Comparative Evaluation of MLP, CNN, and SVM Models for Image Classification on the CIFAR-10 Dataset

Group Number: 127

Student ID: 540303144, 520325186, 530419471, 550251668

Date: October 26, 2025

1 Introduction

This study aims to develop and evaluate image classification models on the CIFAR-10 dataset. Specifically, we seek to compare the performance of different machine learning and deep learning algorithms in accurately classifying small colour images (32×32 pixels) into ten predefined object categories.

The CIFAR-10 dataset holds significance as a widely adopted benchmark for evaluating image classification systems and a realistic representation of small, natural images across multiple object categories. Its diversity and balanced class distribution make it ideal for testing both traditional and deep learning approaches.

MLP captures nonlinear relationships between input pixels and class labels, CNN automatically extracts spatial features from raw images, and SVM can learn clear decision boundaries in high-dimensional feature spaces. Comparing these algorithms helps us understand how different model architectures handle image data, what trade-offs exist between accuracy, interpretability, and computational cost, and which approach is most appropriate for similar classification tasks.

2 Data

2.1 Data Description and Exploration

The dataset used is the CIFAR-10 collection, which contains 60,000 colour images divided into ten mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image is of size 32×32 pixels with three RGB channels. [1]

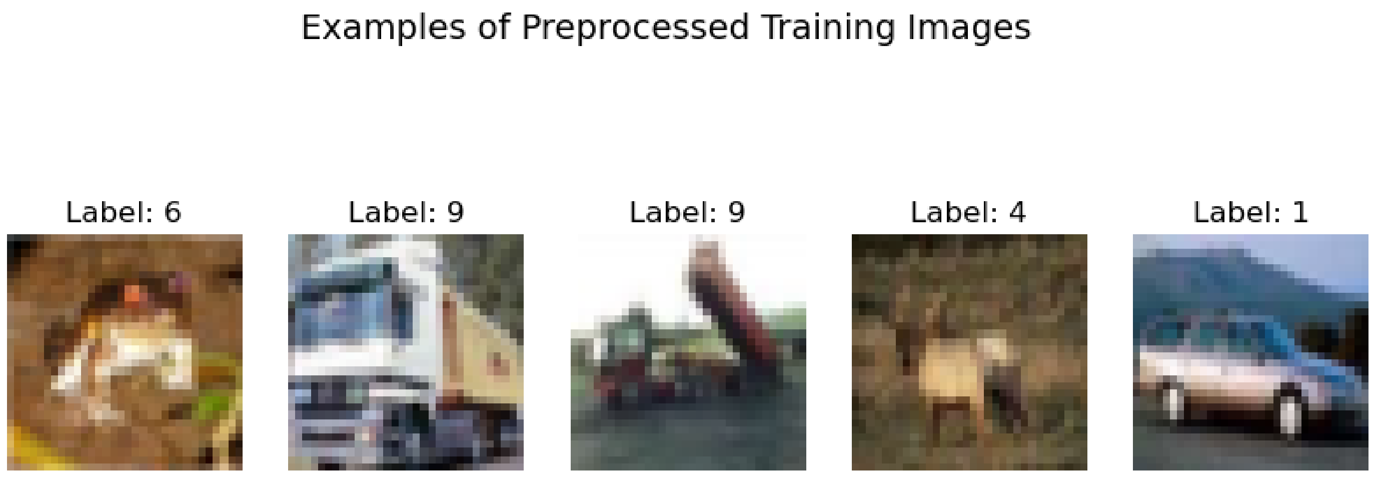


Figure 1: Examples of preprocessed CIFAR-10 training images.

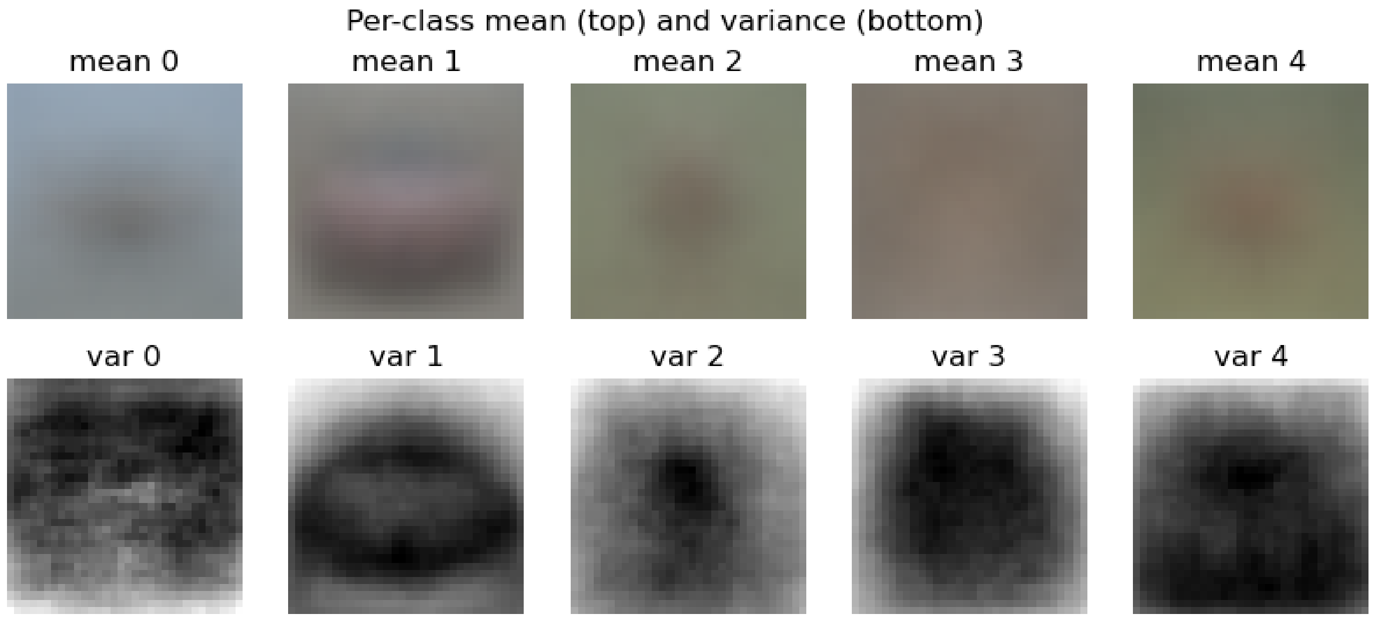
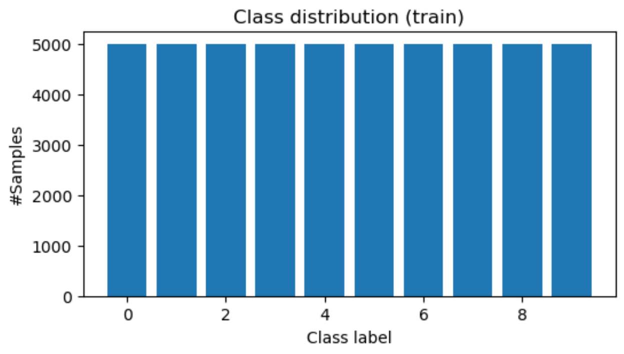
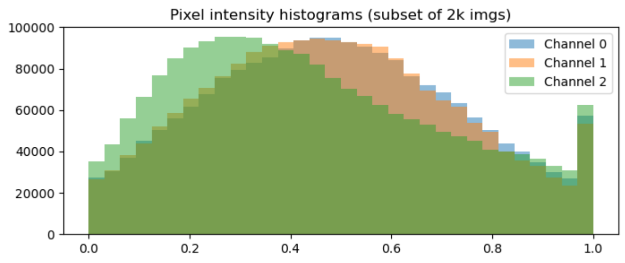


