MEF UNIVERSITY
FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COMP 462

Term Project Report

Berat Kaya
Fatma Zehra Bayir
Yağmur Parmaksız
Kerim Ercan

Advisor: Yassine Drias

2025

1. Problem Definition and Motivation

The multiplication of visual information across fields from autonomous vehicles to medical imaging, security, and e-commerce has made image classification a central machine learning and computer vision problem. As the world is increasingly seen through images by machines, object recognition and classification become critical abilities.While working on the project, we used Python, Google Colab, Google Docs and Google Slides tools.

This project solves the image object recognition task with CIFAR-10 dataset, which is a commonly used benchmarking dataset of 60,000 32x32 color images of 10 classes such as airplanes, cats, cars, and ships. Despite its relatively small image resolution, CIFAR-10 is a realistic and challenging task due to object orientation, lighting, background clutter, and similarity in classes.

Our primary goal is to construct, train, and evaluate machine learning models capable of classifying CIFAR-10 images effectively. In this project, we deployed and contrast two models:

1. **Multi-Layer Perceptron (MLP)** : a straightforward feedforward neural network intended to discover non-linear relationships.
2. **Convolutional Neural Network (CNN)** : a specialized neural network architecture designed to automatically and adaptively learn spatial hierarchies of features, particularly effective for image data but also applicable to structured financial time-series for pattern recognition.

By studying these models, we would like to know not only how well each model performs in terms of accuracy, but also how design choices—such as network depth, regularization, and training method—affect actual-world classification results.

The inspiration for this project is both in practical and educational goals. Practically, object recognition is a core competency in AI systems applied in industry. Educationally, this project is an end-to-end experience in applying course material: data preprocessing, feature engineering, model selection, hyperparameter tuning, and multi-metric evaluation (accuracy, precision, recall, F1-score, and confusion matrix).

Further, comparing a number of models helps in building a more accurate picture of model trade-offs:

- How well can a simple linear model do on image data?
- Does complexity in models always lead to better results?
- What are diminishing returns of deeper networks in small-resolution images?

By systematic deployment and benchmarking, this project develops technical skill as well as our critical eye to evaluate machine learning models in complex situations.

2. Dataset Description

The CIFAR-10 dataset, a benchmark dataset with a significant position in computer vision and machine learning, is used in this study. CIFAR-10, created by the Canadian Institute for Advanced Research, is one of the most used datasets for testing image classification systems at the moment. Its popularity stems from its ability to strike a fair mix between simplicity and complexity. While its tiny picture size facilitates quick testing, real-world unpredictability poses a significant obstacle to model learning.

60,000 32x32 pixel color pictures with three color channels (red, green, and blue) make up CIFAR-10. Ten frequent item classes: aircraft, car, bird, cat, deer, dog, frog, horse, ship, and truck are included in the data collection. These classifications have semantic significance, are mutually exclusive, and present a challenging categorization job. For a completely balanced class distribution, the data set is split into a test set of 10,000 photos and a training set of 50,000 images, with precisely 6,000 images per class.

Each image in CIFAR-10 is a three-dimensional array representing its height, width, and color channels. For the purposes of many machine learning algorithms, particularly classical models like logistic regression these images are typically flattened into one-dimensional feature vectors of length 3,072. The corresponding class labels are encoded as integers from 0 to 9. Unlike many synthetic datasets, CIFAR-10 reflects real-world challenges such as occlusion, background clutter, pose variation, and inter-class similarity. For instance, it is hard to distinguish between trucks and cars or cats and dogs using comparable visual features.

CIFAR-10 has played a significant role in training and testing a wide range of machine learning models. CIFAR-10 is a default benchmark not only for testing new architectures, but also for researching preprocessing methods, feature extraction methods, regularization methods, and training optimizations. Also, its small size makes it perfect for academic projects, exploratory research, and comparing classical versus deep learning performance.

For this project, CIFAR-10 provides a good basis for comparing three machine learning models of different complexity: logistic regression, a baseline linear classifier; a vanilla multi-layer perceptron (MLP); and a wider, deeper extended MLP and convolutional neural network(CNN). The sophistication of the dataset, despite being low resolution, allows us to study how model complexity affects performance. Specifically, it enables us to observe how model depth and non-linearity contribute to the ability of the classifier to generalize from training instances and accurately classify novel instances.

