# DL Assignment 1 Bonus

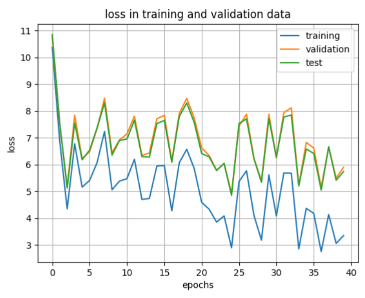
Meinan Gou meinang@kth.se

## 1 Ex 2.1

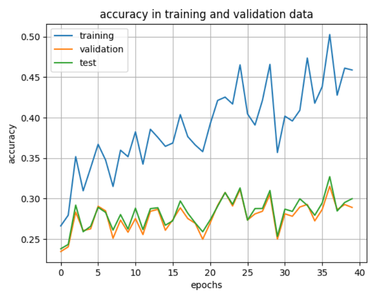
### 1.1 Shuffling at the beginning of each epoch

I defined a function named randomize_data() to shuffle the data and labels.

**Parameters setting 1: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.1**



(a) loss

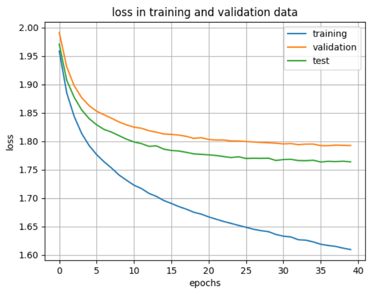


(b) Accuracy

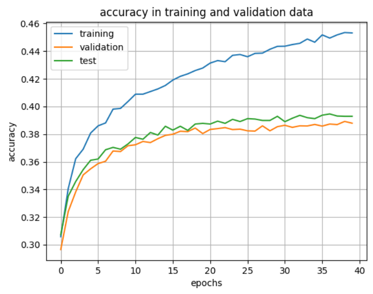
Figure 1: Plots for setting 1 with shuffling

For setting 1, the best test is accuracy=0.3289, which is higher than non-shuffling result 0.3069.

**Parameters setting 2: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) loss

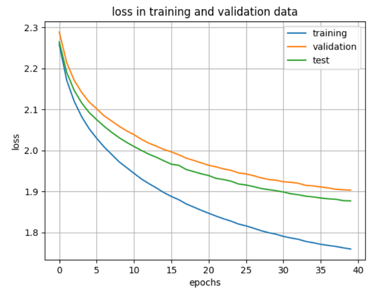


(b) Accuracy

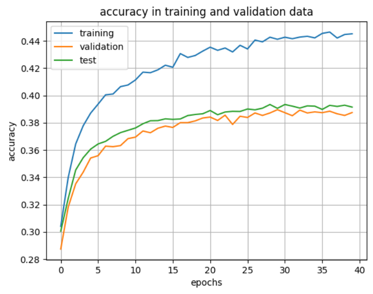
Figure 2: Plots for setting 2 with shuffling

For setting 2, the best test accuracy=0.3961, which is higher than non-shuffling result 0.3898.

**Parameters setting 3: $\lambda$=0.1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) loss

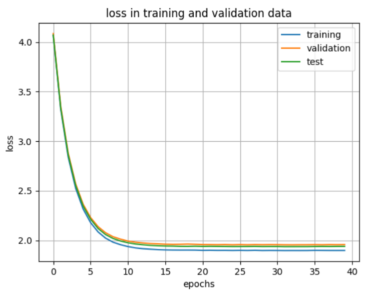


(b) Accuracy

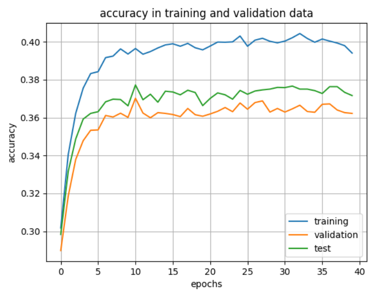
Figure 3: Plots for setting 3 with shuffling

For setting 3, the best test accuracy=0.3949, which is higher than non-shuffling result 0.3924.

**Parameters setting 4: $\lambda$=1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) loss



(b) Accuracy

Figure 4: Plots for setting 4 with shuffling

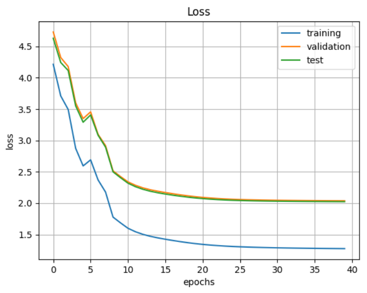
For setting 4, the best test accuracy=0.3764, which is slightly higher than non-shuffling result 0.3763.

