

# DEEP LEARNING APPROACH:

## Image classification using CNN and Transfer Learning

*By: Anirudh Unni, Natalia Dominguez and Zahory Vasquez*

# Overview

▶ Project Overview	03
▶ Dataset Description	04
▶ Data Preprocessing	05
▶ Custom CNN	04
▶ Architecture & Training	
▶ Evaluation & Improvements	
▶ CNN Evaluation	05
▶ Transfer Learning Approach	06
▶ Performance Comparison	07
▶ Conclusions	08
▶ Future Work	09



# PROJECT OVERVIEW

## Goal:

- Classify images from the CIFAR-10 dataset into 10 categories.

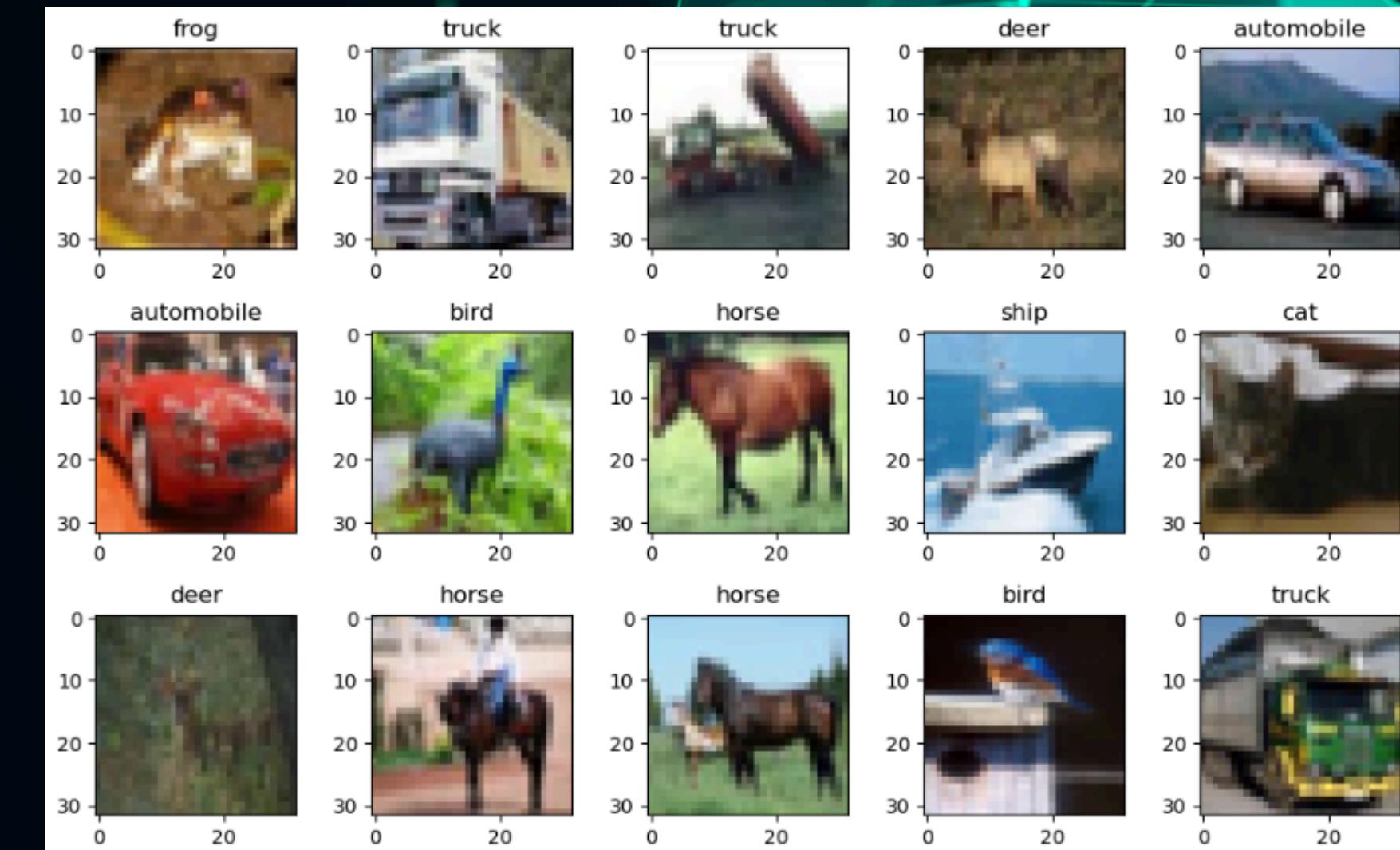
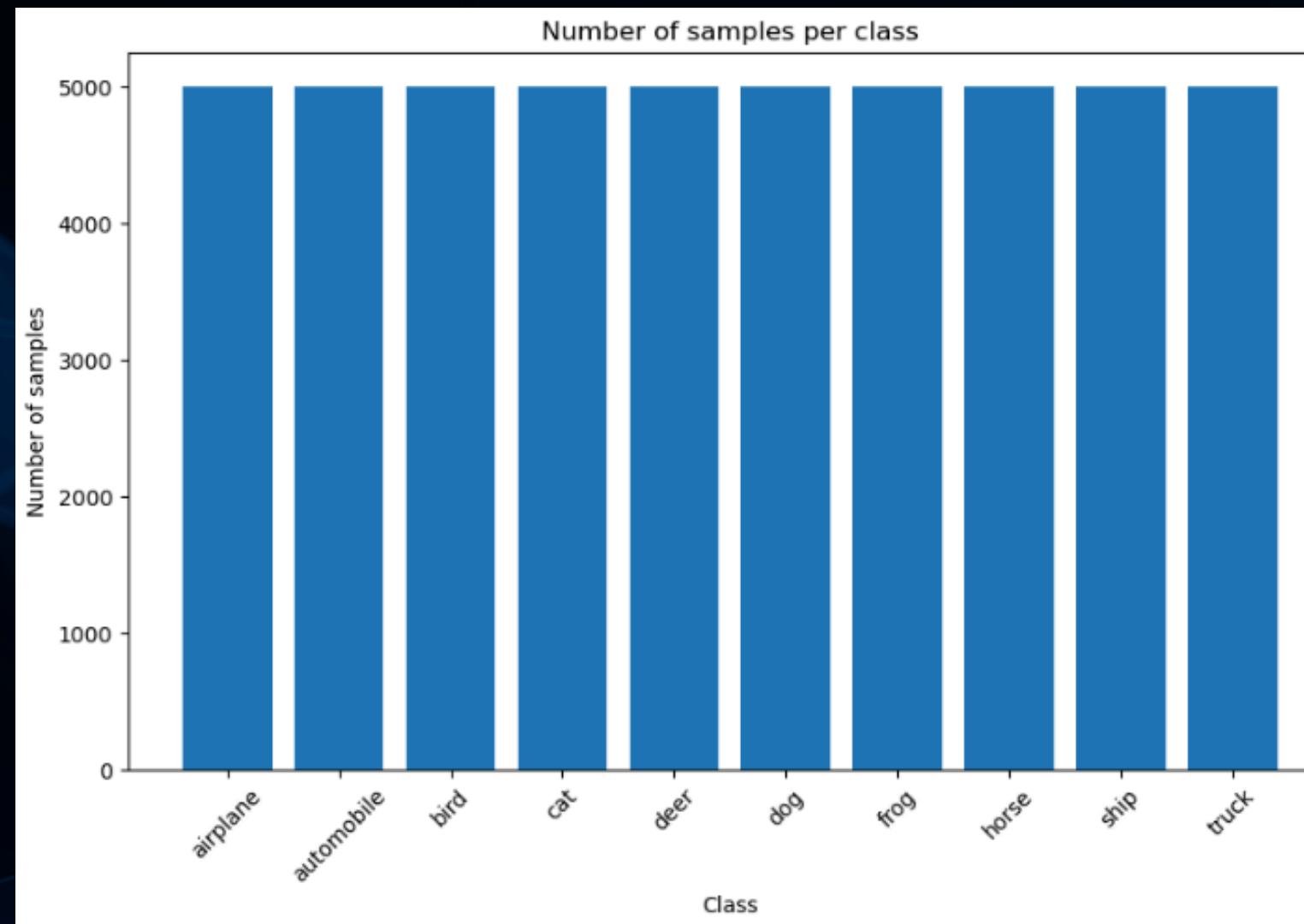
## Approaches:

- Custom CNN built from scratch.
- Transfer Learning using pre-trained models.



