University of Hertfordshire UH

Student ID

24080893

Student Name

Sanaan Ashfaq

Teacher Name

Peter Scicluna

Assignment

Machine Learning Tutorial

Github Link: https://github.com/SanaanAshfaq/cnn_depth_filter_cifar10

Video Link: https://drive.google.com/file/d/1f_-RBEhX6NhgMS_UGIibgss9b9fbdi-G/view?usp=sharing

Abstract

This tutorial introduces Convolutional Neural Networks (CNNs) and demonstrates how network depth affects image classification performance. A shallow CNN and a deeper CNN were designed and trained on the CIFAR-10 dataset under identical conditions. The shallow model achieved 63.57% accuracy, while the deep model reached 75.42%, showing the advantage of deeper architectures in learning hierarchical image features. The tutorial combines explanation, reproducible code, and visual analysis to help learners understand how CNNs work in practice.

Table of Content

1. Introduction:

Deep learning models, especially Convolutional Neural Networks, have become the standard approach for image classification because they can automatically learn useful features directly from raw pixels. Unlike traditional machine-learning methods that rely on manually crafted features, CNNs learn both low-level and high-level patterns during training.

One important design factor in CNNs is **network depth**. Shallow networks can learn simple features, while deeper networks can learn complex, hierarchical representations. This tutorial illustrates the effect of depth by training and comparing two CNN models on CIFAR-10.

2. Background: Convolutional Neural Networks:

A Convolutional Neural Network is a special type of neural network designed to exploit the spatial structure of image data. Instead of connecting every input pixel to every neuron (as in a fully connected layer), a convolutional layer uses **filters** (also called kernels) that slide over the image and compute local dot products. This sliding operation is known as a **convolution**.

Each filter learns to respond to a particular pattern in the image, such as a horizontal edge or a small texture. Early layers tend to learn low-level features, while deeper layers combine those low-level features into higher-level concepts. This idea is known as **hierarchical feature learning**.

Typical building blocks of a CNN include:

- **Convolutional layers**: apply trainable filters to produce feature maps.
- **Activation functions** (e.g., ReLU): introduce non-linearity so the network can model complex relationships.
- **Pooling layers** (e.g., max pooling): downsample feature maps, making the representation more compact and somewhat translation-invariant.
- **Fully connected layers**: operate on the flattened feature representation to perform classification.
- **Softmax output layer**: converts the final scores into probabilities across the target classes.

Deeper CNNs usually contain more convolutional layers and filters, allowing them to represent more complex visual patterns.

3. Dataset Description: CIFAR-10:

The CIFAR-10 dataset is a standard benchmark for image classification. It contains 60,000 colour images of size 32 × 32 pixels, divided into ten classes:

- Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.

The dataset is split into:

- 50,000 training images
- 10,000 test images

Each image is relatively small and low-resolution, which makes the classification task non-trivial. Many classes can look similar at this resolution (for example, cats and dogs), making it a good test case for measuring the effect of model capacity and depth.

