1. **Background and Method Introduction:**

A linear classifier is a machine learning model used to separate classes of data by a straight line, or a hyperplane in higher dimensional spaces. In the context of image classification, a linear classifier can be applied to distinguish between different categories of images based on features extracted from the images.

The process of image classification with a linear classifier generally involves the following steps:

**Feature extraction:** Significant features are extracted from the images, such as edges, colors, textures, etc.

**Feature representation:** The extracted features are represented in a format suitable for the classifier.

**Classifier training:** The parameters of the linear classifier are tuned using a labeled dataset.

**Classification:** Once trained, the classifier can be used to predict the class of new images based on the extracted features.

Linear classifiers are widely used in image classification tasks due to their simplicity and computational efficiency. Although they are relatively simple models, they can be effective in many situations and serve as an important foundation in the field of machine learning and computer vision.

1. **Dataset and Task Description:**

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 different classes, with each class having 6,000 images. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

For classification tasks on the CIFAR-10 dataset, the goal is to classify each image into one of the 10 categories using machine learning algorithms. In linear classification, such as logistic regression or linear SVM, the model learns a linear decision boundary to separate the different classes based on the features extracted from the images.

By training a linear classifier on the CIFAR-10 dataset, we aim to predict the correct class label for each image. The accuracy of the model is then evaluated by comparing the predicted labels to the actual labels in the test set. This accuracy metric helps assess how well the model performs on this specific classification task.

1. **Algorithm Used:**

We used the model is simple linear classification using neural networks to classify the cifar-10 images

**Model Architecture:**

Two models are defined: one without regularization and one with L2 regularization.

Both models consist of a single Dense layer with 10 units (corresponding to the 10 classes in CIFAR-10) and a softmax activation function.

The Flatten layer is used to flatten the input images (32x32x3) into a vector.

**Regularization:**

L2 regularization is applied to the weights of the Dense layer in the model\_with\_reg using kernel\_regularizer=l2(0.0001). This helps prevent overfitting by adding a penalty term to the loss function based on the L2 norm of the weights.

**Compilation:**

Both models are compiled using the Adam optimizer, categorical cross-entropy loss function, and accuracy as the metric to monitor during training.

**Training:**

The models are trained using the fit method on the training data (x\_train and y\_train) for 200 epochs with a batch size of 200 and a validation split of 20%.

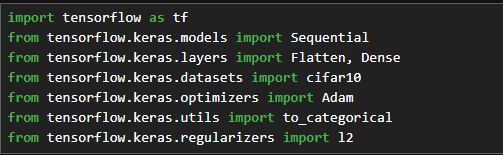
**Evaluation:**

After training, the models are evaluated on the test set (x\_test and y\_test) to calculate the test loss and test accuracy for both models.

**Working with Code Snippets:**

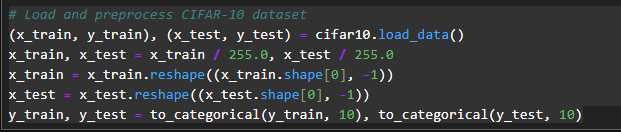
**Importing Necessary Libraries:**

Importing the necessary libraries that will be needing to run the model, to import the dataset directly form keras , optimizers like Adam and l2 , to preprocess the data , to convert the data into 1D etc.

****

**Data Preparation:**

The first step involved loading and preparing the dataset suitable for training and testing the model. This usually includes normalizing the data, splitting it into training and testing sets and possibly augmenting the data to improve generalization.



**Model Definition:**

Two models were defined for comparison purposes. The first model was built without regularization techniques to serve as a baseline (without regularization). The second model included regularization methods to observe their impact on model performance. So we will check results on both.

* **Without regularization:**

A screen shot of a computer program

Description automatically generated

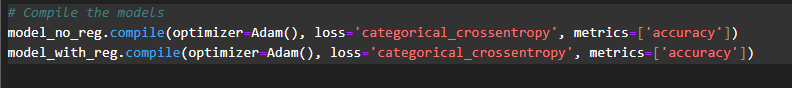
* **With regularization:**

A screen shot of a computer

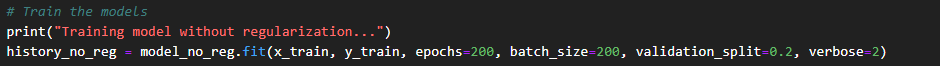
Description automatically generated

**Training:**

Both models were trained on a same dataset. Training parameters such as batch size, number of epochs and optimization algorithms were kept consistent across both models to ensure a fair comparison.



