Tuna Loin Quality Grading Using Image Processing and EfficientNetV2

1st Narendra Agastya
School of Electrical Engineering
Nanosatellite Laboratory
Telkom University
Bandung, Indonesia
narendradhika@student.telkomuniversity.ac.id

2nd Ledya Novamizanti
School of Electrical Engineering
The Center of Excellence of AILO
Telkom University
Bandung, Indonesia
ledyaldn@telkomuniversity.ac.id

3rd Gelar Budiman
School of Electrical Engineering
The Center of Excellence AICOMS
Telkom University
Bandung, Indonesia
gelarbudiman@telkomuniversity.ac.id

***Abstract*—The global tuna industry struggles with inconsistent quality grading due to the subjective and labor-intensive nature of manual methods. This study develops a tuna loin quality grading computer vision model based on EfficientNetV2 using image pre-processing techniques. Tuna loins are classified into three quality grades (A, B, and C) based on color and texture features. To address lighting inconsistencies of the environment and enhance texture detail of the tuna loins, the study evaluates pre-processing methods including Shades of Gray (SOG), Self-Adaptive Illumination Correction (SAIC), and Contrast Limited Adaptive Histogram Equalization (CLAHE). Various optimizers (Adam, AdamW, and SGD) and learning rate schedulers (Cosine Annealing and Cyclic) are also tested. The final model—combining SAIC and CLAHE pre-processing with EfficientNetV2M, the Adam optimizer, and Cyclic Learning Rate Scheduling—achieves 96.9% validation accuracy and 96.0% test accuracy. This demonstrates the effectiveness of the proposed approach in improving grading reliability for real-world applications.**

***Index Terms*—Tuna loin grading, EfficientNetV2, Image Pre-processing, Shades of Gray, Self-Adaptive Illumination Correction, Contrast Limited Adaptive Histogram Equalization, Optimizers, Learning Rate Schedulers.**

I. Introduction

Tuna is one of the most commercially significant seafood products worldwide. The global tuna market, valued at USD 43.14 billion in 2024, is expected to grow due to rising consumer demand for its high-protein content and omega-3 fatty acids, which are essential for brain and heart health [1] [2]. As global consumption increases, maintaining consistent quality standards in tuna products has become critical to ensuring consumer satisfaction and meeting export requirements.

Traditional tuna grading relies on human sensory evaluation of appearance and texture. However, this approach suffers from high inter-rater variability, subjectivity, and inconsistent quality control across different batches [3]. These limitations can pose significant challenges in maintaining grading reliability, especially in large-scale processing environments.

To address these issues, several works have explored computer vision [4] and machine learning techniques [5]–[8] for automating quality assessment in the seafood industry. Convolutional Neural Networks (CNNs), in particular, have shown strong performance in image-based classification tasks due to their ability to learn complex visual patterns. These studies highlight the critical limitations of existing methods, including small and imbalanced datasets, reliance on uniform lighting, and suboptimal generalization to real-world conditions. Given these challenges, this paper investigates whether advanced image pre-processing methods, when combined with the EfficientNetV2, can improve the accuracy and consistency of automated tuna loin quality grading. The main objective of this paper is to achieve a classification accuracy of at least 90% across three grades of tuna loin. This paper is divided into five sections: Introduction (discusses the problem, and the proposed solution), Related work, Methodology (discusses the Dataset, Pre-processing, and Model Configuration for training), Results and Discussions, and Conclusion.

II. Related Works

By leveraging computer vision models and feature extraction techniques, tuna industries can increase throughput, reduce labor costs, and maintain high-quality standards with minimal human intervention while ensuring consistency, accuracy, and efficiency in grading. Several studies have attempted to automate tuna quality grading with those techniques, each with notable limitations relating to limited datasets, lighting conditions, and generalization.

One approach, which is what this paper is inspired by, employs a Faster Region-based CNN (FR-CNN) combined with InceptionV2 (IncV2) [9]. This method, trained on a dataset of 1,517 images (after augmentation), achieved an overall accuracy of 92.8% in classifying three grades of tuna meat [9]. However, the FR-CNN combined with the lightweight Inception V2 model is not designed for real-time grading, as the overall process depends on two stages, region proposal and classification, which can cause inference times to be longer. [9]. Furthermore, this study does not utilize a pre-processing method that normalizes lighting and improves texture details because the dataset was collected from only one location under controlled lighting and background conditions, which limits the model's ability to generalize in real-world scenarios where variations in lighting and background are prevalent [9].

Another study utilizes Discrete Wavelet Transform (DWT) for feature extraction with the k-Nearest Neighbors (KNN) to classify four grades of tuna meat (1, 2+, 2, and 3) [10].

