Assignment 2 COMP4530/5318

Group 73: 520510542, 510458096, 520131808, 550749464

October 2025

1 Introduction

This study aims to provide a comparative analysis of three machine learning algorithms: Convolutional Neural Networks (CNNs), Fully Connected Neural Networks, and Random Forests. These algorithms are trained and tested against the CIFAR-10 dataset. Which consists of 60,000 images across 10 object classes. We evaluate the performance and efficiency of different architectures when handling image classification tasks. The research systematically compares each model's ability to generalise and learn patterns hidden within images, with focuses on their performance efficiency and architectural strengths. The study provides insights into the practical trade-offs between model complexity and classification accuracy.

2 Data

2.1 Data description and exploration

The dataset we have been given for this project is the CIFAR-10 dataset. This dataset consists of 60000 32x32 pixels color images in 10 classes and each class contains 6000 images. The classes are airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships, and trucks, respectively. And the classes are completely mutually exclusive. [1]

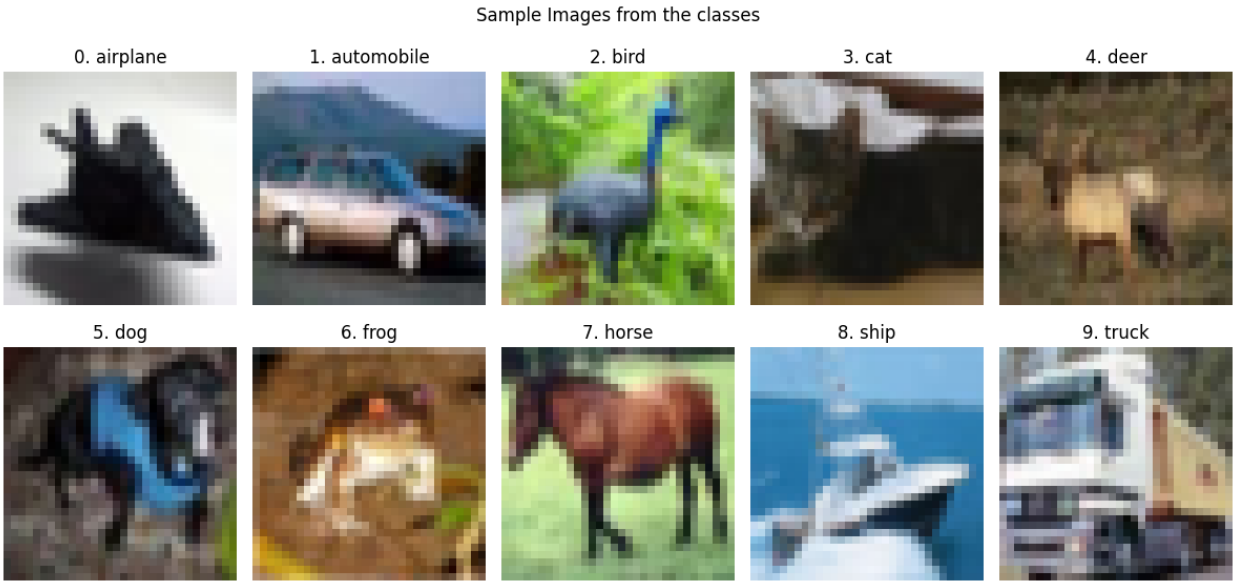

Figure 1: example images from each class

From the table below, we can see that the images are very consistently centered, with generally lower deviations in animal classes and higher deviations in vehicle classes.

Table 1: Center of Mass Analysis by Class

Class	Mean Y	Mean X	Std Y	Std X	Deviation
airplane	14.83	15.41	1.43	0.51	1.31
automobile	14.69	15.45	1.79	0.92	1.42
bird	15.29	15.47	1.36	0.96	0.89
cat	15.39	15.51	1.40	1.25	0.78
deer	15.85	15.57	1.84	0.85	0.46
dog	16.10	15.59	1.49	1.30	0.42
frog	15.70	15.63	1.44	0.87	0.48
horse	15.33	15.41	1.99	0.88	0.89
ship	14.10	15.43	1.55	0.90	1.98
truck	13.82	15.51	1.54	1.11	2.24

We can also inspect each image classes' variance of pixel intensity, mean brightness, and color diversity from the table below.

- **variance of pixel intensity:** The higher variance for trucks and automobiles suggests their images contain more pronounced textures, distinct shapes, and sharper edges. In contrast, the lower variance for birds and deer indicates their images are typically more uniform and smoother in appearance.
- **mean brightness:** Generally, images of airplanes and ships are the brightest, while those of deer and frogs are the dimmest. This difference could stem from the brightness of the subjects themselves or their typical backgrounds.
- **color diversity:** The color diversity metric varies significantly across the classes. images of deer and frog have the highest values, while truck and dog have the lowest. This indicates that, within this dataset, images of deer and frogs contain a wider variety of colors on average, whereas images of trucks and dogs are characterized by a more limited color range.

Table 2: Image Statistics by Class

Class	Mean Variance	Mean Brightness	Color Diversity
airplane	2953.46	146.47	13.73
automobile	3838.82	114.73	13.11
bird	2177.64	121.29	15.54
cat	3032.80	117.45	14.40
deer	2278.38	109.16	17.15
dog	3215.72	112.45	12.79
frog	2413.03	104.12	16.63
horse	3382.81	116.09	15.04
ship	3373.47	136.68	15.73
truck	4245.61	125.63	12.08

Further looking into the intensity of pixels, it reveal clear class characteristics. Images of airplanes are the brightest and have moderate contrast, while frogs are the dimmest. Trucks and automobiles have the highest intensity variation, suggesting strong internal contrast, whereas deer and birds display the most uniform intensity distributions.

Finally, we look at the average RGB value by class, which shows distinct color profiles for each class. Airplanes and ships have a strong blue intensity on average. In contrast, frogs and deer on average have a significantly lower blue intensity. Trucks and automobiles have the most balanced color values among the other classes.

Table 3: Image Intensity Statistics by Class

Class	Mean Intensity	Std Intensity
airplane	142.37	48.28
automobile	116.70	59.14
bird	119.41	42.74
cat	116.23	53.11
deer	111.78	41.64
dog	117.39	53.24
frog	106.56	43.71
horse	118.89	54.34
ship	133.48	53.19
truck	124.28	62.62

Table 4: Average RGB Values by Class

Class	Red	Green	Blue	Overall
airplane	134.04	142.88	150.17	142.37
automobile	120.15	115.91	114.04	116.70
bird	124.76	125.33	108.13	119.41
cat	126.35	116.39	105.96	116.23
deer	120.26	118.63	96.44	111.78
dog	127.48	118.48	106.22	117.39
frog	119.86	111.79	88.03	106.56
horse	128.00	122.37	106.31	118.89
ship	125.01	133.98	141.44	133.48
truck	127.16	123.76	121.91	124.28

2.2 Pre-processing

Normalization: The intensity values of every image are normalized from the range [0, 255] to [0, 1] and represented as a "float32" datatype This preprocessing step ensures numerical stability during training, enables faster convergence, and prevents single pixel values from disproportionately influencing the learning process. All 3 models in this project utilizes this pre-processing technique.

Flattening: This step is necessary for both random forest and multi-layer perceptron models. This operation flattens a 2D (32x32x3) image to a 1D (3072 pixels) vector. This is required for algorithms like MLPs and random forests because they are fundamentally designed to process data as simple feature vectors.

