CV - Assignment 4

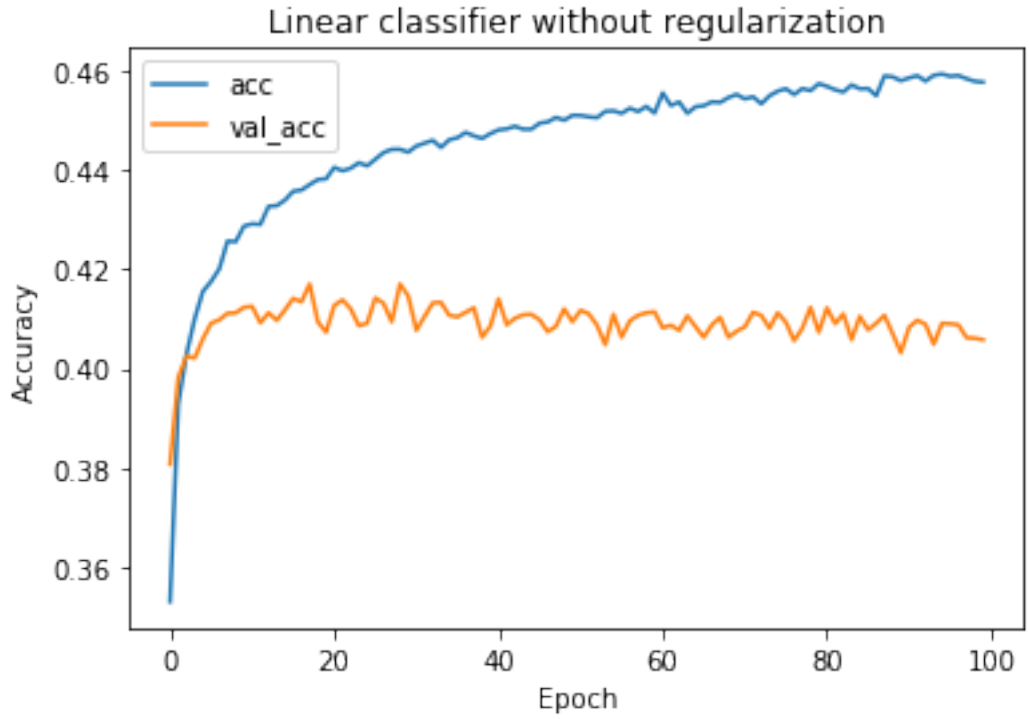
Eitan Kosman - 312146145
eitan.k@campus.technion.ac.il

January 9, 2020

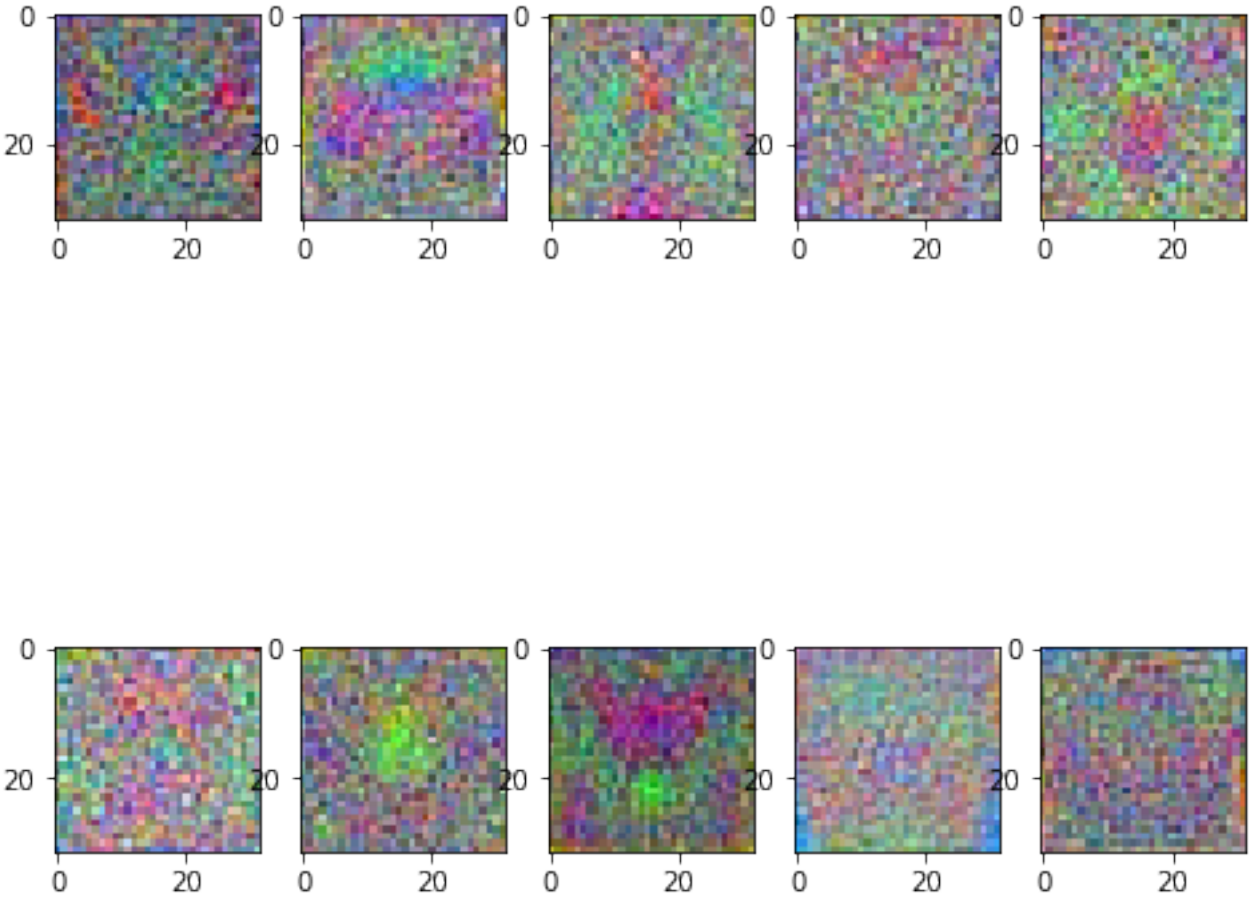
1 Train a linear classifier using a softmax activation on the training set without any regularization

