

JURNAL RESTI

(Rekayasa Sistem dan Teknologi Informasi)

Vol. 9 No. 3 (2025) 436 - 448

e-ISSN: 2580-0760

The Effect of Hyperparameters on Faster R-CNN in Face Recognition Systems

Jasman Pardede^{1*}, Khairul Rijal²

^{1,2}Department of Informatics, Faculty of Industrial Technology, Institut Teknologi Nasional, Bandung, Indonesia ¹jasman@itenas.ac.id, ²rijalk64@mhs.itenas.ac.id

Abstract

Face recognition is one of the main challenges in the development of computer vision technology. This study aims to develop a face recognition system using a Faster R-CNN architecture, optimized through hyperparameter tuning. This research utilizes the "Face Recognition Dataset" from Kaggle, which comprises 2,564 face images across 31 classes. The development process involves creating bounding boxes using the LabelImg application and implementing the Grid Search method. The Grid Search is applied with predefined hyperparameter combinations (3 epochs [10, 25, and 50] \times 3 learning rates [0.001, 0.0001, and 0.00001] \times 3 optimizers [SGD, Adam, and RMS], resulting in 27 models). The evaluation metrics used were accuracy, precision, recall, and F1-score. The experimental results show that the selection of hyperparameters significantly affects the model performance. Based on the experimental results, the combination of the learning rate 0.00001, 50 epochs, and Adam optimizer yielded the highest accuracy and improvement of 8.33% compared to the baseline model. The results indicate that hyperparameter optimization enhances the ability of the model to recognize faces. Compared to conventional models, a Faster R-CNN performs better in detecting faces more accurately. Future research could further enhance the face recognition efficiency and accuracy by exploring other deep learning architectures and more advanced hyperparameter optimization techniques.

Keywords: face recognition; faster R-CNN; hyperparameter optimization; deep learning; grid search

How to Cite: J. Pardede and K. Rijal, "The Effect of Hyperparameters on Faster R-CNN in Face Recognition Systems", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 9, no. 3, pp. 436 - 448, May 2025. *Permalink/*DOI: https://doi.org/10.29207/resti.v9i3.6405

Received: February 19, 2025 Accepted: May 5, 2025 Available Online: May, 28, 2025

This is an open-access article under the CC BY 4.0 License Published by Ikatan Ahli Informatika Indonesia

1. Introduction

Face recognition is one of the primary applications in digital image analysis. This technology is used to identify and verify human faces using computer technology. In the context of image processing, a face recognition system analyzes the unique features of an individual's face [1]. With the advancement of technology, face recognition applications have been implemented in various fields, such as security, surveillance, and human-computer interaction. Accuracy is a crucial factor in face recognition, as it reflects the system's ability to correctly identify or verify an individual's identity [2].

Advancements in machine learning and deep learning have driven the development of more sophisticated techniques in face recognition [3]. Convolutional Neural Networks (CNN) have become one of the most widely used architectures due to their ability to automatically extract facial features [4]. However, despite the strong performance of CNNs, challenges remain, particularly in recognizing faces under varying conditions such as poor lighting, different facial expressions, or unusual orientations [5].

Faster R-CNN is an architecture that integrates object detection and classification into a single framework. This approach utilizes a Region Proposal Network (RPN) to generate object candidates, which are then further processed for classification and bounding box regression [6]. Although Faster R-CNN has proven to be effective, its performance heavily depends on the proper configuration of hyperparameters, such as learning rate, number of epochs, and optimizer type [7]. In the context of face recognition using Faster R-CNN, hyperparameter optimization can enhance the model's accuracy [8].

Previous studies have used default configurations or simple optimization methods in face recognition with Faster R-CNN [6]. This limits the model's potential in achieving its best performance. Furthermore, there has been limited research exploring the impact of hyperparameter variations on datasets with high variability, such as changes in position, lighting, and facial expressions.

This study proposes hyperparameter optimization using the Grid Search approach to enhance the performance of Faster R-CNN in face recognition tasks. The "Face Recognition Dataset" from Kaggle is utilized for this purpose [9] It is used as test data with complex condition variations. By selecting the optimal hyperparameter combinations, this study aims to contribute to improving face recognition accuracy.

2. Methods

Several face recognition studies use deep learning, as shown in Table 1. The previous studies have made significant contributions to the development of related methods and approaches that have been proposed. On [6] proposed the face detection method using Faster R-CNN, so to improve performance, this study proposes the impact of hyperparameter optimization on the Faster R-CNN architecture for face recognition.

Table 1. Related Work

No	Title	Method	Contribution
1	Deep Face Recognition: A Survey [2]	Deep Learning	Deep learning methods that can be applied to face recognition.
2	Deep Learning Convolutional Neural Network for Face Recognition: A Review [10]	Convolutional Neural Network	Discusses face recognition using deep learning techniques.
3	Recent Advances in Deep Learning Techniques for Face Recognition [5]	Deep Learning	Provides insights into other deep learning models relevant to face recognition.
4	Review of Deep Learning: concepts, CNN architectu res, challenges, application, future directions	Deep Learning	Understanding the fundamental concepts of Deep Learning and Convolutional Neural Networks (CNN).
5	A new face detection method based on Faster RCNN [6]	Faster RCNN	This paper proposes a new face detection method using Faster R-CNN.

2.1 Face Recognition

Face recognition is the process of identifying or verifying an individual based on facial features such as the distance between the eyes, nose, and mouth, the proportions of various facial elements (such as the width and height of the face, the contours and protrusions that form the individual's facial characteristics, skin color information, and the texture of the facial surface), and the overall shape of the face (including oval, square, or round) [2],[12]. Face recognition technology is commonly used in various

applications such as security, surveillance, and humancomputer interaction. In a more technical context, face recognition involves algorithms and machine learning methods to analyze and classify facial features [6]. Face recognition encompasses a range of technologies used to build face recognition systems, including face detection, facial landmarking, identity recognition, and image pre-processing. The face detection process works to locate the coordinate system of all faces within an image, while facial landmarking algorithms identify the positions of facial features within that coordinate system [13].