Figure 2: Per-class mean (top) and variance (bottom) images illustrating visual diversity.

Visual Characteristics: Example preprocessed images (Figure 1) show significant intra-class variation and inter-class similarity, posing challenges such as confusing animals (e.g., cats vs. dogs) or vehicles (e.g., trucks vs. automobiles). Pixel intensity histograms (Figure 3b) indicate well-normalized values between 0 and 1 with mean pixel intensity around 0.33 for each of the three channels. Mean and variance maps per class (Figure 2) further reveal that the pixel variance in the background area is higher, while the structure of the central target part is relatively stable.



(a) Class distribution in the training dataset.



(b) Pixel intensity histograms for RGB channels (subset of 2k images).

Figure 3: Exploratory data summary visualizations.

Class Distribution: The dataset is perfectly balanced, with 5,000 images per class in the training set (Figure 3a). This ensures that models are not biased toward any particular class.

Per-Channel Statistics: The dataset has mean pixel intensities of $[0.32768, 0.32768, 0.32768]$ and standard deviations of $[0.2776, 0.2693, 0.2681]$, confirming that each channel is similarly distributed and normalized.

Training and Testing Split: 50,000 images are used for training and 10,000 for testing. Within the training set, an 80/20 split creates 40,000 training and 10,000 validation samples for model tuning and early stopping.

Challenges: Classification is complicated by small image size, low resolution, and visually similar features among certain classes, requiring models that can capture fine-grained spatial details.

2.2 Preprocessing

Global Normalization:

All images were converted from integer pixel intensities in $[0, 255]$ to `float32` and rescaled to $[0, 1]$. This conversion ensures more stable gradient-based optimization. CIFAR-10 contains ten uniformly balanced classes, so no class weighting or resampling was required. A stratified 80/20 split of the training data was used to form validation sets, maintaining equal class proportions for fair model comparison.

Multilayer Perceptron (MLP):

Each $32 \times 32 \times 3$ image was flattened into a 3,072-dimensional feature vector so that it could be processed by fully connected layers [2]. The labels were one-hot encoded to support Softmax-based multi-class classification [3]. The flattening step, while necessary for fully connected networks, loses the spatial structure of the image. This limits the MLP's ability to learn local visual patterns, thus compromising performance. [4, 5].

Convolutional Neural Network (CNN):

To preserve spatial information, images were kept in (H, W, C) tensor form. After global scaling, per-channel standardization was applied using dataset-wide mean and variance, to ensure comparable feature magnitudes across RGB channels and to improve optimization stability [6]. This representation enables convolutional layers to extract localized features such as edges and textures, aligning with the spatial patterns observed in exploratory analysis [4, 7].

Support Vector Machine (SVM):

For SVMs, each image was flattened into a one-dimensional vector to fit the required tabular input format [8]. The vectors were standardized to zero mean and unit variance to ensure all features contributed equally to margin-based decision boundaries [9].

