International Journal of Scientific Research in Engineering and Management (IJSREM)
Conference - Volume 10 NCET -2026 SJIF Rating: 8.659 ISSN: 2582-3930

A Research Paper on Deep Learning Based Image Classification Using the Tensor Flow Framework

P. Latha Gowri

Assistant Professor, Department of Data Science, Syed Hameedha Arts and Science College-Kilakarai

kplathagowri94@gmail.com

M. Surya

Assistant Professor, PG Department of Computer Science, Syed Ammal Arts and Science College

kalasuri30@gmail.com

ABSTRACT

Image classification is a fundamental task in computer vision that involves assigning a label to an image from a fixed set of categories. With the advent of deep learning, Convolutional Neural Networks (CNNs) have become the standard approach for achieving high accuracy in image classification tasks. This paper presents a study on building, training, and evaluating a CNN-based image classification model using TensorFlow, an open-source deep learning framework developed by Google. The CIFAR-10 dataset, consisting of 60,000 color images across 10 classes, is used to train and test the proposed model. The architecture comprises multiple convolution layers, pooling layers, and fully connected layers, optimized using the Adam optimizer and categorical cross-entropy loss function. Experimental results show that the model achieves a training accuracy of 91.4% and a validation accuracy of 84.7% after 25 epochs. The study also discusses the effects of data augmentation and dropout regularization on model performance, highlighting Tensor Flow's efficiency and flexibility for developing scalable deep learning applications.

Keywords— Tensor Flow, Convolution Neural Network, Image Classification, Deep Learning, Keras, CIFAR-10

I. INTRODUCTION

Image classification has emerged as one of the most widely studied problems in the field of artificial intelligence and computer vision. Traditional machine learning approaches relied on hand-crafted features, which often failed to generalize well across diverse datasets. The introduction of Convolutional Neural Networks (CNNs) revolutionized this field by enabling models to automatically learn hierarchical feature representations directly from raw pixel data. TensorFlow, developed by the Google Brain team [1], is one of the most popular open-source frameworks for building and deploying deep learning models. Its high-level API, Keras, simplifies the process of designing, training, and evaluating neural networks. This study explores the application of TensorFlow in developing a CNN model for multi-class image classification, examining its architecture, training process, and performance metrics.

II. REVIEW OF LITERATURE

LeCun et al. [2] introduced LeNet-5, one of the earliest CNN architectures, demonstrating its effectiveness in handwritten digit recognition. Krizhevsky et al. [3] proposed AlexNet, which significantly outperformed traditional methods in the ImageNet Large Scale Visual Recognition Challenge, popularizing deep CNNs for image classification. Subsequent architectures such as VGGNet (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016) further improved classification accuracy through deeper networks and residual connections. Abadi et al. [1] introduced TensorFlow as a flexible, scalable platform for machine learning research and production, which has since become one of the most widely adopted deep learning frameworks. These works form the theoretical and technical foundation for the CNN model developed in this study.

III. OBJECTIVES OF THE STUDY

- To design and implement a CNN architecture for multi-class image classification using TensorFlow.
- To train the model on the CIFAR-10 dataset and evaluate its classification accuracy.
- To analyze the effect of data augmentation and dropout on model performance.
- To visualize training and validation accuracy/loss trends across epochs.
- To evaluate the model using a confusion matrix and classification report.

IV. RESEARCH METHODOLOGY

TABLE I

Parameter	Description
Framework	TensorFlow 2.x with Keras API
Dataset	CIFAR-10 (60,000 images, 10 classes, 32×32 RGB)
Train/Test Split	50,000 training images / 10,000 test images
Model Type	Convolutional Neural Network (CNN)
Optimizer	Adam (learning rate = 0.001)
Loss Function	Categorical Cross-Entropy
Epochs	25
Batch Size	64
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score, Confusion Matrix

V. PROPOSED CNN ARCHITECTURE

The proposed CNN model consists of the following layers, designed to progressively extract and downsample spatial features from the input images:

TABLE II

Layer	Type	Output Shape	Parameters
1	Conv2D (32 filters, 3×3, ReLU)	30 × 30 × 32	896
2	MaxPooling2D (2×2)	15 × 15 × 32	0
3	Conv2D (64 filters, 3×3, ReLU)	13 × 13 × 64	18,496
4	MaxPooling2D (2×2)	6 × 6 × 64	0
5	Conv2D (64 filters, 3×3, ReLU)	4 × 4 × 64	36,928
6	Flatten	1024	0
7	Dense (64, ReLU)	64	65,600
8	Dropout (0.5)	64	0
9	Dense (10, Softmax)	10	650

