoch 57, loss=0.770632
Epoch 58, loss=0.760160
Epoch 59, loss=0.749472
Epoch 60, loss=0.739295
Epoch 61, loss=0.732478
Epoch 62, loss=0.719142
Epoch 63, loss=0.713844
Epoch 64, loss=0.700037
Epoch 65, loss=0.693629
Epoch 66, loss=0.689958
Epoch 67, loss=0.674298
Epoch 68, loss=0.659023
Epoch 69, loss=0.647852
Epoch 70, loss=0.638275
Epoch 71, loss=0.628526
Epoch 72, loss=0.622772
Epoch 73, loss=0.617140
Epoch 74, loss=0.608889
Epoch 75, loss=0.601315
```

```
Epoch 76, loss=0.590965
Epoch 77, loss=0.588483
Epoch 78, loss=0.580111
Epoch 79, loss=0.580237
Epoch 80, loss=0.579590
Epoch 81, loss=0.575265
Epoch 82, loss=0.583167
Epoch 83, loss=0.569299
Epoch 84, loss=0.567051
Epoch 85, loss=0.558750
Epoch 86, loss=0.570455
Epoch 87, loss=0.550303
Epoch 88, loss=0.556647
Epoch 89, loss=0.528476
Epoch 90, loss=0.523165
Epoch 91, loss=0.503276
Epoch 92, loss=0.508981
Epoch 93, loss=0.493973
Epoch 94, loss=0.512211
Epoch 95, loss=0.477031
Epoch 96, loss=0.483331
Epoch 97, loss=0.456475
Epoch 98, loss=0.472357
Epoch 99, loss=0.443569
Epoch 100, loss=0.454687
Epoch 100, loss=0.454687
Epoch 101, loss=0.429413
Epoch 102, loss=0.441666
Epoch 103, loss=0.413838
Epoch 104, loss=0.443494
Epoch 105, loss=0.407146
Epoch 106, loss=0.444991
Epoch 107, loss=0.403453
Epoch 108, loss=0.436572
Epoch 109, loss=0.397764
Epoch 110, loss=0.439797
Epoch 111, loss=0.422846
Epoch 112, loss=0.526893
Epoch 113, loss=0.553876
Epoch 114, loss=0.728167
Epoch 115, loss=0.582263
Epoch 116, loss=0.547702
Epoch 117, loss=0.416240
Epoch 118, loss=0.453693
Epoch 119, loss=0.381732
Epoch 120, loss=0.362521
Epoch 121, loss=0.324897
Epoch 122, loss=0.338047
Epoch 123, loss=0.342623
Epoch 124, loss=0.326819
Epoch 125, loss=0.392589
Epoch 126, loss=0.292861
Epoch 127, loss=0.314854
Epoch 128, loss=0.235608
Epoch 129, loss=0.227234
Epoch 130, loss=0.207276
Epoch 131, loss=0.220121
Epoch 132, loss=0.208397
Epoch 133, loss=0.249017
Epoch 134, loss=0.226502
Epoch 135, loss=0.284894
Epoch 136, loss=0.233826
Epoch 137, loss=0.259128
Epoch 138, loss=0.195131
Epoch 139, loss=0.177242
Epoch 140, loss=0.156995
Epoch 141, loss=0.147287
Epoch 142, loss=0.142642
Epoch 143, loss=0.139515
Epoch 144, loss=0.136126
Epoch 145, loss=0.135978
Epoch 146, loss=0.130053
Epoch 147, loss=0.127937
Epoch 148, loss=0.122477
Epoch 149, loss=0.120641
Epoch 150, loss=0.118350
```

```
Epoch 150, loss=0.118350
Epoch 151, loss=0.117382
Epoch 152, loss=0.112601
Epoch 153, loss=0.111264
Epoch 154, loss=0.106784
Epoch 155, loss=0.105601
Epoch 156, loss=0.105261
Epoch 157, loss=0.103690
Epoch 158, loss=0.099740
Epoch 159, loss=0.097608
Epoch 160, loss=0.094972
Epoch 161, loss=0.094597
Epoch 162, loss=0.093439
Epoch 163, loss=0.092069
Epoch 164, loss=0.090610
Epoch 165, loss=0.087682
Epoch 166, loss=0.085767
Epoch 167, loss=0.084664
Epoch 168, loss=0.083181
Epoch 169, loss=0.082531
Epoch 170, loss=0.081955
Epoch 171, loss=0.079550
Epoch 172, loss=0.078298
Epoch 173, loss=0.077149
Epoch 174, loss=0.075943
Epoch 175, loss=0.074927
Epoch 176, loss=0.074027
Epoch 177, loss=0.074400
Epoch 178, loss=0.072842
Epoch 179, loss=0.071362
Epoch 180, loss=0.070140
Epoch 181, loss=0.069109
Epoch 182, loss=0.068189
Epoch 183, loss=0.067438
Epoch 184, loss=0.066542
Epoch 185, loss=0.065878
Epoch 186, loss=0.065619
Epoch 187, loss=0.065236
Epoch 188, loss=0.064023
Epoch 189, loss=0.063155
Epoch 190, loss=0.062142
Epoch 191, loss=0.061411
Epoch 192, loss=0.060913
Epoch 193, loss=0.060049
Epoch 194, loss=0.059368
Epoch 195, loss=0.059265
Epoch 196, loss=0.058653
Epoch 197, loss=0.057884
Epoch 198, loss=0.057724
Epoch 199, loss=0.056640
Epoch 200, loss=0.055911
Epoch 200, loss=0.055911
Epoch 201, loss=0.055581
Epoch 202, loss=0.054974
Epoch 203, loss=0.054336
Epoch 204, loss=0.053731
Epoch 205, loss=0.053237
Epoch 206, loss=0.052698
Epoch 207, loss=0.052610
Epoch 208, loss=0.052149
Epoch 209, loss=0.051605
Epoch 210, loss=0.051018
Epoch 211, loss=0.050427
Epoch 212, loss=0.050176
Epoch 213, loss=0.050089
Epoch 214, loss=0.049610
Epoch 215, loss=0.048936
Epoch 216, loss=0.048407
Epoch 217, loss=0.048042
Epoch 218, loss=0.047693
Epoch 219, loss=0.047232
Epoch 220, loss=0.047006
Epoch 221, loss=0.046590
Epoch 222, loss=0.046342
Epoch 223, loss=0.045836
Epoch 224, loss=0.045452
```

```
Epoch 225, loss=0.045112
Epoch 226, loss=0.044795
Epoch 227, loss=0.044464
Epoch 228, loss=0.044038
Epoch 229, loss=0.043724
Epoch 230, loss=0.043653
Epoch 231, loss=0.043454
Epoch 232, loss=0.042933
Epoch 233, loss=0.042565
Epoch 234, loss=0.042280
Epoch 235, loss=0.041901
Epoch 236, loss=0.041709
Epoch 237, loss=0.041528
Epoch 238, loss=0.041169
Epoch 239, loss=0.040963
Epoch 240, loss=0.040683
Epoch 241, loss=0.040507
Epoch 242, loss=0.040178
Epoch 243, loss=0.039961
Epoch 244, loss=0.039609
Epoch 245, loss=0.039391
Epoch 246, loss=0.039159
Epoch 247, loss=0.038825
Epoch 248, loss=0.038546
Epoch 249, loss=0.038335
Epoch 250, loss=0.038039
Epoch 250, loss=0.038039
Epoch 251, loss=0.037768
Epoch 252, loss=0.037602
Epoch 253, loss=0.037523
Epoch 254, loss=0.037224
Epoch 255, loss=0.036954
Epoch 256, loss=0.036836
Epoch 257, loss=0.036784
Epoch 258, loss=0.036457
Epoch 259, loss=0.036186
Epoch 260, loss=0.035966
Epoch 261, loss=0.035764
Epoch 262, loss=0.035549
Epoch 263, loss=0.035444
Epoch 264, loss=0.035394
Epoch 265, loss=0.035143
Epoch 266, loss=0.034874
Epoch 267, loss=0.034649
Epoch 268, loss=0.034474
Epoch 269, loss=0.034245
Epoch 270, loss=0.034032
Epoch 271, loss=0.033865
Epoch 272, loss=0.033680
Epoch 273, loss=0.033470
Epoch 274, loss=0.033298
Epoch 275, loss=0.033177
Epoch 276, loss=0.032968
Epoch 277, loss=0.032779
Epoch 278, loss=0.032638
Epoch 279, loss=0.032442
Epoch 280, loss=0.032259
Epoch 281, loss=0.032117
Epoch 282, loss=0.032015
Epoch 283, loss=0.032165
Epoch 284, loss=0.031888
Epoch 285, loss=0.031721
Epoch 286, loss=0.031566
Epoch 287, loss=0.031528
Epoch 288, loss=0.031292
Epoch 289, loss=0.031094
Epoch 290, loss=0.030956
Epoch 291, loss=0.030888
Epoch 292, loss=0.030697
Epoch 293, loss=0.030521
Epoch 294, loss=0.030383
Epoch 295, loss=0.030241
Epoch 296, loss=0.030111
Epoch 297, loss=0.029963
Epoch 298, loss=0.029830
Epoch 299, loss=0.029668
```

```
Epoch 300, loss=0.029528
```

```
Epoch 300, loss=0.029528
```

```
Epoch 301, loss=0.029408
Epoch 302, loss=0.029255
Epoch 303, loss=0.029114
Epoch 304, loss=0.029005
Epoch 305, loss=0.028858
Epoch 306, loss=0.028718
Epoch 307, loss=0.028612
Epoch 308, loss=0.028480
Epoch 309, loss=0.028342
Epoch 310, loss=0.028228
Epoch 311, loss=0.028112
Epoch 312, loss=0.028042
Epoch 313, loss=0.028009
Epoch 314, loss=0.027869
Epoch 315, loss=0.027722
Epoch 316, loss=0.027666
Epoch 317, loss=0.027744
Epoch 318, loss=0.027542
Epoch 319, loss=0.027385
Epoch 320, loss=0.027244
Epoch 321, loss=0.027115
Epoch 322, loss=0.027015
Epoch 323, loss=0.026908
Epoch 324, loss=0.026801
Epoch 325, loss=0.026707
Epoch 326, loss=0.026660
Epoch 327, loss=0.026527
Epoch 328, loss=0.026404
Epoch 329, loss=0.026311
Epoch 330, loss=0.026192
Epoch 331, loss=0.026078
Epoch 332, loss=0.025984
Epoch 333, loss=0.025883
Epoch 334, loss=0.025771
Epoch 335, loss=0.025673
Epoch 336, loss=0.025583
Epoch 337, loss=0.025474
Epoch 338, loss=0.025383
Epoch 339, loss=0.025316
Epoch 340, loss=0.025206
Epoch 341, loss=0.025106
Epoch 342, loss=0.025025
Epoch 343, loss=0.024921
Epoch 344, loss=0.024822
Epoch 345, loss=0.024753
Epoch 346, loss=0.024686
Epoch 347, loss=0.024655
Epoch 348, loss=0.024557
Epoch 349, loss=0.024513
Epoch 350, loss=0.024400
```

```
Epoch 350, loss=0.024400
```

```
Epoch 351, loss=0.024295
Epoch 352, loss=0.024202
Epoch 353, loss=0.024120
Epoch 354, loss=0.024023
Epoch 355, loss=0.023949
Epoch 356, loss=0.023882
Epoch 357, loss=0.023873
Epoch 358, loss=0.023803
Epoch 359, loss=0.023696
Epoch 360, loss=0.023608
Epoch 361, loss=0.023513
Epoch 362, loss=0.023421
Epoch 363, loss=0.023337
Epoch 364, loss=0.023263
Epoch 365, loss=0.023175
Epoch 366, loss=0.023089
Epoch 367, loss=0.023058
Epoch 368, loss=0.022986
Epoch 369, loss=0.022898
Epoch 370, loss=0.022820
Epoch 371, loss=0.022764
Epoch 372, loss=0.022686
Epoch 373, loss=0.022602
```

```
Epoch 374, loss=0.022536
Epoch 375, loss=0.022452
Epoch 376, loss=0.022371
Epoch 377, loss=0.022310
Epoch 378, loss=0.022230
Epoch 379, loss=0.022152
Epoch 380, loss=0.022091
Epoch 381, loss=0.022017
Epoch 382, loss=0.021940
Epoch 383, loss=0.021877
Epoch 384, loss=0.021808
Epoch 385, loss=0.021733
Epoch 386, loss=0.021676
Epoch 387, loss=0.021642
Epoch 388, loss=0.021625
Epoch 389, loss=0.021543
Epoch 390, loss=0.021474
Epoch 391, loss=0.021424
Epoch 392, loss=0.021348
Epoch 393, loss=0.021282
Epoch 394, loss=0.021211
Epoch 395, loss=0.021137
Epoch 396, loss=0.021074
Epoch 397, loss=0.021011
Epoch 398, loss=0.020940
Epoch 399, loss=0.020878
Epoch 400, loss=0.020819
Epoch 400, loss=0.020819
Epoch 401, loss=0.020751
Epoch 402, loss=0.020688
Epoch 403, loss=0.020633
Epoch 404, loss=0.020566
Epoch 405, loss=0.020504
Epoch 406, loss=0.020451
Epoch 407, loss=0.020385
Epoch 408, loss=0.020313
Epoch 409, loss=0.020251
Epoch 410, loss=0.020197
Epoch 411, loss=0.020169
Epoch 412, loss=0.020110
Epoch 413, loss=0.020046
Epoch 414, loss=0.019971
Epoch 415, loss=0.019898
Epoch 416, loss=0.019833
Epoch 417, loss=0.019767
Epoch 418, loss=0.019698
Epoch 419, loss=0.019641
Epoch 420, loss=0.019610
Epoch 421, loss=0.019540
Epoch 422, loss=0.019472
Epoch 423, loss=0.019407
Epoch 424, loss=0.019350
Epoch 425, loss=0.019300
Epoch 426, loss=0.019235
Epoch 427, loss=0.019172
Epoch 428, loss=0.019116
Epoch 429, loss=0.019053
Epoch 430, loss=0.018993
Epoch 431, loss=0.018957
Epoch 432, loss=0.018905
Epoch 433, loss=0.018855
Epoch 434, loss=0.018792
Epoch 435, loss=0.018782
Epoch 436, loss=0.018733
Epoch 437, loss=0.018664
Epoch 438, loss=0.018605
Epoch 439, loss=0.018545
Epoch 440, loss=0.018484
Epoch 441, loss=0.018426
Epoch 442, loss=0.018374
Epoch 443, loss=0.018317
Epoch 444, loss=0.018260
Epoch 445, loss=0.018212
Epoch 446, loss=0.018156
Epoch 447, loss=0.018101
Epoch 448, loss=0.018054
```

