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 [2]:
```
# 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-ipytho
%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)
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
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 [4]:
```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"]
```

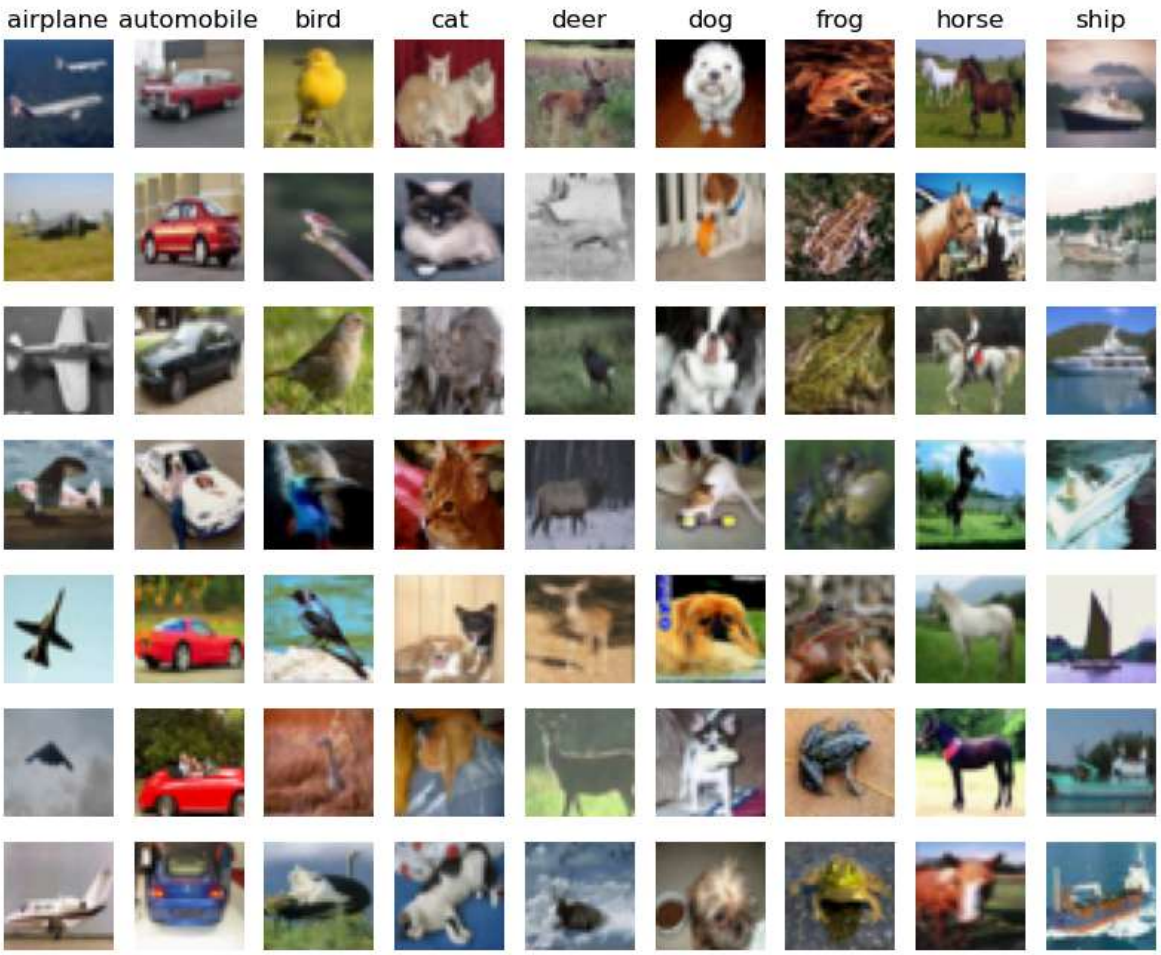
In [5]:
```python
# Import more utilies and the layers you have implemented
from ece285.layers.sequential import Sequential
from ece285.layers.linear import Linear
from ece285.layers.relu import ReLU
from ece285.layers.softmax import Softmax
from ece285.layers.loss_func import CrossEntropyLoss
from ece285.utils.optimizer import SGD
from ece285.utils.dataset import DataLoader
from ece285.utils.trainer import Trainer
```

Visualize some examples from the dataset.

In [6]:
```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,
)
```



Initialize the model

In [7]:
```
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 A
# A 2 layer neural network with non-linearities can approximate any function, gi
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 [8]:
```
# 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
```

In [9]:
```
# Initialize the trainer class by passing the above modules
trainer = Trainer(
    dataset, optim, net, loss_func, epoch, batch_size, validate_interval=3
)
```