# DATASET DESCRIPTION

- Dataset: CIFAR-10
- Size: 60,000 color images (32×32 pixels).
- Classes: 10 categories



# DATA PREPROCESSING

Before training, the dataset must be prepared to fit the model:

1. **Reshape:** Adjust input images to include the channel dimension required by CNNs.
2. **Normalize:** Scale pixel values to the [0, 1] range to improve training efficiency.
3. **Encode:** Convert class labels into one-hot vectors for multi-class classification.

Implementation uses [Keras](#) utilities:

- Rescaling for normalization
- `to_categorical` for label encoding



# CUSTOM CNN ARCHITECTURE & TRAINING

## Model Architecture:

- Convolutional layers with ReLU activations
- MaxPooling layers for feature reduction
- Flatten layer → Fully Connected dense layers

## Model Training:

- Optimizers: Adam
- Loss function: Categorical Crossentropy

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295,040
dense_1 (Dense)	(None, 10)	1,290

Total params: 315,722 (1.20 MB)

Trainable params: 315,722 (1.20 MB)

Non-trainable params: 0 (0.00 B)

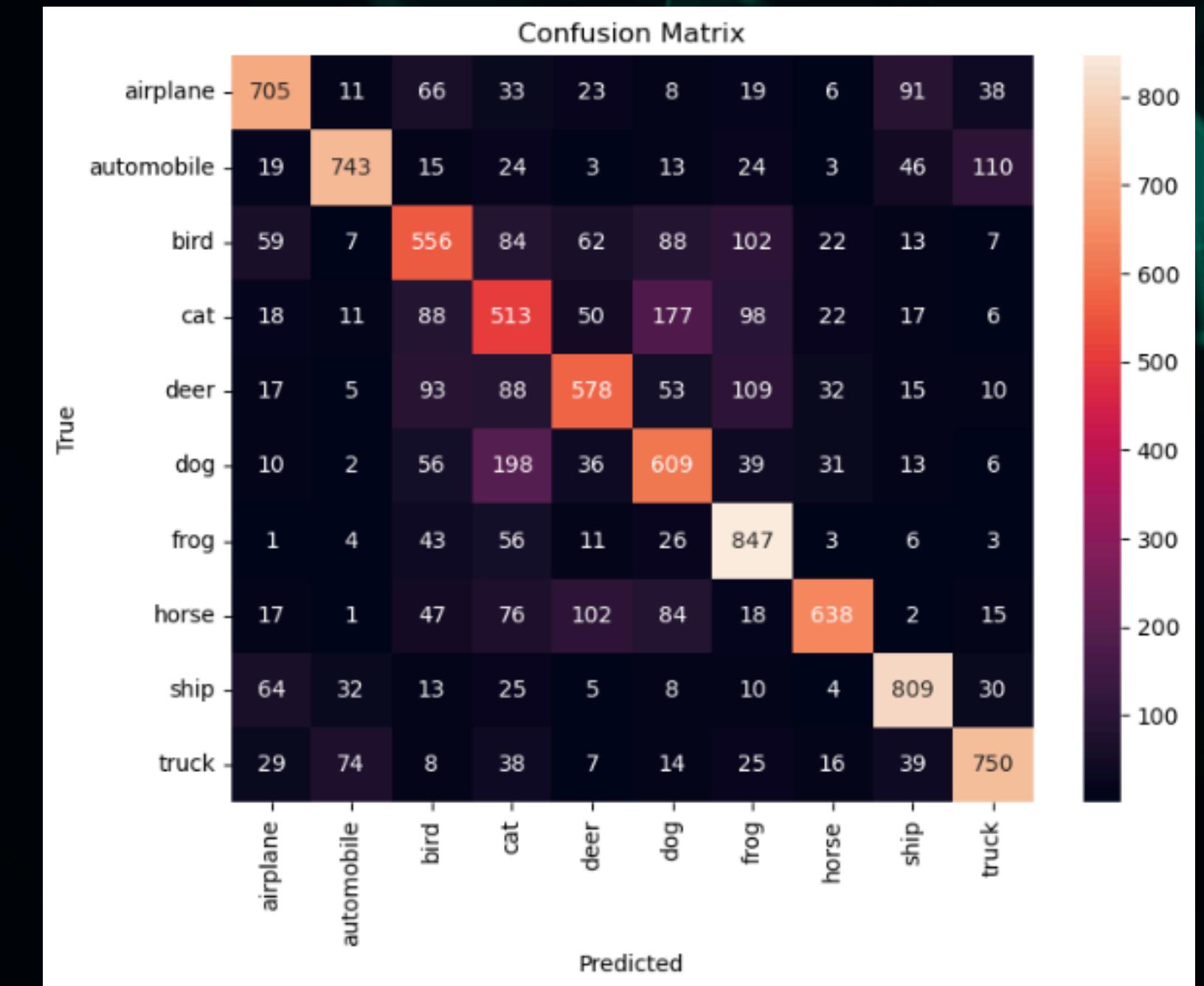
# CUSTOM CNN

## EVALUATION & IMPROVEMENTS

### Model Evaluation:

- Metrics: Accuracy, Precision, Recall, F1-score
- Confusion matrix to analyze class-level performance

Accuracy: 0.6748  
Precision: 0.6841  
Recall: 0.6748  
F1 Score: 0.6761



# CUSTOM CNN EVALUATION & IMPROVEMENTS

## Improving the Model:

- Added more convolutional layers for deeper feature extraction
  - Added 1 more layer (previously only had 2) + Dropout

**Test accuracy with more layers: 0.7343**

- Added BatchNormalization + GlobalAveragePooling2D

**Test accuracy with more layers: 0.7313**

- Added Regularizer

**Test accuracy with more layers: 0.7249**

- Added Hyperparameter tuning - Earling Stopping.