Splitting data training set and validation set: The validation set is not used to directly tune the model's parameters during training. Instead, its role is to evaluate the model's performance on unseen data. It serves as an warning system for overfitting. When the model's accuracy on the training set continues to increase while its accuracy on the validation set stagnates or declines, it indicates that the model is memorizing the training data rather than learning to generalize.

Data augmentation: Data augmentation is performed for CNN models to improve generalization and prevent overfitting. It includes:

- **Horizontal flipping:** This randomly mirrors the image horizontally.
- **Random rotation:** This rotates the image randomly by a small factor (±10%). Doing this adds robustness to rotational variations.
- **Random zoom:** This zooms the image randomly by a small factor (±10%). Doing this helps the model recognize objects at different scales.

3 Methods

3.1 Random Forest

For this classification task, a **Random Forest Classifier (RFC)** was selected over simpler models such as Nearest Neighbour, Rule-based models, Linear Regression, Logistic Regression, and Naïve Bayes. Random Forests are an *ensemble learning method* that combine multiple decision trees trained on random subsets of data and features. Each tree T_i outputs a class prediction $f_i(x)$, and the final output is determined by majority voting:

$$\hat{y} = \text{mode}\{f_1(x), f_2(x), \ldots, f_N(x)\}.$$

This aggregation reduces the variance of individual trees, making Random Forests more robust and less prone to overfitting. Unlike linear models, RFCs can model complex nonlinear decision boundaries, which are essential for distinguishing between visual classes in the CIFAR-10 dataset.

Although convolutional neural networks (CNNs) typically achieve higher accuracy on image data by exploiting spatial hierarchies, Random Forests were chosen as a classical, interpretable baseline to establish fundamental performance limits before moving to deep learning models.

Strengths and Weaknesses: The primary strength of Random Forests lies in their ability to capture nonlinear feature interactions and resist overfitting through ensemble averaging. They require minimal feature scaling and are relatively interpretable, offering feature importance estimates. However, since image pixels are weakly correlated across dimensions, Random Forests struggle to learn spatial dependencies as effectively as CNNs. Additionally, the model becomes computationally intensive with large datasets and many estimators.

Architecture and Hyperparameters: The Random Forest Classifier does not have a fixed network architecture like neural networks; instead, its structure is defined by the number, depth, and branching rules of individual decision trees. Each tree acts as a weak learner trained on a random subset of the data and features, and the ensemble aggregates their predictions via majority voting to improve generalization. The diversity among trees reduces variance and stabilizes performance.

Chosen Design: The baseline model was implemented with `n_estimators=100, max_depth=20, min_samples_split=10, min_samples_leaf=4.` These values were selected to balance training efficiency and overfitting control on the high-dimensional pixel data of CIFAR-10. The configuration achieved a training accuracy of 0.53 and a test accuracy of 0.42, indicating moderate overfitting but good generalization for a tree-based model.

Hyperparameters Tuned: A comprehensive search was performed over the following parameters:

$$\begin{aligned}
\texttt{n_estimators} &\in \{100, 200, 300\},\\
\texttt{max_depth} &\in \{10, 20, 30, \text{None}\},\\
\texttt{min_samples_split} &\in \{2, 5, 10\},\\
\texttt{min_samples_leaf} &\in \{1, 2, 4\},\\
\texttt{max_features} &\in \{\text{sqrt}, \text{log2}\}.
\end{aligned}$$

These hyperparameters were tuned through grid search to identify the optimal configuration for balancing bias–variance trade-offs, controlling model complexity, and improving generalization on unseen samples.

Effect of Each Hyperparameter:

- **n_estimators:** Controls the number of trees. Increasing it generally reduces variance and improves accuracy, but increases training time.
- **max_depth:** Sets the maximum depth of each tree. Deeper trees capture more complex patterns but can overfit.
- **min_samples_split:** Minimum samples required to split a node. Higher values restrict tree growth and improve generalization.
- **min_samples_leaf:** Minimum samples per leaf node. Larger values make trees smoother and prevent overfitting to noise.

- **max_features:** Number of features considered when splitting a node. Using `sqrt` introduces randomness, decorrelating trees and improving ensemble diversity.

Search Method: Hyperparameter tuning was conducted using `HalvingGridSearchCV`, an iterative resource-efficient method that allocates progressively more data to promising configurations while discarding less effective ones. This approach was chosen over exhaustive grid search due to the computational cost of training large ensembles on 20,000 CIFAR-10 samples. The search used 3-fold cross-validation and balanced accuracy as the metric to account for class variation.

In summary, the Random Forest Classifier provided a robust non-linear baseline for CIFAR-10 image classification, achieving around 42% test accuracy. While less effective than CNNs on spatial data, it demonstrated key theoretical advantages of ensemble learning: variance reduction, interpretability, and stable generalization. This experiment serves as an interpretable foundation for comparing classical and deep learning approaches in subsequent work.

3.2 Fully Connected Neural Network (FCN)

Theory and Rationale:

A **Fully Connected Neural Network (FCN)**—also referred to as a *Multi-Layer Perceptron (MLP)*—was implemented as a deep learning baseline for image classification on the CIFAR-10 dataset. The FCN consists of sequential layers where each neuron in a layer is connected to all neurons in the previous layer. Formally, each layer computes

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)}),$$

where $W^{(l)}$ and $b^{(l)}$ denote learnable weights and biases, and σ is a nonlinear activation function such as ReLU or Tanh. The network learns hierarchical feature transformations by minimizing the cross-entropy loss through stochastic gradient descent using the Adam optimizer.

This architecture was chosen as it represents a fundamental neural model covered in the early weeks of the course, serving as a bridge between classical machine learning algorithms (like Random Forests) and convolutional architectures. It enables direct comparison of the benefits of learned representations versus hand-engineered features.

Strengths and Weaknesses:

From a theoretical standpoint, FCNs are universal function approximators capable of modeling highly nonlinear relationships. Their main advantage lies in learning feature representations directly from raw input data without manual feature engineering. However, unlike Convolutional Neural Networks (CNNs), FCNs ignore spatial structure in images by flattening pixel inputs, losing local spatial correlations essential for image understanding.

This leads to weaker performance on image data and higher susceptibility to overfitting, particularly with limited data. The model's interpretability is low, as weight matrices lack direct semantic meaning. The computational cost and number of trainable parameters (**394,634**) are moderate, enabling manageable training times while maintaining expressive capacity. Regularization via dropout layers mitigates overfitting and enhances generalization.

Architecture and Hyperparameters:

The implemented FCN architecture consists of an input layer ($32 \times 32 \times 3$), followed by one or two hidden layers with 128–512 neurons each, and a final dense output layer of 10 units with a Softmax activation for multi-class classification. The model was defined as:

```
Input(32,32,3) → Flatten →
[Dense(units, activation), Dropout(rate)] × L →
Dense(10, activation='softmax')
```

The network was trained using the Adam optimizer with `sparse_categorical_crossentropy` loss.

Tuned Hyperparameters:

$$\begin{aligned}
\texttt{num_layers} &\in \{1, 2\},\\
\texttt{units} &\in \{128, 256, 512\},\\
\texttt{activation} &\in \{\text{ReLU}, \text{Tanh}\},\\
\texttt{dropout_rate} &\in \{0.2, 0.3, 0.4\},\\
\texttt{learning_rate} &\in \{0.001, 0.0001, 0.00001\}.
\end{aligned}$$

Effect of Each Hyperparameter:

- **num_layers:** Controls model depth. Increasing layers enables learning more complex representations but increases risk of overfitting.
- **units:** Determines hidden layer width. Larger units capture more features but increase computational cost.
- **activation:** ReLU accelerates convergence by avoiding vanishing gradients; Tanh may improve smoothness for smaller networks.
- **dropout_rate:** Regularizes the model by randomly deactivating neurons during training to prevent overfitting.
- **learning_rate:** Controls optimization speed and stability. Smaller rates ensure stable convergence; higher ones accelerate learning but risk divergence.

