classification_nn

April 25, 2024

1 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.

[1]:

```
# 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)
```

```python
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,)
```

[2]:

```python
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"]
```

[3]:

```python
# 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
```

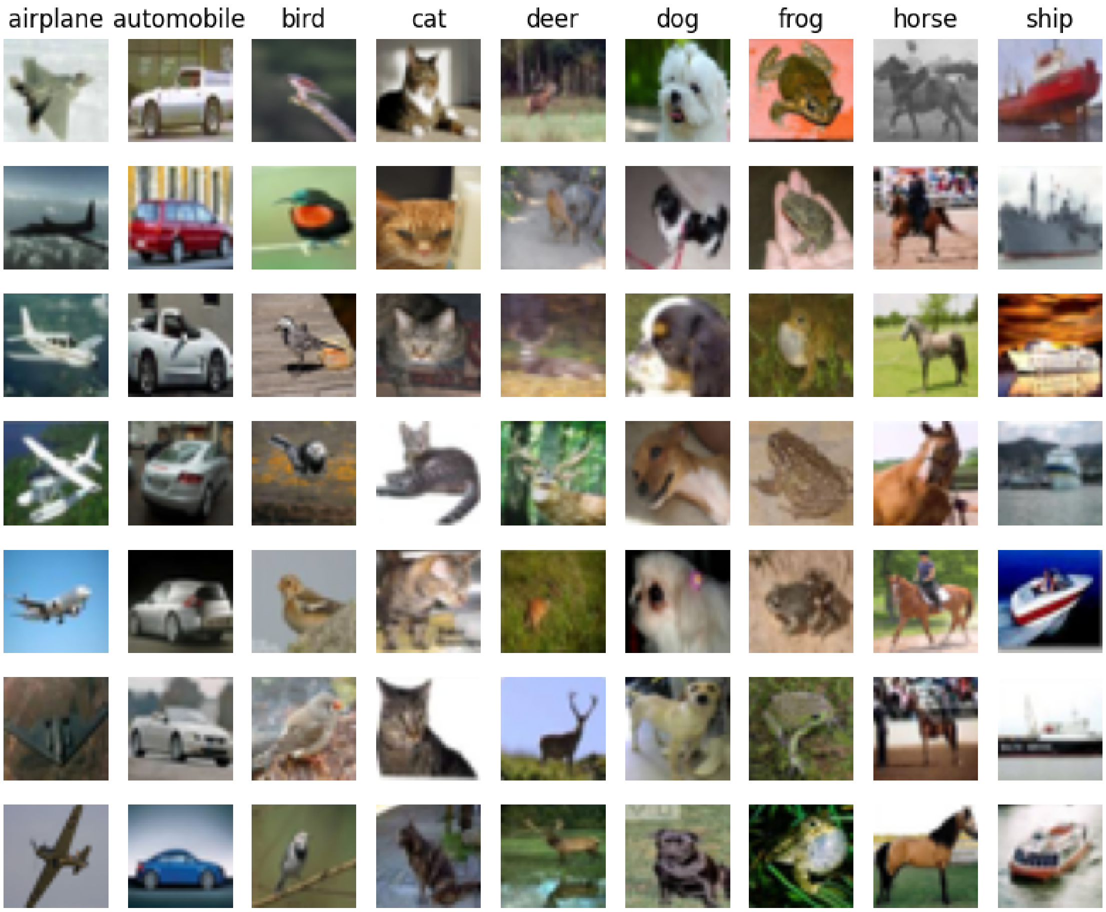
1.1 Visualize some examples from the dataset.