In [10]:
```
# Call the trainer function we have already implemented for you. This trains the
# hyper-parameters. It follows the same procedure as in the last ipython noteboo
train_error, validation_accuracy = trainer.train()
```

```
Epoch Average Loss: 2.302537
Validate Acc: 0.084
Epoch Average Loss: 2.302357
Epoch Average Loss: 2.302150
Epoch Average Loss: 2.301861
Validate Acc: 0.108
Epoch Average Loss: 2.301441
Epoch Average Loss: 2.300843
Epoch Average Loss: 2.299991
Validate Acc: 0.100
Epoch Average Loss: 2.298838
Epoch Average Loss: 2.297339
Epoch Average Loss: 2.295524
Validate Acc: 0.084
Epoch Average Loss: 2.293417
Epoch Average Loss: 2.290900
Epoch Average Loss: 2.287854
Validate Acc: 0.084
Epoch Average Loss: 2.283963
Epoch Average Loss: 2.278960
Epoch Average Loss: 2.272802
Validate Acc: 0.096
Epoch Average Loss: 2.265813
Epoch Average Loss: 2.258376
Epoch Average Loss: 2.250734
Validate Acc: 0.104
Epoch Average Loss: 2.243073
Epoch Average Loss: 2.235618
Epoch Average Loss: 2.228595
Validate Acc: 0.120
Epoch Average Loss: 2.221881
Epoch Average Loss: 2.215906
Epoch Average Loss: 2.210026
Validate Acc: 0.124
Epoch Average Loss: 2.204771
Epoch Average Loss: 2.199859
Epoch Average Loss: 2.195318
Validate Acc: 0.136
Epoch Average Loss: 2.191022
Epoch Average Loss: 2.187192
Epoch Average Loss: 2.183369
Validate Acc: 0.140
Epoch Average Loss: 2.179967
Epoch Average Loss: 2.176628
Epoch Average Loss: 2.173439
Validate Acc: 0.144
Epoch Average Loss: 2.169992
Epoch Average Loss: 2.167227
Epoch Average Loss: 2.164560
Validate Acc: 0.140
Epoch Average Loss: 2.161789
Epoch Average Loss: 2.159188
Epoch Average Loss: 2.156697
Validate Acc: 0.144
Epoch Average Loss: 2.154358
Epoch Average Loss: 2.152055
Epoch Average Loss: 2.150305
Validate Acc: 0.148
Epoch Average Loss: 2.147913
Epoch Average Loss: 2.145961
```

```
Epoch Average Loss: 2.144044
Validate Acc: 0.148
Epoch Average Loss: 2.142169
Epoch Average Loss: 2.140343
Epoch Average Loss: 2.138655
Validate Acc: 0.152
Epoch Average Loss: 2.136682
Epoch Average Loss: 2.135321
Epoch Average Loss: 2.133533
Validate Acc: 0.152
Epoch Average Loss: 2.132009
Epoch Average Loss: 2.130416
Epoch Average Loss: 2.129122
Validate Acc: 0.152
Epoch Average Loss: 2.127431
Epoch Average Loss: 2.125651
Epoch Average Loss: 2.124258
Validate Acc: 0.168
Epoch Average Loss: 2.122828
Epoch Average Loss: 2.121352
Epoch Average Loss: 2.119664
Validate Acc: 0.156
Epoch Average Loss: 2.118078
Epoch Average Loss: 2.116450
Epoch Average Loss: 2.114914
Validate Acc: 0.172
Epoch Average Loss: 2.112643
Epoch Average Loss: 2.110706
Epoch Average Loss: 2.109698
Validate Acc: 0.180
Epoch Average Loss: 2.107314
Epoch Average Loss: 2.104992
Epoch Average Loss: 2.103629
Validate Acc: 0.172
Epoch Average Loss: 2.101204
Epoch Average Loss: 2.098445
Epoch Average Loss: 2.096262
Validate Acc: 0.180
Epoch Average Loss: 2.093518
Epoch Average Loss: 2.091024
Epoch Average Loss: 2.088212
Validate Acc: 0.184
Epoch Average Loss: 2.085072
Epoch Average Loss: 2.082427
Epoch Average Loss: 2.079305
Validate Acc: 0.228
Epoch Average Loss: 2.076111
Epoch Average Loss: 2.073149
Epoch Average Loss: 2.069611
Validate Acc: 0.224
Epoch Average Loss: 2.066161
Epoch Average Loss: 2.063133
Epoch Average Loss: 2.060298
Validate Acc: 0.228
Epoch Average Loss: 2.056917
Epoch Average Loss: 2.053934
Epoch Average Loss: 2.050681
Validate Acc: 0.228
Epoch Average Loss: 2.047618
Epoch Average Loss: 2.044920
```

```
Epoch Average Loss: 2.041692
Validate Acc: 0.248
Epoch Average Loss: 2.038495
Epoch Average Loss: 2.036630
Epoch Average Loss: 2.033501
Validate Acc: 0.248
Epoch Average Loss: 2.031365
Epoch Average Loss: 2.028272
Epoch Average Loss: 2.025925
