



**Prifysgol Cymru**  
Y Drindod Dewi Sant  
**University of Wales**  
Trinity Saint David

**Module-Title:**

Application of Machine Learning.

**Module-Code:**

ACCA7025

**Assignment-Title:**

Image Classification Using Artificial Neural Networks.

---

**Student-Name:**

MD Mahadi Hasan

**Student-ID:**

2505591

**Programme:**

MSc in Software Engineering and Artificial Intelligence

**University-Name:**

University of Wales Trinity Saint David (Swansea Campus)

---

**Lecturer:**

Mark Huntly

**Submission-Date:**

18<sup>th</sup> May 2026

## Abstract

In this report, the design, systematic optimization, and experimental analysis of a light-weight Convolutional Neural Network (CNN) for ten-class image classification in the CIFAR-10 benchmark dataset is presented. A CNN network of 3 blocks was created with TensorFlow/Keras, consisting of Conv2D, Batch Normalization, MaxPooling and Dropout layers. There were four comparative experiments; Adam vs SGD optimization algorithm, Dropout rate of 0.4 vs 0.5, and  $3\times 3$  convolution filter size vs  $5\times 5$  convolution filter size. The best test accuracy (77.3%) and weighted F1-score (0.769) were obtained with experiment D, which used Adam with  $5\times 5$  size of filters and a dropout rate of 0.4. This study shows that, with receptive field size adapted to image size and receptive field size appropriate, adaptive optimization strategies can be used to achieve better classification results on low-resolution images. The result of per-class analysis shows that semantically different classes like automobile and frog had recall of 94% while, visually ambiguous classes like cat (42%) and deer (53%) had the biggest classification problems. The study offers empirically support for architectural and hyperparameter decisions and situates the results in the existing deep learning landscape.

## Table of Contents

|                                                             |    |
|-------------------------------------------------------------|----|
| Abstract.....                                               | 2  |
| 1. Introduction.....                                        | 4  |
| 2. Literature Review.....                                   | 4  |
| 2.1 Foundations of Artificial Neural Networks and CNNs..... | 4  |
| 2.2 CNN Optimisation for CIFAR-10 .....                     | 5  |
| 2.3 Limitations of Existing Lightweight Frameworks.....     | 5  |
| 3. Methodology.....                                         | 6  |
| 3.1 Workflow Overview .....                                 | 6  |
| 3.2 Dataset and Preprocessing .....                         | 6  |
| 3.3 CNN Architecture Design.....                            | 7  |
| 3.4 Experimental Design.....                                | 7  |
| 4. Implementation .....                                     | 8  |
| 5. Results and Evaluation.....                              | 9  |
| 5.1 Training Dynamics.....                                  | 9  |
| 5.2 Optimiser Comparison .....                              | 11 |
| 5.3 Confusion Matrix Analysis .....                         | 11 |
| 5.4 Comparative Performance Summary .....                   | 13 |
| 6. Discussion.....                                          | 15 |
| 7. Conclusion .....                                         | 15 |
| References.....                                             | 16 |

## 1. Introduction

One of the most widely studied problems in computer vision has been image classification and Convolutional Neural Networks (CNNs) have proven themselves to be the most successful paradigm since Krizhevsky, Sutskever, and Hinton [1] developed AlexNet on the ImageNet benchmark. The CIFAR-10 dataset, introduced by Krizhevsky and Hinton [2], is a standardized benchmark with 60,000 color images across 10 balanced object classes, and is an important test case for CNN design and optimization strategies at limited image resolution ( $32 \times 32$  pixels). The near-human accuracy of large-scale networks like ResNets [3] or DenseNets [4] on CIFAR-10, however, comes with a high computational cost that makes them impractical for application in resource-limited settings. Lightweight CNNs, which aim for compactness of parameters and fast inference while preserving classification accuracy, are still an active research field with significant practical importance, therefore [5]. The main research question in this study is how can a compact CNN architecture be systematically optimized by tuning its hyperparameters and regularizing to achieve the highest classification accuracy for CIFAR-10 while maintaining computational efficiency? The report provides a systematic comparison of an experiment involving four different architectural configurations, with comparisons drawn on the basis of both test accuracy and generalization behavior, as well as the performance of each class across the four. The goal is not to design at the cutting edge of technology, but to vigorously test the compromises involved in making lightweight design decisions and to demonstrate grounded arguments for each design decision.

## 2. Literature Review

### 2.1 Foundations of Artificial Neural Networks and CNNs

This course covers the fundamentals of Artificial Neural Networks and CNNs. This course introduces the basics of Artificial Neural Networks and CNNs. The major theoretical basis of CNNs is the biological model of the behavior of neurons in the visual cortex, formalized in a computational model by the Neocognitron of Fukushima [6] and later by LeCun et al. [7] in their early work on gradient based learning for document recognition. An artificial neuron takes in input, calculates a weighted sum of the inputs, then applies a non-linear activation function, and feeds the output to the next layer. All modern architectures of convolutional neural networks are based on a hierarchical structure of convolution, subsampling and fully-connected layers that were

introduced by LeCun in his LeNet-5 architecture [7]. The ability to return the gradient information back through multiple layers was popularized by Rumelhart, Hinton and Williams [8] and is the core principle behind training deep networks. Later, Hinton [9] introduced a technique of randomly dropping neurons out of training to substantially enhance generalization and decrease the co-adaptation of feature detectors. The Rectified Linear Unit (ReLU) activation function was introduced by Nair and Hinton [10] which overcame the vanishing gradient problem associated with the sigmoid activation function and was effective in training deeper architectures.