[4]:

```python
# 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,
)
```



1.2 Initialize the model

[5]:

```python
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])

# 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)

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

# 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 Average Loss: 2.302537
Validate Acc: 0.084
Epoch Average Loss: 2.302358
Epoch Average Loss: 2.302153
Epoch Average Loss: 2.301861
Validate Acc: 0.104
Epoch Average Loss: 2.301433
Epoch Average Loss: 2.300851
Epoch Average Loss: 2.299964
Validate Acc: 0.096
Epoch Average Loss: 2.298798
Epoch Average Loss: 2.297321
Epoch Average Loss: 2.295501
Validate Acc: 0.084
Epoch Average Loss: 2.293346
Epoch Average Loss: 2.290893
Epoch Average Loss: 2.287849
Validate Acc: 0.084
Epoch Average Loss: 2.283949
Epoch Average Loss: 2.278915
Epoch Average Loss: 2.272740
Validate Acc: 0.096
Epoch Average Loss: 2.265747
Epoch Average Loss: 2.258297
Epoch Average Loss: 2.250794
```

```
Validate Acc: 0.100
Epoch Average Loss: 2.243016
Epoch Average Loss: 2.235507
Epoch Average Loss: 2.228456
Validate Acc: 0.116
Epoch Average Loss: 2.221825
Epoch Average Loss: 2.215697
Epoch Average Loss: 2.210149
Validate Acc: 0.124
Epoch Average Loss: 2.204867
Epoch Average Loss: 2.200080
Epoch Average Loss: 2.195285
Validate Acc: 0.128
Epoch Average Loss: 2.191232
Epoch Average Loss: 2.187177
Epoch Average Loss: 2.183314
Validate Acc: 0.136
Epoch Average Loss: 2.179658
Epoch Average Loss: 2.176573
Epoch Average Loss: 2.173230
Validate Acc: 0.144
Epoch Average Loss: 2.170173
Epoch Average Loss: 2.167557
Epoch Average Loss: 2.164493
Validate Acc: 0.148
Epoch Average Loss: 2.161955
Epoch Average Loss: 2.159274
Epoch Average Loss: 2.156913
Validate Acc: 0.144
Epoch Average Loss: 2.154436
Epoch Average Loss: 2.152519
Epoch Average Loss: 2.150320
Validate Acc: 0.144
Epoch Average Loss: 2.148442
Epoch Average Loss: 2.146027
Epoch Average Loss: 2.144040
Validate Acc: 0.152
Epoch Average Loss: 2.142573
Epoch Average Loss: 2.140663
Epoch Average Loss: 2.138572
Validate Acc: 0.152
Epoch Average Loss: 2.137174
Epoch Average Loss: 2.135517
Epoch Average Loss: 2.133763
Validate Acc: 0.148
Epoch Average Loss: 2.132108
Epoch Average Loss: 2.130843
Epoch Average Loss: 2.128342
```

```
Validate Acc: 0.148
Epoch Average Loss: 2.127817
Epoch Average Loss: 2.125841
Epoch Average Loss: 2.124409
Validate Acc: 0.148
Epoch Average Loss: 2.123041
Epoch Average Loss: 2.121470
Epoch Average Loss: 2.119940
Validate Acc: 0.152
Epoch Average Loss: 2.118505
Epoch Average Loss: 2.116697
Epoch Average Loss: 2.115945
Validate Acc: 0.168
Epoch Average Loss: 2.113718
Epoch Average Loss: 2.112385
Epoch Average Loss: 2.110330
Validate Acc: 0.176
Epoch Average Loss: 2.108615
Epoch Average Loss: 2.106778
Epoch Average Loss: 2.104724
Validate Acc: 0.172
Epoch Average Loss: 2.102461
Epoch Average Loss: 2.100388
Epoch Average Loss: 2.098450
Validate Acc: 0.172
Epoch Average Loss: 2.096084
Epoch Average Loss: 2.093232
Epoch Average Loss: 2.091584
Validate Acc: 0.184
Epoch Average Loss: 2.088124
Epoch Average Loss: 2.085273
Epoch Average Loss: 2.082583
Validate Acc: 0.232
Epoch Average Loss: 2.079115
Epoch Average Loss: 2.076047
Epoch Average Loss: 2.072967
Validate Acc: 0.232
Epoch Average Loss: 2.070173
Epoch Average Loss: 2.067068
Epoch Average Loss: 2.063438
Validate Acc: 0.224
Epoch Average Loss: 2.060540
Epoch Average Loss: 2.057288
Epoch Average Loss: 2.054034
Validate Acc: 0.236
Epoch Average Loss: 2.051270
Epoch Average Loss: 2.048177
Epoch Average Loss: 2.045248
```

```
Validate Acc: 0.240
Epoch Average Loss: 2.042197
Epoch Average Loss: 2.039321
Epoch Average Loss: 2.036856
Validate Acc: 0.240
Epoch Average Loss: 2.034228
Epoch Average Loss: 2.031098
Epoch Average Loss: 2.029689
Validate Acc: 0.260
Epoch Average Loss: 2.026804
Epoch Average Loss: 2.024594
Epoch Average Loss: 2.021690
Validate Acc: 0.244
Epoch Average Loss: 2.019476
Epoch Average Loss: 2.017448
Epoch Average Loss: 2.015358
Validate Acc: 0.260
Epoch Average Loss: 2.013005
Epoch Average Loss: 2.011135
Epoch Average Loss: 2.008958
Validate Acc: 0.268
Epoch Average Loss: 2.007098
Epoch Average Loss: 2.005913
Epoch Average Loss: 2.003597
Validate Acc: 0.268
Epoch Average Loss: 2.001598
Epoch Average Loss: 1.999838
Epoch Average Loss: 1.997474
Validate Acc: 0.268
Epoch Average Loss: 1.995711
Epoch Average Loss: 1.993893
Epoch Average Loss: 1.992380
Validate Acc: 0.272
Epoch Average Loss: 1.990470
Epoch Average Loss: 1.989180
Epoch Average Loss: 1.987153
Validate Acc: 0.276
Epoch Average Loss: 1.985827
Epoch Average Loss: 1.983262
Epoch Average Loss: 1.981728
Validate Acc: 0.280
Epoch Average Loss: 1.980198
Epoch Average Loss: 1.978801
Epoch Average Loss: 1.976564
Validate Acc: 0.288
Epoch Average Loss: 1.974345
Epoch Average Loss: 1.972096
Epoch Average Loss: 1.970484
```

```
Validate Acc: 0.288
Epoch Average Loss: 1.967622
Epoch Average Loss: 1.965336
Epoch Average Loss: 1.963110
Validate Acc: 0.304
Epoch Average Loss: 1.961092
Epoch Average Loss: 1.958728
Epoch Average Loss: 1.954832
Validate Acc: 0.308
Epoch Average Loss: 1.951695
Epoch Average Loss: 1.949919
Epoch Average Loss: 1.946959
Validate Acc: 0.300
Epoch Average Loss: 1.943884
Epoch Average Loss: 1.940049
Epoch Average Loss: 1.937556
Validate Acc: 0.288
Epoch Average Loss: 1.933858
Epoch Average Loss: 1.931535
Epoch Average Loss: 1.928331
Validate Acc: 0.292
Epoch Average Loss: 1.925393
Epoch Average Loss: 1.923706
Epoch Average Loss: 1.920337
Validate Acc: 0.292
Epoch Average Loss: 1.918130
Epoch Average Loss: 1.914838
Epoch Average Loss: 1.914686
Validate Acc: 0.296
Epoch Average Loss: 1.911120
Epoch Average Loss: 1.910004
Epoch Average Loss: 1.907682
Validate Acc: 0.284
Epoch Average Loss: 1.903876
Epoch Average Loss: 1.902066
Epoch Average Loss: 1.899852
Validate Acc: 0.292
Epoch Average Loss: 1.898004
Epoch Average Loss: 1.893862
Epoch Average Loss: 1.894274
Validate Acc: 0.312
Epoch Average Loss: 1.891401
Epoch Average Loss: 1.888412
Epoch Average Loss: 1.887628
Validate Acc: 0.304
Epoch Average Loss: 1.883805
Epoch Average Loss: 1.883570
Epoch Average Loss: 1.883284
```

```
Validate Acc: 0.308
Epoch Average Loss: 1.879298
Epoch Average Loss: 1.876486
Epoch Average Loss: 1.876580
Validate Acc: 0.304
Epoch Average Loss: 1.873125
Epoch Average Loss: 1.872160
Epoch Average Loss: 1.868687
Validate Acc: 0.292
Epoch Average Loss: 1.868044
Epoch Average Loss: 1.866335
Epoch Average Loss: 1.863730
Validate Acc: 0.284
Epoch Average Loss: 1.862398
Epoch Average Loss: 1.859831
Epoch Average Loss: 1.857877
Validate Acc: 0.292
Epoch Average Loss: 1.856972
Epoch Average Loss: 1.854295
Epoch Average Loss: 1.852176
Validate Acc: 0.300
Epoch Average Loss: 1.850807
Epoch Average Loss: 1.848620
Epoch Average Loss: 1.846455
Validate Acc: 0.296
Epoch Average Loss: 1.844929
Epoch Average Loss: 1.843053
Epoch Average Loss: 1.838885
Validate Acc: 0.296
Epoch Average Loss: 1.838025
Epoch Average Loss: 1.835789
Epoch Average Loss: 1.834729
Validate Acc: 0.324
Epoch Average Loss: 1.832280
Epoch Average Loss: 1.830350
Epoch Average Loss: 1.828094
Validate Acc: 0.304
Epoch Average Loss: 1.826341
Epoch Average Loss: 1.822406
Epoch Average Loss: 1.821506
Validate Acc: 0.312
Epoch Average Loss: 1.820259
Epoch Average Loss: 1.816891
Epoch Average Loss: 1.816177
Validate Acc: 0.320
Epoch Average Loss: 1.815976
Epoch Average Loss: 1.811736
Epoch Average Loss: 1.811231
```

```
Validate Acc: 0.324
Epoch Average Loss: 1.811461
```