This study focuses on the application of a Faster R-CNN architecture with ResNet-50 as the Feature Pyramid Network (FPN) in a face recognition system. Faster R-CNN is a deep learning-based object detection method capable of providing precise face detection through the use of a Region Proposal Network (RPN). By leveraging ResNet-50, this model can extract deeper and more complex facial features, thereby improving identification accuracy.

Previous studies have shown that ResNet-50 has high capabilities in face classification. One study used ResNet-50 to explore facial features by utilizing a modified dataset with OpenCV, such as random brightness adjustments [14]. This study also discusses the development of face recognition technology prior to ResNet-50 by comparing methods such as Eigenfaces and Fisherfaces. The results showed that the ResNet-50-based model achieved the highest accuracy of 98.75%, demonstrating its robustness under various lighting conditions.

2.2 Faster R-CNN

Faster R-CNN is an object detection method that combines Region Proposal Networks (RPN) with Fast R-CNN to generate region proposals, classification, and bounding box regression [15]. As shown in Figure 1, this architecture works by processing the image through the backbone (ResNet-50) to generate a feature map. The RPN uses this feature map to generate anchor boxes, which are evaluated using Intersection Over Union (IoU) against the ground truth.



Figure 1. Faster R-CNN Architecture [15]

Anchors with high scores are further processed by Region Of Interest (ROI) Pooling or ROI Align to generate fixed-size features. These features are then classified to determine the object type and processed by the bounding box regressor to refine the coordinates. With components such as the Feature Pyramid Network (FPN), Faster R-CNN is capable of detecting objects at various scales, making it highly accurate for object detection tasks.

ResNet-50 is a deep neural network with 50 layers designed to handle degradation in deep networks and is well-known for its performance in image classification. [16],[17]. Feature Pyramid Network (FPN) enhances detection accuracy by combining features from multiple resolution levels to support multi-scale object detection [18].

Region Proposal Network (RPN) works by applying a sliding window to the feature map to generate anchor boxes at each location. These anchors are evaluated using IoU against the ground truth, where IoU > 0.7 is considered positive, IoU < 0.3 is considered negative, and the rest are ignored. This process uses a combined loss function: objectness loss to detect the presence of an object and bounding box regression loss to refine the anchor coordinates [19].

Fast R-CNN is used to classify the region proposals from the RPN and perform bounding box regression [20]. Fast R-CNN uses a convolutional neural network (CNN) to compute features for the entire image and all region proposals in a single processing step. Additionally, Fast R-CNN employs a RoI pooling layer to extract features from each region proposal, eliminating the need to re-crop the region proposals from the image. Subsequently, Fast R-CNN uses fullyconnected layers (fcl) at the end of the network to perform object detection and classification on the region proposals [20].

An anchor is a reference box on the feature map with specific scale and aspect ratio used to predict the locations of varying objects [15] as shown in Figure 2.



Figure 2. Anchor Boxes

An anchor is centered on the sliding window and has specific scale and aspect ratio, as shown in Figure 2. By

default, the anchor box has 3 scales and 3 aspect ratios, resulting in k = 9 anchors at each sliding position [15].

Intersection over Union (IoU) is a metric used to measure how well an object detection model's prediction matches the ground truth (true label). IoU is calculated by comparing the area of intersection between the predicted box and the ground truth box to the area of their union. The intersection is the overlapping area between the predicted box and the ground truth box. The union is the total combined area of both boxes. IoU is used to determine whether an anchor (candidate box) should be retained as a bounding box proposal. Typically, an IoU > 0.7 is considered positive, while IoU < 0.3 is considered negative [21]. The IoU calculation is based on Equation 1.

$$IoU = \frac{Area \ of \ Overlap}{Area \ of \ Union} \tag{1}$$

Where:

Area of Overlap is the area of intersection between the model's prediction (predicted bounding box) and the ground truth label (actual bounding box).

Area of Union is the combined area of the predicted bounding box and the ground truth bounding box (without double-counting the overlapping area).

IoU Interpretation:

IoU = 0: There is no overlap between the prediction and ground truth.

IoU = 1: The predicted bounding box and the ground truth perfectly match.

IoU values are generally considered good if IoU > 0.5, but this threshold may vary depending on the application.

Figure 3 shows Non-Maximum Suppression (NMS), which is an algorithm that retains the detection with the highest score and eliminates duplicate detections [22].



Figure 3. Non-Maximum Suppression

NMS is used in object detection to filter prediction results, ensuring that only the best bounding box (with the highest confidence score) is retained for each detected object.

Region Of Interest Pooling g is used to reduce features from region proposals to a fixed size. However, in

modern implementations, ROI Pooling is often replaced by ROI Align to improve precision by better preserving spatial relationships through bilinear interpolation [23].

The bounding box regressor is a component that optimizes the bounding box parameters to align with the ground truth [24].

The classifier is a model that classifies data based on learned patterns to determine the object category [15].

2.3 Hyperparameter Optimization

Hyperparameter optimization is the process of finding optimal values for parameters set before the model training begins. Hyperparameters have a significant impact on detection accuracy and training speed in the context of object detection [25], [7]. In hyperparameter optimization, several key components need to be considered to improve the performance of a face recognition model using Faster R-CNN, including:

Grid search is a method used to try various combinations of parameters to be tested, with each combination being evaluated to determine which one yields the best performance [26].

An epoch is an algorithm that determines how many times the model will iterate through the entire training dataset. Each epoch allows every dataset sample to update the model's parameters [27].

The effect of the number of epochs on model performance can be explained through the training process in machine learning. An epoch refers to one complete cycle in which the model is trained using the entire training dataset. Each epoch provides the model with an opportunity to update its weights and parameters based on the errors produced in previous predictions. During training, the model learns from the data by optimizing the loss function. As the number of epochs increases, the model can correct errors and learn more effectively from the available data, potentially improving accuracy. However, setting the number of epochs too high may lead to overfitting. This occurs when the model becomes overly adapted to the training data, thereby losing its ability to generalize to new, unseen data [27].

The learning rate is a parameter in machine learning algorithms that controls how much the model's weights are adjusted during the training process [28]. In the use of learning rate, the model demonstrated better performance compared to a fixed learning rate, with higher AUC values observed for the dynamic learning rate. This indicates that selecting an appropriate learning rate can enhance the effectiveness of the optimizer used during model training. Therefore, adjusting the learning rate is a crucial step toward achieving optimal results [28].