Validate Acc: 0.264
Epoch Average Loss: 2.023761
Epoch Average Loss: 2.021007
Epoch Average Loss: 2.018870
Validate Acc: 0.244
Epoch Average Loss: 2.016123
Epoch Average Loss: 2.014135
Epoch Average Loss: 2.011489
Validate Acc: 0.268
Epoch Average Loss: 2.009680
Epoch Average Loss: 2.008137
Epoch Average Loss: 2.005660
Validate Acc: 0.264
Epoch Average Loss: 2.003601
Epoch Average Loss: 2.001001
Epoch Average Loss: 1.999229
Validate Acc: 0.264
Epoch Average Loss: 1.997444
Epoch Average Loss: 1.995157
Epoch Average Loss: 1.993021
Validate Acc: 0.272
Epoch Average Loss: 1.991517
Epoch Average Loss: 1.989319
Epoch Average Loss: 1.987260
Validate Acc: 0.276
Epoch Average Loss: 1.985221
Epoch Average Loss: 1.983659
Epoch Average Loss: 1.981298
Validate Acc: 0.284
Epoch Average Loss: 1.978509
Epoch Average Loss: 1.976127
Epoch Average Loss: 1.973707
Validate Acc: 0.292
Epoch Average Loss: 1.972199
Epoch Average Loss: 1.969201
Epoch Average Loss: 1.966288
Validate Acc: 0.288
Epoch Average Loss: 1.963509
Epoch Average Loss: 1.961164
Epoch Average Loss: 1.958564
Validate Acc: 0.296
Epoch Average Loss: 1.955302
Epoch Average Loss: 1.951548
Epoch Average Loss: 1.948651
Validate Acc: 0.300
Epoch Average Loss: 1.945868
Epoch Average Loss: 1.942914
Epoch Average Loss: 1.939747
Validate Acc: 0.268
Epoch Average Loss: 1.936792
Epoch Average Loss: 1.934389
```

```
Epoch Average Loss: 1.931108
Validate Acc: 0.292
Epoch Average Loss: 1.927453
Epoch Average Loss: 1.926925
Epoch Average Loss: 1.924990
Validate Acc: 0.284
Epoch Average Loss: 1.921991
Epoch Average Loss: 1.919087
Epoch Average Loss: 1.917199
Validate Acc: 0.284
Epoch Average Loss: 1.914289
Epoch Average Loss: 1.911488
Epoch Average Loss: 1.910590
Validate Acc: 0.288
Epoch Average Loss: 1.907297
Epoch Average Loss: 1.906047
Epoch Average Loss: 1.902912
Validate Acc: 0.304
Epoch Average Loss: 1.901049
Epoch Average Loss: 1.899232
Epoch Average Loss: 1.896585
Validate Acc: 0.296
Epoch Average Loss: 1.894317
Epoch Average Loss: 1.892472
Epoch Average Loss: 1.889595
Validate Acc: 0.316
Epoch Average Loss: 1.887675
Epoch Average Loss: 1.885574
Epoch Average Loss: 1.884415
Validate Acc: 0.300
Epoch Average Loss: 1.882466
Epoch Average Loss: 1.879865
Epoch Average Loss: 1.877374
Validate Acc: 0.288
Epoch Average Loss: 1.876498
Epoch Average Loss: 1.872763
Epoch Average Loss: 1.872313
Validate Acc: 0.312
Epoch Average Loss: 1.868820
Epoch Average Loss: 1.867578
Epoch Average Loss: 1.865027
Validate Acc: 0.320
Epoch Average Loss: 1.863167
Epoch Average Loss: 1.861850
Epoch Average Loss: 1.859246
Validate Acc: 0.316
Epoch Average Loss: 1.857995
Epoch Average Loss: 1.855283
Epoch Average Loss: 1.853868
Validate Acc: 0.316
Epoch Average Loss: 1.850525
Epoch Average Loss: 1.850173
Epoch Average Loss: 1.847342
Validate Acc: 0.320
Epoch Average Loss: 1.844937
Epoch Average Loss: 1.842297
Epoch Average Loss: 1.841978
Validate Acc: 0.300
Epoch Average Loss: 1.840899
Epoch Average Loss: 1.837400
```

```
Epoch Average Loss: 1.835204
Validate Acc: 0.308
Epoch Average Loss: 1.836128
Epoch Average Loss: 1.832263
Epoch Average Loss: 1.831628
Validate Acc: 0.312
Epoch Average Loss: 1.828670
Epoch Average Loss: 1.825933
Epoch Average Loss: 1.824297
Validate Acc: 0.336
Epoch Average Loss: 1.822668
Epoch Average Loss: 1.822196
Epoch Average Loss: 1.820392
Validate Acc: 0.312
Epoch Average Loss: 1.815700
Epoch Average Loss: 1.816024
Epoch Average Loss: 1.811975
Validate Acc: 0.328
Epoch Average Loss: 1.811493
Epoch Average Loss: 1.808913
Epoch Average Loss: 1.807433
Validate Acc: 0.324
Epoch Average Loss: 1.807012
Epoch Average Loss: 1.804324
Epoch Average Loss: 1.803040
Validate Acc: 0.336
Epoch Average Loss: 1.801166
```

