Assignment 1 - CIFAR-10 Classification

Martín Bravo, DD2424 Deep Learning, KTH Royal Institute of Technology

April 4, 2025

1 Introduction

In this report, we explore the implementation and evaluation of a single-layer softmax regression model for classifying images from the CIFAR-10 dataset. The primary objective is to understand the behavior of this simple model and analyze its performance under various hyperparameter configurations. We investigate the effects of learning rate and regularization strength on the model's training and generalization capabilities. Additionally, we experiment with techniques such as data augmentation and hyperparameter tuning to improve the model's accuracy. Finally, we compare the softmax regression approach with an alternative method using multiple binary cross-entropy losses to highlight the strengths and limitations of each approach. This study provides insights into the fundamental principles of neural network training and serves as a foundation for more advanced architectures.

2 Dataset and Preprocessing

The CIFAR-10 dataset consists of 32×32 color images belonging to 10 classes. We normalized the input data per feature using the training set mean and standard deviation. The dataset was split into training, validation, and test sets. Below are a few CIFAR-10 examples:

Figure 1: Sample CIFAR-10 images from the dataset

3 Architecture

The architecture of the network is a single-layer softmax regression model. This is one of the simplest forms of neural networks, often referred to as a generalized linear model for classification. The input to the network is a vectorized image of size $d = 32 \times 32 \times 3 = 3072$, where each image is flattened into a one-dimensional vector. The network consists of the following components:

- A weight matrix W of size $K \times d$, where $K = 10$ is the number of classes. Each row of W corresponds to the weights associated with a specific class.

- A bias vector b of size $K \times 1$, which allows the model to shift the decision boundaries independently of the input features.

The forward pass of the network computes the scores for each class as:

$$s = Wx + b$$

where x is the input vector. These scores, s, are then transformed into probabilities using the softmax function:

$$p = \text{SOFTMAX}(s) = \frac{\exp(s)}{\mathbf{1}^T \exp(s)}$$

Here, $\exp(s)$ is applied element-wise, and the denominator ensures that the probabilities sum to 1. This normalization step is crucial for interpreting the output as probabilities.

The predicted class is determined by selecting the class with the highest probability:

$$k^* = \arg \max_{1 \leq k \leq K} p_k$$

This simple architecture allows us to directly map input features to class probabilities using a linear transformation followed by the softmax activation. Despite its simplicity, this model is effective for linearly separable data and serves as a strong baseline for more complex architectures.

The softmax regression model is particularly well-suited for multi-class classification problems, as it generalizes logistic regression to handle multiple classes. However, its performance is limited when the data is not linearly separable, which is why more advanced architectures, such as multi-layer neural networks, are often used in practice. In this assignment, we focus on understanding the behavior of this simple model and its sensitivity to hyperparameters like the learning rate (η) and regularization strength (λ).

4 Results

We conducted four main experiments with varying values of η and λ while keeping $n_{epochs} = 40$ and $batch = 100$.

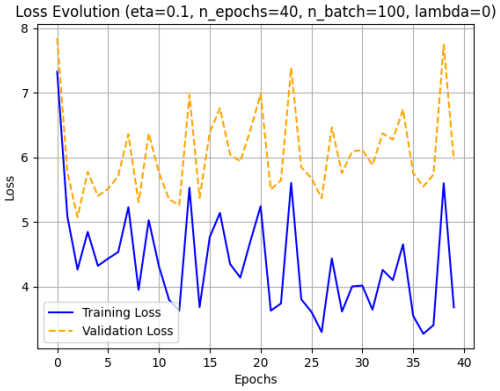
4.1 Effect of the Learning Rate and the Regularization Strength

Figure 2 shows the evolution of training and validation loss for different configurations of learning rate (η) and L2 regularization strength (λ). Table 1 summarizes the final accuracy on train, validation, and test sets for each experiment.

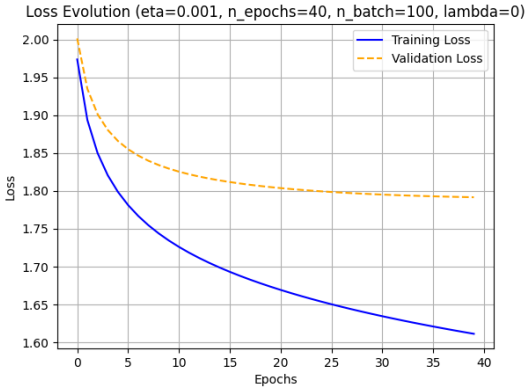
Table 1: Final accuracy (%) for each experiment configuration.