An optimizer is an algorithm used to update the model's weights in a neural network during the training process. The goal is to minimize the loss function and improve the model's accuracy. Commonly used optimizers include Stochastic Gradient Descent (SGD), Adam, and RMSprop [25]. Each optimizer has a different way of operating and performance stability, as well as an optimal learning rate that affects the final results [29].

The use of optimizers has a significant impact on the final results of the model training process. Each optimizer may yield different performance outcomes depending on the learning rate applied. Based on the conducted experiments, SGD demonstrated the best performance at a learning rate of 0.1, achieving a test score of 74.80% and a test cost of 72.55%. In contrast, RMSProp and Adam performed poorly at the same learning rate. At a learning rate of 0.01, RMSProp outperformed both SGD and Adam, although the results were still below expectations. Subsequently, at a learning rate of 0.001, both RMSProp and Adam delivered improved performance, whereas SGD remained unsuitable for this learning rate. These findings indicate that selecting the appropriate optimizer and tuning the learning rate are key factors in enhancing model performance [29].

2.4 Model Evaluation

Evaluation metrics in the context of object detection algorithms refer to the tools used to assess and characterize the performance of a detection system. These metrics are often based on the concept of true positives, which refers to prediction units that successfully detect an object [30]. Evaluation metrics are used to assess how well the model performs object detection. These metrics help measure how effectively an algorithm can detect objects in images or videos [31]. The evaluation metrics used to measure the performance of the model include:

Accuracy is the correct prediction that depends on whether the bounding box and object class match the ground truth, as shown in Equation 2.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$
(2)

Precision measures how much of the area that is correctly part of the face, compared to the pixels incorrectly labeled as a face, as shown in Equation 3.

$$PREC = \frac{TC}{TC + FC} \tag{3}$$

Recall measures how much of the area that is actually part of the face is correctly predicted by the model, compared to all the pixels that make up the face, as shown in Equation 4.

$$REC = \frac{TP}{TP + FN} \tag{4}$$

F1-Score combines both recall and precision. This metric is useful when aiming to balance between precision and recall, providing a single value that accounts for both, as shown in Equation 5.

(5)

 $F1 = 2 \times \frac{precision \times recall}{precision + recall}$

2.5 Dataset



Figure 4. Face Recognition Dataset

Figure 4 shows examples of images from the "Face Recognition Dataset" from Kaggle, consisting of 2,564 face images with 31 classes [9]. This dataset includes variations in face position, expression, and lighting to ensure that the model can recognize faces under different conditions. Each image is labeled with a bounding box using the LabelImg application, which generates an XML file containing the coordinates of the face and the class label.

The dataset used in this study has previously been utilized in a research project involving facial recognition using the ResNet-50 architecture [32]. In that study, a CNN-based facial classification model was trained using the same dataset. The evaluation results demonstrated relatively high performance on the training and validation sets, with a training accuracy of 98.09% and a validation accuracy of 98.09%. However, when tested on the unseen data (testing set), the model's accuracy dropped significantly to 67.76%.

These results indicate that although the model was able to learn patterns effectively during training and validation, it exhibited a considerable degree of overfitting, as it failed to maintain its performance on the testing data. This outcome serves as a motivation to explore alternative approaches such as face detection using Faster R-CNN to improve the model's generalization capability in facial recognition tasks on the same dataset.

In the face recognition process using Faster R-CNN, the first step is to annotate the dataset with bounding boxes to mark the location of faces in the images. The application used for this research is LabelImg, a GUI-based annotation tool.

Figure 5 shows the annotation process. Each image in the dataset is opened one by one, and a bounding box is drawn around the face area using the selection tool in LabelImg. After the bounding box is created, the appropriate label, such as "Elizabeth Olsen," is assigned to each face in the image. LabelImg saves the annotations in the Pascal VOC format (.xml).

After all images in the dataset have been annotated, Figure 6 shows the file generated by LabelImg, which contains important information such as bounding box coordinates, object labels, and image size. This file will be used as the ground truth when training the Faster R-CNN model, helping the neural network recognize facial patterns based on manually marked bounding boxes. The dataset was then divided into three parts: 70% for training, 20% for validation, and 10% for testing.



Figure 5. LabelImg Application

```
<annotation>
       <folder>Alexandra Daddario</folder>
       <filename>Alexandra
Daddario 0.jpg</filename>
       TUGAS
AKHIR 2024\archive\ Original Images\Original
Images\Alexandra
                         Daddario\Alexandra
Daddario 0.jpg</path>
       <source>
              <database>Unknown</database>
       </source>
       <size>
               <width>853</width>
               <height>1280</height>
               <depth>3</depth>
       </size>
       <segmented>0</segmented>
       <object>
               <name>Alexandra
Daddario</name>
               <pose>Unspecified</pose>
               <truncated>0</truncated>
               <difficult>0</difficult>
               <br/>
<br/>
hodbox>
                      <xmin>303
                      <ymin>211</ymin>
                      <xmax>587</xmax>
                      <vmax>511</vmax>
               </bndbox>
       </object>
</annotation>
```

Figure 6. Annotation results of the LabelImg application

2.6 Flowchart and System Flow

In Figure 7, the Faster R-CNN flowchart illustrates the object detection workflow consisting of several key stages. The process begins with the input image, followed by pre-processing (normalization and resizing). Feature extraction is performed on each image using the backbone (ResNet-50 with FPN) to generate feature maps at various scales. The next stage is the Region Proposal Network (RPN), which uses a

sliding window to generate anchor boxes with different scales and ratios. Anchors are evaluated using Intersection over Union (IoU). For IoU > 0.7, the anchor is considered positive, for IoU < 0.3, it is considered negative, and the rest are ignored. The Non-Maximum Suppression (NMS) process filters out

redundancies, resulting in approximately 2,000 of the best proposals. RPN uses a loss function consisting of objectness loss (to differentiate between objects and non-objects) and bounding box regression loss (to refine coordinates).



Figure 7. Flowchart of Face Recognition System Using Faster R-CNN Architecture

The region proposals filtered by the NMS are processed by RoI Pooling/RoI Align to generate features with fixed sizes. These features were then passed to the classifier to determine the object class and bounding box regressor to refine the coordinates. The final output is the object detection, which includes both class information and location coordinates of the detected objects.