```
Epoch 449, loss=0.018001
Epoch 450, loss=0.017947
Epoch 450, loss=0.017947
Epoch 451, loss=0.017900
Epoch 452, loss=0.017862
Epoch 453, loss=0.017808
Epoch 454, loss=0.017761
Epoch 455, loss=0.017711
Epoch 456, loss=0.017659
Epoch 457, loss=0.017613
Epoch 458, loss=0.017564
Epoch 459, loss=0.017513
Epoch 460, loss=0.017468
Epoch 461, loss=0.017421
Epoch 462, loss=0.017371
Epoch 463, loss=0.017327
Epoch 464, loss=0.017281
Epoch 465, loss=0.017232
Epoch 466, loss=0.017188
Epoch 467, loss=0.017143
Epoch 468, loss=0.017096
Epoch 469, loss=0.017052
Epoch 470, loss=0.017009
Epoch 471, loss=0.016962
Epoch 472, loss=0.016919
Epoch 473, loss=0.016877
Epoch 474, loss=0.016831
Epoch 475, loss=0.016825
Epoch 476, loss=0.016779
Epoch 477, loss=0.016733
Epoch 478, loss=0.016688
Epoch 479, loss=0.016648
Epoch 480, loss=0.016615
Epoch 481, loss=0.016571
Epoch 482, loss=0.016538
Epoch 483, loss=0.016511
Epoch 484, loss=0.016489
Epoch 485, loss=0.016449
Epoch 486, loss=0.016401
Epoch 487, loss=0.016354
Epoch 488, loss=0.016312
Epoch 489, loss=0.016269
Epoch 490, loss=0.016226
Epoch 491, loss=0.016184
Epoch 492, loss=0.016159
Epoch 493, loss=0.016143
Epoch 494, loss=0.016097
Epoch 495, loss=0.016054
Epoch 496, loss=0.016014
Epoch 497, loss=0.015972
Epoch 498, loss=0.015933
Epoch 499, loss=0.015906
Epoch 500, loss=0.015866
Epoch 500, loss=0.015866
Epoch 501, loss=0.015825
Epoch 502, loss=0.015785
Epoch 503, loss=0.015748
Epoch 504, loss=0.015709
Epoch 505, loss=0.015670
Epoch 506, loss=0.015635
Epoch 507, loss=0.015597
Epoch 508, loss=0.015559
Epoch 509, loss=0.015523
Epoch 510, loss=0.015487
Epoch 511, loss=0.015450
Epoch 512, loss=0.015415
Epoch 513, loss=0.015380
Epoch 514, loss=0.015343
Epoch 515, loss=0.015308
Epoch 516, loss=0.015275
Epoch 517, loss=0.015238
Epoch 518, loss=0.015203
Epoch 519, loss=0.015171
Epoch 520, loss=0.015135
Epoch 521, loss=0.015100
Epoch 522, loss=0.015069
```

```
Epoch 523, loss=0.015042
Epoch 524, loss=0.015007
Epoch 525, loss=0.014976
Epoch 526, loss=0.014941
Epoch 527, loss=0.014907
Epoch 528, loss=0.014876
Epoch 529, loss=0.014842
Epoch 530, loss=0.014809
Epoch 531, loss=0.014777
Epoch 532, loss=0.014745
Epoch 533, loss=0.014712
Epoch 534, loss=0.014680
Epoch 535, loss=0.014649
Epoch 536, loss=0.014617
Epoch 537, loss=0.014585
Epoch 538, loss=0.014555
Epoch 539, loss=0.014523
Epoch 540, loss=0.014495
Epoch 541, loss=0.014474
Epoch 542, loss=0.014441
Epoch 543, loss=0.014409
Epoch 544, loss=0.014380
Epoch 545, loss=0.014349
Epoch 546, loss=0.014318
Epoch 547, loss=0.014288
Epoch 548, loss=0.014259
Epoch 549, loss=0.014228
Epoch 550, loss=0.014198
Epoch 550, loss=0.014198
Epoch 551, loss=0.014170
Epoch 552, loss=0.014139
Epoch 553, loss=0.014110
Epoch 554, loss=0.014082
Epoch 555, loss=0.014052
Epoch 556, loss=0.014022
Epoch 557, loss=0.014014
Epoch 558, loss=0.014001
Epoch 559, loss=0.013977
Epoch 560, loss=0.013963
Epoch 561, loss=0.013930
Epoch 562, loss=0.013898
Epoch 563, loss=0.013888
Epoch 564, loss=0.013871
Epoch 565, loss=0.013836
Epoch 566, loss=0.013803
Epoch 567, loss=0.013771
Epoch 568, loss=0.013741
Epoch 569, loss=0.013710
Epoch 570, loss=0.013680
Epoch 571, loss=0.013653
Epoch 572, loss=0.013631
Epoch 573, loss=0.013602
Epoch 574, loss=0.013579
Epoch 575, loss=0.013554
Epoch 576, loss=0.013525
Epoch 577, loss=0.013498
Epoch 578, loss=0.013470
Epoch 579, loss=0.013442
Epoch 580, loss=0.013414
Epoch 581, loss=0.013388
Epoch 582, loss=0.013362
Epoch 583, loss=0.013334
Epoch 584, loss=0.013308
Epoch 585, loss=0.013283
Epoch 586, loss=0.013257
Epoch 587, loss=0.013230
Epoch 588, loss=0.013204
Epoch 589, loss=0.013180
Epoch 590, loss=0.013154
Epoch 591, loss=0.013128
Epoch 592, loss=0.013104
Epoch 593, loss=0.013079
Epoch 594, loss=0.013053
Epoch 595, loss=0.013029
Epoch 596, loss=0.013005
Epoch 597, loss=0.012980
```

```
Epoch 598, loss=0.012956
Epoch 599, loss=0.012941
Epoch 600, loss=0.012916
Epoch 600, loss=0.012916
Epoch 601, loss=0.012891
Epoch 602, loss=0.012867
Epoch 603, loss=0.012843
Epoch 604, loss=0.012818
Epoch 605, loss=0.012797
Epoch 606, loss=0.012779
Epoch 607, loss=0.012754
Epoch 608, loss=0.012730
Epoch 609, loss=0.012707
Epoch 610, loss=0.012682
Epoch 611, loss=0.012658
Epoch 612, loss=0.012636
Epoch 613, loss=0.012612
Epoch 614, loss=0.012589
Epoch 615, loss=0.012567
Epoch 616, loss=0.012544
Epoch 617, loss=0.012520
Epoch 618, loss=0.012498
Epoch 619, loss=0.012476
Epoch 620, loss=0.012453
Epoch 621, loss=0.012431
Epoch 622, loss=0.012409
Epoch 623, loss=0.012386
Epoch 624, loss=0.012364
Epoch 625, loss=0.012343
Epoch 626, loss=0.012320
Epoch 627, loss=0.012298
Epoch 628, loss=0.012277
Epoch 629, loss=0.012255
Epoch 630, loss=0.012233
Epoch 631, loss=0.012212
Epoch 632, loss=0.012191
Epoch 633, loss=0.012169
Epoch 634, loss=0.012148
Epoch 635, loss=0.012127
Epoch 636, loss=0.012106
Epoch 637, loss=0.012090
Epoch 638, loss=0.012080
Epoch 639, loss=0.012064
Epoch 640, loss=0.012042
Epoch 641, loss=0.012024
Epoch 642, loss=0.012014
Epoch 643, loss=0.011991
Epoch 644, loss=0.011968
Epoch 645, loss=0.011947
Epoch 646, loss=0.011928
Epoch 647, loss=0.011928
Epoch 648, loss=0.011904
Epoch 649, loss=0.011881
Epoch 650, loss=0.011859
Epoch 650, loss=0.011859
Epoch 651, loss=0.011837
Epoch 652, loss=0.011815
Epoch 653, loss=0.011794
Epoch 654, loss=0.011773
Epoch 655, loss=0.011752
Epoch 656, loss=0.011732
Epoch 657, loss=0.011711
Epoch 658, loss=0.011691
Epoch 659, loss=0.011670
Epoch 660, loss=0.011651
Epoch 661, loss=0.011631
Epoch 662, loss=0.011611
Epoch 663, loss=0.011597
Epoch 664, loss=0.011578
Epoch 665, loss=0.011559
Epoch 666, loss=0.011544
Epoch 667, loss=0.011529
Epoch 668, loss=0.011510
Epoch 669, loss=0.011490
Epoch 670, loss=0.011471
Epoch 671, loss=0.011451
```

```
Epoch 672, loss=0.011433
Epoch 673, loss=0.011414
Epoch 674, loss=0.011395
Epoch 675, loss=0.011377
Epoch 676, loss=0.011359
Epoch 677, loss=0.011340
Epoch 678, loss=0.011322
Epoch 679, loss=0.011304
Epoch 680, loss=0.011286
Epoch 681, loss=0.011267
Epoch 682, loss=0.011250
Epoch 683, loss=0.011232
Epoch 684, loss=0.011214
Epoch 685, loss=0.011197
Epoch 686, loss=0.011180
Epoch 687, loss=0.011162
Epoch 688, loss=0.011149
Epoch 689, loss=0.011132
Epoch 690, loss=0.011114
Epoch 691, loss=0.011097
Epoch 692, loss=0.011080
Epoch 693, loss=0.011062
Epoch 694, loss=0.011045
Epoch 695, loss=0.011029
Epoch 696, loss=0.011011
Epoch 697, loss=0.010994
Epoch 698, loss=0.010977
Epoch 699, loss=0.010961
Epoch 700, loss=0.010943
Epoch 700, loss=0.010943
Epoch 701, loss=0.010927
Epoch 702, loss=0.010910
Epoch 703, loss=0.010893
Epoch 704, loss=0.010877
Epoch 705, loss=0.010861
Epoch 706, loss=0.010844
Epoch 707, loss=0.010828
Epoch 708, loss=0.010812
Epoch 709, loss=0.010795
Epoch 710, loss=0.010779
Epoch 711, loss=0.010763
Epoch 712, loss=0.010747
Epoch 713, loss=0.010731
Epoch 714, loss=0.010716
Epoch 715, loss=0.010700
Epoch 716, loss=0.010684
Epoch 717, loss=0.010676
Epoch 718, loss=0.010663
Epoch 719, loss=0.010647
Epoch 720, loss=0.010632
Epoch 721, loss=0.010616
Epoch 722, loss=0.010600
Epoch 723, loss=0.010584
Epoch 724, loss=0.010569
Epoch 725, loss=0.010553
Epoch 726, loss=0.010538
Epoch 727, loss=0.010523
Epoch 728, loss=0.010510
Epoch 729, loss=0.010497
Epoch 730, loss=0.010482
Epoch 731, loss=0.010467
Epoch 732, loss=0.010451
Epoch 733, loss=0.010436
Epoch 734, loss=0.010421
Epoch 735, loss=0.010406
Epoch 736, loss=0.010391
Epoch 737, loss=0.010378
Epoch 738, loss=0.010369
Epoch 739, loss=0.010354
Epoch 740, loss=0.010338
Epoch 741, loss=0.010324
Epoch 742, loss=0.010309
Epoch 743, loss=0.010294
Epoch 744, loss=0.010280
Epoch 745, loss=0.010272
Epoch 746, loss=0.010257
```

```
Epoch 747, loss=0.010242
Epoch 748, loss=0.010227
Epoch 749, loss=0.010213
Epoch 750, loss=0.010198
Epoch 750, loss=0.010198
Epoch 751, loss=0.010183
Epoch 752, loss=0.010169
Epoch 753, loss=0.010159
Epoch 754, loss=0.010145
Epoch 755, loss=0.010131
Epoch 756, loss=0.010116
Epoch 757, loss=0.010102
Epoch 758, loss=0.010088
Epoch 759, loss=0.010080
Epoch 760, loss=0.010072
Epoch 761, loss=0.010057
Epoch 762, loss=0.010043
Epoch 763, loss=0.010028
Epoch 764, loss=0.010013
Epoch 765, loss=0.009999
Epoch 766, loss=0.009985
Epoch 767, loss=0.009971
Epoch 768, loss=0.009957
Epoch 769, loss=0.009943
Epoch 770, loss=0.009929
Epoch 771, loss=0.009915
Epoch 772, loss=0.009902
Epoch 773, loss=0.009888
Epoch 774, loss=0.009874
Epoch 775, loss=0.009861
Epoch 776, loss=0.009847
Epoch 777, loss=0.009834
Epoch 778, loss=0.009821
Epoch 779, loss=0.009807
Epoch 780, loss=0.009794
Epoch 781, loss=0.009781
Epoch 782, loss=0.009768
Epoch 783, loss=0.009755
Epoch 784, loss=0.009741
Epoch 785, loss=0.009728
Epoch 786, loss=0.009716
Epoch 787, loss=0.009702
Epoch 788, loss=0.009690
Epoch 789, loss=0.009681
Epoch 790, loss=0.009668
Epoch 791, loss=0.009654
Epoch 792, loss=0.009642
Epoch 793, loss=0.009629
Epoch 794, loss=0.009616
Epoch 795, loss=0.009603
Epoch 796, loss=0.009591
Epoch 797, loss=0.009578
Epoch 798, loss=0.009565
Epoch 799, loss=0.009553
Epoch 800, loss=0.009540
Epoch 800, loss=0.009540
Epoch 801, loss=0.009528
Epoch 802, loss=0.009516
Epoch 803, loss=0.009503
Epoch 804, loss=0.009491
Epoch 805, loss=0.009478
Epoch 806, loss=0.009468
Epoch 807, loss=0.009462
Epoch 808, loss=0.009450
Epoch 809, loss=0.009437
Epoch 810, loss=0.009425
Epoch 811, loss=0.009412
Epoch 812, loss=0.009400
Epoch 813, loss=0.009388
Epoch 814, loss=0.009375
Epoch 815, loss=0.009363
Epoch 816, loss=0.009351
Epoch 817, loss=0.009340
Epoch 818, loss=0.009331
Epoch 819, loss=0.009319
Epoch 820, loss=0.009307
```

```
Epoch 821, loss=0.009295
Epoch 822, loss=0.009289
Epoch 823, loss=0.009277
Epoch 824, loss=0.009264
Epoch 825, loss=0.009252
Epoch 826, loss=0.009240
Epoch 827, loss=0.009228
Epoch 828, loss=0.009216
Epoch 829, loss=0.009205
Epoch 830, loss=0.009193
Epoch 831, loss=0.009181
Epoch 832, loss=0.009169
Epoch 833, loss=0.009158
Epoch 834, loss=0.009146
Epoch 835, loss=0.009134
Epoch 836, loss=0.009123
Epoch 837, loss=0.009111
Epoch 838, loss=0.009100
Epoch 839, loss=0.009089
Epoch 840, loss=0.009077
Epoch 841, loss=0.009066
Epoch 842, loss=0.009055
Epoch 843, loss=0.009044
Epoch 844, loss=0.009032
Epoch 845, loss=0.009021
Epoch 846, loss=0.009011
Epoch 847, loss=0.009004
Epoch 848, loss=0.008992
Epoch 849, loss=0.008981
Epoch 850, loss=0.008970
Epoch 850, loss=0.008970
Epoch 851, loss=0.008959
Epoch 852, loss=0.008948
Epoch 853, loss=0.008937
Epoch 854, loss=0.008926
Epoch 855, loss=0.008914
Epoch 856, loss=0.008904
Epoch 857, loss=0.008895
Epoch 858, loss=0.008885
Epoch 859, loss=0.008874
Epoch 860, loss=0.008863
Epoch 861, loss=0.008852
Epoch 862, loss=0.008841
Epoch 863, loss=0.008831
Epoch 864, loss=0.008820
Epoch 865, loss=0.008809
Epoch 866, loss=0.008799
Epoch 867, loss=0.008788
Epoch 868, loss=0.008777
Epoch 869, loss=0.008767
Epoch 870, loss=0.008756
Epoch 871, loss=0.008745
Epoch 872, loss=0.008735
Epoch 873, loss=0.008725
Epoch 874, loss=0.008714
Epoch 875, loss=0.008704
Epoch 876, loss=0.008694
Epoch 877, loss=0.008683
Epoch 878, loss=0.008673
Epoch 879, loss=0.008663
Epoch 880, loss=0.008652
Epoch 881, loss=0.008642
Epoch 882, loss=0.008632
Epoch 883, loss=0.008622
Epoch 884, loss=0.008611
Epoch 885, loss=0.008601
Epoch 886, loss=0.008591
Epoch 887, loss=0.008581
Epoch 888, loss=0.008571
Epoch 889, loss=0.008564
Epoch 890, loss=0.008553
Epoch 891, loss=0.008543
Epoch 892, loss=0.008534
Epoch 893, loss=0.008523
Epoch 894, loss=0.008513
Epoch 895, loss=0.008504
```

