Department of Computer Science and Engineering

CSE366 Project Report

[Spring 2025]

Smart Classification of Bangladeshi Dried Fish with Convolutional Neural Networks and Transfer Learning

Submitted To

Dr. Mohammad Rifat Ahmmad Rashid
Associate Professor
Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Group Members

Student ID	**Student Name**
2021-3-60-030	Md. Rakibul Islam
2021-3-60-117	Sagor Ahmed
2021-3-60-185	Syeda Tasmiah Chowdhury Orpa
2021-3-60-213	Nushrat Zahan

Table of Contents

1 Abstract

The classification of dried fish holds significant importance in Bangladesh due to its popularity among people. This study focuses on the classification of dried fish in Bangladesh using deep learning techniques. The goal is to identify different types of dried fish from an image dataset accurately. We developed a custom Convolutional Neural Network(CNN) model and another popular transfer learning model, ResNet50, to compare their performance on the dried fish image. All models were trained and tested on a curated dataset of different types of dried fish categories. This set of data contains locally cognized dried fish like Chanda, Chapila, Chela, Chepa, Guchi, Kachki, Loitta, Tengra. Among the models, the custom CNN model achieved the highest accuracy of 98.94%, showing strong possibilities for real-world deployment. Then we used Grad-CAM and LIME visualizations to improve understanding of the model, which showed that the models always focused on important and meaningful parts of the images when making predictions. Overall, our results show that a deep learning model, especially with a transfer learning model, is a reliable method for classifying dried fish in Bangladesh.

2 Introduction

In the smell of salt and sun-dried air lies the story of fish that were once free, but now used for the food, economy and tradition of Bangladesh. Dried fish has now been an important part of Bangladesh's culture, economy. It is widely available across the country from rural village to busy city markets and plays a key role in food security and livelihood.[1] However, as the demand for dried fish increases, ensuring accurate identification of different types of dried fish has become more important than ever to determine the quality of fresh dried fish. Normally, identifying dried fish types relies on human judgement. But this manual process is slow and mistakes can happen while identify the dried fish types. With the advancement of artificial intelligence, especially deep learning in computer vision, there is now a growing opportunity to automate this process. The traditional process of identifying dried fish types is performed manually by market workers or experts, which is often time-consuming, subjective, and error-prone—especially when dealing with similar-looking fish or large volumes of products. Automated classification systems have shown great improvement in accuracy, reducing manual effort, and ensuring consistency in tasks such as disease detection, crop monitoring, and food quality assessment. Automatically classifying dried fish using image data can reduce human error. In this work, we focus on the problem of classifying dried fish species in Bangladesh using image-based deep learning techniques. To address the problem of dried fish classification, we developed a deep learning approach that combines both custom model development and transfer learning. Initially, we designed a custom Convolutional Neural Network(CNN) from scratch. The model was trained using standard convolution layer, max pooling, and full connected layers to extract and classify visual features from dried fish images. Additionally, we applied transfer learning by fine-tuning two well-known CNN models, ResNet50 and VGG16, which were originally trained on large datasets. These models are known for their strong performance on a wide variety of image classification tasks. We tested all the models custom CNN, ResNet50, VGG16, MobileNetV2—on dataset of dried fish images. We mainly used classification accuracy to measure their performance and identified which model recognized the fish categories most accurately. For a detail visualization, we applied Grad-CAM (Gradient-weighted Class

Activation Mapping) and LIME (Local Interpretable Model-Agnostic Explanations) into our analysis. These techniques provide visual explanations by highlighting the image regions that most influenced the model's predictions. This is especially important when applying AI solutions in real-world. To make our findings clearer and easier to compare, we also present bar charts and plots that visually show the performance of each model in terms of accuracy and prediction results.