Figure 8. Block diagram of Face Recognition System Using Faster R-CNN Architecture

Figure 8 shows the block diagram of the Face Recognition System using the Faster R-CNN architecture. In the training and validation sections, the process begins with facial image data for training and validation as inputs. These data then underwent preprocessing, which included resizing the images to 512 pixels, normalizing the pixel values to a range of 0-1, as well as labeling the bounding boxes, and splitting the dataset. After pre-processing, hyperparameter optimization was carried out, involving the selection of the learning rate, number of epochs, and type of optimizer used. Once the hyperparameters were optimized, the model was trained using Faster R-CNN to generate the best face recognition model. The performance of the trained model was evaluated using evaluation metrics including accuracy, precision, recall, and F1-score.

The testing section begins with facial image data for testing as input. Similar to the training stage, the images undergo pre-processing, which includes resizing to 512 pixels and normalizing the pixel values. The processed data is then tested using the previously trained Faster R-CNN model. The result of this testing is the face detection, which includes classification and bounding box determination on the images. Finally, the face detection results are evaluated using the same metrics as in the training stage, namely accuracy, precision, recall, and F1-score.

2.7 Training Scheme

In the model training process using grid search, as shown in Table 2.

Table 2. Training Scheme

Model	Learning Rate	Epoch	Optimizer
Model 1	0.001	10	SGD
Model 2	0.001	10	ADAM
Model 3	0.001	10	RMS
Model 4	0.001	25	SGD
Model 5	0.001	25	ADAM
Model 6	0.001	25	RMS
Model 7	0.001	50	SGD
Model 8	0.001	50	ADAM
Model 9	0.001	50	RMS
Model 10	0.0001	10	SGD
Model 11	0.0001	10	ADAM
Model 12	0.0001	10	RMS
Model 13	0.0001	25	SGD
Model 14	0.0001	25	ADAM
Model 15	0.0001	25	RMS
Model 16	0.0001	50	SGD
Model 17	0.0001	50	ADAM
Model 18	0.0001	50	RMS
Model 19	0.00001	10	SGD
Model 20	0.00001	10	ADAM
Model 21	0.00001	10	RMS
Model 22	0.00001	25	SGD
Model 23	0.00001	25	ADAM
Model 24	0.00001	25	RMS
Model 25	0.00001	50	SGD
Model 26	0.00001	50	ADAM
Model 27	0.00001	50	RMS

The first step is to set up the model architecture, in this case using Faster R-CNN, which is configured to accept parameters such as learning rate, optimizer, and the number of epochs. Next, the hyperparameter space to be explored is determined, including learning rate [0.001, 0.0001, 0.00001], epochs [10, 25, 50], and optimizer [SGD, ADAM, RMS]. Grid search is implemented with various hyperparameter combinations, resulting in a total of 27 combinations.

Each model is trained using the training dataset to build a model that corresponds to the given hyperparameters. Each built model is then evaluated using the validation image dataset to achieve the best accuracy performance.

3. Results and Discussions

3.1 Training Results

Based on the training results of various developed models, the performance of each model was evaluated using several metrics, namely Accuracy, Precision, Recall, F1-Score, and Loss. Table 3 shows that several models exhibited no performance at all, with all metrics scoring 0.000, such as models 3, 5, 6, 8, 9, 10, 12, 13, 15, 18, 19, 22, and 25. This indicates that these models failed during the learning process or were unable to recognize patterns within the provided data. In addition, there are models with low performance, such as models 2 and 21, which have very low Accuracy and F1-Score Several models achieved moderate values. performance, with Accuracy values ranging between 0.75 and 0.95, including models 1, 4, 11, 14, 16, 20, and 24.

There are five (5) models that demonstrated excellent performance, namely models 7, 17, 23, 26, and 27, with Accuracy scores above 0.95, F1-Scores close to 1.000, and very small Loss values, below 0.05. These models were able to correctly identify the majority of the data, resulting in only a few prediction errors. The bestperforming models were models 7, 23, and 26, which achieved perfect scores of 1.000 in Accuracy, Precision, Recall, and F1-Score, indicating that they were able to recognize all data flawlessly without any errors. Moreover, model 26 recorded the lowest Loss value (0.023), making it the most optimal model in this experiment.

Regarding the models that failed to demonstrate any performance (all metrics equal to 0.000), several hypotheses can be proposed to explain the cause. One of the possible reasons is the inappropriate combination of hyperparameters (such as learning rate, number of epochs, and optimizer), which prevented the models from learning from the data. The selection of an excessively large learning rate, an insufficient number of epochs, or an unsuitable optimizer may have caused the models to be unable to capture patterns from the data. This hypothesis is supported by several previous studies. Choi et al. (2019) emphasized the importance of optimizer sensitivity to hyperparameter tuning protocols, which can significantly influence model performance [33]. Nurdiati et al. (2022) also reported that optimizers such as Adam, Nadam, and AdamW performed better than other optimizers in facial expression recognition tasks [34]. Furthermore, Kim et al. (2022), in the AdaFace study, demonstrated that adaptive approaches to input quality can enhance model performance, indicating that low-quality input or poor initial weights may result in model failure [35]. Ali and Kumar (2022) also highlighted the significance of selecting the appropriate architecture and activation functions in achieving optimal performance in face recognition systems [36].

Thus, the appropriate selection and combination of hyperparameters, proper data preprocessing, and optimal choice of model architecture and optimizer are crucial in determining the success of model training in facial recognition tasks.