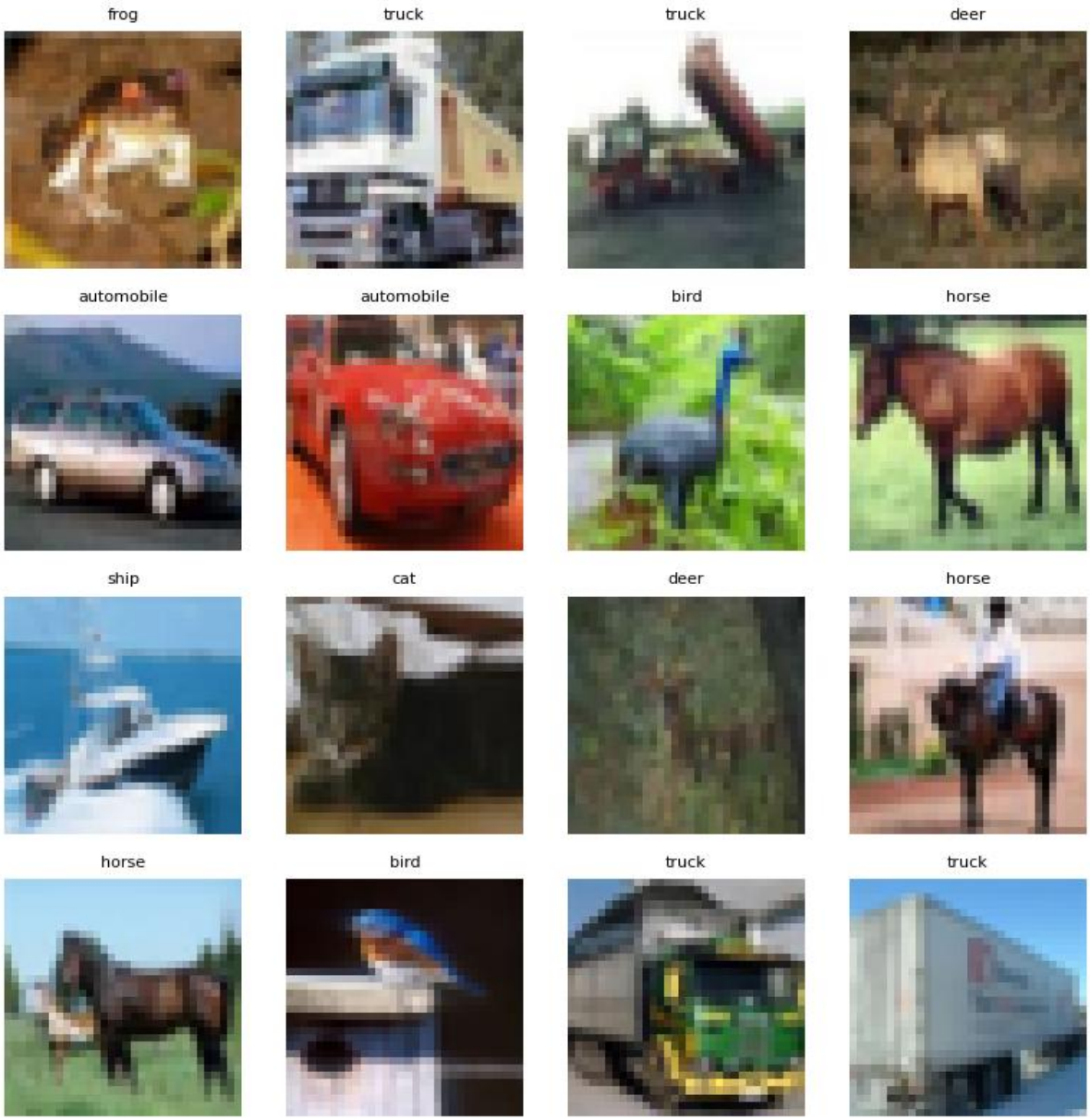

Figure 1: Sample images from the CIFAR-10 training dataset with their class labels.

- ❖ Before training, pixel values were scaled from the integer range [0, 255] to floating-point values in the range [0, 1]. This normalisation step helps the optimiser converge more smoothly.

4. Methodology

4.1 Implementation Environment:

The models were implemented in Python using TensorFlow and Keras within a Jupyter notebook. The entire workflow, including data loading, model definition, training, and evaluation, is contained in a single notebook so that it can be re-run from top to bottom. The same random initialisation strategy was used for both models to ensure a fair comparison.

4.2 Data Preprocessing:

The CIFAR-10 dataset was loaded using keras.datasets.cifar10.load_data(). The images were converted to float32 and divided by 255.0 to normalise pixel values. Labels were kept as integer class indices and used with the sparse_categorical_crossentropy loss, which is well-suited for multi-class classification with integer labels.

4.3 Shallow CNN Architecture:

The **shallow CNN** was designed as a simple baseline network:

- Input: 32 × 32 × 3 RGB image
- One convolutional layer with 32 filters of size 3 × 3, ReLU activation, and "same" padding
- One 2 × 2 max-pooling layer
- A flattening layer
- A fully connected (Dense) layer with 64 units and ReLU activation
- An output layer with 10 units and softmax activation

This architecture contains 525,898 trainable parameters. It is intentionally kept small to represent a low-capacity model that can only learn relatively simple feature representations.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 32)	0
flatten_3 (Flatten)	(None, 8192)	0
dense_6 (Dense)	(None, 64)	524,352
dense_7 (Dense)	(None, 10)	650

Total params: 525,898 (2.01 MB)

Trainable params: 525,898 (2.01 MB)

Non-trainable params: 0 (0.00 B)

Figure 2: Architecture summary of the shallow CNN model.

