Transfer Learning

VGG 16

VGG16 was selected as the pre-trained model for transfer learning because of its strong performance in image classification tasks and its ability to generalize well to new datasets, including fish classification. VGG16 remains a popular choice for transfer learning due to its strong Low-Level Feature Extraction and Simple and Reliable Architecture

```python
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
```

✓ 2m 8.2s Python

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 7s 0us/step
```

```python
# IMPORTS

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


# DATA DIRECTORIES

train_dir = r"C:\Users\96650\Downloads\fishProject\fishdataset\train\fish_image"
val_dir = r"C:\Users\96650\Downloads\fishProject\fishdataset\val"
test_dir = r"C:\Users\96650\Downloads\fishProject\fishdataset\test"


# Preparing data using DATA GENERATORS

IMG_SIZE = (224, 224)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.2
)
```

✓ 1.6s Python

```python
val_test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)

num_classes = len(train_generator.class_indices)
print("Number of classes:", num_classes)
```

✓ 1.6s Python

```
Found 26593 images belonging to 23 classes.
Found 408 images belonging to 23 classes.
Found 369 images belonging to 23 classes.
Number of classes: 23
```

```python
# LOAD VGG16 WITHOUT TOP LAYERS

base_model = VGG16(weights="imagenet", include_top=False, input_shape=(224,224,3))

# Freeze early layers (keep image features)
for layer in base_model.layers[:15]:
    layer.trainable = False
```

✓ 1.4s

```python
# ADD CUSTOM CLASSIFIER

x = Flatten()(base_model.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)
```

✓ 0.4s

```python
# COMPILE MODEL

model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

✓ 0.3s

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_3 (Flatten)	(None, 25088)	0
dense_6 (Dense)	(None, 512)	12,845,568
dropout_3 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 23)	11,799

Total params: 27,572,055 (105.18 MB)

Trainable params: 19,936,791 (76.05 MB)

Non-trainable params: 7,635,264 (29.13 MB)

```python
# TRAIN MODEL

history = model.fit(
    train_generator,
    epochs=15,
    validation_data=val_generator
)
```

✓ 4410m 25.1s

```
Epoch 1/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3763s 5s/step - accuracy: 0.9154 - loss: 0.3127 - val_accuracy: 0.6005 - val_loss: 1.5931
Epoch 2/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3813s 5s/step - accuracy: 0.9680 - loss: 0.1154 - val_accuracy: 0.6373 - val_loss: 1.4794
Epoch 3/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 5252s 6s/step - accuracy: 0.9777 - loss: 0.0809 - val_accuracy: 0.7132 - val_loss: 1.0150
Epoch 4/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 111458s 134s/step - accuracy: 0.9804 - loss: 0.0729 - val_accuracy: 0.7598 - val_loss: 1.1014
Epoch 5/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3437s 4s/step - accuracy: 0.9833 - loss: 0.0569 - val_accuracy: 0.7304 - val_loss: 1.0679
Epoch 6/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 14862s 18s/step - accuracy: 0.9861 - loss: 0.0490 - val_accuracy: 0.7255 - val_loss: 1.2328
Epoch 7/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3251s 4s/step - accuracy: 0.9855 - loss: 0.0498 - val_accuracy: 0.7672 - val_loss: 1.0173
Epoch 8/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 20515s 25s/step - accuracy: 0.9890 - loss: 0.0388 - val_accuracy: 0.7868 - val_loss: 1.0013
Epoch 9/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3455s 4s/step - accuracy: 0.9888 - loss: 0.0412 - val_accuracy: 0.7990 - val_loss: 0.9637
Epoch 10/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 31133s 37s/step - accuracy: 0.9901 - loss: 0.0362 - val_accuracy: 0.8456 - val_loss: 0.7910
Epoch 11/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 3617s 4s/step - accuracy: 0.9918 - loss: 0.0293 - val_accuracy: 0.8064 - val_loss: 1.0049
Epoch 12/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 4899s 6s/step - accuracy: 0.9917 - loss: 0.0293 - val_accuracy: 0.7696 - val_loss: 1.2374
Epoch 13/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 21003s 25s/step - accuracy: 0.9915 - loss: 0.0305 - val_accuracy: 0.8309 - val_loss: 0.7897
Epoch 14/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 21047s 25s/step - accuracy: 0.9920 - loss: 0.0296 - val_accuracy: 0.8113 - val_loss: 1.1837
Epoch 15/15
832/832 ━━━━━━━━━━━━━━━━━━━━ 13116s 16s/step - accuracy: 0.9926 - loss: 0.0246 - val_accuracy: 0.7721 - val_loss: 1.3153
```

```python
# EVALUATE ON TEST SET

test_loss, test_acc = model.evaluate(test_generator)
print("\nTest Accuracy:", test_acc)
```

✓ 35.8s

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 35s 3s/step - accuracy: 0.8672 - loss: 0.5473

Test Accuracy: 0.8672086596488953
```

```python
# PREDICTIONS FOR METRICS

y_pred = model.predict(test_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_generator.classes
class_labels = list(test_generator.class_indices.keys())


# CLASSIFICATION REPORT

print("\nClassification Report:")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))