Briefly, CIFAR-10 is offering engaging opportunities for experimentation with and investigation of the strengths and weaknesses of different machine learning models for image classification.

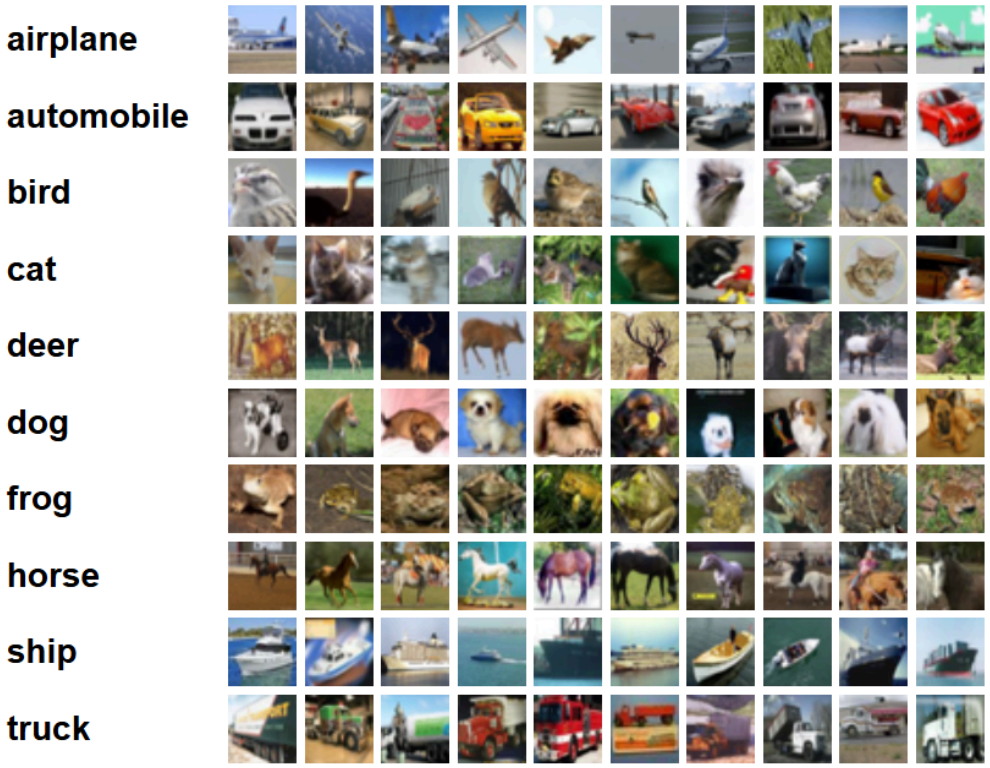

Figure 1. CIFAR-10 Dataset(CIFAR-10 and CIFAR-100 Datasets, 2025)

3. Preprocessing Steps

The CIFAR-10 dataset consists of 60,000 32x32 three-channel color images. The images are of a small size, and it's hard to classify the images since classes resemble one another, there are exposure variations, lighting variations, and incompatibility in the background. Therefore, specially adapted pre-processing steps have been applied for each model. Below, the common steps and the specific processes applied on the basis of the model are described in detail.

General Preprocessing Steps:

- **Image Normalization:** In all models, pixel values of images are scaled from 0-255 to 0-1. This process allows the model to learn more stably and reduces gradient deviations.
- **Data Separation:** The CIFAR-10 dataset is divided into 50,000 training and 10,000 test samples.
- **Labels:** In all models, classes are numbered from 0 to 9. One-hot encoding was used for neural network-based models, and direct numerical format was used in classical models.

3.1. MLP (Multi-Layer Perceptron) – Data Preprocessing Steps

Two distinct models with an MLP architecture were created as part of the project and evaluated using the CIFAR-10 dataset. The TensorFlow/Keras package was used to train both models, and the identical data preparation steps were used for both models. However, their architectural complexity varies. The effect of these differences on model performance has been observed.
During the training of our Multi-Layer Perceptron(MLP), we applied the following steps for data preprocessing and model regularization:

- **Flattening:** Each 32x32x3 image in CIFAR-10 has been converted to a 3,072-dimensional flat vector. Because fully connected layers receive fixed length vector input.
- **Normalization:** All pixel values are scaled from 0–255 to 0–1. This process accelerates and stabilizes the learning process.
- **Label Encoding:** Labels are converted to one-hot encoding format. Thus, the output layer with softmax activation is trained correctly.
- **Data Size:** Images were kept at the original size (32x32x3), and no resizing or cropping was done.