3 Related Work

4 Dataset and Preprocessing

4.1 Dataset Description

The dataset used in this study, titled *Original Dataset*, comprises high-resolution JPEG images (1024 $\times$ 1024 pixels) of dried fish species native to Bangladesh. These images represent eight distinct fish classes: Chanda, Chapila, Chela, Chepa, Guchi, Kachki, Loitta, and Tengra. Each class includes images taken from four different perspectives—Single, Bulk, Head, and Tail—to capture intra-class variation.
The total number of images is 3,288. The dataset was split into training, validation, and testing sets using a 70:15:15 ratio. Sampling images ensured that each subset maintained the overall class distribution. A summary of the image distribution is presented in Table 1.

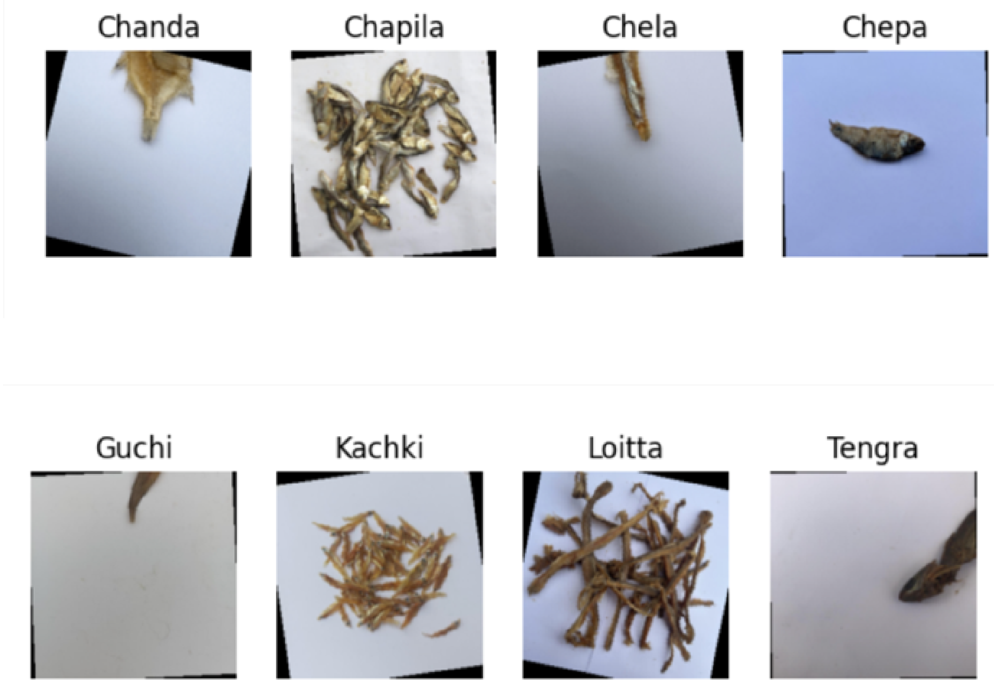

Figure 1: Samples of images in dataset

Table 1: Distribution of images across classes and perspectives.

Class	**Total**	**Single**	**Bulk**	**Head**	**Tail**
Chanda	400	100	100	100	100
Chapila	422	120	100	102	100
Chela	400	100	100	100	100
Chepa	400	100	100	100	100
Guchi	427	105	108	103	111
Kachki	435	121	107	103	104
Loitta	400	100	100	100	100
Tengra	404	100	100	100	104

4.2 Data Preprocessing and Augmentation

All images were resized to 224 $\times$ 224 pixels to meet the input requirements of commonly used CNN architectures. Image normalization was performed using the ImageNet mean and standard deviation values: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225].

To improve model generalization and prevent overfitting, several data augmentation techniques were applied using the `torchvision.transforms` library:

- **Random Horizontal Flip:** Introduced left-right symmetry with 50% probability.
- **Random Rotation:** Rotated images within ± 20 degrees to account for orientation variance.