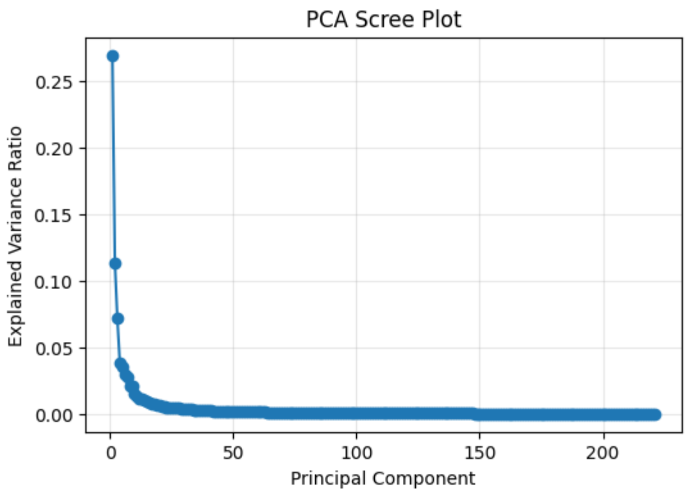


Figure 4: SVM PCA Scree Plot

Due to the high dimensionality (3,072 features), Principal Component Analysis (PCA) was applied to retain $\approx 95\%$ variance while reducing computational complexity and improving generalization [10, 11], as illustrated in Figure 4 .

3 Methods

3.1 Theory

3.1.1 Multilayer Perceptron

A Multi-Layer Perceptron (MLP) is a feedforward neural network composed of an input layer, one or more hidden layers, and an output layer [2]. Each layer applies a weighted affine transformation followed by a nonlinear activation:

$$\mathbf{a}^{(l)} = g(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}),$$

where $g(\cdot)$ is typically ReLU or sigmoid. Non-linear activations allow MLPs to model functions that are not linearly separable, such as XOR [3]. The model is trained using gradient-based optimization with backpropagation to compute derivatives efficiently [5]. Deeper MLPs can learn increasingly abstract representations, enabling them to approximate complex input–output mappings [6].

3.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are designed to exploit the spatial structure of image data [4]. A convolutional layer applies small learnable filters across the image to detect local features such as edges and textures:

$$Z(i,j) = \sum_{m,n} X(i+m, j+n) \cdot K(m,n),$$

where K is the convolutional kernel. Weight sharing and local connectivity significantly reduce the number of parameters compared to fully connected layers [5]. Pooling layers downsample feature maps, providing translation invariance and reducing overfitting [6]. Stacking multiple convolutional layers allows CNNs to learn hierarchical features, progressing from low-level edges to high-level semantic patterns [4].

3.1.3 Support Vector Machine

Support Vector Machines (SVMs) aim to learn a decision boundary that maximises the margin between classes [12]. For the linear case, the classifier is:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b,$$

and maximizing the margin $\frac{2}{\|\mathbf{w}\|}$ improves generalisation [3]. Soft-margin SVM introduces a regularization parameter C to balance margin width and classification error [11]. To model non-linear boundaries, the *kernel trick* maps inputs into a higher-dimensional feature space implicitly via kernel functions such as the RBF kernel [5]. This enables SVMs to learn flexible decision boundaries without explicitly constructing high-dimensional feature representations.

3.2 Strengths and Weakness

3.2.1 Multilayer Perceptron

MLPs can model complex non-linear relationships through layered feature transformations and non-linear activation functions [3]. However, because the input is flattened, MLPs treat pixels independently and lose spatial structure that is essential for visual recognition tasks [4]. This leads to weaker performance compared to CNNs on image data. Additionally, fully connected layers introduce a large number of parameters, making MLPs prone to overfitting and computationally expensive when input dimensionality is high [5]. Regularization such as dropout and early stopping is therefore necessary to improve generalization. The model's interpretability remains limited as hidden-layer features do not correspond to human-interpretable visual concepts.

3.2.2 Convolutional Neural Network

CNNs leverage local connectivity and weight sharing to extract hierarchical spatial features (edges $\rightarrow$ textures $\rightarrow$ objects), making them well-suited for image classification tasks where spatial structure is informative [4]. This architectural bias allows CNNs to achieve strong performance while using fewer parameters than equivalently sized MLPs [5]. Nonetheless, CNNs can still overfit without appropriate regularization (e.g., data augmentation or dropout), and training is computationally demanding due to convolutional operations and backpropagation through multiple layers. Although intermediate feature maps can sometimes be visualized, full interpretability remains limited.

3.2.3 Support Vector Machine

SVMs maximize decision margin, which contributes to robust generalization, and can model non-linear boundaries via kernel functions [3]. Yet, when applied directly to raw images, SVMs lack hierarchical feature extraction and therefore require explicit dimensionality reduction or hand-crafted features to perform well [4]. Kernel SVMs also become computationally expensive on large datasets because training complexity scales poorly with the number of samples [11]. While linear SVMs offer some interpretability through feature weights, kernel SVMs are more opaque and require careful tuning of hyperparameters such as C and γ to avoid both underfitting and overfitting.

3.3 Architecture and Hyperparameter Settings

3.3.1 Multilayer Perceptron

The MLP uses a fully connected architecture applied to flattened $32 \times 32 \times 3$ CIFAR-10 images (3,072 features). Flattening the input removes spatial relationships, meaning the model to learn patterns purely through non-linear feature transformations. The network consists of a dense ReLU-activated hidden layer, followed by dropout for regularization, a second reduced hidden layer for feature compression, and a final 10-unit softmax output layer. The model is trained using the Adam optimizer with categorical cross-entropy loss. Key hyperparameters include the number of hidden units (capacity), dropout rate (regularization strength), learning rate (optimization stability), and number of epochs. Hyperparameter tuning was performed using random search over $\{128, 256, 512\}$ units, dropout $\{0.2, 0.3, 0.5\}$, and learning rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$, with early stopping and fixed random seeds for reproducibility.

3.3.2 Convolutional Neural Network

The CNN architecture consists of two convolution–pooling blocks (Conv 3×3 with 32 filters, MaxPool; Conv 3×3 with 64 filters, MaxPool), followed by flattening, dropout, and a softmax classifier. Convolutional layers learn hierarchical spatial features (edges $\rightarrow$ textures $\rightarrow$ object parts), while max pooling reduces spatial resolution and aids generalization. Hyperparameters include the number of filters (representation capacity), dropout rate (regularization), and learning rate (optimization stability). Random search was used to tune these parameters, with model selection based on validation accuracy to ensure generalization rather than overfitting.