3.2. CNN (Convolutional Neural Network) Data Preprocessing Steps

The project involved creating and training a convolutional neural network (CNN) model using the CIFAR-10 dataset. Because CNNs can learn spatial hierarchies from image input, unlike MLP-based models, they are particularly well-suited for visual recognition uses. TensorFlow/Keras was used to create and train the CNN for this project. Because it was capable of dealing directly with image tensors, it used a different preprocessing method.
During the training of our Convolutional Neural Network (CNN), we applied the following steps for data preprocessing and model regularization:

- **Tensor Formatting:** Each image was kept in its original 32x32x3 format without flattening. CNNs can process three-dimensional input tensors, which preserve the spatial structure of the data and enable effective feature extraction through convolution operations.
- **Normalization:** Pixel values of the images were scaled from 0–255 to a 0–1 range using simple normalization. This is a standard step to help improve model convergence and numerical stability during training.
- **Label Encoding:** Class labels were one-hot encoded to align with the categorical cross-entropy loss function and the softmax activation function used in the output layer.
- **Data Splitting:** The dataset was divided so that 80% was used for training and 20% for validation. This split lets us monitor overfitting and adjust hyperparameters based on a reliable validation set.

- **Data Augmentation:**To increase the variety of the training set and improve generalization, we randomly applied these transformations:
  Rotation of images: Width and height shifts, Horizontal flips

Multiple layers of convolution and pooling, followed by flattening and dense layers, form the CNN model architecture. Local patterns like edges, forms, and textures were recovered from the image with convolutional layers, while computational load and spatial dimensions were decreased by max pooling operations. After the output was flattened, it was run through all of the layers to get a 10-class softmax prediction.

4. Feature Engineering

In the models used in this project, unprocessed pixel values within images were directly used as features without manual feature extraction or conventional image processing. With CIFAR-10 dataset consisting of low-resolution (32×32 pixels), uniformly distributed RGB images from 10 classes, using raw pixel inputs as a viable and efficient alternative, especially when deep learning architectures are used.

4.1 MLP Feature Processing

In the case of the Multi-Layer Perceptron (MLP) model, each image of size 32 × 32 × 3 (width × height × color channels) was flattened into a one-dimensional vector of length 3.072 (i.e. 32×32×3). This is due to the fact that fully connected (dense) layers take a fixed-size input. The model then tried to learn abstract representations from the unfolded pixel values through multiple dense layers.
Although such a representation does not respect spatial structure, the MLP model learned the basic class discriminative patterns and achieved a test accuracy of 58.37% with precision and recall of 58.71% and 58.37%, respectively.

4.2 CNN Feature Processing

In contrast, the Convolutional Neural Network (CNN) model preserved the same three-dimensional image structure of the input images (32×32×3). This allowed the model to exploit spatial locality by using convolutional filters that are hierarchically capable of detecting patterns such as edges, textures, and shapes.

Because of this structure-sensitive processing, the CNN achieved significantly better performance with test accuracy of 85.00% and very high average precision (85.42%) and F1 score (84.76%). This huge accuracy difference of over 26 percentage points in favor of the CNN model over the MLP model evidently demonstrates the utility of spatial feature learning in image classification.

Both designs avoided hand-tuned features or traditional preprocessing techniques. Instead, data-driven learning was used across the board, in line with the spirit of modern deep learning. This allowed the models to automatically learn useful patterns from unprocessed image data. By leveraging the computational advantages of its design,

CNN vastly surpassed at capturing spatial relationships, maintaining its validity in the presence of challenging image recognition tasks such as those presented by CIFAR-10.