- ❖ The shallow model was compiled with the Adam optimiser, sparse_categorical_crossentropy loss, and accuracy as the main evaluation metric.

4.4 Deep CNN Architecture:

The **deep CNN** extends the shallow model by adding more convolutional layers and increasing the number of filters:

- Input: 32 × 32 × 3
- Block 1: two 3 × 3 convolutional layers with 32 filters, followed by max pooling
- Block 2: two 3 × 3 convolutional layers with 64 filters, followed by max pooling
- Block 3: one 3 × 3 convolutional layer with 128 filters, followed by max pooling
- Flattening layer
- Dense layer with 128 units and ReLU activation
- Dropout layer with a rate of 0.5 to reduce overfitting
- Output softmax layer with 10 units

Compared to the shallow CNN, this network has a much higher number of parameters and a deeper feature hierarchy. It is expected to learn more complex visual patterns.

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 32, 32, 32)	896
conv2d_9 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d_6 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_10 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_11 (Conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_12 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_4 (Flatten)	(None, 2048)	0
dense_8 (Dense)	(None, 128)	262,272
dropout_1 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 10)	1,290

Total params: 402,986 (1.54 MB)

Trainable params: 402,986 (1.54 MB)

Non-trainable params: 0 (0.00 B)

Figure 3: Architecture summary of the deep CNN model.

- ❖ The deep model was also compiled with Adam and the same loss function and metric, ensuring that depth is the main varying factor.

4.5 Training Procedure:

Both models were trained using the same settings:

- Epochs: 10
- Batch size: 64
- Validation split: 20% of the training data reserved for validation
- Optimiser: Adam with default learning rate
- Loss: Sparse categorical cross-entropy

- Metric: Accuracy

Using identical training parameters ensures that differences in performance can be attributed primarily to differences in model depth rather than training configuration.

5. Results

5.1 Shallow CNN Training Behaviour:

The shallow CNN reached a final test accuracy of **63.57%** with a test loss of **1.0826**. The training and validation curves show steady improvement, but validation accuracy plateaus relatively early.

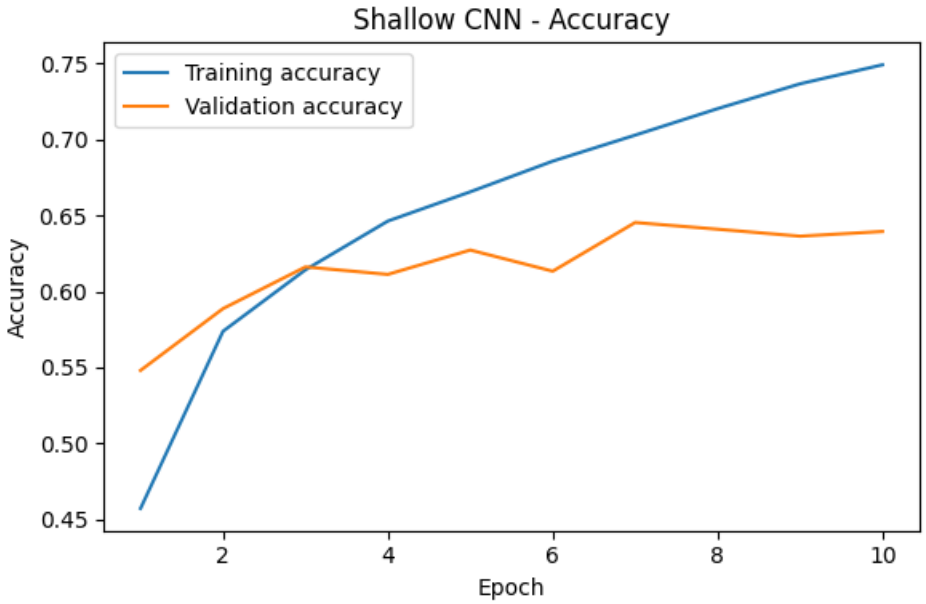

Figure 3: Training and validation accuracy of the shallow CNN.