```
Epoch 896, loss=0.008494
Epoch 897, loss=0.008483
Epoch 898, loss=0.008474
Epoch 899, loss=0.008464
Epoch 900, loss=0.008454
Epoch 900, loss=0.008454
Epoch 901, loss=0.008448
Epoch 902, loss=0.008441
Epoch 903, loss=0.008431
Epoch 904, loss=0.008422
Epoch 905, loss=0.008413
Epoch 906, loss=0.008403
Epoch 907, loss=0.008393
Epoch 908, loss=0.008383
Epoch 909, loss=0.008373
Epoch 910, loss=0.008368
Epoch 911, loss=0.008364
Epoch 912, loss=0.008358
Epoch 913, loss=0.008348
Epoch 914, loss=0.008337
Epoch 915, loss=0.008327
Epoch 916, loss=0.008317
Epoch 917, loss=0.008307
Epoch 918, loss=0.008297
Epoch 919, loss=0.008287
Epoch 920, loss=0.008277
Epoch 921, loss=0.008267
Epoch 922, loss=0.008258
Epoch 923, loss=0.008248
Epoch 924, loss=0.008238
Epoch 925, loss=0.008229
Epoch 926, loss=0.008219
Epoch 927, loss=0.008210
Epoch 928, loss=0.008201
Epoch 929, loss=0.008194
Epoch 930, loss=0.008185
Epoch 931, loss=0.008175
Epoch 932, loss=0.008166
Epoch 933, loss=0.008157
Epoch 934, loss=0.008147
Epoch 935, loss=0.008138
Epoch 936, loss=0.008129
Epoch 937, loss=0.008120
Epoch 938, loss=0.008110
Epoch 939, loss=0.008102
Epoch 940, loss=0.008092
Epoch 941, loss=0.008083
Epoch 942, loss=0.008074
Epoch 943, loss=0.008065
Epoch 944, loss=0.008056
Epoch 945, loss=0.008047
Epoch 946, loss=0.008039
Epoch 947, loss=0.008030
Epoch 948, loss=0.008021
Epoch 949, loss=0.008012
Epoch 950, loss=0.008003
Epoch 950, loss=0.008003
Epoch 951, loss=0.007994
Epoch 952, loss=0.007985
Epoch 953, loss=0.007977
Epoch 954, loss=0.007968
Epoch 955, loss=0.007959
Epoch 956, loss=0.007951
Epoch 957, loss=0.007943
Epoch 958, loss=0.007935
Epoch 959, loss=0.007927
Epoch 960, loss=0.007918
Epoch 961, loss=0.007909
Epoch 962, loss=0.007901
Epoch 963, loss=0.007892
Epoch 964, loss=0.007884
Epoch 965, loss=0.007875
Epoch 966, loss=0.007867
Epoch 967, loss=0.007858
Epoch 968, loss=0.007850
Epoch 969, loss=0.007844
```

```
Epoch 970, loss=0.007836
Epoch 971, loss=0.007828
Epoch 972, loss=0.007820
Epoch 973, loss=0.007811
Epoch 974, loss=0.007802
Epoch 975, loss=0.007794
Epoch 976, loss=0.007786
Epoch 977, loss=0.007777
Epoch 978, loss=0.007769
Epoch 979, loss=0.007761
Epoch 980, loss=0.007752
Epoch 981, loss=0.007744
Epoch 982, loss=0.007736
Epoch 983, loss=0.007728
Epoch 984, loss=0.007719
Epoch 985, loss=0.007711
Epoch 986, loss=0.007703
Epoch 987, loss=0.007695
Epoch 988, loss=0.007687
Epoch 989, loss=0.007679
Epoch 990, loss=0.007671
Epoch 991, loss=0.007662
Epoch 992, loss=0.007655
Epoch 993, loss=0.007646
Epoch 994, loss=0.007638
Epoch 995, loss=0.007630
Epoch 996, loss=0.007622
Epoch 997, loss=0.007614
Epoch 998, loss=0.007606
Epoch 999, loss=0.007601
Epoch 1000, loss=0.007593
Epoch 1000, loss=0.007593
```

In [ ]:

```
# Test your predictions. The predictions must match the labels
print(net.predict(X))
print(y)
```

```
[2 1 0 1 2 0 0 2 0 0]
[2 1 0 1 2 0 0 2 0 0]

[2 1 0 1 2 0 0 2 0 0]
```

In [ ]:

```
# You should be able to achieve a training loss of less than 0.02 (10%)
print("Final training loss", epoch_loss[-1])
```

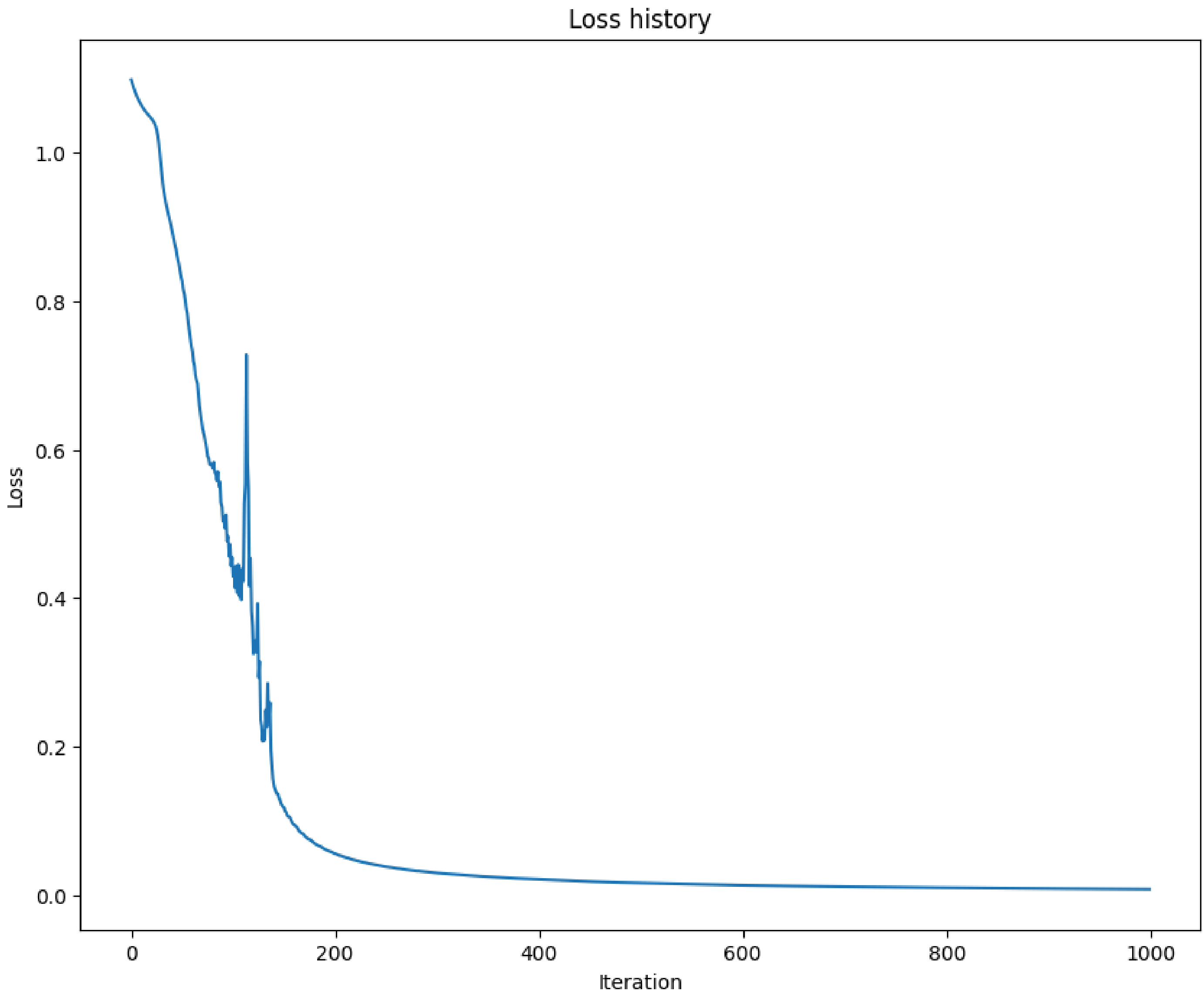
```
Final training loss 0.007593419801731252
```

In [ ]:

```
# Plot the training loss curve. The loss in the curve should be decreasing (20%)
plt.plot(epoch_loss)
plt.title("Loss history")
plt.xlabel("Iteration")
plt.ylabel("Loss")
```

Out[ ]:

```
Text(0, 0.5, 'Loss')
```


Loss history
Loss
Iteration
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

ECE 285 Assignment 1: Classification using Neural Network

Now that you have developed and tested your model on the toy dataset set. It's time to get down and get dirty with a standard dataset such as cifar10. At this point, you will be using the provided training data to tune the hyper-parameters of your network such that it works with cifar10 for the task of multi-class classification.

Important: Recall that now we have non-linear decision boundaries, thus we do not need to do one vs all classification. We learn a single non-linear decision boundary instead. Our non-linear boundaries (thanks to relu non-linearity) will take care of differentiating between all the classes

TO SUBMIT: PDF of this notebook with all the required outputs and answers.

In [ ]:

```
# Prepare Packages
import numpy as np
import matplotlib.pyplot as plt

from ece285.utils.data_processing import get_cifar10_data
from ece285.utils.evaluation import get_classification_accuracy


%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0)  # set default size of plots

# For auto-reloading external modules
# See http://stackoverflow.com/questions/1907993/autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2

# Use a subset of CIFAR10 for the assignment
dataset = get_cifar10_data(
    subset_train=5000,
    subset_val=250,
    subset_test=500,
)

print(dataset.keys())
print("Training Set Data  Shape: ", dataset["x_train"].shape)
print("Training Set Label Shape: ", dataset["y_train"].shape)
print("Validation Set Data  Shape: ", dataset["x_val"].shape)
print("Validation Set Label Shape: ", dataset["y_val"].shape)
print("Test Set Data  Shape: ", dataset["x_test"].shape)
print("Test Set Label Shape: ", dataset["y_test"].shape)
```

```
dict_keys(['x_train', 'y_train', 'x_val', 'y_val', 'x_test', 'y_test'])
Training Set Data  Shape:  (5000, 3072)
Training Set Label Shape:  (5000,)
Validation Set Data  Shape:  (250, 3072)
Validation Set Label Shape:  (250,)
Test Set Data  Shape:  (500, 3072)
Test Set Label Shape:  (500,)
```

In [ ]:

```
x_train = dataset["x_train"]
y_train = dataset["y_train"]
x_val = dataset["x_val"]
y_val = dataset["y_val"]
x_test = dataset["x_test"]
y_test = dataset["y_test"]
```

In [ ]:

```
# Import more utilies and the layers you have implemented
from ece285.layers.sequential import Sequential
from ece285.layers.linear import Linear
from ece285.layers.relu import ReLU
from ece285.layers.softmax import Softmax
from ece285.layers.loss_func import CrossEntropyLoss
from ece285.utils.optimizer import SGD
from ece285.utils.dataset import DataLoader
from ece285.utils.trainer import Trainer
```

Visualize some examples from the dataset.

In [ ]:
```
# We show a few examples of training images from each class.
classes = [
    "airplane",
    "automobile",
    "bird",
    "cat",
    "deer",
    "dog",
    "frog",
    "horse",
    "ship",
]
samples_per_class = 7


def visualize_data(dataset, classes, samples_per_class):
    num_classes = len(classes)
    for y, cls in enumerate(classes):
        idxs = np.flatnonzero(y_train == y)
        idxs = np.random.choice(idxs, samples_per_class, replace=False)
        for i, idx in enumerate(idxs):
            plt_idx = i * num_classes + y + 1
            plt.subplot(samples_per_class, num_classes, plt_idx)
            plt.imshow(dataset[idx])
            plt.axis("off")
            if i == 0:
                plt.title(cls)
    plt.show()


# Visualize the first 10 classes
visualize_data(
    x_train.reshape(5000, 3, 32, 32).transpose(0, 2, 3, 1),
    classes,
    samples_per_class,
)
```

Initialize the model

In [ ]:
```
input_size = 3072
hidden_size = 100  # Hidden layer size (Hyper-parameter)
num_classes = 10  # Output

# For a default setting we use the same model we used for the toy dataset.
# This tells you the power of a 2 layered Neural Network. Recall the Universal Approximation Theorem.
# A 2 layer neural network with non-linearities can approximate any function, given large enough hidden layer
def init_model():
    # np.random.seed(0) # No need to fix the seed here
    l1 = Linear(input_size, hidden_size)
    l2 = Linear(hidden_size, num_classes)

    r1 = ReLU()
    softmax = Softmax()
    return Sequential([l1, r1, l2, softmax])
```

In [ ]:
```
# Initialize the dataset with the dataloader class
dataset = DataLoader(x_train, y_train, x_val, y_val, x_test, y_test)
net = init_model()
optim = SGD(net, lr=0.01, weight_decay=0.01)
loss_func = CrossEntropyLoss()
epoch = 200  # (Hyper-parameter)
batch_size = 200  # (Reduce the batch size if your computer is unable to handle it)
```

In [ ]:
```
# Initialize the trainer class by passing the above modules
trainer = Trainer(
    dataset, optim, net, loss_func, epoch, batch_size, validate_interval=3
)
```

In [ ]:
```
# Call the trainer function we have already implemented for you. This trains the model for the given
# hyper-parameters. It follows the same procedure as in the last ipython notebook you used for the toy-dataset
```