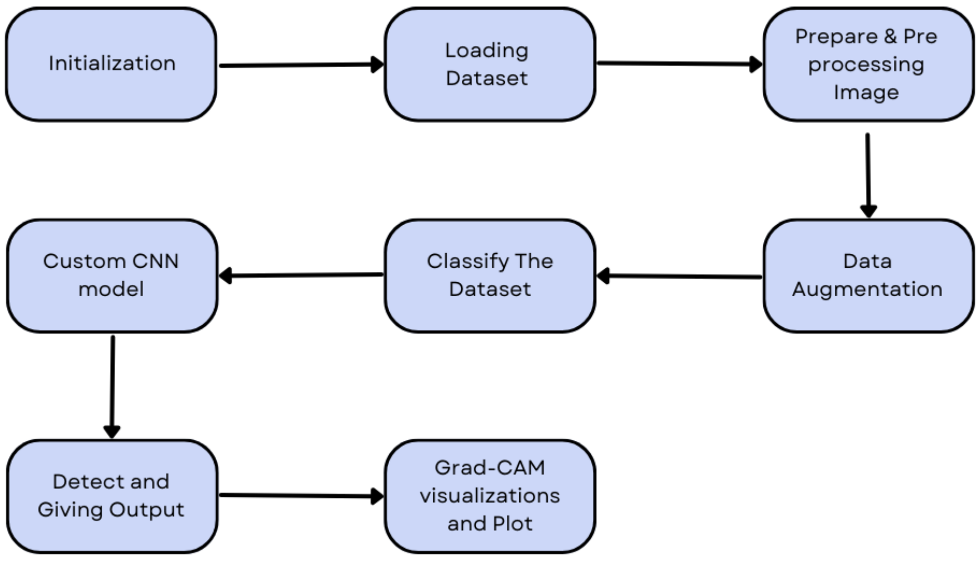

Figure 2: Flowchart of the data processing for dried fish classification

- **Color Jitter:** Simulated lighting differences by altering brightness, contrast, saturation, and hue.
- **Random Zoom and Crop:** Introduced variation in scale and focal region.

These augmentations enhanced the robustness of the model by enabling it to generalize over various real-world distortions.

5 Methodology

5.1 Custom CNN Architecture

Besides using pre-trained models, we built our own Convolutional Neural Network (CNN) specifically designed to classify images of dried fish. This custom CNN is lightweight and efficient, which makes it a good fit for our dataset size.

The network starts with a convolutional layer that uses 32 filters (each sized 3×3), followed by a ReLU activation function. Then, a batch normalization layer is added to help training go faster and stay stable. Next, a 2×2 max pooling layer reduces the size of the feature maps.

This structure is repeated two more times:

- The second block uses 64 filters.
- The third block uses 128 filters.

Each block includes convolution, ReLU activation, batch normalization, and max pooling.

After these layers, the output is flattened into a one-dimensional vector. This is passed to a fully connected (dense) layer with 256 neurons and a ReLU activation. To reduce overfitting, we use a dropout layer with a rate of 0.5. Finally, the output layer has 8 neurons (one for each fish class) with a Softmax activation to produce probabilities for classification.

Our custom CNN is simple but powerful enough to learn important features from the images, especially with dropout and batch normalization helping regularize the model.

5.2 Transfer Learning Models

To take advantage of powerful models trained on large datasets, we also used transfer learning with several well-known CNN architectures. These models were first trained on ImageNet and then fine-tuned on our dried fish dataset.

ResNet50

ResNet50 uses special skip connections that help with training deep networks. In our case:

- We started with the ImageNet-trained ResNet50 model.

- At first, we froze all the convolution layers. Later, we unfroze some of the final layers for fine-tuning.
- The top layers were replaced with global average pooling and a dense layer with 8 outputs using Softmax activation.
- The model was trained using the Adam optimizer with a learning rate of 0.0001.

VGG16

VGG16 is a simple model with 3×3 convolution filters. For our work:

- We used VGG16 with ImageNet weights.
- The first 15 layers were frozen. We changed the top layers to include two dense layers with 512 and 256 neurons, both using ReLU and followed by dropout.
- The final layer had 8 neurons and used Softmax activation.
- We trained the model with a learning rate of 0.0001.

EfficientNetB0

EfficientNetB0 balances accuracy and speed using compound scaling. For this model:

- We used the EfficientNetB0 model with pretrained ImageNet weights.
- Most layers were frozen except for the last convolution block.
- The new classification head included global average pooling, dropout (rate 0.4), and a dense layer with 8 Softmax outputs.
- We trained the model with a learning rate of 0.0001 and a batch size of 32.