1.1 Plot the train and test accuracy vs. num of epochs

Based on the following graph, I can clearly determine that there's a lack of generalization. Moreover, the accuracy is quiet bad since the dataset isn't linearly separable.



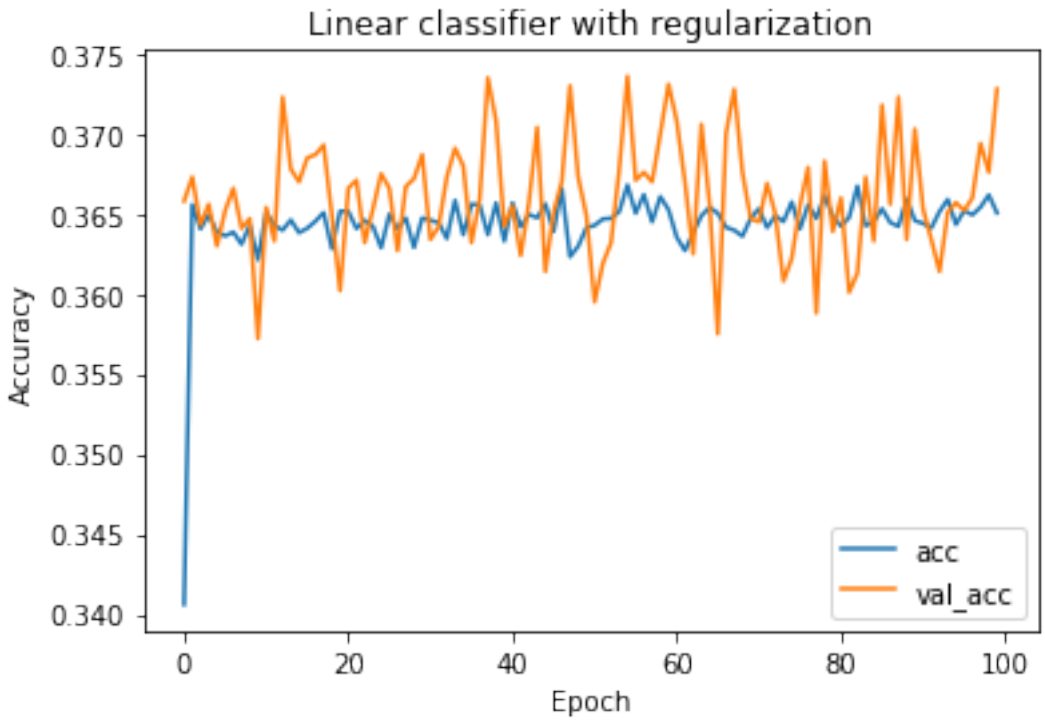
1.2 Plot the rows of the linear classifier