```
train_error, validation_accuracy = trainer.train()
```

```
Epoch 0 Average Loss: 2.302541
Validate Acc: 0.084 at epoch 0
Epoch 1 Average Loss: 2.302365
Epoch 2 Average Loss: 2.302152
Epoch 3 Average Loss: 2.301855
Validate Acc: 0.100 at epoch 3
Epoch 4 Average Loss: 2.301441
Epoch 5 Average Loss: 2.300837
Epoch 6 Average Loss: 2.299978
Validate Acc: 0.100 at epoch 6
Epoch 7 Average Loss: 2.298822
Epoch 8 Average Loss: 2.297318
Epoch 9 Average Loss: 2.295512
Validate Acc: 0.100 at epoch 9
Epoch 10 Average Loss: 2.293362
Epoch 11 Average Loss: 2.290846
Epoch 12 Average Loss: 2.287781
Validate Acc: 0.084 at epoch 12
Epoch 13 Average Loss: 2.283847
Epoch 14 Average Loss: 2.278884
Epoch 15 Average Loss: 2.272635
Validate Acc: 0.096 at epoch 15
Epoch 16 Average Loss: 2.265677
Epoch 17 Average Loss: 2.258239
Epoch 18 Average Loss: 2.250647
Validate Acc: 0.096 at epoch 18
Epoch 19 Average Loss: 2.242969
Epoch 20 Average Loss: 2.235748
Epoch 21 Average Loss: 2.228497
Validate Acc: 0.112 at epoch 21
Epoch 22 Average Loss: 2.221814
Epoch 23 Average Loss: 2.215869
Epoch 24 Average Loss: 2.209942
Validate Acc: 0.124 at epoch 24
Epoch 25 Average Loss: 2.204750
Epoch 26 Average Loss: 2.199639
Epoch 27 Average Loss: 2.195551
Validate Acc: 0.132 at epoch 27
Epoch 28 Average Loss: 2.191180
Epoch 29 Average Loss: 2.187068
Epoch 30 Average Loss: 2.183377
Validate Acc: 0.140 at epoch 30
Epoch 31 Average Loss: 2.180063
Epoch 32 Average Loss: 2.176303
Epoch 33 Average Loss: 2.173175
Validate Acc: 0.144 at epoch 33
Epoch 34 Average Loss: 2.170143
Epoch 35 Average Loss: 2.167357
Epoch 36 Average Loss: 2.164459
Validate Acc: 0.144 at epoch 36
Epoch 37 Average Loss: 2.161905
Epoch 38 Average Loss: 2.159285
Epoch 39 Average Loss: 2.156704
Validate Acc: 0.144 at epoch 39
Epoch 40 Average Loss: 2.154519
Epoch 41 Average Loss: 2.152186
Epoch 42 Average Loss: 2.150165
Validate Acc: 0.152 at epoch 42
Epoch 43 Average Loss: 2.148013
Epoch 44 Average Loss: 2.146433
Epoch 45 Average Loss: 2.144005
Validate Acc: 0.152 at epoch 45
Epoch 46 Average Loss: 2.142087
Epoch 47 Average Loss: 2.140532
Epoch 48 Average Loss: 2.138592
Validate Acc: 0.156 at epoch 48
Epoch 49 Average Loss: 2.136629
Epoch 50 Average Loss: 2.135073
Epoch 51 Average Loss: 2.133314
Validate Acc: 0.156 at epoch 51
Epoch 52 Average Loss: 2.131737
Epoch 53 Average Loss: 2.130350
Epoch 54 Average Loss: 2.128908
Validate Acc: 0.148 at epoch 54
Epoch 55 Average Loss: 2.127337
Epoch 56 Average Loss: 2.125973
```

```
Epoch 57 Average Loss: 2.123959
Validate Acc: 0.160 at epoch 57
Epoch 58 Average Loss: 2.122621
Epoch 59 Average Loss: 2.120998
Epoch 60 Average Loss: 2.119617
Validate Acc: 0.160 at epoch 60
Epoch 61 Average Loss: 2.117845
Epoch 62 Average Loss: 2.116030
Epoch 63 Average Loss: 2.114214
Validate Acc: 0.172 at epoch 63
Epoch 64 Average Loss: 2.112813
Epoch 65 Average Loss: 2.110930
Epoch 66 Average Loss: 2.109183
Validate Acc: 0.172 at epoch 66
Epoch 67 Average Loss: 2.107065
Epoch 68 Average Loss: 2.104692
Epoch 69 Average Loss: 2.102414
Validate Acc: 0.180 at epoch 69
Epoch 70 Average Loss: 2.100362
Epoch 71 Average Loss: 2.097985
Epoch 72 Average Loss: 2.095113
Validate Acc: 0.180 at epoch 72
Epoch 73 Average Loss: 2.092905
Epoch 74 Average Loss: 2.090000
Epoch 75 Average Loss: 2.086763
Validate Acc: 0.224 at epoch 75
Epoch 76 Average Loss: 2.083981
Epoch 77 Average Loss: 2.080646
Epoch 78 Average Loss: 2.077791
Validate Acc: 0.220 at epoch 78
Epoch 79 Average Loss: 2.074392
Epoch 80 Average Loss: 2.071346
Epoch 81 Average Loss: 2.068268
Validate Acc: 0.220 at epoch 81
Epoch 82 Average Loss: 2.064981
Epoch 83 Average Loss: 2.061380
Epoch 84 Average Loss: 2.058402
Validate Acc: 0.228 at epoch 84
Epoch 85 Average Loss: 2.055538
Epoch 86 Average Loss: 2.052580
Epoch 87 Average Loss: 2.049169
Validate Acc: 0.236 at epoch 87
Epoch 88 Average Loss: 2.046946
Epoch 89 Average Loss: 2.044161
Epoch 90 Average Loss: 2.041275
Validate Acc: 0.252 at epoch 90
Epoch 91 Average Loss: 2.038576
Epoch 92 Average Loss: 2.035442
Epoch 93 Average Loss: 2.033913
Validate Acc: 0.260 at epoch 93
Epoch 94 Average Loss: 2.031616
Epoch 95 Average Loss: 2.029026
Epoch 96 Average Loss: 2.026542
Validate Acc: 0.264 at epoch 96
Epoch 97 Average Loss: 2.024428
Epoch 98 Average Loss: 2.022678
Epoch 99 Average Loss: 2.020123
Validate Acc: 0.260 at epoch 99
Epoch 100 Average Loss: 2.018194
Epoch 101 Average Loss: 2.016189
Epoch 102 Average Loss: 2.014272
Validate Acc: 0.268 at epoch 102
Epoch 103 Average Loss: 2.012021
Epoch 104 Average Loss: 2.010419
Epoch 105 Average Loss: 2.008994
Validate Acc: 0.276 at epoch 105
Epoch 106 Average Loss: 2.006911
Epoch 107 Average Loss: 2.005666
Epoch 108 Average Loss: 2.003779
Validate Acc: 0.276 at epoch 108
Epoch 109 Average Loss: 2.001665
Epoch 110 Average Loss: 2.000210
Epoch 111 Average Loss: 1.998838
Validate Acc: 0.280 at epoch 111
Epoch 112 Average Loss: 1.997198
Epoch 113 Average Loss: 1.995064
```

```
Epoch 114 Average Loss: 1.993435
Validate Acc: 0.276 at epoch 114
Epoch 115 Average Loss: 1.991765
Epoch 116 Average Loss: 1.990279
Epoch 117 Average Loss: 1.988810
Validate Acc: 0.260 at epoch 117
Epoch 118 Average Loss: 1.987550
Epoch 119 Average Loss: 1.985893
Epoch 120 Average Loss: 1.984558
Validate Acc: 0.272 at epoch 120
Epoch 121 Average Loss: 1.982592
Epoch 122 Average Loss: 1.980815
Epoch 123 Average Loss: 1.979830
Validate Acc: 0.284 at epoch 123
Epoch 124 Average Loss: 1.979492
Epoch 125 Average Loss: 1.976958
Epoch 126 Average Loss: 1.975790
Validate Acc: 0.288 at epoch 126
Epoch 127 Average Loss: 1.973922
Epoch 128 Average Loss: 1.972704
Epoch 129 Average Loss: 1.971068
Validate Acc: 0.292 at epoch 129
Epoch 130 Average Loss: 1.969227
Epoch 131 Average Loss: 1.966909
Epoch 132 Average Loss: 1.967189
Validate Acc: 0.288 at epoch 132
Epoch 133 Average Loss: 1.965416
Epoch 134 Average Loss: 1.962349
Epoch 135 Average Loss: 1.961825
Validate Acc: 0.296 at epoch 135
Epoch 136 Average Loss: 1.959486
Epoch 137 Average Loss: 1.957134
Epoch 138 Average Loss: 1.955703
Validate Acc: 0.308 at epoch 138
Epoch 139 Average Loss: 1.953794
Epoch 140 Average Loss: 1.952608
Epoch 141 Average Loss: 1.949948
Validate Acc: 0.296 at epoch 141
Epoch 142 Average Loss: 1.948022
Epoch 143 Average Loss: 1.946442
Epoch 144 Average Loss: 1.944107
Validate Acc: 0.296 at epoch 144
Epoch 145 Average Loss: 1.941448
Epoch 146 Average Loss: 1.938762
Epoch 147 Average Loss: 1.935934
Validate Acc: 0.292 at epoch 147
Epoch 148 Average Loss: 1.932782
Epoch 149 Average Loss: 1.930702
Epoch 150 Average Loss: 1.927413
Validate Acc: 0.296 at epoch 150
Epoch 151 Average Loss: 1.924571
Epoch 152 Average Loss: 1.921181
Epoch 153 Average Loss: 1.917878
Validate Acc: 0.304 at epoch 153
Epoch 154 Average Loss: 1.915468
Epoch 155 Average Loss: 1.912472
Epoch 156 Average Loss: 1.909902
Validate Acc: 0.300 at epoch 156
Epoch 157 Average Loss: 1.905868
Epoch 158 Average Loss: 1.904474
Epoch 159 Average Loss: 1.901745
Validate Acc: 0.312 at epoch 159
Epoch 160 Average Loss: 1.900239
Epoch 161 Average Loss: 1.896641
Epoch 162 Average Loss: 1.894475
Validate Acc: 0.316 at epoch 162
Epoch 163 Average Loss: 1.892287
Epoch 164 Average Loss: 1.889980
Epoch 165 Average Loss: 1.887225
Validate Acc: 0.304 at epoch 165
Epoch 166 Average Loss: 1.886012
Epoch 167 Average Loss: 1.883562
Epoch 168 Average Loss: 1.881281
Validate Acc: 0.304 at epoch 168
Epoch 169 Average Loss: 1.879884
Epoch 170 Average Loss: 1.878462
```

```
Epoch 171 Average Loss: 1.874707
Validate Acc: 0.308 at epoch 171
Epoch 172 Average Loss: 1.873375
Epoch 173 Average Loss: 1.870900
Epoch 174 Average Loss: 1.868673
Validate Acc: 0.300 at epoch 174
Epoch 175 Average Loss: 1.866253
Epoch 176 Average Loss: 1.864626
Epoch 177 Average Loss: 1.862189
Validate Acc: 0.296 at epoch 177
Epoch 178 Average Loss: 1.860192
Epoch 179 Average Loss: 1.859438
Epoch 180 Average Loss: 1.857224
Validate Acc: 0.300 at epoch 180
Epoch 181 Average Loss: 1.855129
Epoch 182 Average Loss: 1.853133
Epoch 183 Average Loss: 1.853107
Validate Acc: 0.320 at epoch 183
Epoch 184 Average Loss: 1.848501
Epoch 185 Average Loss: 1.848029
Epoch 186 Average Loss: 1.843732
Validate Acc: 0.300 at epoch 186
Epoch 187 Average Loss: 1.842956
Epoch 188 Average Loss: 1.841446
Epoch 189 Average Loss: 1.839246
Validate Acc: 0.296 at epoch 189
Epoch 190 Average Loss: 1.836272
Epoch 191 Average Loss: 1.833739
Epoch 192 Average Loss: 1.831773
Validate Acc: 0.316 at epoch 192
Epoch 193 Average Loss: 1.829897
Epoch 194 Average Loss: 1.829503
Epoch 195 Average Loss: 1.827165
Validate Acc: 0.304 at epoch 195
Epoch 196 Average Loss: 1.824672
Epoch 197 Average Loss: 1.822816
Epoch 198 Average Loss: 1.819579
Validate Acc: 0.308 at epoch 198
Epoch 199 Average Loss: 1.816834
```

Print the training and validation accuracies for the default hyper-parameters provided

In [ ]:

```python
from ece285.utils.evaluation import get_classification_accuracy

out_train = net.predict(x_train)
acc = get_classification_accuracy(out_train, y_train)
print("Training acc: ", acc)
out_val = net.predict(x_val)
acc = get_classification_accuracy(out_val, y_val)
print("Validation acc: ", acc)
```

```
Training acc:  0.3414
Validation acc:  0.32
```

Debug the training

With the default parameters we provided above, you should get a validation accuracy of around ~0.2 on the validation set. This isn't very good.

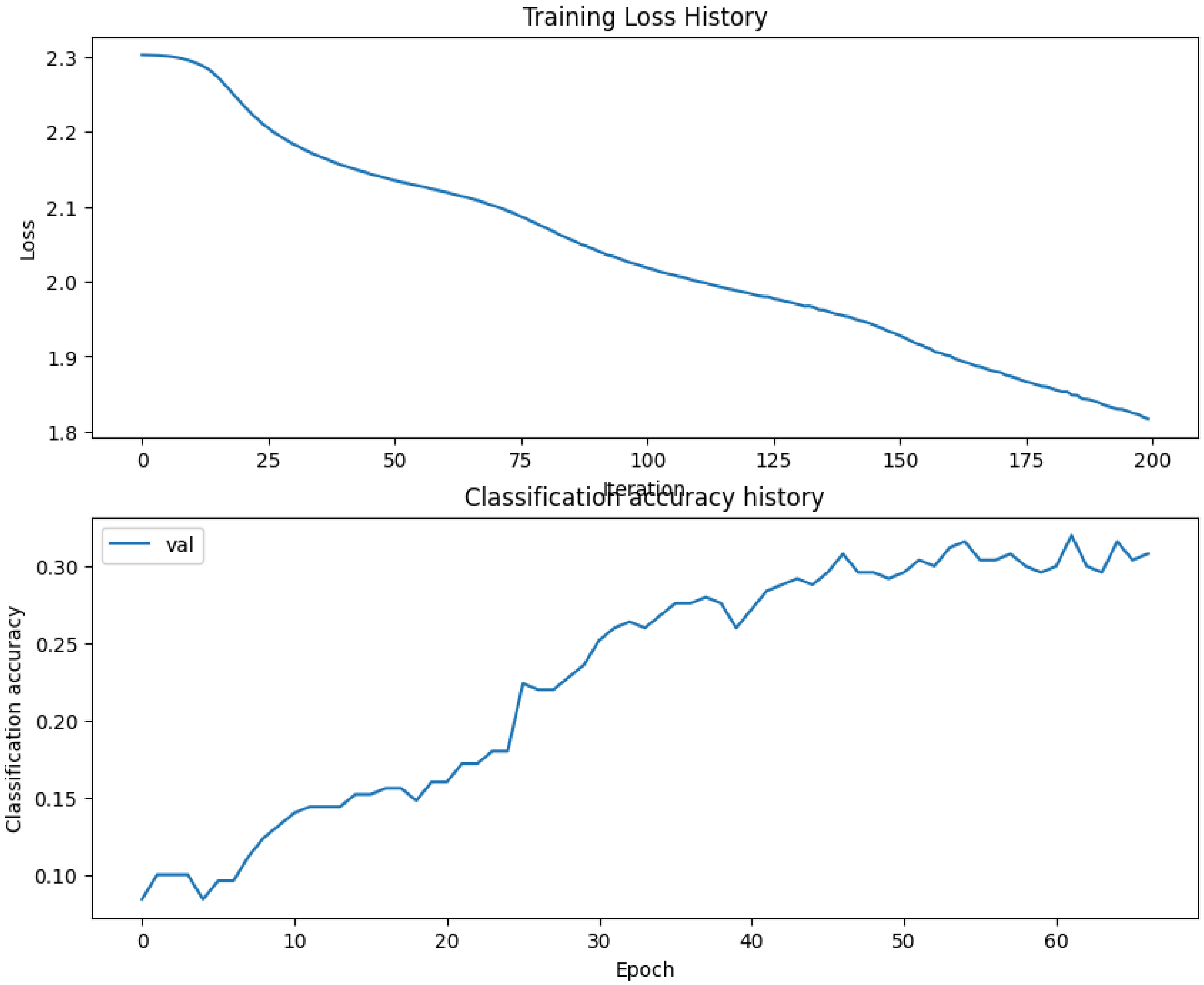
One strategy for getting insight into what's wrong is to plot the training loss function and the validation accuracies during optimization.

Another strategy is to visualize the weights that were learned in the first layer of the network. In most neural networks trained on visual data, the first layer weights typically show some visible structure when visualized.

In [ ]:

```python
# Plot the training loss function and validation accuracies
plt.subplot(2, 1, 1)
plt.plot(train_error)
plt.title("Training Loss History")
plt.xlabel("Iteration")
plt.ylabel("Loss")
```

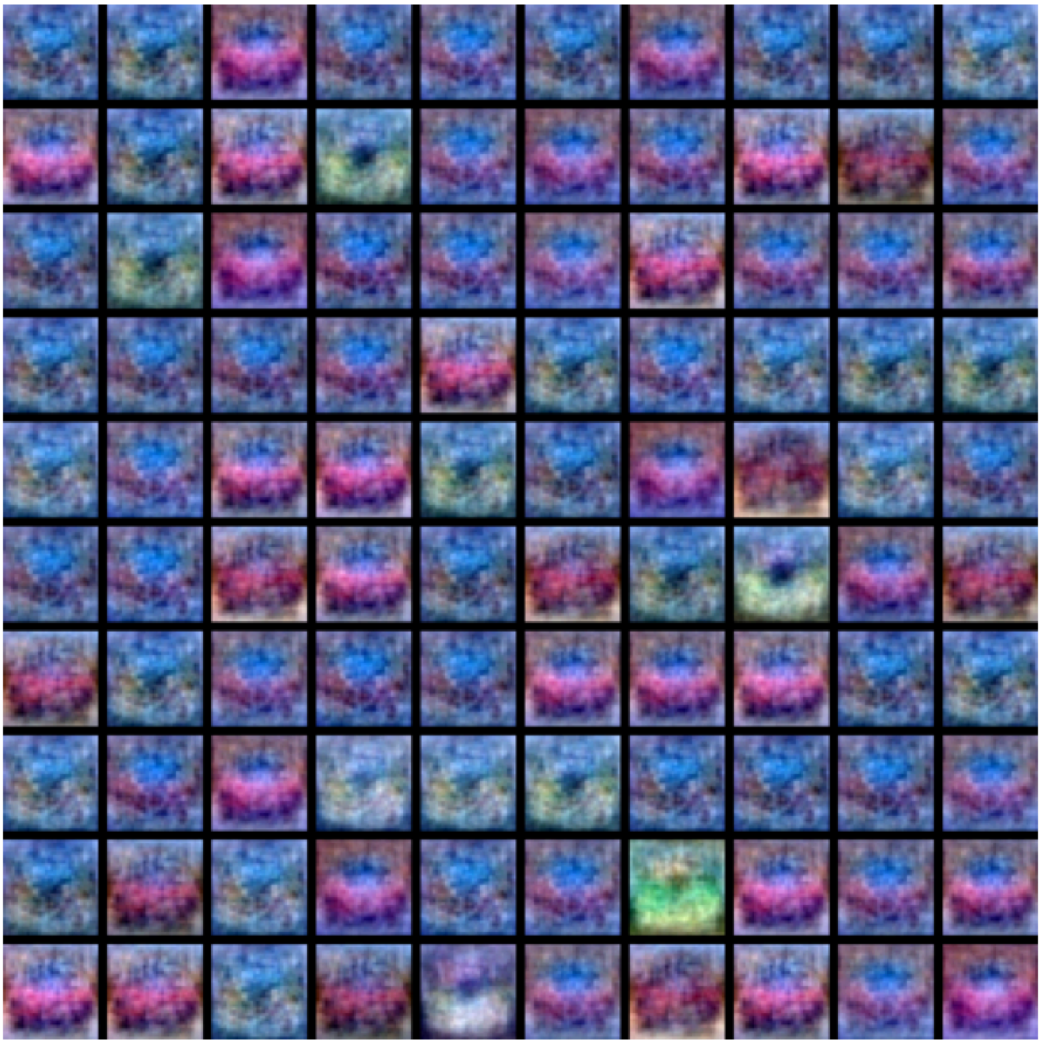
```
plt.subplot(2, 1, 2)
# plt.plot(stats['train_acc_history'], label='train')
plt.plot(validation_accuracy, label="val")
plt.title("Classification accuracy history")
plt.xlabel("Epoch")
plt.ylabel("Classification accuracy")
plt.legend()
plt.show()
```



In [ ]:

```
from ece285.utils.vis_utils import visualize_grid

# Credits: http://cs231n.stanford.edu/

# Visualize the weights of the network

def show_net_weights(net):
    W1 = net._modules[0].parameters[0]
    W1 = W1.reshape(3, 32, 32, -1).transpose(3, 1, 2, 0)
    print(W1.shape)
    plt.imshow(visualize_grid(W1, padding=3).astype("uint8"))
    plt.gca().axis("off")
    plt.show()


show_net_weights(net)
```

```
(100, 32, 32, 3)
```



Tune your hyperparameters (50%)

What's wrong?. Looking at the visualizations above, we see that the loss is decreasing more or less linearly, which seems to suggest that the learning rate may be too low. Moreover, there is no gap between the training and validation accuracy, suggesting that the model we used has low capacity, and that we should increase its size. On the other hand, with a very large model we would expect to see more overfitting, which would manifest itself as a very large gap between the training and validation accuracy.

Tuning. Tuning the hyperparameters and developing intuition for how they affect the final performance is a large part of using Neural Networks, so we want you to get a lot of practice. Below, you should experiment with different values of the various hyperparameters, including hidden layer size, learning rate, numer of training epochs, and regularization strength.

Approximate results. You should be aim to achieve a classification accuracy of greater than 40% on the validation set. Our best network gets over 40% on the validation set.

Experiment: You goal in this exercise is to get as good of a result on cifar10 as you can (40% could serve as a reference), with a fully-connected Neural Network.

Explain your hyperparameter tuning process below.

Your Answer:

In [ ]:
```
#################################################################################
# TODO: Tune hyperparameters using the validation set. Store your best trained  #
# model hyperparams in best_net.                                                #
#                                                                               #
# To help debug your network, it may help to use visualizations similar to the  #
# ones we used above; these visualizations will have significant qualitative    #
# differences from the ones we saw above for the poorly tuned network.          #
#                                                                               #
```

```python
# You are now free to test different combinations of hyperparameters to build   #
# various models and test them according to the above plots and visualization   #

import itertools
import time
import matplotlib.pyplot as plt


# Define the range of hyperparameters to search over
lr_range = [0.02]
weight_decay_range = [0.005]
hidden_size_range = [300]
epochs_range = [500]
batch_size = 200

results = []

# Perform grid search
for lr, weight_decay, hidden_size, epochs in itertools.product(lr_range, weight_decay_range, hidden_size_range, epo
    # Initialize the model with current hyperparameters
    def get_model(hidden_size):
        np.random.seed(1)
        l1 = Linear(input_size, hidden_size)
        l2 = Linear(hidden_size, num_classes)

        r1 = ReLU()
        softmax = Softmax()
        return Sequential([l1, r1, l2, softmax])

    net2 = get_model(hidden_size)

    # Initialize the optimizer
    optim = SGD(net2, lr=lr, weight_decay=weight_decay)

    # Initialize the trainer
    trainer = Trainer(
        dataset, optim, net2, loss_func, epochs, batch_size, validate_interval=1
    )

    # Train the model
    start_time = time.time()
    train_error, validation_accuracy = trainer.train()
    end_time = time.time()
    training_time = end_time - start_time

    # Save results
    results.append({
        'lr': lr,
        'weight_decay': weight_decay,
        'hidden_size': hidden_size,
        'epochs': epochs,
        'batch_size': batch_size,
        'accuracy': validation_accuracy[-1],
        'loss': train_error[-1],  # Final training error
        'training_time': training_time
    })

best_net = net2
```

```
Epoch 0 Average Loss: 2.302287
Validate Acc: 0.100 at epoch 0
Epoch 1 Average Loss: 2.301260
Validate Acc: 0.104 at epoch 1
Epoch 2 Average Loss: 2.299623
Validate Acc: 0.100 at epoch 2
Epoch 3 Average Loss: 2.297076
Validate Acc: 0.092 at epoch 3
Epoch 4 Average Loss: 2.293505
Validate Acc: 0.084 at epoch 4
Epoch 5 Average Loss: 2.288444
Validate Acc: 0.084 at epoch 5
Epoch 6 Average Loss: 2.280095
Validate Acc: 0.096 at epoch 6
Epoch 7 Average Loss: 2.267249
Validate Acc: 0.096 at epoch 7
Epoch 8 Average Loss: 2.252320
Validate Acc: 0.104 at epoch 8
Epoch 9 Average Loss: 2.236878
Validate Acc: 0.104 at epoch 9
Epoch 10 Average Loss: 2.222986
Validate Acc: 0.124 at epoch 10
Epoch 11 Average Loss: 2.210983
Validate Acc: 0.124 at epoch 11
Epoch 12 Average Loss: 2.200245
Validate Acc: 0.128 at epoch 12
Epoch 13 Average Loss: 2.191528
Validate Acc: 0.132 at epoch 13
Epoch 14 Average Loss: 2.183409
Validate Acc: 0.136 at epoch 14
Epoch 15 Average Loss: 2.176750
Validate Acc: 0.140 at epoch 15
Epoch 16 Average Loss: 2.169953
Validate Acc: 0.144 at epoch 16
Epoch 17 Average Loss: 2.164328
Validate Acc: 0.140 at epoch 17
Epoch 18 Average Loss: 2.158772
Validate Acc: 0.148 at epoch 18
Epoch 19 Average Loss: 2.153917
Validate Acc: 0.144 at epoch 19
Epoch 20 Average Loss: 2.149670
Validate Acc: 0.156 at epoch 20
Epoch 21 Average Loss: 2.145634
Validate Acc: 0.144 at epoch 21
Epoch 22 Average Loss: 2.141000
Validate Acc: 0.148 at epoch 22
Epoch 23 Average Loss: 2.137414
Validate Acc: 0.156 at epoch 23
Epoch 24 Average Loss: 2.133891
Validate Acc: 0.156 at epoch 24
Epoch 25 Average Loss: 2.131091
Validate Acc: 0.156 at epoch 25
Epoch 26 Average Loss: 2.126100
Validate Acc: 0.164 at epoch 26
Epoch 27 Average Loss: 2.123419
Validate Acc: 0.152 at epoch 27
Epoch 28 Average Loss: 2.119605
Validate Acc: 0.164 at epoch 28
Epoch 29 Average Loss: 2.116576
Validate Acc: 0.180 at epoch 29
Epoch 30 Average Loss: 2.111327
Validate Acc: 0.156 at epoch 30
Epoch 31 Average Loss: 2.108281
Validate Acc: 0.156 at epoch 31
Epoch 32 Average Loss: 2.103733
Validate Acc: 0.180 at epoch 32
Epoch 33 Average Loss: 2.097425
Validate Acc: 0.184 at epoch 33
Epoch 34 Average Loss: 2.091542
Validate Acc: 0.212 at epoch 34
Epoch 35 Average Loss: 2.084608
Validate Acc: 0.224 at epoch 35
Epoch 36 Average Loss: 2.079721
Validate Acc: 0.232 at epoch 36
Epoch 37 Average Loss: 2.072323
Validate Acc: 0.220 at epoch 37
```

```
Epoch 38 Average Loss: 2.064732
Validate Acc: 0.208 at epoch 38
Epoch 39 Average Loss: 2.059197
Validate Acc: 0.240 at epoch 39
Epoch 40 Average Loss: 2.052938
Validate Acc: 0.240 at epoch 40
Epoch 41 Average Loss: 2.046486
Validate Acc: 0.240 at epoch 41
Epoch 42 Average Loss: 2.039875
Validate Acc: 0.240 at epoch 42
Epoch 43 Average Loss: 2.034940
Validate Acc: 0.256 at epoch 43
Epoch 44 Average Loss: 2.029816
Validate Acc: 0.264 at epoch 44
Epoch 45 Average Loss: 2.025140
Validate Acc: 0.248 at epoch 45
Epoch 46 Average Loss: 2.020608
Validate Acc: 0.264 at epoch 46
Epoch 47 Average Loss: 2.016039
Validate Acc: 0.240 at epoch 47
Epoch 48 Average Loss: 2.011806
Validate Acc: 0.272 at epoch 48
Epoch 49 Average Loss: 2.007585
Validate Acc: 0.276 at epoch 49
Epoch 50 Average Loss: 2.004820
Validate Acc: 0.272 at epoch 50
Epoch 51 Average Loss: 2.000376
Validate Acc: 0.264 at epoch 51
Epoch 52 Average Loss: 1.996367
Validate Acc: 0.264 at epoch 52
Epoch 53 Average Loss: 1.992538
Validate Acc: 0.272 at epoch 53
Epoch 54 Average Loss: 1.988861
Validate Acc: 0.280 at epoch 54
Epoch 55 Average Loss: 1.984706
Validate Acc: 0.272 at epoch 55
Epoch 56 Average Loss: 1.980969
Validate Acc: 0.284 at epoch 56
Epoch 57 Average Loss: 1.978159
Validate Acc: 0.288 at epoch 57
Epoch 58 Average Loss: 1.971185
Validate Acc: 0.272 at epoch 58
Epoch 59 Average Loss: 1.966747
Validate Acc: 0.292 at epoch 59
Epoch 60 Average Loss: 1.961042
Validate Acc: 0.300 at epoch 60
Epoch 61 Average Loss: 1.956753
Validate Acc: 0.292 at epoch 61
Epoch 62 Average Loss: 1.948269
Validate Acc: 0.296 at epoch 62
Epoch 63 Average Loss: 1.941670
Validate Acc: 0.272 at epoch 63
Epoch 64 Average Loss: 1.939292
Validate Acc: 0.292 at epoch 64
Epoch 65 Average Loss: 1.928235
Validate Acc: 0.280 at epoch 65
Epoch 66 Average Loss: 1.923156
Validate Acc: 0.292 at epoch 66
Epoch 67 Average Loss: 1.919814
Validate Acc: 0.288 at epoch 67
Epoch 68 Average Loss: 1.914639
Validate Acc: 0.300 at epoch 68
Epoch 69 Average Loss: 1.910710
Validate Acc: 0.280 at epoch 69
Epoch 70 Average Loss: 1.903573
Validate Acc: 0.272 at epoch 70
Epoch 71 Average Loss: 1.900047
Validate Acc: 0.312 at epoch 71
Epoch 72 Average Loss: 1.895670
Validate Acc: 0.308 at epoch 72
Epoch 73 Average Loss: 1.890475
Validate Acc: 0.304 at epoch 73
Epoch 74 Average Loss: 1.891105
Validate Acc: 0.296 at epoch 74
Epoch 75 Average Loss: 1.881726
Validate Acc: 0.276 at epoch 75
```