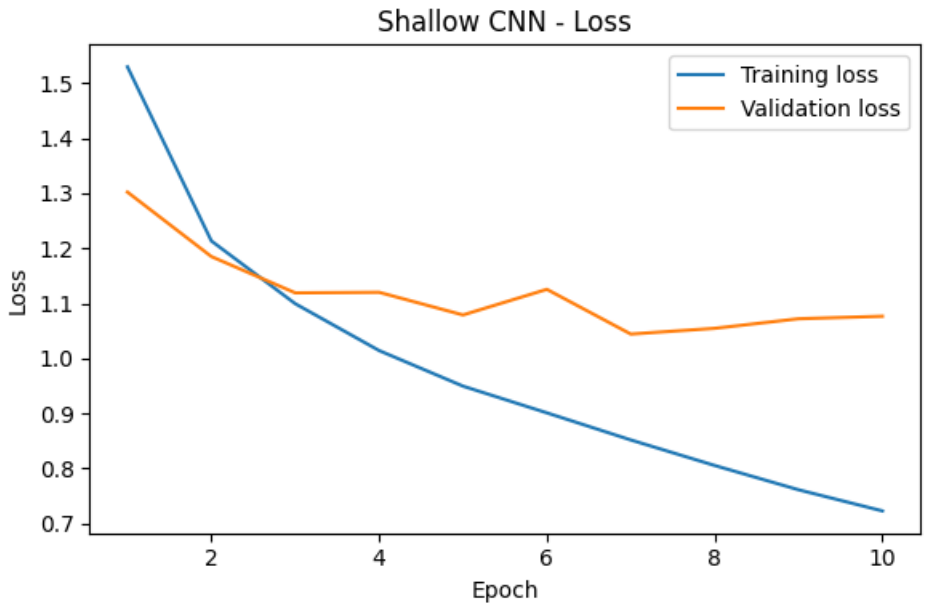

Figure 4: Training and validation loss of the shallow CNN.

- These curves suggest that while the shallow model is able to learn meaningful features, its limited capacity prevents it from fully capturing the complexity of the CIFAR-10 images.

5.2 Deep CNN Training Behaviour:

The deep CNN achieved a significantly higher test accuracy of **75.42%** with a test loss of **0.7568** after the same number of epochs.

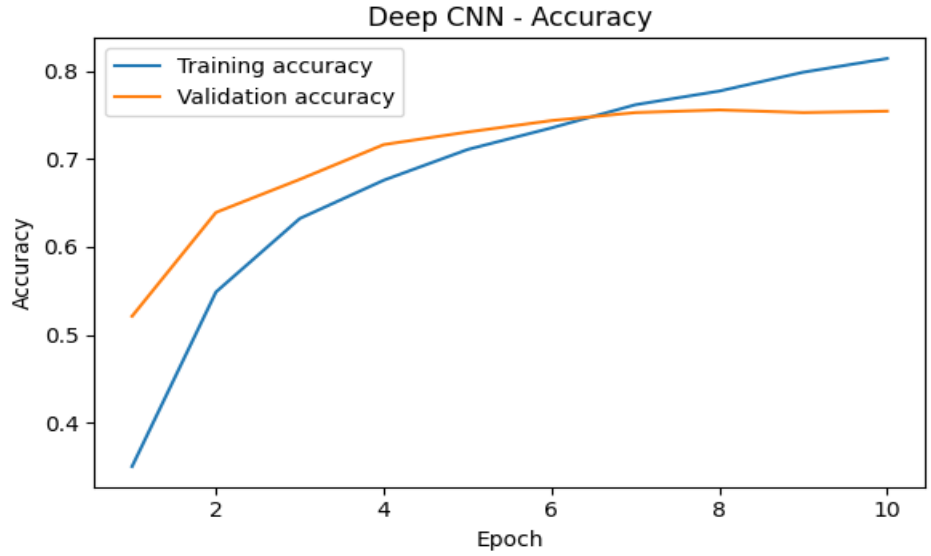

Figure 5: Training and validation accuracy of the deep CNN.