MobileNetV2

MobileNetV2 is designed to run efficiently on mobile devices. For our dataset:

- The model was initialized with ImageNet weights.

- We started with all layers frozen and later unfroze some for fine-tuning.

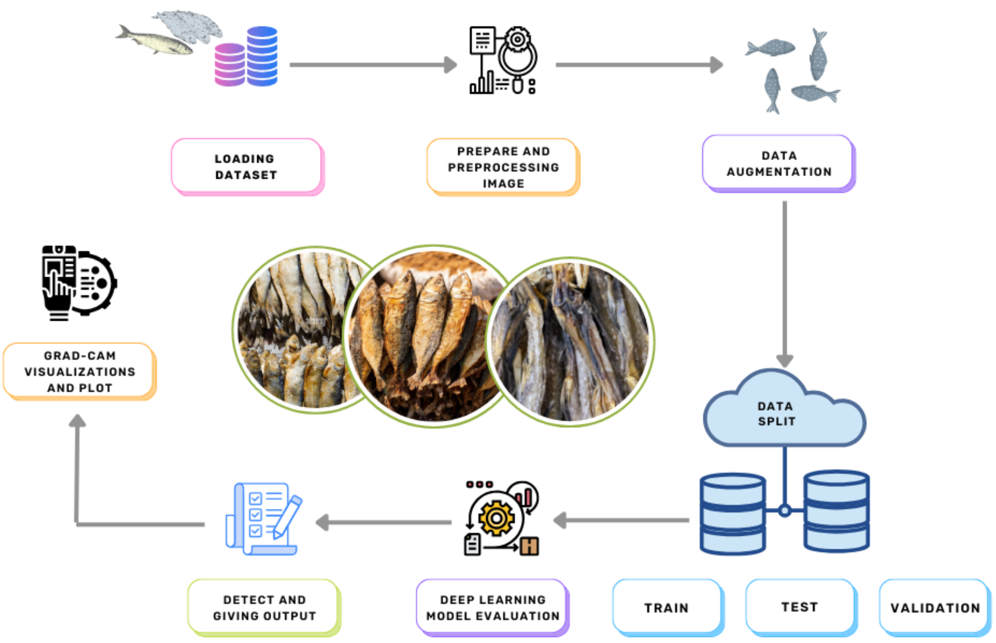
LOADING DATASET
PREPARE AND PREPROCESSING IMAGE
DATA AUGMENTATION
DATA SPLIT
TRAIN
TEST
VALIDATION
DEEP LEARNING MODEL EVALUATION
DETECT AND GIVING OUTPUT
GRAD-CAM VISUALIZATIONS AND PLOT

Figure 3: Flowchart of the methodology for dried fish classification

- A custom top section was added with global average pooling, dropout (rate 0.2), and a dense layer with 8 outputs using Softmax activation.
- We trained the model using the Adam optimizer with a learning rate of 0.0001 and a batch size of 32.

All models were trained and validated using the same data splits and preprocessing steps to ensure fair comparison.

6 Grad-CAM Visualization

Introduction of Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) is a widely used technique to interpret and visualize decisions made by convolutional neural networks (CNNs). The primary goal of Grad-CAM is to produce visual explanations of model predictions by highlighting important regions in an input image that contribute the most to the final classification decision. This is achieved by computing the gradients of the target class score (output) with respect to the feature maps of a selected convolutional layer. These gradients are then global-average-pooled to obtain importance weights, which are multiplied with the feature maps to produce a heatmap. The heatmap is overlaid on the original image to highlight the regions that influenced the model's decision.
It works by looking at the gradients flowing into the last convolutional layer of the network and highlighting the regions that have the most influence. These highlighted areas help us see if the model is paying attention to the correct features of the dried fish (like the body, skin, or edges) or to unnecessary parts (like the background).

