Multi-Class Fish Image Classification

# Abstract:

This project focuses on developing a robust deep learning-based solution to classify multiple fish species from images using Convolutional Neural Networks (CNNs) and transfer learning with pre-trained models such as VGG16, ResNet50, MobileNetV2, InceptionV3, and EfficientNetB0. The model with the best accuracy is deployed via Streamlit for real-time fish species prediction.

# Objectives:

* Build a CNN from scratch for fish classification.
* Apply transfer learning using 5 pre-trained models.
* Compare model performances using accuracy, precision, recall, and F1 score.
* Deploy the best model using a Streamlit web interface.

# Technologies & Tools Used:

* Python, TensorFlow/Keras
* Pre-trained models: VGG16, ResNet50, MobileNetV2, InceptionV3, EfficientNetB0
* Jupyter Notebook / VS Code
* CUDA-enabled GPU
* Streamlit for deployment
* Matplotlib/Seaborn for visualization

# Methodology:

## DataPreprocessing Augmentation

* Images are rescaled to a pixel range of [0, 1].
* Extensive augmentation techniques were applied to the training set to enhance model robustness and generalization:
  + **Rotation:** Random rotations up to 20 degrees.
  + **Zoom:** Random zoom in and out up to 20%.
  + **Horizontal Flip:** Random flipping of images horizontally.
  + **Vertical Flip:** Random vertical flipping to simulate different orientations.
  + **Width & Height Shift:** Random shifts in width and height by 10%.
  + **Shear Transformation:** Random shearing up to 15%.
  + **Brightness Adjustment:** Varying brightness randomly in the range [0.8, 1.2].
  + **Channel Shift:** Randomly altering the color channels by a factor of 0.2.
  + **Fill Mode:** Used 'nearest' strategy to fill in newly created pixels during transformations.
* This helps improve generalization and prevent overfitting.
* Targeted augmentation was specifically applied to the “animal fish bass” class to boost the number of training samples and improve the model’s ability to learn underrepresented features.

## Model Building

* A CNN model was built from scratch using a multi-block architecture with Batch Normalization and Dropout regularization. The final layers used GlobalAveragePooling2D() followed by fully connected Dense layers to ensure robust learning and effective classification.
* Five pre-trained models were used for transfer learning to boost performance and reduce training time:
  + **VGG16**
  + **ResNet50**
  + **MobileNetV2**
  + **InceptionV3**
  + **EfficientNetB0**
* All models were fine-tuned specifically for the fish image classification task. Initially, the base layers of each pre-trained model were frozen to retain learned features from the ImageNet dataset. After initial training, selected layers were unfrozen for fine-tuning.
* Special consideration was given to **InceptionV3**, which required input images resized to **299x299** due to its architectural constraints.

**Early stopping** was implemented in all training runs to prevent overfitting and ensure model generalization.

## Model Evaluation

* After training, each model's performance was thoroughly evaluated using the test dataset.
* The evaluation involved multiple performance metrics to understand the strengths and weaknesses of each model:
  + **Accuracy:** Measures the overall percentage of correct predictions.
  + **Precision:** Indicates how many of the predicted positive classes were actually correct.
  + **Recall:** Measures how well the model identifies all relevant instances.
  + **F1 Score:** Harmonic mean of precision and recall, giving a balanced view.
  + **Confusion Matrix:** Provides detailed insight into model predictions versus actual classes.
* Each model’s training history—including loss and accuracy curves—was plotted to monitor learning trends and detect overfitting or underfitting.
* A comparative analysis was conducted across all six models (CNN, VGG16, ResNet50, MobileNetV2, InceptionV3, and EfficientNetB0), and the best-performing model was selected based on the highest test accuracy and balanced precision-recall metrics.
* Visual examples from the predictions, including correctly and incorrectly classified fish species, were also included to better interpret the model behavior in real scenarios.

## Model Saving

Once the models were trained and evaluated, the best-performing model was preserved for future use and deployment. This ensured reproducibility and avoided retraining every time a prediction was needed.

### Model Formats Used:

* .h5 — HDF5 format commonly used for saving Keras models including architecture, weights, and optimizer state.