* **Without regularization**



* **With regularization**



**Evaluation:** After training, both models were evaluated on a test set to compare their performance. The key metrics for evaluation were loss and accuracy.

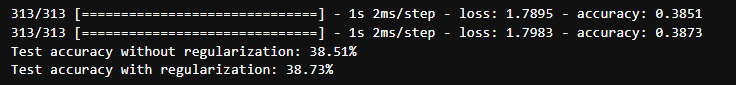
**For Without Regularization:**

****

**For Regularization:**

****

**For both:**

****

1. **Results:**

Based on the information provided, here are the results and discussions regarding the classification performance of the models:

**Non-Regularized Model:**

Test Accuracy: 38.51%

The non-regularized model achieved a test accuracy of 38.51%. This indicates that the model without any regularization techniques was able to correctly classify around 38.51% of the test data.

**Regularized Model:**

Test Accuracy: 38.73%

The regularized model, which included regularization techniques, achieved a slightly higher test accuracy of 38.73%. Despite the regularization, the improvement in accuracy was marginal compared to the non-regularized model.

**Reasons for Limited Improvement After Regularization:**

Over-regularization: Setting regularization parameters too high can lead to underfitting, where the model is overly constrained and unable to capture the underlying patterns in the data.

Inappropriate Regularization Technique: The choice of regularization technique (e.g., L1, L2, dropout) may not have been optimal for this specific dataset and model architecture.

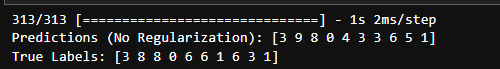
Model Complexity: The baseline model's complexity may not have been high enough to require regularization. Simpler models can sometimes perform well without regularization, especially on smaller or less complex datasets.

**Conclude:**

In conclusion, while regularization is essential for preventing overfitting in many cases, in this scenario, the limited improvement in test accuracy after applying regularization suggests that the models may not have been significantly affected by overfitting.

**Predicting for both on each class:**

**Predictions (No Regularization**): These are the categories/classes predicted by the model that was trained without any regularization techniques applied. Regularization is a method used to prevent overfitting by adding a penalty on the size of the weights. Overfitting occurs when the model performs well on the training data but poorly on unseen data (like the test set).



**Predictions (With Regularization):** These are the predictions made by the model that was trained with regularization applied. The idea is to see if regularization helps improve the model's performance on unseen data by making it generalize better.

A screen shot of a computer

Description automatically generated

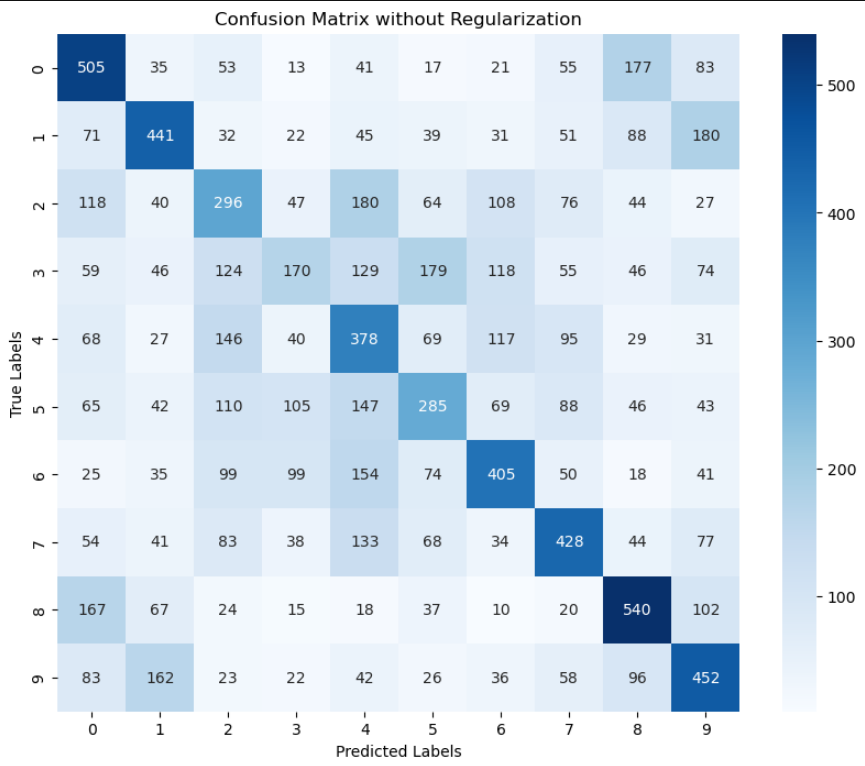
Comparing the predictions with and without regularization to the true labels gives insight into the model's performance and the effect of regularization:

* For some data points, predictions are the same for both models and correct (the first prediction for both models is 3, and the true label is 3).
* For other points, both models make incorrect predictions, but the errors differ (the second prediction without regularization is 9, and with regularization is 9, while the true label is 8).
* There are also cases where regularization seems to help the model make a correct prediction where the non-regularized model does not (the sixth prediction changes from incorrect without regularization (3) to correct with regularization (6), aligning with the true label).