3.3.3 Support Vector Machine

An RBF-kernel SVM was trained on PCA-transformed image features to reduce dimensionality while preserving most discriminative variance. PCA also reduces computational load, which is necessary because kernel SVM training scales poorly with sample size. The key hyperparameters are the regularization parameter C, controlling the margin–error trade-off, and the kernel width γ, controlling boundary smoothness. A two-stage tuning procedure was applied: coarse random search to identify a suitable parameter region, followed by fine-grained search using stratified cross-validation and evaluation using accuracy and macro-F1. This staged approach balances computational efficiency with reliable model selection.

4 Results and Discussion

4.1 Hyperparameter Tuning Results

4.1.1 Multilayer Perceptron

Table 1: Summary of MLP Hyperparameter Tuning Results

Metric	**Value**
Best validation accuracy	0.5009
Validation accuracy of final trial	0.4630
Number of trials	12
Epochs per trial	15
Batch size	128
Total tuning time	13.65 minutes

The MLP tuning results in Table 1 show that the best validation accuracy reached approximately 0.50, indicating that the model did not learn sufficiently discriminative representations from flattened pixel inputs. This outcome aligns with expectations, as removing spatial structure limits the model's ability to capture local visual patterns, making CIFAR-10 a challenging dataset for fully connected architectures.

Table 2: MLP hyperparameter tuning results (sorted by validation accuracy)

Units	**Dropout**	**Learning rate**	**Validation Accuracy**
512	0.2	0.0001	0.5012
512	0.3	0.0001	0.4977
256	0.2	0.0001	0.4819
512	0.2	0.0010	0.4788
256	0.2	0.0010	0.4685
128	0.2	0.0001	0.4629
256	0.3	0.0010	0.4372
128	0.2	0.0010	0.4354
512	0.5	0.0010	0.4031
256	0.5	0.0010	0.3524

Table 2 shows that the choice of hidden layer size, dropout rate, and learning rate has a substantial effect on MLP performance. The best validation accuracy (0.5012) was achieved with 512 units, a dropout rate of 0.2, and a learning rate of 0.0001. This configuration suggests that a relatively larger hidden layer provides sufficient representational capacity for modeling the variation in CIFAR-10 images, while a moderate dropout rate helps prevent overfitting without disrupting learning stability. Trials with a higher dropout rate (e.g., 0.5) consistently underperformed, likely because excessive regularization restricted the model's ability to learn meaningful features. Similarly, trials using a learning rate of 0.001 tended to show less stable convergence and lower validation accuracy, indicating that the step size was too large for reliable optimization in this high-dimensional setting.

The tuning trends align with expectations: models that are too small struggle to capture complex image patterns, whereas models that are too heavily regularized fail to learn effectively. Moderate model capacity and a conservative learning rate resulted in the most stable and generalizable performance. These results reinforce that the MLP can learn useful class-distinguishing patterns on CIFAR-10, but its performance remains constrained by the loss of spatial structure inherent to flattening the input images.

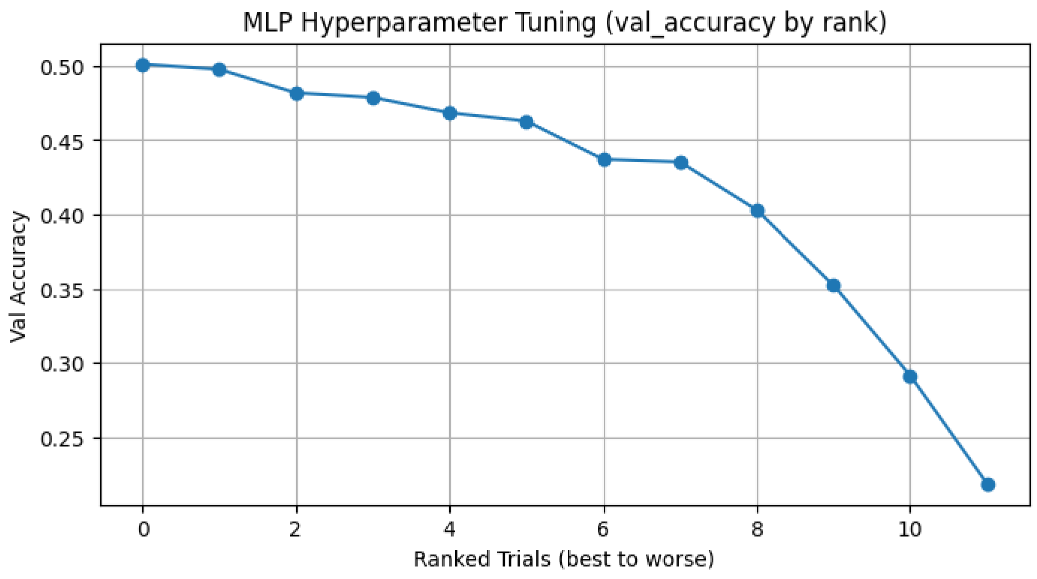


Figure 5: Validation accuracy across hyperparameter combinations for MLP.

Figure 5 illustrates the validation accuracy achieved across different hyperparameter configurations for the MLP. A clear downward trend is observed as we move from the best-performing trials to the poorer ones. The top-ranked models achieve validation accuracies close to 0.50, whereas later configurations fall gradually and eventually drop below 0.30. This pattern suggests that the MLP is highly sensitive to the balance between representational capacity and regularization.