Experiment	$\boldsymbol{\eta}$	$\boldsymbol{\lambda}$	**Train Acc.**	**Val/Test Acc.**
1	0.1	0	42.95%	26.32% / 27.00%
2	0.001	0	45.57%	38.46% / 39.21%
3	0.001	0.1	44.63%	38.62% / 39.30%
4	0.001	1	39.85%	36.32% / 37.55%

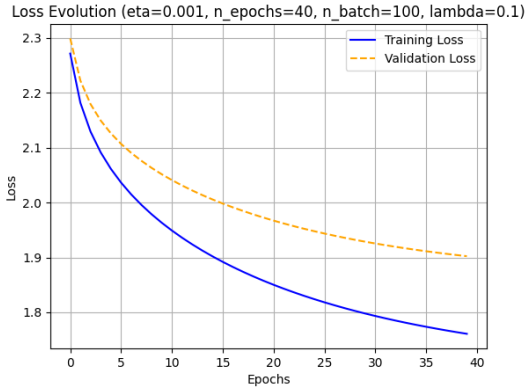
The results indicate that a high learning rate leads to unstable training, with validation accuracy dropping sharply, suggesting poor generalization and divergence. Conversely, a small learning rate ensures stable convergence, resulting in the highest test accuracy with smooth loss curves and effective generalization. Introducing moderate regularization, such as $\lambda = 0.1$, slightly improves validation accuracy while keeping training accuracy nearly unchanged. However, heavy regularization, like $\lambda = 1$, harms generalization. These findings confirm that moderate regularization can enhance generalization, while excessive regularization reduces model capacity. Similarly, an appropriately small learning rate ensures convergence, whereas a large one destabilizes training.



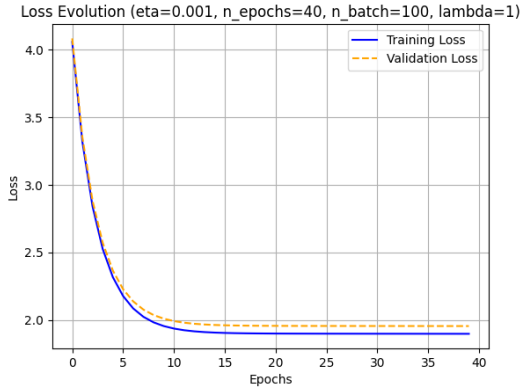
(a) $\eta = 0.1$, $\lambda = 0$



(b) $\eta = 0.001$, $\lambda = 0$

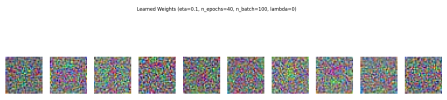


(c) $\eta = 0.001$, $\lambda = 0.1$

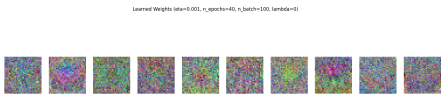


(d) $\eta = 0.001$, $\lambda = 1$

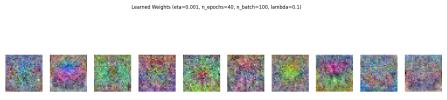
Figure 2: Training and validation loss curves for different hyperparameter configurations.



(a) $\eta = 0.1$, $\lambda = 0$



(b) $\eta = 0.001$, $\lambda = 0$



(c) $\eta = 0.001$, $\lambda = 0.1$

(d) $\eta = 0.001$, $\lambda = 1$

Figure 3: Visualization of the learned weight matrices for each experiment. Each matrix is reshaped into a $32 \times 32 \times 3$ image and normalized for display.

