CCS 229 – Intelligent Systems

Fish Species Classification using VGG16

A Final Project

Presented to **Louie Cervantes**

College of Information and Communications Technology

West Visayas State University – La Paz, Iloilo City

In Fulfillment of the Requirements for

Intelligent Systems (CCS 229)

by

Gumacal, Rhazel Jay

BSCS 3A - AI

May 2025

Table of Contents

I. Project Overview & Objectives

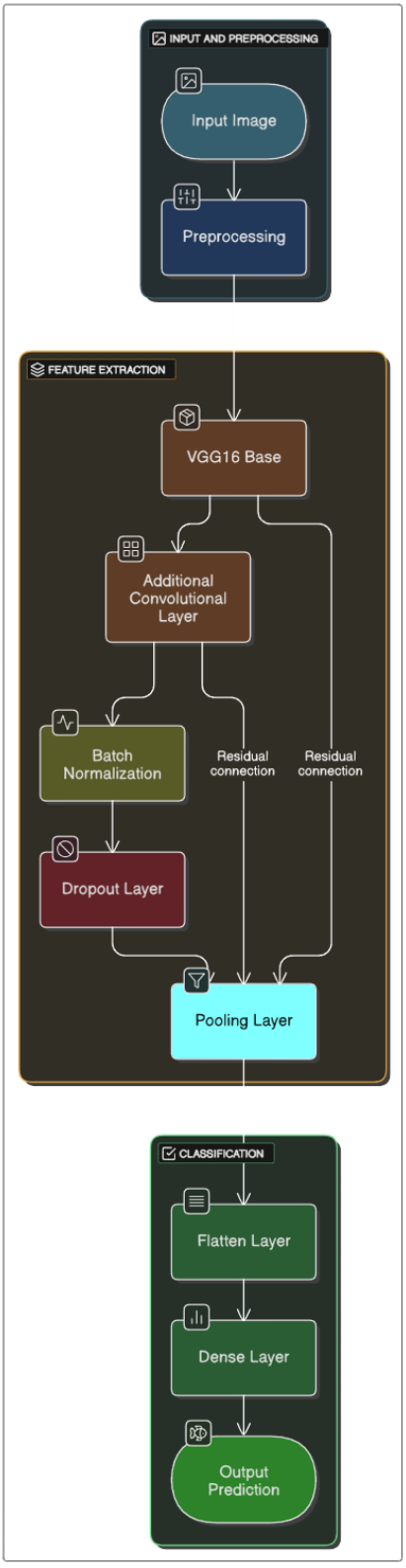
This project is focused on the classification of various fish species using a deep learning approach built upon the VGG16 architecture. VGG16 is a well-established convolutional neural network (CNN) model known for its effectiveness in image recognition tasks.

By leveraging the learned features of VGG16, such as edge detection, texture recognition, and shape identification, the model can more effectively extract meaningful patterns from the fish images, even with a limited dataset. The final layers of the VGG16 model are customized and retrained to suit the specific requirements of fish species classification. This approach not only boosts accuracy but also significantly reduces the computational cost and time required compared to training a model from scratch.

The primary objectives are:

1. To develop an image classifier capable of identifying 31 distinct fish species.
2. To utilize a VGG16 model and adapt it for the fish classification task.
3. To evaluate model performance using accuracy and loss metrics on training, validation, and test sets.
4. To deploy a prediction function capable of classifying unseen fish images.

II. Model Architecture Diagram



III. Dataset Description

The dataset used in this project is organized under the root directory ../fish-dataset/FishImgDataset/, which is structured into three main subsets:

- **train/** – Contains the training images used to train the model.
- **val/** – Contains the validation images used to fine-tune model parameters and monitor for overfitting during training.
- **test/** – Contains the testing images used for final model evaluation and performance assessment.

Each of these directories includes **31 subdirectories**, with each subdirectory representing a distinct **fish species**. The subdirectory names serve as the **class labels**, and they contain images of the respective fish species.

Examples of Fish Classes:

- Tilapia
- Catfish
- Snakehead
- Janitor Fish
- Gold Fish
- Goby
- *(and 25 other species)*

Image Specifications:

- All images have been **resized to 224×224 pixels**, conforming to the input requirements of the VGG16 model.
- Images are stored in standard formats (e.g., JPG or PNG), and are uniformly distributed across the train, validation, and test sets to maintain balanced class representation.

Preprocessing:

The preprocessing for the dataset uses the preprocess_input function from the VGG16 model, applied through a Lambda layer. This function adjusts the input image data to match the format that VGG16 expects. Specifically, it converts the RGB image from the [0, 255] range to the format used during VGG16's original training: zero-centered with respect to the ImageNet dataset, by subtracting the mean RGB values. This ensures that the input data is properly normalized, which helps improve the model's performance and convergence during training.