Search Method: Hyperparameters were tuned using `RandomSearch` from Keras Tuner, with 20 trials optimizing validation accuracy. Random search was preferred over grid search due to its efficiency in exploring high-dimensional parameter spaces.

Overall, the FCN achieved reasonable accuracy while maintaining computational efficiency and interpretability as a baseline deep learning model. Although it underperforms CNNs on spatial data, it effectively demonstrates the progression from classical machine learning models to neural representation learning within the CIFAR-10 classification task.

3.3 Convolutional Neural Network (CNN)

Theory. Convolutional Neural Networks (CNNs) are designed to exploit the spatial hierarchies present in image data. Unlike fully connected networks, CNNs use convolutional layers with local receptive fields that share weights across spatial dimensions, enabling translation invariance and parameter efficiency. Each convolutional layer extracts spatial features (edges, textures, shapes) that progressively combine into higher-level representations, making CNNs particularly suited for visual recognition tasks. Pooling layers further condense spatial information, reducing dimensionality and improving robustness to minor variations. This inductive bias towards spatial locality and feature hierarchy allows CNNs to generalize effectively on image datasets such as CIFAR-10.

Strengths and Weaknesses. The main strength of CNNs lies in their ability to automatically learn spatially invariant hierarchical features, leading to strong generalization and performance on image classification tasks. They require fewer parameters compared to dense networks of equivalent representational power, improving both computational efficiency and generalization. However, CNNs are computationally more intensive than traditional models such as Random Forests, particularly during training due to convolutional operations and large numbers of filters. Furthermore, CNNs can overfit small datasets without proper regularization and are less interpretable than tree-based methods. Their training time and memory usage also scale with the number of layers and feature maps.

Architecture and Hyperparameters. The implemented CNN follows a standard deep convolutional structure. An input of shape $(32, 32, 3)$ first passes through a data augmentation block performing random horizontal flips, rotations, and zooms to improve generalization. The convolutional feature extractor consists of three stages with filter sizes of 32, 64, and 128, each stage including two 3×3 convolutional layers with ReLU activations, a 2×2 max pooling layer, and batch normalization for stable training. The output

feature maps are aggregated via a Global Average Pooling layer, followed by a dense layer with 256 or 512 units (selected via tuning) and a dropout of 0.25 for regularization. The final softmax layer outputs class probabilities for 10 CIFAR-10 categories. The total trainable parameters are approximately 1.6 million.

Hyperparameters Tuned. The following hyperparameters were tuned using `RandomSearch` with 10 trials and validation accuracy as the optimization objective:

$$\texttt{dense\_units} \in \{256, 512\}, \quad \texttt{learning\_rate} \in \{0.001, 0.0005, 0.0001\}.$$

The `dense_units` parameter controls the model's capacity to combine spatial features into class representations—larger values improve expressivity but may increase overfitting. The `learning_rate` governs the step size of gradient updates; higher values accelerate convergence but risk overshooting minima, while lower values improve stability but slow training. The `RandomSearch` approach was preferred over grid search due to its computational efficiency and ability to explore diverse configurations within limited trials. Batch normalization and dropout were used to mitigate overfitting and stabilize convergence.

The optimal CNN configuration selected `dense_units=512` and `learning_rate=0.0005`, achieving a validation accuracy of 0.74. This model effectively balanced bias and variance, demonstrating the CNN's theoretical strength in capturing spatial hierarchies and outperforming fully connected models on visual data.

4 Results and discussion

4.1 Hyperparameter Results

4.1.1 Random Forest

The Random Forest (RF) classifier was tuned using `HalvingGridSearchCV` on a subset of 20,000 training samples from CIFAR-10. The following hyper parameters were explored:

- `n_estimators`: [100, 200, 300] – number of trees in the forest.
- `max_depth`: [10, 20, 30, None] – maximum depth of each tree.
- `min_samples_split`: [2, 5, 10] – minimum samples required to split an internal node.
- `min_samples_leaf`: [1, 2, 4] – minimum samples required at a leaf node.
- `max_features`: ['sqrt', 'log2'] – number of features considered at each split.

After hyper parameter tuning was run, the top ten combinations are listed below:

Rank	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators	Mean CV Accuracy	Mean Train Time (s)
1	30	sqrt	1	2	300	**0.4431**	466.30
2	20	sqrt	1	2	300	0.4395	355.74
3	30	sqrt	1	2	200	0.4355	324.90
4	20	sqrt	1	2	300	0.4080	120.11
5	30	sqrt	1	2	200	0.4039	85.51
6	30	sqrt	1	2	300	0.4034	129.09
7	20	sqrt	1	2	200	0.4033	85.79
8	None	sqrt	1	2	200	0.4028	85.69
9	None	sqrt	1	2	300	0.4010	128.10
10	30	sqrt	2	5	200	0.3993	78.29

Table 5: Top 10 Random Forest Model Results

During the random forest hyper parameter tuning, a couple of trends were displayed in regards to each variable. The impact of number of trees had diminishing returns, initially starting at a low mean CV accuracy of 0.282 for 100 trees, rising to 0.304 with 200 trees, and then steadily declining to 0.301 with 300 trees. This aligns with the idea that adding more trees to the random forest reduces variance and stabilises predictions, and thus as we add more trees, the results stabilise instead of improving by any significant amount.

In regards to the impact of maximum depth, as the maximum depth increases, the mean CV accuracy also increases, with a max depth of 10 returning a 0.288 mean CV accuracy, a max depth of 20 returning a 0.298 mean CV accuracy, and finally a 30 max depth returning a 0.301 mean CV accuracy. Having a max depth of these values helped ensure that the model did not underfit, or overfit, and thus could obtain the highest accuracy and capture patterns.

Finally, in regards to the training time and accuracy trade off, we noticed that it was increasing at a decreasing rate. The diminishing returns makes sense, as the increased number of trees leads to more stabilised results, despite the increased running times. In conjunction, the maximum tree depth hits a certain point before the results start to become biased, thus lowering the accruracy despite the increased training time.

Some final trends also include the fact that smaller min_samples_leaf and min_samples_split allow for deeper trees, capturing more patterns at the cost of higher training time. As well as this, max_features=sqrt consistently outperformed log2, suggesting better feature subset selection for CIFAR-10 images.