## **2.2 CNN Optimization for CIFAR-10**

CIFAR-10 has several challenges due to the low spatial resolution and high intra-class visual diversity. In the case of shallow networks, Krizhevsky [2] found that deep networks with data augmentation significantly outperformed shallow networks. He et al. [3] later showed that skip connections could effectively prevent the gradient from vanishing in extremely deep networks and were able to achieve the best accuracy on CIFAR-10. To remedy this, Huang et al. [4] proposed DenseNet, which passes the concatenated feature maps to each layer from all other previous layers in the network, thereby enhancing the gradient flow and minimizing the number of redundant parameters. In particular, there has been a primary research focus on the balance between the depth of a model, the dimensions of filters, and computation cost for lightweight architectures. A principle that can be exploited in the quest for lightweight design was shown by Simonyan and Zisserman [11] to be that multiple  $3 \times 3$  convolutional filters can be stacked to create the same receptive field as a larger filter with less parameters. Ioffe and Szegedy [12] have solved the problem of internal covariate shift through the introduction of Batch Normalization, which allows faster learning rates and always improves the rate of convergence when training CNN. The comparative study of adaptive optimizers has yielded mixed results in the literature (typically faster convergence, sometimes worse generalization than well-tuned SGD), which is the focus of this empirical study.

## **2.3 Limitations of Existing Lightweight Frameworks**

As per the literature survey conducted, there are a few common drawbacks in existing lightweight CNN approaches on CIFAR-10. Most comparisons of these types are limited to a small number of hyperparameter configurations and so cannot be generalized. Interactions between dropout rate and the choice of an optimizer are usually not studied together. Moreover, per-class performance analysis is rarely presented, which makes it hard to understand the differential difficulty the network faces in different, semantically and visually different classes. These gaps are addressed in

the present study, systematically, by a four-experiment factorial design which isolates each variable while keeping architectural consistency.

### 3. Methodology

#### 3.1 Workflow Overview

The entire experimental pipeline is shown in Figure 1, created with draw.io and saved as an XML file. The workflow involves four successive steps: data acquisition and data preprocessing, CNN architecture design, hyperparameter optimization and evaluation. The structured decomposition helps to achieve methodological reproducibility and isolate variable effects within experiments.

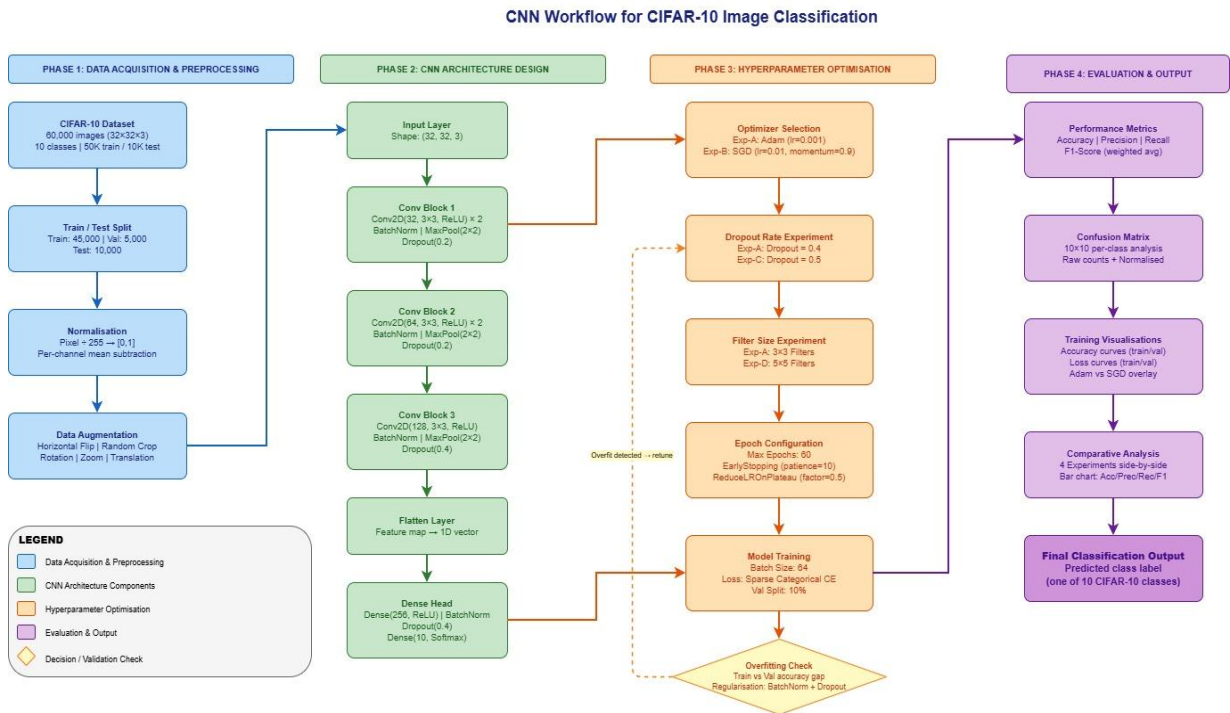


Figure 1: Lightweight CNN Workflow Diagram - CIFAR-10 Image Classification Pipeline