4.2 Improving performance

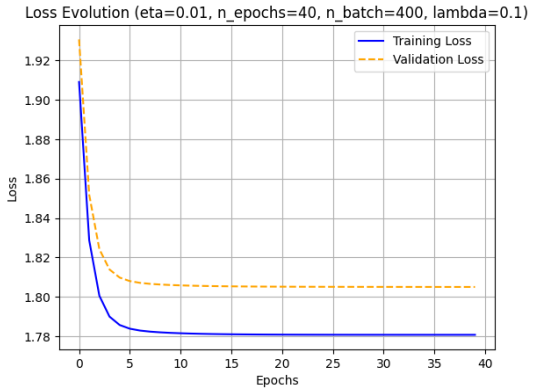
We improved the performance of the network by implementing the following techniques:

- More data samples: We increased the number of training samples by training on the entire CIFAR-10 dataset instead of a subset. This allowed the model to learn from a larger variety of examples, improving its generalization capabilities.

- Data augmentation: We applied random transformations to the training images, such as horizontal flipping to increase the diversity of the training set and reduce overfitting.
- GridSearch: We performed a grid search over the hyperparameters, including learning rate (η), batch size, and L2 regularization strength (λ). This helped us find the optimal combination of hyperparameters for our model:

```
# Define the grid search parameters
eta_values = [0.001, 0.01, 0.1, 1]
n_epochs_values = [40]
n_batch_values = [50, 100, 200, 400]
lam_values = [0, 0.1, 1]
```

We get that the best configuration is $\eta = 0.01$, $n_{epochs} = 40$, $n_{batch} = 400$, and $\lambda = 0.1$. The final accuracy on the test set was 43.20%, achieving an extra 4% accuracy compared to the previous configuration. The training and validation loss curves for this configuration are shown in Figure 4a.



(a) $\eta = 0.01$, $\lambda = 0.1$

4.3 Training with Multiple Binary Cross-Entropy Losses

We also trained the network using *multiple binary cross-entropy* (BCE) loss, replacing the softmax activation and cross-entropy loss with sigmoid activations and a BCE loss applied independently to each class. This approach allows the model to treat each class independently, which can be beneficial in certain scenarios such as multi-label classification.

The multiple binary cross-entropy loss is defined as:

$$\ell_{\text{bce}}(\mathbf{x}, \mathbf{y}) = -\frac{1}{K} \sum_{k=1}^{K} [(1 - y_k) \log(1 - p_k) + y_k \log(p_k)]$$

where $y_k \in \{0, 1\}$ is the ground-truth label for class k, and p_k is the predicted probability for class k, given by the sigmoid function:

$$p_k = \sigma(s_k) = \frac{1}{1 + \exp(-s_k)}$$

where s_k is the logit (score before activation) for class k. The model outputs one probability per class, and classification is typically done by thresholding p_k, e.g., using 0.5.

Gradient. The gradient of the loss with respect to the score vector $\mathbf{s}$ is:

$$\frac{\partial \ell_{\text{bce}}}{\partial \mathbf{s}} = \frac{1}{K}(\boldsymbol{p} - \mathbf{y})$$

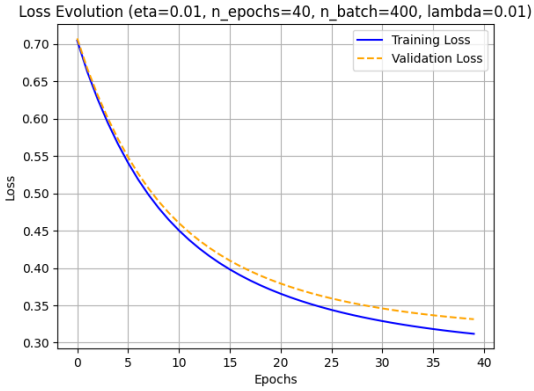
where $\boldsymbol{p} = \sigma(\mathbf{s})$ is the vector of predicted probabilities. This result implies that we can reuse most of the existing backpropagation code by simply replacing the softmax with the sigmoid activation and using this new gradient expression.

We train the network on the original dataset to compare the test accuracy with the softmax regression model. The results are shown in Table 2.

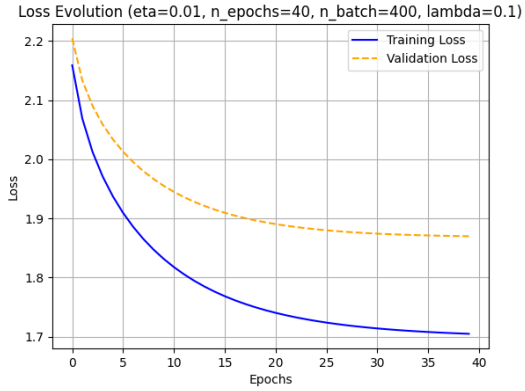
Table 2: Final accuracy (%) for the multiple binary cross-entropy loss and sigmoid experiment vs softmax regression.