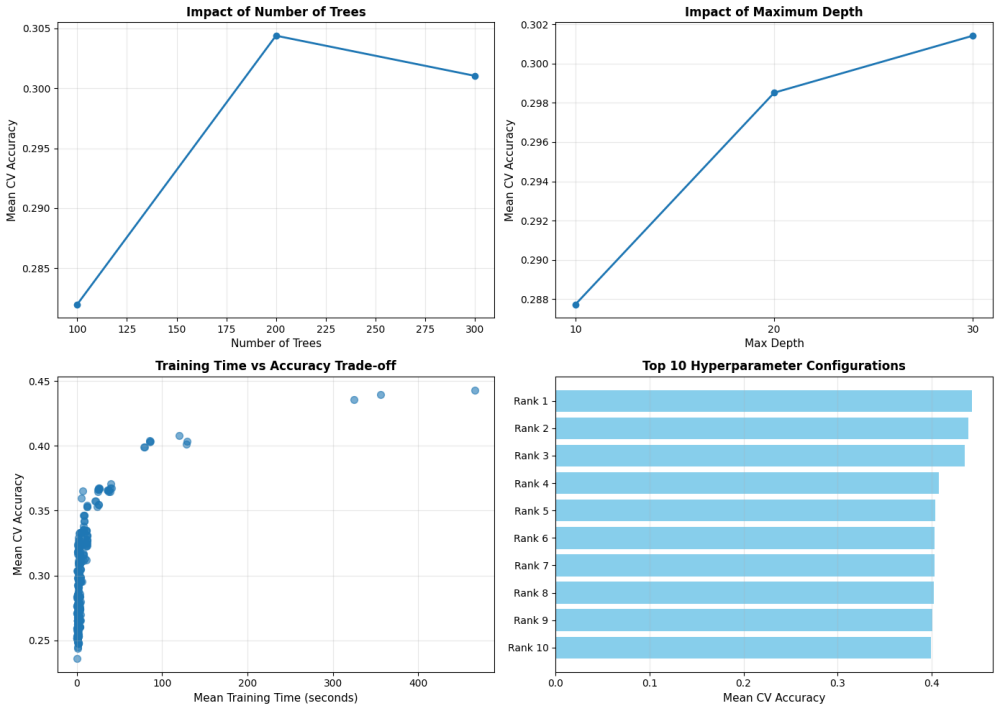

Figure 2: Random forest hyper parameter visualisations

The trends confirm theoretical expectations: more complex forests improve predictive performance but increase computational cost. Lower-ranked configurations converge faster but under perform, consistent with the bias-variance trade-off. Hyperparameter tuning successfully identified a configuration optimizing accuracy and efficiency within practical limits.

Our final results for the parameters were a max tree depth of 30, a sqrt max_features value for generalisation, a min_samples_leaf value of 1, a min_samples_split value of 2 and a n_estimators value of 300. This produced a low value of 0.4431 Mean CV accuracy, and a mean train time of 466.30 seconds, which was significantly longer than the other models. Other observations to note with the Random Forest hyper parameter training process include the fact that it took significantly longer than the other models, with the whole process taking 93.89 minutes. Initial attempts to use a regular Grid Search with a subset size of 10,000 were abandoned due to the extremely long training time. To allow for further experimentation, we also did

an initial test of a 10,000 training subset in the halving grid search, which also produced sub optimal results, which suggests that random forest is not the ideal machine learning algorithm for the CIFAR-10 dataset.

4.1.2 Fully Connected Neural Network

The FCN was tuned using `Keras Tuner RandomSearch` with early stopping (patience=5) on 20,000 training samples. The following hyperparameters were explored:

- `num_layers`: [1, 2] – number of dense layers.
- `units_i`: [128, 256, 512] – number of neurons per dense layer.
- `activation_i`: ['relu', 'tanh'] – activation functions per layer.
- `dropout_i`: [0.2, 0.3, 0.4] – dropout rate after each layer.
- `learning_rate`: [0.001, 0.0001, 0.00001] – learning rate for Adam optimizer.

After hyper parameter tuning was run, the top ten combinations are listed below:

Trial	num_layers	units_1	activation_1	dropout_1	learning_rate	units_2	activation_2	dropout_2	Score
02	1	128	relu	0.2	0.0001	–	–	–	**0.4938**
13	1	512	relu	0.4	1e-05	128	tanh	0.3	0.4914
18	1	128	relu	0.2	0.0001	512	tanh	0.2	0.4828
16	1	256	relu	0.3	1e-05	128	relu	0.3	0.4816
00	1	256	tanh	0.3	0.0001	–	–	–	0.4812
07	2	128	tanh	0.4	0.0001	128	relu	0.2	0.4810
06	1	128	relu	0.4	0.0001	512	tanh	0.2	0.4628
09	1	512	tanh	0.2	1e-05	256	tanh	0.2	0.4592
03	2	128	tanh	0.2	1e-05	128	relu	0.2	0.4498
14	1	512	tanh	0.2	1e-05	128	relu	0.2	0.4422

Table 6: Top 10 MLP Hyperparameter Tuning Results

In the hyper parameter tuning, we can see the following trends in regards to variable values. The learning rate, in regards to achieving mean validation accuracy, reaches a peak, and then decreases afterwards. A 0.00001 learning rate achieves a 0.43 mean validation accuracy, with a 0.0001 learning rate reaching 0.45 mean validation accuracy, and a 0.001 learning rate achieves a 0.34 mean validation accuracy. This aligns with the theory that lower learning rates tend to under fit since the weights don't update as quick, while larger learning rates can cause higher variance, resulting in lower and more unstable accuracy.

In regards to the dropout rate, we can see no major change in accuracy, with a 0.2 dropout rate returning a 0.43 mean validation accuracy rate, a 0.3 dropout rate returning a 0.39 mean validation accuracy, and a 0.4 dropout rate returning a 0.42 mean validation accuracy. This is expected since the closer the drop rate is to 0, the increase in over fitting, while the closer to 1, the increased regularisation, thus causing decreasing accuracy.

We can see an increasing accuracy with an increase in number of units (in regards to layer 1), with 128, 256, and 512 all having values above 0.4 mean validation accuracy, and increasing in order. This matches our theory of the higher the units per layer, the increased validation as they can capture more patterns and thus, are better at predicting. However, we didn't go to high as that would result in larger training times and over fitting of the model. Some final remarks about the model include that tanh activations slightly outperformed relu for this subset, possibly due to better gradient flow for small batch sizes, but the difference was marginal.