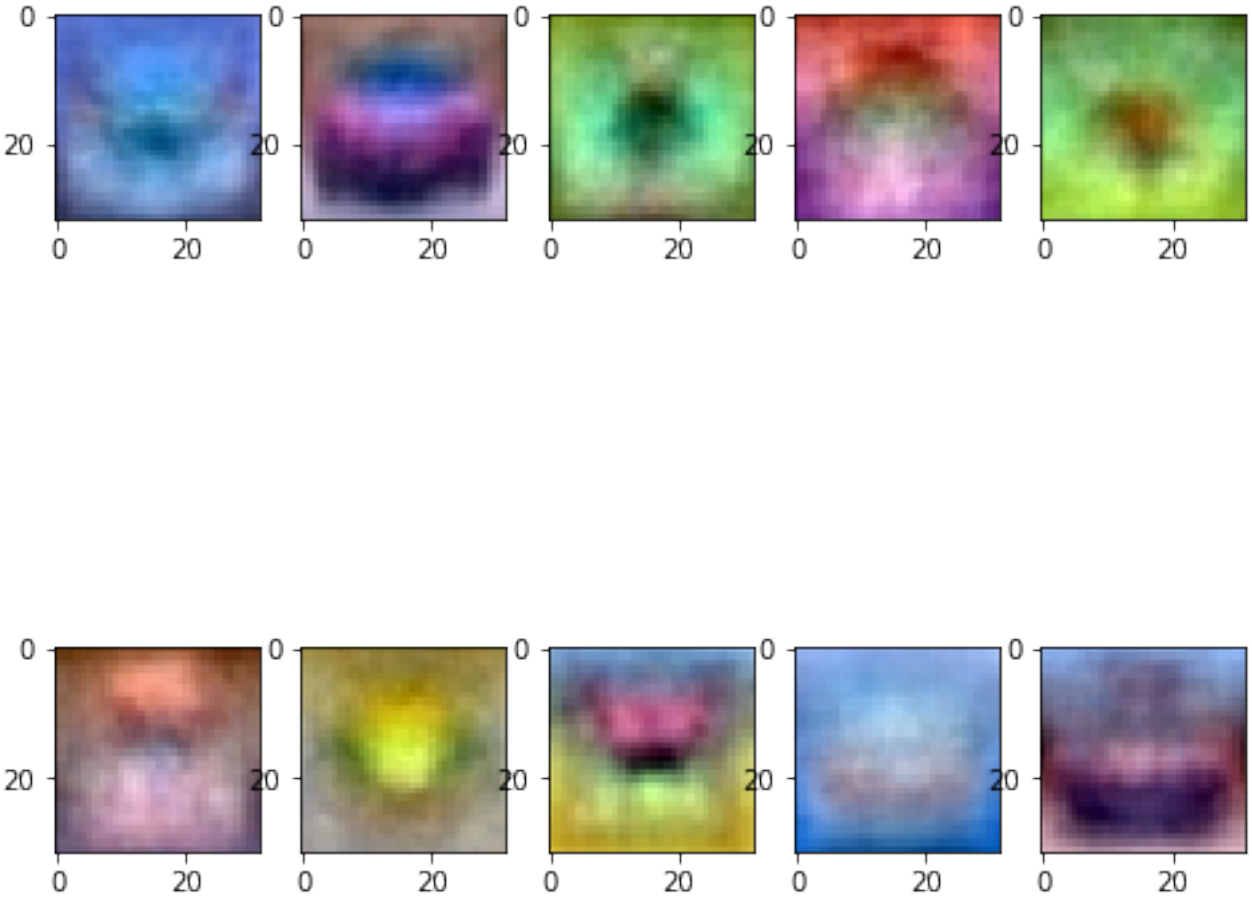
2 Train another linear classifier, this time use L2 regularization with a multiplier of 0.1

Seems like the regularization closed the difference between the train and test accuracy. However, I stay with a single-layer, thus the model has a very high bias and is incapable of representing the data too well.

