Multiclass Fish Image Classification

Project Overview

This repository hosts a comprehensive deep learning project dedicated to the multiclass classification of fish images into 11 distinct species categories. The project explores both training a Custom Convolutional Neural Network (CNN) from scratch and leveraging transfer learning with five powerful pre-trained models: VGG16, ResNet50, MobileNetV2, InceptionV3, and EfficientNetB0.

Beyond model development and rigorous evaluation, this project culminates in a Flask-based web application that enables real-time predictions on user-uploaded fish images, offering a practical demonstration of model deployment.

Key Features & Technologies

This project provides hands-on experience and demonstrates proficiency in:

Deep Learning Fundamentals: Training and fine-tuning neural networks for image classification.

Python Programming: Core scripting and application logic.

TensorFlow & Keras: Building, configuring, and managing deep learning models.

Flask for Model Deployment: Creating a user-friendly web interface for real-time inference.

Data Preprocessing & Augmentation: Techniques to prepare and enhance image datasets for robust model training.

Transfer Learning: Effectively utilizing pre-trained architectures for domain-specific tasks.

Comprehensive Model Evaluation: Analyzing performance using various metrics (accuracy, precision, recall, F1-score, confusion matrices).

Data Visualization: Plotting training history (accuracy and loss) for insights.

Problem Statement & Business Applications

The core objective of this project is to accurately classify fish images into their respective 11 species categories using advanced deep learning techniques. This involves training and comparing multiple models, saving their best states, and deploying them within an accessible web application.

Potential Business Use Cases:

Fisheries Management: Automated identification of fish species for stock assessment and monitoring.

Aquaculture: Disease detection and growth monitoring through visual inspection.

Retail & Quality Control: Ensuring correct labeling and quality assurance for seafood products.

Environmental Monitoring: Tracking biodiversity and species distribution in aquatic ecosystems.

Educational Tools: Interactive applications for learning about marine life.

Approach & Methodology

The project follows a structured methodology, encompassing data handling, model development, evaluation, and deployment:

Data Preprocessing and Augmentation:

Images are rescaled to a [0, 1] range for optimal model input.

Applied real-time data augmentation (e.g., rotation, zoom, horizontal flipping) using ImageDataGenerator to prevent overfitting and enhance model generalization.

Model Training:

Custom CNN: A Convolutional Neural Network architecture is designed and trained from scratch.

Transfer Learning Models: Pre-trained models (VGG16, ResNet50, MobileNetV2, InceptionV3, EfficientNetB0) are loaded and fine-tuned by unfreezing top layers and adding custom classification heads.

Models with the highest validation accuracy are saved in the .h5 format.

Model Evaluation:

Performance metrics such as accuracy, precision, recall, and F1-score are calculated for each model.

Confusion matrices are generated to understand class-wise performance.

Training history (accuracy and loss curves) are visualized to monitor model learning.

Deployment (Flask Application):

A lightweight Flask web application is developed to host the trained models.

The app supports image uploads (.jpg, .jpeg, .png formats).

It predicts and displays the fish category along with a confidence score.

It also displays evaluation metrics for the currently selected model.

Dataset

Source: A zipped dataset containing fish images categorized into 11 distinct species. Examples include animal fish, animal fish bass, fish sea_food black_sea_sprat, etc.

Processing: Images are loaded and prepared for training and validation using TensorFlow's ImageDataGenerator, which handles resizing, normalization, and augmentation.

Project Deliverables
Trained Models: .h5 files for the Custom CNN, VGG16, ResNet50, MobileNetV2, InceptionV3, and EfficientNetB0 models, representing their best-performing weights.

Flask Application: app.py and associated HTML/CSS templates for the interactive web predictor.

Python Scripts: Separate scripts for model training (train_custom_cnn.py, adapt_vgg16.py, adapt_eff_net.py, etc.), data loading, and utility functions.

Comparison Report: Detailed performance metrics and visualizations in a human-readable format, often presented within Jupyter Notebooks or additional markdown files.

Model Performance Comparison
The table below summarizes the performance of all trained models as of 06:45 PM IST, Saturday, August 09, 2025.

Model

Accuracy

Precision

Recall

F1-score

Remarks

Custom CNN

75.9%

Medium

High

Medium

Lightweight, relatively good for rare classes.

VGG16

80%+

High

High

High

Stable and production-ready.

ResNet50

79%+

High

Medium

High

Deeper architecture, potential for tuning.

MobileNetV2

~78%

Medium

Medium

Medium

Efficient and suitable for mobile deployment.

InceptionV3

80%+

High

High

High

Strong feature extractor, robust performance.

EfficientNetB0

81%+

High

High

High

Best overall trade-off: accuracy & size.