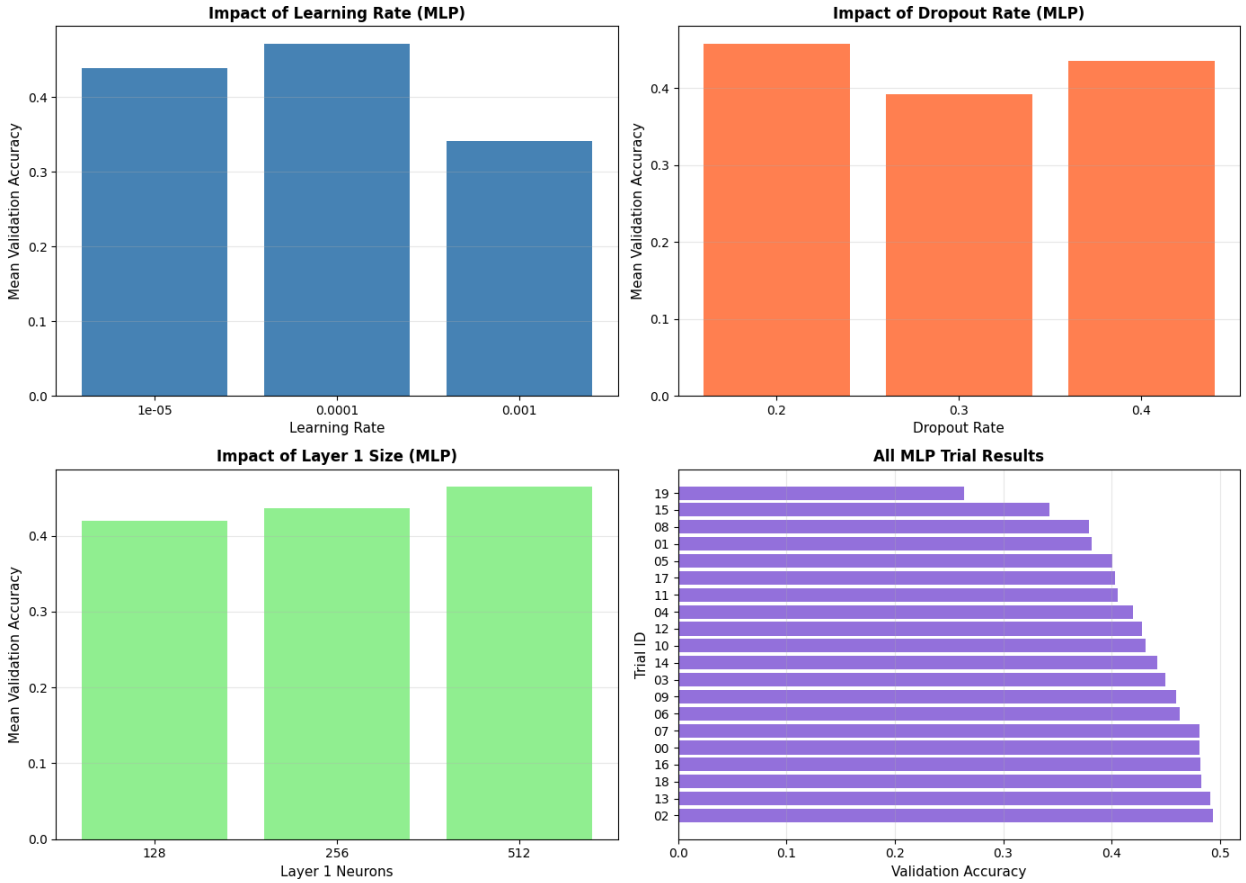

Figure 3: Fully Connected Neural Network hyper parameter visualisations

In the end, our best trial was number two, with a unit value of 128, a dropout rate of 0.2, and an impact layer of 0.0001. The dropout and learning rate aligns with the hyper parameter impact analysis, with a best dropout rate of 0.2 retuning 0.4580 mean validation accuracy, while the best learning rate of 0.0001 returns 0.4717 mean validation accuracy. However, the best trial units (128) contradicts the best unit size of 512. The total tuning time takes 47 minutes and 20 seconds, with a total of 20 trials performed. The best accuracy returned was 0.4938 which is suboptimal, with a mean accuracy of 0.4292, a standard deviation of 0.0577, and a minimum accuracy of 0.2636. Another observation to note is that the early stop prevented over fitting while reducing total turning time.

4.1.3 Convolutional Neural Network

The CNN was tuned using `Keras Tuner RandomSearch` with early stopping (patience=7) on a subset of 20,000 training samples. The hyperparameters explored were:

- `dense_units`: [256, 512] – number of neurons in the fully connected layer before output.
- `learning_rate`: [0.0001, 0.0005, 0.001] – learning rate for Adam optimizer.
- `dropout_rate`: [0.2, 0.3, 0.4, 0.5] - dropout rate for each layer

After hyper parameter tuning was run, the top ten combinations are listed below:

Trial ID	Val Accuracy	Dense Units	Dropout Rate	Learning Rate
04	**0.8108**	512	0.2	0.0010
01	0.8050	256	0.2	0.0010
16	0.8004	256	0.4	0.0010
03	0.7926	512	0.5	0.0005
08	0.7924	512	0.5	0.0010
13	0.7922	512	0.4	0.0005
18	0.7886	256	0.3	0.0010
17	0.7884	256	0.5	0.0005
07	0.7862	256	0.2	0.0005
02	0.7862	256	0.5	0.0010

Table 7: Top 10 CNN Hyper parameter tuning results

When analysing each trial further, we can see the effects of each parameter. For the dense layer size, having the values of 256 and 512 both production highly accurate results, with both values around the 0.75 mean validation accuracy mark. This aligns without prediction as lower unit size would produce an under fitting model, while higher values can cause slight improvements but at the cost of over fitting.

In regards to dropout rates, we can see that the more the dropout rate increases, the more the accuracy drops. This matches our predictions, as the more the drop out rate, the more the neurons are dropped and thus underfitting begins. In conjunction, 0.2 is too high of a rate for the CNN model to overfit.

Increasing the learning rate of the model results in an accuracy rate increase, as seen in the diagram below. This is due to both the higher number of epoches, and the adam optimisers ensuring that 0.001 reaches the best validation accuracy. One impaortant detail to note is that early stopping efficiently prevented overfitting while reducing total tuning time.

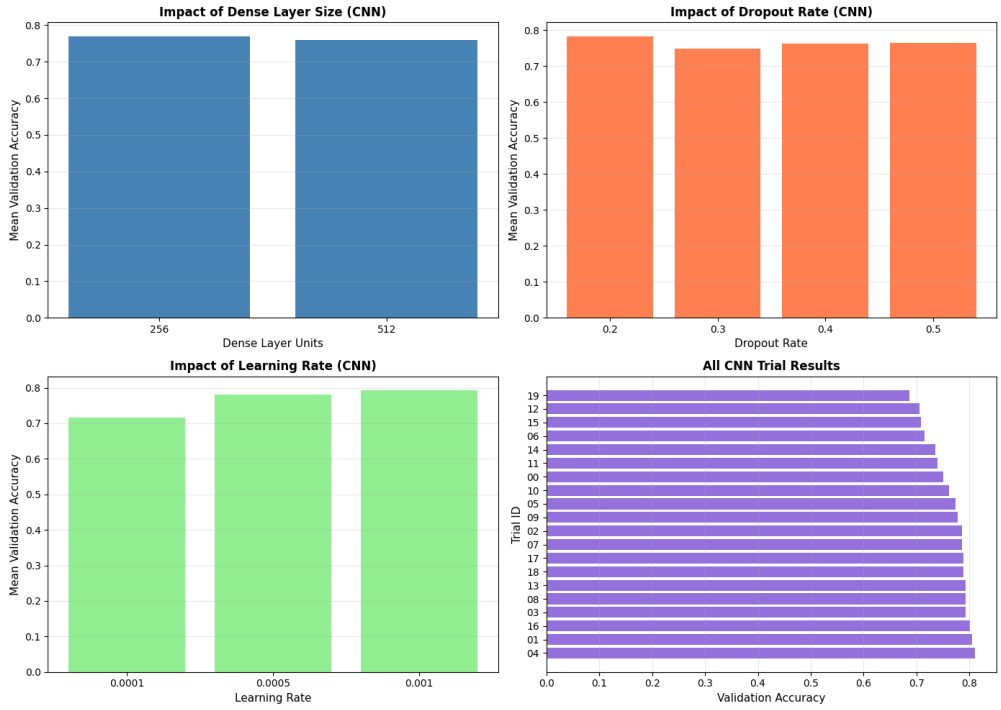

Figure 4: CNN hyper parameter visualisations

The best trial parameters given were a dense unit value of 512, a dropout rate of 0.2, and a learning rate

of 0.001. Similar to the Fully connected neural network results, the dropout rate and the learning rate in the best trial were the same as the best values in the hyper parameter impact analysis, with a dropout rate value of 0.2 producing a 0.7831 mean validation accuracy, while a 0.001 learning rate value produces 0.7921 mean validation accuracy. The highest mean validation accuracy came from a dense unit value of 256, instead of 512, however, given the closeness of mean validation accuracy values, the best trial dense layer size of 512 is not a surprise. The hyper parameter tuning process of CNNs was fairly quick, with the total time taking 1 hour,,minutes and 44 seconds over 20 trials, even with an epoch value of 50. All trials produced a high accuracy, with the best accuracy being 0.8108, the mean accuracy being 0.7649, and the lowest accuracy being 0.6864. Given we had a smaller subset and low amount of network layers, we can predict that a deeper CNN with a larger training set would produce an even higher accuracy result, and thus the CNN is optimal when analysing the CIFAR-10 dataset. Overall, CNN achieved significantly higher accuracy than FCN due to convolutional layers capturing spatial hierarchies in image data.