979-8-3315-8649-2/25/$31.00 ©2025 IEEE

This method possibly offers computational efficiency to an extent since a deep learning model was not used. However, not using a deep learning model can also be a disadvantage, since detailed hierarchical characteristics of the tuna meat cannot be learned [10]. The images are converted from RGB to HSV color space to obtain a certain color composition that can facilitate color segmentation [10]. Feature extraction is then performed on each channel using Symlet wavelet and Haar wavelet [10]. However, the dataset used in this method was limited in number. 95 images were used for training and 66 images for testing [10]. Symlet achieved an accuracy of 81.8% and Haar achieved an accuracy of 80.3% [10].

Another method utilizes the combination of the color histograms in Red-Green-Blue (RGB) and Hue Saturation Value (HSV) color spaces for color features and Grey Level Co-occurence Matrix (GLCM) for texture features to classify three grades of tuna meat (Grades A, B, and D) using a Support Vector Machine (SVM) classifier [11]. The GLCM is a statistical method used in image processing to analyze texture by examining how pixel intensities (gray levels) relate to one another in a spatial neighborhood [11]. The study tested four experimental scenarios, evaluating various combinations of features and classifier tuning [11]. The best result—81.6% accuracy—was achieved using both RGB and HSV histograms along with GLCM features and a tuned SVM [11]. However, the dataset was relatively small (only 36 images) under uniform background and lighting conditions, raising concerns about the model's ability to generalize to other real-wold scenarios [11].

Compared to deep learning approaches, this method benefits from lower computational cost, though it lacks the robustness and scalability of deep learning approaches [11]. The study highlights the potential of texture and color-based features in quality control systems but highlights the need for larger datasets, the utilization of a deep learning model, and other advanced pre-processing techniques for broader applications [11].

III. Methodology

This study focuses on developing an EfficientNetV2 model based on the PyTorch framework using the Python programming language with hyperparameter tuning to classify three grades of tuna loin based on their color and texture details as they are the key features to determining the loins' freshness and quality. The workflow begins with data acquisition, followed by pre-processing and augmentation techniques to enhance the dataset and improve model performance.

The proposed classification method is illustrated in Fig. 1, which provides a schematic overview of the tuna loin grading process. It begins with the acquisition of the dataset, followed by the implementation of augmentation and image pre-processing. Then, all images in the dataset are labeled with their respective quality grades and then split into training, validation, and test sets. The training and validation sets are used to train the model and the results will be taken into consideration for tuning the model in order to achieve the best performing model. The best performing model will then be tested by classifying the tuna loin images in the test set.

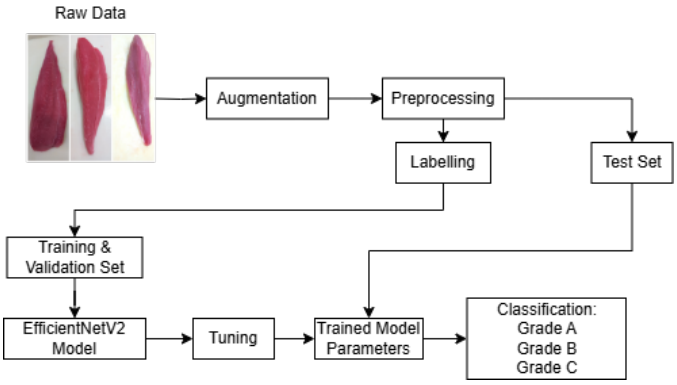

Fig. 1. Schematic representation of the Proposed Tuna Loin Quality Assessment using EfficientNetV2

A. Dataset

The dataset used in this study was provided by PT. Aruna Jaya Nuswantara from their two facilities located in Ternate and Dempo. The dataset of three different grades of tuna loins (Grades A, B, and C) were captured using a mobile device at resolutions of 3000×4000, 4000×3000, 2250×4000 and 4000×2250 pixels under varying lighting conditions and background settings. During the acquisition, 418 images were taken for Grade A, 602 images for Grade B, and 247 images for Grade C, resulting in a total of 1,267 images with an issue of unbalanced distribution across the three grades. Fig. 2. shows a sample of the tuna loin dataset used in this study and Table I shows the characteristics that determine each of the three grades.

Fig. 2. Tuna loins: Grade A, Grade B, Grade C (left to right)

TABLE I
Tuna Loin Grades Characteristics

Grades	Color	Texture & Details
Grade A	Deep red	Smooth with thin fibers/tendons
Grade B	Light red	Slightly coarse with moderately thick fibers/tendons
Grade C	Pale red	Coarse with thick fibers/tendons

B. Augmentations

To enhance the dataset size and improve the robustness of the EfficientNetV2 model, data augmentation techniques were applied using the OpenCV and Albumentations libraries in Python to increase and balance dataset size across the three grades. These included combinations of flipping (horizontally and vertically), rotations (90°, 180°, and 270°), shearing (20°), and zoom transformations (70%), which expanded the dataset to 12,593 images. Then, the dataset was split into three sets: training (80%), validation (10%), and test (10%). Table II shows the dataset quantity per grade before and after applying the augmentations.

