```
import os
import json
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
```

```
# Paths to the dataset
img_dir = '/content/drive/MyDrive/Mini-Project-Sem-4/Datasets/BOUNDING_BOXES_ALLINONE/agri_data/data'
classes_file = '/content/drive/MyDrive/Mini-Project-Sem-4/Datasets/BOUNDING_BOXES_ALLINONE/classes.txt'

# Loading class names
with open(classes_file, 'r') as f:
    classes = f.read().splitlines()
    classes = {i: cls for i, cls in enumerate(classes)}

# Loading images and annotations
def load_data(img_dir):
    images = []
    labels = []
    n = 0
    for img_name in os.listdir(img_dir):
        if img_name.endswith('.jpeg') or img_name.endswith('.png'):
            img_path = os.path.join(img_dir, img_name)
            ann_path = os.path.join(img_dir, os.path.splitext(img_name)[0] + '.txt')

            # Load image
            img = Image.open(img_path)
            img = img.resize((224, 224))
            images.append(np.array(img) / 255.0)  # Normalize to [0, 1]
            n += 1

            # Load annotation and set label (1 for weed, 0 for crop)
            with open(ann_path, 'r') as f:
                anns = f.read().strip().split('\n')
                is_weed = False
                for ann in anns:
                    cls_id = int(ann.split(' ')[0])
                    if classes[cls_id] == 'weed':
                        is_weed = True
                        break
                labels.append(1 if is_weed else 0)
    print(n, "images found")
    return np.array(images), np.array(labels)

# Loading the dataset
images, labels = load_data(img_dir)
```

```
1300 images found
```

```
X_train, X_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, random_state=42)
```

```python
# Data augmentation for training data
train_datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

# Data augmentation for validation data (only rescaling)
val_datagen = ImageDataGenerator()

# Create data generators
train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
val_generator = val_datagen.flow(X_val, y_val, batch_size=32)
```

Model Training

```python
# Load VGG16 model pre-trained on ImageNet, excluding top layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Adding custom top layers for binary classification
x = base_model.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# Create the optimizer without weight_decay or other unsupported parameters
optimizer = Adam(learning_rate=0.0001)

model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

```python
# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('weed_detection_VGG-16_.keras', save_best_only=True, monitor='val_loss')

# Train the model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=15,
    callbacks=[early_stopping, model_checkpoint]
)

# Save the model
model.save('weed_detection_VGG-16_.h5')
```

```
Epoch 1/15
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
33/33 ━━━━━━━━━━━━━━━━━━━━ 117s 2s/step - accuracy: 0.6275 - loss: 0.7944 - val_accuracy: 0.9462 - val_loss: 0.2026
Epoch 2/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 28s 722ms/step - accuracy: 0.9324 - loss: 0.2093 - val_accuracy: 0.9385 - val_loss: 0.1831
Epoch 3/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 33s 903ms/step - accuracy: 0.9524 - loss: 0.1679 - val_accuracy: 0.9577 - val_loss: 0.1601
Epoch 4/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 27s 769ms/step - accuracy: 0.9561 - loss: 0.1533 - val_accuracy: 0.9692 - val_loss: 0.1541
Epoch 5/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 46s 920ms/step - accuracy: 0.9289 - loss: 0.2014 - val_accuracy: 0.9577 - val_loss: 0.1454
Epoch 6/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 30s 575ms/step - accuracy: 0.9682 - loss: 0.1505 - val_accuracy: 0.9577 - val_loss: 0.1788
Epoch 7/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 41s 570ms/step - accuracy: 0.9384 - loss: 0.1985 - val_accuracy: 0.9615 - val_loss: 0.1546
Epoch 8/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 41s 575ms/step - accuracy: 0.9655 - loss: 0.1400 - val_accuracy: 0.9692 - val_loss: 0.1578
Epoch 9/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 21s 571ms/step - accuracy: 0.9764 - loss: 0.1182 - val_accuracy: 0.9615 - val_loss: 0.1550
Epoch 10/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 50s 850ms/step - accuracy: 0.9567 - loss: 0.1561 - val_accuracy: 0.9692 - val_loss: 0.1341
Epoch 11/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 21s 585ms/step - accuracy: 0.9716 - loss: 0.1407 - val_accuracy: 0.9692 - val_loss: 0.1635
```