The best-performing configurations generally used a relatively large number of hidden units (e.g., 256–512) combined with moderate dropout rates (0.2–0.3) and small learning rates (e.g., 10^{-3} or 10^{-4}). These settings provide sufficient expressive power while maintaining optimization stability. In contrast, models with fewer units or higher dropout (e.g., 0.5) tend to underfit, resulting in lower accuracy, while larger learning rates (e.g., 10^{-2}) cause unstable or divergent training. These outcomes align with expectations: since the MLP processes flattened images and therefore does not leverage spatial structure, it relies more heavily on parameter count to model visual patterns, making appropriate regularization and careful learning rate control essential to avoid overfitting or optimization instability.

4.1.2 Convolutional Neural Network

The CNN tuning results shown in Table 3 indicate that the model achieved a peak validation accuracy of approximately 0.76, substantially higher than the MLP. This performance stability suggests that convolutional layers effectively capture spatial patterns in CIFAR-10 images, making model performance less sensitive to

Table 3: Summary of CNN Hyperparameter Tuning Results

Metric	**Value**
Best validation accuracy	0.7562
Validation accuracy of final trial	0.7562
Number of trials completed	10
Epochs per trial	15
Batch size	128
Total tuning time	143.84 minutes

moderate hyperparameter variation. The tuning process thus confirms that representational capacity and spatial inductive bias are key factors enabling the CNN to generalize more effectively than the MLP.

Table 4: CNN hyperparameter tuning results (sorted by validation accuracy)

Base Filters	**Dropout Rate**	**Learning Rate**	**Validation Accuracy**
48	0.50	0.0010	0.7562
64	0.35	0.0010	0.7562
64	0.50	0.0005	0.7518
48	0.35	0.0005	0.7478
48	0.35	0.0010	0.7428
32	0.50	0.0010	0.7426
32	0.20	0.0010	0.7398
64	0.50	0.0001	0.6924
48	0.35	0.0001	0.6754
32	0.50	0.0001	0.6528

Table 4 shows that the highest-performing configurations consistently involve moderate to large base filter sizes (48–64 filters) combined with medium dropout rates (around 0.35). These settings provide sufficient representational capacity to extract informative spatial features while maintaining regularization to prevent overfitting. Models with smaller filter banks (e.g., 32 filters) generally yield lower validation accuracy, indicating a reduced ability to capture the visual variability present in CIFAR-10.

Learning rate also played a key role. Moderate values (e.g., 10^{-3} and 5×10^{-4}) supported stable convergence and effective learning, while very small learning rates slowed optimization and occasionally led to underfitting. Conversely, the poorest-performing configurations tended to combine either insufficient filters or overly aggressive regularization with very low learning rates, resulting in limited feature learning.

Overall, the tuning results align with theoretical expectations: CNNs benefit from increased feature extraction capacity, but require appropriately balanced dropout and well-scaled learning rates to achieve good generalization.

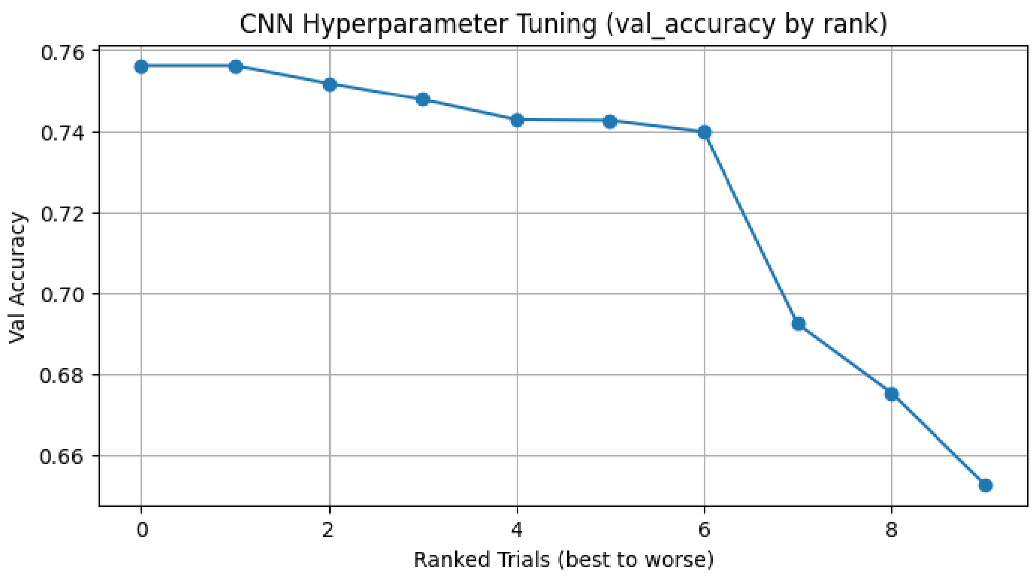


Figure 6: Validation accuracy across hyperparameter combinations for CNN

Figure 6 shows the validation accuracy across ranked CNN hyperparameter configurations. The top-performing trials are tightly clustered (approximately 0.74–0.76), indicating that the CNN is relatively robust to moderate changes in filter size, dropout rate, and learning rate. This stability reflects the inductive bias of convolutional layers, which effectively capture spatial structure. In contrast, the lower-ranked trials show a sharp decline in accuracy, mainly when the base number of filters is reduced or the learning rate becomes too small, leading to insufficient feature learning and underfitting. Overall, the tuning curve confirms that adequate representational capacity and moderate regularization are key to achieving strong CNN performance.