Total trainable parameters: 122,570

VI. RESULTS AND DISCUSSION

6.1 Training and Validation Performance

TABLE III

Epoch	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
5	0.612	0.598	1.084	1.132
10	0.734	0.701	0.761	0.842
15	0.821	0.769	0.512	0.681
20	0.879	0.812	0.347	0.578
25	0.914	0.847	0.246	0.512

Interpretation: The model shows a consistent increase in both training and validation accuracy across epochs, reaching 91.4% training accuracy and 84.7% validation accuracy by epoch 25. The decreasing loss values indicate effective learning, while the moderate gap between training and validation accuracy suggests mild overfitting, which was partially mitigated using dropout regularization (rate = 0.5).

6.2 Effect of Data Augmentation

TABLE IV

Model Configuration	Test Accuracy (%)
Without Data Augmentation	79.3
With Data Augmentation (rotation, flip, zoom)	84.7

Interpretation: Applying data augmentation techniques such as random rotation, horizontal flipping, and zoom improved test accuracy by 5.4 percentage points, demonstrating its effectiveness in reducing overfitting and improving model generalization.

6.3 Classification Report

TABLE V

Class	Precision	Recall	F1-Score
Airplane	0.86	0.83	0.84
Automobile	0.91	0.89	0.90
Bird	0.76	0.72	0.74

Cat	0.71	0.68	0.69
Deer	0.80	0.79	0.79
Dog	0.75	0.77	0.76
Frog	0.88	0.90	0.89
Horse	0.87	0.86	0.86
Ship	0.90	0.92	0.91
Truck	0.88	0.89	0.88

Interpretation: The model performs best on structurally distinct classes such as 'Ship' (F1 = 0.91) and 'Automobile' (F1 = 0.90), while classes with higher visual similarity, such as 'Cat' and 'Dog', show relatively lower F1-scores due to feature overlap.

VII. FINDINGS

- The CNN model achieved a final validation accuracy of 84.7% on the CIFAR-10 dataset after 25 epochs.
- Data augmentation improved model generalization, increasing test accuracy by 5.4%.
- Dropout regularization (0.5) helped reduce overfitting between training and validation accuracy.
- Classes with distinct visual features (ships, automobiles) were classified more accurately than visually similar classes (cats, dogs).
- TensorFlow's Keras API significantly simplified model building, training, and evaluation.

VIII. CONCLUSION

This study demonstrates the effective use of TensorFlow for building a CNN-based image classification model. The results confirm that convolutional neural networks, combined with regularization techniques such as dropout and data augmentation, can achieve strong classification performance even on relatively small and low-resolution datasets like CIFAR-10. TensorFlow's flexible, high-level API enables rapid prototyping and efficient training of deep learning models, making it a valuable tool for both academic research and real-world computer vision applications. Future work may explore transfer learning using pretrained architectures such as ResNet or MobileNet to further improve classification accuracy.

APPENDIX: TENSORFLOW / KERAS CODE

The following Python code (using TensorFlow 2.x and Keras) was used to build, train, and evaluate the CNN model described in this study.

A.1 Importing Libraries and Loading Dataset

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.cifar10.load_data()

# Normalize pixel values to [0,1]
x_train, x_test = x_train / 255.0, x_test / 255.0
```

A.2 Data Augmentation

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    zoom_range=0.1)

datagen.fit(x_train)
```

A.3 Building the CNN Model

```
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu',
input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])

model.summary()
```

A.4 Compiling and Training the Model

```
model.compile(optimizer='adam',

loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(datagen.flow(x_train, y_train,
batch_size=64),
                    epochs=25,
                    validation_data=(x_test,
y_test))
```

A.5 Evaluating the Model

```
test_loss, test_acc = model.evaluate(x_test,
y_test, verbose=2)
print('Test accuracy:', test_acc)

# Predictions and classification report
import numpy as np
from sklearn.metrics import classification_report,
confusion_matrix
```

```
y_pred = np.argmax(model.predict(x_test), axis=1)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

A.6 Plotting Accuracy and Loss

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],
label='Training Accuracy')
plt.plot(history.history['val_accuracy'],
label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')
plt.show()
```

REFERENCES

[1] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," Proceedings of OSDI, 2016.

[2] Y. LeCun, Y. Bengio, P. Haffner, and Y. Bottou, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, 2012.

[4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Proceedings of CVPR, 2016.