HỌC VIỆN BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

MÔN LẬP TRÌNH PYTHON

PYTHON ASSIGNMENT REPORT

Giảng viên hướng dẫn: Kim Ngọc Bách

Họ và tên sinh viên : Phan Anh Minh Đức

Mã sinh viên : B23DCCE022

Lớp : D23CQCE04-B

Hà Nội – 2025

Abstract:

This report presents a comparative study on the efficacy of different neural network architectures for the task of image classification, utilizing the widely recognized CIFAR-10 dataset as a benchmark. The investigation focuses on three distinct models: a foundational Multi-Layer Perceptron (MLP), a custom-designed Convolutional Neural Network (CNN), and a pre-trained ResNet18 model, representing a deep residual learning architecture. The CIFAR-10 dataset, with its 10 classes of 32x32 color images, provides a suitable platform for evaluating these models. Key findings from the experiments indicate a substantial performance hierarchy. The MLP achieved a modest test accuracy of 48.55%. The custom CNN demonstrated significantly improved performance, attaining a test accuracy of 87.40%. The pre-trained ResNet18 model, adapted for CIFAR-10, yielded a test accuracy of 83.08% under the specified training conditions.1 These results underscore the critical role of architectural design, particularly the incorporation of spatial feature extraction mechanisms inherent in CNNs. The comparative analysis between the custom CNN and the pre-trained ResNet18 further highlights nuances in model selection, suggesting that factors such as dataset characteristics, model adaptation, and training duration are pivotal. The implications of these findings are discussed, offering insights for selecting appropriate neural network architectures for image classification tasks, especially when dealing with datasets characterized by small image dimensions and moderate scale.

1. Introduction

1.1. The Significance of Image Classification

Image classification stands as a cornerstone task within the domain of computer vision. Its fundamental objective is to assign a categorical label to an entire image based on its visual content.2 This capability is not merely an isolated academic exercise; rather, it forms the foundational basis upon which numerous other complex computer vision problems are built, including object detection, semantic segmentation, and image retrieval.3 The advancements in image classification directly translate to progress in a wide array of real-world applications. These span from enhancing autonomous vehicles' understanding of their surroundings and aiding in medical image analysis for diagnostic purposes, such as identifying diseases from radiological scans 5, to bolstering security systems through object recognition, improving industrial automation via quality control, and enabling sophisticated content-based image retrieval systems.3 The pervasive impact of image classification underscores its role as an enabling technology for a multitude of artificial intelligence (AI)-driven solutions, making research into its improvement a critical driver for the broader field.

1.2. The CIFAR-10 Dataset: A Benchmark for Image Recognition

The CIFAR-10 dataset, an acronym for Canadian Institute For Advanced Research, was curated by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and has become a standard benchmark in the machine learning community for evaluating image classification algorithms.6 It comprises 60,000 color images, each with a resolution of 32x32 pixels, distributed across 10 mutually exclusive classes: airplane, automobile (car), bird, cat, deer, dog, frog, horse, ship, and truck.1 The dataset is partitioned into a training set of 50,000 images and a test set of 10,000 images, with an equal distribution of 6,000 images per class (5,000 for training, 1,000 for testing).6

The selection of CIFAR-10 for this comparative study is motivated by several factors. Its widespread adoption facilitates direct comparison of results with a vast body of existing literature, providing context and a measure of relative performance.6 Furthermore, despite its relatively modest size compared to datasets like ImageNet, CIFAR-10 presents a non-trivial challenge that effectively tests the robustness and generalization capabilities of different model architectures. This dual characteristic of being accessible for experimentation, particularly for researchers with limited computational resources 9, while still posing significant classification difficulties, makes it an ideal testbed for evaluating and contrasting neural network models.

1.3. Challenges Inherent to CIFAR-10

The CIFAR-10 dataset, while a valuable benchmark, presents several inherent challenges that can complicate the image classification task:

- **Low Resolution:** The small image size of 32x32 pixels severely limits the amount of visual detail available for feature extraction.[9] This low resolution makes it difficult for models to discern fine-grained textures and subtle differences that might be crucial for distinguishing between classes, effectively creating a bottleneck for feature engineering, especially for models that do not inherently capture spatial hierarchies well.

- **Intra-class Variability and Inter-class Similarity:** The dataset exhibits considerable visual diversity within each class (e.g., various models of cars, different breeds of dogs) and, simultaneously, notable visual similarities between different classes (e.g., cats and dogs, or certain types of trucks and cars) [User Query Point 2]. This ambiguity increases the likelihood of misclassification, requiring models to learn highly discriminative features. The confusion matrices generated in this study (presented later) provide empirical evidence of these inter-class similarities.
- **Dataset Scale and Overfitting:** With 50,000 training images, CIFAR-10 is moderately sized. While sufficient for training reasonably complex models, it is not as extensive as larger-scale datasets. This makes models, particularly deeper architectures with many parameters, susceptible to overfitting if appropriate regularization techniques are not employed [User Query Point 2]. Issues such as potential label noise and duplicated examples within the dataset can further exacerbate this tendency.[9]

1.4. Overview of Evaluated Models

This report focuses on a comparative evaluation of three distinct neural network architectures:

- **Multi-Layer Perceptron (MLP):** A foundational, fully connected neural network. It serves as a baseline to underscore the challenges of image classification when spatial information is not explicitly leveraged.
- **Convolutional Neural Network (CNN):** A custom-designed CNN architecture incorporating convolutional, batch normalization, and pooling layers, specifically tailored to process image data and extract spatial features.
- **ResNet18:** A pre-trained 18-layer deep residual network. This model is included to assess the benefits of deeper architectures and the efficacy of transfer learning from a large-scale dataset (ImageNet) to the CIFAR-10 task.

The primary objective of this study is to systematically evaluate and compare the performance of these models on the CIFAR-10 dataset. This involves analyzing their architectural strengths and weaknesses, the impact of their design choices on classification accuracy, and their learning behaviors under consistent experimental conditions.

1.5. Report Structure

The remainder of this report is organized as follows: Section 2 details the data preprocessing techniques and the overall experimental setup. Section 3 provides a thorough description of the model architectures and the rationale behind their design choices. Section 4 discusses the training enhancements and regularization strategies employed. Section 5 presents the experimental results, including performance metrics, learning dynamics, and error analysis. Section 6 offers a comprehensive discussion of the findings, comparing model performances and analyzing their behaviors. Finally, Section 7 concludes the report with a summary of key findings, limitations of the study, and directions for future research.

2. Data and Experimental Methodology

2.1. Dataset Description and Splitting

The experiments were conducted using the CIFAR-10 dataset, which consists of 60,000 32x32 color images categorized into 10 classes: 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck'.1 For the purpose of training and validation, the original official training set of 50,000 images was divided. Specifically, 80% of these images (40,000) were allocated to the training subset, and the remaining 20% (10,000 images) formed the validation subset. The official test set, comprising 10,000 images, was reserved exclusively for the final evaluation of the trained models.1 To ensure the reproducibility of the data splits and subsequent experimental results, a global random seed of 42 was utilized for all random processes, including the dataset splitting procedure.1 The use of a consistent, shared validation set derived from the original training data for all models is a critical aspect of the methodology. This practice ensures a fair and unbiased comparison of model performance during the development phase, including hyperparameter tuning and model selection, by preventing any model from having an advantage due to variations in validation data and by avoiding data leakage from the test set.

2.2. Data Preprocessing and Augmentation

A series of preprocessing and augmentation steps were applied to the CIFAR-10 images to prepare them for model training and evaluation.

- **Normalization:** All images were first converted into PyTorch tensors. Subsequently, the pixel values of these tensors were normalized. This normalization was performed using the mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] derived from the ImageNet dataset.[1] Normalization is a crucial step as it ensures that all input features (pixel values across different channels) are on a similar scale. This helps to stabilize the training process, potentially accelerate convergence, and is particularly important when employing models pre-trained on ImageNet, such as the ResNet18 used in this study, as these models expect input data to be normalized in a consistent manner.[10]

- **Basic Augmentation (for MLP and Custom CNN):** For the training data fed to the MLP and the custom CNN models, a set of basic data augmentation techniques was applied. These included RandomHorizontalFlip(), which randomly flips images horizontally with a 50% probability, and RandomCrop(32, padding=4), which first pads the 32x32 image by 4 pixels on all sides and then takes a random 32x32 crop.[1] These augmentation methods artificially increase the diversity of the training set. By exposing the models to slightly altered versions of the original images, they help to reduce overfitting and enhance the models' ability to generalize to unseen data that may exhibit similar variations [User Query Point 2].