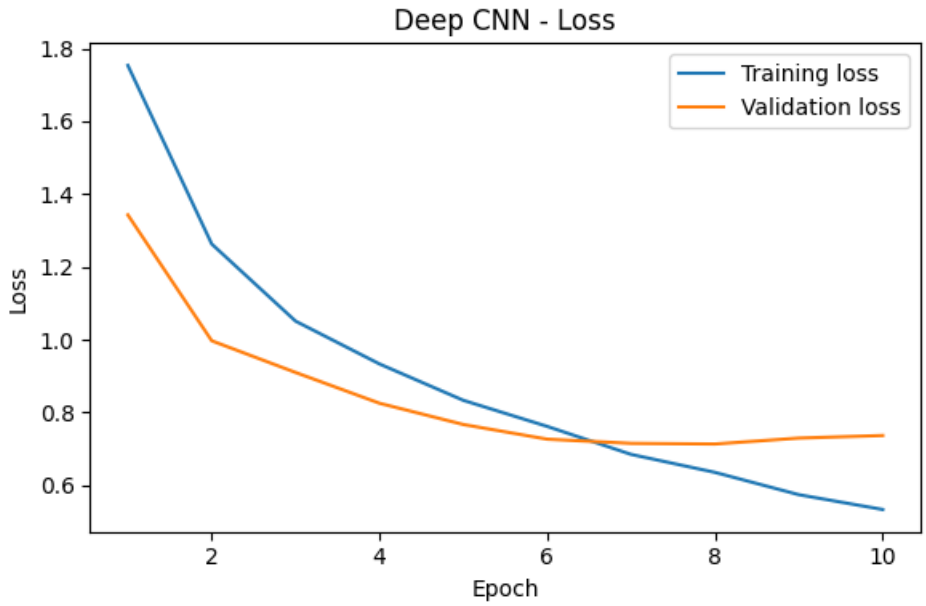

Figure 6: Training and validation loss of the deep CNN..

- The deep model's validation accuracy curve continues to rise for longer and reaches a much higher level than the shallow model. The validation loss also converges to a lower value, indicating better generalisation to unseen data.

5.3 Direct Comparison of Validation Curves:

To visualise the difference more clearly, the validation accuracies and losses of the two models were plotted together.

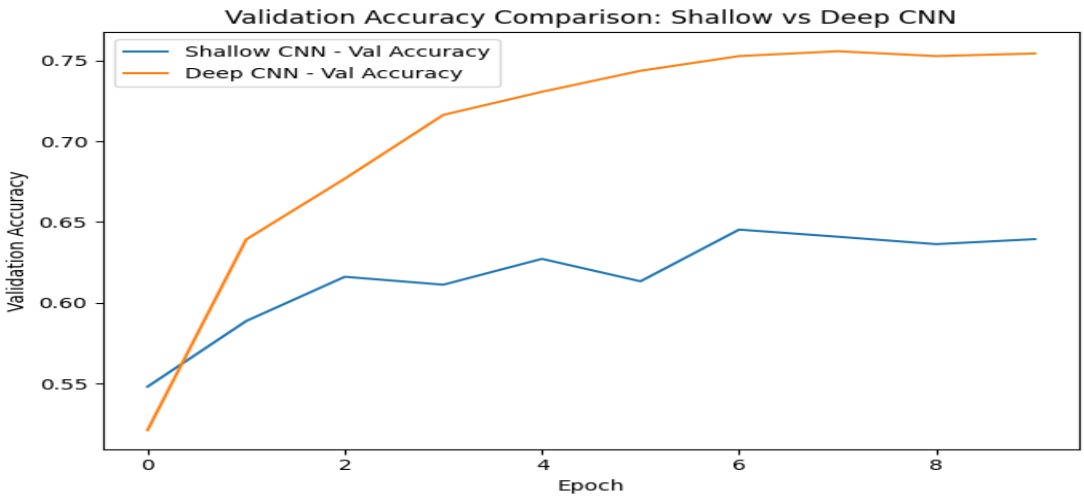

Figure 7: Validation accuracy comparison between shallow and deep CNN models.