```
Epoch 76 Average Loss: 1.877506
Validate Acc: 0.304 at epoch 76
Epoch 77 Average Loss: 1.875199
Validate Acc: 0.304 at epoch 77
Epoch 78 Average Loss: 1.872609
Validate Acc: 0.288 at epoch 78
Epoch 79 Average Loss: 1.866659
Validate Acc: 0.300 at epoch 79
Epoch 80 Average Loss: 1.864681
Validate Acc: 0.280 at epoch 80
Epoch 81 Average Loss: 1.863894
Validate Acc: 0.312 at epoch 81
Epoch 82 Average Loss: 1.853216
Validate Acc: 0.304 at epoch 82
Epoch 83 Average Loss: 1.851423
Validate Acc: 0.336 at epoch 83
Epoch 84 Average Loss: 1.846774
Validate Acc: 0.304 at epoch 84
Epoch 85 Average Loss: 1.840786
Validate Acc: 0.300 at epoch 85
Epoch 86 Average Loss: 1.836279
Validate Acc: 0.316 at epoch 86
Epoch 87 Average Loss: 1.833165
Validate Acc: 0.320 at epoch 87
Epoch 88 Average Loss: 1.828864
Validate Acc: 0.308 at epoch 88
Epoch 89 Average Loss: 1.823550
Validate Acc: 0.328 at epoch 89
Epoch 90 Average Loss: 1.818268
Validate Acc: 0.312 at epoch 90
Epoch 91 Average Loss: 1.815595
Validate Acc: 0.320 at epoch 91
Epoch 92 Average Loss: 1.810634
Validate Acc: 0.308 at epoch 92
Epoch 93 Average Loss: 1.813736
Validate Acc: 0.324 at epoch 93
Epoch 94 Average Loss: 1.803260
Validate Acc: 0.328 at epoch 94
Epoch 95 Average Loss: 1.804507
Validate Acc: 0.324 at epoch 95
Epoch 96 Average Loss: 1.794509
Validate Acc: 0.324 at epoch 96
Epoch 97 Average Loss: 1.785685
Validate Acc: 0.308 at epoch 97
Epoch 98 Average Loss: 1.790206
Validate Acc: 0.336 at epoch 98
Epoch 99 Average Loss: 1.782510
Validate Acc: 0.336 at epoch 99
Epoch 100 Average Loss: 1.779262
Validate Acc: 0.332 at epoch 100
Epoch 101 Average Loss: 1.782409
Validate Acc: 0.340 at epoch 101
Epoch 102 Average Loss: 1.771371
Validate Acc: 0.336 at epoch 102
Epoch 103 Average Loss: 1.775362
Validate Acc: 0.324 at epoch 103
Epoch 104 Average Loss: 1.764602
Validate Acc: 0.344 at epoch 104
Epoch 105 Average Loss: 1.768665
Validate Acc: 0.336 at epoch 105
Epoch 106 Average Loss: 1.757527
Validate Acc: 0.364 at epoch 106
Epoch 107 Average Loss: 1.758063
Validate Acc: 0.344 at epoch 107
Epoch 108 Average Loss: 1.755917
Validate Acc: 0.360 at epoch 108
Epoch 109 Average Loss: 1.746460
Validate Acc: 0.360 at epoch 109
Epoch 110 Average Loss: 1.745129
Validate Acc: 0.340 at epoch 110
Epoch 111 Average Loss: 1.748022
Validate Acc: 0.352 at epoch 111
Epoch 112 Average Loss: 1.739298
Validate Acc: 0.348 at epoch 112
Epoch 113 Average Loss: 1.730480
Validate Acc: 0.360 at epoch 113
```

```
Epoch 114 Average Loss: 1.729121
Validate Acc: 0.356 at epoch 114
Epoch 115 Average Loss: 1.723029
Validate Acc: 0.356 at epoch 115
Epoch 116 Average Loss: 1.715264
Validate Acc: 0.376 at epoch 116
Epoch 117 Average Loss: 1.720253
Validate Acc: 0.328 at epoch 117
Epoch 118 Average Loss: 1.718464
Validate Acc: 0.388 at epoch 118
Epoch 119 Average Loss: 1.714330
Validate Acc: 0.372 at epoch 119
Epoch 120 Average Loss: 1.709779
Validate Acc: 0.360 at epoch 120
Epoch 121 Average Loss: 1.704780
Validate Acc: 0.380 at epoch 121
Epoch 122 Average Loss: 1.702836
Validate Acc: 0.372 at epoch 122
Epoch 123 Average Loss: 1.704085
Validate Acc: 0.400 at epoch 123
Epoch 124 Average Loss: 1.699547
Validate Acc: 0.344 at epoch 124
Epoch 125 Average Loss: 1.713360
Validate Acc: 0.360 at epoch 125
Epoch 126 Average Loss: 1.695894
Validate Acc: 0.372 at epoch 126
Epoch 127 Average Loss: 1.686289
Validate Acc: 0.372 at epoch 127
Epoch 128 Average Loss: 1.689693
Validate Acc: 0.376 at epoch 128
Epoch 129 Average Loss: 1.679675
Validate Acc: 0.352 at epoch 129
Epoch 130 Average Loss: 1.685885
Validate Acc: 0.360 at epoch 130
Epoch 131 Average Loss: 1.687767
Validate Acc: 0.384 at epoch 131
Epoch 132 Average Loss: 1.677277
Validate Acc: 0.364 at epoch 132
Epoch 133 Average Loss: 1.670431
Validate Acc: 0.392 at epoch 133
Epoch 134 Average Loss: 1.676042
Validate Acc: 0.360 at epoch 134
Epoch 135 Average Loss: 1.677665
Validate Acc: 0.396 at epoch 135
Epoch 136 Average Loss: 1.667005
Validate Acc: 0.392 at epoch 136
Epoch 137 Average Loss: 1.665115
Validate Acc: 0.364 at epoch 137
Epoch 138 Average Loss: 1.679031
Validate Acc: 0.416 at epoch 138
Epoch 139 Average Loss: 1.661640
Validate Acc: 0.400 at epoch 139
Epoch 140 Average Loss: 1.649807
Validate Acc: 0.396 at epoch 140
Epoch 141 Average Loss: 1.654271
Validate Acc: 0.412 at epoch 141
Epoch 142 Average Loss: 1.641158
Validate Acc: 0.404 at epoch 142
Epoch 143 Average Loss: 1.658287
Validate Acc: 0.400 at epoch 143
Epoch 144 Average Loss: 1.648589
Validate Acc: 0.372 at epoch 144
Epoch 145 Average Loss: 1.659130
Validate Acc: 0.400 at epoch 145
Epoch 146 Average Loss: 1.632315
Validate Acc: 0.392 at epoch 146
Epoch 147 Average Loss: 1.633325
Validate Acc: 0.400 at epoch 147
Epoch 148 Average Loss: 1.635695
Validate Acc: 0.400 at epoch 148
Epoch 149 Average Loss: 1.636261
Validate Acc: 0.388 at epoch 149
Epoch 150 Average Loss: 1.630714
Validate Acc: 0.412 at epoch 150
Epoch 151 Average Loss: 1.629453
Validate Acc: 0.388 at epoch 151
```

```
Epoch 152 Average Loss: 1.623542
Validate Acc: 0.408 at epoch 152
Epoch 153 Average Loss: 1.634781
Validate Acc: 0.404 at epoch 153
Epoch 154 Average Loss: 1.625910
Validate Acc: 0.400 at epoch 154
Epoch 155 Average Loss: 1.619330
Validate Acc: 0.412 at epoch 155
Epoch 156 Average Loss: 1.630657
Validate Acc: 0.388 at epoch 156
Epoch 157 Average Loss: 1.617302
Validate Acc: 0.392 at epoch 157
Epoch 158 Average Loss: 1.622002
Validate Acc: 0.440 at epoch 158
Epoch 159 Average Loss: 1.607441
Validate Acc: 0.400 at epoch 159
Epoch 160 Average Loss: 1.619429
Validate Acc: 0.404 at epoch 160
Epoch 161 Average Loss: 1.602463
Validate Acc: 0.412 at epoch 161
Epoch 162 Average Loss: 1.607001
Validate Acc: 0.428 at epoch 162
Epoch 163 Average Loss: 1.598181
Validate Acc: 0.420 at epoch 163
Epoch 164 Average Loss: 1.593769
Validate Acc: 0.424 at epoch 164
Epoch 165 Average Loss: 1.595689
Validate Acc: 0.392 at epoch 165
Epoch 166 Average Loss: 1.594539
Validate Acc: 0.380 at epoch 166
Epoch 167 Average Loss: 1.605638
Validate Acc: 0.400 at epoch 167
Epoch 168 Average Loss: 1.588761
Validate Acc: 0.400 at epoch 168
Epoch 169 Average Loss: 1.592115
Validate Acc: 0.384 at epoch 169
Epoch 170 Average Loss: 1.578623
Validate Acc: 0.392 at epoch 170
Epoch 171 Average Loss: 1.585699
Validate Acc: 0.376 at epoch 171
Epoch 172 Average Loss: 1.600988
Validate Acc: 0.416 at epoch 172
Epoch 173 Average Loss: 1.572812
Validate Acc: 0.400 at epoch 173
Epoch 174 Average Loss: 1.578653
Validate Acc: 0.408 at epoch 174
Epoch 175 Average Loss: 1.577165
Validate Acc: 0.440 at epoch 175
Epoch 176 Average Loss: 1.585925
Validate Acc: 0.388 at epoch 176
Epoch 177 Average Loss: 1.588278
Validate Acc: 0.404 at epoch 177
Epoch 178 Average Loss: 1.584754
Validate Acc: 0.404 at epoch 178
Epoch 179 Average Loss: 1.566541
Validate Acc: 0.388 at epoch 179
Epoch 180 Average Loss: 1.557121
Validate Acc: 0.396 at epoch 180
Epoch 181 Average Loss: 1.567122
Validate Acc: 0.432 at epoch 181
Epoch 182 Average Loss: 1.573544
Validate Acc: 0.364 at epoch 182
Epoch 183 Average Loss: 1.568988
Validate Acc: 0.404 at epoch 183
Epoch 184 Average Loss: 1.570875
Validate Acc: 0.416 at epoch 184
Epoch 185 Average Loss: 1.563361
Validate Acc: 0.396 at epoch 185
Epoch 186 Average Loss: 1.549533
Validate Acc: 0.392 at epoch 186
Epoch 187 Average Loss: 1.553683
Validate Acc: 0.428 at epoch 187
Epoch 188 Average Loss: 1.549736
Validate Acc: 0.388 at epoch 188
Epoch 189 Average Loss: 1.558268
Validate Acc: 0.380 at epoch 189
```

```
Epoch 190 Average Loss: 1.550711
Validate Acc: 0.432 at epoch 190
Epoch 191 Average Loss: 1.564052
Validate Acc: 0.396 at epoch 191
Epoch 192 Average Loss: 1.542323
Validate Acc: 0.412 at epoch 192
Epoch 193 Average Loss: 1.536830
Validate Acc: 0.424 at epoch 193
Epoch 194 Average Loss: 1.541654
Validate Acc: 0.436 at epoch 194
Epoch 195 Average Loss: 1.547231
Validate Acc: 0.396 at epoch 195
Epoch 196 Average Loss: 1.545555
Validate Acc: 0.368 at epoch 196
Epoch 197 Average Loss: 1.549384
Validate Acc: 0.416 at epoch 197
Epoch 198 Average Loss: 1.533199
Validate Acc: 0.380 at epoch 198
Epoch 199 Average Loss: 1.547566
Validate Acc: 0.388 at epoch 199
Epoch 200 Average Loss: 1.531301
Validate Acc: 0.400 at epoch 200
Epoch 201 Average Loss: 1.523991
Validate Acc: 0.404 at epoch 201
Epoch 202 Average Loss: 1.526939
Validate Acc: 0.420 at epoch 202
Epoch 203 Average Loss: 1.531253
Validate Acc: 0.356 at epoch 203
Epoch 204 Average Loss: 1.520999
Validate Acc: 0.428 at epoch 204
Epoch 205 Average Loss: 1.534998
Validate Acc: 0.408 at epoch 205
Epoch 206 Average Loss: 1.515452
Validate Acc: 0.412 at epoch 206
Epoch 207 Average Loss: 1.511439
Validate Acc: 0.396 at epoch 207
Epoch 208 Average Loss: 1.515807
Validate Acc: 0.396 at epoch 208
Epoch 209 Average Loss: 1.503241
Validate Acc: 0.372 at epoch 209
Epoch 210 Average Loss: 1.504135
Validate Acc: 0.408 at epoch 210
Epoch 211 Average Loss: 1.506031
Validate Acc: 0.428 at epoch 211
Epoch 212 Average Loss: 1.511057
Validate Acc: 0.412 at epoch 212
Epoch 213 Average Loss: 1.484203
Validate Acc: 0.420 at epoch 213
Epoch 214 Average Loss: 1.489965
Validate Acc: 0.400 at epoch 214
Epoch 215 Average Loss: 1.511189
Validate Acc: 0.416 at epoch 215
Epoch 216 Average Loss: 1.491926
Validate Acc: 0.416 at epoch 216
Epoch 217 Average Loss: 1.482612
Validate Acc: 0.424 at epoch 217
Epoch 218 Average Loss: 1.495352
Validate Acc: 0.444 at epoch 218
Epoch 219 Average Loss: 1.493476
Validate Acc: 0.396 at epoch 219
Epoch 220 Average Loss: 1.479093
Validate Acc: 0.384 at epoch 220
Epoch 221 Average Loss: 1.487525
Validate Acc: 0.392 at epoch 221
Epoch 222 Average Loss: 1.481535
Validate Acc: 0.424 at epoch 222
Epoch 223 Average Loss: 1.475802
Validate Acc: 0.420 at epoch 223
Epoch 224 Average Loss: 1.486694
Validate Acc: 0.416 at epoch 224
Epoch 225 Average Loss: 1.512594
Validate Acc: 0.404 at epoch 225
Epoch 226 Average Loss: 1.484659
Validate Acc: 0.420 at epoch 226
Epoch 227 Average Loss: 1.473051
Validate Acc: 0.400 at epoch 227
```

```
Epoch 228 Average Loss: 1.480664
Validate Acc: 0.416 at epoch 228
Epoch 229 Average Loss: 1.482037
Validate Acc: 0.380 at epoch 229
Epoch 230 Average Loss: 1.465613
Validate Acc: 0.424 at epoch 230
Epoch 231 Average Loss: 1.469296
Validate Acc: 0.356 at epoch 231
Epoch 232 Average Loss: 1.459581
Validate Acc: 0.456 at epoch 232
Epoch 233 Average Loss: 1.471948
Validate Acc: 0.416 at epoch 233
Epoch 234 Average Loss: 1.459135
Validate Acc: 0.368 at epoch 234
Epoch 235 Average Loss: 1.468795
Validate Acc: 0.408 at epoch 235
Epoch 236 Average Loss: 1.452732
Validate Acc: 0.392 at epoch 236
Epoch 237 Average Loss: 1.453745
Validate Acc: 0.416 at epoch 237
Epoch 238 Average Loss: 1.457049
Validate Acc: 0.428 at epoch 238
Epoch 239 Average Loss: 1.453939
Validate Acc: 0.424 at epoch 239
Epoch 240 Average Loss: 1.469812
Validate Acc: 0.412 at epoch 240
Epoch 241 Average Loss: 1.464340
Validate Acc: 0.408 at epoch 241
Epoch 242 Average Loss: 1.452883
Validate Acc: 0.424 at epoch 242
Epoch 243 Average Loss: 1.444831
Validate Acc: 0.408 at epoch 243
Epoch 244 Average Loss: 1.446930
Validate Acc: 0.432 at epoch 244
Epoch 245 Average Loss: 1.425847
Validate Acc: 0.400 at epoch 245
Epoch 246 Average Loss: 1.423172
Validate Acc: 0.400 at epoch 246
Epoch 247 Average Loss: 1.453592
Validate Acc: 0.416 at epoch 247
Epoch 248 Average Loss: 1.439309
Validate Acc: 0.392 at epoch 248
Epoch 249 Average Loss: 1.428883
Validate Acc: 0.428 at epoch 249
Epoch 250 Average Loss: 1.432066
Validate Acc: 0.400 at epoch 250
Epoch 251 Average Loss: 1.417653
Validate Acc: 0.420 at epoch 251
Epoch 252 Average Loss: 1.418281
Validate Acc: 0.436 at epoch 252
Epoch 253 Average Loss: 1.426685
Validate Acc: 0.432 at epoch 253
Epoch 254 Average Loss: 1.414609
Validate Acc: 0.412 at epoch 254
Epoch 255 Average Loss: 1.410159
Validate Acc: 0.420 at epoch 255
Epoch 256 Average Loss: 1.426375
Validate Acc: 0.412 at epoch 256
Epoch 257 Average Loss: 1.410016
Validate Acc: 0.396 at epoch 257
Epoch 258 Average Loss: 1.410985
Validate Acc: 0.428 at epoch 258
Epoch 259 Average Loss: 1.419075
Validate Acc: 0.420 at epoch 259
Epoch 260 Average Loss: 1.391994
Validate Acc: 0.432 at epoch 260
Epoch 261 Average Loss: 1.407750
Validate Acc: 0.440 at epoch 261
Epoch 262 Average Loss: 1.396783
Validate Acc: 0.416 at epoch 262
Epoch 263 Average Loss: 1.414258
Validate Acc: 0.408 at epoch 263
Epoch 264 Average Loss: 1.404881
Validate Acc: 0.428 at epoch 264
Epoch 265 Average Loss: 1.398321
Validate Acc: 0.436 at epoch 265
```