- **Advanced Augmentation (for ResNet18):** The training data for the ResNet18 model underwent a more aggressive suite of augmentation techniques. This included RandomCrop(32, padding=4), RandomHorizontalFlip(), RandomRotation(degrees=15) (randomly rotating images by up to 15 degrees), ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1) (randomly altering color properties), and RandomErasing(p=0.5, scale=(0.02, 0.2), ratio=(0.3, 3.3), value='random') (randomly erasing a rectangular region of the image).[1] Deeper models like ResNet18 possess a higher capacity to learn complex functions and are consequently more susceptible to overfitting, especially when trained on datasets of moderate size like CIFAR-10. The application of stronger augmentation techniques is a common and effective strategy to regularize these models, thereby improving their generalization performance.[10] RandomErasing, in particular, encourages the model to learn features from different parts of an object rather than relying excessively on a few salient regions.[10] The differing intensities of data augmentation—basic for the MLP and custom CNN, and more advanced for ResNet18—reflect a deliberate approach to tailor the regularization strategy to the complexity and learning capacity of each model. Simpler models might be overwhelmed or hindered by overly aggressive transformations, whereas more complex models benefit significantly from such techniques to prevent memorization of the training data.

- **Validation and Test Set Transformation:** For all models, the images in the validation and test sets were subjected to only two transformations: conversion to PyTorch tensors (ToTensor()) and normalization using the ImageNet statistics (Normalize()).[1] No random augmentation techniques were applied to these sets. This ensures that model evaluation is performed on consistent, unmodified data, providing a deterministic and comparable measure of performance.

2.3. Evaluation Metrics

Model performance was assessed using a combination of quantitative metrics:

- The primary metric for comparing the overall performance of the models was **test accuracy**. This metric represents the proportion of images in the unseen test set that were correctly classified by the model.[1]

- During the training process, **cross-entropy loss** was employed as the objective function to guide the optimization of model parameters. This loss function is standard for multi-class classification problems.[1]

- To enable a more granular analysis of performance at the class level, **precision, recall, and F1-score** were calculated for each of the 10 CIFAR-10 classes. These metrics were derived from the predictions on the test set.[1] Precision measures the accuracy of positive predictions for a class, recall (or sensitivity) measures the proportion of actual positives for a class that were correctly identified, and the F1-score provides their harmonic mean. These metrics are particularly insightful for understanding class-specific strengths and weaknesses of a model and can reveal biases or imbalances in performance, even if overall accuracy is high [User Query Point 2].

- **Confusion matrices** were generated for each model based on its predictions on the test set. These matrices provide a visual breakdown of classification performance, showing the counts of true positives, true negatives, false positives, and false negatives for each class.[1] They are invaluable for qualitatively identifying which classes are frequently confused with one another, thereby highlighting potential ambiguities in the dataset or specific areas where a model struggles [User Query Point 5].

2.4. Experimental Environment

The experiments were conducted in a computational environment configured as follows:

- **Hardware:** A CUDA-enabled Graphics Processing Unit (GPU) was utilized for model training and inference, as indicated by the device log: cuda.[1] The specific model of the GPU was not detailed in the provided logs but access to GPU acceleration was confirmed.

- **Software:** The primary software framework used was PyTorch, along with its companion library torchvision for dataset handling and pre-trained models. Additional Python libraries employed for data manipulation, metrics calculation, and visualization included numpy, scikit-learn, seaborn, and matplotlib.[1]

- **Reproducibility:** To foster reproducibility of the experimental results, a global random seed of 42 was set at the beginning of the experimental script. This seed governed the behavior of Python's built-in random module, numpy's random number generators, and PyTorch's random operations, ensuring consistency in processes such as data splitting, model weight initialization, and dropout patterns.[1]

3. Model Architectures and Design Justifications

3.1. Multi-Layer Perceptron (MLP)

- Architecture:

The MLP model implemented in this study is a standard feedforward neural network. Input images from the CIFAR-10 dataset, with dimensions 3x32x32 (3 color channels, 32x32 pixels), are first flattened into a one-dimensional vector of 3072 features (3×32×32=3072).1 This flattened vector serves as the input to the network.

The core of the MLP consists of three hidden layers. The first hidden layer contains 512 neurons, the second contains 256 neurons, and the third contains 128 neurons. Each of these hidden layers employs the Rectified Linear Unit (ReLU) as its activation function. To mitigate overfitting, a dropout layer with a rate of 0.3 is applied after the first hidden layer and again after the second hidden layer.1

The final layer is an output layer, which is a fully connected layer with 10 neurons. Each neuron corresponds to one of the 10 classes in the CIFAR-10 dataset. The output of this layer typically passes through a Softmax function to produce class probabilities; however, in this implementation, the CrossEntropyLoss function (used during training) internally combines the Softmax activation and the Negative Log-Likelihood loss, so an explicit Softmax layer is not required in the model definition for training.

- Design Justification:

The MLP architecture was included in this study primarily to serve as a baseline model. Its relatively simple structure, which does not inherently account for the spatial nature of image data, helps to establish a performance benchmark. This benchmark effectively highlights the complexities of image classification on CIFAR-10 when specialized spatial feature extraction mechanisms are absent.

The choice of a three-layer hidden architecture with a decreasing number of neurons (512 → 256 → 128) is a common heuristic. Such a design aims to gradually reduce the dimensionality of the representation while extracting increasingly abstract features. While the Universal Approximation Theorem posits that an MLP with even a single hidden layer of sufficient width can approximate any continuous function, practical deep learning often favors deeper, somewhat narrower networks for better generalization and feature hierarchy. The specified architecture represents a reasonable configuration for a non-spatial model on this task.

The most significant limitation of the MLP for image classification is its initial flattening step. This operation discards all spatial relationships between pixels, treating each pixel's value as an independent feature. For image understanding, local pixel neighborhoods, textures, and structural arrangements are paramount. By converting the 2D (or 3D, with channels) image structure into a 1D vector, this critical spatial context is lost, severely hampering the MLP's ability to

learn effective visual features [User Query Point 3]. The MLP's performance, therefore, acts as a direct measure of the value that spatially aware architectures, like CNNs, bring to image-based tasks. Its struggles emphasize why architectures specifically designed for grid-like data are indispensable for achieving competent image classification.

3.2. Custom Convolutional Neural Network (CNN)

- Architecture:

The custom CNN designed for this study comprises three main convolutional blocks followed by a fully connected classification head.1

  - **Convolutional Block 1:**
    - Layer 1: 2D Convolution (input channels: 3, output filters: 64, kernel size: 3x3, padding: 1), followed by Batch Normalization (64 features) and ReLU activation.
    - Layer 2: 2D Convolution (input filters: 64, output filters: 64, kernel size: 3x3, padding: 1), followed by Batch Normalization (64 features) and ReLU activation.
    - Pooling: Max Pooling (kernel size: 2x2, stride: 2). This reduces the feature map dimensions from 32x32 to 16x16 (output shape: 16x16x64).
  - **Convolutional Block 2:**
    - Layer 1: 2D Convolution (input filters: 64, output filters: 128, kernel size: 3x3, padding: 1), followed by Batch Normalization (128 features) and ReLU activation.
    - Layer 2: 2D Convolution (input filters: 128, output filters: 128, kernel size: 3x3, padding: 1), followed by Batch Normalization (128 features) and ReLU activation.
    - Pooling: Max Pooling (kernel size: 2x2, stride: 2). This reduces feature map dimensions from 16x16 to 8x8 (output shape: 8x8x128).
  - **Convolutional Block 3:**
    - Layer 1: 2D Convolution (input filters: 128, output filters: 256, kernel size: 3x3, padding: 1), followed by Batch Normalization (256 features) and ReLU activation.
    - Layer 2: 2D Convolution (input filters: 256, output filters: 256, kernel size: 3x3, padding: 1), followed by Batch Normalization (256 features) and ReLU activation.
    - Pooling: Max Pooling (kernel size: 2x2, stride: 2). This reduces feature map dimensions from 8x8 to 4x4 (output shape: 4x4x256).
  - **Flatten Layer:** The output from the third convolutional block (4x4x256) is flattened into a 1D vector of 4096 features (256×4×4=4096).
  - **Fully Connected Block:**
    - Layer 1: Linear layer (input features: 4096, output features: 512), followed by ReLU activation and Dropout (rate: 0.5).
    - Layer 2: Linear layer (input features: 512, output features: 10), providing the final logits for the 10 CIFAR-10 classes.