5. Models Used and Rationale

5.1 MLP (Multi-Layer Perceptron)

Multilayer Perceptrons (MLPs), one of the most widely used forms of neural networks used during deep learning, are utilized in our project. MLPs consist of more than a single layer of neurons: an input layer, multiple hidden layers, and an output layer. They use non-linear activation functions and can learn complicated patterns and thus are very suitable for activities such as classification, regression, and pattern recognition. The three-dimensional structure of the image data is converted into a one-dimensional vector for these models and provided as input to the model. In low-resolution datasets such as CIFAR-10, such structures can often provide reasonable results in learning the underlying patterns of images.

The Multilayer Perceptron (MLP) structure was used as a feedforward neural network with fully connected layers. This is because MLPs do not inherently work on spatial data, and therefore the 32x32x3 CIFAR-10 images were first flattened into 3072-dimensional vectors. They were then passed through a series of dense layers. The model consisted of three hidden layers with 1024, 512 and 256 units, respectively, each followed by ReLU activation. Dropout layers were introduced to prevent overfitting by randomly turning off half of the neurons during training. In order to strengthen the training process and promote convergence, Batch Normalization was applied after each dense layer. In addition, an image data augmentation strategy of random rotation, horizontal flip, and translations was adopted. This helped improve model robustness and dataset diversity. Training employed Adam optimizer with initial learning rate set to 0.001. Learning rate scheduler (ReduceLROnPlateau) dynamically reduced the learning rate when improvement stagnated. The model was trained for more than 30 epochs on mini-batches of 128 images. While training, training and validation accuracy improved linearly. Because of the pairing of augmentation and normalization approaches, the test accuracy of the MLP model was 58.37%, and the precision on average was 58.71%, recall was 58.37%, and F1-score was 57.71%. These are reasonable but modest results, given the limitations of MLPs over spatial data like images.