TABLE II
DATASET QUANTITY

Grades	Original	Augmented	Original + Augmented
Grade A	418	3792	4180
Grade B	602	3612	4214
Grade C	247	3952	4199
Total	1267	12326	12593

C. Image Pre-processing

To ensure consistent lighting conditions while preserving essential features such as texture and color variations, image pre-processing techniques were applied before training the EfficientNetV2 model for this study. These techniques were implemented using the OpenCV and NumPy libraries in Python. The methods used include Shades of Gray (SOG), Self-Adaptive Illumination Correction (SAIC), and Contrast Limited Adaptive Histogram Equalization (CLAHE).

The SOG algorithm is a white balance correction method that adjusts the color balance of an image by normalizing the illumination across different lighting conditions [12]. The algorithm works by first converting the input image to a floating-point format for precise calculations [12]. It then computes the Minkowski norm (p) for each color channel (Red, Green, Blue) using a parameter (default is 2, which corresponds to the Euclidean norm). These norms represent the average intensity of each channel, raised to the power of p, averaged, and then the p-th root is taken [12]. The algorithm calculates the mean of these three channel norms and uses it to scale each channel so that their intensities are balanced relative to each other [12]. This scaling adjusts the image colors, making them more consistent regardless of the original lighting [12].

SAIC is a pre-processing technique used to adjust the overall brightness of an image so that its average intensity matches a predefined target value, typically representing a neutral or mid-level brightness (here, 128 out of 255) [13]. The process begins by reading the image and converting it to grayscale to calculate its mean intensity, which reflects the image's overall brightness [13]. Then a scaling factor is calculated as the ratio of the target intensity to the current mean intensity [13]. This factor is applied uniformly to all color channels of the original image, effectively brightening or darkening the image as needed to bring its average brightness closer to the target [13].

CLAHE is an advanced image pre-processing technique designed to enhance the local contrast of images, particularly in regions that are darker or lighter than most of the image [14]. Unlike traditional histogram equalization, CLAHE applies local contrast adjustments within small image regions, ensuring that details in both bright and dark areas are preserved [14]. This technique enhances the visibility of fine-grained texture variations, fibers, and tendons in tuna loins, which is critical for accurate grading.

To ensure consistency across pre-processing techniques, all images were resized to the default input resolution of the EfficientNetV2M model (480×480 pixels) whereas the EfficientNetV2S had a smaller image input resolution (384×384 pixels) [15]. Resizing to lower-resolution images may introduce loss of detail, which could affect texture-based feature extraction. However, this can be mitigated by using contrast enhancement techniques like CLAHE.

By combining these pre-processing techniques, the dataset is normalized for lighting inconsistencies, enhanced for contrast and texture visibility, and optimized for computer vision feature extraction, ultimately improving the model's robustness in real-world grading scenarios. Fig. 3. shows the same sample of the dataset shown in Fig. 2. but with the SOG + CLAHE pre-processing combination in the first row whereas the second row shows the same sample with the SAIC + CLAHE combination.

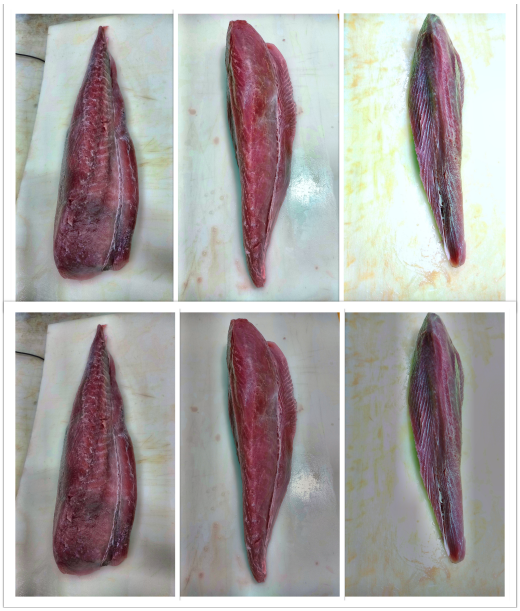

Fig. 3. SOG + CLAHE (First Row), SAIC + CLAHE (Second Row)

D. Model Training Configuration

Convolutional Neural Networks (CNNs) are widely recognized for their ability to automatically learn hierarchical features from images [16]. EfficientNetV2, a state-of-the-art

CNN architecture, was selected to be the model for tuna loin classification as it offers an optimal balance of accuracy and computational efficiency which also makes it ideal for real-time inference and scalable classification tasks in mobile applications, such as automated tuna loin grading, where fast and reliable predictions are critical. Fig. 4. shows the architecture of the EfficientNetV2M model in PyTorch [15].