- Design Justification:

The architecture of the custom CNN is predicated on established principles for effective image feature learning:

  - **Hierarchical Feature Extraction:** Convolutional layers employ learnable filters that automatically scan the input image to detect patterns. Early layers typically learn to identify basic features such as edges, corners, and simple textures. Subsequent, deeper layers combine these simpler features to detect more complex patterns and eventually object parts, forming a hierarchy of representations [User Query Point 3]. This process of creating feature maps through filter operations is fundamental to

CNNs.[12] This hierarchical approach is a primary advantage over MLPs, which lack this inherent capability.

- **Parameter Sharing and Translation Invariance:** A key characteristic of convolutional layers is parameter sharing, where the same filter (set of weights) is applied across all spatial locations of its input. This drastically reduces the total number of learnable parameters compared to a fully connected layer attempting to capture similar local relationships, making the model more efficient and less prone to overfitting. It also endows the network with a degree of translation invariance, meaning the model can recognize a feature regardless of its exact position in the image.
- **Pooling Layers:** Max pooling layers are interspersed between convolutional blocks to progressively reduce the spatial dimensions of the feature maps. This reduction makes the learned representations more robust to minor translations and distortions in the input image and decreases the computational load for subsequent layers.
- **Batch Normalization:** Incorporated after each convolutional layer (and before the ReLU activation), batch normalization serves to stabilize the training process by normalizing the mean and variance of the inputs to each layer. This helps to mitigate the problem of internal covariate shift, allows for higher learning rates, accelerates convergence, and can also act as a form of regularization.[13]
- **Dropout:** A dropout layer with a rate of 0.5 is included in the fully connected part of the network to combat overfitting by randomly deactivating a fraction of neurons during training, thus preventing the network from becoming overly reliant on specific neurons or features [User Query Point 4]. The overall structure, featuring three convolutional blocks with an increasing number of filters (64 → 128 → 256), is a common design pattern in CNNs. This allows the network to learn a richer set of more complex features at deeper stages, while pooling layers manage the spatial resolution. The consistent use of 3x3 kernels is a standard practice, offering an effective balance between the receptive field size and computational cost. This architecture represents a deliberate balance between model capacity required for CIFAR-10 and the risk of overfitting, drawing inspiration from VGG-style networks but with a shallower depth suitable for the dataset size. The effectiveness of this CNN arises not from any single component in isolation, but from the synergistic interaction of its convolutional layers (for feature extraction), pooling layers (for dimensionality reduction and invariance), batch normalization (for training stability), and ReLU activation functions (for introducing non-linearity).

3.3. Pre-trained ResNet18 (Comparative Advanced Model)

- Architecture Overview:

The ResNet18 model is based on the deep residual network architecture introduced by He et al..13 The defining characteristic of ResNet architectures is the incorporation of residual connections, also known as skip connections. These connections allow the network to learn residual functions with reference to the layer inputs, effectively enabling gradients to bypass one or more layers. This architectural innovation is crucial for mitigating the vanishing gradient problem, which historically hindered the training of very deep neural networks, thereby allowing for the successful training of networks with significantly greater depth.12 ResNet18 is a specific variant with 18 layers containing these residual blocks.

- Transfer Learning Implementation:

For this study, the ResNet18 model was implemented using a transfer learning approach 1:

- The model was initialized with weights pre-trained on the large-scale ImageNet-1K dataset, as specified by weights=torchvision.models.ResNet18_Weights.IMAGENET1K_V1.
- The original final fully connected layer of the pre-trained ResNet18, which was designed to classify 1000 ImageNet classes, was removed. It was replaced with a new nn.Linear layer. This new layer takes the same number of input features as the original ResNet18's classifier but has 10 output units, corresponding to the 10 classes of the CIFAR-10 dataset. The weights of this newly added classification layer are typically initialized randomly and are trained from scratch on the CIFAR-10 data, while the weights of the pre-trained convolutional base (feature extractor) are fine-tuned during the training process on CIFAR-10.

- Design Justification:

The inclusion of ResNet18 is motivated by several factors:

  - **Addressing Vanishing Gradients for Deeper Networks:** As mentioned, residual connections are fundamental to ResNet's success, enabling the training of much deeper networks than was previously feasible by ensuring smoother gradient flow.[12]
  - **Leveraging Pre-trained Features:** Features learned by a deep network on a large and diverse dataset like ImageNet (e.g., detectors for edges, textures, shapes, and object parts) are often generalizable and can serve as a powerful starting point for new visual tasks, including CIFAR-10 classification. This transfer of knowledge can lead to faster convergence, better performance, and reduced data requirements for the target task, especially when the target dataset is relatively small.[15]
  - The ResNet18 architecture itself is a well-established and highly effective model for various image classification tasks, with many variants achieving state-of-the-art or near state-of-the-art results on benchmarks like CIFAR-10.[8] While transfer learning offers significant advantages, its application to datasets with substantially different characteristics from the pre-training dataset (e.g., ImageNet's larger, higher-resolution images versus CIFAR-10's small 32x32 images) requires careful consideration. The features learned on ImageNet might not be perfectly optimal for smaller images, and effective adaptation may necessitate careful fine-tuning strategies and sufficient training epochs on the target dataset. The provided ResNet18 implementation used ImageNet normalization statistics and more aggressive data augmentation [1], which are considered good practices for transfer learning. However, unlike some recommendations for adapting ResNets to CIFAR-10 (e.g., modifying the initial large-kernel convolutional layer or removing early max pooling to preserve spatial resolution [17]), the specific ResNet18 used in the experiments [1] does not appear to incorporate these initial layer modifications. This architectural detail, coupled with the number of training epochs, could influence its performance relative to a custom-designed network.

Several training parameters were common across models or chosen specifically to suit each architecture 1:

- **Optimizers:**
  - The MLP was trained using the Adam optimizer with a learning rate of 0.001. Adam is a widely adopted adaptive learning rate optimization algorithm known for its efficiency and good performance across a diverse range of deep learning tasks.
  - The custom CNN and the ResNet18 model were trained using the AdamW optimizer. For the CNN, the learning rate was 0.001 with a weight decay of $1\times10-5$. For ResNet18, the maximum learning rate was 0.01 with a weight decay of $1\times10-4$. AdamW is a variant of Adam that improves weight decay regularization by decoupling it from the adaptive learning rate mechanism, which can lead to better generalization and model performance [User Query Point 4].
- **Learning Rate Schedulers:**
  - For the MLP and the custom CNN, a ReduceLROnPlateau learning rate scheduler was employed. This scheduler monitors the validation loss and reduces the learning rate by a factor of 0.2 if no improvement is observed for a patience of 5 epochs.[1] This strategy allows for finer adjustments to the learning rate as training progresses and the model approaches a minimum in the loss landscape.
  - The ResNet18 model utilized a OneCycleLR scheduler with a maximum learning rate of 0.01 over 20 epochs, incorporating specific ramp-up and cool-down phases for the learning rate.[1] The OneCycleLR policy, proposed by Leslie Smith, often leads to faster convergence and can achieve a state of "super-convergence," frequently resulting in improved model performance, particularly with modern neural network architectures.[17] The training logs for ResNet18 [1] confirm the dynamic adjustment of the learning rate at each batch, characteristic of this scheduler.
- **Loss Function:** The nn.CrossEntropyLoss function was used for all three models. This is the standard loss function for multi-class classification tasks as it combines a Softmax activation with the negative log-likelihood loss, making it suitable for training models to output class probabilities.[1]

- **Batch Size:** A batch size of 128 was used for training all models.[1] This value is a common choice, offering a practical balance between the accuracy of the gradient estimate (which tends to improve with larger batch sizes) and computational resource constraints (memory and processing time).
- **Number of Epochs:** The MLP and the custom CNN were trained for 30 epochs, while the ResNet18 model was trained for 20 epochs.[1] The number of epochs was likely determined based on preliminary experiments or common practices for these models on CIFAR-10, aiming for convergence without incurring excessive training time. The difference in the number of training epochs, particularly the fewer epochs for ResNet18, is an important factor to consider when comparing the final performances of the models. The progression in the choice of optimizers (from Adam to AdamW) and learning rate schedulers (from ReduceLROnPlateau to OneCycleLR) as model complexity increases from MLP to CNN and then to ResNet18 reflects a common practice of employing more sophisticated optimization techniques for more advanced and potentially harder-to-train architectures.