4.2 Performance Results and Analysis

4.2.1 Test accuracy and loss

Both the Random Forest (RF) and Fully Connected Neural Network (FNN) have low accuracy while the the Convolutional Neural Network is much more effective and has a higher accuracy. Despite having a training accuracy of 0.9999 (99%), the RF testing accuray is only 0.4886 (48.86%), with an overfitting value of 51.13%. This is due to the nature of the CIFAR dataset, with its high dimensionality and spatial structure causing issues, as RF struggles with high dimensional and highly correlated features, choosing to treat the pixels as independent features instead of patterns, thus decreasing accuracy. In conjunction, image features require smooth non linear boundaries, as objects defined by complex hierarchical features such as edges and shapes. The RF model can't learn these abstractions and therefore the accuracy of the model decreases.

The FNN model has an test accuracy of 0.4892 and a test loss of 1.4401, suggesting poor accuracy and inefficiencies. This stems from the fact that FNNs destroy the spatial awareness between pixels, choosing to flatten the images into 1D vectors and thus rendering the model unable to learn any shape patterns. As well as this, the model needs to relearn the image/class pattern at every location, and thus in classes and images where non centering of images is an issue, the FNN needs to relearn the pattern, leading to increased training times and lower accuracy. This results in a poor model that is not suitable for image identification.

The CNN performed extremely well, producing a final test accuracy was 0.8341 with a test loss of 0.4915. Notably, this performance was obtained using a subset of the dataset and reduced parameter values during hyperparameter tuning, highlighting the CNN's robustness and efficiency. This improved performance stems from the use of convolutional layers to capture spatial dependencies, that combine into complex shapes, allowing for pattern and image identification. The translation invariance due to pooling layers results in smaller shifts (off centre images) don't affect the predictions and pattern learning, leading to a more accurate model. Combining this with each filters weights being shared across all spatial features means that the model is more robust and able to identify patterns in the image in any location. Therefore, the CNN model is extremely well-suited to dealing with the CIFAR dataset.

4.2.2 Classification Report

Table 8: Classification Report for Random Forest

Class	Precision	Recall	F1-Score	Support
airplane	0.55	0.56	0.56	1000
automobile	0.55	0.56	0.55	1000
bird	0.43	0.36	0.39	1000
cat	0.37	0.27	0.31	1000
deer	0.43	0.41	0.42	1000
dog	0.45	0.42	0.43	1000
frog	0.47	0.60	0.52	1000
horse	0.54	0.49	0.52	1000
ship	0.59	0.62	0.60	1000
truck	0.48	0.58	0.53	1000
Accuracy			**0.49**	10000
Macro Avg	0.48	0.49	0.48	10000
Weighted Avg	0.48	0.49	0.48	10000

With the above table, we have more detail about how each class performed, and can use this to find key insights about the model. Airplane, automobile, and ship all performed well, achieving in the F1 score range of 0.55-0.60. This highlights how distinct and consistent feature categories tended to perform the best, as patterns were able to be formed easier. Contrasting this, we have the lower tier categories of bird, deer, dog and cat which a f1 score of 0.30-0.42. The lower performance stems from these classes being mainly identified by shape and texture features rather than colour, which the Random Forest model is unable to capture effectively. Furthermore, the model's poor accuracy result (0.49), and poor macro average (0.48) reinforces the idea that Random Forests are not well suited for visual pattern recognition tasks involving spatial dependencies or hierarchical feature relationships.

Table 9: Classification Report for Fully Connected Neural Network

Class	Precision	Recall	F1-Score	Support
airplane	0.58	0.52	0.55	1000
automobile	0.64	0.56	0.59	1000
bird	0.37	0.36	0.36	1000
cat	0.36	0.29	0.33	1000
deer	0.40	0.41	0.41	1000
dog	0.44	0.34	0.39	1000
frog	0.48	0.57	0.52	1000
horse	0.51	0.58	0.55	1000
ship	0.60	0.64	0.62	1000
truck	0.49	0.61	0.54	1000
Accuracy			**0.49**	10000
Macro Avg	0.49	0.49	0.49	10000
Weighted Avg	0.49	0.49	0.49	10000

The FNN's downfalls and inaccuracies stem from the same reasons as the RF's inaccuracies. Since the flattening of the data destroys the spatial structure, the FNN model tends to also struggle with classes that rely on shape and spatial relationship between pixels, such as cat, dog, deer, and bird classes. The easily distinguishable texture classes such airplane, automobile, and ship all tend to perform pretty well in regards to the model's performance. The accuracy, macro average, and weighted average of 0.49 demonstrate the FNN's limitations of fully connected structures for image classification tasks, and thus is not suited for visual datasets/identification.

Table 10: Classification Report for Convolutional Neural Network

Class	Precision	Recall	F1-Score	Support
airplane	0.82	0.87	0.85	1000
automobile	0.90	0.92	0.91	1000
bird	0.85	0.75	0.80	1000
cat	0.68	0.73	0.71	1000
deer	0.84	0.81	0.82	1000
dog	0.76	0.79	0.78	1000
frog	0.87	0.86	0.86	1000
horse	0.92	0.84	0.88	1000
ship	0.94	0.86	0.90	1000
truck	0.79	0.92	0.85	1000
Accuracy			**0.83**	10000
Macro Avg	0.84	0.83	0.83	10000
Weighted Avg	0.84	0.83	0.83	10000

The classification report for the Convolutional Neural Network (CNN) demonstrates a significant improvement over the Random Forest and Fully Connected Neural Network models, achieving an accuracy, macro average, and weighted average of 0.83. This demonstrates a strong consistent performance, with not one class having a ¡0.7 f1 score value. Classes with highly distinctive visual features, such as airplane, automobile, frog, horse, ship and truck, all achieved a high F1 score, in the range of 0.85-0.91, reflecting the model's ability to capture clear spatial patterns. Contrasting with the previous models, classes with reliance on shapes or textures, (bird, cat, dog, deer), also show strong performance (F1 scores 0.71–0.84). This demonstrates the CNN's capability to learn hierarchical and spatial features through convolutional layers and pooling operations. The improved performance stems from translation invariance and weight sharing, allowing for patterns to be captured efficiently. Overall, the CNN is highly effective for visual classification tasks, particularly for datasets like CIFAR-10 that require spatial and hierarchical feature learning.

4.2.3 Confusion Matrix Visualisation and analysis

To further understand misclassification patterns, the confusion matrix was plotted, as shown in Figure 5. Diagonal dominance is weaker than that of the CNN model, reflecting the FCN's limited ability to capture spatial dependencies.