In conclusion, shuffling before training at each epoch can generally improve the best test accuracy.

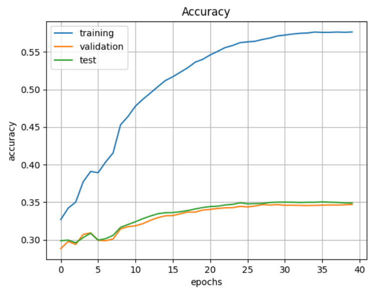
## 1.2 Learning rate decay

For the last three settings, decaying learning rate improves little on the performance, sometimes even makes the case worse. This is because $\eta$ is already small enough (0.001) for these settings. Decaying $\eta$ would reduce the learning ability even more. However, the case for the first setting is quite different. Decaying $\eta$ by a factor 0.9 at the beginning of each epoch improves the test accuracy by 5%.

**$\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.1**



(a) loss



(b) Accuracy

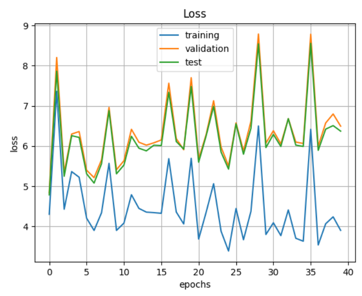
Figure 5: Plots for setting 1 with learning rate decay

The best test accuracy=0.3512, which is much higher than 0.3069.

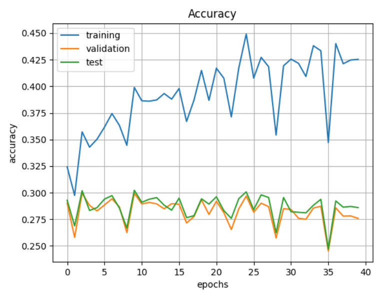
## 1.3 Xavier

Xavier initialization refers to keeping the variance to remain the same as passing through each layer, also known as keeping mean and spread of activations similar in each layer, which concludes that the weight matrix initialization should be a Gaussian distribution with zero mean and variance equals to $\frac{1}{n_{in}}$, where $n_{avg}$ is the number of input neurons. In this assignment, $n_{in} = 3072$. Thus, I set $\sigma = \sqrt{\frac{1}{3072}}$.

**Parameters setting 1: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.1**



(a) Loss

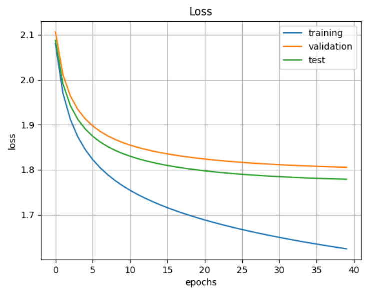


(b) Accuracy

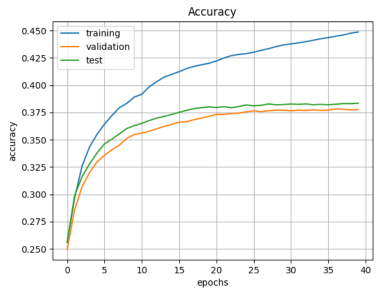
Figure 6: Plots for setting 1 with Xavier

Best test accuracy=0.3023.

**Parameters setting 2: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss

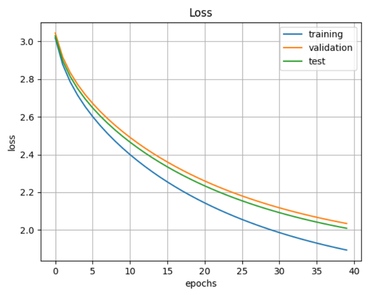


(b) Accuracy

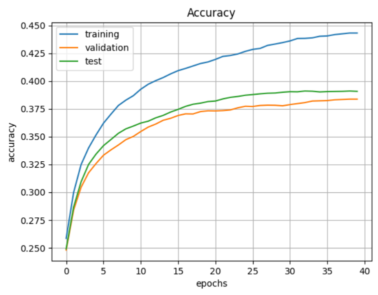
Figure 7: Plots for setting 2 with Xavier

Best test accuracy=0.3835.