4.1.3 Support Vector Machine

Table 5: Stage 1 (Coarse) Hyperparameter Search Results for RBF-SVM

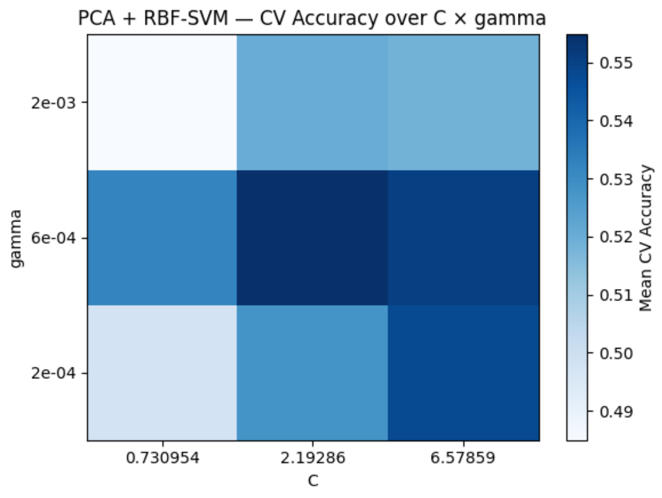
C	γ	**Kernel**	**Mean CV Accuracy**	**Mean Time (s)**
2.19	5.87e-4	rbf	0.4650	34.64
5.02	4.14e-4	rbf	0.4608	35.91
4.92	4.44e-4	rbf	0.4602	35.31
4.59	3.34e-4	rbf	0.4593	30.00
1.15	6.97e-4	rbf	0.4589	33.62

As shown in Table 5, the top-performing configurations yielded mean cross-validation accuracy of approximately 0.46, indicating moderate classification performance under the coarse search setting. The average runtime per configuration ranged from 30 to 36 seconds when combining training and scoring time, reflecting the computational cost of fitting RBF-SVMs even on the reduced 10,000-sample subset. These results highlight the trade-off between model complexity and efficiency: while increasing C and γ can refine the decision boundary, excessively high values do not improve accuracy but still incur substantial computational cost, reinforcing the need for a two-stage tuning strategy.

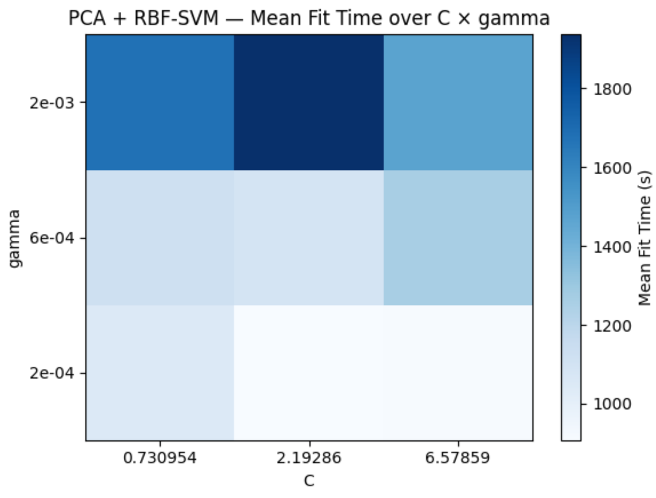
Table 6: Stage 2 (Fine) Hyperparameter Search Results for RBF-SVM

C	γ	**Mean CV Accuracy**	**Mean Time (s)**
2.19	5.87e-4	0.5548	1357.18
6.58	5.87e-4	0.5505	1517.48
6.58	1.96e-4	0.5476	1177.01
0.73	5.87e-4	0.5320	1426.00
2.19	1.96e-4	0.5280	1200.29

Table 6 shows that the fine search resulted in a clear performance improvement over the coarse search, with the best configuration achieving a mean cross-validation accuracy of approximately 0.55. However, this performance gain came at a substantial computational cost, with each configuration requiring between 1,100 and 1,500 seconds to evaluate due to the full training set being used. These results reinforce the need for a two-stage tuning strategy, where a small-scale coarse search identifies a promising region before expensive fine-grained optimization is performed.



(a)

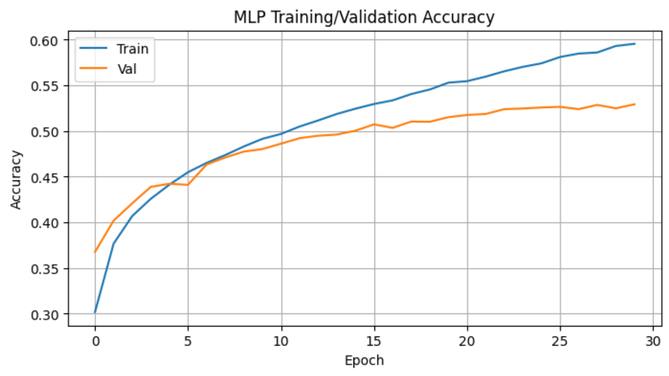


(b)

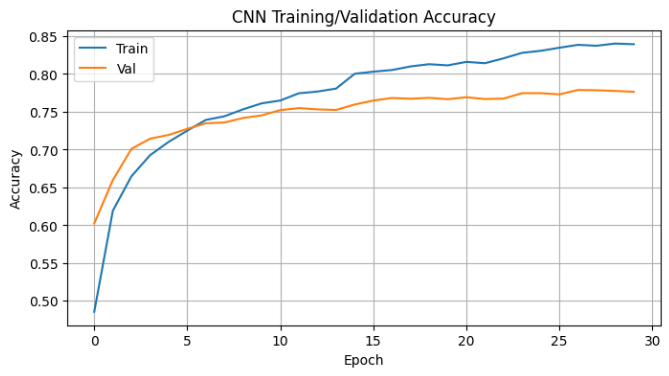
Figure 7

Figure 7 visualises the effect of the hyperparameters C and γ on both validation accuracy (Figure 7a) and computational cost (Figure 7b). The accuracy heatmap shows that performance peaks around $C \approx 2.2$ and $\gamma \approx 6 \times 10^{-4}$, indicating that a moderate margin constraint and a moderately narrow RBF kernel offer the best trade-off between flexibility and generalisation. Regions with very small γ underfit due to overly smooth decision boundaries, whereas very large C settings increase model capacity but risk overfitting without providing accuracy gains. In contrast, the fit-time heatmap demonstrates that larger C and larger γ dramatically increase training time, reflecting the higher number of support vectors and the computational burden of more complex decision boundaries. Taken together, the heatmaps highlight that optimal SVM performance emerges from balancing model flexibility against computational cost, reinforcing theoretical expectations for RBF-SVM behaviour.

