Convolutional Neural Network (CNN) for Image Classification on CIFAR-10

Group 1 Project3

Zhaoyang Li | ZL3327
Yinpei Wang| yw4123
Yuqi Liu| yl5478
Kechen Lu| kl3458
Professor Ying Liu

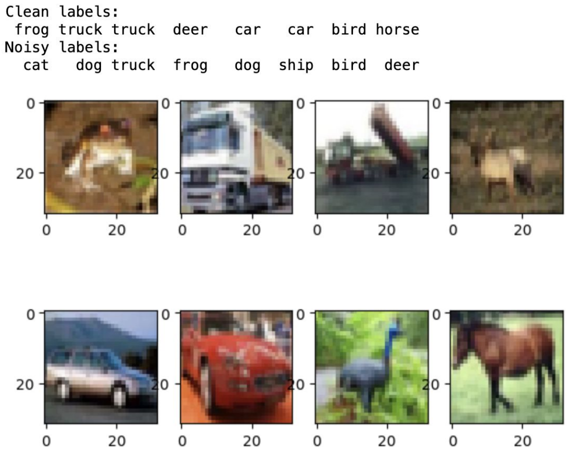
COLUMBIA

Overview

- **Introduction:**
  - Image classification is a fundamental problem in computer vision
  - Labeling large datasets can be time-consuming and costly
  - Presence of label noise is common in real-world scenarios
- **Project Objective:**
  - Develop robust and accurate predictive models for image classification
  - Handle the presence of label noise in the training data
  - Explore techniques to improve model performance and generalization
- **Challenges:**
  - Presence of label noise in the training data
  - Limited availability of clean labels
  - Need for models that can learn effectively from noisy data

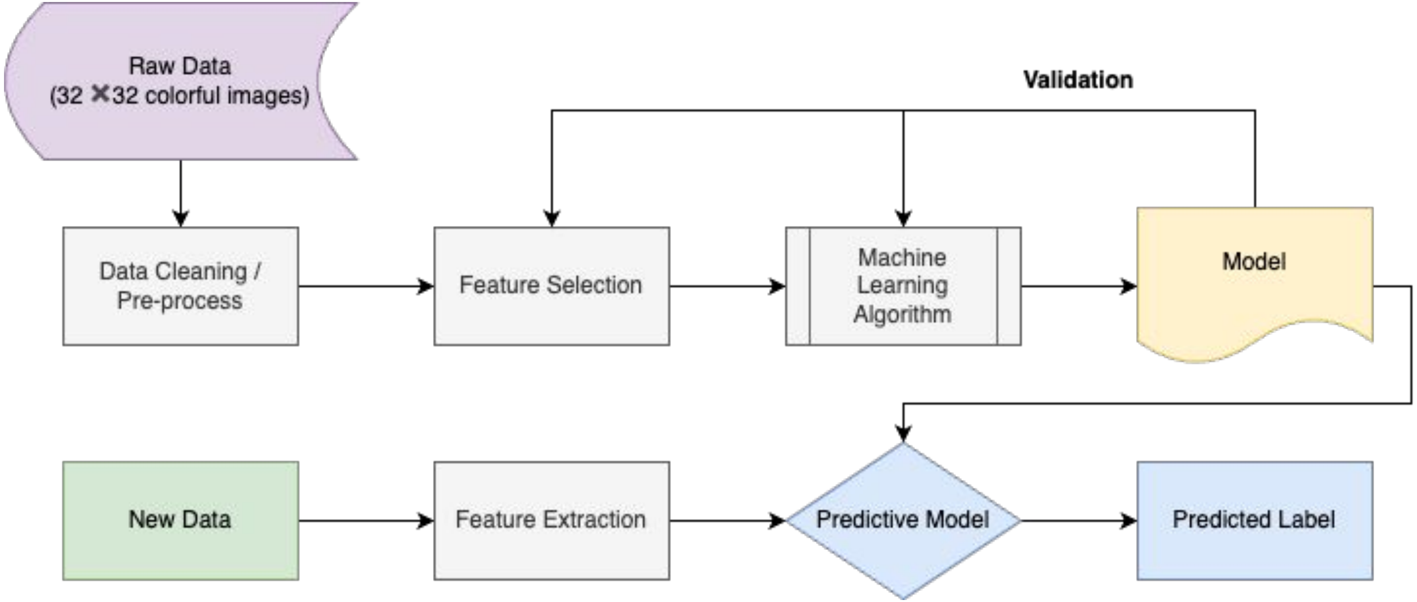
Data Set

- **Original CIFAR-10 dataset**:
  - 60,000 32x32 pixel color images categorized into 10 distinct classes. Each class represents a kind of object like animals and vehicles.
- **Noisy Version (what we used)**
  - A subset of the original dataset containing 50,000 images, with random noises added to labels
  - Noisy labels for all images in '../data/noisy_label.csv'
  - Clean labels for the first 10,000 images in '../data/clean_labels.csv'



Methods

- **A general procedure of machine learning**
  - We tried different algorithms and models in our analyses and compared their performance.