**Parameters setting 3: $\lambda$=0.1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss

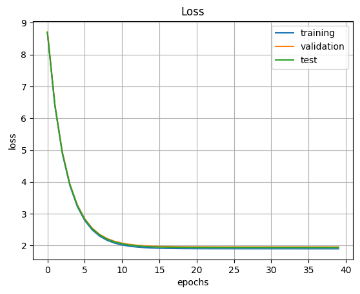


(b) Accuracy

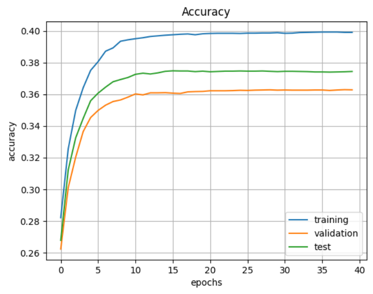
Figure 8: Plots for setting 3 with Xavier

Best test accuracy=0.391.

**Parameters setting 4: $\lambda$=1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss



(b) Accuracy

Figure 9: Plots for setting 4 with Xavier

Best test accuracy=0.3749.

Initializing with Xavier does not really improve the test accuracy. However, it maintains the variance of activation and back-propagated gradients all the way up or down the layers of a network. This effect would be more clear in multi-layer network. In conclusion, using Xavier would stabilize the network and the training process.

## 2 Ex 2.2

SVM multi-class loss is defined as:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i,$$

where

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1), s = Wx_i + b.$$

Gradients:

$$\frac{\partial L_i}{\partial w_{y_i}} = -(\sum_{i \neq y_i} \mathbf{1}(s_j - s_{y_i} + 1))x_i,$$

$$\frac{\partial L_i}{\partial w_j} = -\frac{\partial L_i}{\partial w_{y_i}},$$

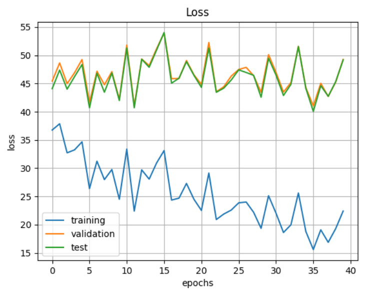
$$\frac{\partial L_i}{\partial b_{y_i}} = -\mathbf{1}(s_j - s_{y_i} + 1),$$

$$\frac{\partial L_i}{\partial b_j} = -\frac{\partial L_i}{\partial b_{y_i}},$$

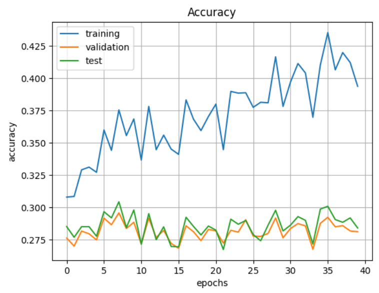
where

$$\mathbf{1}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

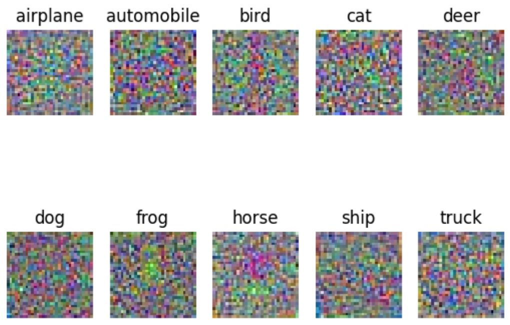
**Parameters setting 1: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.1**



(a) Loss



(b) Accuracy

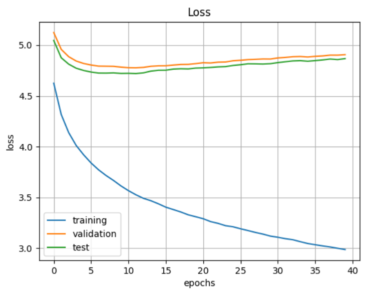


(c) W

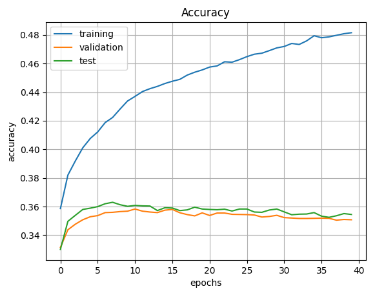
Figure 10: Plots for setting 1 with SVM

Best test accuracy=0.3045, lower than cross-entropy result 0.3069.