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

[6]:

```
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:  0.3426
Validation acc:  0.328
```

1.2.2 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.

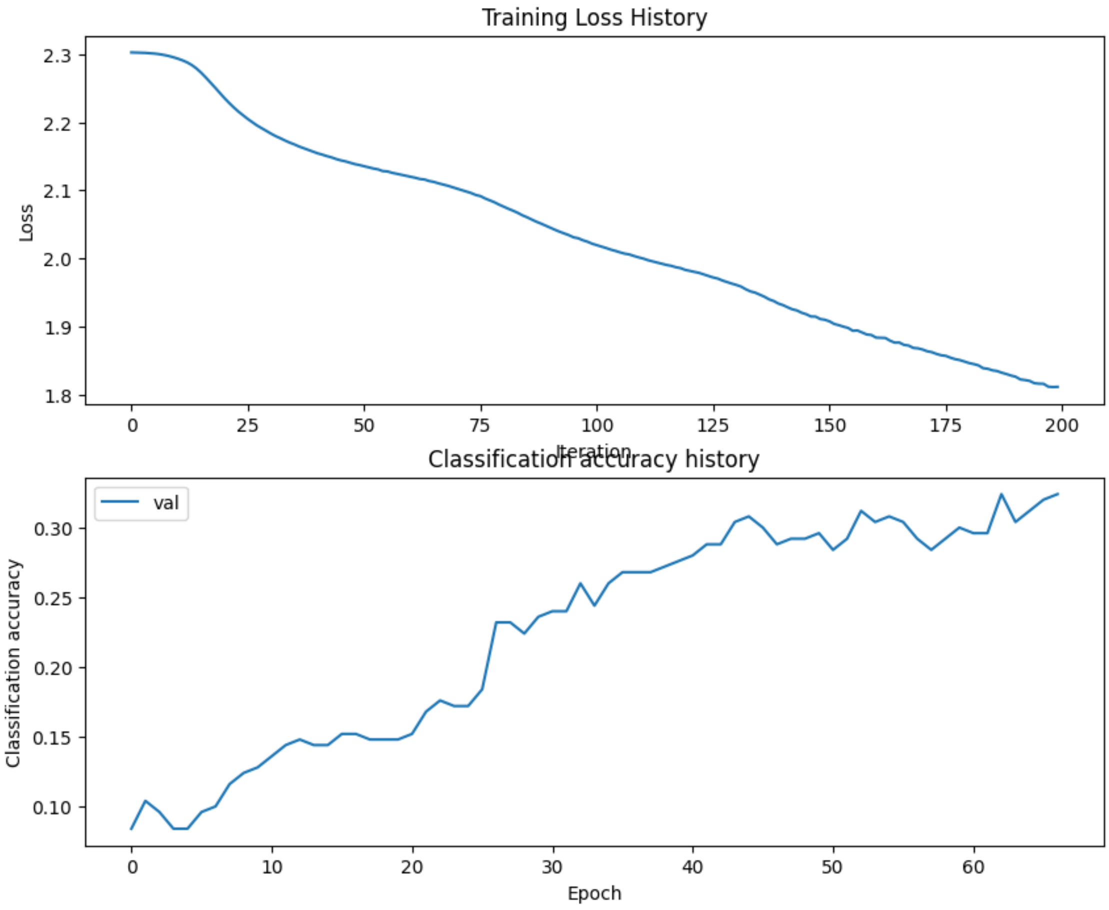
One strategy for getting insight into what's wrong is to plot the training loss function and the validation accuracies during optimization.

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.

