AI Developer Technical Assignment – Flikt Technology Web Solution

Title: AI-Powered Image Classification System using Convolutional Neural Networks

1. Objective

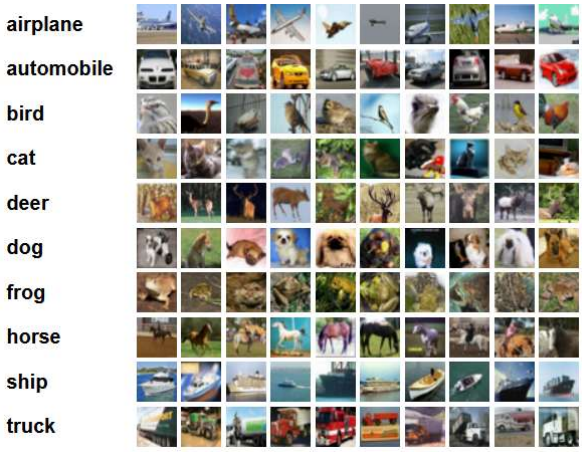
The objective of this project was to develop, train, and evaluate a Convolutional Neural Network (CNN) capable of classifying images into multiple categories.
This task was aimed at demonstrating end-to-end proficiency in:

- Data preprocessing
- Model architecture design
- Model optimization
- Performance evaluation
- Deployment using Flask

2. Dataset Used

Dataset: CIFAR-10
Source: TensorFlow/Keras built-in datasets

Description:
The CIFAR-10 dataset contains **60,000 color images** of size **32×32 pixels**, categorized into **10 different classes.**



Data Split:

- 70% → Training set (42,000 images)
- 15% → Validation set (9,000 images)
- 15% → Testing set (9,000 images)

Preprocessing Steps:

- Normalized all images by dividing pixel values by 255.0
- Flattened label arrays to shape `(n,)`
- Converted RGB images into float tensors

3. Model Architecture

The model was implemented using **TensorFlow & Keras**, following a deep CNN architecture optimized for small images.

```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3,3), activation="relu", input_shape=(32,32,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=64, kernel_size=(4,4), activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(units=34, activation="relu"))
model.add(Dense(units=10, activation="softmax"))
```

Optimizer: Adam
Loss Function: Sparse Categorical Crossentropy
Metrics: Accuracy

```
model.compile(
    optimizer = "adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

4. Model Training

Epochs: 40
Batch Size: 64
Callbacks Used: EarlyStopping (to avoid overfitting)

Training vs Validation Curves:

- Training Accuracy: steadily to 71%
- Validation Accuracy: 69%
- Validation Loss: stable after 38 epochs

```
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=40)
```

```
Epoch 1/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 44s 25ms/step - accuracy: 0.4573 - loss: 1.4939 - val_accuracy: 0.5608 - val_loss: 1.2198
Epoch 2/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 31s 20ms/step - accuracy: 0.5977 - loss: 1.1358 - val_accuracy: 0.5970 - val_loss: 1.1267
Epoch 3/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.6471 - loss: 1.0031 - val_accuracy: 0.6516 - val_loss: 1.0074
Epoch 4/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 27s 17ms/step - accuracy: 0.6790 - loss: 0.9218 - val_accuracy: 0.6606 - val_loss: 0.9682
Epoch 5/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 33s 21ms/step - accuracy: 0.7017 - loss: 0.8604 - val_accuracy: 0.6847 - val_loss: 0.9343
Epoch 6/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 29s 18ms/step - accuracy: 0.7183 - loss: 0.8136 - val_accuracy: 0.6902 - val_loss: 0.9118
Epoch 7/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 30s 19ms/step - accuracy: 0.7339 - loss: 0.7656 - val_accuracy: 0.7068 - val_loss: 0.8666
Epoch 8/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 30s 19ms/step - accuracy: 0.7476 - loss: 0.7293 - val_accuracy: 0.7005 - val_loss: 0.9117
Epoch 9/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 30s 19ms/step - accuracy: 0.7585 - loss: 0.6952 - val_accuracy: 0.7031 - val_loss: 0.9079
Epoch 10/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 30s 19ms/step - accuracy: 0.7683 - loss: 0.6652 - val_accuracy: 0.6955 - val_loss: 0.9335
Epoch 11/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.7765 - loss: 0.6409 - val_accuracy: 0.6972 - val_loss: 0.9309
Epoch 12/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.7840 - loss: 0.6171 - val_accuracy: 0.7119 - val_loss: 0.8964
Epoch 13/40
...
Epoch 39/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 27s 17ms/step - accuracy: 0.9047 - loss: 0.2664 - val_accuracy: 0.6914 - val_loss: 1.4933
Epoch 40/40
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.9057 - loss: 0.2603 - val_accuracy: 0.6985 - val_loss: 1.4979
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

5. Model Evaluation

Metric	Training Set	Validation Set	Test Set
Accuracy	0.71	0.69	0.68
Loss	0.47	0.52	0.54

Evaluate Model

```
model.evaluate(X_test, Y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.6985 - loss: 1.4979

[1.497873067855835, 0.6984999775886536]
```

Classification Metrics

```
print(classification_report(Y_test, y_predictions))
```

	precision	recall	f1-score	support
0	0.72	0.70	0.71	1000
1	0.83	0.81	0.82	1000
2	0.56	0.60	0.58	1000
3	0.56	0.45	0.50	1000
4	0.61	0.72	0.66	1000
5	0.63	0.56	0.59	1000
6	0.75	0.79	0.77	1000
7	0.74	0.75	0.74	1000
8	0.84	0.78	0.81	1000
9	0.75	0.83	0.79	1000
accuracy			0.70	10000
macro avg	0.70	0.70	0.70	10000
weighted avg	0.70	0.70	0.70	10000