EfficientNetV2 improves upon its predecessor by integrating Fused-MBConv layers, which merges the expansion and depthwise convolution stages into a single standard convolution. In contrast, EfficientNetV1 relies on depthwise separable convolutions [17]. This design significantly reduces memory access costs and accelerates training, particularly on modern hardware accelerators. The medium variant of EfficientNetV2, EfficientNetV2M, was used as the primary model. The small variant, EfficientNetV2S, was also tested, while the large variant, EfficientNetV2L, a more complex version of the three, was excluded due to hardware limitations.

A key feature of EfficientNetV2 is its compound scaling strategy, which systematically adjusts the model's depth, width, and input resolution to achieve balanced performance across various deployment environments such as mobile devices [17]. Unlike traditional scaling, which modifies one parameter at a time, compound scaling optimizes depth, width, and resolution simultaneously, improving stability and generalization [17]. This allows the model to begin with smaller, lightly regularized images for rapid early-stage learning and progressively transition to larger, more regularized images, improving final accuracy [17]. This approach enhances generalization while optimizing training efficiency [17].

EfficientNetV2 balances accuracy and computational efficiency, making it ideal for real-time inference and scalable classification tasks [17]. Its efficiency makes it particularly ideal for mobile applications, such as automated tuna loin grading, where fast and reliable predictions are essential.

Training deep learning models effectively also requires the use of robust optimization techniques. Training is performed with three different optimizers: Adam, AdamW, and Stochastic Gradient Descent (SGD). Additionally, each optimizer is experimented with two learning rate schedulers (LRS): Cosine Annealing and Cyclic.

Adam is an optimizer that adaptively adjusts the learning rate for each parameter by combining momentum (which tracks the moving average of past gradients) and RMSProp (which tracks the moving average of squared gradients), allowing for faster and more stable convergence during training. [18]. AdamW is a refined version of Adam that correctly applies weight decay (a regularization technique to prevent overfitting) by decoupling it from the gradient update step, which addresses a known flaw in the original Adam formulation [19]. This separation leads to more effective regularization and improved generalization, making AdamW the preferred choice in modern deep learning architectures [19]. SGD uses only one data sample (or a small mini-batch) at a time to gradually update the model's parameters (or weights) in the direction that reduces the error [20].

For training with the Cosine Annealing LRS, the initial learning rate was set to 0.001, and the final learning rate was set to 0.000001. Cosine Annealing gradually decreases the learning rate following a cosine curve, allowing for a smooth reduction in learning rate over time, which helps prevent overshooting the optimal solution and improves convergence stability [20].

For training with the Cyclic LRS, the base learning rate was set to 0.00001, and the maximum learning rate was 0.001. Cyclic LRS oscillates the learning rate between these values in a cyclic manner, which can help escape local minima and improve generalization by periodically reintroducing higher learning rates [21]. The number of training epochs was fixed at 50, and the batch size was set at 8 to balance computational efficiency with model performance. However, for this paper, only the results from the best epoch will be shown from each training run.

IV. Results and Discussions

A. Experiment Setup

In this study, all training, validation, and testing experiments were conducted using the NVIDIA RTX 4060Ti GPU with 8GB VRAM and 32 GB of RAM. However, the inference experiment was conducted on a computer equipped with an Intel® Core™ i7-14700KF processor and 32 GB of RAM due to time constraints with mobile deployment.

B. Training, Validation, and Testing

This subsection provides training results for EfficientNetV2M and EfficientNetV2S models with different hyperparameters and dataset variations. Three types of accuracies are used to evaluate the models' performances.

Training accuracy measures how well the model predicts the correct labels on the same data it was trained on. It is calculated by running the model on the training set, counting the number of correct predictions, and dividing by the total number of samples. High training accuracy means the model has learned to fit the training data, but it does not guarantee good performance on new, unseen data.

Validation accuracy is calculated on a separate set of data (the validation set) that the model has not seen during training. It is calculated similarly to training accuracy, but using the validation set (which the model does not see during training). Validation accuracy aids in monitoring how well the model generalizes to new data and is often used to tune hyperparameters or decide when to stop training (to avoid overfitting).

Test accuracy is measured on a third, completely unseen dataset (the test set) after all training and model selection are complete. Similar to the previous accuracies, it counts correct predictions on the test set and divides by the total number of test samples. Test accuracy provides an unbiased estimate of how well the model will perform in real-world scenarios.

