AI ASSIGNMENT 3

Sudheshna Dharmapuri (Name)

20BCR7029 (Registration Number)

VIT AP (University)

sudheshna.20bcr7029@vitap.ac.in (Email)

Build a CNN model for Bird species Bird species classification is the process of using machine learning and computer vision techniques to identify and categorize different species of birds based on their visual characteristics. By analyzing images of birds, models can extract features and patterns to accurately classify bird species. This classification is vital for ecological research, wildlife monitoring, and conservation efforts. Advancements in deep learning and the availability of large annotated datasets have improved the accuracy of bird species classification models. Challenges include variations in lighting, pose, and background clutter. Ongoing research focuses on methods like transfer learning and data augmentation to enhance classification performance and contribute to avian biodiversity understanding and conservation. Dataset Link: https://www.kaggle.com/datasets/akash2907/bird-species-classification

---

```
import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator


# Step 1: Preprocess the dataset

# Set the path to the dataset directory

dataset_dir = '/path/to/dataset'


# Define the input image size for the CNN model

image_size = (224, 224)


# Split the dataset into training and testing sets

train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

# Step 3: Design and build the CNN model architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

# Step 4: Train the CNN model
model.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])

model.fit(train_generator,
          validation_data=validation_generator,
          epochs=10)

# Step 5: Evaluate the CNN model
test_generator = test_datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

model.evaluate(test_generator)

# Step 6: Fine-tune the CNN model (optional)
# You can apply techniques like transfer learning or data augmentation here

# Step 7: Iterate and optimize
```

```
# Load a pre-trained model (e.g., ResNet50) without the top classification layer
base_model = tf.keras.applications.ResNet50(include_top=False, weights='imagenet',
input_shape=(224, 224, 3))

# Freeze the layers of the pre-trained model
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification layers on top of the pre-trained model
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dense(256, activation='relu')(x)
output = tf.keras.layers.Dense(num_classes, activation='softmax')(x)

# Create the fine-tuned model
model = tf.keras.models.Model(inputs=base_model.input, outputs=output)

# Compile and train the fine-tuned model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_generator,
          validation_data=validation_generator,
          epochs=10)
```