4. Training Enhancements and Regularization Strategies

Effective training of deep neural networks, especially on datasets of moderate size like CIFAR-10, necessitates the use of various enhancement and regularization techniques to ensure convergence to good solutions and to prevent overfitting.

4.1. Learning Rate Scheduling

The dynamic adjustment of the learning rate during training is a crucial technique for efficient optimization.

- The ReduceLROnPlateau scheduler, used for the MLP and custom CNN models [1], adaptively modifies the learning rate. It monitors the validation loss and reduces the learning rate when this metric ceases to improve for a defined number of epochs (patience). This allows the model to make large steps in the loss landscape initially and then take smaller, more refined steps as it approaches an optimal solution, aiding in convergence to better local minima [User Query Point 4].
- The OneCycleLR scheduler, employed for the ResNet18 model [1], implements a cyclical learning rate policy. It gradually increases the learning rate from a small initial value to a pre-defined maximum value over a portion of the training epochs, and then gradually decreases it to a very small value over the remaining epochs. This approach can lead to faster convergence and often results in better generalization performance, a phenomenon sometimes referred to as super-convergence. The cyclical nature allows the optimizer to traverse saddle points more effectively and explore a wider area of the loss landscape.[17] The training logs for ResNet18 clearly show the learning rate changing with each batch, consistent with the operation of OneCycleLR.[1]

4.2. Advanced Optimizers

The choice of optimizer significantly influences training dynamics and final model performance.

- The AdamW optimizer was used for training the custom CNN and ResNet18 models.[1] AdamW is an improvement over the standard Adam optimizer, particularly in how it handles L2 regularization (weight decay). In Adam, weight decay is often conflated with the adaptive learning rate updates, diminishing its effectiveness as true L2 regularization. AdamW decouples weight decay from the gradient-based updates, applying it directly to the weights. This can lead to improved model generalization and more stable training, especially for complex models [User Query Point 4].

4.3. Regularization and Overfitting Mitigation

A multi-faceted approach to regularization was employed to prevent the models from overfitting to the training data and to enhance their ability to generalize to unseen test data.

- **Dropout:** This technique was applied in the fully connected layers of the MLP (with a rate of 0.3 after the first two hidden layers [1]) and the custom CNN (with a rate of 0.5 after the first dense layer [1]). During training, dropout randomly sets a fraction of neuron activations to zero. This prevents neurons from co-adapting too much and forces the network to learn more robust and redundant features, thereby reducing overfitting.
- **Batch Normalization:** Utilized extensively in the custom CNN (after each convolutional layer, before ReLU activation [1]) and is an integral part of the ResNet18 architecture.[13] Batch normalization standardizes the inputs

to each layer by adjusting and scaling the activations. This helps to stabilize training by reducing internal covariate shift, allows for the use of higher learning rates, accelerates convergence, and also provides a slight regularization effect.

- **Weight Decay:** L2 regularization, commonly known as weight decay, was applied through the AdamW optimizer for the custom CNN (with a value of $1\times10-5$) and the ResNet18 model (with a value of $1\times10-4$).[1] Weight decay adds a penalty term to the loss function proportional to the square of the magnitude of the model weights, discouraging large weights and thus promoting simpler models that are less likely to overfit.
- **Data Augmentation:** As detailed in Section 2.2, various data augmentation techniques were applied to the training images. This is a powerful form of regularization as it artificially expands the effective size of the training dataset, exposing the model to a wider range of variations and improving its invariance to transformations like flips, crops, rotations, and color shifts.
- **Implicit Early Stopping:** While the training loop in the provided code [1] does not implement an explicit early stopping criterion (e.g., halting training if validation loss does not improve for a set number of epochs, as suggested in User Query Point 4), a form of implicit early stopping was achieved. The training script saves the model weights (best_model_path) whenever the validation accuracy improves. For final testing, this best-performing model checkpoint is loaded.[1] This practice effectively selects the model state that demonstrated the best generalization on the validation set, preventing the use of a model that might have continued training into an overfitted state.

The combination of these diverse regularization methods—dropout, batch normalization, weight decay, extensive data augmentation, and implicit early stopping—constitutes a comprehensive strategy to combat overfitting. Such a layered approach is generally more effective than relying on a single technique, particularly for complex models trained on datasets like CIFAR-10 where the risk of overfitting is significant.

4.4. Hyperparameter Tuning Considerations

The hyperparameters used in the experiments (e.g., learning rates, batch sizes, dropout probabilities, weight decay coefficients) were fixed for each model configuration as specified in the provided experimental setup.1 It is important to acknowledge that these specific values represent a single point in a vast hyperparameter space. Optimizing these hyperparameters through systematic tuning procedures, such as grid search, random search, or more advanced techniques like Bayesian optimization 19, can often lead to further significant improvements in model performance [User Query Point 4]. Studies have shown that performance on datasets like CIFAR-10 can be sensitive to hyperparameter choices.20 While a comprehensive hyperparameter search was beyond the scope of the current experiments, it remains a critical aspect for maximizing model efficacy and represents a valuable direction for future work.

5. Experimental Results and Analysis

This section presents a detailed analysis of the experimental results obtained from training and evaluating the MLP, custom CNN, and ResNet18 models on the CIFAR-10 dataset.

5.1. Overall Performance Comparison

The final test accuracy achieved by each model provides a high-level summary of their classification capabilities on unseen data. Table 1 presents these results.

Table 1: Final Test Accuracy Comparison

Model	Test Accuracy
MLP	0.4855
Custom CNN (3 Conv)	0.8740

Optimized ResNet18	0.8308

This table immediately highlights a clear performance hierarchy. The MLP achieves a relatively low accuracy, significantly outperformed by both convolutional architectures. Notably, the custom CNN achieved the highest test accuracy in this set of experiments, surpassing the pre-trained ResNet18. This outcome warrants a more detailed investigation in the subsequent discussion section. This summary table is crucial as it provides a direct, quantitative basis for comparing the effectiveness of the different architectural approaches under the specific experimental conditions of this study.

5.2. Detailed Performance Metrics per Model

To gain a more nuanced understanding of each model's performance, detailed classification reports including per-class precision, recall, and F1-score were generated. These metrics allow for an examination of how well each model performs on individual classes, revealing potential biases or specific difficulties.

Table 2: Detailed Classification Report for MLP

Class	Precision	Recall	F1-Score	Support
plane	0.62	0.50	0.56	1000
car	0.56	0.63	0.60	1000
bird	0.49	0.22	0.30	1000
cat	0.31	0.36	0.33	1000
deer	0.45	0.33	0.38	1000
dog	0.42	0.38	0.40	1000
frog	0.51	0.55	0.53	1000
horse	0.51	0.58	0.54	1000
ship	0.60	0.67	0.64	1000
truck	0.42	0.63	0.50	1000
accuracy			**0.49**	**10000**
macro avg	0.49	0.49	0.48	10000
weighted avg	0.49	0.49	0.48	10000

(Data sourced from [1])

The MLP exhibits generally low precision, recall, and F1-scores across all classes, with particularly poor performance for 'bird' (F1=0.30) and 'cat' (F1=0.33). This indicates a weak discriminative ability overall.

Table 3: Detailed Classification Report for Custom CNN

Class	Precision	Recall	F1-Score	Support
plane	0.82	0.94	0.87	1000
car	0.92	0.97	0.94	1000
bird	0.77	0.87	0.82	1000
cat	0.77	0.75	0.76	1000
deer	0.86	0.89	0.87	1000
dog	0.88	0.76	0.82	1000
frog	0.89	0.93	0.91	1000
horse	0.94	0.88	0.91	1000
ship	0.98	0.85	0.91	1000
truck	0.94	0.91	0.92	1000
accuracy			**0.87**	**10000**
macro avg	0.88	0.87	0.87	10000
weighted avg	0.88	0.87	0.87	10000

(Data sourced from [1])

The custom CNN demonstrates substantially higher precision, recall, and F1-scores for most classes. Classes like 'car' (F1=0.94) and 'truck' (F1=0.92) show excellent performance. However, 'cat' (F1=0.76) and 'dog' (F1=0.82, with lower recall of 0.76) remain more challenging.