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

In [11]:
```
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:  0.35
Validation acc:  0.316
```

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.

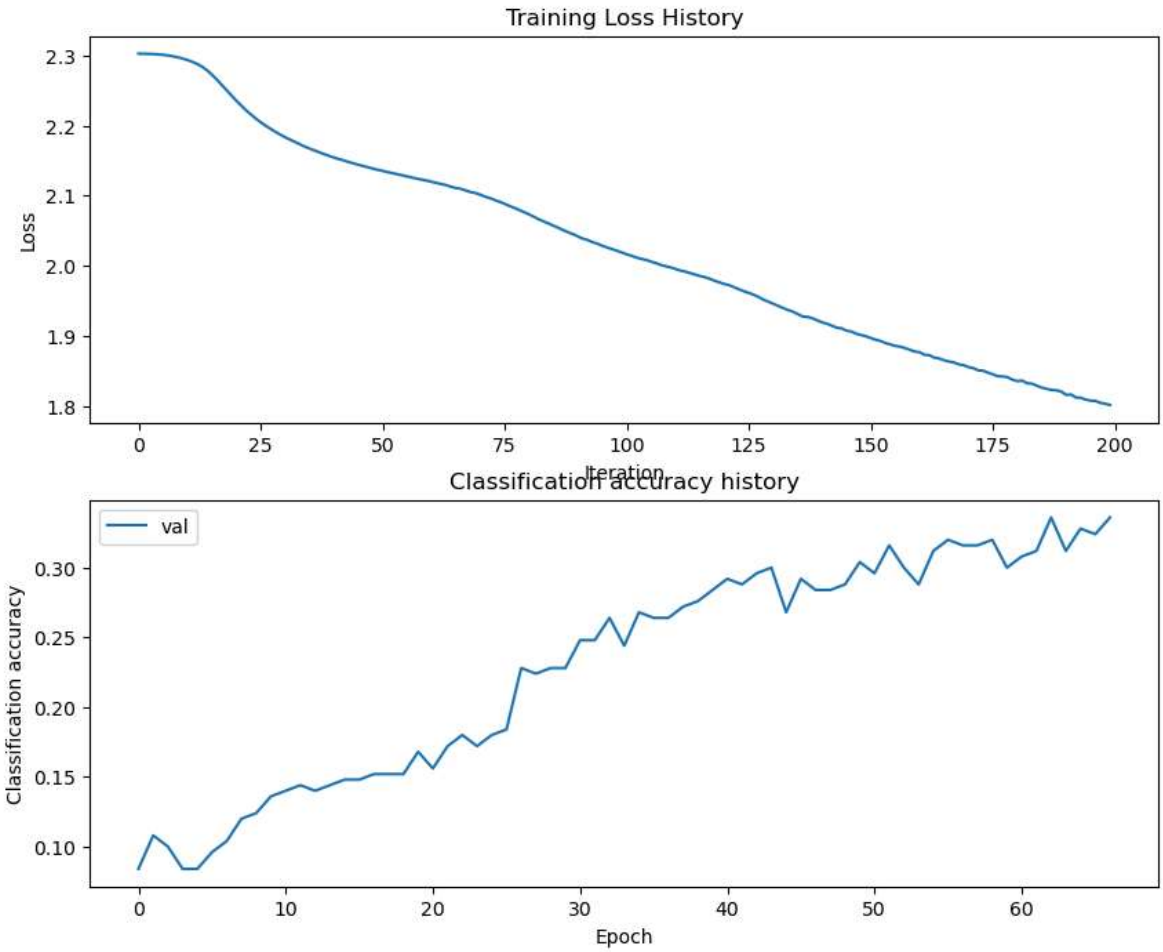
One strategy for getting insight into what's wrong is to plot the training loss function and the validation accuracies during optimization.