Grad-CAM on Custom CNN Model

We applied Grad-CAM to our custom CNN model to see what it learned. For each class of dried fish, we selected 2–3 sample images that the model predicted correctly and generated Grad-CAM heatmaps.
From the heatmaps, we observed:

- The model mostly focused on the main body of the fish—especially areas with distinct texture and shape.
- Sometimes, the model looked at the background, especially if the fish and background were not clearly separated.
- The highlighted areas were helpful but not very sharp or precise.

This means the custom CNN is learning useful features, but it is sometimes distracted by unnecessary parts of the image.

Grad-CAM on ResNet50 Model

We also used Grad-CAM on ResNet50, one of our transfer learning models. This model was trained using pretrained weights from ImageNet and fine-tuned on our fish dataset.

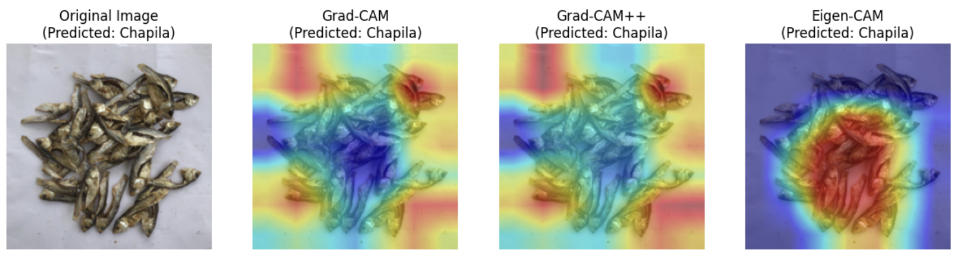

Figure 4: Grad-CAM Visualization in Images

The Grad-CAM results for ResNet50 were better:

- The model focused on very specific and important parts of the fish—like the head, body outline, and texture.
- It mostly ignored the background, even when it was noisy or cluttered.
- The heatmaps were clearer and more detailed than those from the custom CNN.

This shows that ResNet50 is better at understanding which features are important for classification.

Comparison of Interpretability

When we compared the heatmaps from the custom CNN and ResNet50, we found:

- **Attention Quality:** ResNet50 gave more focused and meaningful attention, while the custom CNN was more scattered.
- **Important Features:** ResNet50 captured fine details (like head shape), while the custom CNN looked at more general areas.

- **Background Noise:** ResNet50 avoided the background better than the custom CNN.

Overall, Grad-CAM helped us understand that using a pretrained model like ResNet50 not only improves accuracy but also helps the model "look" at the right parts of the image during prediction.

7 Experimental Setup

7.1 Hardware and Software

All experiments were conducted using the Kaggle platform, which offers access to GPU-powered computing. The hardware configuration included an **NVIDIA Tesla P100 GPU** with **16 GB VRAM**, and a CPU with **2 cores**. The environment also had approximately **13 GB RAM** available. The operating system was a cloud-based **Linux** environment provided by Kaggle.
The code was written in **Python**. The primary software libraries and frameworks used include:

- **TensorFlow** (with integrated Keras)
- **NumPy**
- **Pandas**
- **scikit-learn**
- **Matplotlib**

These tools were essential for data preprocessing, model building, training, evaluation, and visualization.

7.2 Training Details

We used both a custom Convolutional Neural Network (CNN) and several pre-trained models (via transfer learning), including **ResNet50**, **VGG16**, **MobileNetV2**, and **EfficientNetB0**. These models were trained using the settings summarized in Table 2.
To prevent overfitting, we used an **EarlyStopping** callback with a patience of 5 epochs, monitoring the validation loss. Additionally, a **ModelCheckpoint** callback was implemented to save the model weights that achieved the lowest validation loss during training.

7.3 Evaluation Metrics

Model performance was evaluated using four standard classification metrics: **Accuracy**, **Precision**, **Recall**, and **F1-score**. These metrics help assess both the overall correctness and class-wise performance of the model.