**Test accuracy with early stopping: 0.7293**

**Training accuracy: 0.85**

**Training accuracy: 0.91**

**Training accuracy: 0.93**

- Applied data augmentation (rotations, flips, shifts) to improve generalization

**Test accuracy with data augmentation: 0.7694**

**Training accuracy: 0.94**

**Training accuracy: 0.74**

# TRANSFER LEARNING APPROACH

Instead of evaluating multiple architectures, our approach focused exclusively on MobileNetV2 (pre-trained on ImageNet). The model was tested in three different configurations, varying which layers were frozen and the extent of fine-tuning, in order to compare performance and training time.



# METRICS & EVALUATION

**MobileNetV2.1**

## Key Features

- Simple dense head: GlobalAveragePooling → Dense(128, ReLU) → Dense(10, softmax)
- Training: 5 epochs, batch size 64

## Results

Test accuracy: 0.8658

**MobileNetV2.2**

- Dense head with dropout: GlobalAveragePooling → Dropout(0.3) → Dense(10, softmax)
- Training: Frozen base first phase, 10 epochs, default batch size

Test accuracy: 0.8562

**MobileNetV2.3**

- Same architecture as V2.2
- Training: Frozen base first phase, 10 epochs, batch size 64

Test accuracy: 0.8557

# CONCLUSIONS

## Custom CNN (with Data Augmentation):

- Solid baseline model.
- Achieved moderate performance with accuracy = 0.7694.
- Requires longer training time since all weights are learned from scratch.

## Transfer Learning (MobileNetV2.1):

- Achieved higher accuracy = 0.8658.
- Faster convergence by leveraging pre-trained features.
- Better generalization compared to training from scratch.

## Takeaway:

- Transfer learning is the recommended approach for similar image classification tasks.

# FUTURE STEPS

- Test other pre-trained models (e.g., EfficientNet, ResNet).
- Fine-tune more layers for better feature learning.
- Apply advanced data augmentation techniques.
- Optimize hyperparameters for higher accuracy.