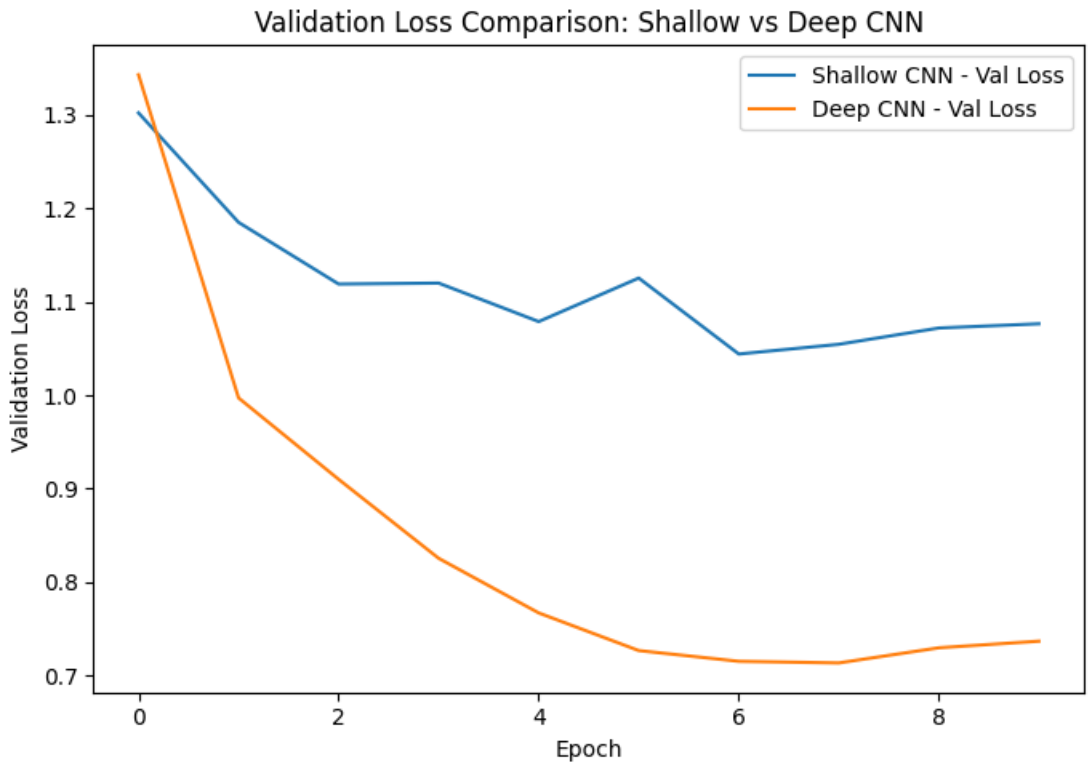

Figure 8: Validation loss comparison between shallow and deep CNN models.

- The comparison plots show that the deep CNN consistently outperforms the shallow CNN at almost every epoch. This supports the intuition that deeper networks, when trained correctly, can learn more informative representations.

5.4 Confusion Matrix for the Deep CNN:

A confusion matrix was generated for the deep CNN to analyse performance per class.

The matrix shows high values along the diagonal, indicating correct predictions for most classes. Some off-diagonal entries reveal confusion between similar categories, such as cats and dogs, or airplanes and ships, which are visually similar at low resolution.