Results in Table III show the EfficientNetV2M model performance with different optimizer and LRS settings when trained using the original dataset. Training and validation

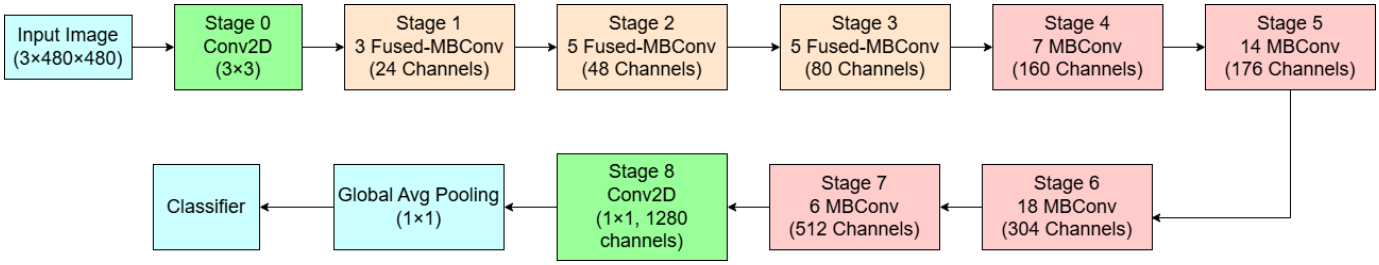

Fig. 4. EfficientNetV2M Architecture

TABLE III
TRAINING RESULTS ON EFFICIENTNETV2M WITH ORIGINAL DATASET

Scenario	Optimizer	LRS	Train Acc (%)	Val Acc (%)
1	SGD	CosineAnnealing	33.4	35.4
2	SGD	Cyclic	34.2	33.9
3	AdamW	CosineAnnealing	99.4	89.0
4	AdamW	Cyclic	99.0	93.8
5	Adam	CosineAnnealing	99.5	89.4
6	Adam	Cyclic	**99.5**	**95.6**

accuracies were used to evaluate the performance of the model to determine whether they were overfitting.

Notably, the Adam optimizer with the Cyclic LRS achieved a training accuracy of 99.5% alongside validation accuracy of 95.6% (Scenario 6), making it the best-performing combination in this study. The AdamW optimizer also performed well, achieving a 93.8% validation accuracy (Scenario 4) when paired with Cyclic LRS. This suggests that both Adam-based optimizers contribute to strong model generalization, with Adam slightly outperforming AdamW for the original dataset.

On the other hand, model configurations using the SGD optimizer performed significantly worse, with validation accuracy dropping below 36% regardless of the LRS used (Scenarios 1 and 2). This suggests that SGD is not well-suited for this dataset and model architecture, likely due to its slower convergence and tendency to get stuck in local minima. The significant performance gap between Adam-based and SGD optimizers underscores the importance of adaptive gradient methods for high classification accuracy.

Cyclic LRS consistently outperforms Cosine Annealing LRS across all optimizers, as shown by the comparison of learning rate strategies. For instance, with the Adam optimizer, Cyclic LRS achieved a validation accuracy of 95.6%, while Cosine Annealing LRS reached only 89.4% (Scenario 5). Similarly, with AdamW, Cyclic LRS resulted in 93.8% accuracy, outperforming Cosine Annealing LRS, which achieved 89.0%. This suggests that Cyclic LRS, which cyclically adjusts the learning rate within a defined range, enhances model generalization more effectively than Cosine Annealing LRS, which gradually decreases the learning rate over time.

In summary, the Adam optimizer with the Cyclic LRS is the most effective configuration for training EfficientNetV2M on this dataset, yielding the highest validation accuracy while ensuring strong generalization. Although AdamW also performed well, it was slightly less effective than Adam. In contrast, SGD proved unsuitable, likely due to its slow convergence and inability to optimize the EfficientNetV2M model efficiently. Additionally, Cyclic LRS emerged as the superior LRS, consistently improving validation accuracy across different optimizers. Therefore, the best combinations for training the EfficientNetV2M and EfficientNetV2S models on the pre-processed dataset are the Adam optimizer paired with the Cyclic LRS.

Table IV contains results for EfficientNetV2M and EfficientNetV2S models with the Adam optimizer with Cyclic LRS. Validation and test accuracies have been utilized to analyze them, giving insight into performance and generalization on unseen datasets.

Analyzing the results, EfficientNetV2S (Scenario 7) achieved a validation accuracy of 96.9% and a testing accuracy of 96.1% when trained with SOG + CLAHE. However, when using SAIC + CLAHE (Scenario 8), its validation accuracy remained at 96.1%, but the testing accuracy dropped significantly to 94.3%. This suggests that while both pre-processing techniques allowed EfficientNetV2S to perform well on validation data, SOG + CLAHE provided better generalization on the test set.