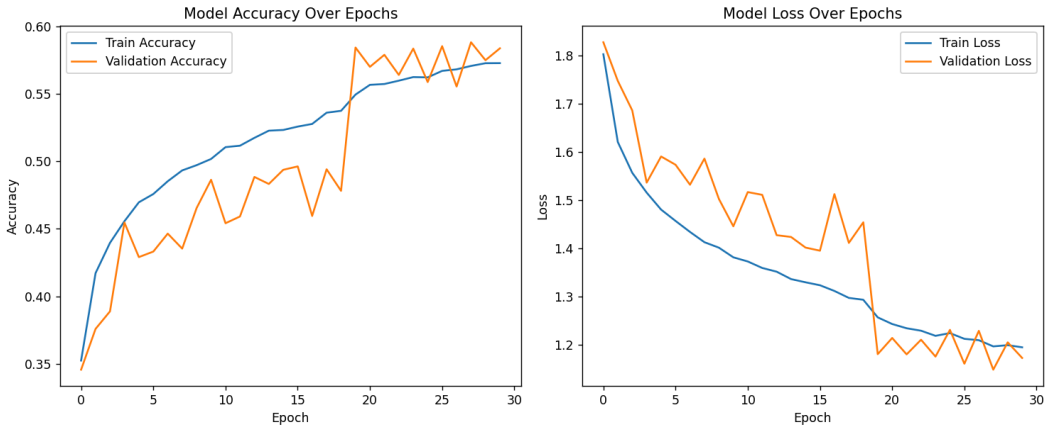

Figure 2.Learning Curves

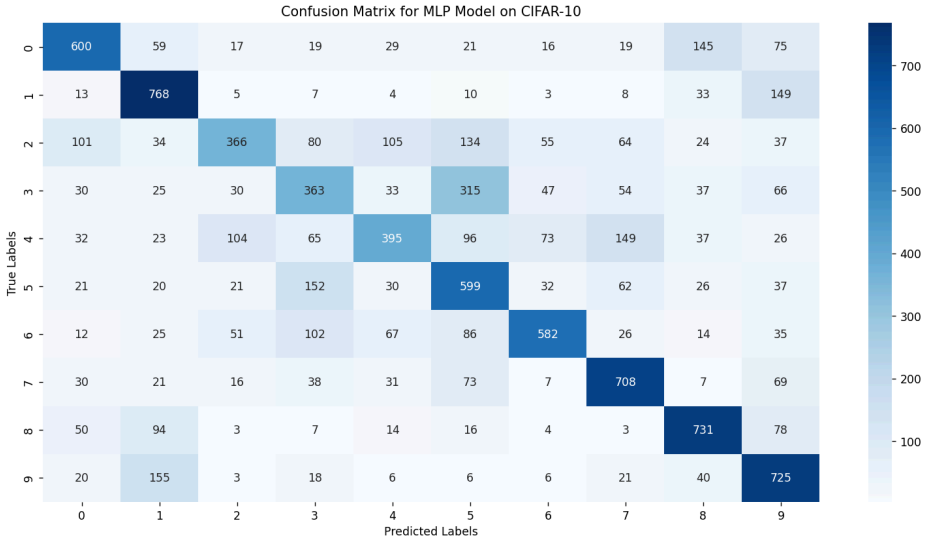

Figure 3.Confusion Matrix for MLP on CIFAR-10

During model training, both validation accuracy and training accuracy increased on a regular basis. Risk of overfitting was, nonetheless, noted to set in early. Regularization techniques such as Dropout and Batch Normalization were used to reduce the situation. Through the implementation of ReduceLROnPlateau, the model was stabilized by automatically switching to a lower learning rate depending on the val_loss metrics during learning.

The main flaw of the MLP architecture is that it cannot detect local patterns in the image. Since such patterns are highly determining in visual data, MLP models are disadvantaged in this aspect. This was explicitly observed especially in the comparisons with CNN. Qualitative analysis of misclassified images from the MLP model shows recurring failure cases on:

- Occluded or blurred images

- Fine-grained classes like cat vs. dog
- Off-centered or background-dominated objects

These challenges again point to the shortcomings of the MLP in capturing context-aware features.

The classification report shows the model performed outstandingly well on classes such as:

Class 1 (automobile)	F1-score: 0.69
Class 8 (ship)	F1-score: 0.70
Class 9 (truck)	F1-score: 0.63
Class 7 (horse)	F1-score: 0.67

But performed very poorly on:

Class 2 (bird)	F1-score: 0.45
Class 3 (cat)	F1-score: 0.39
Class 4 (deer)	F1-score: 0.46

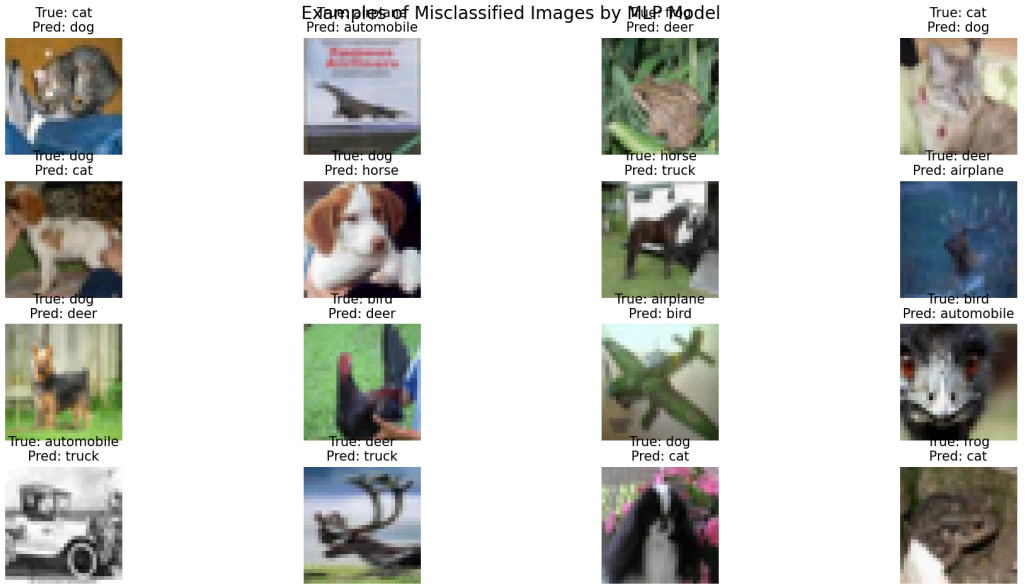

Figure 4.Sample Misclassified Images by MLP

The model was enhanced by a schedule of the learning rate (ReduceLROnPlateau), particularly after Epoch 19, when the learning rate was decreased to half. This made the performance remarkably improved in Epochs 20–30.