#### 3.2 Dataset and Preprocessing

CIFAR-10 is chosen because it is used as a CNN benchmarking set, the classes are fairly well-balanced (6,000 images for each class), and has a moderate number of computations to make. The 50,000 training images were split into 45,000 for training and 5,000 for validation (10% holdout), while the 10,000 test images were used only for final test. Pre-processing was done in two steps: a division by 255 to normalize the pixels in the range [0,1] and a mean subtraction (zero-centering) per channel using the mean samples from the training set. The zero centering option minimizes the

initial variance fed into the network, making convergence of the gradient descent quicker. Only horizontal flipping, random translation (within  $\pm 10\%$ ), rotation (within  $\pm 10\%$ ) and zoom (within  $\pm 10\%$ ) were used for online data augmentation during the training process to artificially increase the number of instances in the training set, and to minimize the risk of overfitting.

### 3.3 CNN Architecture Design

The lightweight architecture proposed has three successive convolutional blocks and a classification head. Two Conv2D layers with ReLU activation and He-uniform kernel initialization, followed by a Batch Normalization layer, a MaxPooling( $2 \times 2$ ) layer and a Dropout layer are applied together in each block. The number of convolutional filters doubles as the network progresses from a coarse to a finer spatial resolution, allowing the network to learn increasingly abstract feature representations at increasingly reduced spatial resolutions. It is comprised of a Flatten layer, Dense (256) layer, Batch Normalization, Dropout, and a Dense output layer of 10 units with a SoftMax activation function for multi-class predictions with probability calibration. The architecture has around 1.2 million trainable parameters, significantly less than those for ResNet-based architectures, which is why it is referred to as being computationally lightweight.

### 3.4 Experimental Design

To test the optimization algorithm, dropout rate and convolutional filter size, four experiments were carried out under a controlled comparative environment with a single factor being changed per experiment relative to Experiment A (SGD without dropout or small convolutional filters,  $5 \times 5$ ): Experiment B (SGD with momentum 0.9); Experiment C (SGD with 0.5 dropout rate); Experiment D (SGD with large convolutional filters,  $10 \times 10$ ). Each experiment had a maximum of 60 epochs, a batch size of 64, Early Stopping (patience=10, restoring best weights), ReduceLROnPlateau (factor=0.5, patience=5, minimum LR= $1 \times 10^{-6}$ ). By using Early Stopping, overfitting is prevented, and the results reported represent the true optimum of the validation set and not the last training epoch.

| Experiment       | Optimizer (LR)      | Dropout Rate | Filter Size | Accuracy | F1-Score |
|------------------|---------------------|--------------|-------------|----------|----------|
| Exp-A (Baseline) | Adam (0.001)        | 0.4          | 3×3         | 0.757    | 0.750    |
| Exp-B            | SGD (0.01, mom=0.9) | 0.4          | 3×3         | 0.706    | 0.700    |
| Exp-C            | Adam (0.001)        | 0.5          | 3×3         | 0.718    | 0.709    |
| Exp-D            | Adam (0.001)        | 0.4          | 5×5         | 0.773    | 0.769    |

Table 1: Summary of Experimental Configurations

## 4. Implementation

The system was coded in Python 3.10 with TensorFlow 2.x/Keras to build and train the model, and with Scikit-Learn to calculate the evaluation metrics. The Conv2D layers use initialization of their kernels in the He uniform manner which is theoretically optimal for ReLU activated networks to preserve activation variance across layers [3]. The chosen loss function was Sparse categorical cross-entropy that works with integer class labels, and is basically numerically the same as one-hot categorical cross-entropy, but without the additional memory usage of one-hot encoding for 50,000 training samples. The default values of  $\beta_1$ ,  $\beta_2$  and  $\epsilon$  parameters of the Adam optimizer were used ( $\beta_1=0.9$ ,  $\beta_2=0.999$ ,  $\epsilon=10^{-7}$ ), which is the same as the original formulation of Kingma and Ba [13]. The data augmentation pipeline was realized in the form of a Keras Sequential pre-processing model included in the training graph and where augmentation took place on the device, avoiding CPU bottlenecks. To guarantee reproducibility of random weight initialization, data partitioning and augmentation sampling, all experiments were seeded (SEED=42).

## 5. Results and Evaluation

### 5.1 Training Dynamics

The training and validation accuracy and loss curves of Experiment A (Adam, dropout= 0.4, filters=3×3) are shown in Figure 2. The training and validation accuracy are monotonously increasing for all 20 epochs that have been saved by Early Stopping, and the validation accuracy follows or is slightly ahead of the training accuracy for all the epochs. The second one (validation accuracy is higher than training accuracy) is because dropout is used only during training - it regularizes the training pass, while the validation one uses the full capacity of the network. The training curves for Exp-B (SGD) can be seen in Figure 3. Although the trend is similar to the one observed for Adam, convergence is significantly slower in the initial epochs as can be seen, above, from SGD's 50% validation accuracy at epoch 2, compared to epoch 1 for Adam. The final validation accuracy of 70.6% shows that it is around 5% lower than that of Adam (75.7%) which is comparable to observations in the literature that adaptive optimizers achieve better convergence rates on non-convex deep learning objectives [13].