EfficientNetV2M (Scenarios 9 and 10) demonstrated an opposite trend. With SOG + CLAHE, the model obtained a validation accuracy of 96.1% and a testing accuracy of 95.3%. In contrast, SAIC + CLAHE yielded an improved validation accuracy of 97.0% and an enhanced testing accuracy of 96.0%. This indicates that, unlike EfficientNetV2S, the EfficientNetV2M model benefited from SAIC + CLAHE pre-processing, achieving the highest overall performance across both validation and testing sets.

From these observations, we can conclude that pre-processing techniques play a significant role in model performance, with varying effects depending on the network architecture. For EfficientNetV2S, SOG + CLAHE provided superior generalization, whereas EfficientNetV2M performed

TABLE IV
PERFORMANCE COMPARISON BETWEEN PRE-PROCESSING COMBINATIONS AND EFFICIENTNETV2 VARIANTS

Scenario	Model	Pre-processing	Val Acc (%)	Test Acc (%)
7	EfficientNetV2S	SOG+CLAHE	96.9	96.1
8		SAIC+CLAHE	96.1	94.3
9	EfficientNetV2M	SOG+CLAHE	96.1	95.3
10		SAIC+CLAHE	**97.0**	**96.0**

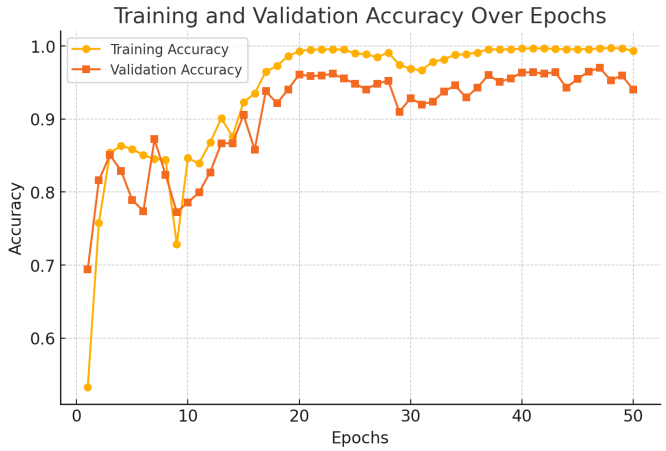

Fig. 5. Scenario 10 Model Training & Validation Accuracy

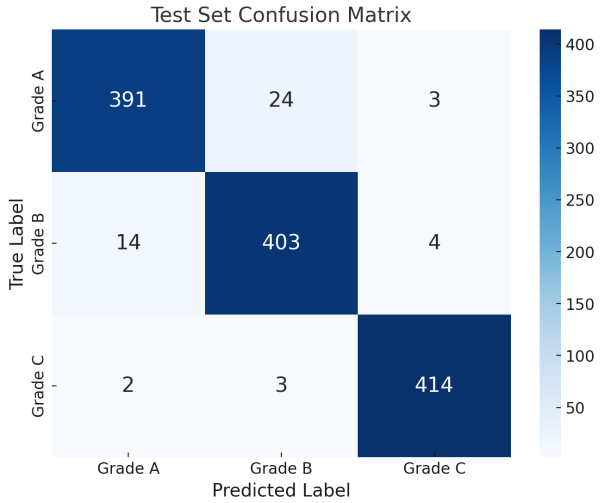

Fig. 6. Scenario 10 Test Confusion Matrix

best with SAIC + CLAHE. This suggests that deeper or more complex models may benefit slightly from different preprocessing strategies compared to smaller models. The results also emphasize the importance of evaluating both validation and testing accuracy to ensure a model's ability to generalize effectively beyond the training data.

Fig. 5. shows the graph of the training and validation accuracy of the best model from Table IV (Scenario 10) during its training and validation. EfficientNetV2M had a strong learning curve with the Adam optimizer and Cyclic Learning Rate Scheduling. Training accuracy rose from epoch 1's 53.3% to epoch 50's 99.3%, and validation accuracy from 69.4% to 94.0%. Early epochs saw steep rises, with validation accuracy already crossing 80% by epoch 2 and plateauing in later epochs in high-90% territory. Validation accuracy reached a peak of 97.0% at epoch 47 but saw some minor declines in later epochs. By the final epoch (epoch 50), the training accuracy was 99.3%, and the validation accuracy settled at 94.0%.

Even with some surges in validation loss, robust generalization was maintained by the model. Periodic adjustments to the learning rate by Cyclic LRS likely averted it from overfitting. Overall, the model performed adequately with very minimal fluctuations that could be further refined with fine-tuning. Fig. 6. presents Scenario 10's confusion matrix for testing. In terms of classification performance, the model correctly identified most instances in each grade. Grade A was classified correctly 391 times, with 24 misclassified as Grade B and 3 as Grade C. Grade B showed strong classification performance, with 403 correct predictions, 14 misclassified as Grade A, and 4 as Grade C. Grade C had the highest accuracy, with 414 correct classifications and only 5 misclassified instances. This suggests that the model was particularly effective at distinguishing Grade C from the other categories.