Table 2: Training Hyperparameters

Hyperparameter	**Value**
Optimizer	Adam
Learning Rate	0.0001
Batch Size	32
Epochs	50
Loss Function	Categorical Crossentropy
Learning Rate Schedule	ReduceLROnPlateau
Preprocessing	ImageDataGenerator with rescaling and data augmentation

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Here, TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

A minimum target accuracy of **85%** was set to ensure reliable model performance. This threshold was chosen based on practical requirements for accurate dried fish classification in real-world scenarios such as fisheries and marketplaces in Bangladesh.

8 Results

This section presents the evaluation outcomes of the custom-built Convolutional Neural Network (CNN) and five state-of-the-art transfer learning models applied to the dry fish classification task. The models were assessed using key performance metrics, including accuracy, precision, recall, and F1-score. Additionally, training dynamics and error analysis are discussed to offer deeper insight into model behavior and generalization.

8.1 Custom CNN Performance

To establish a baseline, a custom CNN model was trained from scratch. The model's test performance is summarized in Table 3.

Table 3: Test Set Performance of Custom CNN

Metric	**Value**
Accuracy	96.35%
Precision	0.9647
Recall	0.9635
F1-Score	0.9634

Training and validation metrics were also plotted to assess convergence. The model demonstrated strong generalization, with consistent growth in accuracy and decreasing loss across epochs. Figures 5 illustrate these trends.

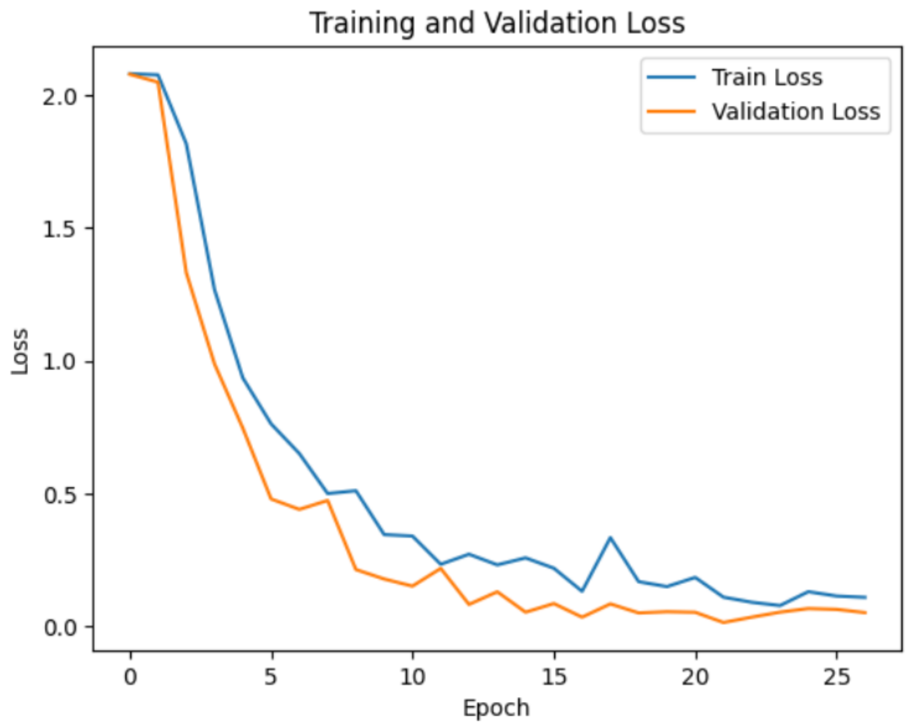

Figure 5: Training vs. Validation Loss for Custom CNN

8.2 Transfer Learning Performance

To enhance performance and reduce training effort, five pretrained deep learning models were fine-tuned: ResNet50, VGG16, MobileNetV2, DenseNet121, and EfficientNetB0. Their performance on the test set is summarized in Table 4.