[7]:

```
# 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()
```



[8]:

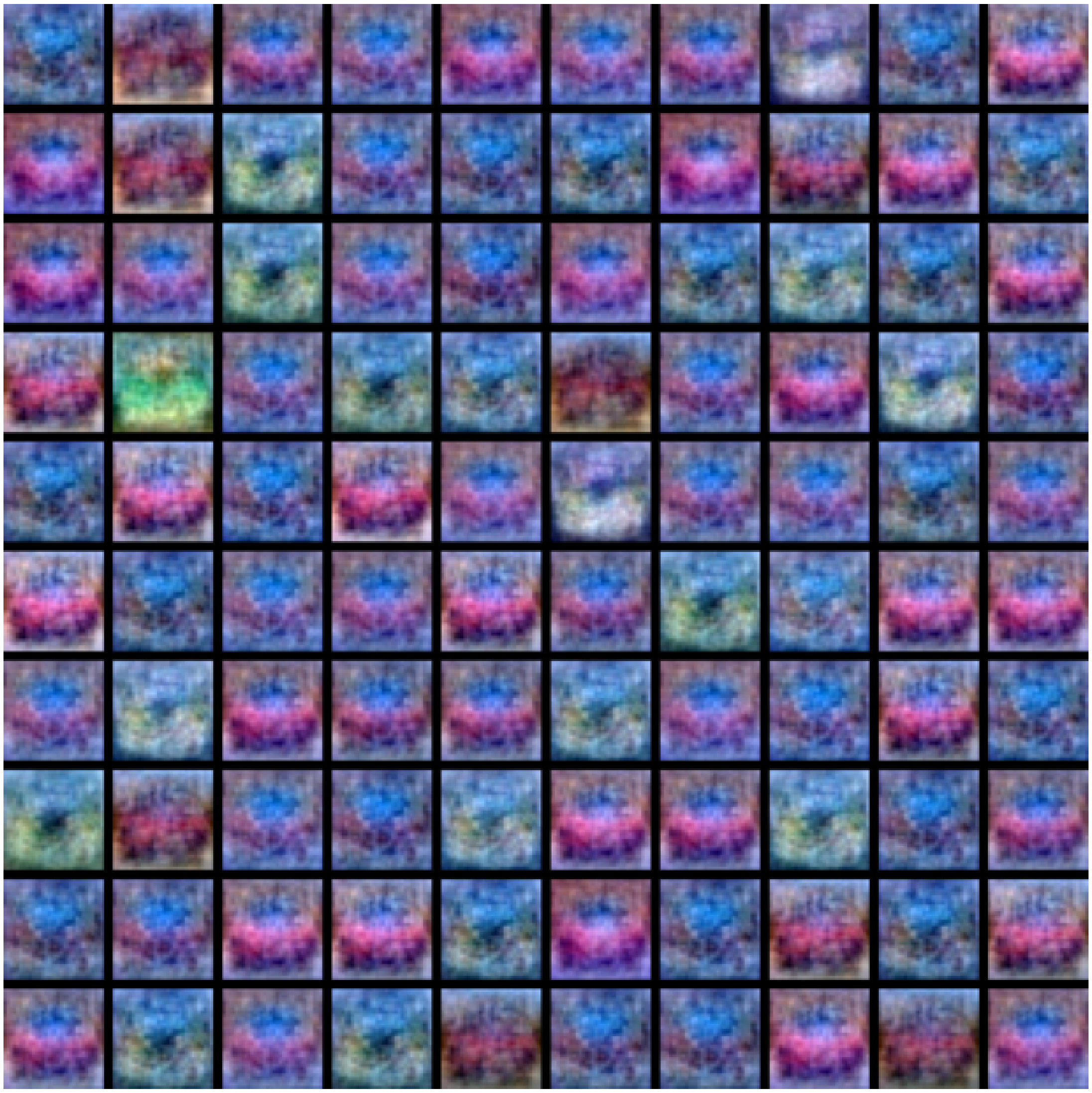
```python
from ece285.utils.vis_utils import visualize_grid

# 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)
    plt.imshow(visualize_grid(W1, padding=3).astype("uint8"))
    plt.gca().axis("off")
    plt.show()


show_net_weights(net)
```



2 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: Increase the learning rate to 0.2, and find the training loss does not decrease, so I search and tried the learning rate between 0.02 and 0.1.
I also tried to increase the hidden layer size to 512, however, it seems to be overfitting.
I also noticed that there is a gap between the training and validation accuracy, so I tried to add weight decay to 0.015.
Finally, I noticed the training loss is still decreasing for 200 epoch, so I increased the epoch to 400.

