**Report on CNN-based Image Classification**

**1. Chosen CNN Architecture**

Two different CNN architectures were implemented:

**Custom CNN Model**

* **Input Shape**: (32, 32, 3)
* **Layers**:
  + Conv2D (32 filters, 3x3, ReLU)
  + MaxPooling2D (2x2)
  + Conv2D (64 filters, 3x3, ReLU)
  + MaxPooling2D (2x2)
  + Flatten
  + Dense (64 neurons, ReLU)
  + Dense (10 neurons, softmax)

**Pre-trained VGG16-based Model**

* **Base Model**: VGG16 (pre-trained on ImageNet, include\_top=False)
* **Custom Layers**:
  + Flatten
  + Dense (256 neurons, ReLU) or Dense (64 neurons, ReLU)
  + Dropout (0.5 in one variation)
  + Dense (10 neurons, softmax)

**2. Preprocessing Steps**

* **Batch-wise Preprocessing & Saving**:
  + Data split into batches of 1000 samples and saved as .npy files.
  + Efficient data loading using tf.data.Dataset.from\_tensor\_slices().
* **Batching**:
  + Training and test datasets are batched with a size of 16 to optimize memory usage.

**3. Training Process**

* **Optimizer**: Adam
* **Loss Function**: Categorical Crossentropy
* **Metrics**: Accuracy
* **Batch Size**: 16
* **Epochs**: 50
* **Data Handling**:
  + Training performed on dynamically loaded preprocessed batches to manage memory efficiently.

**4. Results & Model Performance**

* **Test Loss & Accuracy**:
  + Evaluated using cnn.evaluate(X\_test, y\_test).
  + Accuracy logged using MLflow.
* **Classification Report & Confusion Matrix**:
  + classification\_report(y\_true, y\_pred, target\_names=classes)
  + accuracy\_score(y\_true, y\_pred)

**5. Best Model Selection**

* The **VGG16-based model** outperformed the custom CNN in accuracy and stability.
* The addition of **Dropout (0.5)** in one variation helped prevent overfitting.
* The pre-trained **VGG16 model with 256 dense neurons** showed the best performance due to its richer feature extraction.

**6. Insights Gained**

* **Preprocessing efficiency matters**: Storing and batching preprocessed data significantly improved memory management.
* **Transfer learning improves accuracy**: Using a pre-trained model (VGG16) helped leverage rich feature representations from ImageNet.
* **Dropout helps**: A dropout rate of 0.5 improved generalization by reducing overfitting.

**7. Visualizations & MLflow Tracking**

* Model accuracy and loss were logged in MLflow.
* Classification report and confusion matrix were generated for further analysis.
* The trained model was stored and version-controlled using MLflow for reproducibility.

**Conclusion**: The best performing model was the **VGG16-based architecture with a 256-neuron dense layer and dropout (0.5)**, demonstrating superior accuracy and generalization capability.