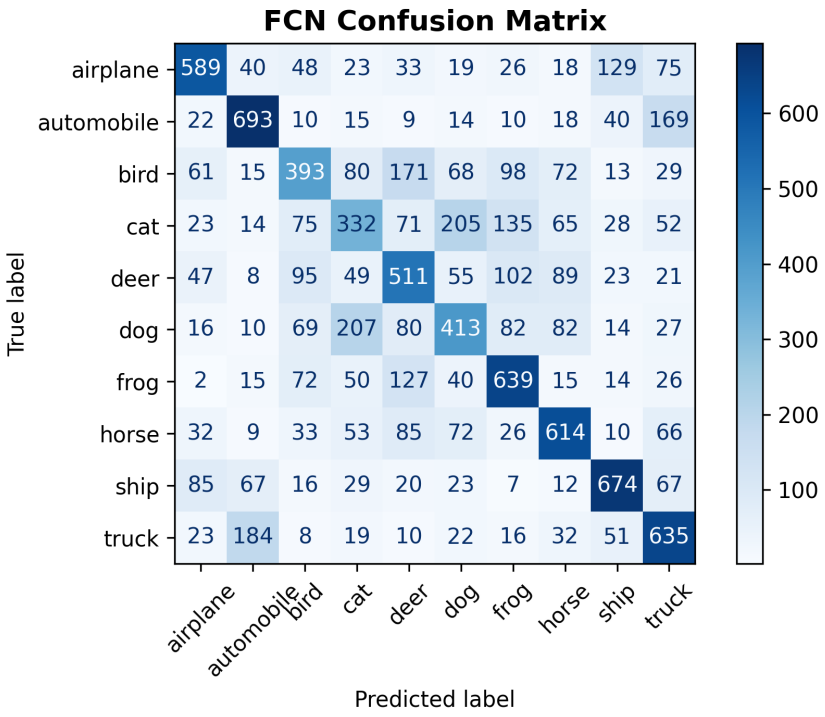

Figure 5: Confusion matrix for FCN model predictions on CIFAR-10.

Empirical observations reveal that the FCN performs relatively well on simpler, less texture-dependent classes such as *automobile*, *ship*, and *airplane*, where global colour and shape features are discriminative. However, classes like *cat*, *dog*, and *bird*—which share high intra-class variance and overlapping texture patterns—exhibit lower precision and recall.

The FCN lacks convolutional filters and therefore fails to capture local spatial hierarchies and translational invariance inherent to image data. Its reliance on flattened pixel representations limits the model's ability to learn robust feature hierarchies. This explains why classes characterised by fine-grained spatial patterns (animals) are confused with each other, while structured, geometrically distinct classes (vehicles) are relatively better classified.

The confusion matrix supports this, as significant confusion occurs among animal categories (*cat-dog-bird-deer*), indicating the FCN's struggle to distinguish subtle differences without spatial inductive bias. The CNN model, by contrast, overcomes this limitation through localised receptive fields and parameter sharing.

To further interpret the CNN's classification behaviour, we visualise the model's predictions using a confusion matrix and per-class performance metrics. These visualisations provide insights into inter-class confusion and highlight the strengths and weaknesses of the trained network.

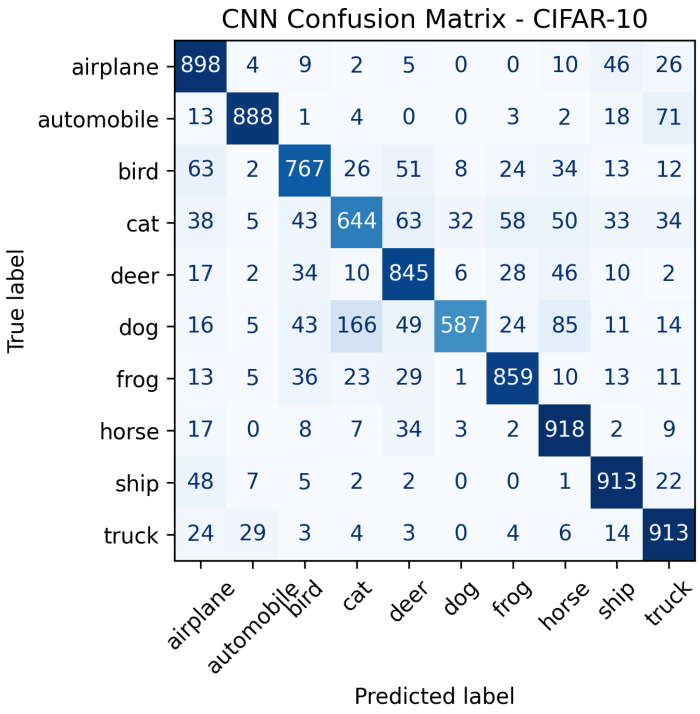

Figure 6: Confusion matrix of CNN predictions on the CIFAR-10 test set. The diagonal entries indicate correct classifications, while off-diagonal values represent misclassifications. Strong diagonals across most classes indicate effective feature learning, with noticeable confusion between semantically similar classes such as *cat–dog* and *deer–horse*.

The confusion matrix (Figure 6) shows that the CNN achieves strong class separation for objects with distinct geometric or colour features, such as *automobile*, *ship*, and *airplane*. However, misclassifications appear mainly between visually similar categories. The per-class F1-score distribution aligns with this observation, highlighting the impact of inter-class similarity on performance. The relatively balanced macro and weighted F1-scores indicate that the model generalises well across most classes, though improvements in discriminative feature learning for fine-grained categories could further enhance accuracy.