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 [12]:
```
# 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 [13]:

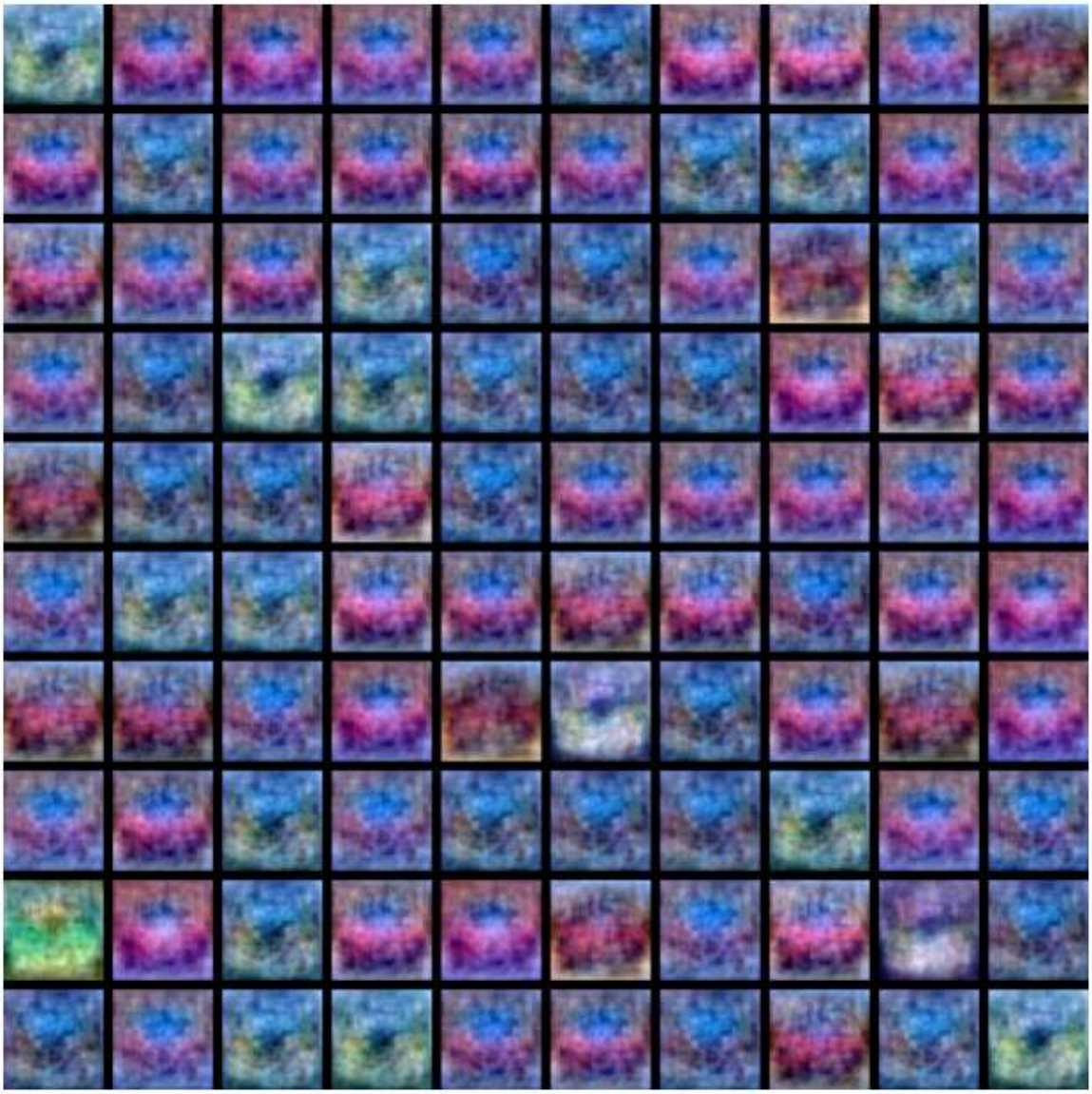
```
from ece285.utils.vis_utils import visualize_grid

# 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)
```



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:

1. Since the loss is decreasing approximate linearly, we can use a higher learning rate to converge faster, therefore I set learning rate to 0.1, a commonly used lr value.
2. There is no gap between the training and validation accuracy, suggesting that the model we used has low capacity. Thus I increase the model size by increasing hidden layer to 300.
3. After tuning above hyperparameters in step 1 and step 2, I found that the training loss converges before 200 epochs. So I set epochs as default 200, and it works well! The validation accuracy goes to 42.8%.

In [20]:
```
best_net_hyperparams = None  # store the best model into this

################################################################################
# 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

input_size = 3072
hidden_size = 300  # Hidden layer size (Hyper-parameter)
num_classes = 10  # Output

# Initialize the dataset with the dataloader class
dataset = DataLoader(x_train, y_train, x_val, y_val, x_test, y_test)
best_net = init_model()
optim = SGD(best_net, lr = 0.1, 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

trainer = Trainer(
    dataset, optim, best_net, loss_func, epoch, batch_size, validate_interval=3
)
train_error, validation_accuracy = trainer.train()

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)
```

```python
acc = get_classification_accuracy(out_val, y_val)
print("Validation acc: ", acc)

# 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]
################################################################################
best_net_hyperparams = [0.01, 0.01, 200, 100] # default
best_net_hyperparams = [0.1, 0.01, 200, 300]
```