Experiment	$\boldsymbol{\eta}$	$\boldsymbol{\lambda}$	**Train Acc.**	**Val/Test Acc.**
Softmax Regression	0.01	0.1	45.46%	38.42% / 38.88%
Multiple BCE + Sigmoid	0.01	0.01	45.64%	38.17% / 38.10%

As shown in Table 2, the softmax regression model slightly outperforms the multiple binary cross-entropy model in terms of both validation and test accuracy. This is expected, as softmax is specifically designed for multi-class classification tasks, where each input belongs to exactly one class. The softmax activation introduces competition among classes by normalizing the output probabilities to sum to one, enforcing mutual exclusivity. In contrast, the sigmoid activation used in combination with the binary cross-entropy loss treats each class as an independent binary classification task. This can lead to ambiguous predictions where multiple classes are assigned high probabilities, which is suboptimal in a single-label setting like CIFAR-10. Therefore, while multiple BCE can be effective for multi-label problems, softmax remains the preferred choice for multi-class classification due to its inductive bias and superior performance.



(a) $\eta = 0.01$, $\lambda = 0.1$ with BCE loss

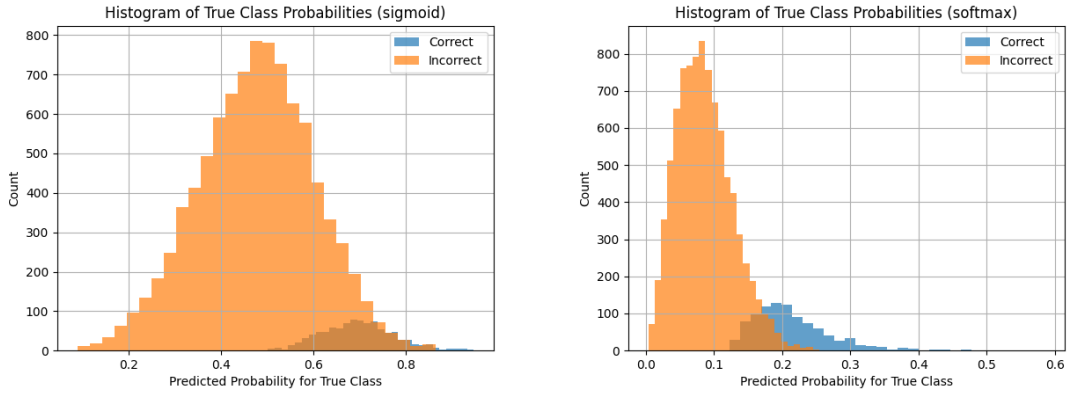


(b) $\eta = 0.01$, $\lambda = 0.1$ with Softmax

In Figure 6, we can see the histograms of the correctness of the predictions using BCE loss and softmax. The softmax classifier shows a higher confidence on the true class when it is correct, but also makes overconfident mistakes. The sigmoid model tends to be more conservative, assigning mid-range probabilities even when correct. This might be a sign of better regularization but could also lead to underconfident predictions.

5 Conclusion

In this report, we analyzed the performance of a single-layer softmax regression model for CIFAR-10 classification. Through experiments, we observed the impact of learning rate and regularization strength on training stability and generalization. Moderate regularization and a small learning rate were found to yield the best results. Additionally, data augmentation and hyperparameter tuning further improved performance.



(a) Histogram of the correctness of the predictions using BCE loss.

(b) Histogram of the correctness of the predictions using Softmax.

Figure 6: Histograms of the correctness of the predictions using BCE loss and Softmax.

We also compared the softmax regression model with an alternative approach using multiple binary cross-entropy losses. While the softmax model performed slightly better for single-label classification, the BCE approach demonstrated potential for multi-label tasks.

Overall, this study highlights the importance of hyperparameter selection and the trade-offs between different loss functions in neural network training. These insights provide a foundation for exploring more complex architectures in future work.