4.2.4 Comparison of Feature Representations in CNN and FCN:

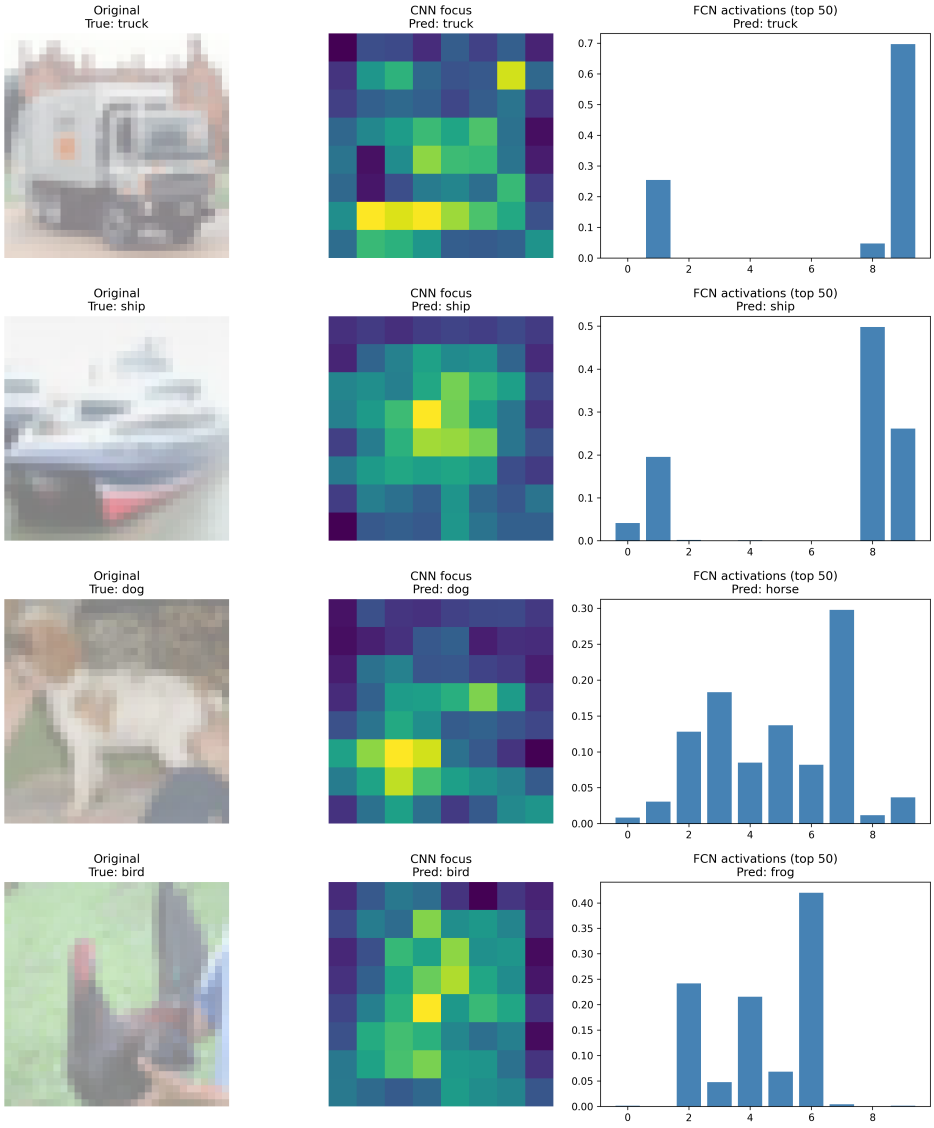

Figure 7: CNN hyper parameter visualisations

Figure 7 illustrates how the CNN and FCN models focus on different aspects of the input images for classification. The left column shows the original test images, with the true class labels indicated. The middle column presents the CNN feature maps, averaged across channels, highlighting the spatial regions that the convolutional layers prioritise when making predictions. The right column depicts the FCN neuron activations in the last hidden layer, represented as bar plots for the top 50 neurons.

The visualisation of the CNN predictions and the FCN activations reiterate the fact that the local receptive fields of CNNs help the CNN model perform better on classes with high intra-class variance. FCNs rightly predicts truck and ship in the first two rows of 7 which has clear global deterministic semantics and structures but struggles for classes with overlapping texture patterns. FCNs have uncertain activations spread across various classes for the dog and the bird images in row 3 and row 4.

4.3 Runtime Analysis

In regards to runtime, the slowest implementation was RF classifier, 19.88 minutes, followed by the CNN, 6.23 minutes, and then the FNN, 5.2 minutes. Differences between the RF classifier and the neural networks stem from the use of a GPU when training, thus allowing for more parallelisation. The large dataset of high dimensional features also make the RF model computationally expensive and slow, leading to a larger training and fitting time. A notable feature that also decreased run time was the use of an early stoppage in the neural networks, which was absent for RF, resulting in even quicker times for neural network models.

Since the dataset was reduced for speed, the difference in total parameters between FNN (411,000) and CNN (359,000) was relatively small compared to the possibility of utilising the full dataset. Both models also used 50 epochs, which increased the run time despite the early stop since it involved rerunning training samples. A key difference between the neural networks was batch size, with FNN having a 32 batch size, while CNN has a 64 batch size. This resulted in a reduction of iterations per epoch, but meant an increased memory usage. This therefore explains the difference in not only running the final models, but also the training time discrepancies.

5 Conclusion

The report systematically compared Random Forest, Fully Connected Neural Network (FCN), and Convolutional Neural Network (CNN) models for CIFAR-10 image classification. The results demonstrate that while Random Forest and FCN provide valuable baselines, their performance is fundamentally limited by the loss of spatial relationships within image data. The CNN significantly outperformed both, achieving over 83% test accuracy by effectively leveraging convolutional layers to learn hierarchical spatial features.

Empirical findings confirmed theoretical expectations: Random Forests excel in interpretability but lack spatial sensitivity, FCNs capture global non-linear relations yet fail to model local dependencies, and CNNs achieve superior generalisation by learning translation-invariant patterns. The progression from traditional to deep learning models highlights the critical role of architecture in aligning model design with data structure.

Overall, this work reinforces the importance of convolutional architectures for visual recognition tasks and demonstrates how methodological design, data understanding, and appropriate regularisation jointly determine model performance and generalisation capability.

While our results provided insights into each algorithm, the experiment could be improved. The usage of subsets and lower complexity models (CNN and FNN being limited in the number of layers) was necessary to keep training times low, but limited the performance of models and prevented additional trends from being analysed. Future improvements to the experiment could be made with increased subset size, as well as increased layers for the models. In conjunction, the usage of another similar visual dataset could provide more insight into models and allow for further comparisons.

6 Reflection

The importance of data understanding (SID: 520131808): I learned that the choices we make about the data directly affects how the models perform. For example, when we flatten the images for the FCN and RF models, the implied assumption was that the model could generalise spatial patterns from a 1d list of pixels. As it turns out, this process makes it very difficult for the models to recognise spatial structures the images. In contrast, the CNN's superior performance stemmed directly from its design, which respects and utilises the spatial nature of images.

Similarly, our initial data exploration showed suggested that objects like trucks and automobiles had higher contrast and more complex edges. This finding affected our choice to use data augmentation techniques for

the CNN, as we believed adding rotational and zoom variety to the images could help the CNN learn more robust features rather than just memorising the training images and thereby improve model performance.

Trade-offs between model complexity and performance (SID: 510458096): Another important learning was the balance between model complexity, computational cost, and performance. The random forest model, although interpretable, was computationally expensive and produced modest accuracy, illustrating the bias-variance and efficiency trade offs of ensemble methods. The FCN demonstrated that depth and nonlinearity can improve learning, but without convolutional inductive bias, the model struggled to capture spatial hierarchies. The CNN, which was the most resource intensive to train, ended up achieving the best generalisation, highlighting that higher complexity can be justified when aligned with data structure.

Two Stages (SID: 550749464): During the two stages of the assignment, I learned about the importance of methodological understanding of various ML models. Specifically in the stage 2, the importance of understanding how the models interpret and perform the task at hand helps us choose the appropriate models for the task. As in the case of image classification on the Cifar-10 dataset, the superior performance of CNNs and neural networks over traditional ML algorithm taught me how CNNs propagate the spatial pixel data through various layers to understand the 2D image data much better than the tradition Random forest or even a simple MLP. The receptive fields of CNNs can keep the spatial information of the images when it is going from the RGB space to a very high-dimensional representation of the image which helps it retain structure and semantics which the other two models fail to capture.

From the entire process of training and performing classification using different models, I learned how to approach a ML problem not just for image data but any modality. The key is to appropriately understand the task at hand, the data and the understanding of how the different models are learning from the data.

The effects of hyper parameters when tuning data (SID: 520510542): Through creating and modifying the models, the hyper parameter tuning, and implementing the final hyper parameters, I got a first class view of how each variable not only affects accuracy, but also run time. As mentioned earlier in the report, I had to change from a full grid search to a halving grid CV search in order to keep the run time down. In conjunction to the use of subsets, different batch sizes, and epochs, it opened my eyes to how each variable and modifications can cause massive changes in runtime for little to no improvements. In terms of the hyper parameters, I got to experiment with each, often singling out one parameter with multiple options while keeping the rest of the parameters stable to see the full effects of each change. This helped deepen my understanding of what and why changes where occurring, and allowed the final experiment hypertuning process to be so accurate (accurate as high as the model would allow for).

References

[1] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report, University of Toronto.
`https://www.cs.toronto.edu/ kriz/cifar.html`