Validation accuracy peaked at ~58.8%, and corresponding loss went down to ~1.17.

This class-by-class performance is reflected in the confusion matrix, which indicates frequent confusion between visually similar classes (e.g., cats confused with dogs, or birds with airplanes), especially those with fine-grained or subtle visual differences.

Training dynamics also show monotonically growing training and validation accuracy (to ~57%), while validation loss went down overall but with some volatility. These suggest the model did not catastrophically overfit, yet still did not generalize perfectly to the test set.

Input: (32x32x3) → Flatten (3072) →

Dense(1024) → BatchNorm → ReLU → Dropout(0.3) →

Dense(512) → BatchNorm → ReLU → Dropout(0.3) →

Dense(256) → BatchNorm → ReLU →

Dense(10) → Softmax

Although the MLP model is not directly applicable to visual data, it has been doing quite well with certain regularization techniques (dropout, batch normalization) and training procedures (learning rate manipulation, data augmentation). The model is a valuable learning material for deep learning newbies.

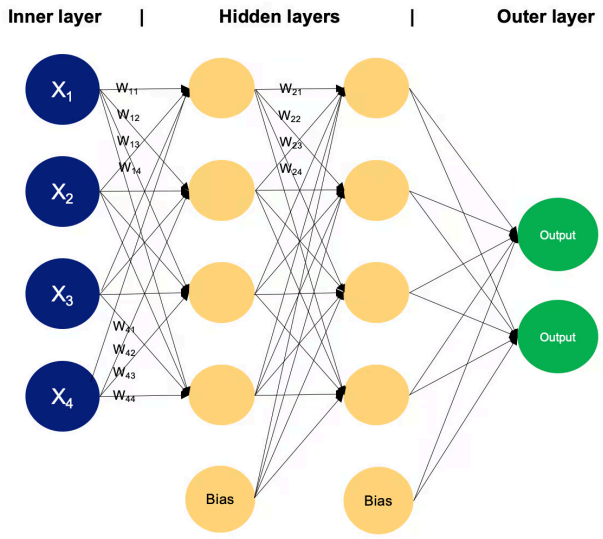

Figure 5. MLP Model Architecture

5.2 CNN (Convolutional Neural Network)

In this project, a CNN model was used to classify images from the CIFAR-10 dataset. The model has three main convolutional blocks. Each block is composed of two Conv2D layers with ReLU activation followed by a Batch Normalization layer and a Dropout layer to avoid overfitting. For each block, filter count is increased: 32, 64, and 128.Then it uses a GlobalAveragePooling2D layer instead of flattening to reduce parameters. Then, it applies a Dense layer with 256 units and ReLU activation, and finally, a softmax layer for classification.

Data augmentation methods such as rotation, shifting, and horizontal flip were used to improve training. Further, ReduceLROnPlateau and EarlyStopping callbacks were used to stop training at the right time and adjust the learning rate. This CNN model was more accurate compared to the MLP models because convolutional layers are specially designed for image feature comprehension. A Convolutional Neural Network (CNN) was utilized in this project for image classification in the CIFAR-10 data set. CNNs are special deep machine learning models specifically designed to be utilized with image data. They can learn important image features such as edges, shapes, and colors.

The CNN model in this project has three main blocks. Each block has two convolutional layers with ReLU activation, followed by batch normalization and dropout. These layers help the model to learn better and avoid overfitting. After these blocks, a GlobalAveragePooling2D layer was used to reduce the number of parameters. Then, a fully connected (dense) layer with 256 neurons and a softmax output layer was added for classification. To improve performance, data augmentation was used. The training images were randomly rotated, shifted, and flipped. This helped the model generalize better. Also, EarlyStopping and ReduceLROnPlateau callbacks were used during training to stop early if the model stopped improving and to reduce the learning rate when needed.At the end of training, the CNN model reached about 85% accuracy on the test data. This is much higher than the results from the MLP models. These results show that CNNs are very effective for image classification tasks because they can learn complex patterns in images.