[16]:

```
input_size = 3072
hidden_size = 300 # 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 tuned_model():
    # np.random.seed(0) # No need to fix the seed here
    l1 = Linear(input_size, hidden_size)
    # l2 = Linear(hidden_size, hidden_size)
    l3 = Linear(hidden_size, num_classes)

    r1 = ReLU()
    # r2 = ReLU()
    softmax = Softmax()
    return Sequential([l1, r1,  l3, softmax])

# Initialize the dataset with the dataloader class
dataset = DataLoader(x_train, y_train, x_val, y_val, x_test, y_test)
best_net = tuned_model()
optim = SGD(best_net, lr=0.065, weight_decay=0.015)
loss_func = CrossEntropyLoss()
epoch = 350  # (Hyper-parameter)
batch_size = 256# (Reduce the batch size if your computer is unable to handle␣
 ↪it)

# Initialize the trainer class by passing the above modules
trainer = Trainer(
    dataset, optim, best_net, loss_func, epoch, batch_size, validate_interval=5
)
```

```python
# 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 Average Loss: 2.301753
Validate Acc: 0.104
Epoch Average Loss: 2.296516
Epoch Average Loss: 2.282466
Epoch Average Loss: 2.248706
Epoch Average Loss: 2.216719
Epoch Average Loss: 2.190613
Validate Acc: 0.140
Epoch Average Loss: 2.178611
Epoch Average Loss: 2.166060
Epoch Average Loss: 2.151628
Epoch Average Loss: 2.143922
Epoch Average Loss: 2.139498
Validate Acc: 0.168
Epoch Average Loss: 2.127751
Epoch Average Loss: 2.126133
Epoch Average Loss: 2.103754
Epoch Average Loss: 2.103363
Epoch Average Loss: 2.081593
Validate Acc: 0.176
Epoch Average Loss: 2.071344
Epoch Average Loss: 2.046820
Epoch Average Loss: 2.043875
Epoch Average Loss: 2.027386
Epoch Average Loss: 2.040150
Validate Acc: 0.276
Epoch Average Loss: 2.037707
Epoch Average Loss: 2.015576
Epoch Average Loss: 2.021603
Epoch Average Loss: 2.001865
Epoch Average Loss: 1.983514
Validate Acc: 0.276
Epoch Average Loss: 1.999110
Epoch Average Loss: 2.007294
Epoch Average Loss: 1.956605
Epoch Average Loss: 1.940943
Epoch Average Loss: 1.938132
Validate Acc: 0.284
Epoch Average Loss: 1.923839
Epoch Average Loss: 1.918498
Epoch Average Loss: 1.926222
Epoch Average Loss: 1.932101
Epoch Average Loss: 1.908986
Validate Acc: 0.260
Epoch Average Loss: 1.903900
Epoch Average Loss: 1.905333
Epoch Average Loss: 1.871296
Epoch Average Loss: 1.878767
Epoch Average Loss: 1.868555
Validate Acc: 0.292
Epoch Average Loss: 1.876370
Epoch Average Loss: 1.844241
Epoch Average Loss: 1.879067
Epoch Average Loss: 1.853410
Epoch Average Loss: 1.814960
Validate Acc: 0.328
Epoch Average Loss: 1.838543
Epoch Average Loss: 1.830557
Epoch Average Loss: 1.814447
Epoch Average Loss: 1.799894
Epoch Average Loss: 1.837435
Validate Acc: 0.316
Epoch Average Loss: 1.806856
Epoch Average Loss: 1.801321
Epoch Average Loss: 1.776798
Epoch Average Loss: 1.782245
Epoch Average Loss: 1.780860
Validate Acc: 0.308
Epoch Average Loss: 1.779238
Epoch Average Loss: 1.782210
Epoch Average Loss: 1.782986
Epoch Average Loss: 1.768374
Epoch Average Loss: 1.764355
Validate Acc: 0.368
Epoch Average Loss: 1.761973
Epoch Average Loss: 1.758296
Epoch Average Loss: 1.753897
Epoch Average Loss: 1.767010
Epoch Average Loss: 1.744071
Validate Acc: 0.368
Epoch Average Loss: 1.729908
Epoch Average Loss: 1.741825
Epoch Average Loss: 1.723569
Epoch Average Loss: 1.755336
Epoch Average Loss: 1.741392
Validate Acc: 0.320
Epoch Average Loss: 1.707641
Epoch Average Loss: 1.746871
Epoch Average Loss: 1.722200
Epoch Average Loss: 1.732470
Epoch Average Loss: 1.710167
Validate Acc: 0.316
Epoch Average Loss: 1.729676
Epoch Average Loss: 1.731321
Epoch Average Loss: 1.709032
Epoch Average Loss: 1.723418
Epoch Average Loss: 1.724400
Validate Acc: 0.368
Epoch Average Loss: 1.695499
Epoch Average Loss: 1.695465
Epoch Average Loss: 1.719601
Epoch Average Loss: 1.713343
Epoch Average Loss: 1.685659
Validate Acc: 0.316
Epoch Average Loss: 1.718871
Epoch Average Loss: 1.667153
Epoch Average Loss: 1.716438
Epoch Average Loss: 1.680555
Epoch Average Loss: 1.687059
Validate Acc: 0.364
Epoch Average Loss: 1.684935
Epoch Average Loss: 1.701476
Epoch Average Loss: 1.669628
Epoch Average Loss: 1.689408
Epoch Average Loss: 1.668413
Validate Acc: 0.360
Epoch Average Loss: 1.670627
Epoch Average Loss: 1.669474
Epoch Average Loss: 1.648149
Epoch Average Loss: 1.668217
Epoch Average Loss: 1.661556
Validate Acc: 0.328
Epoch Average Loss: 1.660184
Epoch Average Loss: 1.672420
Epoch Average Loss: 1.668235
Epoch Average Loss: 1.624360
Epoch Average Loss: 1.665213
Validate Acc: 0.396
Epoch Average Loss: 1.657199
Epoch Average Loss: 1.684161
Epoch Average Loss: 1.652947
Epoch Average Loss: 1.663393
Epoch Average Loss: 1.641019
Validate Acc: 0.388
Epoch Average Loss: 1.618663
Epoch Average Loss: 1.655396
Epoch Average Loss: 1.630275
Epoch Average Loss: 1.647604
Epoch Average Loss: 1.651459
Validate Acc: 0.344
Epoch Average Loss: 1.655329
Epoch Average Loss: 1.613166
Epoch Average Loss: 1.624466
Epoch Average Loss: 1.636580
Epoch Average Loss: 1.629607
Validate Acc: 0.296
Epoch Average Loss: 1.641421
Epoch Average Loss: 1.607866
Epoch Average Loss: 1.634251
Epoch Average Loss: 1.607163
Epoch Average Loss: 1.655783
Validate Acc: 0.408
Epoch Average Loss: 1.608675
Epoch Average Loss: 1.613215
Epoch Average Loss: 1.622214
Epoch Average Loss: 1.577627
Epoch Average Loss: 1.594959
Validate Acc: 0.312
Epoch Average Loss: 1.622612
Epoch Average Loss: 1.609518
Epoch Average Loss: 1.623246
Epoch Average Loss: 1.609658
Epoch Average Loss: 1.618266
Validate Acc: 0.400
Epoch Average Loss: 1.609498
Epoch Average Loss: 1.570585
Epoch Average Loss: 1.569558
Epoch Average Loss: 1.613508
Epoch Average Loss: 1.614992
Validate Acc: 0.380
Epoch Average Loss: 1.609732
Epoch Average Loss: 1.576295
Epoch Average Loss: 1.606699
Epoch Average Loss: 1.580100
Epoch Average Loss: 1.594790
Validate Acc: 0.364
Epoch Average Loss: 1.582620
Epoch Average Loss: 1.618606
Epoch Average Loss: 1.591107
Epoch Average Loss: 1.616387
Epoch Average Loss: 1.612133
Validate Acc: 0.368
Epoch Average Loss: 1.561637
Epoch Average Loss: 1.587011
Epoch Average Loss: 1.539004
Epoch Average Loss: 1.592941
Epoch Average Loss: 1.582667
Validate Acc: 0.360
Epoch Average Loss: 1.589324
Epoch Average Loss: 1.545207
Epoch Average Loss: 1.567754
Epoch Average Loss: 1.585496