Table 4: Detailed Classification Report for ResNet18

Class	Precision	Recall	F1-Score	Support
plane	0.83	0.86	0.84	1000

car	0.91	0.90	0.91	1000
bird	0.81	0.78	0.79	1000
cat	0.69	0.65	0.67	1000
deer	0.81	0.83	0.82	1000
dog	0.76	0.72	0.74	1000
frog	0.88	0.88	0.88	1000
horse	0.88	0.87	0.87	1000
ship	0.89	0.92	0.91	1000
truck	0.84	0.91	0.87	1000
accuracy			**0.83**	**10000**
macro avg	0.83	0.83	0.83	10000
weighted avg	0.83	0.83	0.83	10000

(Data sourced from [1])

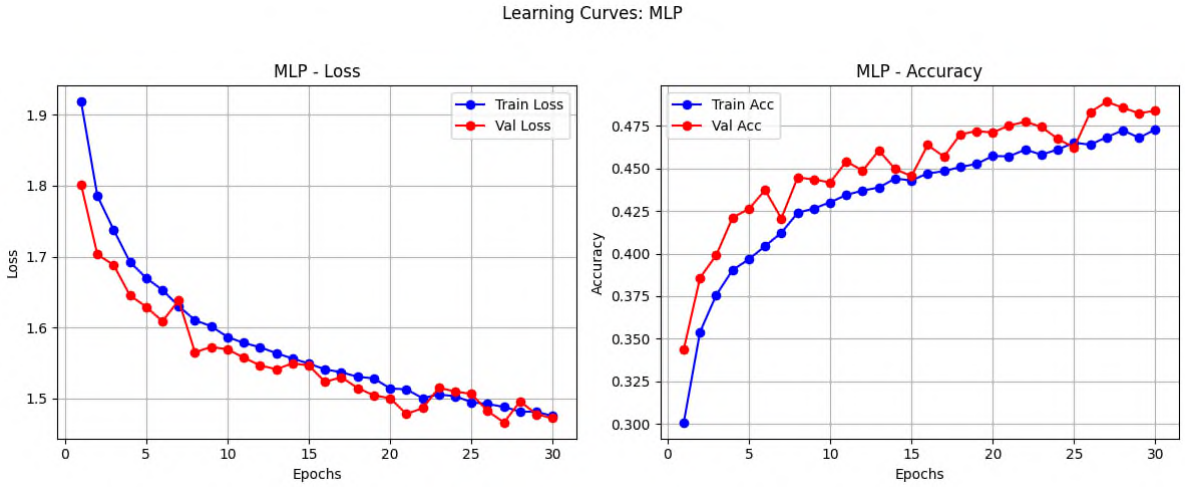
The ResNet18 model also shows competent performance across many classes, though its F1-scores for 'cat' (0.67) and 'dog' (0.74) are notably lower than those of the custom CNN and are the lowest among its own class scores. This suggests particular difficulty with these visually similar animal classes.

These detailed tables (Tables 2, 3, and 4) are invaluable because they move beyond aggregate accuracy to reveal the nuances of each model's behavior. They highlight which specific categories are well-handled and which pose persistent challenges, offering avenues for targeted model improvement or indicating inherent ambiguities within the dataset for certain classes.

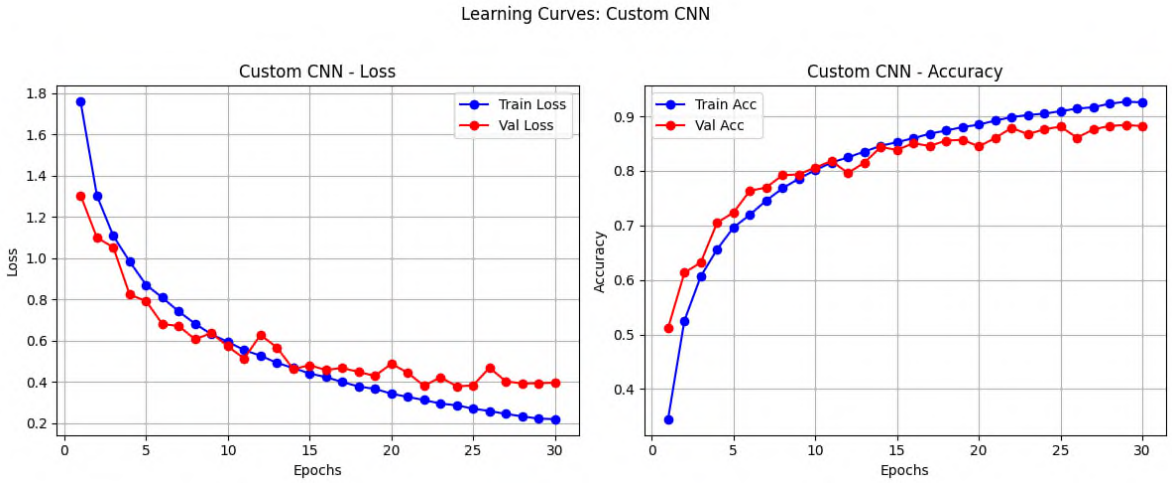
5.3. Learning Dynamics and Convergence

The learning curves, plotting training and validation loss and accuracy against epochs, provide insights into the training process and model generalization.

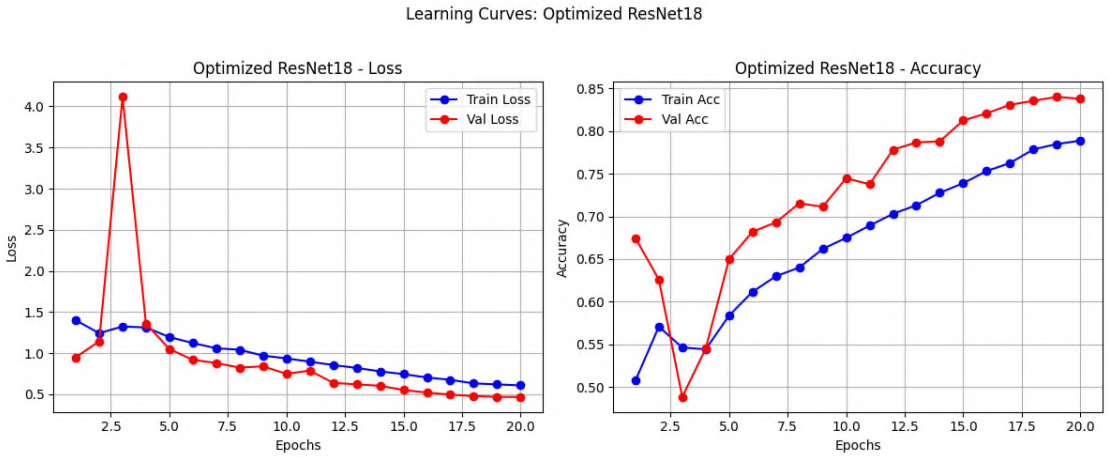
- **MLP:** The learning curves for the MLP [1] show a slow and limited improvement in both accuracy and loss. Training accuracy struggles to rise significantly above 47%, while validation accuracy plateaus around 48-49% after approximately 27 epochs. A persistent gap between training and validation metrics, coupled with relatively high final loss values, indicates poor generalization and an inability of the model to learn complex patterns from the data.



- **Custom CNN:** The custom CNN exhibits a much more favorable learning dynamic.[1] Both training and validation losses decrease rapidly in the initial epochs, while accuracies show a steep increase. Training accuracy surpasses 92%, and validation accuracy reaches approximately 88%. However, a noticeable trend is the widening gap between training and validation accuracy in the later epochs (after epoch 22 approximately). Concurrently, the validation loss begins to flatten and then slightly increase, while the training loss continues to decrease. This pattern is a classic indicator of the onset of overfitting, where the model starts to memorize the training data rather than learning generalizable features.