```
Epoch 266 Average Loss: 1.383947
Validate Acc: 0.416 at epoch 266
Epoch 267 Average Loss: 1.407304
Validate Acc: 0.416 at epoch 267
Epoch 268 Average Loss: 1.417141
Validate Acc: 0.404 at epoch 268
Epoch 269 Average Loss: 1.390210
Validate Acc: 0.424 at epoch 269
Epoch 270 Average Loss: 1.390388
Validate Acc: 0.440 at epoch 270
Epoch 271 Average Loss: 1.383689
Validate Acc: 0.444 at epoch 271
Epoch 272 Average Loss: 1.387980
Validate Acc: 0.408 at epoch 272
Epoch 273 Average Loss: 1.396812
Validate Acc: 0.440 at epoch 273
Epoch 274 Average Loss: 1.372141
Validate Acc: 0.404 at epoch 274
Epoch 275 Average Loss: 1.395404
Validate Acc: 0.428 at epoch 275
Epoch 276 Average Loss: 1.367631
Validate Acc: 0.412 at epoch 276
Epoch 277 Average Loss: 1.359639
Validate Acc: 0.412 at epoch 277
Epoch 278 Average Loss: 1.361498
Validate Acc: 0.420 at epoch 278
Epoch 279 Average Loss: 1.374428
Validate Acc: 0.428 at epoch 279
Epoch 280 Average Loss: 1.376235
Validate Acc: 0.432 at epoch 280
Epoch 281 Average Loss: 1.374040
Validate Acc: 0.428 at epoch 281
Epoch 282 Average Loss: 1.374708
Validate Acc: 0.416 at epoch 282
Epoch 283 Average Loss: 1.357879
Validate Acc: 0.392 at epoch 283
Epoch 284 Average Loss: 1.386909
Validate Acc: 0.432 at epoch 284
Epoch 285 Average Loss: 1.384413
Validate Acc: 0.424 at epoch 285
Epoch 286 Average Loss: 1.359817
Validate Acc: 0.424 at epoch 286
Epoch 287 Average Loss: 1.357835
Validate Acc: 0.424 at epoch 287
Epoch 288 Average Loss: 1.346156
Validate Acc: 0.424 at epoch 288
Epoch 289 Average Loss: 1.369591
Validate Acc: 0.424 at epoch 289
Epoch 290 Average Loss: 1.373656
Validate Acc: 0.380 at epoch 290
Epoch 291 Average Loss: 1.352666
Validate Acc: 0.416 at epoch 291
Epoch 292 Average Loss: 1.357474
Validate Acc: 0.412 at epoch 292
Epoch 293 Average Loss: 1.367865
Validate Acc: 0.440 at epoch 293
Epoch 294 Average Loss: 1.355537
Validate Acc: 0.432 at epoch 294
Epoch 295 Average Loss: 1.350724
Validate Acc: 0.408 at epoch 295
Epoch 296 Average Loss: 1.337046
Validate Acc: 0.428 at epoch 296
Epoch 297 Average Loss: 1.349844
Validate Acc: 0.444 at epoch 297
Epoch 298 Average Loss: 1.371905
Validate Acc: 0.420 at epoch 298
Epoch 299 Average Loss: 1.334595
Validate Acc: 0.384 at epoch 299
Epoch 300 Average Loss: 1.344624
Validate Acc: 0.428 at epoch 300
Epoch 301 Average Loss: 1.342096
Validate Acc: 0.400 at epoch 301
Epoch 302 Average Loss: 1.334619
Validate Acc: 0.440 at epoch 302
Epoch 303 Average Loss: 1.339460
Validate Acc: 0.432 at epoch 303
```

```
Epoch 304 Average Loss: 1.314487
Validate Acc: 0.416 at epoch 304
Epoch 305 Average Loss: 1.326375
Validate Acc: 0.440 at epoch 305
Epoch 306 Average Loss: 1.329255
Validate Acc: 0.424 at epoch 306
Epoch 307 Average Loss: 1.325932
Validate Acc: 0.440 at epoch 307
Epoch 308 Average Loss: 1.339802
Validate Acc: 0.400 at epoch 308
Epoch 309 Average Loss: 1.322237
Validate Acc: 0.432 at epoch 309
Epoch 310 Average Loss: 1.335419
Validate Acc: 0.428 at epoch 310
Epoch 311 Average Loss: 1.327730
Validate Acc: 0.372 at epoch 311
Epoch 312 Average Loss: 1.314266
Validate Acc: 0.408 at epoch 312
Epoch 313 Average Loss: 1.318192
Validate Acc: 0.452 at epoch 313
Epoch 314 Average Loss: 1.305455
Validate Acc: 0.436 at epoch 314
Epoch 315 Average Loss: 1.306809
Validate Acc: 0.416 at epoch 315
Epoch 316 Average Loss: 1.312786
Validate Acc: 0.440 at epoch 316
Epoch 317 Average Loss: 1.312696
Validate Acc: 0.428 at epoch 317
Epoch 318 Average Loss: 1.329452
Validate Acc: 0.384 at epoch 318
Epoch 319 Average Loss: 1.319604
Validate Acc: 0.432 at epoch 319
Epoch 320 Average Loss: 1.305728
Validate Acc: 0.452 at epoch 320
Epoch 321 Average Loss: 1.286521
Validate Acc: 0.456 at epoch 321
Epoch 322 Average Loss: 1.302915
Validate Acc: 0.416 at epoch 322
Epoch 323 Average Loss: 1.333184
Validate Acc: 0.448 at epoch 323
Epoch 324 Average Loss: 1.306762
Validate Acc: 0.392 at epoch 324
Epoch 325 Average Loss: 1.296105
Validate Acc: 0.448 at epoch 325
Epoch 326 Average Loss: 1.310173
Validate Acc: 0.440 at epoch 326
Epoch 327 Average Loss: 1.287759
Validate Acc: 0.448 at epoch 327
Epoch 328 Average Loss: 1.273367
Validate Acc: 0.440 at epoch 328
Epoch 329 Average Loss: 1.280926
Validate Acc: 0.408 at epoch 329
Epoch 330 Average Loss: 1.327137
Validate Acc: 0.424 at epoch 330
Epoch 331 Average Loss: 1.261664
Validate Acc: 0.408 at epoch 331
Epoch 332 Average Loss: 1.304896
Validate Acc: 0.408 at epoch 332
Epoch 333 Average Loss: 1.307071
Validate Acc: 0.440 at epoch 333
Epoch 334 Average Loss: 1.293840
Validate Acc: 0.428 at epoch 334
Epoch 335 Average Loss: 1.308010
Validate Acc: 0.436 at epoch 335
Epoch 336 Average Loss: 1.275352
Validate Acc: 0.424 at epoch 336
Epoch 337 Average Loss: 1.287759
Validate Acc: 0.416 at epoch 337
Epoch 338 Average Loss: 1.276949
Validate Acc: 0.432 at epoch 338
Epoch 339 Average Loss: 1.273676
Validate Acc: 0.436 at epoch 339
Epoch 340 Average Loss: 1.276180
Validate Acc: 0.416 at epoch 340
Epoch 341 Average Loss: 1.273288
Validate Acc: 0.404 at epoch 341
```

```
Epoch 342 Average Loss: 1.282604
Validate Acc: 0.428 at epoch 342
Epoch 343 Average Loss: 1.280316
Validate Acc: 0.420 at epoch 343
Epoch 344 Average Loss: 1.266950
Validate Acc: 0.412 at epoch 344
Epoch 345 Average Loss: 1.277305
Validate Acc: 0.416 at epoch 345
Epoch 346 Average Loss: 1.240453
Validate Acc: 0.436 at epoch 346
Epoch 347 Average Loss: 1.275535
Validate Acc: 0.372 at epoch 347
Epoch 348 Average Loss: 1.294388
Validate Acc: 0.448 at epoch 348
Epoch 349 Average Loss: 1.272090
Validate Acc: 0.428 at epoch 349
Epoch 350 Average Loss: 1.287654
Validate Acc: 0.424 at epoch 350
Epoch 351 Average Loss: 1.266725
Validate Acc: 0.420 at epoch 351
Epoch 352 Average Loss: 1.243217
Validate Acc: 0.408 at epoch 352
Epoch 353 Average Loss: 1.249888
Validate Acc: 0.436 at epoch 353
Epoch 354 Average Loss: 1.255751
Validate Acc: 0.416 at epoch 354
Epoch 355 Average Loss: 1.251579
Validate Acc: 0.408 at epoch 355
Epoch 356 Average Loss: 1.261490
Validate Acc: 0.448 at epoch 356
Epoch 357 Average Loss: 1.254549
Validate Acc: 0.428 at epoch 357
Epoch 358 Average Loss: 1.247319
Validate Acc: 0.424 at epoch 358
Epoch 359 Average Loss: 1.260748
Validate Acc: 0.432 at epoch 359
Epoch 360 Average Loss: 1.221053
Validate Acc: 0.436 at epoch 360
Epoch 361 Average Loss: 1.238669
Validate Acc: 0.436 at epoch 361
Epoch 362 Average Loss: 1.237011
Validate Acc: 0.440 at epoch 362
Epoch 363 Average Loss: 1.232818
Validate Acc: 0.416 at epoch 363
Epoch 364 Average Loss: 1.248395
Validate Acc: 0.400 at epoch 364
Epoch 365 Average Loss: 1.227985
Validate Acc: 0.424 at epoch 365
Epoch 366 Average Loss: 1.276764
Validate Acc: 0.412 at epoch 366
Epoch 367 Average Loss: 1.253486
Validate Acc: 0.456 at epoch 367
Epoch 368 Average Loss: 1.231773
Validate Acc: 0.416 at epoch 368
Epoch 369 Average Loss: 1.249176
Validate Acc: 0.416 at epoch 369
Epoch 370 Average Loss: 1.219696
Validate Acc: 0.436 at epoch 370
Epoch 371 Average Loss: 1.225082
Validate Acc: 0.416 at epoch 371
Epoch 372 Average Loss: 1.226542
Validate Acc: 0.456 at epoch 372
Epoch 373 Average Loss: 1.226006
Validate Acc: 0.448 at epoch 373
Epoch 374 Average Loss: 1.247439
Validate Acc: 0.436 at epoch 374
Epoch 375 Average Loss: 1.217156
Validate Acc: 0.464 at epoch 375
Epoch 376 Average Loss: 1.230655
Validate Acc: 0.436 at epoch 376
Epoch 377 Average Loss: 1.228957
Validate Acc: 0.456 at epoch 377
Epoch 378 Average Loss: 1.245523
Validate Acc: 0.412 at epoch 378
Epoch 379 Average Loss: 1.197236
Validate Acc: 0.440 at epoch 379
```

```
Epoch 380 Average Loss: 1.216319
Validate Acc: 0.416 at epoch 380
Epoch 381 Average Loss: 1.202051
Validate Acc: 0.420 at epoch 381
Epoch 382 Average Loss: 1.203228
Validate Acc: 0.440 at epoch 382
Epoch 383 Average Loss: 1.258798
Validate Acc: 0.432 at epoch 383
Epoch 384 Average Loss: 1.201246
Validate Acc: 0.452 at epoch 384
Epoch 385 Average Loss: 1.197963
Validate Acc: 0.416 at epoch 385
Epoch 386 Average Loss: 1.208937
Validate Acc: 0.476 at epoch 386
Epoch 387 Average Loss: 1.205152
Validate Acc: 0.420 at epoch 387
Epoch 388 Average Loss: 1.185760
Validate Acc: 0.412 at epoch 388
Epoch 389 Average Loss: 1.217174
Validate Acc: 0.436 at epoch 389
Epoch 390 Average Loss: 1.201911
Validate Acc: 0.440 at epoch 390
Epoch 391 Average Loss: 1.209857
Validate Acc: 0.396 at epoch 391
Epoch 392 Average Loss: 1.215749
Validate Acc: 0.432 at epoch 392
Epoch 393 Average Loss: 1.204035
Validate Acc: 0.420 at epoch 393
Epoch 394 Average Loss: 1.213793
Validate Acc: 0.444 at epoch 394
Epoch 395 Average Loss: 1.205247
Validate Acc: 0.464 at epoch 395
Epoch 396 Average Loss: 1.200047
Validate Acc: 0.384 at epoch 396
Epoch 397 Average Loss: 1.211609
Validate Acc: 0.392 at epoch 397
Epoch 398 Average Loss: 1.202976
Validate Acc: 0.424 at epoch 398
Epoch 399 Average Loss: 1.190198
Validate Acc: 0.420 at epoch 399
Epoch 400 Average Loss: 1.200648
Validate Acc: 0.432 at epoch 400
Epoch 401 Average Loss: 1.208881
Validate Acc: 0.436 at epoch 401
Epoch 402 Average Loss: 1.169715
Validate Acc: 0.416 at epoch 402
Epoch 403 Average Loss: 1.177297
Validate Acc: 0.440 at epoch 403
Epoch 404 Average Loss: 1.166082
Validate Acc: 0.444 at epoch 404
Epoch 405 Average Loss: 1.166915
Validate Acc: 0.456 at epoch 405
Epoch 406 Average Loss: 1.152798
Validate Acc: 0.456 at epoch 406
Epoch 407 Average Loss: 1.178302
Validate Acc: 0.408 at epoch 407
Epoch 408 Average Loss: 1.180461
Validate Acc: 0.416 at epoch 408
Epoch 409 Average Loss: 1.181172
Validate Acc: 0.416 at epoch 409
Epoch 410 Average Loss: 1.186371
Validate Acc: 0.456 at epoch 410
Epoch 411 Average Loss: 1.161173
Validate Acc: 0.408 at epoch 411
Epoch 412 Average Loss: 1.187811
Validate Acc: 0.440 at epoch 412
Epoch 413 Average Loss: 1.170770
Validate Acc: 0.432 at epoch 413
Epoch 414 Average Loss: 1.182958
Validate Acc: 0.436 at epoch 414
Epoch 415 Average Loss: 1.160209
Validate Acc: 0.444 at epoch 415
Epoch 416 Average Loss: 1.177707
Validate Acc: 0.444 at epoch 416
Epoch 417 Average Loss: 1.214496
Validate Acc: 0.412 at epoch 417
```