4.2 Final Model Comparison



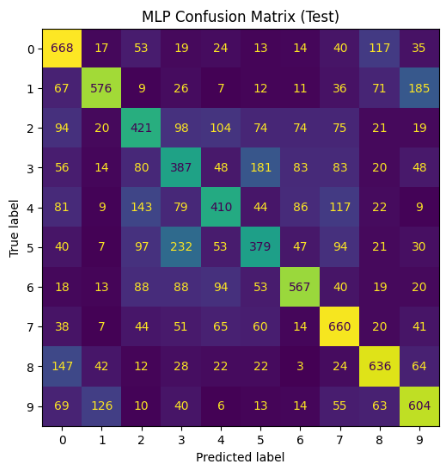
(a) MLP training and validation accuracy over epochs.



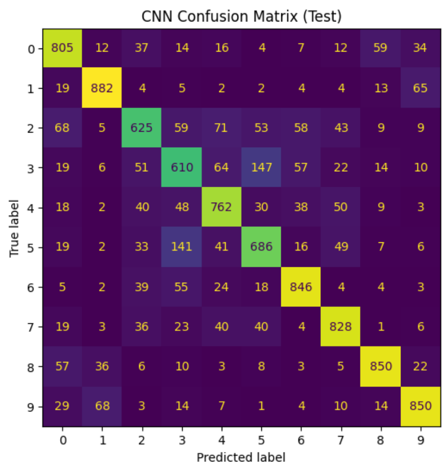
(b) CNN training and validation accuracy over epochs.

Figure 8: Comparison of training and validation accuracy trends for the MLP (left) and CNN (right) models.

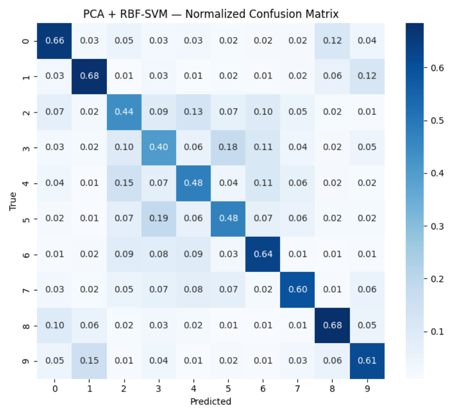
Figure 8 compares the training and validation accuracy curves for the MLP and CNN models. The CNN shows faster convergence within the first few epochs and consistently achieves higher validation accuracy, reflecting the benefit of convolutional layers in learning spatial patterns in image data. In contrast, the MLP improves more slowly and plateaus at a substantially lower validation performance, indicating that treating images as flattened vectors limits representational capacity. Both models exhibit a gap between training and validation performance, but the gap is noticeably smaller for the CNN, suggesting that the CNN achieves better generalisation relative to model capacity. Overall, the learning curves highlight that the CNN is better suited to the CIFAR-10 classification task due to its ability to capture hierarchical visual features.



(a) MLP



(b) CNN



(c) PCA + RBF-SVM

Figure 9: Comparison of confusion matrices for the final MLP, CNN, and PCA+RBF-SVM models on the CIFAR-10 test set.

Figure 9 compares the confusion matrices of the MLP, CNN, and PCA+RBF-SVM models on the CIFAR-10 test set. The MLP exhibits relatively high confusion across many classes, particularly among visually similar categories such as cats, dogs, and deer, indicating limited ability to extract meaningful spatial features from flattened inputs. The CNN shows the strongest diagonal dominance, reflecting consistently accurate predictions across all classes; this highlights the advantage of convolutional filters in capturing hierarchical visual patterns. The SVM performs moderately well after PCA reduction, achieving clearer separation than the MLP but still struggling where fine-grained spatial cues are important, as linearized features cannot fully preserve image structure. Overall, the comparison reinforces that models incorporating spatial inductive biases (CNN) classify image data more effectively than vector-based MLPs or kernel methods applied after dimensionality reduction.

Table 7: Comparison of final tuned models on CIFAR-10 test set

Model	**Accuracy**	**Macro-F1**	**Training Time (min)**	**Best Hyperparameters**
MLP	0.531	0.529	3.59	units=512, dropout=0.2, lr=10^{-4}
CNN	0.774	0.773	27.57	filters=48, dropout=0.50, lr=10^{-3}
SVM	0.566	0.566	$\sim$7	C=2.19, $\gamma = 5.86 \times 10^{-4}$

Table 7 shows that the CNN substantially outperformed both the MLP and the PCA+RBF-SVM, achieving the highest test accuracy (0.774) and Macro-F1 (0.773). This aligns with theoretical expectations: unlike the MLP and SVM, the CNN leverages convolutional kernels and local receptive fields that explicitly model spatial structure, enabling it to learn translation-invariant features from images. In contrast, the MLP operates on flattened pixel vectors, discarding spatial relationships entirely, which leads to weaker representational power and higher confusion among visually similar classes (e.g., cat–dog–deer), as reflected in its confusion matrix. The SVM benefits from the RBF kernel's non-linear decision boundaries, and applying PCA (96 components) significantly reduces input dimensionality and training cost; however, PCA cannot fully preserve fine-grained spatial relationships, which limits classification performance compared to the CNN.