Table V presents the results of Scenario 10's testing. The precision, recall, and F1-score, all ranged between 95% and 99%, indicating a well-balanced model. Precision is defined as the ratio of correctly identified positive instances to the total instances identified as positive, reflecting the model's accuracy in labeling positive cases. Recall measures the ratio of correctly identified positive instances to the total actual positive instances, indicating the model's ability to capture all relevant cases. The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both false positives and false negatives. Grade C had the highest recall (99%), meaning nearly all Grade C instances were correctly identified. In contrast, Grade A had the lowest recall (94%), making it slightly more prone to misclassification. The minor misclassification between Grade A and Grade B suggests some overlapping characteristics in appearance, which may be addressed through additional training data or refined feature extraction. Overall, the model demonstrates excellent classification performance with strong F1-scores (95% to 99%), confirming its ability to generalize well to unseen data.

TABLE V
SCENARIO 10 TEST RESULTS

Metric	Grade A	Grade B	Grade C
Precision (%)	96	94	98
Recall (%)	94	96	99
F1-Score (%)	95	95	99

C. Model Inferencing

The best performing model (Scenario 10) was exported to the Open Neural Network Exchange (ONNX) using PyTorch in Python, which is an open format built to represent machine learning and deep learning models for mobile deployment [22] [23]. Scenario 10's model was tested on three images of tuna loin grades that were not included in the dataset used in the previous subsections. Table VI presents the inference results, demonstrating excellent performance, particularly for Grades B and C. The model correctly classified all samples, with the Grade A class showing a minor misclassification of

1.38% as Grade B. Meanwhile, the Grade C class exhibited a slight misclassification, with values close to 0.0% incorrectly predicted as Grade A and Grade B. This suggests that while the model performs well overall, it has a minimal challenge distinguishing between adjacent quality grades, particularly Grade A and Grade B. These findings align with previous observations, indicating the robustness of the model while highlighting areas for further refinement. Table VII summarizes the inference time and resource usage. The Grade A image had the longest inference time (92.23 ms), while the Grade B image was the fastest (69.30 ms), and the Grade C image took 73.63 ms. On average, inference time was approximately 78.38 ms. Memory usage had a maximum at 209.90 MB for the Grade C image, with the Grade B image being in the middle with 209.79 MB, followed by Grade A with the least (209.77 MB), averaging 209.82 MB across all the three experiments.

TABLE VI
SCENARIO 10 ONNX INFERENCING RESULTS

Actual (%)	**Predicted (%)**		
	Grade A	**Grade B**	**Grade C**
Grade A	98.6	1.4	0.0
Grade B	0.0	100.0	0.0
Grade C	<0.1	<0.1	99.9

TABLE VII
SCENARIO 10 ONNX INFERENCE TIME AND MEMORY USAGE

Grade	**Inference Time (ms)**	**Memory Usage (MB)**
Grade A	92.23	209.77
Grade B	69.30	209.79
Grade C	73.63	209.90

D. Comparison with Previous Works

Table VIII shows the comparison between proposed automated tuna grading methods that were mentioned in Section II. It describes each methods' model, pre-processing, dataset quantity, and the best accuracy achieved overall. This paper introduces the use of alternative techniques of image pre-processing. The quantity of the dataset in this paper (12,593 images) is also significantly larger than those of the other methods, ensuring that the trained model has excellent generalization in the classification of the tuna loin under various lighting conditions. The final test accuracy of 96.0% in this paper is also significantly higher than two of the three previous methods mentioned, with the FR-CNN and IncV2 combination achieving the second best overall accuracy of 92.8%.

TABLE VIII
COMPARISON OF AUTOMATED TUNA GRADING METHODS

Model	**Pre-processing**	**Dataset Quantity**	**Acc (%)**
FR-CNN & IncV2	None	1,517	92.8
KNN	DWT	161	81.8
SVM	RGB, HSV, GLCM	36	81.6
EfficientNetV2M	**SAIC+CLAHE**	**12,593**	**96.0**

V. CONCLUSION

This study presents an automated tuna loin grading system that uses the SAIC and CLAHE pre-processing with EfficientNetV2M, the Adam optimizer, and Cyclic Learning Rate Scheduling. This configuration achieved the best performance, with a validation accuracy of 97.0% and a test accuracy of 96.0%. The EfficientNetV2S variant also demonstrated strong results when paired with SOG + CLAHE pre-processing, the Adam optimizer, and Cyclic LRS, achieving 96.9% validation accuracy and 96.1% test accuracy. However, EfficientNetV2M is the preferred model due to its higher input resolution, which more effectively preserves tuna loin features. This work has produced results that are significantly better than what was achieved in previous works mentioned in this paper. Future work may focus on addressing dataset imbalance prior to augmentation, investigating additional pre-processing methods to further enhance lighting consistency and texture representation, and evaluating other computer vision and deep learning models for potential accuracy improvements and deploying them on a mobile device.