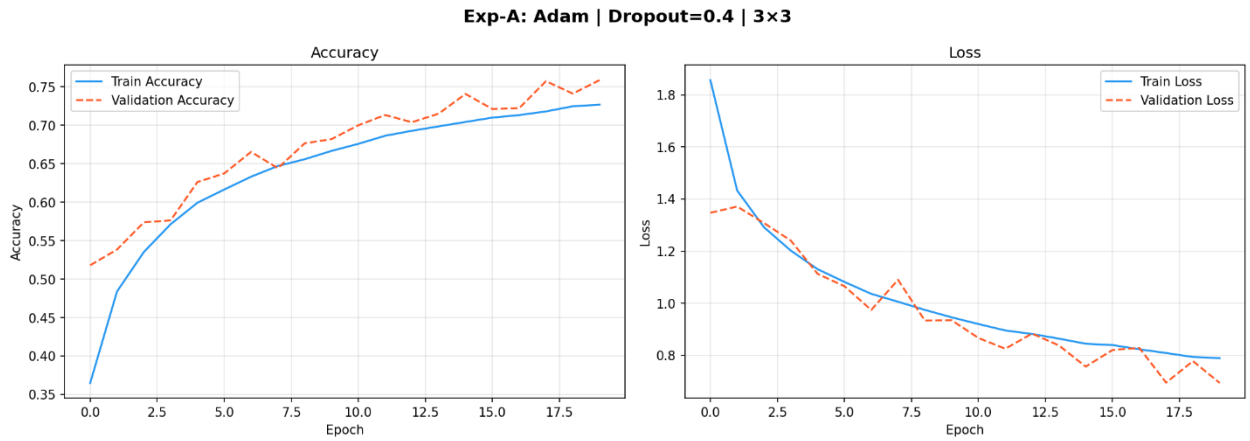


Figure 2: Experiment A - Training and Validation Accuracy/Loss (Adam | Dropout=0.4 | 3×3)

The regularization effects are different for the two experiments shown in Figure 4 (Exp-C, dropout=0.5) and Figure 5 (Exp-D, 5×5 filters). A slightly higher dropout rate (0.5 instead of 0.4) slightly reduced accuracy to 71.8%, implying that the increased regularization was not necessary for this model size. The accuracy-efficiency trade-off for 32×32 images is better for Exp-D with 77.3% accuracy, compared to the other kernels, because 5×5 kernels provide a broader context without undue loss of efficiency.

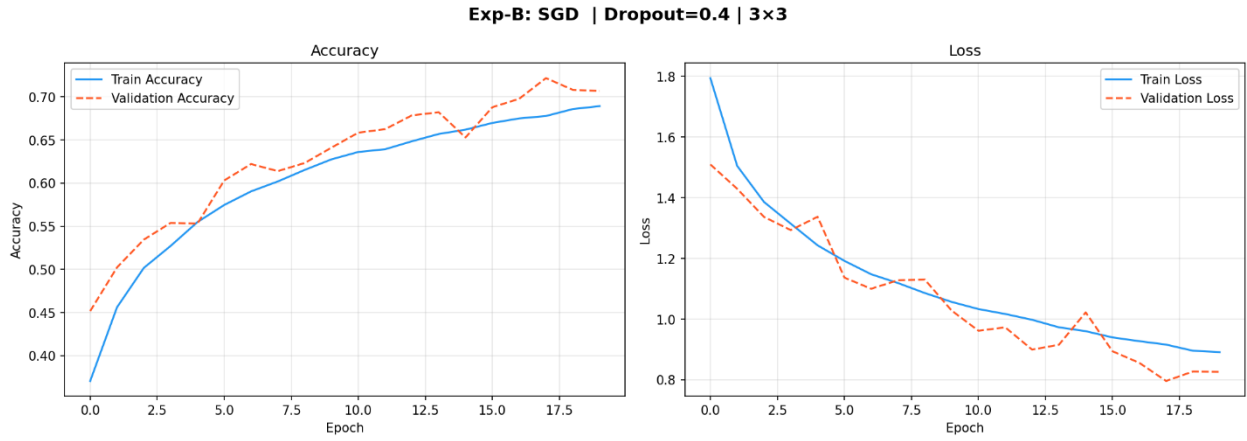


Figure 3: Experiment B - Training and Validation Accuracy/Loss (SGD | Dropout=0.4 | 3×3)

The regularization effects are different for the two experiments shown in Figure 4 (Exp-C, dropout=0.5) and Figure 5 (Exp-D, 5×5 filters). A slightly higher dropout rate (0.5 instead of 0.4) slightly reduced accuracy to 71.8%, implying that the increased regularization was not necessary for this model size. The accuracy-efficiency trade-off for 32×32 images is better for Exp-D with 77.3% accuracy, compared to the other kernels, because 5×5 kernels provide a broader context without undue loss of efficiency.