In terms of runtime, the CNN required the longest training time ($\sim$28 minutes), driven by its multi-layer convolutional architecture and the need to train over many epochs to converge. The MLP trained the fastest ($\sim$3.6 minutes), as each pass involves only matrix multiplications over two dense layers, though this efficiency comes at the cost of reduced accuracy. The SVM's runtime ($\sim$7 minutes) was dominated by kernel matrix computation and support vector optimization, which scale poorly with dataset size; dimensionality reduction via PCA significantly reduced this cost but did not close the accuracy gap with the CNN.

The confusion matrices further illustrate characteristic error patterns across models. The CNN exhibits strong diagonal dominance across all classes, indicating consistent correct predictions. The MLP frequently confuses classes with similar textures and shapes (e.g., classes 3 vs. 5 and 7 vs. 9), reflecting its difficulty in capturing visual hierarchy. The SVM shows moderate performance, correctly classifying classes with distinctive shapes (e.g., trucks and airplanes) but struggling when fine texture cues are needed. Overall, the results reinforce that models with strong inductive biases for image structure (CNN) outperform generic vector-based classifiers on image data, particularly when classes require spatial feature discrimination.

5 Conclusion

This study compared the performance of MLP, CNN, and PCA-based RBF-SVM models on the CIFAR-10 image classification task, revealing clear trade-offs between accuracy, runtime, and interpretability. The CNN achieved the strongest performance (Accuracy = 0.774, Macro-F1 = 0.773), reflecting its ability to learn hierarchical spatial features directly from raw images. In contrast, the MLP trained fastest but performed the weakest, as flattening inputs discards spatial information that is essential for distinguishing visually similar classes. The PCA-RBF-SVM model achieved moderate performance and offered a balance between generalization and computational cost, but its reliance on dimensionality reduction limited its ability to capture fine-grained visual cues. Key limitations of this work include relatively shallow model architectures, limited data augmentation, and the use of a single global PCA projection for SVM, which may have removed important class-specific structure. Future work could therefore focus on (i) incorporating data augmentation to improve CNN generalization, particularly for visually similar animal classes; (ii) exploring lightweight CNN variants (e.g., MobileNet or EfficientNet) to reduce training time while retaining spatial feature learning; and (iii) replacing global PCA in the SVM pipeline with feature extractors such as Histogram of Oriented Gradients (HOG) or pretrained CNN feature embeddings, which could improve representational richness without requiring full deep training. These extensions would directly address current performance and efficiency limitations while remaining well-justified for this classification task.

6 Reflection

6.1 540303144

While working on this assignment, I learned a lot about how to clean and prepare data properly before training a model. I realized that good data is just as important as the model itself. I also got hands-on experience building a CNN and trying out different hyperparameter to see how they affect the results. It was really interesting to see how small changes like adjusting the learning rate or dropout could make a big difference. Overall, this task helped me understand not just how CNNs work, but also how to think like a machine learning engineer — testing, tweaking, and improving step by step.

6.2 520325186

The main thing I learned in this assignment was how to organise and present a complete machine learning workflow clearly. My role was to merge the notebooks and structure the report in LaTeX, which required understanding the purpose of each modelling step rather than just copying code. I realised that explaining why choices were made is just as important as showing results. I also learned the importance of consistency when combining work from different sources, including formatting figures, tables, and terminology.

6.3 530419471

In this assignment, I completed the full pipeline of SVM model construction, including data preprocessing and hyperparameter tuning. I realised that data standardisation has a greater influence on SVM performance than expected, and introducing PCA effectively reduced feature redundancy while improving interpretability. However, the RBF kernel significantly increased computational cost, making parameter tuning more time-consuming. By comparing the CNN and MLP models built by other group members, I understood that different model architectures respond very differently to preprocessing choices and parameter settings. This experience helped me recognise the need to balance data characteristics and computational efficiency when selecting models in real tasks.

6.4 550251668

Through training the MLP for this classification task, I gained experience in using a validation set along with the EarlyStopping to effectively control overfitting, as well as how dropout and learning rate may influence model generalization and convergence speed. Hyperparameters tunning can have a great impact on the stability of the neural network model like MLP and the final results. Additionally, experimental reproducibility is an essential part of this study. Effective ways to ensure reproducibility are fixing random seeds, separating a dedicated validation set, and recording every hyperparameter and result.I also learned how to build a clear and modular pipeline from data preprocessing, model construction, and hyperparameter tuning to independent training, testing, and visualization. Moreover, the report should be supported by solid evidence especially learning curves, confusion matrices, and tuning tables.

References

[1] A. Krizhevsky, "Cifar-10 and cifar-100 datasets," https://www.cs.toronto.edu/ kriz/cifar.html, 2009, [Online; accessed 24-Oct-2025].

[2] Y. Yao and I. Koprinska, "Deep learning i: Feedforward neural networks," 2025, cOMP5318 Lecture Slides.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[6] J. Kelleher, *Deep Learning.* MIT Press, 2019.

[7] Y. Yao and I. Koprinska, "Deep learning ii: Convolutional and recurrent networks," 2025, cOMP5318 Lecture Slides, Week 8.

[8] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2012.

[9] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Morgan Kaufmann, 2017.

[10] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[11] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. O'Reilly Media, 2019.

[12] T. Zhao, I. Koprinska, and Y. Yao, "Support vector machines & dimensionality reduction," 2025, cOMP5318 Lecture Slides, Week 6.