Fish Species Recognition Using Convolutional Neural Networks

Title Page

Project Name: Fish Species Recognition Using Convolutional Neural Networks

Student Names & IDs:

- Hazem Mostafa Ahmed - 2227328
- Taha Eid Gomaa - 2227338
- Mohamed Ayman - 2227027
- Magdy Abdelkhateeb - 2227157
- Mostafa Ahmed - 2227463

Submission Date: December 14, 2025

Course Name: Neural Networks

Instructor Names: Omar Mourad, Mahmoud Esmat

Introduction

What is the problem?

This project addresses **fish species recognition** as an image classification task. Given an input image of a fish, the goal is to predict its species label from a fixed set of classes. Fish recognition is a fine-grained visual problem because multiple species can share similar shapes, textures, and color patterns. The project implements and compares two approaches:

1. A **Baseline CNN** trained from scratch.
2. A **MobileNet transfer learning model** (pretrained on ImageNet) with a custom classification head.

Both models are trained and evaluated using the same dataset split, preprocessing pipeline, and evaluation metrics to ensure a fair comparison.

Why is it important?

Automated fish recognition can support multiple real-world use cases:

- **Marine biodiversity monitoring**: helping researchers track species distribution.
- **Fisheries management**: supporting sustainability and stock assessment.
- **Fish markets and supply chain**: enabling automated sorting and quality control.
- **Education**: assisting students and practitioners with species identification.

- **Conservation**: tracking vulnerable or overfished species over time. ## Dataset

Source of Dataset

The dataset used in this project is publicly available and can be accessed at: **https://www.kaggle.com/datasets/crowww/a-large-scale-fish-dataset**

The dataset consists of labeled fish images organized by species, where each class represents a distinct fish species. All images share consistent visual properties, such as similar image dimensions, which simplifies preprocessing and model training.

Number of Classes

The dataset includes the following **nine fish species**:

- Black Sea Sprat
- Gilt-Head Bream
- Horse Mackerel
- Red Mullet
- Red Sea Bream
- Sea Bass
- Shrimp
- Striped Red Mullet
- Trout

Number of Images per Class

Original Dataset Size:

The original dataset provided two separate folders: - One folder containing **augmented images** - Another folder containing **non-augmented images** with a very limited number of samples

In the non-augmented folder: - Most classes contained **50 images per class** - The **Trout** class contained only **30 images**, making it significantly underrepresented

Due to the small size of the non-augmented dataset, it was excluded from training.

Augmented Dataset Selection:

To ensure sufficient data for effective model training, only the **augmented dataset folder** was used. After augmentation, each class contained approximately: - **1000 images per class**

This resulted in a balanced dataset across all classes and improved the robustness of the trained models.

Preprocessing Steps

Dataset Split:

The final augmented dataset was split into training, validation, and test sets using the following ratios: - **Training set:** 70% - **Validation set:** 15% - **Test set:** 15%

Data Leakage Handling:

During initial experiments, an overlap between training and test sets was detected (data leakage), caused by duplicate image files appearing across different splits. To resolve this issue, the dataset was re-split carefully, and **unique file naming** was enforced to ensure that no identical images appeared in more than one split. This step was essential to obtain a valid and realistic evaluation of model performance.

Image Preprocessing:

All images are processed consistently before training: - **Resizing** images to a fixed input size (e.g., 128×128) - **Normalization**: pixel values are scaled to the range **[0, 1]** by dividing by 255

This ensures stable gradients and faster convergence.

Methodology

Model Architecture

Baseline CNN (From Scratch):

The baseline model is a simple convolutional neural network trained from scratch. Its purpose is to: - Validate the full training/evaluation pipeline - Provide a reference point for comparison against transfer learning

Typical structure: - Convolution + ReLU + MaxPooling blocks - Fully connected layer(s) - Dropout for regularization - Softmax output for multi-class classification

MobileNet (Transfer Learning):

MobileNet is used as a second model to improve performance and training efficiency: - The MobileNet backbone is loaded with **ImageNet pretrained weights** - The top classification layer is removed (include_top=False) - The backbone is frozen initially (feature extractor mode) - A custom head is added: - Global Average Pooling - Dense layer - Dropout - Softmax output layer with number of classes

This approach benefits from strong pretrained visual features and typically converges faster than training from scratch.

Data Augmentation

To improve generalization and reduce overfitting, data augmentation is applied **only to the training set**. Augmentation operations include:

- Random rotations
- Horizontal flipping
- Shifts (width/height)
- Zoom
- Brightness adjustments

Validation and test sets are not augmented; they are only normalized to measure real performance.

Training Procedure

Exploratory Data Analysis (EDA):

EDA was performed to understand dataset structure and quality before training. The main checks included: - **Number of classes** and images per class - **Class balance**: the dataset is reasonably balanced across classes, so no resampling or class weighting was required - **Image size distribution**: images showed consistent dimensions, reducing risk of irregular resizing issues - **Corrupted images detection**: images that failed to load correctly were identified

Data Cleaning:

A cleaning step removed corrupted or unreadable images. This prevents runtime errors and improves training stability. After cleaning, the dataset was re-verified to ensure all images could be loaded successfully.