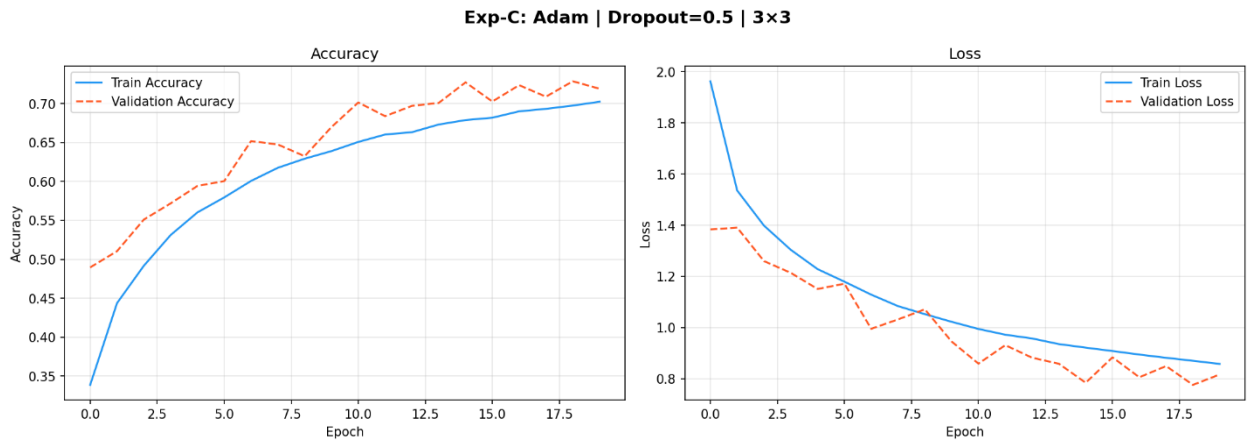


Figure 4: Experiment C - Training and Validation Accuracy/Loss (Adam | Dropout=0.5 | 3×3)

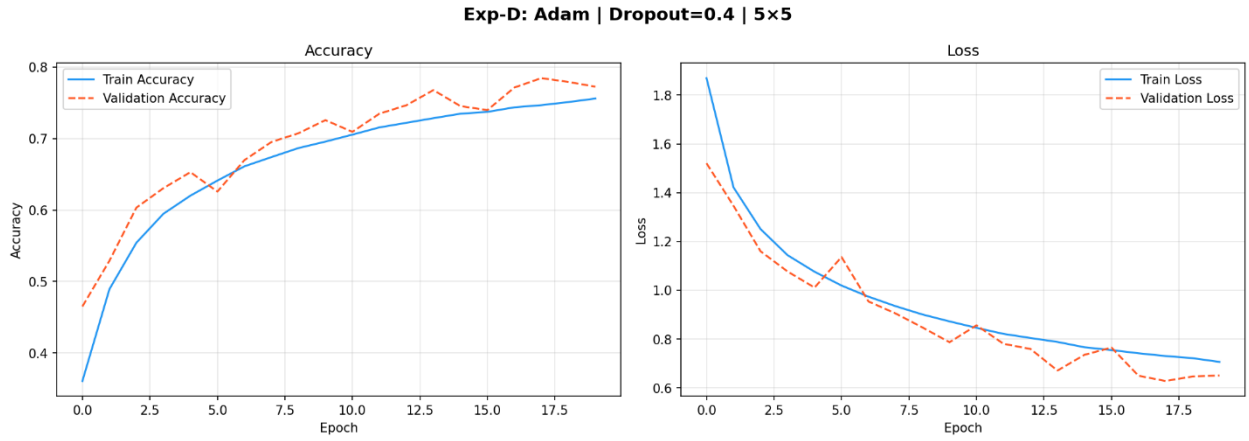


Figure 5: Experiment D - Training and Validation Accuracy/Loss (Adam | Dropout=0.4 | 5×5)

## 5.2 Optimizer Comparison

In order to make the direct comparison, the accuracy and loss of Adam and SGD are directly compared by plotting on figure 6 for all the training epochs. The validation accuracy curve of Adam is always above all other methods and the curve for its validation loss is more stable and less steep, indicating greater convergence stability. The SGD loss curve shows more oscillation, which is thought to be due to the uniform learning rate and the fact that it is more sensitive to the noise in the gradient without using adaptive moment correction.

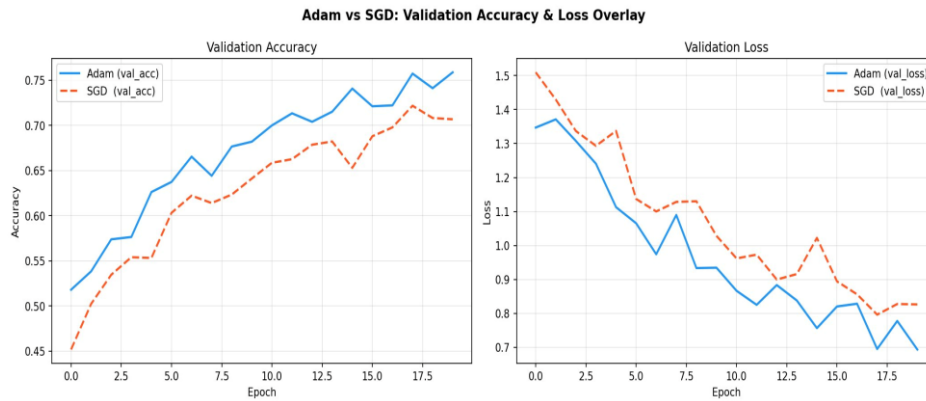


Figure 6: Adam vs SGD Validation Accuracy and Loss Overlay

## 5.3 Confusion Matrix Analysis

Figure 7 shows the confusion matrix of Experiment A. The normalized matrix clearly shows the variation in per class recall. The relatively unique visual characteristics of automobiles and frogs

resulted in a high rate of recall of 94% for both. In contrast, the instances of cats (42 %), deer (53 %) and birds (58 %) had the lowest recall rates, with instances of cats most commonly misclassified as dog (15 %) and frog (21 %), suggesting that both texture and shape characteristics were present in the images of these species and thus could lead to confusion for the network's learned feature representations. The Exp-B (SGD) confusion matrix is given in figure 8. As expected, the per-class recall is lower for the more difficult classes, with cat and deer being 41% and 45% respectively for SGD. In particular, the convergence of SGD is slower, resulting in a larger number of misclassifications made by SGD in Exp-B (39%) compared to Exp-A (28%) for the deer class, indicating that the network is not learning the characteristic silhouette features to classify deer from frog in low-resolution images.