Epoch Average Loss: 1.562687
Validate Acc: 0.376
Epoch Average Loss: 1.579881
Epoch Average Loss: 1.545406
Epoch Average Loss: 1.584983
Epoch Average Loss: 1.597885
Epoch Average Loss: 1.589558
Validate Acc: 0.368
Epoch Average Loss: 1.565554
Epoch Average Loss: 1.574760
Epoch Average Loss: 1.570246
Epoch Average Loss: 1.536783
Epoch Average Loss: 1.566466
Validate Acc: 0.404
Epoch Average Loss: 1.568723
Epoch Average Loss: 1.558358
Epoch Average Loss: 1.562357
Epoch Average Loss: 1.557687
Epoch Average Loss: 1.543685
Validate Acc: 0.408
Epoch Average Loss: 1.571142
Epoch Average Loss: 1.583448
Epoch Average Loss: 1.590934
Epoch Average Loss: 1.580021
Epoch Average Loss: 1.543568
Validate Acc: 0.424
Epoch Average Loss: 1.550869
Epoch Average Loss: 1.549791
Epoch Average Loss: 1.578133
Epoch Average Loss: 1.583584
Epoch Average Loss: 1.534206
Validate Acc: 0.416
Epoch Average Loss: 1.542186
Epoch Average Loss: 1.587810
Epoch Average Loss: 1.551319
Epoch Average Loss: 1.529754
Epoch Average Loss: 1.523737
Validate Acc: 0.380
Epoch Average Loss: 1.577863
Epoch Average Loss: 1.543801
Epoch Average Loss: 1.546237
Epoch Average Loss: 1.494706
Epoch Average Loss: 1.541451
Validate Acc: 0.368
Epoch Average Loss: 1.536401
Epoch Average Loss: 1.522190
Epoch Average Loss: 1.548335
Epoch Average Loss: 1.548021
Epoch Average Loss: 1.524689
Validate Acc: 0.428
Epoch Average Loss: 1.543772
Epoch Average Loss: 1.528681
Epoch Average Loss: 1.531962
Epoch Average Loss: 1.519973
Epoch Average Loss: 1.523594
Validate Acc: 0.356
Epoch Average Loss: 1.560731
Epoch Average Loss: 1.525214
Epoch Average Loss: 1.531816
Epoch Average Loss: 1.523027
Epoch Average Loss: 1.561042
Validate Acc: 0.368
Epoch Average Loss: 1.531487
Epoch Average Loss: 1.542539
Epoch Average Loss: 1.530839
Epoch Average Loss: 1.546576
Epoch Average Loss: 1.523077
Validate Acc: 0.360
Epoch Average Loss: 1.566358
Epoch Average Loss: 1.531781
Epoch Average Loss: 1.514562
Epoch Average Loss: 1.510652
Epoch Average Loss: 1.537120
Validate Acc: 0.376
Epoch Average Loss: 1.564607
Epoch Average Loss: 1.504075
Epoch Average Loss: 1.547519
Epoch Average Loss: 1.522866
Epoch Average Loss: 1.499864
Validate Acc: 0.424
Epoch Average Loss: 1.544818
Epoch Average Loss: 1.521793
Epoch Average Loss: 1.518819
Epoch Average Loss: 1.526504
Epoch Average Loss: 1.501296
Validate Acc: 0.412
Epoch Average Loss: 1.513195
Epoch Average Loss: 1.530199
Epoch Average Loss: 1.500367
Epoch Average Loss: 1.509301
Epoch Average Loss: 1.498574
Validate Acc: 0.416
Epoch Average Loss: 1.511133
Epoch Average Loss: 1.502791
Epoch Average Loss: 1.532824
Epoch Average Loss: 1.477778
Epoch Average Loss: 1.499101
Validate Acc: 0.424
Epoch Average Loss: 1.462172
Epoch Average Loss: 1.528518
Epoch Average Loss: 1.551203
Epoch Average Loss: 1.513319
Epoch Average Loss: 1.501783
Validate Acc: 0.428
Epoch Average Loss: 1.525013
Epoch Average Loss: 1.500367
Epoch Average Loss: 1.521106
Epoch Average Loss: 1.472512
Epoch Average Loss: 1.512831
Validate Acc: 0.392
Epoch Average Loss: 1.518030
Epoch Average Loss: 1.495580
Epoch Average Loss: 1.464169
Epoch Average Loss: 1.492667
Epoch Average Loss: 1.495597
Validate Acc: 0.404
Epoch Average Loss: 1.474860
Epoch Average Loss: 1.497158
Epoch Average Loss: 1.507914
Epoch Average Loss: 1.503335
Epoch Average Loss: 1.473537
Validate Acc: 0.348
Epoch Average Loss: 1.495819
Epoch Average Loss: 1.499314
Epoch Average Loss: 1.468144
Epoch Average Loss: 1.485549
Epoch Average Loss: 1.499831
Validate Acc: 0.384
Epoch Average Loss: 1.485386
Epoch Average Loss: 1.532145
Epoch Average Loss: 1.498870
Epoch Average Loss: 1.498958
Epoch Average Loss: 1.483886
Validate Acc: 0.420
Epoch Average Loss: 1.483339
Epoch Average Loss: 1.502875
Epoch Average Loss: 1.480103
Epoch Average Loss: 1.523304
Epoch Average Loss: 1.564988
Validate Acc: 0.396
Epoch Average Loss: 1.497646
Epoch Average Loss: 1.512994
Epoch Average Loss: 1.492358
Epoch Average Loss: 1.509864
Epoch Average Loss: 1.503724
Validate Acc: 0.384
Epoch Average Loss: 1.483003
Epoch Average Loss: 1.506733
Epoch Average Loss: 1.452618
Epoch Average Loss: 1.503709
Epoch Average Loss: 1.492359
Validate Acc: 0.324
Epoch Average Loss: 1.511281
Epoch Average Loss: 1.492560
Epoch Average Loss: 1.506510
Epoch Average Loss: 1.494733
Epoch Average Loss: 1.472759
Validate Acc: 0.408
Epoch Average Loss: 1.486197
Epoch Average Loss: 1.477468
Epoch Average Loss: 1.487132
Epoch Average Loss: 1.477691
Epoch Average Loss: 1.460404
Validate Acc: 0.428
Epoch Average Loss: 1.472240
Epoch Average Loss: 1.492068
Epoch Average Loss: 1.481140
Epoch Average Loss: 1.452341
Epoch Average Loss: 1.498336
Validate Acc: 0.424
Epoch Average Loss: 1.500220
Epoch Average Loss: 1.473680
Epoch Average Loss: 1.456573
Epoch Average Loss: 1.481263
Epoch Average Loss: 1.463041
Validate Acc: 0.412
Epoch Average Loss: 1.473260
Epoch Average Loss: 1.484846
Epoch Average Loss: 1.482217
Epoch Average Loss: 1.463598
Epoch Average Loss: 1.474441
Validate Acc: 0.380
Epoch Average Loss: 1.470875
Epoch Average Loss: 1.464585
Epoch Average Loss: 1.479795
Epoch Average Loss: 1.484590
Epoch Average Loss: 1.497924
Validate Acc: 0.408
Epoch Average Loss: 1.456147
Epoch Average Loss: 1.471779
Epoch Average Loss: 1.455268
Epoch Average Loss: 1.469981
Epoch Average Loss: 1.485015
Validate Acc: 0.372
Epoch Average Loss: 1.448128
Epoch Average Loss: 1.478286
Epoch Average Loss: 1.458857
Epoch Average Loss: 1.465127
Epoch Average Loss: 1.457996
Validate Acc: 0.392
Epoch Average Loss: 1.481294
Epoch Average Loss: 1.466897
Epoch Average Loss: 1.453162
Epoch Average Loss: 1.471954
Epoch Average Loss: 1.452563
Validate Acc: 0.420
Epoch Average Loss: 1.462648
Epoch Average Loss: 1.468751
Epoch Average Loss: 1.466728
Epoch Average Loss: 1.464488
Epoch Average Loss: 1.464801
Validate Acc: 0.416
Epoch Average Loss: 1.460953
Epoch Average Loss: 1.495575
Epoch Average Loss: 1.440437
Epoch Average Loss: 1.460605
Epoch Average Loss: 1.482682
Validate Acc: 0.356
Epoch Average Loss: 1.440953
Epoch Average Loss: 1.504439
Epoch Average Loss: 1.520626
Epoch Average Loss: 1.461737
Epoch Average Loss: 1.465475
Validate Acc: 0.432
Epoch Average Loss: 1.465432
Epoch Average Loss: 1.445655
Epoch Average Loss: 1.467228
Epoch Average Loss: 1.483421
```