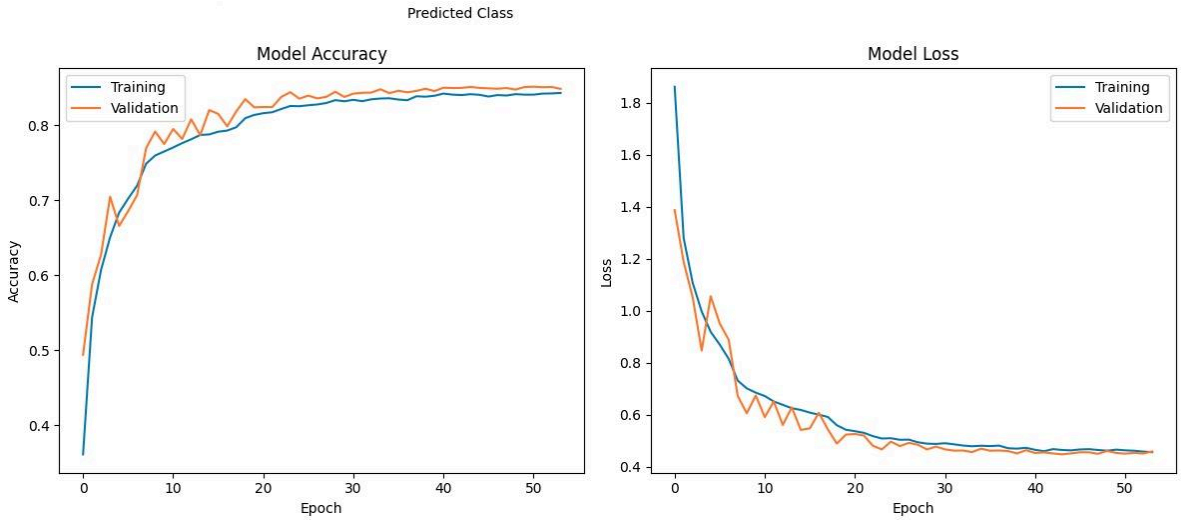

Figure 6.Learning Curves

Figure 7.Confusion Matrix for CNN on CIFAR-10

```
===== Classification Report =====
              precision    recall  f1-score   support

    airplane       0.88      0.86      0.87      1000
  automobile       0.90      0.96      0.93      1000
        bird       0.84      0.78      0.81      1000
         cat       0.82      0.62      0.71      1000
        deer       0.84      0.84      0.84      1000
         dog       0.87      0.72      0.79      1000
        frog       0.71      0.97      0.82      1000
       horse       0.88      0.90      0.89      1000
        ship       0.93      0.92      0.93      1000
       truck       0.88      0.93      0.90      1000

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

Figure 8.Clasification Report for CNN

Input: (32×32×3) →

Conv2D(32) → BatchNorm → ReLU →

Conv2D(32) → BatchNorm → ReLU → MaxPooling → Dropout(0.2) →

Conv2D(64) → BatchNorm → ReLU →

Conv2D(64) → BatchNorm → ReLU → MaxPooling → Dropout(0.3) →

Conv2D(128) → BatchNorm → ReLU →

Conv2D(128) → BatchNorm → ReLU → MaxPooling → Dropout(0.4) →

GlobalAveragePooling2D() →

Dense(256) → BatchNorm → ReLU → Dropout(0.5) →

Dense(10) → Softmax

The CNN model leverages spatial hierarchies through three convolutional blocks, each increasing in depth (32 → 64 → 128 filters) and followed by progressive dropout regularization. Global average pooling flattens the final feature maps efficiently without adding too many parameters. Dense classification layers with batch normalization and dropout help stabilize and regularize training. The final softmax layer outputs class probabilities for the 10 CIFAR-10 categories.

6. Evaluation Results and Comparison

Within this project, we have compared the performance of two neural network architectures on the task of image classification in CIFAR-10: a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN). We have compared their performance on more than one measure: accuracy, precision, recall, and F1-score. The results are reported on in the table below:

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)
MLP	0.5837	0.5871	0.5837	0.5771
CNN	0.85	0.8542	0.8498	0.8476

The reason why the CNN model is 24% more accurate than MLP is because of its ability to efficiently learn spatial information (e.g. contours, edges) of images. While each pixel is independently viewed by MLP architecture, CNN learns local relationships through windows.In contrast, MLPs flatten the image input, essentially destroying spatial relations among pixels. As a result, the model must learn complex global patterns using fully connected layers alone, which is inefficient for high-dimensional image data.

Despite regularization techniques (dropout, batch normalization) and learning rate scheduling, the MLP model fails to generalize well to unseen data, particularly in fine-grained classifications.The MLP model worked better for classes where there were distinct global features (e.g., ships, cars), but did not discriminate well between classes where there were finer differences.The CNN model was more consistent across all classes, including those with higher intra-class variability. This is indicated by its more balanced class-wise precision and recall values.The comparative results underscore the imperative of making architectural decisions based on data modality. For image classification tasks, especially with difficult and cluttered scenes like CIFAR-10, CNNs offer a superior solution by being able to exploit spatial coherence in images. MLPs, while competitive otherwise, are limited by the fact that they are not able to model such spatial dependencies to good effect.

7. Challenges Faced and Insights Gained