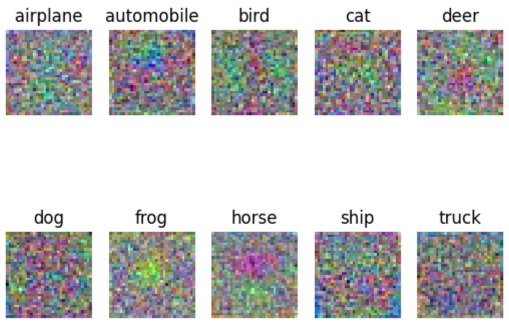
**Parameters setting 2: $\lambda$=0, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss



(b) Accuracy

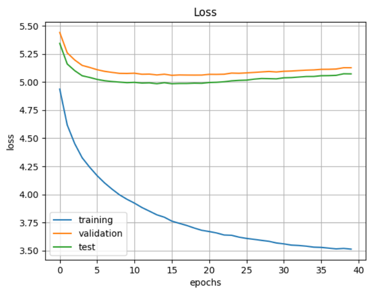


(c) W

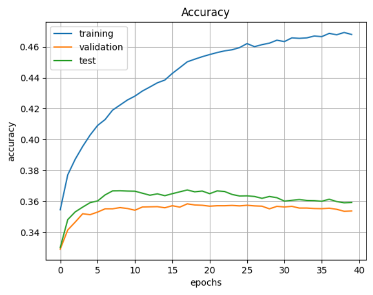
Figure 11: Plots for setting 2 with SVM

Best test accuracy=0.363, lower than cross-entropy result 0.3898.

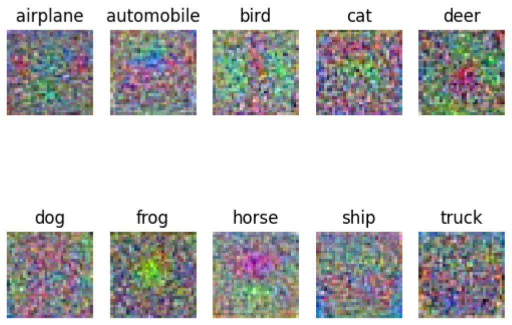
**Parameters setting 3: $\lambda$=0.1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss



(b) Accuracy

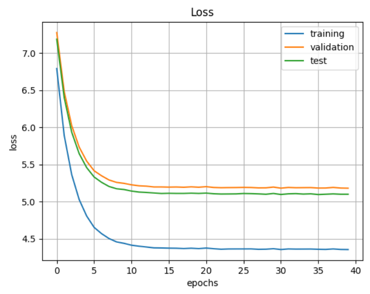


(c) W

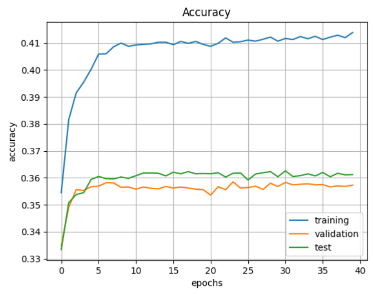
Figure 12: Plots for setting 3 with SVM

Best test accuracy=0.3673, lower than cross-entropy result 0.3924.

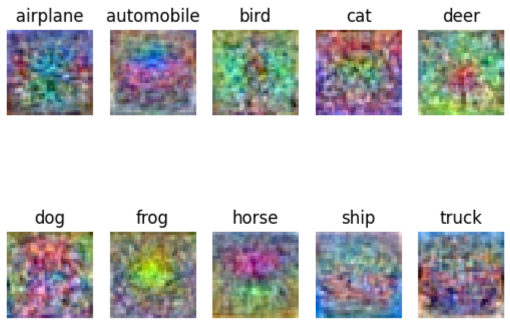
**Parameters setting 4: $\lambda$=1, n_epochs=40, n_batch=100, $\eta$=0.001**



(a) Loss



(b) Accuracy



(c) W

Figure 13: Plots for setting 4 with SVM

Best test accuracy=0.3626, higher than cross-entropy result 0.3763.

Generally, the best test accuracy of SVM is lower than that of cross-entropy. The loss defined by SVM is much higher than that defined by cross-entropy. Also, I can see that the SVM multi-class loss should be regularized by large value of $\lambda$, because the overfitting in Figure 11 and Figure 12 are obvious. For the specific assignment, cross-entropy is a better choice.