ACKNOWLEDGMENT

This research was supported by the Directorate of Research and Community Service, Telkom University, through the Internal Fund Research Grant. The authors would like to thank the Nanosatellite Laboratory at Telkom University, for providing the hardware and workspace to conduct the experiments. The authors also extend gratitude to PT. Aruna Jaya Nuswantara, Indonesia, for providing the tuna loin dataset and sharing their expertise.

REFERENCES

[1] "Tuna Fish Market Size, Share, Growth Report, Forecast, 2032 — fortunebusinessinsights.com." https://www.fortunebusinessinsights.com/industry-reports/tuna-fish-market-100744. [Accessed 08-05-2025].

[2] N. Swetha and S. Mathanghi, "Towards sustainable omega-3 fatty acids production – a comprehensive review on extraction methods, oxidative stability and bio-availability enhancement," *Food Chemistry Advances*, vol. 4, p. 100603, 2024.

[3] K. Naziba Tahsin, "A review on the techniques for quality assurance of fish and fish products," *International Journal of Advanced Research in Science and Engineering*, vol. 4, pp. 4190–4206, 07 2017.

[4] A. K. Aziz, M. D. Maulana, R. F. Adawiyah, R. F. Firdaus, L. Novamizanti, and F. Ramdhon, "Comparative analysis of yolov8 models in skipjack fish quality assessment system," in *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pp. 237–242, IEEE, 2023.

[5] N. R. Pratama, L. Novamizanti, and D. R. Wijaya, "Automated tuna freshness assessment via gas sensors and machine learning algorithms," in *2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pp. 1008–1012, IEEE, 2024.

[6] H. A. Nurqamaradillah, L. Novamizanti, and D. R. Wijaya, "Comparing knn and svm for aroma-based tuna freshness detection," in *2024 IEEE 2nd International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT)*, pp. 123–128, IEEE, 2024.

[7] M. R. M. Setyagraha, L. Novamizanti, and D. R. Wijaya, "Comparison of k-nn and naive bayes algorithms for classifying mackerel tuna freshness through gas sensors," in *2024 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pp. 157–161, IEEE, 2024.

[8] L. M. Hermawan, L. Novamizanti, and D. R. Wijaya, "Crab quality detection with gas sensors using a machine learning," in *2024 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, pp. 270–275, IEEE, 2024.

[9] J. E. C. Rosal, D. I. E. Hisola, and M. S. Demabildo, "Grade classification of yellowfin tuna meat using f-rcnn with inception v2 architecture," in *2023 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pp. 252–256, 2023.
[10] I. G. S. E. Putra, I. K. G. Darma Putra, M. Sudarma, and A. A. Kompiang Oka Sudana, "Classification of tuna meat grade quality based on color space using wavelet and k-nearest neighbor algorithm," in *2023 International Conference on Smart-Green Technology in Electrical and Information Systems (ICSGTEIS)*, pp. 35–40, 2023.
[11] M. Naimullah, F. Sthevanie, and K. N. Ramadhani, "Tuna meat grade classification using color histogram and grey level co-occurrence matrix," *eProceedings of Engineering*, vol. 7, no. 2, 2020.
[12] B. S. Thai, G. Deng, and R. Ross, "A fast white balance algorithm based on pixel greyness," *Signal, Image and Video Processing*, vol. accepted, 03 2017.
[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Global Edition*. London, England: Pearson Education, 4 ed., Sept. 2017.
[14] T. P. H. Nguyen, Z. Cai, K. Nguyen, S. Keth, N. Shen, and M. Park, "Pre-processing image using brightening, clahe and retinex," 2020.
[15] "EfficientNetV2 &x2014; Torchvision main documentation — pytorch.org." https://pytorch.org/vision/main/models/efficientnetv2. [Accessed 08-05-2025].
[16] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, p. 52, 2023.
[17] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," 2021.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.
[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.
[20] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017.
[21] L. N. Smith, "Cyclical learning rates for training neural networks," 2017.
[22] "ONNX — Home — onnx.ai." https://onnx.ai/. [Accessed 14-05-2025].
[23] "torch.onnx &x2014; PyTorch 2.7 documentation — pytorch.org." https://pytorch.org/docs/stable/onnx.html. [Accessed 08-05-2025].