Model	Training				
	Accuracy	Precision	Recall	F1-Score	Loss
Model 1	0,857	0,462	0,429	0,500	0.214
Model 2	0,614	0,047	0,077	0,059	0.224
Model 3	0,000	0,000	0,000	0,000	0,000
Model 4	0,767	0,673	0,639	0,617	0.148
Model 5	0,000	0,000	0,000	0,000	0,000
Model 6	0,000	0,000	0,000	0,000	0,000
Model 7	0,995	0,996	0,933	0,994	0.049
Model 8	0,000	0,000	0,000	0,000	0,000
Model 9	0,000	0,000	0,000	0,000	0,000
Model 10	0,000	0,000	0,000	0,000	0,000
Model 11	0,810	0,746	0,643	0,644	0.108
Model 12	0,037	0,001	0,032	0,002	2.904
Model 13	0,000	0,000	0,000	0,000	0,000
Model 14	0,838	0,861	0,838	0,832	0.072
Model 15	0,000	0,000	0,000	0,000	0,000
Model 16	0,941	0,630	0,667	0,647	0.231
Model 17	0,983	0,982	0,980	0,981	0.047
Model 18	0,000	0,000	0,000	0,000	0,000
Model 19	0,000	0,000	0,000	0,000	0,000
Model 20	0,855	0,760	0,684	0,694	0.132
Model 21	0,519	0,275	0,278	0,227	0.137
Model 22	0,000	0,000	0,000	0,000	0,000
Model 23	0,999	1,000	0,999	0,999	0.039
Model 24	0,839	0,871	0,825	0,832	0.076
Model 25	0,000	0,000	0,000	0,000	0,000
Model 26	1,000	1,000	1,000	1,000	0.023
Model 27	0,976	0,979	0,973	0,975	0.045

Table 3. Training Results

This explanation reinforces that the combination of specific parameters a small learning rate (0.00001), 50 training epochs, and the use of the Adam optimizer significantly contributed to the optimal performance achieved by Model 26. These findings are also supported by several previous studies that have demonstrated how proper parameter selection directly influences model performance in face recognition tasks.

First, the use of the Adam optimizer has been proven effective in various studies [37] showed that Adam achieved up to 97.93% accuracy in a 2.5D face recognition system based on the EfficientNet architecture. This highlights Adam's strengths in automatically adjusting the learning rate and avoiding the vanishing gradient problem, making it highly suitable for deep learning models in this domain.

Second, a small learning rate allows the model to learn gradually and stably, enabling more precise convergence [38] reported that a small learning rate tends to produce lower loss values and more stable training, particularly when used with optimizers such as Adam or AdamW. This aligns with the results of Model 26, which demonstrated a very low loss value (0.023) and perfect performance across all evaluation metrics.

Third, training the model for 50 epochs proved to be an optimal choice in this experiment. This number of epochs is sufficient for the model to capture patterns in the data without overfitting or underfitting. Related literature has emphasized that too few epochs may lead to underfitting, while too many may result in overfitting, thereby reducing the model's generalization ability [39], [40].

Fourth, these findings are further supported by a study conducted [34], which concluded that Adam outperformed other optimizers in facial expression recognition tasks due to its ability to accelerate convergence and maintain training stability.

Finally, the success of Model 26 can serve as a benchmark for evaluating other models in the experiment that showed poor or failed performance (such as Models 3, 5, 6, etc.). The suboptimal performance of these models is likely due to less effective parameter configurations, such as a larger learning rate or the use of less adaptive optimizers like SGD without momentum.

3.2 Training Model Performance



Figure 9. Performance comparison of training models

Figure 9 shows a comparison of the best models based on the parameters used, where a learning rate (lr) of 0.00001 provided the most optimal results compared to 0.001 or 0.0001. Models with a larger learning rate, such as 0.001 (Model 7), achieved high accuracy (0.995), but the resulting loss was higher than that of models with smaller learning rates. Meanwhile, a learning rate of 0.00001 (Model 23 and Model 26) demonstrated the best performance, with Model 26 even achieving perfect accuracy (1.000), although there was an initial indication of overfitting.

From the number of epochs, Model 23 with 25 epochs was sufficient to achieve an accuracy of 0.999 with a smaller loss (0.039). In contrast, Model 26 with 50 epochs achieved perfect results, although it was initially suspected of overfitting. However, after testing on the testing data, Model 26 demonstrated the highest evaluation metrics compared to other models, indicating that this model was still able to generalize well to unseen data.

The choice of optimizer also affects the model's performance. ADAM proved to be the best choice, as seen in Model 23 and Model 26, which achieved optimal results with low loss. The use of the SGD optimizer (Model 7) was still quite good but less optimal compared to ADAM, while RMSProp (Model 27) showed a performance with an accuracy of 0.976, indicating that this optimizer was less effective in this case.

Although Model 23 was initially considered the best choice based on training results, testing evaluation showed that Model 26 is the most optimal model, as it has a small learning rate (0.00001), a relatively high number of epochs (50), and the ADAM optimizer, which helped the model learn better without losing generalization. The small learning rate allowed the model to update weights gradually without overshooting the optimal solution, while the higher number of epochs enabled the model to capture more complex patterns. The lower loss (0.023) compared to other models also indicates that Model 26 is more stable and has better optimization.

3.3 Testing Model Performance

Figure 10 shows a comparison between Model 7 (baseline) and Model 26 (best), indicating that Model 26 performs better in all evaluation metrics: accuracy, precision, recall, and F1-Score. The key difference between these two models lies in the choice of learning rate and optimizer, which have a significant impact on the stability and effectiveness of the model's training process.

In Model 7, a learning rate of 0.001 is used with the SGD (Stochastic Gradient Descent) optimizer. This relatively large learning rate causes the weight updates to be made with larger steps, which risks the model skipping the optimal point and struggling with convergence. Additionally, the use of SGD as the optimizer has the drawback of high gradient oscillations, particularly if not combined with the proper momentum. This can cause the model to struggle in finding the optimal loss minimum, resulting in suboptimal performance.



Figure 10. Performance comparison of testing models

Meanwhile, Model 26 uses a much smaller learning rate of 0.00001 and the ADAM (Adaptive Moment Estimation) optimizer. The smaller learning rate allows for a smoother and more stable learning process, avoiding the risk of overshooting during the optimization process. The use of ADAM as the optimizer also offers advantages, as it combines the best features of Momentum SGD and RMSProp, making it more adaptive in adjusting learning based on the gradients obtained. ADAM has a mechanism that accelerates convergence without experiencing high oscillations like SGD, making it a better choice for deep learning models like Faster R-CNN.

Although both models use the same number of epochs (50), the performance difference is significantly influenced by the choice of learning rate and optimizer. The Figure 10 shows that Model 26 achieves higher values for accuracy, precision, recall, and F1-Score compared to Model 7, indicating that this model is more optimal in recognizing faces with a better balance between precision and recall. Therefore, Model 26 is more effective at detecting faces without making too many mistakes in classifying faces as correct or incorrect.