```
Epoch 418 Average Loss: 1.156825
Validate Acc: 0.440 at epoch 418
Epoch 419 Average Loss: 1.170137
Validate Acc: 0.412 at epoch 419
Epoch 420 Average Loss: 1.184413
Validate Acc: 0.476 at epoch 420
Epoch 421 Average Loss: 1.166067
Validate Acc: 0.432 at epoch 421
Epoch 422 Average Loss: 1.158509
Validate Acc: 0.448 at epoch 422
Epoch 423 Average Loss: 1.140766
Validate Acc: 0.436 at epoch 423
Epoch 424 Average Loss: 1.164252
Validate Acc: 0.436 at epoch 424
Epoch 425 Average Loss: 1.124362
Validate Acc: 0.396 at epoch 425
Epoch 426 Average Loss: 1.175235
Validate Acc: 0.424 at epoch 426
Epoch 427 Average Loss: 1.126797
Validate Acc: 0.440 at epoch 427
Epoch 428 Average Loss: 1.147540
Validate Acc: 0.444 at epoch 428
Epoch 429 Average Loss: 1.129111
Validate Acc: 0.452 at epoch 429
Epoch 430 Average Loss: 1.152104
Validate Acc: 0.420 at epoch 430
Epoch 431 Average Loss: 1.136970
Validate Acc: 0.424 at epoch 431
Epoch 432 Average Loss: 1.152295
Validate Acc: 0.444 at epoch 432
Epoch 433 Average Loss: 1.167922
Validate Acc: 0.436 at epoch 433
Epoch 434 Average Loss: 1.138171
Validate Acc: 0.420 at epoch 434
Epoch 435 Average Loss: 1.156585
Validate Acc: 0.456 at epoch 435
Epoch 436 Average Loss: 1.153898
Validate Acc: 0.412 at epoch 436
Epoch 437 Average Loss: 1.188933
Validate Acc: 0.424 at epoch 437
Epoch 438 Average Loss: 1.120477
Validate Acc: 0.440 at epoch 438
Epoch 439 Average Loss: 1.123150
Validate Acc: 0.396 at epoch 439
Epoch 440 Average Loss: 1.129018
Validate Acc: 0.456 at epoch 440
Epoch 441 Average Loss: 1.130515
Validate Acc: 0.452 at epoch 441
Epoch 442 Average Loss: 1.130546
Validate Acc: 0.444 at epoch 442
Epoch 443 Average Loss: 1.110817
Validate Acc: 0.444 at epoch 443
Epoch 444 Average Loss: 1.173244
Validate Acc: 0.440 at epoch 444
Epoch 445 Average Loss: 1.120402
Validate Acc: 0.412 at epoch 445
Epoch 446 Average Loss: 1.092842
Validate Acc: 0.428 at epoch 446
Epoch 447 Average Loss: 1.134276
Validate Acc: 0.444 at epoch 447
Epoch 448 Average Loss: 1.126886
Validate Acc: 0.432 at epoch 448
Epoch 449 Average Loss: 1.130596
Validate Acc: 0.440 at epoch 449
Epoch 450 Average Loss: 1.140701
Validate Acc: 0.448 at epoch 450
Epoch 451 Average Loss: 1.111706
Validate Acc: 0.424 at epoch 451
Epoch 452 Average Loss: 1.100036
Validate Acc: 0.488 at epoch 452
Epoch 453 Average Loss: 1.131471
Validate Acc: 0.464 at epoch 453
Epoch 454 Average Loss: 1.142082
Validate Acc: 0.412 at epoch 454
Epoch 455 Average Loss: 1.142527
Validate Acc: 0.432 at epoch 455
```

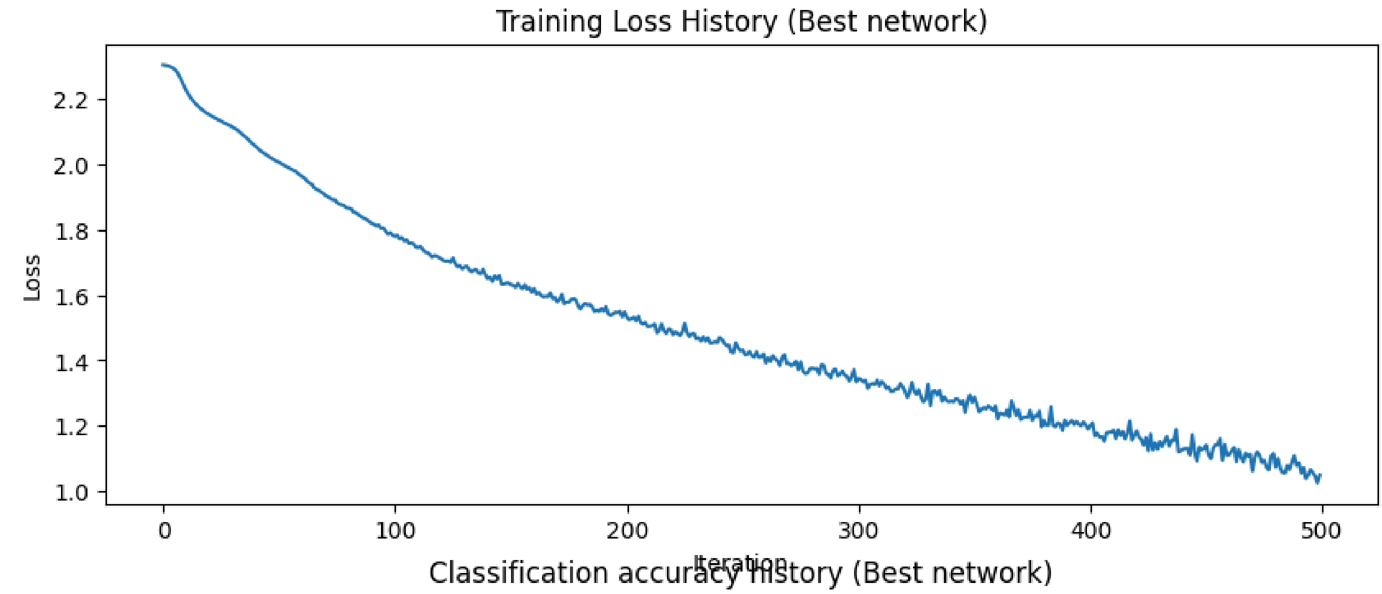
```
Epoch 456 Average Loss: 1.161393
Validate Acc: 0.420 at epoch 456
Epoch 457 Average Loss: 1.095419
Validate Acc: 0.440 at epoch 457
Epoch 458 Average Loss: 1.142858
Validate Acc: 0.460 at epoch 458
Epoch 459 Average Loss: 1.109541
Validate Acc: 0.424 at epoch 459
Epoch 460 Average Loss: 1.112500
Validate Acc: 0.432 at epoch 460
Epoch 461 Average Loss: 1.124814
Validate Acc: 0.472 at epoch 461
Epoch 462 Average Loss: 1.133211
Validate Acc: 0.424 at epoch 462
Epoch 463 Average Loss: 1.090956
Validate Acc: 0.436 at epoch 463
Epoch 464 Average Loss: 1.127877
Validate Acc: 0.472 at epoch 464
Epoch 465 Average Loss: 1.104366
Validate Acc: 0.456 at epoch 465
Epoch 466 Average Loss: 1.089790
Validate Acc: 0.468 at epoch 466
Epoch 467 Average Loss: 1.115934
Validate Acc: 0.436 at epoch 467
Epoch 468 Average Loss: 1.107314
Validate Acc: 0.428 at epoch 468
Epoch 469 Average Loss: 1.113385
Validate Acc: 0.440 at epoch 469
Epoch 470 Average Loss: 1.062608
Validate Acc: 0.392 at epoch 470
Epoch 471 Average Loss: 1.104523
Validate Acc: 0.436 at epoch 471
Epoch 472 Average Loss: 1.120085
Validate Acc: 0.432 at epoch 472
Epoch 473 Average Loss: 1.110177
Validate Acc: 0.460 at epoch 473
Epoch 474 Average Loss: 1.088513
Validate Acc: 0.396 at epoch 474
Epoch 475 Average Loss: 1.095042
Validate Acc: 0.468 at epoch 475
Epoch 476 Average Loss: 1.072977
Validate Acc: 0.432 at epoch 476
Epoch 477 Average Loss: 1.065616
Validate Acc: 0.432 at epoch 477
Epoch 478 Average Loss: 1.107164
Validate Acc: 0.424 at epoch 478
Epoch 479 Average Loss: 1.115376
Validate Acc: 0.448 at epoch 479
Epoch 480 Average Loss: 1.074082
Validate Acc: 0.472 at epoch 480
Epoch 481 Average Loss: 1.100279
Validate Acc: 0.436 at epoch 481
Epoch 482 Average Loss: 1.069626
Validate Acc: 0.416 at epoch 482
Epoch 483 Average Loss: 1.057286
Validate Acc: 0.456 at epoch 483
Epoch 484 Average Loss: 1.056353
Validate Acc: 0.448 at epoch 484
Epoch 485 Average Loss: 1.078896
Validate Acc: 0.468 at epoch 485
Epoch 486 Average Loss: 1.066833
Validate Acc: 0.440 at epoch 486
Epoch 487 Average Loss: 1.088700
Validate Acc: 0.396 at epoch 487
Epoch 488 Average Loss: 1.108666
Validate Acc: 0.404 at epoch 488
Epoch 489 Average Loss: 1.078978
Validate Acc: 0.420 at epoch 489
Epoch 490 Average Loss: 1.087188
Validate Acc: 0.460 at epoch 490
Epoch 491 Average Loss: 1.055574
Validate Acc: 0.404 at epoch 491
Epoch 492 Average Loss: 1.072021
Validate Acc: 0.432 at epoch 492
Epoch 493 Average Loss: 1.039541
Validate Acc: 0.432 at epoch 493
```

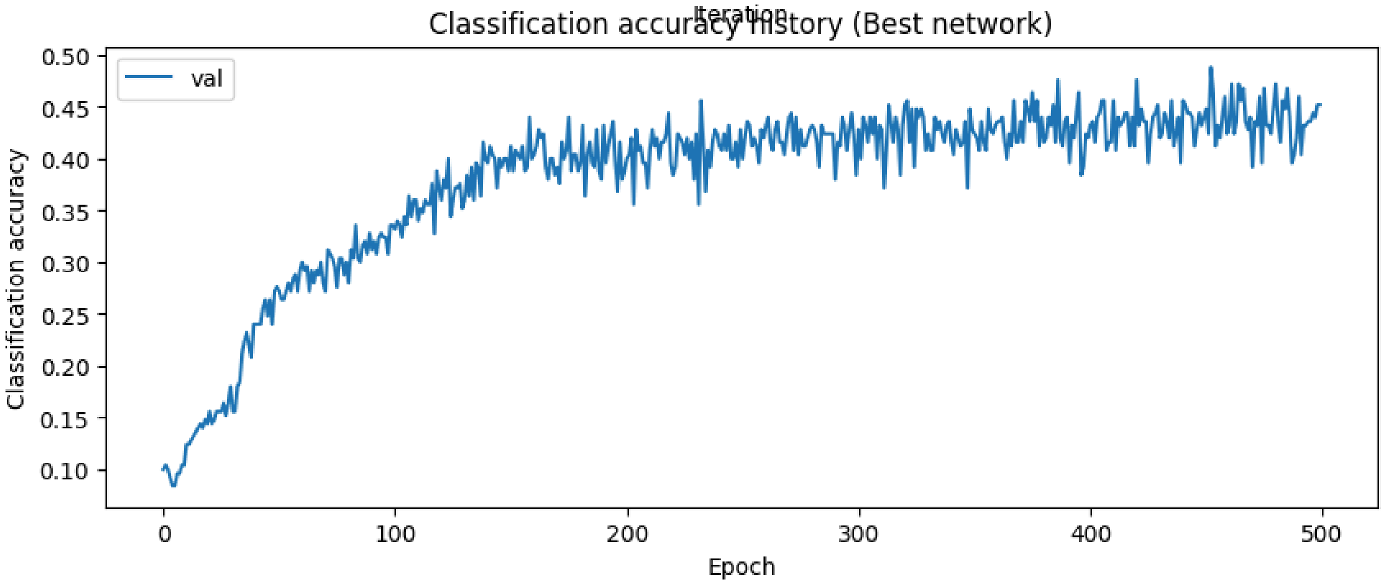
```
Epoch 494 Average Loss: 1.049294
Validate Acc: 0.436 at epoch 494
Epoch 495 Average Loss: 1.066009
Validate Acc: 0.436 at epoch 495
Epoch 496 Average Loss: 1.057221
Validate Acc: 0.444 at epoch 496
Epoch 497 Average Loss: 1.046460
Validate Acc: 0.440 at epoch 497
Epoch 498 Average Loss: 1.026350
Validate Acc: 0.452 at epoch 498
Epoch 499 Average Loss: 1.049862
Validate Acc: 0.452 at epoch 499
```

In [ ]:

```
# TODO: Show the above plots and visualizations for the default params (already #
# done) and the best hyper-params you obtain. You only need to show this for 2  #
# sets of hyper-params.                                                         #
# You just need to store values for the hyperparameters in best_net_hyperparams #
# as a list in the order
# best_net_hyperparams = [lr, weight_decay, epoch, hidden_size]
#################################################################################
# TODO: Plot the training_error and validation_accuracy of the best network (5%)

# Plot the training loss function and validation accuracies
best_lr = 0.02
best_weight_decay = 0.005
best_hidden_size = 300
best_epochs = 500
best_net_hyperparams = [best_lr, best_weight_decay, best_epochs, best_hidden_size]

print(f"Best network hyperparameters: {best_net_hyperparams}")
print(f"Validation accuracy: {validation_accuracy[-1]}")

plt.subplot(2, 1, 1)
plt.plot(train_error)
plt.title("Training Loss History (Best network)")
plt.xlabel("Iteration")
plt.ylabel("Loss")

plt.subplot(2, 1, 2)
# plt.plot(stats['train_acc_history'], label='train')
plt.plot(validation_accuracy, label="val")
plt.title("Classification accuracy history (Best network)")
plt.xlabel("Epoch")
plt.ylabel("Classification accuracy")
plt.legend()
plt.show()

# TODO: visualize the weights of the best network (5%)
def show_net_weights(net):
    W1 = net._modules[0].parameters[0]
    W1 = W1.reshape(3, 32, 32, -1).transpose(3, 1, 2, 0)
    # print(W1.shape)
    plt.imshow(visualize_grid(W1, padding=3).astype("uint8"))
    plt.gca().axis("off")
    plt.show()

print("Best network's weight visualizations:")
show_net_weights(net2)
```

```
Best network hyperparameters: [0.02, 0.005, 500, 300]
Validation accuracy: 0.452
```






```
Best network's weight visualizations:
```

Run on the test set (30%)

When you are done experimenting, you should evaluate your final trained network on the test set; you should get above 35%.

```
In [ ]: test_acc = (best_net.predict(x_test) == y_test).mean()
        print("Test accuracy: ", test_acc)
```

```
Test accuracy:  0.396
```

Inline Question (10%)

Now that you have trained a Neural Network classifier, you may find that your testing accuracy is much lower than the training accuracy. In what ways can we decrease this gap? Select all that apply.

1. Train on a larger dataset.
2. Add more hidden units.
3. Increase the regularization strength.
4. None of the above.

Your Answer: 1, 2, 3

Your Explanation: Training on a larger dataset exposes the model to more diverse examples, helping it generalize better to more objects with diverse features. Adding more hidden units increases the model's capacity to learn complex patterns, hence improving its performance on a more diverse testing dataset. Increasing regularization strength helps prevent overfitting by penalizing large weights or dropping units during training, hence making the model generalize better to the test set.