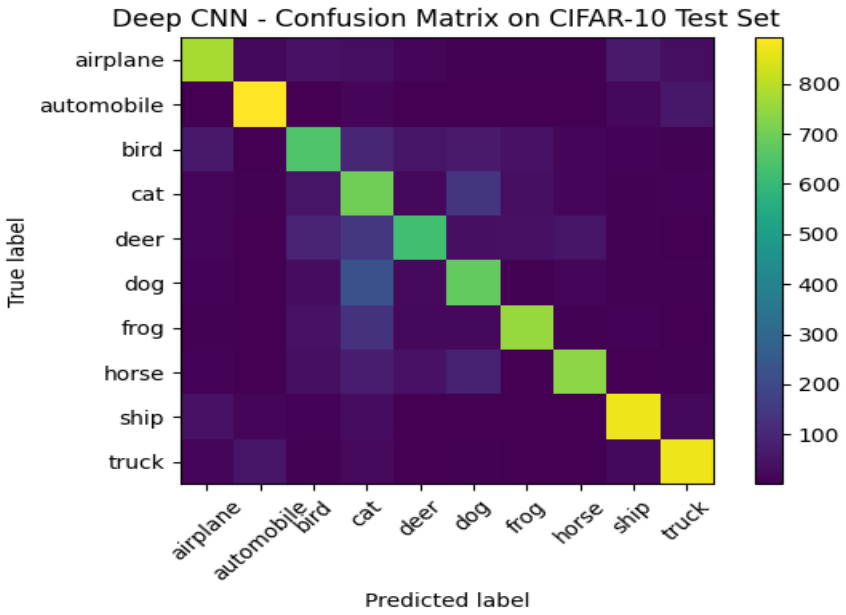

Figure 9: Confusion matrix of the deep CNN on the CIFAR-10 test set.

- The matrix shows high values along the diagonal, indicating correct predictions for most classes. Some off-diagonal entries reveal confusion between similar categories, such as cats and dogs, or airplanes and ships, which are visually similar at low resolution.

5.5 Correct and Incorrect Predictions:

To make the results more interpretable, several test images were visualised with their true and predicted labels.

Figure 10: Examples of correct and incorrect predictions made by the deep CNN on CIFAR-10 test images.

- ❖ Correct predictions are shown in green, while incorrect ones are shown in red. This qualitative inspection confirms that the deep CNN captures many object categories well but still struggles with ambiguous or noisy images.

6. Discussion:

The results show a clear performance difference between the two models, demonstrating the importance of network depth. The shallow CNN learns basic features but lacks the capacity to model complex patterns in CIFAR-10. Its validation accuracy stagnates early, indicating limited generalisation ability.

The deep CNN, with more convolutional layers and filters, forms richer hierarchical features. This results in higher accuracy and lower loss. The confusion matrix and sample predictions show strong performance, though natural ambiguity between some classes remains.

While deeper networks perform better, they also require more computation and may overfit if not regularised. Techniques such as dropout, data augmentation, and batch normalisation can further enhance performance.

7. Conclusion:

This tutorial presented a practical comparison between a shallow and a deep CNN on the CIFAR-10 dataset. Under identical training conditions, the shallow CNN achieved a test accuracy of 63.57%, while the deep CNN achieved 75.42%. The training curves, comparison plots, confusion matrix, and example predictions all support the conclusion that deeper CNNs are more effective at learning rich, hierarchical features and generalising to new images.

For learners, the key takeaway is not only how to implement CNNs in code, but also how to reason about architectural choices such as depth. The accompanying Jupyter notebook and GitHub repository provide a complete, reproducible pipeline that students can extend by experimenting with different hyperparameters, regularisation techniques, or alternative architectures.

8. Accessibility Considerations:

To improve accessibility:

- Plots were designed with clear axis labels and legible font sizes.
- Colour schemes with sufficient contrast were used to support colour-blind users.
- The written tutorial is compatible with screen readers through a logical heading structure.

- The accompanying video tutorial can be paired with a transcript and subtitles to support learners with hearing difficulties.

9. GitHub Repository:

All code, models, and instructions needed to reproduce the results in this tutorial are available at:

[https://github.com/SanaanAshfaq/cnn_depth_filter_cifar10**]**

The repository includes:

• The Jupyter notebook containing all experiments
• Saved CNN model files
• Training history files
• The full written report (PDF)
• The recorded video presentation
• The video transcript (PDF)
• A README file explaining how to run the code
• A license file specifying usage conditions

10. References:

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (Technical Report). University of Toronto.
https://www.cs.toronto.edu/~kriz/cifar.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
https://www.deeplearningbook.org

Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.
https://www.manning.com/books/deep-learning-with-python-second-edition

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
https://doi.org/10.1038/nature14539

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.
https://arxiv.org/abs/1409.1556

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.
https://arxiv.org/abs/1512.03385

Brownlee, J. (2019). *A gentle introduction to convolutional neural networks*. Machine Learning Mastery.
https://machinelearningmastery.com/convolutional-neural-networks-for-deep-learning/

TensorFlow Developers. (2023). *Image classification with convolutional neural networks*. TensorFlow Documentation.
https://www.tensorflow.org/tutorials/images/cnn

.END.