```
Epoch Average Loss: 2.300071
Validate Acc: 0.084
Epoch Average Loss: 2.268512
Epoch Average Loss: 2.214933
Epoch Average Loss: 2.179944
Validate Acc: 0.156
Epoch Average Loss: 2.167844
Epoch Average Loss: 2.150929
Epoch Average Loss: 2.141386
Validate Acc: 0.212
Epoch Average Loss: 2.112754
Epoch Average Loss: 2.092109
Epoch Average Loss: 2.068048
Validate Acc: 0.232
Epoch Average Loss: 2.057241
Epoch Average Loss: 2.037002
Epoch Average Loss: 2.045416
Validate Acc: 0.292
Epoch Average Loss: 2.041739
Epoch Average Loss: 2.012062
Epoch Average Loss: 1.976290
Validate Acc: 0.264
Epoch Average Loss: 2.000353
Epoch Average Loss: 1.983939
Epoch Average Loss: 1.953881
Validate Acc: 0.316
Epoch Average Loss: 1.929595
Epoch Average Loss: 1.928706
Epoch Average Loss: 1.909803
Validate Acc: 0.296
Epoch Average Loss: 1.904079
Epoch Average Loss: 1.877448
Epoch Average Loss: 1.873545
Validate Acc: 0.276
Epoch Average Loss: 1.897851
Epoch Average Loss: 1.834607
Epoch Average Loss: 1.846808
Validate Acc: 0.352
Epoch Average Loss: 1.838307
Epoch Average Loss: 1.826593
Epoch Average Loss: 1.827618
Validate Acc: 0.356
Epoch Average Loss: 1.814641
Epoch Average Loss: 1.803634
Epoch Average Loss: 1.787345
Validate Acc: 0.328
Epoch Average Loss: 1.782202
Epoch Average Loss: 1.803197
Epoch Average Loss: 1.760699
Validate Acc: 0.348
Epoch Average Loss: 1.786442
Epoch Average Loss: 1.754840
Epoch Average Loss: 1.777949
Validate Acc: 0.384
Epoch Average Loss: 1.792728
Epoch Average Loss: 1.749026
Epoch Average Loss: 1.735431
Validate Acc: 0.360
Epoch Average Loss: 1.749346
Epoch Average Loss: 1.754655
```

```
Epoch Average Loss: 1.755395
Validate Acc: 0.332
Epoch Average Loss: 1.800520
Epoch Average Loss: 1.733904
Epoch Average Loss: 1.765536
Validate Acc: 0.344
Epoch Average Loss: 1.727875
Epoch Average Loss: 1.741139
Epoch Average Loss: 1.724926
Validate Acc: 0.380
Epoch Average Loss: 1.737598
Epoch Average Loss: 1.691864
Epoch Average Loss: 1.725540
Validate Acc: 0.380
Epoch Average Loss: 1.695348
Epoch Average Loss: 1.707106
Epoch Average Loss: 1.690490
Validate Acc: 0.408
Epoch Average Loss: 1.673739
Epoch Average Loss: 1.669382
Epoch Average Loss: 1.723950
Validate Acc: 0.404
Epoch Average Loss: 1.706270
Epoch Average Loss: 1.655192
Epoch Average Loss: 1.692525
Validate Acc: 0.340
Epoch Average Loss: 1.681834
Epoch Average Loss: 1.655527
Epoch Average Loss: 1.656362
Validate Acc: 0.368
Epoch Average Loss: 1.693376
Epoch Average Loss: 1.669411
Epoch Average Loss: 1.665840
Validate Acc: 0.348
Epoch Average Loss: 1.688954
Epoch Average Loss: 1.694815
Epoch Average Loss: 1.663867
Validate Acc: 0.372
Epoch Average Loss: 1.655381
Epoch Average Loss: 1.639057
Epoch Average Loss: 1.669988
Validate Acc: 0.384
Epoch Average Loss: 1.677361
Epoch Average Loss: 1.626184
Epoch Average Loss: 1.623320
Validate Acc: 0.388
Epoch Average Loss: 1.643714
Epoch Average Loss: 1.626226
Epoch Average Loss: 1.632185
Validate Acc: 0.368
Epoch Average Loss: 1.664252
Epoch Average Loss: 1.617582
Epoch Average Loss: 1.593336
Validate Acc: 0.392
Epoch Average Loss: 1.589056
Epoch Average Loss: 1.623359
Epoch Average Loss: 1.685273
Validate Acc: 0.416
Epoch Average Loss: 1.645217
Epoch Average Loss: 1.618991
```