1. **Visualization**

Visualizations can provide insights into the training process and the model's performance. Common techniques include:

**Confusion Matrix:** A confusion matrix for the test set can reveal specific areas where the model performs well or poorly, indicating potential areas for improvement.



**Without Regularization:**

This matrix also shows high diagonal values and but there are differences in the distribution of misclassifications.

Class 7 and class 0 still have the highest correct predictions, with 540 and 505 respectively, and class 3 remains the lowest with 170 correct predictions.

Compared to the regularized version, this matrix seems to have higher misclassifications for some classes, indicating that the model may be overfitting.

A graph with blue squares

Description automatically generated

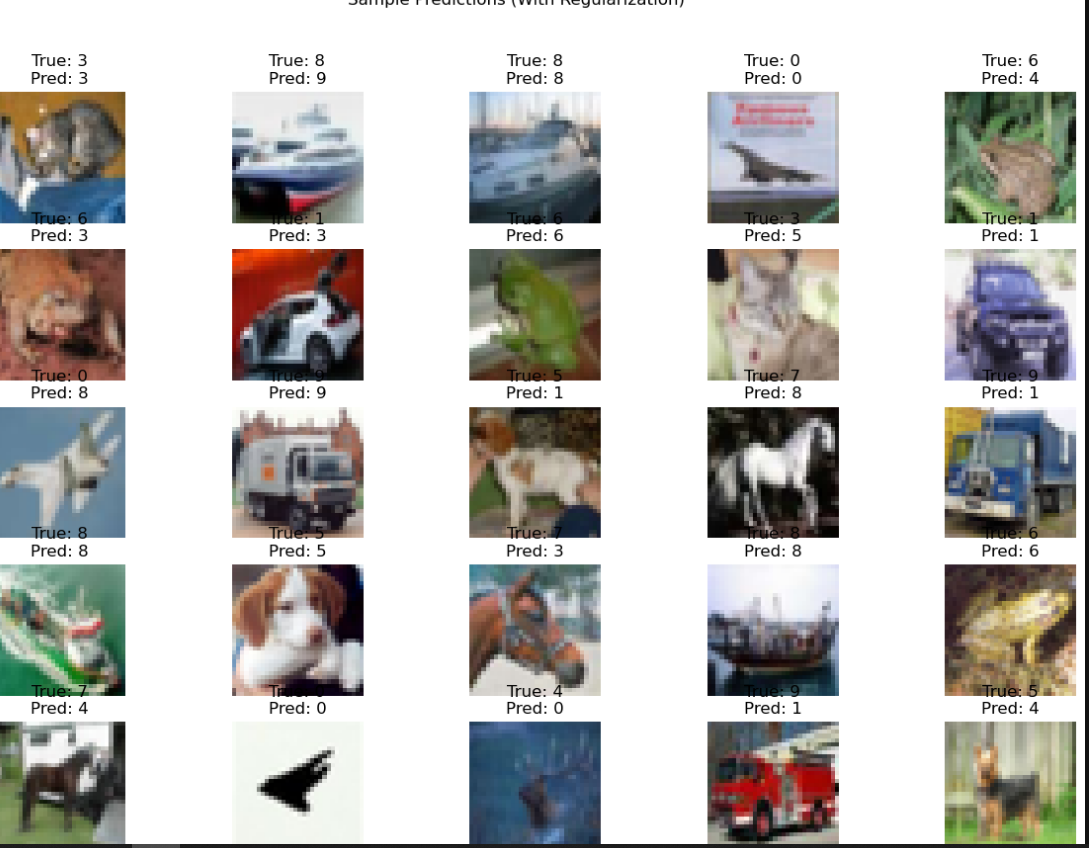
**With Regularization:**

The diagonal values, which represent correct classifications, are reasonably high across all classes, indicating a balanced performance across different categories.

Class 7 and class 1 have the highest correct predictions with values of 539 and 506 respectively, while class 3 has the lowest with 298.