```
Epoch 12/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 22s 576ms/step - accuracy: 0.9649 - loss: 0.1415 - val_accuracy: 0.9500 - val_loss: 0.1723
Epoch 13/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 21s 566ms/step - accuracy: 0.9733 - loss: 0.1249 - val_accuracy: 0.9692 - val_loss: 0.1468
Epoch 14/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 23s 555ms/step - accuracy: 0.9757 - loss: 0.1149 - val_accuracy: 0.9654 - val_loss: 0.1443
Epoch 15/15
33/33 ━━━━━━━━━━━━━━━━━━━━ 39s 548ms/step - accuracy: 0.9674 - loss: 0.1267 - val_accuracy: 0.9654 - val_loss: 0.1696
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is
```

Evaluation Metrics

```python
from sklearn.metrics import confusion_matrix, f1_score, accuracy_score, precision_score, recall_score

# Load the best model
model = tf.keras.models.load_model('weed_detection_VGG-16_.h5')
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
```

```python
# Confusion Matrix
cm = confusion_matrix(y_val, np.argmax(model.predict(X_val), axis=-1))
print("Confusion Matrix:\n",cm)

# F1 Score
f1 = f1_score(y_val, np.argmax(model.predict(X_val), axis=-1), average='weighted')
print("F1 Score:", f1)

# Accuracy
accuracy = accuracy_score(y_val, np.argmax(model.predict(X_val), axis=-1))
print("Accuracy:", accuracy)

# Precision
precision = precision_score(y_val, np.argmax(model.predict(X_val), axis=-1), average='weighted')
print("Precision:", precision)

# Recall
recall = recall_score(y_val, np.argmax(model.predict(X_val), axis=-1), average='weighted')
print("Recall:", recall)
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 2s 232ms/step
Confusion Matrix:
 [[130   0]
 [130   0]]
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 126ms/step
F1 Score: 0.3333333333333333
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 126ms/step
Accuracy: 0.5
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 124ms/step
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, msg_start, len(result))
Precision: 0.25
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 125ms/step
Recall: 0.5
```

```python
train_loss, train_acc = model.evaluate(train_generator)
val_loss, val_acc = model.evaluate(val_generator)

print(f"Training Loss: {train_loss:.4f}, Training Accuracy: {train_acc:.4f}")
print(f"Validation Loss: {val_loss:.4f}, Validation Accuracy: {val_acc:.4f}")
```

```
33/33 ━━━━━━━━━━━━━━━━━━━━ 24s 715ms/step - accuracy: 0.9623 - loss: 0.1437
9/9 ━━━━━━━━━━━━━━━━━━━━ 2s 167ms/step - accuracy: 0.9765 - loss: 0.1040
Training Loss: 0.1313, Training Accuracy: 0.9644
Validation Loss: 0.1341, Validation Accuracy: 0.9692
```

```python
import numpy as np
from PIL import Image
import tensorflow as tf

# Function to load and preprocess a single image
def load_and_preprocess_image(img_path):
    img = Image.open(img_path)
    img = img.resize((224, 224))
    img = np.array(img) / 255.0  # Normalize to [0, 1]
    img = np.expand_dims(img, axis=0)  # Add batch dimension
    return img

# Function to make a prediction on a single image
def predict_image(model, img_path):
    img = load_and_preprocess_image(img_path)
    display(Image.fromarray((img[0] * 255).astype(np.uint8)))
    prediction = model.predict(img)
    return np.argmax(prediction[0])

# An example image
example_img_path = '/content/drive/MyDrive/Mini-Project-Sem-4/Datasets/BOUNDING_BOXES_ALLINONE/agri_data/data/agri_0_177.jpeg'

# Load the model
model = tf.keras.models.load_model('weed_detection_VGG-16_.h5')

# Make a prediction
prediction = predict_image(model, example_img_path)


# Output the prediction
if prediction > 0.5:
    print(f"The image is predicted to be a weed with a probability of {prediction:.4f}")
else:
    print(f"The image is predicted to be a crop with a probability of {1 - prediction:.4f}")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until y
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 548ms/step
The image is predicted to be a crop with a probability of 1.0000
```