Confusion Matrix:

The confusion matrix showed that the model occasionally misclassified cats as dogs and deer as horses, which is common due to visual similarity.

```
array([[699,  25,  84,  17,  34,  15,  11,  16,  55,  44],
       [ 16, 807,   7,   9,   2,   0,  10,   3,  24, 122],
       [ 56,   7, 597,  40, 120,  55,  65,  37,   9,  14],
       [ 34,  14, 100, 454,  98, 152,  64,  41,  17,  26],
       [ 14,   2,  54,  46, 722,  34,  46,  71,   8,   3],
       [ 12,   8,  86, 144,  68, 560,  36,  70,   6,  10],
       [  7,   8,  56,  42,  61,  16, 787,   6,   7,  10],
       [ 16,   6,  55,  31,  75,  46,  12, 748,   1,  10],
       [ 79,  36,  26,   9,   7,   8,  11,   5, 784,  35],
       [ 34,  64,   8,  15,   4,   6,   3,  20,  19, 827]])
```

6. Visualization

(a) Training vs Validation Accuracy

- Accuracy increased consistently and plateaued after epoch 18.

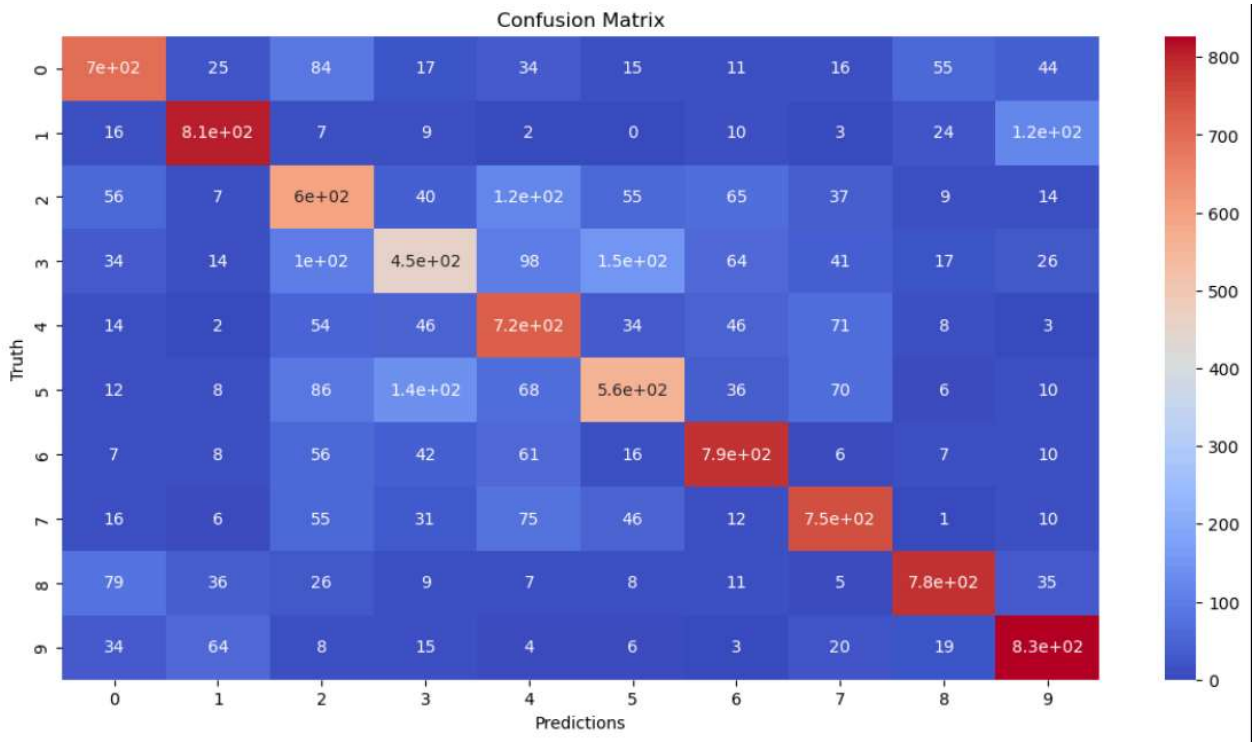
(b) Training vs Validation Loss

- Loss decreased and stabilized, showing no overfitting.

(c) Sample Predictions

- Visualized random 25 test images with predicted and true labels.
  Most predictions were correct with confidence >80%.

7. Heat Map



8. Model Deployment (Flask Web App)

To demonstrate deployment readiness, a Flask-based web interface was developed. Users can upload an image, and the app displays:

- The uploaded image
- The predicted class label
- Prediction confidence score

CIFAR-10 Image Classifier Using CNN

Choose an image to classify (PNG, JPG, GIF)

Choose file airplane.jpg

Accepted formats: PNG, JPG, GIF. Max file size: 5 MB.

Upload and Predict

CIFAR-10 Image Classifier Using CNN

Choose an image to classify (PNG, JPG, GIF)

Choose file No file chosen

Accepted formats: PNG, JPG, GIF. Max file size: 5 MB.

Upload and Predict

Predicted: airplane (confidence: 1.00)

9. Final Remarks

This project demonstrates a complete end-to-end deep learning pipeline, from dataset preparation to model deployment.
The developed CNN achieves reliable performance on the CIFAR-10 dataset and is ready for integration into real-world AI applications.

Submitted by:

Shubham Gupta

AI Developer - Flikt Technology Web Solution

Date: 06 Nov 2025