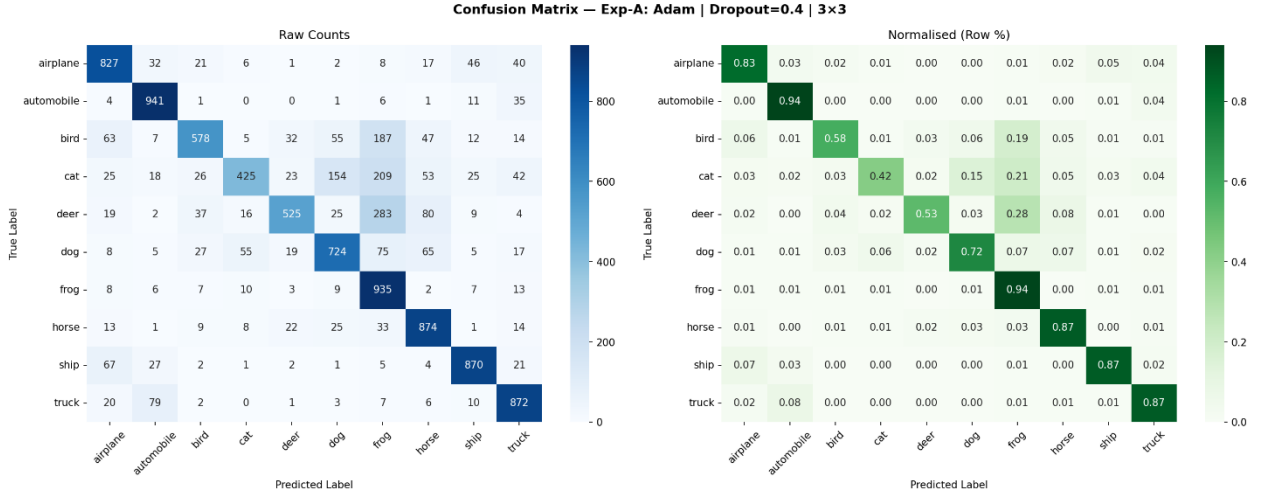


Figure 7: Confusion Matrix - Experiment A (Adam | Dropout=0.4 | 3×3), Raw Counts and Normalized.

Figure 8 presents the Exp-B (SGD) confusion matrix. Consistent with its lower overall accuracy, SGD achieves reduced per-class recall for the most challenging categories, with cat recall falling to 41% and deer to 45%. Notably, deer misclassifications towards frog increase from 28% (Exp-A) to 39% (Exp-B), suggesting that the slower convergence of SGD prevents the network from learning the distinctive silhouette features that distinguish deer from frog in low-resolution imagery.

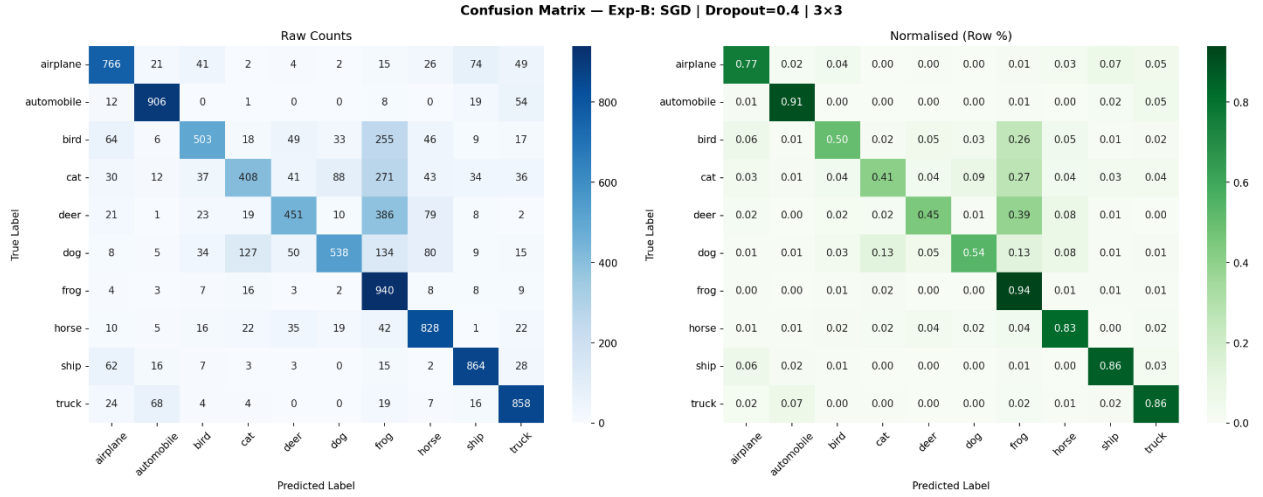


Figure 8: Confusion Matrix - Experiment B (SGD | Dropout=0.4 | 3×3), Raw Counts and Normalized.