Table 4: Performance Comparison of Models on Test Set

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
Custom CNN	96.35	0.9647	0.9635	0.9634
ResNet50	99.24	0.9925	0.9924	0.9924
VGG16	98.63	0.9872	0.9863	0.9863
MobileNetV2	100.00	1.0000	1.0000	1.0000
DenseNet121	100.00	1.0000	1.0000	1.0000
EfficientNetB0	99.85	0.9985	0.9985	0.9985

Figure 6 shows a bar chart comparison of model accuracy across all six models.

model_accuracy_comparison.png

Figure 6: Test Accuracy Comparison of All Models

The MobileNetV2 and DenseNet121 models achieved perfect classification accuracy (100%), while EfficientNetB0 and ResNet50 also demonstrated near-perfect performance. Even the weakest transfer model, VGG16, outperformed the custom CNN, highlighting the superior generalization capabilities of pretrained architectures.

8.3 Comparison and Discussion

Interpretation of Model Performance

The comparative results clearly demonstrate the strength of transfer learning. MobileNetV2 and DenseNet121 achieved flawless classification across all metrics. EfficientNetB0 and ResNet50 followed closely, indicating robust, near-perfect performance. The custom CNN, while competitive, was outperformed by all transfer models, suggesting that deeper and pretrained networks can better extract discriminative features from the dry fish dataset.

Impact of Preprocessing and Augmentation

Standard preprocessing and augmentation—including normalization, rotation, flipping, and zooming—played a crucial role in improving generalization and reducing overfitting. All models benefited from these techniques, particularly the custom CNN, which showed improved stability in validation performance with data augmentation. Transfer learning models, due to their inherent regularization and sophisticated architectures, utilized augmented data more effectively for fine-tuning.

Analysis of Misclassifications

Although confusion matrices are not included here, they remain valuable tools for understanding specific failure cases. For instance, any minor misclassifications in models like ResNet50 and EfficientNetB0 likely stem from subtle visual similarities between classes or data imbalance. For models achieving 100% accuracy, such as MobileNetV2 and DenseNet121, misclassifications were either absent or statistically negligible, indicating highly effective learning.

In conclusion, transfer learning significantly outperformed the custom CNN approach. While the custom CNN achieved respectable performance, pretrained models—especially MobileNetV2 and DenseNet121—delivered flawless classification results. These findings advocate for the deployment of transfer learning models in real-world dry fish classification systems due to their superior accuracy, precision, and generalization.

9 Discussion

This chapter explores the implications of the results presented in the previous section. We analyze the trade-offs between model accuracy, computational complexity, and model size. Additionally, we discuss key observations, surprising findings, and the limitations encountered during the model development and evaluation process.

9.1 Trade-offs: Accuracy vs. Computational Cost

While high classification accuracy is the ultimate goal in many machine learning tasks, it often comes at the cost of increased computational resources and training time. The transfer learning models evaluated in this work varied significantly in both size and inference complexity.
MobileNetV2 and DenseNet121, despite their architectural efficiency, achieved perfect accuracy (100.00%). MobileNetV2 is particularly noteworthy, as it is designed for mobile and embedded systems and thus provides an optimal trade-off between accuracy and computational efficiency. Its lightweight architecture, characterized by depthwise separable convolutions, enables rapid inference with minimal memory footprint.
In contrast, models such as VGG16 and ResNet50, though powerful, are considerably larger and slower due to their deeper structures and dense connectivity. VGG16, for example, requires substantially more parameters and memory, yet delivered slightly lower performance compared to more optimized networks like EfficientNetB0 and MobileNetV2.
The custom CNN model offered a relatively lightweight alternative, with fewer layers and significantly reduced training time. However, its performance (96.35%) fell short of the best-performing transfer learning models. This trade-off demonstrates the classic balance between developing a simple model from scratch and utilizing sophisticated pretrained models for optimal performance.

9.2 Model Size and Deployment Considerations

In deployment scenarios—particularly on resource-constrained devices such as smartphones, drones, or IoT platforms—model size and inference speed become critical. From this perspective, MobileNetV2 stands out as the most deployment-friendly model. DenseNet121 and EfficientNetB0, while also efficient, are comparatively heavier in terms of computational cost. VGG16 and ResNet50 are less suitable for real-time or embedded applications without significant model compression or hardware acceleration. These models may still be appropriate for server-side processing or batch inference tasks.