- **ResNet18:** The learning curves for ResNet18 [1] also show learning progress, with training accuracy reaching around 79% and validation accuracy peaking at 84% (best saved model) within 20 epochs. An interesting feature is a significant spike in validation loss around epoch 3 (from around 0.9 to 4.1) [1], after which it recovers and continues to decrease. This temporary instability could be attributed to the aggressive learning rates used by the OneCycleLR scheduler in its initial phase or a particularly challenging batch of validation data. A gap between training and validation accuracy is also present, suggesting some degree of overfitting or that the model could benefit from further fine-tuning.

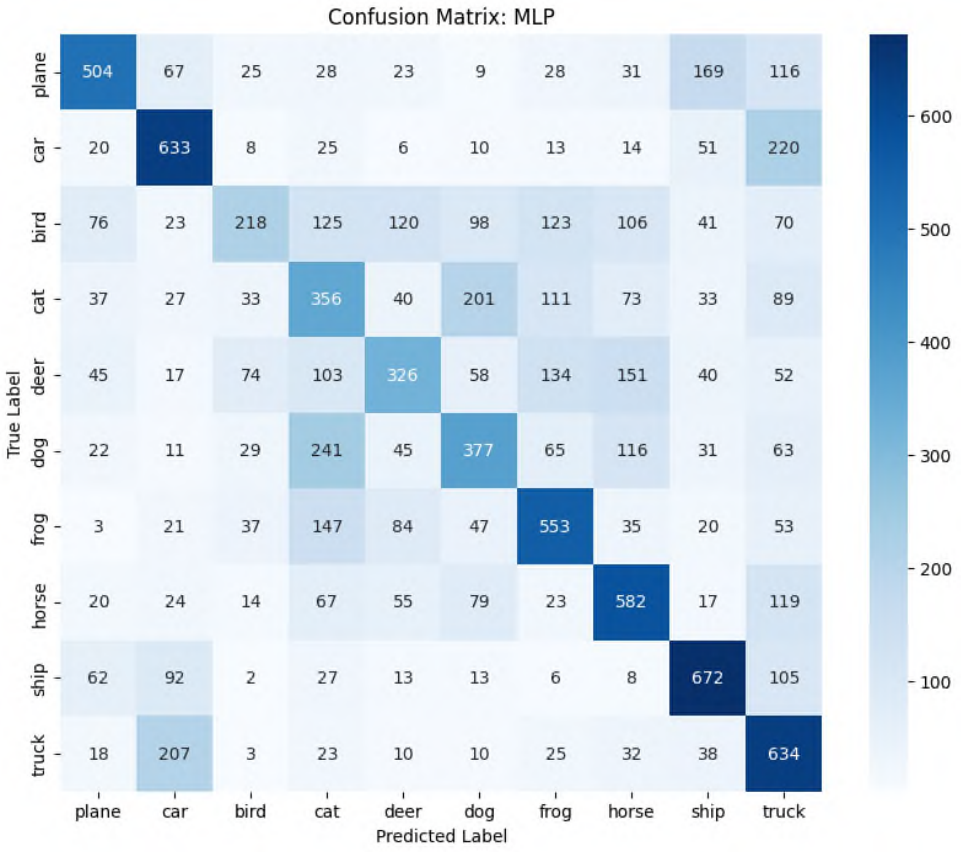


The learning curves are critical diagnostic tools. For the custom CNN, they clearly signal that while the model learns effectively, regularization could be strengthened or training could be stopped earlier to prevent performance degradation due to overfitting. For ResNet18, the curves highlight both the benefits of advanced schedulers and potential instabilities that might require tuning.

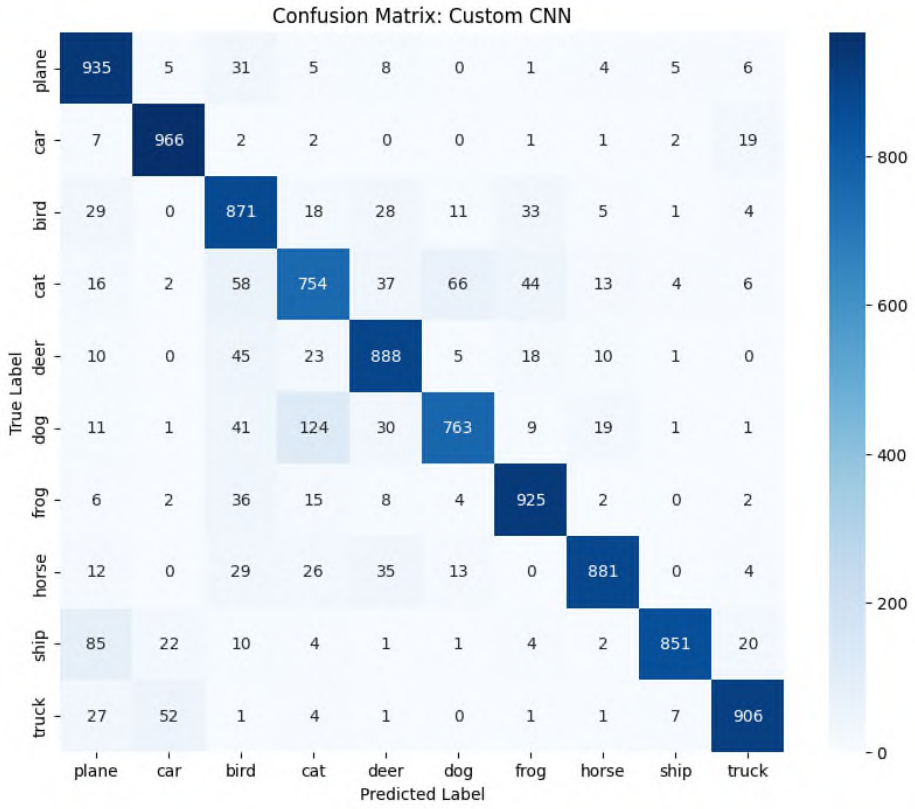
5.4. Error Analysis through Confusion Matrices

Confusion matrices visualize the classification performance for each class, detailing correct and incorrect predictions.

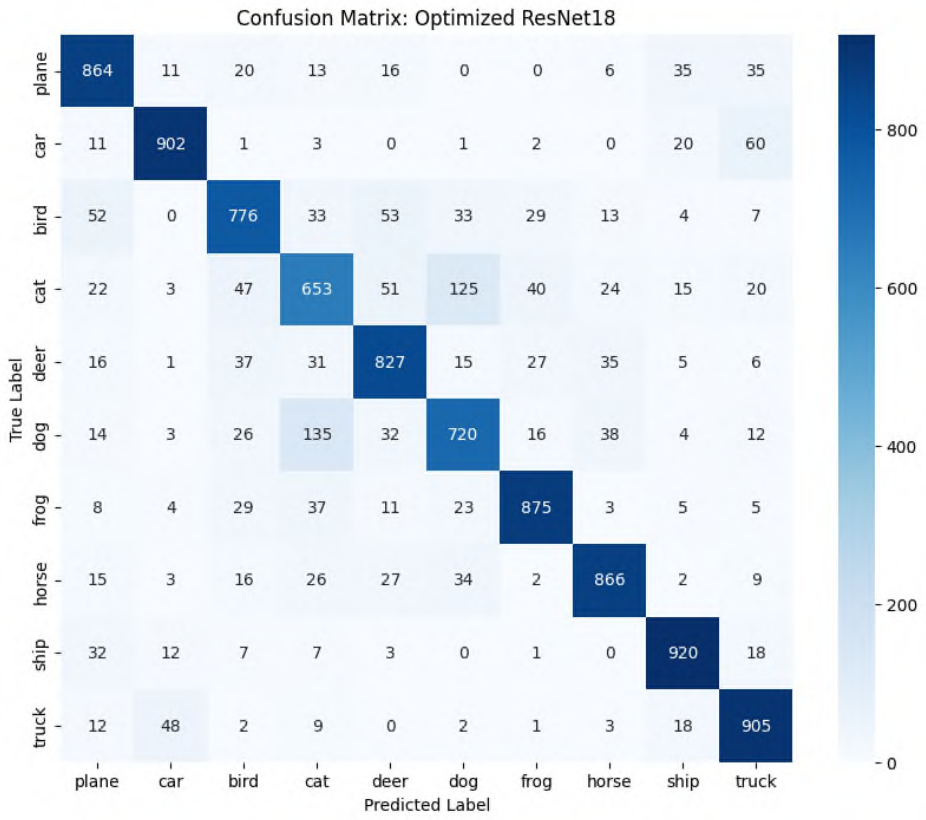
- **MLP:** The confusion matrix for the MLP [1] reflects its poor overall performance. The diagonal entries (correct classifications) are relatively low for most classes (e.g., only 220 'bird' images correctly classified out of 1000). There is widespread confusion, with significant numbers of instances being misclassified into various other classes, indicating a lack of strong discriminative features.