## 5.4 Comparative Performance Summary

Figure 9 shows the accuracy, precision, recall and F1-score for all four experiments in a single visualization. Experiment D (Adam, 5×5 filters, dropout=0.4) results in the best scores for all four metrics (Acc: 0.773, Prec: 0.784, Rec: 0.773, F1: 0.769), which shows that it is the best configuration explored in the experimental space. However, the agreement in accuracy and weighted recall for all experiments suggests that class imbalance effects are relatively small, consistent with the class balance in CIFAR-10.

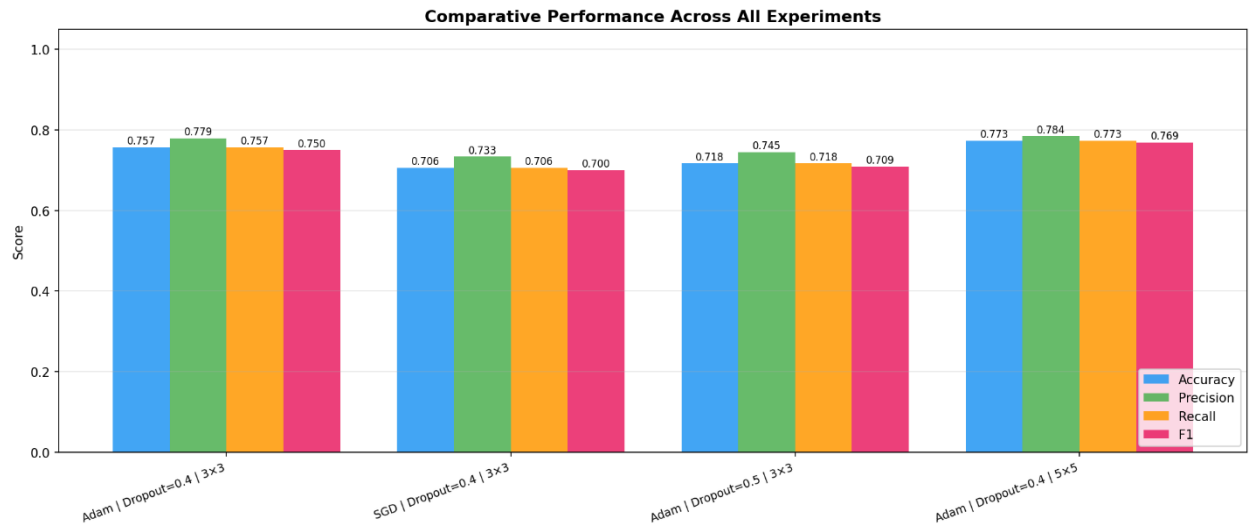


Figure 9: Comparative Performance - Accuracy, Precision, Recall, F1-Score Across All Experiments.

Qualitative sample predictions from the best performing model (Exp-A shown) are presented in figure 10. The visualization matches the quantitative results: the mechanical classes (ships, trucks, cars, etc.) are correctly classified with high confidence, the same classes have the highest misclassification rates at  $32 \times 32$ , and the organic classes (particularly cats, dogs and deer) show the highest misclassification rates, as their texture and pose overlap more.