The Baseline Model - Logistic Regression

- **Model: Logistic Regression**
  - A simple and interpretable model for classification tasks
- **Data Used:**
  - Training set: 50,000 images from the noisy CIFAR-10 dataset
  - Noisy labels: Used for all 50,000 images during training
- **Features:**
  - RGB Color Histogram with 6 bins
  - Extracted from each image channel (R, G, B)
  - Feature vector: counts for each bin of the histogram of each channel
- **Training:**
  - The logistic regression model is trained using the noisy labels
  - Treats the noisy labels as if they were clean, without considering the label noise

Model I - CNN

- **Model: Convolutional Neural Network (CNN)**
  - A more sophisticated and powerful model compared to the baseline logistic regression
  - Utilizes convolutional layers to learn spatial hierarchies of features from the input images
- **Data Used:**
  - First 10,000 images from the noisy CIFAR-10 dataset
  - Used Clean labels for training Model I
  - Data split: 90% training (9,000 images), 10% validation (1,000 images)
- **Model Architecture:**
  - Input shape: (32, 32, 3) - RGB images of size 32x32 pixels
  - 3 convolutional layers with increasing depth (32, 64, 64 filters)
  - 2 max pooling layers following the convolutional layers
  - Flatten layer to convert the feature maps into a 1D feature vector
  - Dense layer with 64 units and ReLU activation
- **Training:**
  - Optimizer: Adam
  - Loss function: Sparse Categorical Cross-Entropy
  - Epochs: 200
- **Evaluation:**
  - The trained model is evaluated on a separate test set with clean labels
  - Uses the sklearn.metrics.classification_report to report performance metrics

Model II - CNN with Data Augmentation

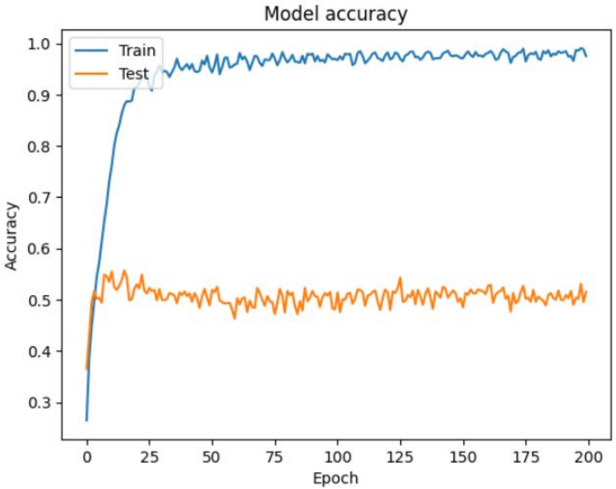
- **Model: Convolutional Neural Network (CNN) with Data Augmentation**
  - Utilizes the same CNN architecture as Model I
  - Incorporates data augmentation techniques to improve model robustness and generalization
- **Data Used:**
  - Training set: First 10,000 images from the noisy CIFAR-10 dataset
  - Clean labels: Used for training Model II
  - Data split: 70% training (7,000 images), 30% validation (3,000 images)
- **Data Augmentation:**
  - Applies various data augmentation techniques to the training images:
  - Augmentation is performed using the ImageDataGenerator from Keras
- **Model Architecture:**
  - Similar to Model I, but with some modifications:
    - Additional convolutional layers and filters (64, 128, 256)
    - Batch Normalization layers after each convolutional layer
- **Training:**
  - Optimizer: SGD (Stochastic Gradient Descent) with learning rate=0.01 and momentum=0.9
  - Loss function: Sparse Categorical Cross-Entropy
  - Epochs: 200

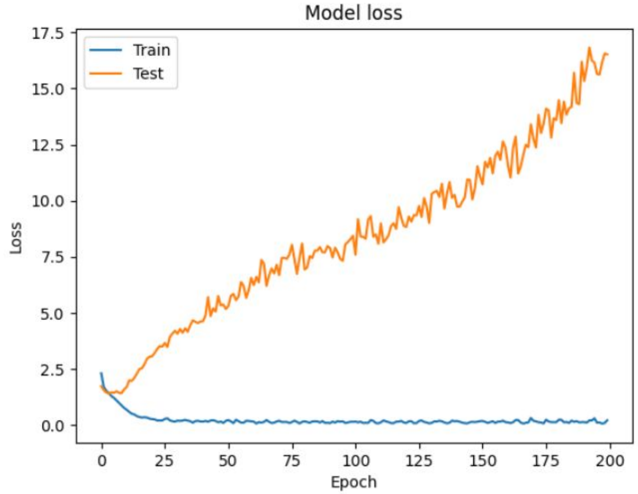
Evaluation - Baseline Model

The overall accuracy is 0.24, which is better than random guess (which should have a accuracy around 0.10).

	precision	recall	f1-score	support
0	0.33	0.46	0.38	1000
1	0.21	0.31	0.25	1000
2	0.20	0.04	0.07	1000
3	0.19	0.12	0.14	1000
4	0.24	0.48	0.32	1000
5	0.20	0.11	0.14	1000
6	0.24	0.34	0.28	1000
7	0.31	0.04	0.08	1000
8	0.27	0.43	0.33	1000
9	0.20	0.12	0.15	1000
accuracy			0.24	10000
macro avg	0.24	0.24	0.21	10000
weighted avg	0.24	0.24	0.21	10000