3.4 Model Evaluation

Table 4 presents the Classification Report, which displays the evaluation of the model's performance in facial recognition across various individuals. The evaluation metrics used include precision, recall, F1-Score, and Support for each class (individual name). Based on the information provided earlier, these are the evaluation results using Model 26, which achieved an accuracy of 1.000 on the training data. However, despite the model performing exceptionally well during training, there are variations in performance during testing, as seen in the values of precision, recall, and F1-Score for different individuals.

Images with good performance, such as those of Dwayne Johnson, Ellen Degeneres, and Lisa Kudrow, show precision, recall, and F1-Score values of 1.00, meaning the model recognizes them extremely well without errors. This high performance may be due to distinct facial features, such as unique facial structures that are easily distinguishable from other individuals. Additionally, the consistent image quality, with good lighting and high resolution, enables the model to capture facial features accurately.

Name	Precision	Recall	F1-Score	Support
Akshay Kumar	0.62	0.71	0.67	7
Alexandra Daddario	0.83	0.94	0.88	16
Alia Bhatt	0.82	0.78	0.80	18
Amitabh Bachchan	1.00	0.93	0.96	14
Andy Samberg	0.82	0.88	0.85	16
Anushka Sharma	0.88	0.93	0.90	15
Billie Eilish	1.00	0.95	0.97	19
Brad Pitt	0.84	0.84	0.84	19
Camila Cabello	0.95	0.83	0.88	23
Charlize Theron	0.93	0.68	0.79	19
Claire Holt	0.76	0.95	0.84	20
Courtney Cox	0.93	0.93	0.93	14
Dwayne Johnson	1.00	1.00	1.00	12
Elizabeth Olsen	0.91	1.00	0.95	21
Ellen Degeneres	1.00	1.00	1.00	15
Henry Cavill	0.90	0.95	0.92	19
Hrithik Roshan	0.90	0.90	0.90	20
Hugh Jackman	0.83	0.87	0.85	23
Jessica Alba	0.94	0.89	0.91	18
Kashyap	0.67	1.00	0.80	4
Lisa Kudrow	1.00	1.00	1.00	9
Margot Robbie	0.92	0.79	0.85	14
Marmik	1.00	0.80	0.89	5
Natalie Portman	0.86	1.00	0.93	19
Priyanka Chopra	0.91	0.88	0.89	24
Robert Downey Jr	0.95	0.91	0.93	22
Roger Federer	0.94	0.94	0.94	18
Tom Cruise	0.71	0.77	0.74	13
Vijay Deverakonda	0.96	0.89	0.92	27
Virat Kohli	0.88	0.78	0.82	9
Zac Efron	1.00	1.00	1.00	21
Accuracy			0.90	513
Macro Avg	0.89	0.89	0.89	513
Weighted Avg	0.90	0.90	0.90	513

Table 4. Classification Report

On the other hand, images with lower performance, such as those of Akshay Kumar, Kashyap, and Charlize Theron, exhibit lower values for precision, recall, and F1-Score compared to other individuals. One of the main factors contributing to this decreased performance is the smaller number of images in the dataset, with Akshay Kumar having only 7 images and Kashyap 5, limiting the model's ability to learn facial patterns effectively. Additionally, other factors such as significant pose variation, uneven lighting, or low-quality images can further complicate accurate identification.

From this analysis, the model performs very well on individuals with distinct facial features, good image quality, and sufficient data. However, individuals with lower-quality images, significant pose variation, or limited data experience a drop in accuracy. This indicates that while Model 26 shows high performance, there are still factors within the dataset that could be improved to enhance its overall performance. Improving dataset diversity, image quality, and ensuring a sufficient number of samples for each individual could help the model generalize better across various conditions.



Figure 11. Confusion matrix

From the evaluation results, the model has an accuracy of 0.90, meaning that 90% of the model's predictions match the correct labels. Overall, the macro average and weighted average values are 0.89 and 0.90, respectively, indicating that the model performs relatively consistently across all classes. To ensure dataset balance, a sample distribution analysis was conducted. The dataset consists of 31 classes, with the number of samples per class ranging from 4 to 27 images. This analysis revealed that the dataset is imbalanced, with some classes having fewer samples than others. Therefore, a weighted loss method was employed to ensure that classes with fewer samples are proportionally considered during model training. Additionally, the model was evaluated using Intersection over Union (IoU). The best model showed an average IoU of 0.902, while models with lower performance had an IoU around 0.861.

The confusion matrix in Figure 11 shows the performance of Model 26 for face recognition in classifying 31 different classes. Each row of the matrix represents the true labels, while each column represents the predicted labels by the model. The values on the main diagonal represent the number of correct predictions (true positives), with higher values indicating better performance in recognizing the correct faces. For example, for the class Elizabeth Olsen, the model successfully identified her face 21 times, indicating good performance for that particular class. However, there are some misclassifications indicated by the numbers outside the main diagonal. For example, for the class Akshay Kumar, although there were 5 correct predictions, the model also misclassified his face into other classes multiple times. These misclassifications can occur due to similarities in faces between individuals or insufficient training data, which limits the model's ability to distinguish facial features accurately.

In terms of visualization, the darker the color of the boxes on the main diagonal, the higher the number of correct predictions in that category. Conversely, lighter colors outside the main diagonal indicate small mispredictions. Overall, the model performs quite well as most predictions are on the main diagonal, but there are still a few errors that need to be addressed.

Figure 12 shows the testing results of the best model, Model 26, in detecting and classifying faces of several individuals. Each sample in the image displays a person's face with the model's prediction and the original label. Out of the six test images, Model 26 correctly predicted the faces, and the generated bounding boxes accurately identified the faces corresponding to the original labels. This indicates that Model 26 performs very well in face recognition, which aligns with the previous evaluation results where the model achieved high precision, recall, and F1-Score (even reaching 1.00 for some individuals).



Figure 12. Face recognition results using Model 26

This high accuracy can be attributed to several factors, such as distinct facial features of the individuals, uniform lighting in the images, and sufficient data during training. For example, the faces of Camila Cabello and Zac Efron are recognized very well, likely because the model has been trained on a sufficient number of images of them with representative variations.