### Saving the Best Model:

* Only the model with the highest validation accuracy (or F1-score, based on the evaluation criteria) was saved for deployment.

### Benefits:

* **Efficiency:** No need to retrain each time — the model is ready for inference.
* **Portability:** Easily share the saved model file across systems.
* **Integration:** Compatible with deployment frameworks like Streamlit, Flask, or cloud platforms.

# Deployment:

The best-performing model was deployed using **Streamlit**, a Python-based framework for building interactive web applications.

The Streamlit app enables users to upload fish images and get real-time predictions of the fish species.

## Deployment Workflow:

* Load the saved .h5 model using Keras.
* Preprocess the uploaded image to match model input requirements (e.g., resizing, normalization).
* Perform prediction using the loaded model.
* Display the predicted fish species and the model confidence score.

## Extended Deployment:

To ensure scalability, portability, and consistent performance across systems, the Streamlit application was **Dockerized** and deployed on an **AWS EC2 instance**.

* **Dockerization:** The app was containerized with all dependencies, allowing consistent performance across any environment.
* **EC2 Hosting:** The Docker container was deployed on an Amazon EC2 instance, configured to expose the app over a public IP, making it accessible from any browser.

## Benefits:

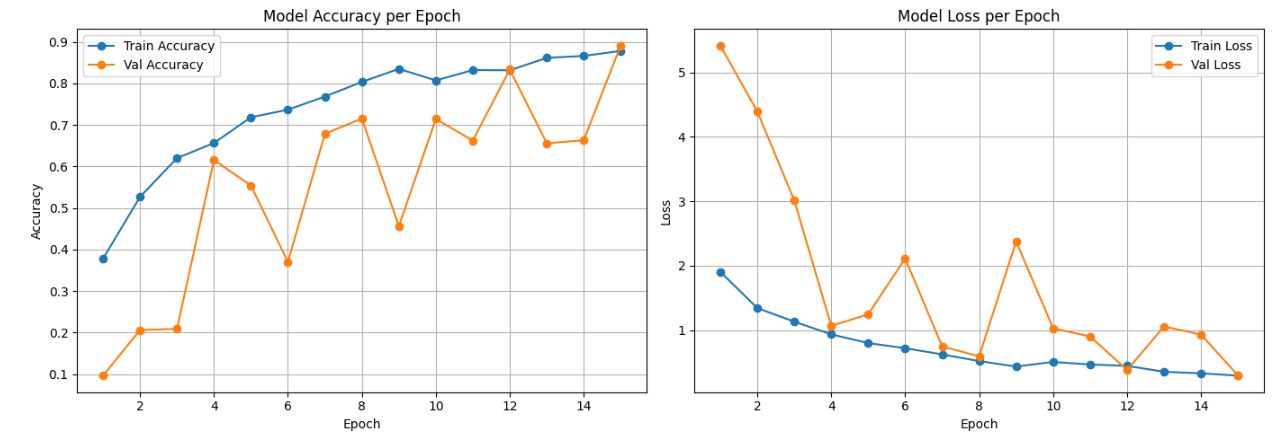
* **User-Friendly Interface:** No coding required for end-users.
* **Lightweight and Fast:** Streamlit ensures quick loading and smooth UI.
* **Portable:** Can be shared via GitHub and hosted on platforms like Streamlit Cloud.

The deployed model allows easy access to predictions, making it suitable for marine research teams, educational tools, or use in the fishing industry.

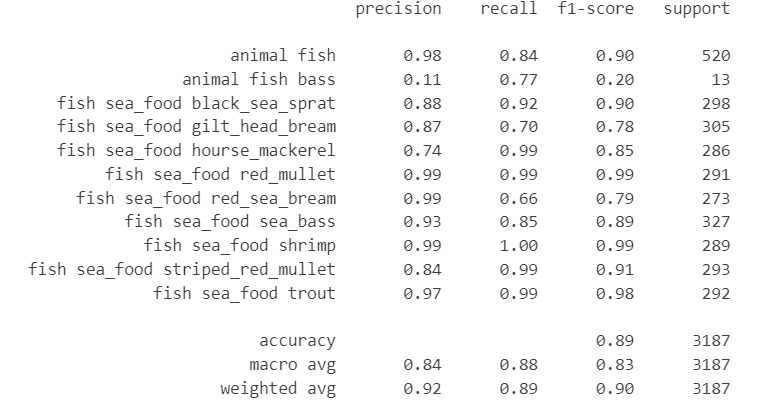
# Results:

## CNN Architecture Built From Scratch

Training & Validation Accuracy and Loss:

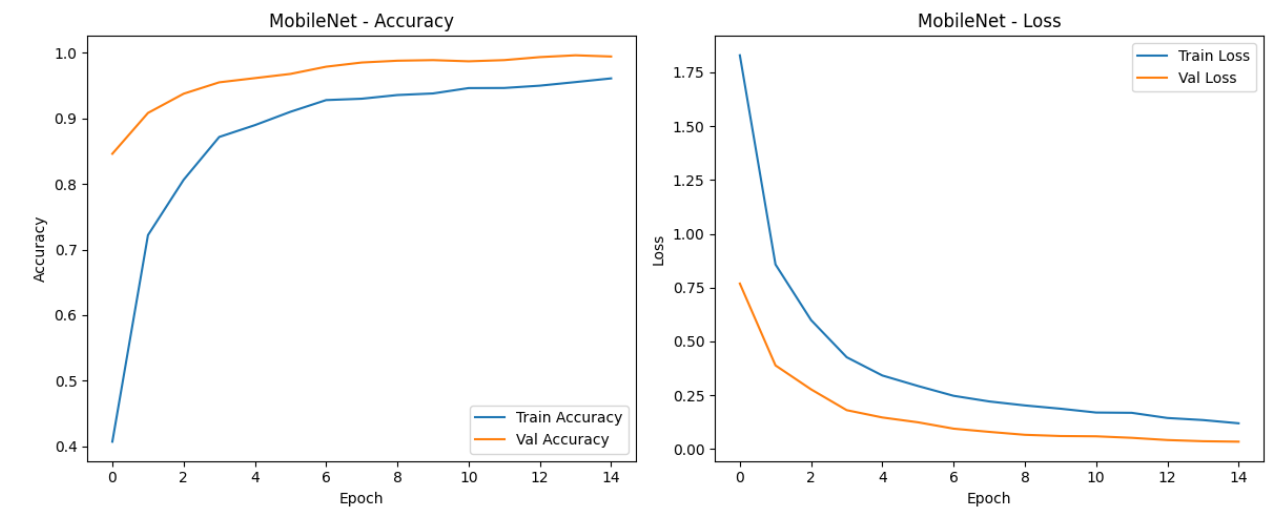


Classification Report:

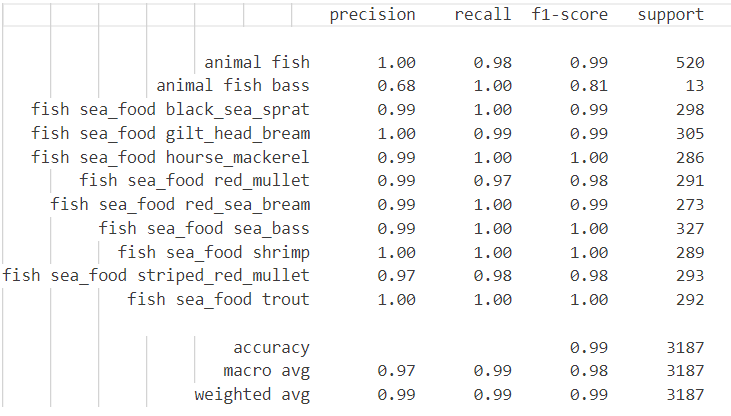


## MOBILNET

Training & Validation Accuracy and Loss:

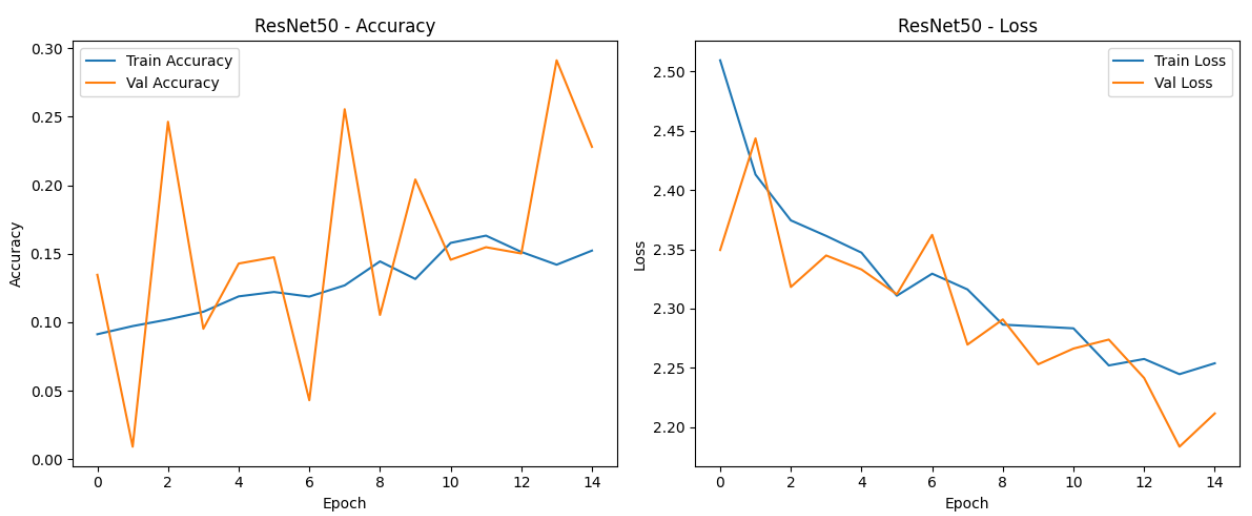


Classification Report:

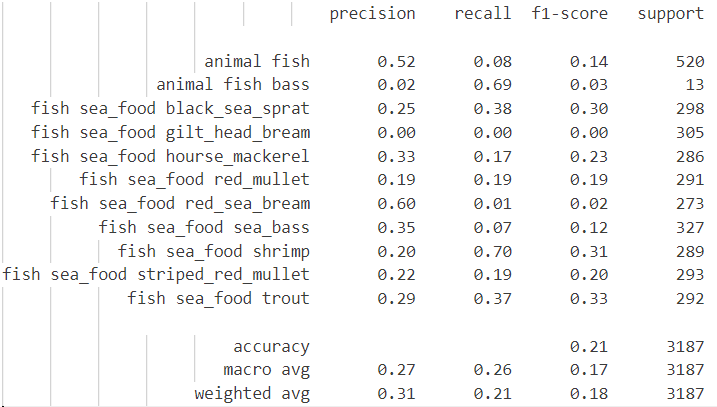


## RESNET50

Training & Validation Accuracy and Loss:

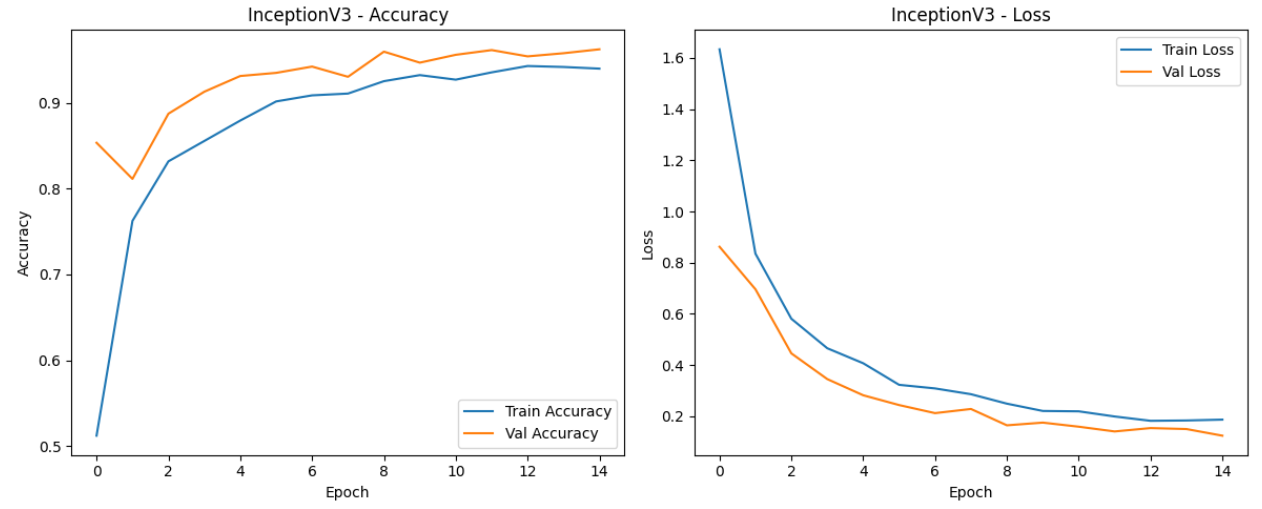


Classification Report:

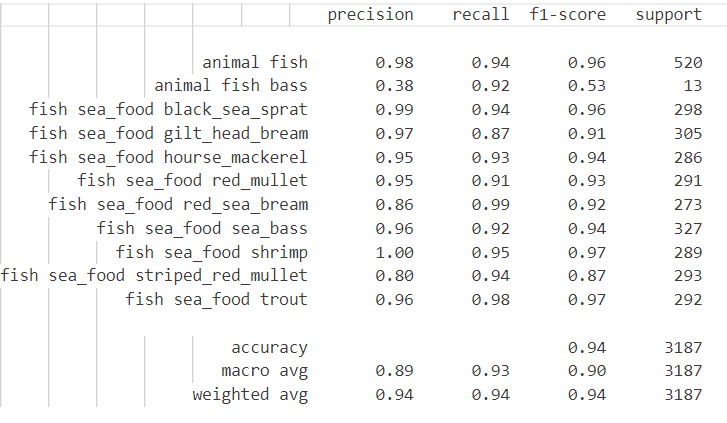


## INCEPTIONV3

Training & Validation Accuracy and Loss:

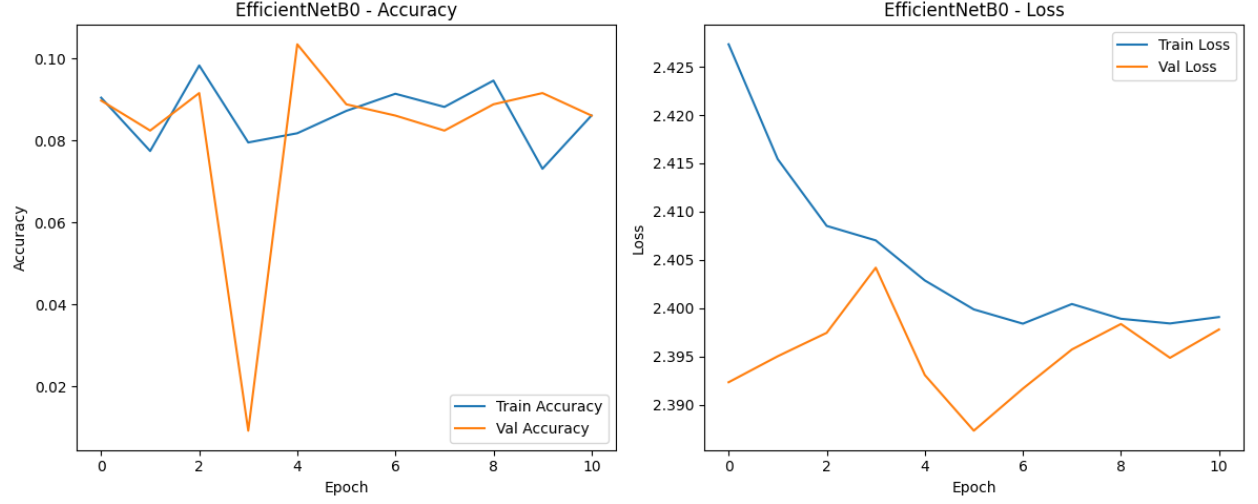


Classification Report:



## EFFICIENTNETB0

Training & Validation Accuracy and Loss:

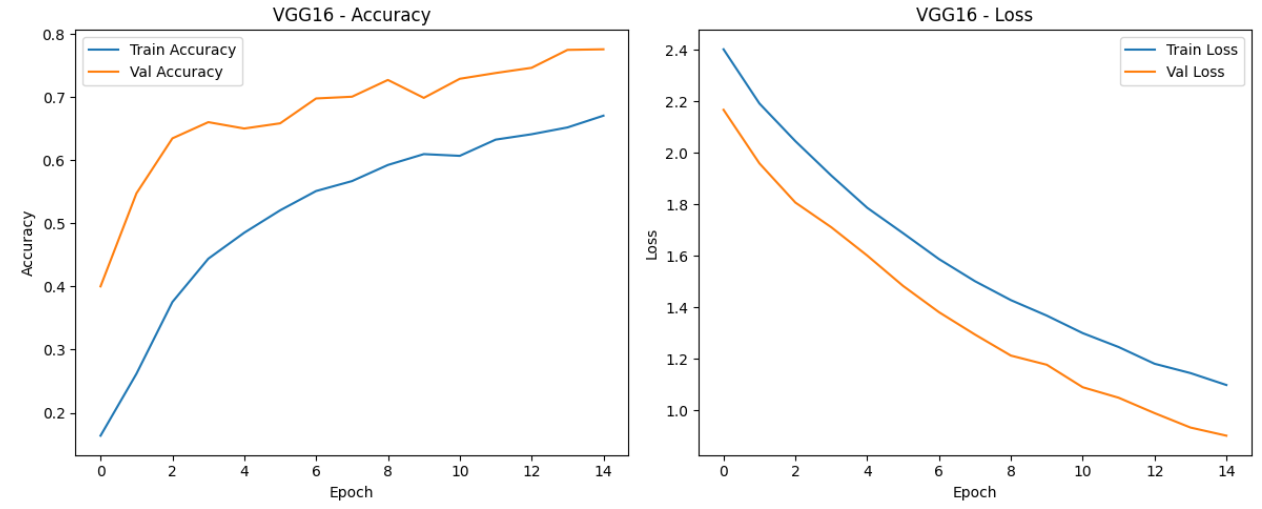


Classification Report:

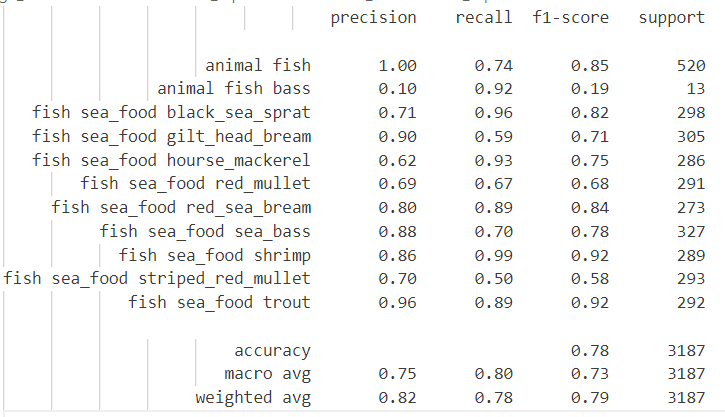


## VGG16

Training & Validation Accuracy and Loss:



Classification Report:



# Conclusion:

## Summary of Achievements

* This project successfully demonstrates the application of deep learning and transfer learning techniques for multiclass fish image classification. By training a CNN model from scratch and fine-tuning five state-of-the-art pre-trained models (VGG16, ResNet50, MobileNetV2, InceptionV3, and EfficientNetB0), we identified the most effective model for this task. The use of data augmentation, early stopping, and fine-tuning enhanced model performance and stability.
* A lightweight, user-friendly Streamlit web application was developed to deploy the best-performing model. This allowed users to upload images and receive instant predictions along with confidence scores, making the tool practical for real-world use.

## Future Scope and Recommendations

* **CI/CD Integration:** Automate deployment workflows using CI/CD pipelines for smoother updates and collaboration.