- **Custom CNN:** The confusion matrix for the custom CNN [1] shows a much stronger diagonal. However, it also highlights specific inter-class confusions. For instance, 'cat' is frequently confused with 'dog' (124 'cat' images predicted as 'dog', and 41 'dog' images predicted as 'cat'). 'Bird' is misclassified as 'deer' (45 instances) and 'cat' (58 instances). 'Ship' is sometimes mistaken for 'plane' (85 instances). These confusions often occur between classes that share visual similarities, especially at low resolution.



- **ResNet18:** The ResNet18 confusion matrix [1] reveals similar patterns of confusion. The 'cat' vs. 'dog' confusion is even more pronounced (125 'cat' images predicted as 'dog', and 135 'dog' images predicted as 'cat'). 'Bird' is also confused with 'deer' (53 instances) and 'cat' (47 instances). The consistent confusion between certain pairs, such as 'cat' and 'dog', by both the custom CNN and ResNet18—two capable but distinct architectures—suggests that these misclassifications may not solely be due to model-specific deficiencies. Instead, they likely point to inherent ambiguities and visual similarities between these classes within the 32x32 pixel CIFAR-10 images, making them intrinsically challenging to distinguish perfectly, even for advanced models.



5.5. Qualitative Analysis of Predictions

A qualitative examination of sample predictions provides anecdotal evidence of model behavior.1

- For a sample image of a 'cat', all three models (MLP, CNN, ResNet18) correctly classified it. The probability distributions indicated high confidence for the CNN and ResNet18.
- For a sample image of a 'ship', the MLP incorrectly classified it as a 'car'. The CNN and ResNet18 both correctly identified it as a 'ship' with high confidence.
- For another 'ship' image, all three models provided the correct classification. The softmax probability distributions visualized alongside these predictions offer insights into model certainty. Correct predictions from stronger models like the CNN and ResNet18 generally exhibit more sharply peaked probability distributions, indicating higher confidence. In contrast, the MLP's incorrect prediction ('car' for 'ship') would still have an associated probability, and comparing its distribution to the confident correct predictions of the other models visually underscores the differences in discriminative power.

5.6. Proposed Visualizations

To further enhance the understanding of model behavior, several additional visualizations would be beneficial, as suggested in the user query [User Query Point 5]:

- **Model Architecture Diagrams:** Visual diagrams of the MLP, custom CNN, and ResNet18 architectures would provide readers with a clear, schematic overview of their structures, aiding comprehension. Tools like Netron or even simplified block diagrams drawn with Matplotlib could serve this purpose.
- **Feature Maps (for CNNs):** Visualizing the feature maps produced by various convolutional layers within the custom CNN and ResNet18 could offer valuable insights into the hierarchical feature learning process. This would allow observation of how early layers might respond to simple features like edges and textures, while deeper layers learn to represent more complex patterns and object parts.
- **Dimensionality Reduction Plots (t-SNE/PCA):** Applying dimensionality reduction techniques like t-SNE or PCA to the feature embeddings extracted from the penultimate layer of each model could help visualize the separability of the classes in the learned feature space. Such plots would illustrate how effectively each model clusters instances of the same class while separating them from other classes.

5.7. Hardware and Training Details

The experiments were conducted utilizing a CUDA-enabled GPU 1, which significantly accelerates the training of neural networks. The training durations for each model were recorded:

- MLP: Approximately 7 minutes and 56 seconds.[1]
- Custom CNN: Approximately 9 minutes and 17 seconds.[1]
- ResNet18: Approximately 14 minutes and 2 seconds.[1] These training times are relatively modest, indicating that the experiments are feasible on commonly available GPU hardware. This accessibility makes CIFAR-10 a practical dataset for comparative studies and educational purposes. The longer training time for ResNet18, despite fewer epochs, is expected due to its greater architectural depth and computational complexity per epoch.

6. Discussion

This section synthesizes the experimental results, offering a comparative analysis of the models, discussing their behaviors, the impact of training enhancements, and addressing unexpected outcomes.

6.1. Comparative Performance Analysis

- **MLP vs. CNN:** The performance disparity between the MLP (48.55% test accuracy) and the custom CNN (87.40% test accuracy) is stark and underscores a fundamental principle in image classification.[1] The MLP's significantly lower performance is primarily attributable to its inability to process spatial information effectively. By flattening the input image into a 1D vector, the MLP discards the crucial 2D structure and local pixel relationships that define visual patterns [User Query Point 3, User Query Point 8]. In contrast, the CNN, through its convolutional and pooling layers, is explicitly designed to learn hierarchical spatial features, from simple edges to complex object parts. This architectural advantage allows the CNN to build much more

powerful and relevant representations for image data, leading to its superior accuracy. The MLP's performance, while better than random guessing (10% for 10 classes), highlights its limitations for tasks requiring nuanced visual understanding.

- **Custom CNN vs. Pre-trained ResNet18:** A key and somewhat unexpected finding of this study is that the custom-designed CNN (87.40% accuracy after 30 epochs) outperformed the pre-trained ResNet18 model (83.08% accuracy after 20 epochs) under the specific experimental conditions.[1] This outcome merits careful consideration, as deeper, pre-trained models like ResNet are often expected to yield superior results due to their greater capacity and the benefit of transfer learning.[15] Several factors could contribute to this observation:
    - **Training Duration and Convergence:** The ResNet18 model was trained for 20 epochs, whereas the custom CNN was trained for 30 epochs. It is plausible that the ResNet18 model was undertrained and had not fully converged to its optimal performance on the CIFAR-10 dataset within this shorter training period. The ResNet18 learning curves [1] show validation accuracy generally still improving or capable of further improvement, suggesting that additional epochs might have led to higher accuracy.
    - **Architectural Suitability for 32x32 Images:** Standard ResNet architectures, including ResNet18, are typically designed with ImageNet (often 224x224 pixels) in mind. Their initial layers often involve aggressive downsampling (e.g., a 7x7 convolutional kernel with a stride of 2, followed by max pooling). If these initial layers are not specifically adapted for the much smaller 32x32 resolution of CIFAR-10 images (e.g., by using smaller kernels/strides or removing early pooling layers as suggested in some literature [17]), they might discard too much spatial information too early in the network. The ResNet18 implementation used in this study [1] does not appear to include such modifications to its initial layers, which could disadvantage it on small images. The custom CNN, being designed from scratch, might have an architecture more directly suited to the 32x32 input size.
    - **Optimization and Hyperparameter Sensitivity:** While ResNet18 employed advanced techniques like the OneCycleLR scheduler and stronger data augmentation, the specific hyperparameters (e.g., maximum learning rate of 0.01, weight decay of $1\times10-4$) might not have been perfectly optimal for this architecture and dataset over 20 epochs. The observed instability in the ResNet18 validation loss (spike at epoch 3 [1]) could also indicate sensitivity to the learning rate schedule or batch characteristics. The custom CNN's simpler ReduceLROnPlateau scheduler and different optimizer settings over a longer training period might have, in this instance, led to a more favorable point in the loss landscape.
    - **Complexity versus Dataset Size and Training Regime:** ResNet18 is a significantly more complex model than the custom CNN. While transfer learning helps to regularize and provide a good initialization, adapting a very deep model to a smaller dataset like CIFAR-10 can still be challenging. Without extensive fine-tuning or a very carefully calibrated training regime, a simpler model that is well-matched to the dataset's complexity and size can sometimes achieve competitive or even superior performance, especially if the training budget (e.g., number of epochs) is constrained. Some studies have noted that simply increasing layers does not always guarantee better performance, and can sometimes lead to degradation if not managed properly.[12]

This comparison illustrates that there is no universally "best" model; the "no free lunch" theorem holds relevance. The optimal choice depends on a confluence of factors including dataset characteristics (size, resolution), available computational budget (epochs, tuning time), the specifics of architectural adaptation for the input data, and the thoroughness of hyperparameter optimization. A more complex or pre-trained model is not an automatic guarantee of superior performance over a well-designed, appropriately regularized simpler model, particularly if the former is not optimally tuned or trained for the specific constraints of the task.