However, even though this model shows excellent results, it is important to consider the potential for dataset bias for instance, if certain individuals have fewer images in the training data, the model might not perform as well for them. To further improve generalization, additional data could be added to ensure the model remains accurate under various lighting conditions, poses, and facial expressions.

As shown in Figure 13, the misclassification of faces observed in the image is likely caused by several factors related to the dataset used. One of the primary factors is data imbalance, in which the number of images for each individual in the dataset is unevenly distributed. If certain individuals have significantly more samples than others, the model tends to classify faces with similar features into a more dominant category during training. This can lead to misidentification, particularly when the model encounters a face that is underrepresented in the training data.



Figure 13. Face Recognition errors

Additionally, similar facial features between different individuals another major cause are of misclassification. Face recognition models often rely on features such as facial shape, bone structure, or accessories (e.g., glasses and hats) to make classifications. Misclassifications, such as identifying Brad Pitt as Akshav Kumar or Tom Cruise as Henry Cavill, can occur due to the strong resemblance between their facial features in the images used. If the model is not trained with a sufficiently diverse set of images for each individual, these errors are more likely to occur.

Another influencing factor is the variation in lighting conditions and facial angles within the dataset. If the model is trained using images with uniform lighting and limited facial angles, its performance declines when recognizing faces under dim lighting conditions or from different angles. As seen in some misclassification examples, faces captured under well-lit conditions are easier to recognize than those captured under dim lighting or tilted positions. This indicates that the model struggles to generalize variations in facial appearance across different conditions.

4. Conclusions

The results of this study indicate that the combination of hyperparameters in "Face Recognition Using Faster R-CNN Architecture with Hyperparameter Optimization," such as learning rate, number of epochs, and optimizer type, significantly influences the performance of the model. The model with a learning rate of 0.00001, 50 epochs, and Adam optimizer (Model 26) achieved the best performance based on evaluation metrics such as accuracy, precision, recall, and F1-score on the test data.

From the confusion matrix, it is evident that Model 26 has a high accuracy in classifying faces, with minimal

errors in distinguishing between classes of faces that share similar lighting or expressions. This indicates that the model is effective in recognizing facial patterns but still faces challenges in differentiating faces with similar features. The IoU calculation showed that Model 26 had an average IoU value of 0.902. This indicates that the bounding boxes generated by the detection process closely matched the ground truth. The higher the IoU value, the more accurate the model is in placing the face detection boxes at the correct location.

Visual analysis of the detection results showed that the model performed better in recognizing faces under good lighting conditions and when the face was in a straightforward position, as shown in Figure 12. However, under low-light conditions or when the face is tilted, detection errors still occur. This suggests that image characteristics such as lighting, angle, and image resolution significantly affect the performance of the model. From various experiments, it was observed that the selection of a learning rate of 0.00001 played a crucial role in determining the stability of the training process. A larger learning rate tends to make it difficult for the model to converge because of drastic weight changes, whereas a smaller learning rate slows down the learning process. A value of 0.00001 provides an optimal balance, allowing the model to learn gradually without overfitting or underfitting. Furthermore, the use of the Adam optimizer improves the training stability compared to SGD.

Overall, this study demonstrates that hyperparameter optimization in the Faster R-CNN architecture significantly impacts facial recognition performance. Proper hyperparameter selection can improve the accuracy, ensure optimal bounding box detection, and adapt the model to varying image characteristics.

Acknowledgment

The author would like to thank the LPPM at Institut Teknologi Nasional (Itenas) Bandung for supporting this research project. The author declares no conflict of interest.

References

- [1] F. Boutros, N. Damer, F. Kirchbuchner, and A. Kuijper, "ElasticFace: Elastic Margin Loss for Deep Face Recognition," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2022-June, pp. 1577–1586, 2022, doi: 10.1109/CVPRW56347.2022.00164.
- [2] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021, doi: 10.1016/j.neucom.2020.10.081.
- P. Payal and M. M. Goyani, "A comprehensive study on face recognition: methods and challenges," *Imaging Sci. J.*, vol. 68, no. 2, pp. 114–127, 2020, doi: 10.1080/13682199.2020.1738741.
- [4] M. Feurer and F. Hutter, "Parameter Optimization, (eds) Automated Machine Learning," *The Springer Series on Challenges in Machine Learning. Springer, Cham.*, doi: 10.1007/978-3-030-05318-5_1.
- [5] M. T. H. Fuad et al., "Recent advances in deep learning techniques for face recognition," *IEEE Access*, vol. 9, no.

July, pp. 99112–99142, 2021, doi: 10.1109/ACCESS.2021.3096136.

- [6] H. Yan, X. Wang, Y. Liu, Y. Zhang, and H. Li, "A new face detection method based on Faster RCNN," *J. Phys. Conf. Ser.*, vol. 1754, no. 1, 2021, doi: 10.1088/1742-6596/1754/1/012209.
- [7] M. Zhou, B. Li, and J. Wang, "Optimization of Hyperparameters in Object Detection Models Based on Fractal Loss Function," *Fractal Fract.*, vol. 6, no. 12, 2022, doi: 10.3390/fractalfract6120706.
- [8] J. Selvaganesan et al., "Enhancing face recognition performance: a comprehensive evaluation of deep learning models and a novel ensemble approach with hyperparameter tuning," *Soft Comput.*, 2024, doi: 10.1007/s00500-024-09954-y.
- [9] Kaggle, "Face Recognition Dataset," 2021, [Online]. Available: https://www.kaggle.com/datasets/vasukipatel/face-