Misclassifications are spread out among the other classes, but no single class is predominantly confused with another.

**Sample predictions for the model with regularization**



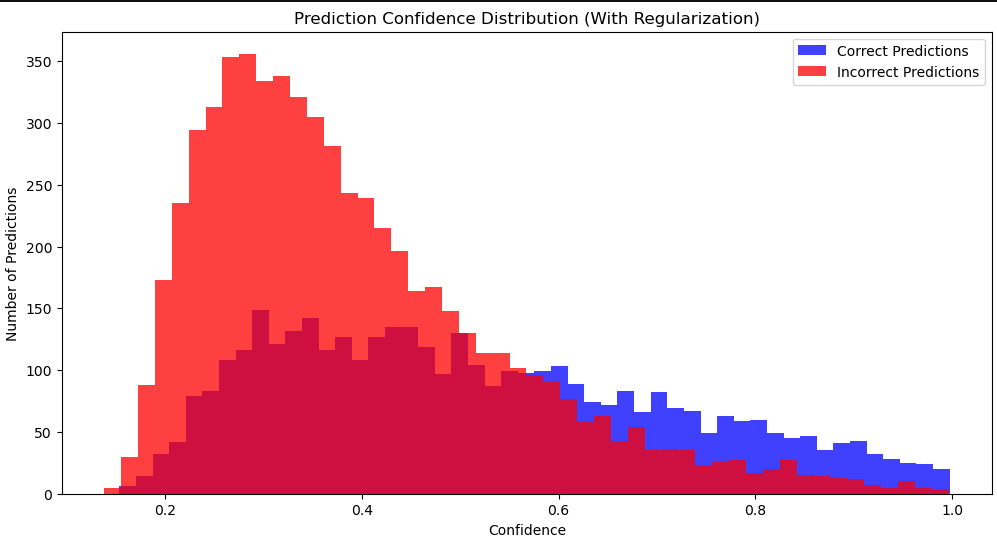
**Correct Predictions:**

Instances where the predicted class matches the true class, such as the first image in the top row (True: 3 and Pred: 3) and the first image in the bottom row (True: 8 and Pred: 8).

**Incorrect Predictions:**

Instances where the predicted class does not match the true class, such as the second image in the second row (True: 0 and Pred: 8) and the last image in the bottom row (True: 5 and Pred: 4).

**Prediction Confidence Distribution (With Regularization)**



* A significant number of correct predictions (blue) are made with high confidence, as there is a large blue bar on the right side of the histogram near the confidence level of 1.
* Incorrect predictions (red) are spread across various confidence levels but are particularly concentrated in the middle range of confidence levels.
* There is a notable amount of overlap between correct and incorrect predictions in the mid-confidence range (around 0.4 to 0.6). This overlap suggests that the model has a degree of uncertainty in its predictions in this range.
* The shape of the distribution suggests that the model is more often confident in its predictions when it is correct, but there is also a substantial number of cases where it is incorrectly confident.

1. **Method of Improvements:**

CNNs are particularly effective for image recognition tasks due to their ability to capture spatial hierarchies in images. By using convolutional and pooling layers, CNNs can learn more complex patterns with fewer parameters compared to fully connected networks. Implementing a CNN architecture could significantly improve model accuracy by efficiently extracting features from the images.

**Other Steps to Increase Accuracy**

**Data Augmentation:** Increasing the diversity of training set through rotations, translations, flips, and other transformations can help model generalize better to unseen data.

**Hyperparameter Tuning:** Optimizing learning rates, batch sizes, and other hyperparameters can significantly impact model performance.

**Advanced Regularization Techniques:** Beyond L1/L2 regularization and dropout, techniques like batch normalization or layer normalization can also help improve model training and generalization.

**Ensemble Methods**: Combining the predictions from multiple models can reduce variance and improve accuracy.

**Transfer Learning:** Leveraging pre-trained models and fine-tuning them on the specific dataset can yield significant improvements, especially when available data is limited.

**Concluding Remarks**

In this experiment the introduction of regularization with a lambda value of 0.001 resulted in a slight improvement in test accuracy, from 38.51% to 38.73%, over 200 epochs and with a batch size of 200. This minimal difference suggests that the chosen regularization strength and model configuration were not significantly overfitting to the training data. To further explore the potential benefits of regularization, one might consider varying the regularization strength, experimenting with different types of regularization, or adjusting model complexity. Additionally, employing other strategies such as data augmentation, hyperparameter tuning, and advanced model architectures like CNNs could provide more substantial improvements in model accuracy and generalization.