2.1 Plot the train and test accuracy vs. num of epochs



2.2 Plot the rows of the linear classifier



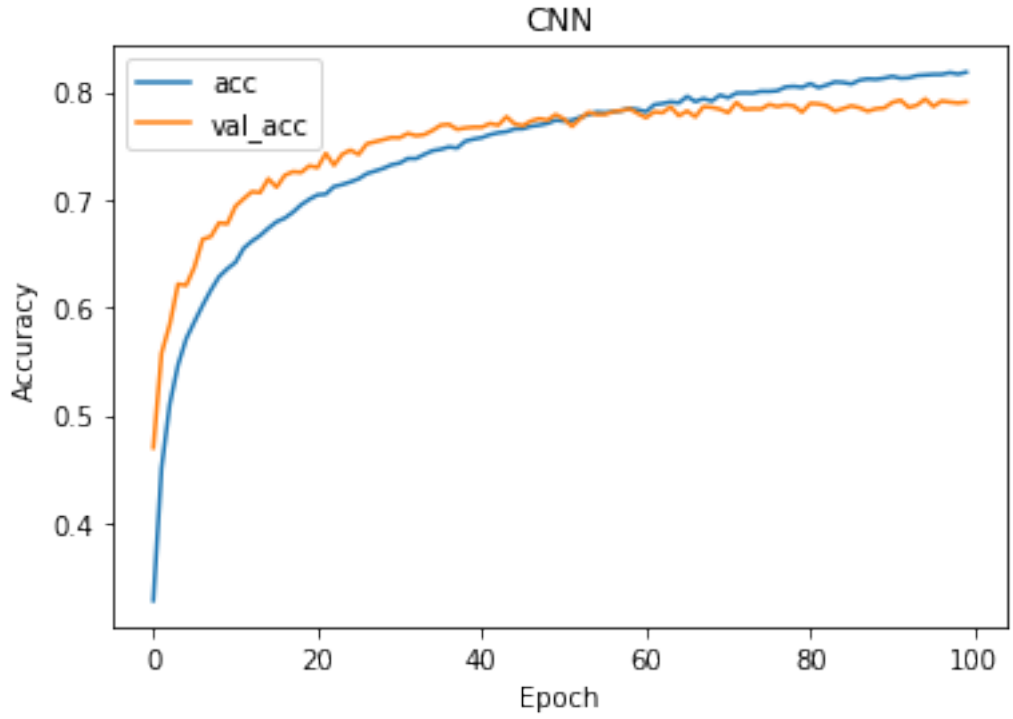
This time, I observe meaningful patterns in the resulting dictionary. The L_2 regularization penalizes the entries of our matrix. It has a square form, thus the bigger the entry, the larger the penalty. This leads to small values with small variance, which results in natural-looking images when looking at the dictionary.

3 Train a convolutional neural network

The architecture I used for this task is in the following picture. The full description can be found in the code. The activation between the layers is *Relu* and the last activation is $Softmax$. Notice that the architecture has $<$ 50K parameters and still achieves good accuracy rate!

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
batch_normalization_1 (Batch (None, 30, 30, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 44)        5676
_________________________________________________________________
batch_normalization_2 (Batch (None, 30, 30, 44)        176
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 44)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 48)        8496
_________________________________________________________________
batch_normalization_3 (Batch (None, 14, 14, 48)        192
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 56)        10808
_________________________________________________________________
batch_normalization_4 (Batch (None, 14, 14, 56)        224
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 56)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 6, 64)          14400
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
batch_normalization_5 (Batch (None, 6, 6, 64)          256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 2, 2, 32)          8224
_________________________________________________________________
batch_normalization_6 (Batch (None, 2, 2, 32)          128
_________________________________________________________________
global_average_pooling2d_1 ( (None, 32)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
=================================================================
Total params: 49,934
Trainable params: 49,382
Non-trainable params: 552
```

3.1 Plot the train and test accuracy vs. num of epochs



4 Find 2 images on the web and use them as input to your trained model

The 2 images I've chosen are:

Figure 1: image1

Figure 2: image2

The probabilities from the prediction results are:

- image1 - [[9.9699378e-01 9.5261133e-04 1.5820772e-03 1.2705749e-04 9.2898117e-05 9.5264031e-06 8.0205900e-06 1.7407530e-05 2.0618134e-04 1.0513910e-05]]
- image2 - [[1.9292253e-03 9.7336811e-01 6.1348372e-04 2.1071297e-03 3.3282992e-04 8.2015339e-04 1.6128223e-02 9.2006980e-05 8.4367336e-04 3.7650766e-03]]

The actual labels (after $argmax$) are:

image1 - 0

image2 - 1

This means that *image* gets label 0 and *image*2 gets label 1 (counting from 0). After looking in the description of $CIFAR-10$ (https://www.cs.toronto.edu/~kriz/cifar.html), I see that airplanes should be classified as 0 and cars should be classified as 1. This means that both classifications were correct! In addition, notice that the confidence of the model is very high, giving a score of 0.97 to the correct label!