IV. Training Logs & Charts

Training Configuration:

The fish classification model was trained using the **Adam optimizer** and the **categorical crossentropy** loss function, which is ideal for multi-class classification tasks. Training was carried out over **10 epochs** with a **batch size of 32**, enabling efficient learning while managing memory usage. The Adam optimizer's adaptive learning rate and momentum properties helped accelerate convergence and stabilize training.

To enhance the model's generalization and avoid overfitting, **early stopping** was implemented. This technique monitored the validation loss and automatically halted training when no further improvement was observed, ensuring the model did not over-train on the dataset. These training strategies collectively contributed to a more accurate and reliable fish species classification system.

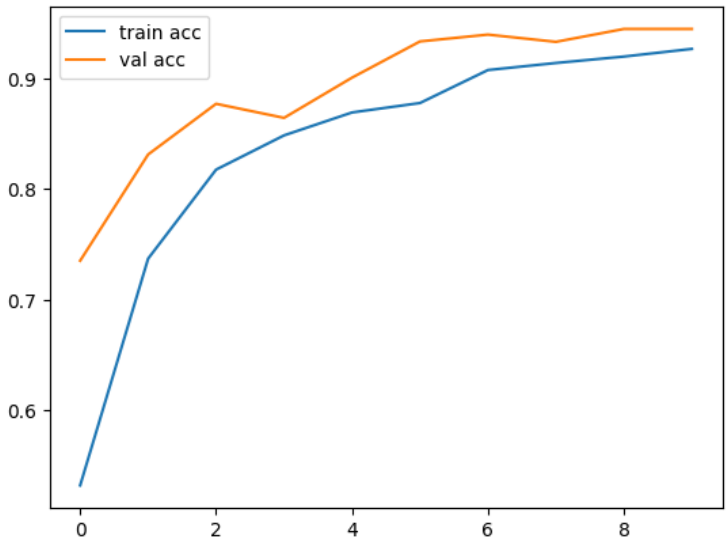
```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
```

