Transfer Learning-Based Classification of Poultry Diseases

requirements.txt

```
tensorflow
keras
matplotlib
numpy
pandas
opencv-python
scikit-learn
```

train.py

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

# Paths
train_dir = 'dataset/train'
val_dir = 'dataset/val'

# Image preprocessing
img_size = 224
batch_size = 32

train_datagen = ImageDataGenerator(rescale=1./255, horizontal_flip=True, rotation_range=20)
val_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(train_dir, target_size=(img_size, img_size), batch_size=batch_size, class_mode='categorical')
val_gen = val_datagen.flow_from_directory(val_dir, target_size=(img_size, img_size), batch_size=batch_size, class_mode='categorical')

# Load pretrained model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(img_size, img_size, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(train_gen.num_classes, activation='softmax')(x)

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

# Freeze base layers
for layer in base_model.layers:
    layer.trainable = False
```

```
# Compile
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# Train
model.fit(train_gen, epochs=10, validation_data=val_gen)

# Save model
model.save('models/poultry_classifier.h5')
```

predict.py

```
import tensorflow as tf
import numpy as np
import cv2

model = tf.keras.models.load_model('models/poultry_classifier.h5')
class_names = ['Newcastle', 'Marek', 'Healthy']  # Update based on your dataset

def predict_image(img_path):
    img = cv2.imread(img_path)
    img = cv2.resize(img, (224, 224))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    predicted_class = class_names[np.argmax(prediction)]
    return predicted_class

# Example
print(predict_image('test_image.jpg'))
```

main.py

```
from predict import predict_image

img_path = input("Enter path to poultry image: ")
result = predict_image(img_path)
print(f"Predicted Disease: {result}")
```