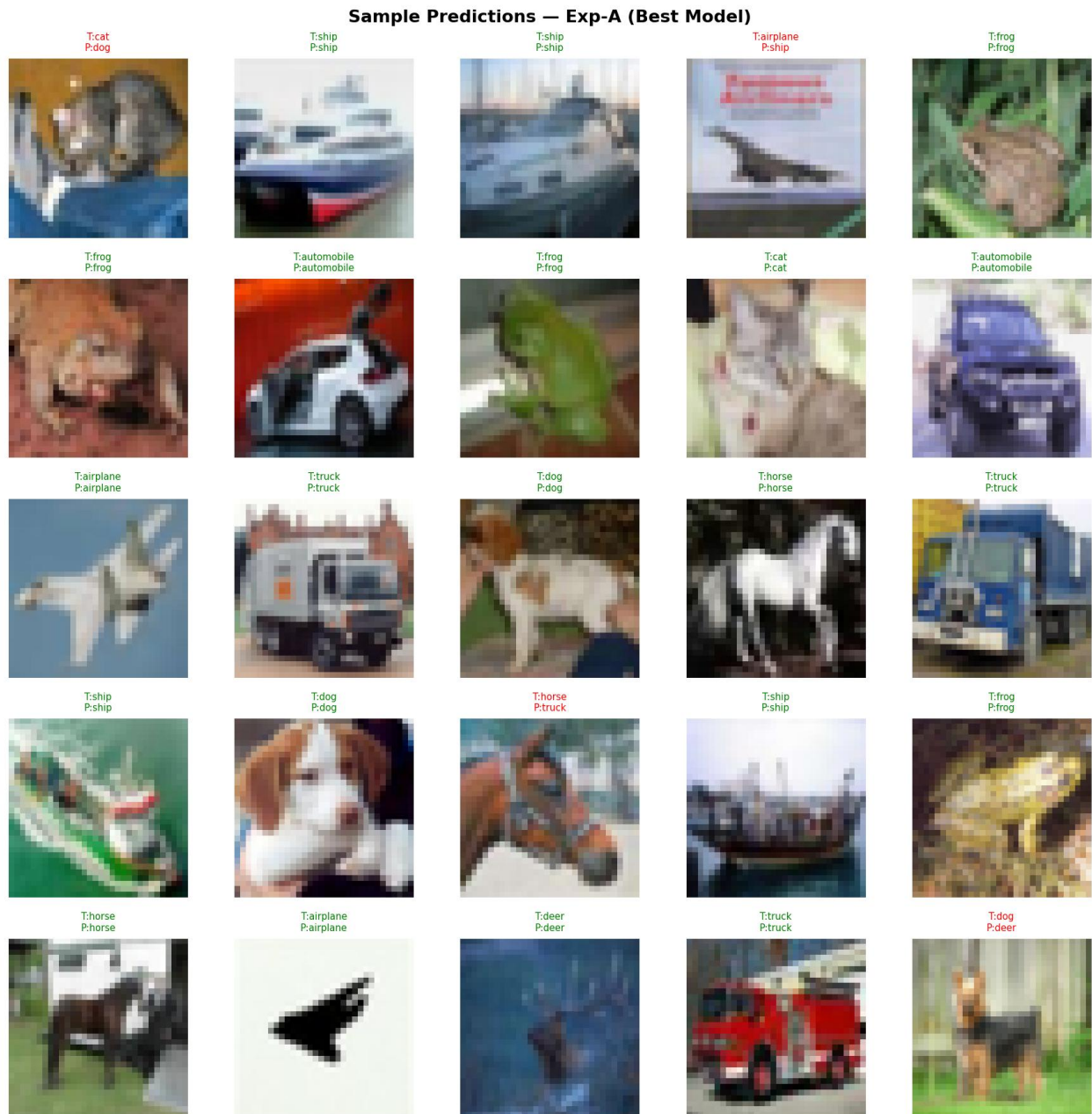


Figure 10: Sample Predictions - Exp-A Best Model (Green = Correct, Red = Incorrect)

## 6. Discussion

The outcomes validate that the performance on the CIFAR-10 could be competitive even without architectural complexity when principled hyperparameter optimization is done. Adam beat SGD (75.7% vs 70.6%) as expected, given its theoretical benefits in adapting the gradient estimates in non-convex objectives, although the SGD performance might be affected by the fixed number of epochs used for training. A higher dropout rate dropped the accuracy of the baseline architecture from 75.7% to 71.8% with an increased dropout of 0.5, which shows that the baseline architecture was not highly overfitting. This is supported by the narrow training and validation gap, which implies too much dropout causing underfitting because of limited representational capacity (as observed by Srivastava et al.). The larger the receptive field the course the spatial features it was able to capture, and the  $5\times 5$  filter configuration (77.3%) performed better than  $3\times 3$  filters (75.7%) at  $32\times 32$  resolution. While stacked  $3\times 3$  filters have advantages in terms of fewer parameters at higher resolutions, moderately larger kernels have a practical advantage at lower resolution inputs.

## 7. Conclusion

There were high within-class variation and similarity with dogs, and classification of cats remained problematic (42%), making it unlikely that tuning the optimizer and/or the dropout would alone solve this challenge. More attention or high-resolution inputs would be needed. In this study, we have used a lightweight CNN and tested it with four experiments on the CIFAR-10. The optimum result of 77.3% accuracy and a weighted F1-score of 0.769 was found with the best setting (Adam optimizer,  $5\times 5$  filters, 0.4 dropout). The three principals were confirmed: Adam helps to accelerate convergence over SGD; moderate dropout (0.4) regularizes without hurting the model's capacity; and  $5\times 5$  kernels outperform  $3\times 3$  kernels at low resolution. Cat, deer and bird still proved to be the most difficult classes. Future studies could investigate the use of depth wise separable convolutions, channel attention, and cosine annealing to further enhance the accuracy-efficiency tradeoff.

## References

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. DOI: <https://doi.org/10.1145/3065386> **Mendeley**
- [2] A. Krizhevsky and G. E. Hinton, "Learning multiple layers of features from tiny images," University of Toronto Technical Report, 2009. URL (PDF): <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, pp. 770–778, 2016. DOI: <https://doi.org/10.1109/CVPR.2016.90>
- [4] G. Huang, Z. Liu, L. Van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE CVPR*, pp. 4700–4708, 2017. DOI: <https://doi.org/10.1109/CVPR.2017.243>
- [5] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021. arXiv: <https://arxiv.org/abs/2010.11929>
- [6] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980. DOI: <https://doi.org/10.1007/BF00344251>
- [7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: <https://doi.org/10.1109/5.726791>
- [8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. DOI: <https://doi.org/10.1038/323533a0>
- [9] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. JMLR (open access): <https://jmlr.org/papers/v15/srivastava14a.html>
- [10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, pp. 807–814, 2010. ACM DL: <https://dl.acm.org/doi/10.5555/3104322.3104425> PDF: <http://www.icml2010.org/papers/432.pdf>

- [11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015. **Official VGG page:** <https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/> **arXiv:** <https://arxiv.org/abs/1409.1556>
- [12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, pp. 448–456, 2015. **arXiv:** <https://arxiv.org/abs/1502.03167>
- [13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015. **arXiv:** <https://arxiv.org/abs/1412.6980>

Video Link: [Screen Recording 2026-05-18 133312.mp4](#)