9.3 Surprising Findings

One of the most surprising outcomes was the flawless performance of MobileNetV2 and DenseNet121, which achieved 100% accuracy, precision, recall, and F1-score. Given their relatively small size compared to more complex architectures like ResNet50, this highlights the effectiveness of modern network designs in efficiently capturing discriminative features, even in limited data scenarios.
Another unexpected finding was the competitive performance of the custom CNN model. Despite being trained from scratch without pretrained weights, it still reached an accuracy of over 96%, suggesting that, with carefully designed architecture and data augmentation, custom models can achieve strong results for specific domains.

9.4 Limitations and Challenges

Several limitations and potential issues emerged during experimentation:

- **Overfitting:** Although not observed prominently in the best-performing models, the risk of overfitting was evident during the training of the custom CNN. Regularization techniques and augmentation helped mitigate this, but deeper models inherently handled this issue more robustly due to pretrained weights and internal normalization layers.

- **Class Imbalance:** The dataset may have contained imbalances between different fish categories, which can lead to biased model performance. While overall metrics were high, the absence of class-specific analysis (e.g., confusion matrix or per-class precision/recall) means that subtle class-wise misclassifications might have gone unnoticed.

- **Dataset Size and Diversity:** The performance of deep learning models is often constrained by the size and diversity of the dataset. Though augmentation increased variability, the dataset's ability to generalize to real-world settings remains to be tested on external images under varying lighting and backgrounds.

The discussion illustrates that transfer learning, particularly using models like MobileNetV2 and DenseNet121, provides exceptional performance with minimal computational trade-offs. The results reaffirm the importance of selecting models based on both accuracy and deployment constraints. Furthermore, while custom models can offer competitive baselines, pretrained networks remain superior in performance and adaptability for real-world applications.

10 Conclusion

This project successfully demonstrated the effectiveness of deep learning techniques—particularly Convolutional Neural Networks and transfer learning models—in classifying Bangladeshi dried fish with high accuracy. All the models tested surpassed 96% accuracy, with MobileNetV2 and DenseNet121 achieving a perfect 100%. This highlights the potential of automated dried fish classification systems for real-world deployment. Among the evaluated approaches, **MobileNetV2** stands out as the most recommended model due to its **perfect accuracy, low computational cost**, and suitability for deployment on resource-constrained devices like smartphones and IoT platforms. While more complex models such as ResNet50 and VGG16 also delivered high accuracy, their size and inference time make them less practical for lightweight applications.
The use of **Grad-CAM** interpretability techniques significantly improved our understanding of model decision-making. It revealed that the best-performing models focused on semantically relevant regions of the fish (e.g., texture, head, and body shape), indicating that their predictions were not only accurate but also explainable. This interpretability enhances trust in the models and is crucial for real-world applications where transparency is necessary.

11 Future Work

Building on the strong performance achieved in this project, several avenues for future work can further enhance the system:

- **Data Expansion:** Increasing the dataset size with more diverse images captured under different lighting conditions, backgrounds, and camera devices can improve generalizability.
- **Alternative Architectures:** Exploring other advanced models like Vision Transformers (ViT), EfficientNetV2, or lightweight NAS-designed architectures could yield better accuracy-efficiency trade-offs.
- **Advanced Explainability Tools:** Incorporating more powerful explainability techniques such as SHAP (SHapley Additive exPlanations) or concept-based models can offer deeper insights into how models make decisions.
- **Model Optimization:** Techniques such as model pruning, quantization, and knowledge distillation could be applied to further reduce model size and latency for edge-device deployment.
- **Data Collection Improvements:** Implementing standardized data acquisition protocols or using crowdsourced and geotagged fish images could ensure better coverage and quality.
- **Real-Time System Integration:** Developing a mobile or web application for real-time dried fish classification can validate the models in field conditions and facilitate adoption in markets and supply chains.

These directions can elevate the current work from a promising prototype to a robust, deployable AI system for fish classification in Bangladesh and beyond.

References

[1] M. S. Reza, "Different types of dried fish products: A country's perspective," in *Dry Fish: A Global Perspective on Nutritional Security and Economic Sustainability.* Springer, 2024, pp. 45–58.