Evaluation - Model I



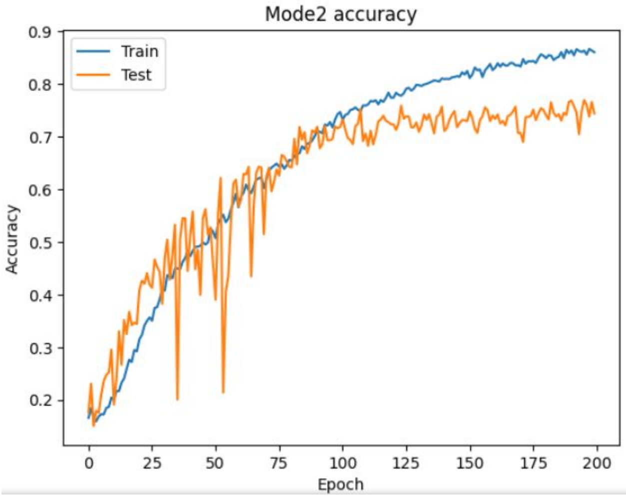


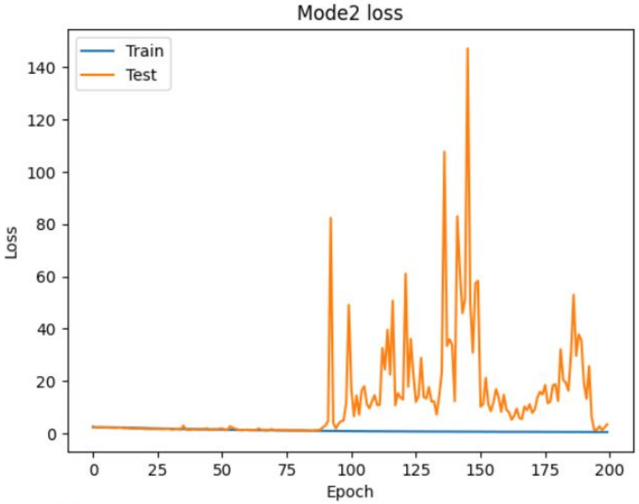
Evaluation

	precision	recall	f1-score	support
0	0.12	0.14	0.13	102
1	0.11	0.10	0.10	112
2	0.13	0.14	0.13	99
3	0.06	0.07	0.06	92
4	0.10	0.10	0.10	99
5	0.12	0.13	0.12	85
6	0.09	0.08	0.09	107
7	0.08	0.07	0.07	102
8	0.07	0.07	0.07	99
9	0.10	0.09	0.09	103
accuracy			0.10	1000
macro avg	0.10	0.10	0.10	1000
weighted avg	0.10	0.10	0.10	1000

```
32/32 - 0s - loss: 16.5300 - accuracy: 0.5150
Test Loss: 16.529953002929688, Test Accuracy: 0.5149999856948853
```

Evaluation - Model II





Evaluation

	precision	recall	f1-score	support
0	0.09	0.09	0.09	102
1	0.12	0.12	0.12	112
2	0.12	0.12	0.12	99
3	0.11	0.12	0.12	92
4	0.10	0.10	0.10	99
5	0.08	0.08	0.08	85
6	0.13	0.11	0.12	107
7	0.07	0.07	0.07	102
8	0.11	0.11	0.11	99
9	0.06	0.07	0.07	103
accuracy			0.10	1000
macro avg	0.10	0.10	0.10	1000
weighted avg	0.10	0.10	0.10	1000

```
94/94 - 0s - loss: 3.3523 - accuracy: 0.7443
Test Loss: 3.3523166179656982, Test Accuracy: 0.7443333268165588
```


COLUMBIA

Summary

- **Results**
  - Baseline model accuracy: **0.24**
    - Limited performance due to simple model and noisy labels
  - Model I accuracy: **0.515**
    - Improved performance compared to the baseline
    - Utilization of a more sophisticated CNN architecture
  - Model II accuracy: **0.7443**
    - Further improvement over Model I
    - Incorporation of data augmentation and regularization techniques
- **Conclusions**
  - Demonstrated the effectiveness of using CNNs for image classification
  - Highlighted the impact of label noise on model performance
  - Showed the benefits of data augmentation and architectural enhancements
  - Identified the need for explicit handling of label noise in future work

Discussion

- For this project, we can improve the performance by the following strategies:
  1. Consider a better choice of model architectures, hyperparameters, or training scheme for the predictive model;
  2. Use both clean_noisy_trainset and noisy_trainset for model training via **weakly supervised learning** methods. One possible solution is to train a "label-correction" model using the former, correct the labels in the latter, and train the final predictive model using the corrected dataset;
  3. Apply techniques such as k-fold cross validation to avoid overfitting.


Thank you for listening
Q&A