recognition-dataset/data

- [10] R. J.Hassan and A. M. Abdulazeez, "Deep Learning Convolutional Neural Network for Face Recognition: A Review," *Int. J. Sci. Bus.*, vol. 5, no. 2, pp. 114–127, 2021, doi: 10.5281/zenodo.4471013.
- [11] L. Alzubaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 53, 2021, doi: 10.1186/s40537-021-00444-8.
- [12] S. B. Mane, N. Shah, V. Garje and A. Tejwani, "A Comprehensive Survey of Face Recognition Advancements," 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2024, pp. 1-5, doi: 10.1109/ICCUBEA61740.2024.10774962.
- [13] L. Li, X. Mu, S. Li, and H. Peng, "A Review of Face Recognition Technology," *IEEE Access*, vol. 8, pp. 139110– 139120, 2020, doi: 10.1109/ACCESS.2020.3011028.
- [14] J. Liu, "Face recognition technology based on ResNet-50," *Appl. Comput. Eng.*, vol. 39, no. 1, pp. 160–165, 2024, doi: 10.54254/2755-2721/39/20230593.
- [15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2015, doi: 10.1109/TPAMI.2016.2577031.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770– 778, 2016, doi: 10.1109/CVPR.2016.90.
- [17] B. Mandal, A. Okeukwu, and Y. Theis, "Masked Face Recognition using ResNet-50," arXiv, 2021, doi: 10.48550/arXiv.2104.08997
- [18] G. Ghiasi, T. Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 7029–7038, 2019, doi: 10.1109/CVPR.2019.00720.
- [19] K. H. Shih, C. Te Chiu, J. A. Lin, and Y. Y. Bu, "Real-Time Object Detection with Reduced Region Proposal Network via Multi-Feature Concatenation," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 6, pp. 2164–2173, 2020, doi: 10.1109/TNNLS.2019.2929059.
- [20] R. Girshick, "Fast R-CNN," Proc. IEEE Int. Conf. Comput. Vis., pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.
- [21] S. A. K. Mohammed, A. H. A. Rahman, M. A. Bakar, and M. Z. A. Razak, "An Efficient Intersection Over Union Algorithm with Angle Orientation for an Improved 3D Object Detection," *6th IEEE Int. Conf. Artif. Intell. Eng. Technol. IICAIET 2024*, pp. 312–316, 2024, doi: 10.1109/IICAIET62352.2024.10730667.
- [22] J. Hosang, R. Benenson, and B. Schiele, "Learning nonmaximum suppression," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition*, CVPR 2017, pp. 6469–6477, 2017, doi: 10.1109/CVPR.2017.685.
- [23] M. -C. Roh and J. -y. Lee, "Refining faster-RCNN for

accurate object detection," 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), Nagoya, Japan, 2017, pp. 514-517, doi: 10.23919/MVA.2017.7986913.

- [24] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 658– 666, 2019, doi: 10.1109/CVPR.2019.00075.
- [25] T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," arXiv, 2020, doi: 10.48550/arXiv.2003.05689
- [26] A. Prasetya, C. Fatichah, and U. L. Yuhana, "Parsing the semantic structure of Indonesian math word problems using the recursive neural network," *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 5, no. 2, pp. 106–115, 2019, doi: 10.26594/register.v5i2.1537.
- [27] C. Hu, P. Coen-Pirani, and X. Jiang, "Empirical Study of Overfitting in Deep FNN Prediction Models for Breast Cancer Metastasis," *arXiv*, 2022, doi: 10.48550/arXiv.2208.02150.
- [28] A. Johny and K. N. Madhusoodanan, "Dynamic Learning Rate in Deep CNN Model for Metastasis Detection and Classification of Histopathology Images," *Comput. Math. Methods Med.*, vol. 2021, 2021, doi: 10.1155/2021/5557168.
- [29] D. Irfan, T. S. Gunawan, and W. Wanayumini, "Comparison Of SGD, Rmsprop, and Adam Optimation In Animal Classification Using CNNs," *Int. Conf. Inf. Sci. Technol. Innov.*, vol. 2, no. 1, pp. 45–51, 2023, doi: 10.35842/icostec.v2i1.35.
- [30] M. Breton and P. Eng, "Overview of two performance metrics for object detection algorithms evaluation," *Def. Res. Dev. Canada Ref. Doc.*, no. December, 2019.
- [31] R. Padilla, S. L. Netto, and E. A. B. Da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," *Int. Conf. Syst. Signals, Image Process.*, vol. 2020-July, no. July, pp. 237–242, 2020, doi: 10.1109/IWSSIP48289.2020.9145130.
- [32] Kaggle, "Face Recognition with Resnet50." [Online].

Available:

https://www.kaggle.com/code/sushanshrestha0690/face-recognition-with-resnet50

- [33] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," *arXiv*, 2020, doi: 10.48550/arXiv.1910.05446.
- [34] S. Nurdiati, M. K. Najib, F. Bukhari, R. Revina, and F. N. Salsabila, "Performance Comparison Of Gradient-Based Convolutional Neural Network Optimizers For Facial Expression Recognition," *BAREKENG: Journal of Mathematics and Its Applications*, vol. 16, no. 3, pp. 927– 938, 2022, doi: 10.30598/barekengvol16iss3pp927-938.
- [35] M. Kim, A. K. Jain, and X. Liu, "AdaFace : Quality Adaptive Margin for Face Recognition" arXiv, 2023, doi: 10.48550/arXiv.2204.00964.
- [36] M. E. A. Ali and D. Kumar, "The Impact of Optimization Algorithms on The Performance of Face Recognition Neural Networks," J. Adv. Eng. Comput., vol. 6, no. 4, p. 248, 2022, doi: 10.55579/jaec.202264.370.
- [37] M. E. Teo, L. Y. Chong, S. C. Chong, and P. Y. Goh, "2.5D Face Recognition System using EfficientNet with Various Optimizers," *Int. J. Informatics Vis.*, vol. 8, no. 4, pp. 2388– 2399, 2024, doi: 10.62527/joiv.8.4.3030.
- [38] S. Hamid, H. Madni, H. Muhammad, F. Shahzad, S. Shah, and M. Faheem, "Exploring optimizer efficiency for facial expression recognition with convolutional neural networks," *The Journal of Engineering*, vol. 2025, no. 1, pp. 1–29, 2025, doi: 10.1049/tje2.70060.
- [39] D. Bashir, G. D. Montañez, S. Sehra, P. S. Segura, and J. Lauw, "An Information-Theoretic Perspective on Overfitting and Underfitting," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12576 LNAI, pp. 347–358, 2020, doi: 10.1007/978-3-030-64984-5_27.
- [40] O. A. Montesinos López, A. Montesinos López, and J. Crossa, "Multivariate Statistical Machine Learning Methods for Genomic Prediction," *Springer*, 2022. doi: 10.1007/978-3-030-89010-0.