6.2. Analysis of Model Behaviors and Limitations

- **MLP:** The primary limitation of the MLP is its inherent inability to process and leverage spatial information from images, leading to poor performance on visual tasks beyond the very simplest. It serves as a valuable baseline but is not a practical choice for complex image classification.
- **Custom CNN:** This model demonstrates a strong capacity for learning relevant visual features from CIFAR-10. The learning curves [1] indicate a tendency towards overfitting in the later stages of training, suggesting that

additional regularization (e.g., more aggressive data augmentation, stronger weight decay) or more training data could be beneficial for pushing performance further. The confusion matrix [1] reveals specific inter-class confusions (e.g., 'cat' vs. 'dog', 'ship' vs. 'plane'), highlighting areas where the model's discriminative power could be improved or where dataset ambiguities persist.

- **ResNet18:** The ResNet18 model showcases the potential of deep learning architectures and transfer learning. However, its performance in this study suggests that realizing this full potential on a specific target dataset like CIFAR-10 requires careful adaptation of the architecture (potentially the early layers for small image sizes) and sufficient training epochs for effective fine-tuning. The observed instability in validation loss during training and the noticeable gap between training and validation accuracy also point to areas for further optimization in the training process or hyperparameter settings. Its confusion matrix [1] indicates similar difficulties with visually ambiguous classes as the custom CNN.

6.3. Impact of Training Enhancements

The training techniques employed played a significant role in the achieved performances.

- The use of **AdamW** for the CNN and ResNet18 likely provided more effective weight decay compared to standard Adam, aiding regularization.
- The **learning rate schedulers** (ReduceLROnPlateau and OneCycleLR) were crucial for navigating the loss landscape. OneCycleLR, used with ResNet18, is particularly known for facilitating rapid training and potentially enabling models to reach better performance levels by allowing for exploration of higher learning rates followed by annealing.[17]
- **Batch normalization**, present in both the custom CNN and ResNet18, contributed to stabilizing training, allowing for faster convergence, and providing a slight regularization effect.
- The **data augmentation** strategies, especially the more aggressive set applied to ResNet18, were vital for regularizing these models and improving their generalization by exposing them to a wider variety of input transformations. The success of modern deep learning relies not only on sophisticated architectures but equally on these advanced training methodologies. The pairing of ResNet18 with techniques like AdamW and OneCycleLR is an example of this co-evolution, where optimizers and learning rate schedules are developed and refined to effectively train increasingly complex models.

6.4. Addressing Challenges and Unexpected Results

The onset of overfitting observed in the custom CNN's learning curves was managed to some extent by the existing regularization measures (dropout, weight decay, data augmentation) and the implicit early stopping mechanism (saving the best model based on validation accuracy). Further mitigation could involve more aggressive data augmentation techniques (discussed in Future Work), increased dropout rates, or stronger weight decay.

The performance of ResNet18 relative to the custom CNN is the most notable unexpected result. This should not be interpreted as a definitive statement that ResNet18 is inferior for CIFAR-10, but rather as an outcome specific to the experimental setup used. As discussed, factors like the number of training epochs, the lack of adaptation of early ResNet layers for 32x32 inputs, and potentially suboptimal hyperparameters for the 20-epoch training run are likely contributors. This highlights the importance of meticulous experimental design and thorough tuning when comparing models, especially when transfer learning is involved.

7. Conclusion and Future Work

7.1. Summary of Key Findings

This comparative study evaluated the performance of an MLP, a custom CNN, and a pre-trained ResNet18 model on the CIFAR-10 image classification task. The key findings are:

- The custom CNN significantly outperformed the MLP, achieving a test accuracy of 87.40% compared to the MLP's 48.55%. This stark difference underscores the critical importance of architectures that can learn and exploit spatial hierarchies of features in image data.

- Under the specific experimental conditions of this study (notably, 30 epochs for the custom CNN versus 20 epochs for ResNet18, and no adaptation of ResNet18's initial layers for small images), the custom CNN achieved a higher test accuracy than the pre-trained ResNet18 model (83.08%).
- The analysis highlighted the benefits of various training enhancements, including advanced optimizers (AdamW), dynamic learning rate scheduling (ReduceLROnPlateau, OneCycleLR), batch normalization, and data augmentation, in achieving the reported performances and regularizing the models.
- Error analysis through confusion matrices revealed common misclassification patterns (e.g., between 'cat' and 'dog'), suggesting inherent visual ambiguities in the CIFAR-10 dataset at its 32x32 resolution.

7.2. Implications of the Study

The results of this study have several practical implications:

- They reinforce the well-established principle that for image classification tasks, even on datasets with small image dimensions like CIFAR-10, CNNs are fundamentally more appropriate and effective than general-purpose MLPs due to their inherent ability to process spatial information.
- The study suggests that for moderately sized datasets and potentially constrained training budgets (in terms of epochs or tuning time), a well-designed custom CNN tailored to the dataset's characteristics can sometimes be more effective or easier to optimize to a high level of performance than a more complex, pre-trained deep model if the latter is not carefully adapted or sufficiently trained and fine-tuned for the specific target task.

7.3. Limitations of the Current Study

This study, while providing valuable comparisons, has certain limitations:

- Only a single custom CNN architecture and one specific pre-trained model (ResNet18) were evaluated. The vast landscape of other CNN architectures and pre-trained models was not explored.
- The hyperparameters for all models were fixed based on the provided experimental script and were not subjected to extensive, systematic tuning.
- The ResNet18 model was trained for fewer epochs (20) compared to the custom CNN and MLP (30 epochs), which may have limited its ability to reach its full performance potential.
- The adaptation of the pre-trained ResNet18 model involved replacing the final classifier but did not include modifications to its early convolutional layers, which might be beneficial for optimizing performance on small 32x32 input images.[17]

7.4. Proposed Future Work

Based on the findings and limitations of this study, several avenues for future research are proposed:

- **Exploration of Deeper and Wider Architectures:** Investigate the performance of more advanced CNN architectures, such as deeper variants of ResNet (e.g., ResNet34, ResNet50, which have shown strong results on CIFAR-10 [8]) or other state-of-the-art architectures like EfficientNets or Vision Transformers adapted for this scale.
- **Enhanced Transfer Learning Strategies:** Implement and evaluate more sophisticated fine-tuning techniques for pre-trained models. This could include layer-wise unfreezing (gradually unfreezing more layers of the pre-trained network), using differential learning rates for pre-trained versus newly initialized layers, and specifically adapting the early layers of pre-trained models (e.g., modifying kernel sizes, strides, or removing initial pooling) to better suit the 32x32 resolution of CIFAR-10 images.[15]
- **Advanced Data Augmentation Techniques:** Incorporate and assess the impact of state-of-the-art data augmentation methods such as AutoAugment [22], Mixup [24], or CutMix.[26] These techniques have demonstrated the potential to significantly boost performance on CIFAR-10 by further improving model regularization and robustness.
- **Extensive Hyperparameter Optimization:** Conduct a systematic and thorough hyperparameter tuning process for all models. This would involve searching over a range of values for key hyperparameters (e.g.,

learning rates, weight decay coefficients, batch sizes, dropout rates, optimizer-specific parameters) using methods like random search, grid search, or Bayesian optimization [19] to maximize the performance of each architecture.

- **Longer Training for ResNet18 and Comparative Models:** Re-run experiments, particularly for ResNet18, with an increased number of training epochs (e.g., matching or exceeding the 30 epochs used for the custom CNN) to ensure that models are given sufficient opportunity to converge and reach their optimal performance.
- **Implementation of Additional Visualizations:** Generate the proposed visualizations, including model architecture diagrams, feature maps from convolutional layers, and t-SNE/PCA plots of feature embeddings [User Query Point 5], to gain deeper qualitative insights into model behavior and feature representations.
- **Broader Evaluation Metrics:** Consider incorporating evaluation metrics that assess model robustness beyond standard accuracy, such as performance against adversarial attacks or on out-of-distribution samples.

By addressing these areas, a more comprehensive understanding of the relative strengths and optimal application contexts for different neural network architectures on CIFAR-10 and similar image classification tasks can be achieved.