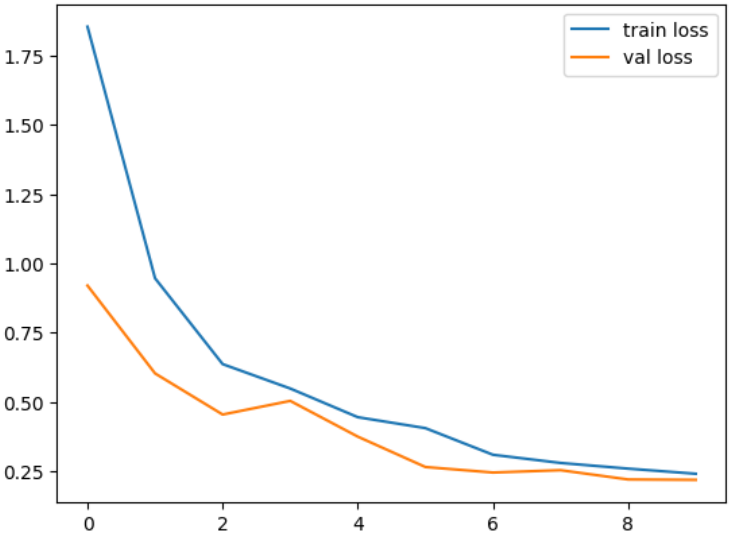
```
# Model Training
FishModel = model.fit(
  training_set,
  validation_data=val_set,
  epochs=10,
  steps_per_epoch=len(training_set),
  validation_steps=len(val_set),
  callbacks=[early_stop, checkpoint]
)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

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

training_set = train_datagen.flow_from_directory(train_dir,
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
val_set= val_datagen.flow_from_directory(val_dir,
                                         target_size = (224, 224),
                                         batch_size = 32,
                                         class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_dir,
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

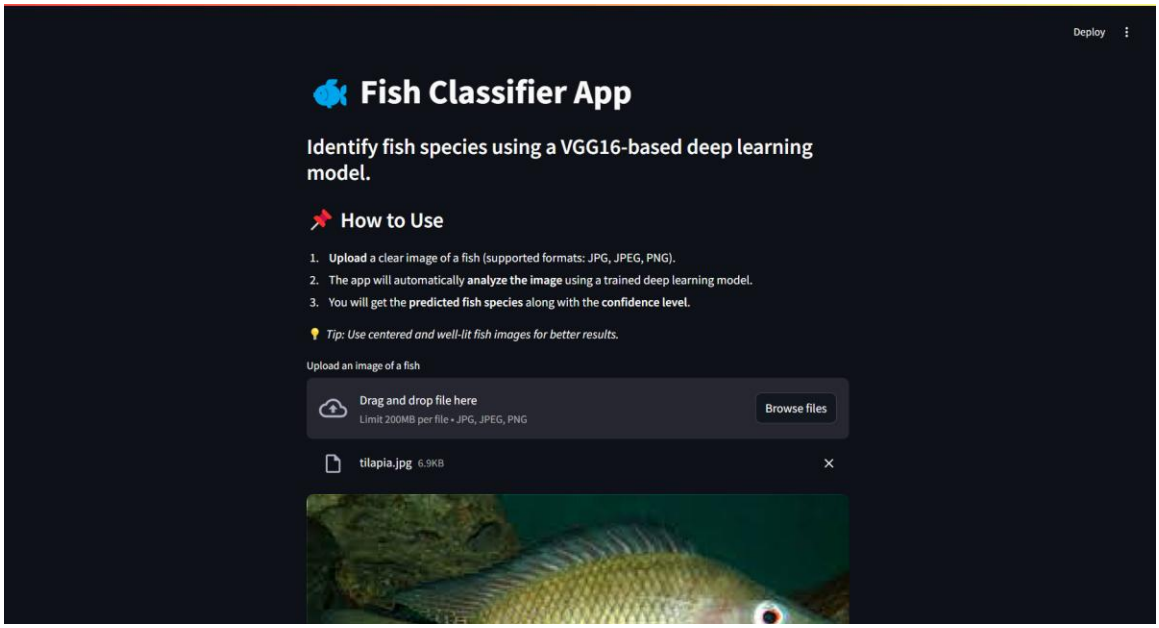
Training & Validation Accuracy:

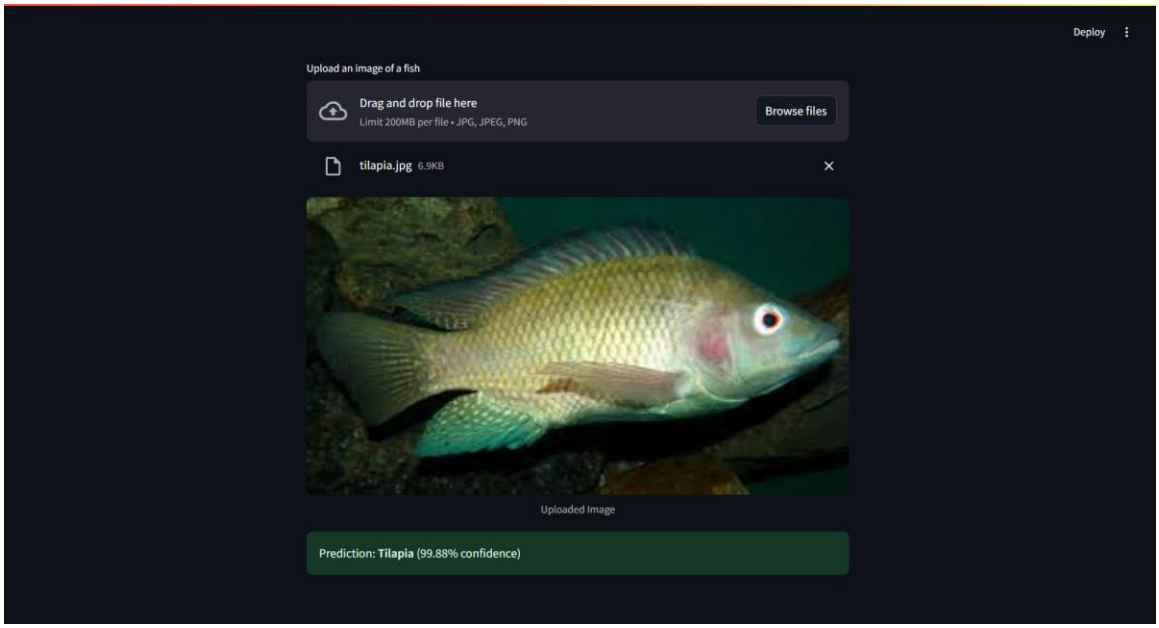


Training & Validation Loss:



V. Sample Predictions





VI. Limitations & Improvements

Limitations:

1. The model uses VGG16, which is computationally heavy.

2. Limited to fish types included in the training dataset.

3. Model performance could degrade on low-quality or ambiguous images.

Future Improvements:

1. Fine-tuning top VGG16 layers for better performance.

2. Using more recent architectures like EfficientNet or MobileNetV3.

3. Incorporating object detection to locate fish before classification.

4. Expanding dataset diversity and size.

VII. References

1. Simonyan, K., & Zisserman, A. (2015). *Very deep convolutional networks for large-scale image recognition*. arXiv. https://arxiv.org/abs/1409.1556
2. TensorFlow. (n.d.). *TensorFlow/Keras API documentation*. TensorFlow. https://www.tensorflow.org/api_docs
3. Lampa, M. D. (n.d.). *Fish dataset*. Kaggle. https://www.kaggle.com/datasets/markdaniellampa/fish-dataset
4. Keras. (n.d.). *VGG16 – Keras applications*. Keras. https://keras.io/api/applications/vgg/