Hyperparameters Used

Training Configuration: - Optimizer: **Adam** - Loss: **Categorical Cross-Entropy** - Metrics: **Accuracy** - Batch size: **32** - Epochs: **30** - EarlyStopping: used to stop training when validation performance stops improving - ModelCheckpoint: saves the best model weights

Note: Training was performed on CPU due to local hardware limitations. Model training speed can be significantly improved using cloud GPU platforms if needed.

Saved Artifacts:

The workflow saves: - Trained model weights (.h5) - Classification report text file - Confusion matrix images (raw counts + normalized) - Accuracy/loss curves ## Results

The models were evaluated using multiple metrics to capture both overall and class-level behavior.

Accuracy

Accuracy measures the percentage of correct predictions across the test set. While accuracy is a useful global measure, it does not fully explain which classes are confused with others.

- **Baseline test accuracy:** [To be filled with actual results]
- **MobileNet test accuracy:** [To be filled with actual results]

Loss Curves

[Training and validation loss curves to be included showing model convergence over epochs]

Confusion Matrix

The confusion matrix shows which classes the model confuses. A strong model shows a clear diagonal structure with minimal off-diagonal errors. In this project, most misclassifications occur between visually similar fish species, which is expected in fine-grained classification tasks.

[Confusion matrices for both models to be included]

F1-Scores

The classification report provides per-class metrics:

- **Precision:** how many predicted samples for a class are correct
- **Recall:** how many true samples of a class were correctly detected
- **F1-score:** harmonic mean of precision and recall (most informative overall)

Results Summary: - **Baseline macro/weighted F1:** [To be filled with actual results] - **MobileNet macro/weighted F1:** [To be filled with actual results]

Discussion

What Worked Well

Baseline CNN: - Strength: Simple, interpretable, validates pipeline - Successfully demonstrated the complete training/evaluation pipeline - Provided a solid reference point for comparison

MobileNet Transfer Learning: - Strength: Uses pretrained features, faster convergence, typically better generalization - Leveraged ImageNet pretrained weights effectively - Reduced training time compared to training from scratch

What Failed and Why

Challenges Encountered: - **Data leakage:** Initial dataset splits contained duplicate images across training and test sets, requiring careful re-splitting -

Hardware limitations: Training performed on CPU due to local hardware constraints, limiting model complexity and training speed - **Class imbalance:** Original non-augmented dataset had insufficient samples, particularly for Trout class

Model-Specific Issues: - Errors mostly occurred between visually similar species (as shown in confusion matrix) - Some confusion in difficult class pairs, though reduced in MobileNet compared to baseline

Limitations

1. **Hardware constraints:** Training performed on CPU limited model complexity and training speed
2. **Dataset limitations:** Reliance on augmented data due to insufficient original samples
3. **Fine-grained classification challenge:** Visual similarity between species makes classification inherently difficult
4. **Limited diversity:** Dataset may not capture full range of real-world variations (lighting, backgrounds, poses)

Performance Analysis:

Possible reasons MobileNet performs better (if applicable): - Pretrained backbone captures general visual features (edges, textures, shapes) - Less data required to learn strong representations - Better generalization under real-world variation

If MobileNet performance is similar to baseline: - Dataset is clean and consistent; baseline is already strong - Classes are visually separable; transfer learning provides marginal gains

Conclusion & Future Work

Conclusion

This project successfully demonstrates a full deep learning pipeline for fish species recognition, including:

- Comprehensive EDA and data cleaning
- Effective preprocessing and augmentation strategies
- Two modeling approaches (baseline CNN and MobileNet transfer learning)
- Thorough evaluation using accuracy, classification report metrics, and confusion matrices

The final results show strong performance on the test set, and confusion matrices confirm that most predictions are correct with limited confusion mainly among visually similar species. The project validates the effectiveness of both traditional CNN architectures and transfer learning approaches for fine-grained image classification tasks.

Future Work

Possible extensions to improve robustness and real-world performance:

1. **Fine-tuning MobileNet:** Unfreeze top layers with low learning rate for better adaptation
2. **Advanced architectures:** Try EfficientNet or ResNet for stronger feature extraction
3. **Background segmentation:** Add preprocessing to reduce background bias
4. **Dataset diversity:** Increase dataset diversity (different lighting, environments, camera angles)
5. **Hard-example mining:** Add systematic hard-example mining to focus training on difficult cases
6. **GPU acceleration:** Utilize cloud GPU platforms for faster training and larger model capacity
7. **Ensemble methods:** Combine multiple models for improved accuracy
8. **Real-time deployment:** Optimize models for mobile or edge device deployment ## References

9. TensorFlow Developers. (2023). TensorFlow Documentation. Retrieved from https://www.tensorflow.org/
10. Keras Team. (2023). Keras Documentation. Retrieved from https://keras.io/
11. Kaggle Dataset. A Large-Scale Fish Dataset. Retrieved from https://www.kaggle.com/datasets/crowww/a-large-scale-fish-dataset
12. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861. Retrieved from https://arxiv.org/abs/1704.04861
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
14. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
15. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255).