During the development of the project, we encountered several challenges that presented learning opportunities, both technically and in a way related to teamwork and project management. One of these challenges was that MLP classifiers do not create meaningful relationships between pixel-level images, which is necessary to distinguish between visually similar classes (e.g. cats and dogs; cars and trucks). Therefore, it confirmed the need for a more suitable architecture such as CNN.Another challenge encountered during the project was the computational time cost associated with training deep models, especially CNN. While data augmentation was used to improve the generalization ability of the models, this similarly increased the training time. To overcome some of this training time, we used various methods such as early stopping and decreasing the learning rate to limit the training, which allowed us to avoid problems such as the model overfitting on the training dataset and spending too much time for training. It was also important to monitor the validation accuracy and loss during training to evaluate whether adjustments could be made to the model parameters or whether model training could be performed.The CIFAR-10 dataset itself, the small size of the images and the presence of background clutter, differences in object orientations and low-resolution details made it easier for the models to learn features consistently. Nevertheless, these challenges led us to use preprocessing and editing techniques to better effect, and we also gained a deeper understanding of the role of data quality on the model's ability to perform.Through this project, we gained a more comprehensive understanding of how convolutional architectures have an advantage over dense models in image classification, and the effects of model tuning, preprocessing and visualization on interpreting and improving model results.Overall, our project was a practical assignment to help us understand machine learning concepts in many ways, but it also had the opportunity to see more holistic learning, which includes analytical thinking, collaboration and the capacity to critique the behavior of models under operating constraints.

8. Team Contribution Statement

This project was carried out collaboratively by a team of four people, and the team members took an active part in different stages of the process. In the harmony of the project, different rates such as technical applications, analysis, reporting and presentation preparation were shared and a fair division of labor was ensured within the team. Below is a detailed description of one of its members' projects:

Kerim: The creation, coding, training and evaluation of the CNN model were completely carried out by Kerim. In addition, it is his responsibility to organize the model's visualization outputs, accuracy analysis and layer structure.

Berat: Berat worked on implementing data preprocessing steps, training MLP models and tuning hyperparameters. He also contributed to the completion of the presentation by taking an active role in the visual layout, graphic layout and technical content of the presentation file.

Fatma Zehra: Fatma Zehra Worked with Yağmur and Berat on issues such as installing the extended MLP model and implementing dropout and regularization methods. In addition, she took an active role in writing the project report, content monitoring and final control process. She also provided team coordination in planning the presentation.

Yağmur: Yağmur took an active role in creating the basic MLP model, monitoring the training process and evaluating the results. She also made significant contributions to the writing and editing of the project report, especially in ensuring the integrity of language and expression. She also took part in the content layout and text editing stages of the project presentation.

9. References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems* (arXiv:1603.04467). https://arxiv.org/abs/1603.04467

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Chollet, F. (2015). *Keras* (Version 2.x) [Computer software]. https://keras.io

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing, 7*(3–4), 197–387. https://doi.org/10.1561/2000000039

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern

Jaiswal, S. (2024, February 7). Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. Datacamp.com; DataCamp. https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

Recognition (CVPR) (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images* (Technical Report). University of Toronto. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems, 25*, 1097–1105. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539
Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)* (pp. 807–814).
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958.
Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)* (pp. 818–833). Springer. https://doi.org/10.1007/978-3-319-10590-1_53