```
Epoch Average Loss: 1.606476
Validate Acc: 0.324
Epoch Average Loss: 1.614198
Epoch Average Loss: 1.606709
Epoch Average Loss: 1.606458
Validate Acc: 0.388
Epoch Average Loss: 1.612916
Epoch Average Loss: 1.625931
Epoch Average Loss: 1.614406
Validate Acc: 0.404
Epoch Average Loss: 1.619226
Epoch Average Loss: 1.616951
Epoch Average Loss: 1.599097
Validate Acc: 0.388
Epoch Average Loss: 1.593995
Epoch Average Loss: 1.563755
Epoch Average Loss: 1.605891
Validate Acc: 0.400
Epoch Average Loss: 1.597959
Epoch Average Loss: 1.617890
Epoch Average Loss: 1.592449
Validate Acc: 0.380
Epoch Average Loss: 1.550429
Epoch Average Loss: 1.611632
Epoch Average Loss: 1.577012
Validate Acc: 0.396
Epoch Average Loss: 1.555697
Epoch Average Loss: 1.581042
Epoch Average Loss: 1.624416
Validate Acc: 0.384
Epoch Average Loss: 1.592271
Epoch Average Loss: 1.600728
Epoch Average Loss: 1.546571
Validate Acc: 0.428
Epoch Average Loss: 1.610162
Epoch Average Loss: 1.546903
Epoch Average Loss: 1.578859
Validate Acc: 0.392
Epoch Average Loss: 1.608172
Epoch Average Loss: 1.603163
Epoch Average Loss: 1.564900
Validate Acc: 0.372
Epoch Average Loss: 1.586613
Epoch Average Loss: 1.534824
Epoch Average Loss: 1.561804
Validate Acc: 0.380
Epoch Average Loss: 1.548545
Epoch Average Loss: 1.511561
Epoch Average Loss: 1.601804
Validate Acc: 0.412
Epoch Average Loss: 1.592309
Epoch Average Loss: 1.541224
Epoch Average Loss: 1.544430
Validate Acc: 0.432
Epoch Average Loss: 1.548974
Epoch Average Loss: 1.541045
Epoch Average Loss: 1.571556
Validate Acc: 0.356
Epoch Average Loss: 1.533488
Epoch Average Loss: 1.549456
```