[17]:

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

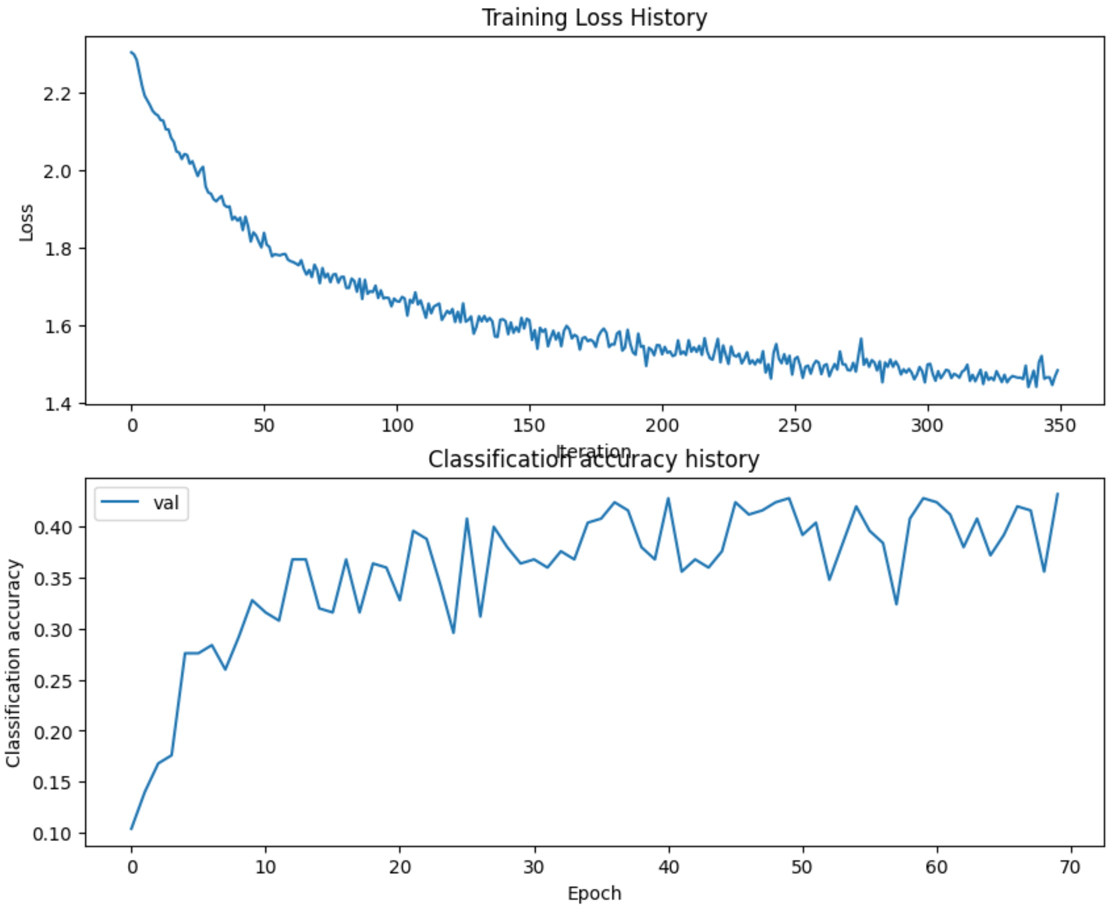
```
out_train = best_net.predict(x_train)
acc = get_classification_accuracy(out_train, y_train)
print("Training acc: ", acc)
out_val = best_net.predict(x_val)
acc = get_classification_accuracy(out_val, y_val)
print("Validation acc: ", acc)
```

```
Training acc:  0.478
Validation acc:  0.412
```