Detailed Classification Report for Custom CNN

Here's the detailed classification report for the Custom CNN, highlighting its per-class performance:

Classification Report for Custom CNN:

	precision	recall	f1-score	support
animal fish	0.99	0.60	0.75	520
animal fish bass	0.05	0.77	0.10	13

fish sea_food black_sea_sprat 0.92 0.86 0.89 298
fish sea_food gilt_head_bream 0.58 0.31 0.40 305
fish sea_food hourse_mackerel 0.67 0.83 0.74 286
fish sea_food red_mullet 0.94 1.00 0.97 291
fish sea_food red_sea_bream 0.74 0.84 0.79 273
fish sea_food sea_bass 0.63 0.49 0.55 327
fish sea_food shrimp 0.73 0.98 0.84 289
fish sea_food striped_red_mullet 0.90 0.60 0.72 303
fish sea_food trout 0.68 0.98 0.80 302

accuracy			0.73	3207
macro avg	0.71	0.75	0.69	3207
weighted avg	0.79	0.73	0.74	3207

Note: The animal fish bass class has very low support (only 13 samples), which significantly impacts its precision and F1-score. This imbalance is a key area for future improvement.

Setup and Usage Guide

Follow these steps to get the project up and running locally:

1. Clone the Repository
   git clone
   cd multi_fish_classifier

2. Install Dependencies
   It's recommended to use a virtual environment.

python -m venv venv
source venv/bin/activate # On Windows, use `venv\Scripts\activate`
pip install flask tensorflow numpy pillow

3. Prepare the Dataset
   Obtain the dataset (zip file containing fish images categorized into 11 species).

Crucially, extract the dataset into a data/ directory within your project root. The structure should resemble:

```
multi_fish_classifier/
├── data/
│ ├── train/
│ │ ├── animal fish/
│ │ ├── animal fish bass/
│ │ └── ... (other species folders)
│ └── val/
│ ├── animal fish/
│ ├── animal fish bass/
│ └── ... (other species folders)
├── app.py
├── adapt_eff_net.py
└── ... (other project files)
```

4. Run Training Scripts (Optional)
   If you wish to train the models from scratch or re-train them, execute the respective Python scripts:

```
python train_custom_cnn.py
python adapt_vgg16.py
python adapt_resnet50.py
python adapt_mobilenetv2.py
python adapt_inceptionv3.py
python adapt_eff_net.py
```

These scripts will save the best-performing models (e.g., efficientnetb0_best.h5) and training history files (.pkl) in the project directory.

5. Run the Flask Application
   Before running, ensure your trained model files (e.g., efficientnetb0_best.h5, custom_cnn_best.h5) are present in the root directory where app.py resides.

```
python app.py
```

6. Access the Web Application
   Open your web browser and navigate to:

http://127.0.0.1:5000

You can select different models from a dropdown.

Upload a .jpg, .jpeg, or .png image of a fish.

The application will display the predicted fish category and its confidence score.

Example Output
Web Interface
The web interface will feature a dropdown to select the model and an area to upload images.

Sample Prediction Output (Custom CNN)
After uploading an image, you'll see a prediction similar to this:

Predicted Fish Category: animal fish
Confidence: 54.78%

Model Evaluation Metrics (Custom CNN)
Below the prediction, model evaluation metrics for the selected model will be displayed:

Validation Accuracy: 0.7592
Validation Loss: 0.6023

Note: For classes with low sample support, like animal fish bass, the confidence scores and reliability of predictions might be lower.

Predicted Fish Category: animal fish bass
Confidence: 45.00%

Notes and Future Improvements
Best Performing Model: EfficientNetB0 demonstrated the highest accuracy (81%+) and is set as the default model in the Flask application due to its excellent balance of performance and efficiency. VGG16 and InceptionV3 also performed very well (80%+ accuracy).

Addressing Data Imbalance: The animal fish bass class suffers from a significant data imbalance (only 13 samples). Future work could focus on:

Oversampling: Techniques like SMOTE or simply duplicating existing samples.

Augmentation Strategies: More aggressive augmentation specifically for underrepresented classes.

Weighted Loss Functions: Adjusting the loss function to give more weight to minority classes during training.

Model Optimization: Further hyperparameter tuning for ResNet50 and other models could yield better results.

Performance Monitoring: Implement more advanced logging and monitoring for deployment.

Contributions
Contributions are highly welcome! If you have suggestions, bug reports, or want to add new features, please feel free to:

Open an Issue: Describe the bug or feature request clearly.

Submit a Pull Request:

Fork the repository.

Create a new branch for your changes (git checkout -b feature/my-new-feature).

Make your changes and ensure tests (if any) pass.

Commit your changes (git commit -m 'feat: Add new feature X').

Push to your branch (git push origin feature/my-new-feature).

Open a pull request to the main branch.

Please ensure your code adheres to consistent naming conventions and modular design.

License