```
Epoch Average Loss: 1.552967
Validate Acc: 0.420
Epoch Average Loss: 1.557000
Epoch Average Loss: 1.556592
Epoch Average Loss: 1.517465
Validate Acc: 0.424
Epoch Average Loss: 1.529638
Epoch Average Loss: 1.551317
Epoch Average Loss: 1.543507
Validate Acc: 0.344
Epoch Average Loss: 1.523296
Epoch Average Loss: 1.609245
Epoch Average Loss: 1.575333
Validate Acc: 0.432
Epoch Average Loss: 1.548461
Epoch Average Loss: 1.490865
Epoch Average Loss: 1.542712
Validate Acc: 0.424
Epoch Average Loss: 1.511250
Epoch Average Loss: 1.517506
Epoch Average Loss: 1.531557
Validate Acc: 0.324
Epoch Average Loss: 1.596256
Epoch Average Loss: 1.530521
Epoch Average Loss: 1.550064
Validate Acc: 0.364
Epoch Average Loss: 1.509095
Epoch Average Loss: 1.531194
Epoch Average Loss: 1.480860
Validate Acc: 0.364
Epoch Average Loss: 1.558873
Epoch Average Loss: 1.589553
Epoch Average Loss: 1.517868
Validate Acc: 0.388
Epoch Average Loss: 1.524039
Epoch Average Loss: 1.493848
Epoch Average Loss: 1.499255
Validate Acc: 0.408
Epoch Average Loss: 1.542604
Epoch Average Loss: 1.537824
Epoch Average Loss: 1.501010
Validate Acc: 0.408
Epoch Average Loss: 1.641612
Epoch Average Loss: 1.568103
Epoch Average Loss: 1.509431
Validate Acc: 0.328
Epoch Average Loss: 1.502914
Epoch Average Loss: 1.517265
Epoch Average Loss: 1.522744
Validate Acc: 0.404
Epoch Average Loss: 1.527083
Epoch Average Loss: 1.505994
Epoch Average Loss: 1.499278
Validate Acc: 0.316
Epoch Average Loss: 1.593333
Epoch Average Loss: 1.494689
Epoch Average Loss: 1.550106
Validate Acc: 0.416
Epoch Average Loss: 1.547482
Epoch Average Loss: 1.485010
```

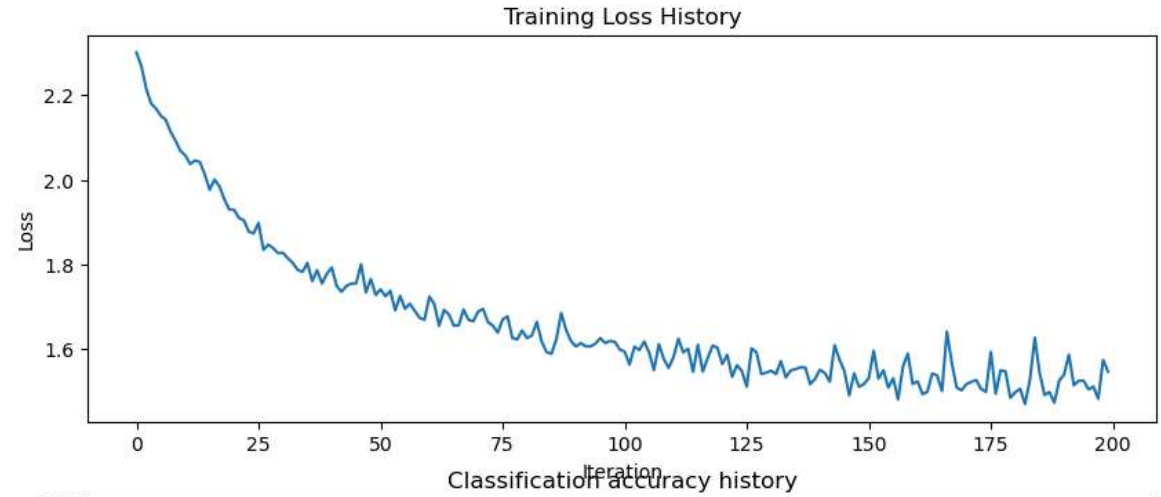
```
Epoch Average Loss: 1.498139
Validate Acc: 0.360
Epoch Average Loss: 1.506377
Epoch Average Loss: 1.470451
Epoch Average Loss: 1.526824
Validate Acc: 0.436
Epoch Average Loss: 1.626705
Epoch Average Loss: 1.543492
Epoch Average Loss: 1.491672
Validate Acc: 0.376
Epoch Average Loss: 1.498718
Epoch Average Loss: 1.473770
Epoch Average Loss: 1.525536
Validate Acc: 0.408
Epoch Average Loss: 1.539913
Epoch Average Loss: 1.586935
Epoch Average Loss: 1.514925
Validate Acc: 0.440
Epoch Average Loss: 1.525352
Epoch Average Loss: 1.525501
Epoch Average Loss: 1.505125
Validate Acc: 0.368
Epoch Average Loss: 1.511678
Epoch Average Loss: 1.482767
Epoch Average Loss: 1.574502
Validate Acc: 0.412
Epoch Average Loss: 1.546934
Training acc:  0.4982
Validation acc:  0.428
```

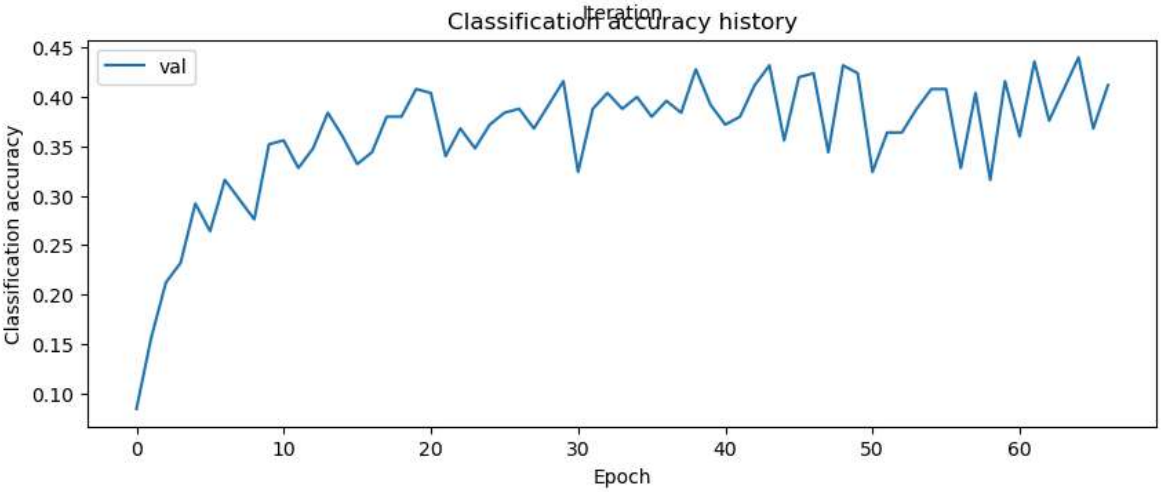
In [21]:

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

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


Training Loss History
Loss
2.2
2.0
1.8
1.6
0
25
50
75
100
125
150
175
200
Iteration


Classification accuracy history
val
Classification accuracy
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
10
20
30
40
50
60
Epoch

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 [23]:
```
test_acc = (best_net.predict(x_test) == y_test).mean()
print("Test accuracy: ", test_acc)
```

```
Test accuracy:  0.362
```

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:

Both 1 and 3 are correct.

Your Explanation:

1. Training on a larger dataset is correct because more data helps the model generalize better by exposing it to more variations.
2. Adding more hidden units increases model capacity, which will worsen overfitting (increase the gap between the training and testing accuracy)
3. Increase the regularization strength is correct. Regularization such as L2 weight decay penalizes complex models with large weights. This prevents overfitting.