# CONFUSION MATRIX

cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(18, 18))
sns.heatmap(cm, cmap="Blues", annot=False)
plt.title("Confusion Matrix for 23-Class Fish Classification")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```python
# OPTIONAL UNFREEZE DEEP LAYERS FOR FINE-TUNING

fine_tune = False

if fine_tune:
    for layer in base_model.layers[15:]:
        layer.trainable = True

    model.compile(
        optimizer=Adam(learning_rate=1e-5),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    history_ft = model.fit(
        train_generator,
        epochs=5,
        validation_data=val_generator
    )
```

✓ 51.9s

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 50s 4s/step

Classification Report:
C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\skl
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\skl
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\skl
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

:Classification Report

C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\sklearn\metrics_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use .`zero_division` parameter to control this behavior

} is", result.shape[0])()warn_prf(average, modifier, f"{metric.capitalize_

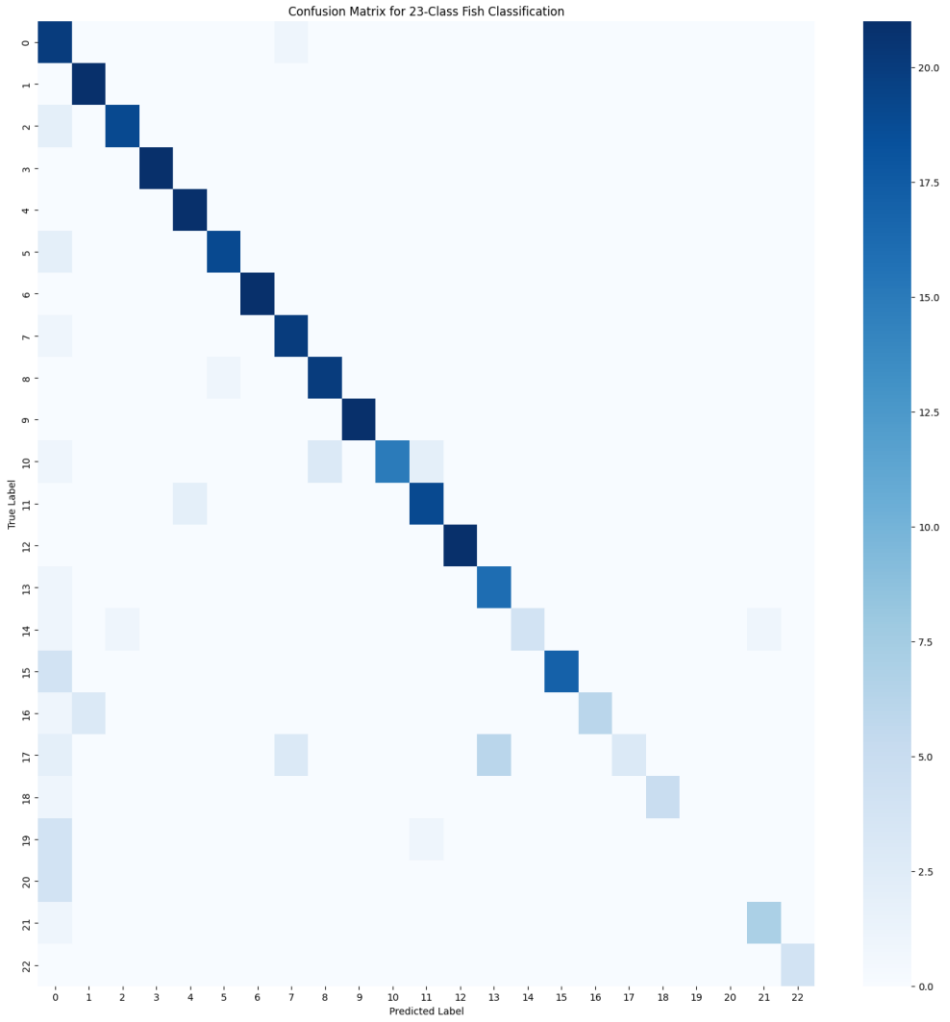
C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\sklearn\metrics_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use .`zero_division` parameter to control this behavior

} is", result.shape[0])()warn_prf(average, modifier, f"{metric.capitalize_

C:\Users\96650\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\site-packages\sklearn\metrics_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use .`zero_division` parameter to control this behavior

} is", result.shape[0])()warn_prf(average, modifier, f"{metric.capitalize_

	precision	recall	f1-score	support
fish_01	0.44	0.95	0.61	21
fish_02	0.88	1.00	0.93	21
fish_03	0.95	0.90	0.93	21
fish_04	1.00	1.00	1.00	21
fish_05	0.91	1.00	0.95	21
fish_06	0.95	0.90	0.93	21
fish_07	1.00	1.00	1.00	21
fish_08	0.83	0.95	0.89	21
fish_09	0.87	0.95	0.91	21
fish_10	1.00	1.00	1.00	21
fish_11	1.00	0.71	0.83	21
fish_12	0.86	0.90	0.88	21
fish_13	1.00	1.00	1.00	21
fish_14	0.73	0.94	0.82	17
fish_15	1.00	0.57	0.73	7
fish_16	1.00	0.81	0.89	21
fish_17	1.00	0.60	0.75	10
fish_18	1.00	0.21	0.35	14
fish_19	1.00	0.83	0.91	6
fish_20	0.00	0.00	0.00	5
fish_21	0.00	0.00	0.00	4
fish_22	0.88	0.88	0.88	8
fish_23	1.00	1.00	1.00	4
accuracy			0.87	369
macro avg	0.84	0.79	0.79	369
weighted avg	0.89	0.87	0.86	369


Confusion Matrix for 23-Class Fish Classification
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0