[18]:

```
# 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()
```



[19]:

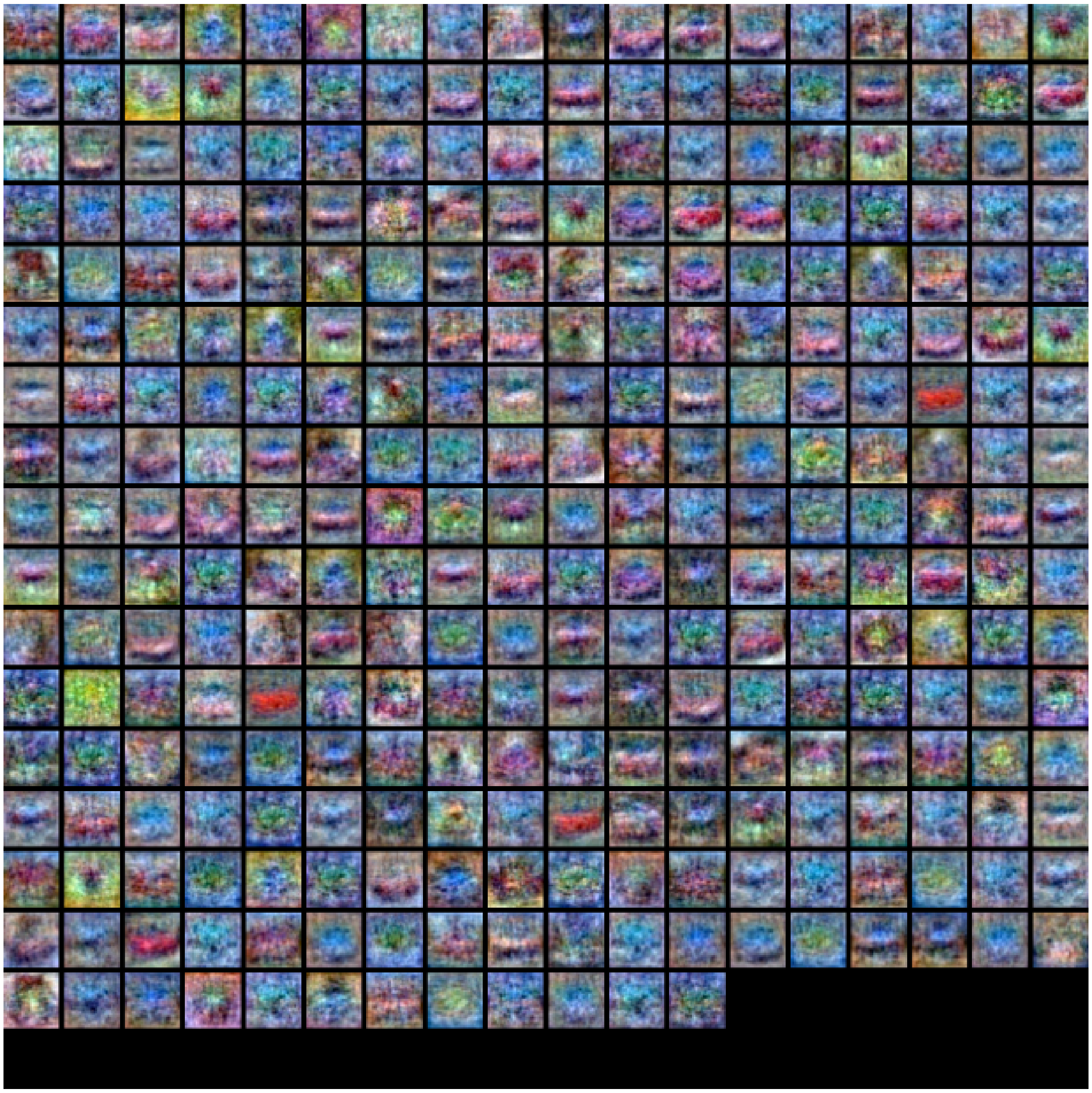
```
from ece285.utils.vis_utils import visualize_grid

# 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)
    plt.imshow(visualize_grid(W1, padding=3).astype("uint8"))
    plt.gca().axis("off")
    plt.show()


show_net_weights(best_net)
```



[20]:

```
best_net_hyperparams = [0.065, 0.015, 350, 300]
################################################################################
# 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.          ␣
 ↪#
#                                                                               ␣
 ↪#
# You are now free to test different combinations of hyperparameters to build   ␣
 ↪#
# various models and test them according to the above plots and visualization   ␣
 ↪#


# 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]
##################################################################################

pass
```

[21]:

```
# TODO: Plot the training_error and validation_accuracy of the best network (5%)

# TODO: visualize the weights of the best network (5%)
```

3 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%.

[22]:

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

```
Test accuracy:  0.356
```

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: The correct options to help decrease the gap between training and testing accuracy in a neural network classifier are: #### Your Explanation: Train on a larger dataset - Training on a larger dataset can help the model generalize better to unseen data by providing more examples from which it can learn. This reduces overfitting, which is a common reason for the discrepancy between training and testing performance. Increase the regularization strength - Regularization techniques such as L1 or L2 regularization, or using dropout, help prevent the model from fitting too closely to the noise in the training data. By increasing the regularization strength, the model is encouraged to develop simpler, more generalizable patterns, thus reducing overfitting. Option 2, adding more hidden units, is generally not recommended to decrease overfitting, as it can actually lead to more complex models that overfit the training data even more. Thus, it might increase the gap between training and testing accuracy rather than decrease it.

Therefore, the best options are 1 and 3.

[15]: