electronics

MDPI

Article

Semaphore Recognition Using Deep Learning

Yan Huan and Weiqi Yan *

Department of Computer Science, Auckland University of Technology, Auckland 1010, New Zealand; pyg6410@autuni.ac.nz

* Correspondence: weiqi.yan@aut.ac.nz

Abstract: This study explored the application of deep learning models for signal flag recognition, comparing **YOLO11** with basic CNN, ResNet18, and DenseNet121. Experimental results demonstrated that **YOLO11** outperformed the other models, achieving superior performance across all common evaluation metrics. The confusion matrix further confirmed that **YOLO11** exhibited the highest classification accuracy among the tested models. Moreover, by integrating MediaPipe's human posture data with image data to create multimodal inputs for training, it was observed that the posture data significantly enhanced the model's performance. Leveraging MediaPipe's posture data for annotation generation and model training enabled **YOLO11** to achieve an impressive 99% accuracy on the test set. This study highlights the effectiveness of **YOLO11** for flag signal recognition tasks. Furthermore, it demonstrates that when handling tasks involving human posture, MediaPipe not only enhances model performance through posture feature data but also facilitates data processing and contributes to validating prediction results in subsequent stages.

Keywords: YOLO11; semaphore recognition; convolutional neural network (CNN); deep learning; MediaPipe; feature extraction; data enhancement; pre-training model

Academic Editors: Antoni Morell and Ping-Feng Pai

Received: 16 November 2024
Revised: 16 December 2024
Accepted: 10 January 2025
Published: 12 January 2025

Citation: Huan, Y.; Yan, W. Semaphore Recognition Using Deep Learning. *Electronics* **2025**, *14*, 286. https://doi.org/10.3390/electronics14020286

1. Introduction

The semaphore is an ancient method of long-distance communication that employs flags or gestures to convey letters, numbers, or specific information. Typically, a semaphore flag measures 45 cm in both length and width, and is affixed to a stick [1]. In the absence of modern communication technologies, traditional visual signaling systems, such as signal flags, have been extensively employed in military command and maritime navigation to enable long-distance information transmission and communication between vessels [2]. Today, it continues to be employed for communication on ships [3]. Due to its independence from electronic components, this system is straightforward to operate and remains largely resilient to external electromagnetic interference under favorable visual conditions. These attributes render the semaphore a reliable emergency backup and an effective means of communication.

In practice, the semaphore sender needs a person who can see that the sender is in the correct position when delivering the semaphore code. Learning the semaphore code is crucial, as each letter has a unique position in the semaphore signal. This ensures that the sender accurately conveys the message, and the receiver correctly understands it. However, the accuracy of human eye recognition is affected by various factors, for example, the learning of the semaphore by the sender and receiver, weather and light conditions, distance and background [4].

Putra et al. employed an approach that combines wavelet transform with a back-propagation neural network. This method requires the manual extraction of frequency

domain features from images before performing classification [5]. These traditional methods rely heavily on human intervention in feature extraction, making the process of feature selection and extraction cumbersome and dependent on high-quality feature engineering.

Deep learning models significantly reduce the need for human intervention by automating the extraction of features from raw image data. This capability gives them a distinct advantage over traditional methods in image and video processing tasks. For instance, convolutional neural networks (CNNs) are able to derive hierarchical and intricate characteristics from image or video inputs, removing the requirement for manual feature specification. This approach greatly simplifies the data processing workflow. Li et al. demonstrated the effectiveness of CNNs in capturing spatial features in images through their application to human action detection [6]. The model automatically learns the details of human postures and movements, achieving efficient recognition without relying on predefined geometric features or manual labeling.

Furthermore, Rehman et al. proposed a deep learning architecture that combines 3D-CNN and LSTM for gesture recognition tasks, further enhancing automatic feature extraction capabilities [7]. The 3D-CNN component extracts spatial features from image sequences, while LSTM captures temporal dependencies in sequence data. This combination allows the model to understand both the spatial positioning and temporal evolution of gestures, leading to more accurate identification of continuous gesture movements. Unlike traditional methods, this architecture does not require the manual separation of spatial and temporal features. The model learns complex spatiotemporal relationships directly from data, significantly improving the accuracy and speed of dynamic gesture recognition.

Deep learning tasks heavily depend on high-quality labeled data and impose stringent requirements on image quality. Complex backgrounds further complicate model training. This study investigates the performance advantages of various CNN models in flag classification tasks, integrating MediaPipe pose data into the training process. Experimental results indicate that YOLO11 achieves superior performance metrics in flag classification tasks, demonstrating the highest classification accuracy as reflected in the confusion matrix. These findings provide preliminary evidence of the strong compatibility between the combination of YOLO11 and MediaPipe with flag classification tasks.

2. Materials and Methods

In this section, we describe the methodology utilized to obtain the relevant results. We also provide a literature review of the background research on related techniques as well as the experimental approach applied in this project.

In this project, CNN models were employed for comparison. We leveraged basic CNNs, ResNet18 and DenseNet121, as well as YOLO11. At the same time, MediaPipe was employed as a human pose detection tool, generating pose data and assisting in the creation of YOLO11 annotation data.

2.1. CNN

A convolutional neural network (CNN) is a deep learning model and a fundamental algorithm in the field of artificial intelligence. It draws inspiration from human visual systems and can be more precisely characterized as an emulation of human brain functions [2].

Convolutional neural networks (CNNs), unlike traditional neural networks, are composed of convolutional, pooling, and fully connected layers. Key strength lies in the ability to extract hierarchical data features through a layered approach. The incorporation of convolutional operations and weight-sharing mechanisms significantly reduces the number of parameters and computational complexity. This approach effectively mitigates the issue of local information loss. Moreover, CNNs are able to autonomously extract and propagate

data features through each layer, enabling the learning of complex patterns, making them particularly well suited for image processing tasks [8].

The basic CNN model employed in this project consists of three convolutional layers and one max pooling layer. This straightforward architecture is designed for image classification tasks. It accepts 128 × 128 RGB image inputs and maps the extracted features to five distinct categories.

Max pooling layers follow each convolutional layer. The ReLU activation function reduces the dimensionality of the feature map [9]. Finally, the data are flattened into a one-dimensional vector, which is then input for classification.

The input layer accepts a 128 × 128 RGB image, which is processed by convolution and pooling in sequence. At the final stage, the input passes through the fully connected layer and then the activation function, resulting in the classification output.

2.2. ResNet18

ResNet-18 is a classic deep learning model with 18 layers, mainly utilized for image classification tasks. For example, some researchers utilized a pre-trained ResNet-18 model to classify yoga poses. The researchers employed a dataset from Kaggle, which contains images of various yoga poses. By fine-tuning the pre-trained model, the model achieved 98% accuracy on the training set [10].

The core architecture of ResNet-18 leverages skip connections through residual blocks, effectively mitigating the gradient vanishing issue in deep networks and enhancing the model's training stability. ResNet-18 comprises an initial convolutional layer, followed by four residual stages—each containing two convolutional layers—a global average pooling layer, and a fully connected layer. As a relatively shallow ResNet variant, it is well suited for scenarios with limited computational resources, while still delivering high performance in image classification tasks [11].

An efficient method for identifying individual goats using machine vision was developed. The researchers leveraged deep learning technology and four convolutional neural networks (CNNs), including ResNet18, to extract features and classify goat facial images. By constructing and training a specialized CNN model, the method achieved accurate identification of individual goats even in complex backgrounds. The effectiveness of ResNet18 in image learning and classification tasks was demonstrated, making it a suitable choice for incorporation into the experimental model [12].

2.3. DenseNet121

DenseNet121 is a classic model in the DenseNet series, proposed in 2017. It passes the output features of each layer to all subsequent layers through a dense connection mechanism, thereby significantly improving the feature reuse rate and gradient propagation efficiency [13]. The idea of dense connection comes from the residual connection of ResNet, but it expands the jump connection from adjacent layers to each layer, connecting all subsequent layers. DenseNet no longer simply adds the input and output, but concatenates the output features of each previous layer with the features of the current layer to form a denser information flow. DenseNet121 consists of 121 layers, including 4 dense blocks and transition layers, which are utilized to reduce the dimension and computational complexity of the feature map. DenseNet applies fewer parameters than other deep networks (such as ResNet) with comparable performance, and has stronger generalization ability and efficiency. It performs well in tasks such as image classification and object detection. In 2024, researchers developed an efficient human–computer interaction system based on gesture recognition for people with disabilities. a gesture classification method was proposed based on the DenseNet121 model for recognizing sign language and other gestures. It

provides a new human–computer interaction method for people with disabilities and promotes the development of sign language recognition and assistive technology [14]. It also demonstrated strong performance in the classification tasks involving sour jujube seeds. Its dense connection mechanism improves feature reuse and gradient propagation efficiency, effectively addressing the gradient vanishing problem. This enables the model to achieve high accuracy and robustness, with particularly stable performance in scenarios with complex backgrounds and varying lighting conditions. Compared to ResNet-50 and Inception-v3, DenseNet-121 offers notable advantages in classification accuracy and high-resolution detail processing. Additionally, its reduced parameter count and higher computational efficiency make it a reliable and efficient solution for automated sour jujube quality control [15].

2.4. YOLO11

You Only Look Once (YOLO) is a unified, real-time object detection method proposed in 2016 [16]. YOLO integrates the object detection task into a single neural network model, predicting the object category and bounding box location directly from the input image, and realizing an end-to-end detection process. It greatly improves the detection speed by dividing the image into grids, each of which predicts multiple bounding boxes and category confidences. YOLO laid the foundation for real-time object detection and promoted the development of subsequent versions. The latest version, **YOLO11**, introduces significant improvements over its predecessors. It features a smaller model size and fewer parameters, while achieving enhanced feature extraction, optimized efficiency, and faster processing speeds. **YOLO11** supports a wide range of tasks and delivers higher accuracy despite its reduced parameter count. Additionally, it demonstrates robust cross-environment adaptability, making it suitable for deployment across diverse platforms, including edge devices and cloud environments.

Edge devices often face constraints such as limited computational resources, memory capacity, and power consumption, which pose challenges for deep learning tasks. To address these limitations, many models are continually optimized to offer lightweight versions, such as Tiny YOLO, ResNet-Lite, and FaceNet. These advancements highlight the ongoing efforts to balance performance and efficiency in resource-constrained environments.

Currently, there is a synthetic dataset generated by a large language model (LLM) to train an apple detection model based on YOLO11 and YOLOv10. Our experiments show that synthetic data are effective in improving model performance and generalization ability. The detection accuracy of the model on real orchard images is as high as 0.84, and mAP@50 is 0.89. This expands the application of YOLO in agricultural target detection tasks [17].

2.5. MediaPipe

MediaPipe is an open source framework developed by Google AI. It adopts a modular design and defines data flows through computational graphs. It supports multiple data types such as images, videos, and audio, and is particularly suitable for real-time applications [18].

In MediaPipe human pose recognition module (Pose) is an efficient and accurate real-time key point detection tool that can identify 33 key points and support 2D and 3D pose estimation. Its end-to-end pipeline integrates pose detection and tracking, optimizes real-time performance under low latency, and is suitable for multiple platforms such as Android, iOS, and Web. This module is widely leveraged and provides real-time pose feedback and interactive functions.

A system was proposed based on the MediaPipe framework and convolutional neural network (CNN) for recognizing static gestures in the Mexican Sign Language (MSL) finger

spelling alphabet. By using MediaPipe to extract hand key point information and input it into the CNN model for training and classification, the system achieved an accuracy of 83.63% on 336 test images. In addition, for each letter sample, the system had an accuracy of 84.57%, a sensitivity of 83.33%, and a specificity of 99.17%. The system can run in real time on low-power devices, improving its usability and portability. This shows that MediaPipe has good reliability in human posture recognition [19].

2.6. Methods

2.6.1. Data Collection and Process

The training set of this experiment needed to be collected and processed by the device. The process of data collection and processing is shown in Figure 1.

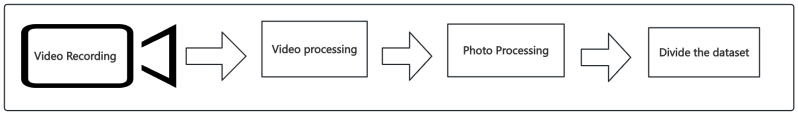

Figure 1. Dataset collecting and processing.

We utilized a mobile phone to record videos of flag signals representing the 26 letters. The recordings were conducted in two distinct environments: sunny outdoor parks and indoor environments with white walls illuminated by artificial lighting. Two individuals served as subjects. Each video was approximately one minute long, recorded at a frame rate of 60 frames per second.

Although we shot video footage of 26 letters, this experiment mainly trained on the first five letters of the alphabet. The purpose of this experiment was to conduct preliminary exploration for subsequent research, with the focus on exploring and selecting a suitable research plan. To enhance the model's generalization ability, data augmentation techniques were applied. Images were extracted from the videos at intervals of every six frames to prevent excessive repetition of the same gesture. Subsequently, two random augmentation operations were performed on the extracted images, including random rotation ($\pm 15^{\circ}$), scaling ($\pm 10\%$), brightness adjustment ($\pm 20\%$), sharpness adjustment, and contrast enhancement. These augmentations increased the diversity of the dataset and reduced the risk of misclassification for previously unseen gestures.The video material was captured from multiple angles around the subject, introducing variability to the dataset. Combined with image enhancement operations, this approach effectively enriches the training dataset, making it sufficient to meet the requirements of this experiment. Additionally, the original image resolution 1920×1080 was downscaled to 640×640, resulting in a total of 11,813 images, of which 3075 were extracted without augmentation.

The images were partitioned into training, validation, and test sets in an 8:1:1 ratio, yielding 8241 images for training, 1775 for validation, and 1797 for testing. The unaugmented dataset was similarly partitioned following this ratio.

In this experiment, the pose module of MediaPipe was employed to extract arm posture features relevant to flag gestures. The pose module can identify 33 anatomical landmarks on the human body and provides their three-dimensional coordinates. As flag recognition primarily relies on arm movements, we selected six key points closely related to arm posture: left and right shoulders, elbows, and wrists. Each key point included (x, y) coordinates to describe the geometric position of the arm. These coordinates were stored alongside the image file names and their corresponding classification labels for subsequent model training. The dataset containing these features is referred to as the Feature Dataset (FData), while the dataset without them is called the Common Dataset (CData). Initially, we considered using labelImg to generate annotation data for training the CNN model. The coordinates of the character range were roughly calculated based on the MediaPipe

pose points in the picture, and lines were drawn in the image. Then, a **YOLO11** annotation file was generated based on the picture name. After that, the visualization tool labelImg fine-tuned the bounding box position to ensure that it accurately matched the actual gesture position. Finally, the refined dataset trained the **YOLO11** model.

We fed the CNN model input during the training phase by combining the gesture features with the original image data. The images were normalized and the gesture feature coordinates were standardized to [0, 1]. With this multimodal input method, the model could learn both image pixel information and arm gesture information, thereby improving the accuracy of gesture classification.

2.6.2. Operating Method

In this experiment, we utilized the latest version of **YOLO11** to train the dataset containing the labeled files. The PyTorch framework was utilized to create distinct classes representing three models, as well as three models that support the simultaneous input of MediaPipe data and image data. These models are referred to as Simple CNN (S-CNN), ResNet CNN (R-CNN), DenseNet CNN (D-CNN), Pose Simple CNN (PS-CNN), Pose ResNet CNN (PR-CNN), and Pose DenseNet CNN (PD-CNN).

CData (image dataset) and FData (MediaPipe feature + image dataset) were employed after applying standardized data normalization and augmentation, with the learning rate, optimizer, loss function, training epochs, and other hyperparameters kept consistent. The following models were trained: S-CNN, R-CNN, D-CNN, PS-CNN, PR-CNN, and PD-CNN.

The 6 groups of CNN models used the same parameter settings: the loss function was Cross-Entropy Loss, the optimizer was Adam, the initial learning rate was 0.001, and the learning rate was dynamically adjusted with a step size of 10 and a decay factor of 0.1. The training rounds were set to 100, and the input images were resized to 128×128.

The YOLO11 model uses an image input size of 640×640, with training rounds set to 100.

2.6.3. Ablation Experiments

Ablation experiments are a crucial method for evaluating the contributions of various modules, components, and techniques within a model or system. By systematically altering specific modules or hyperparameters while keeping other variables constant, their actual impact and necessity can be assessed through performance metrics analysis. This approach helps to identify components that significantly enhance performance while uncovering redundant modules or parameters, enabling the simplification of the model structure. Additionally, ablation experiments validate the rationality of design choices, improve model interpretability, and provide researchers with a clearer understanding of the model's underlying working principles.

This study performed an ablation experiment on the YOLO11 model, analyzing its results across four key factors: the number of epochs, batch size, data augmentation, and the application of Mosaic enhancement. The experiment evaluated the model under varying configurations, where the number of epochs was set to 30, 50, 80, and 100, while the batch size was adjusted to 8, 16, and 32. Additionally, the impact of enabling or disabling Mosaic enhancement, as well as the inclusion of general data augmentation techniques, was systematically examined to assess their influence on model performance.

2.6.4. Evaluation Indicators

The indicators for model evaluation include accuracy, precision, recall, the F1 score, the ROC curve, and confusion matrix.

Accuracy: Accuracy is the ratio of correct samples to the total number of samples:

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

where TP represents the number of correctly predicted positive classes. TN refers to negative classes that are predicted accurately. FP corresponds to the negative classes incorrectly predicted as positive, while FN denotes positive classes that are mistakenly classified as negative [20].

Precision: Precision is the ratio of correctly predicted positive samples to all predicted positive samples:

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

where TP is the number of correctly predicted positive classes. FP is the number of incorrectly predicted positive classes.

Recall: Out of all actual positive samples, recall is the percentage of accurately anticipated positive samples:

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

where TP is the number of samples correctly predicted as positive. FN is the number of samples incorrectly predicted as negative, which are actually positive.

F1-Score: In terms of precision and recall, the F1-score is the harmonic mean:

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

3. Results

3.1. Performance Comparison

As shown in Table 1, S_CNN, R_CNN, and D_CNN are models that were not trained with MediaPipe pose data. PS_CNN, PR_CNN, and PD_CNN are models that were trained with MediaPipe pose data combined with image data. A comparison reveals that incorporating human pose data as auxiliary training significantly enhances model performance. In contrast, models trained without pose data rely solely on extracting all features from the image, learning patterns autonomously. This approach increases the complexity of learning and prediction due to interference from factors such as background noise, lighting variations, and pose discrepancies. By integrating pose data, the model places greater emphasis on human pose features within the training data, effectively mitigating the impact of image noise. Consequently, the overall performance of the model improves. These findings highlight that multi-dimensional training data facilitate the model's ability to quickly identify critical features, thereby reducing the computational complexity and learning cost. The data annotation method employed by YOLO11 differs from that of the other three CNN models. YOLO11 utilizes rectangular box coordinates to delineate the effective regions of an image, assign the corresponding category, and define both the location and size of the target. This annotated information serves as supervisory input, guiding the model in accurately identifying and localizing the target. By reducing the influence of extraneous information in the image, this approach enables the model to efficiently learn the target's distinctive features. As a result, YOLO11 demonstrates superior performance metrics compared to the other models.

Table 1. Performance comparison of three models.

Model	Accuracy	Precision	Recall	F1-Score
S_CNN	65%	0.68	0.65	0.65
R_CNN	82%	0.85	0.82	0.82
D_CNN	90%	0.91	0.90	0.89
PS_CNN	86%	0.86	0.86	0.86
PR_CNN	97%	0.97	0.97	0.97
PD_CNN	98%	0.98	0.98	0.98
YOLO11	99%	0.99	0.99	0.99

Figure 2 shows the changes in the mAP50-95 of the **YOLO11** model for Class B gestures during training. In the early stages, mAP50-95 quickly rose from 0.3 to 0.7, indicating a significant improvement in the model's detection ability for Class B gestures. Between steps 20 and 60, the accuracy fluctuated, reflecting the model's further fitting to the data during this period. After 60 steps, mAP50-95 gradually stabilized, remaining around 0.8, suggesting that the model accuracy for Class B gestures reached a high and stable level.

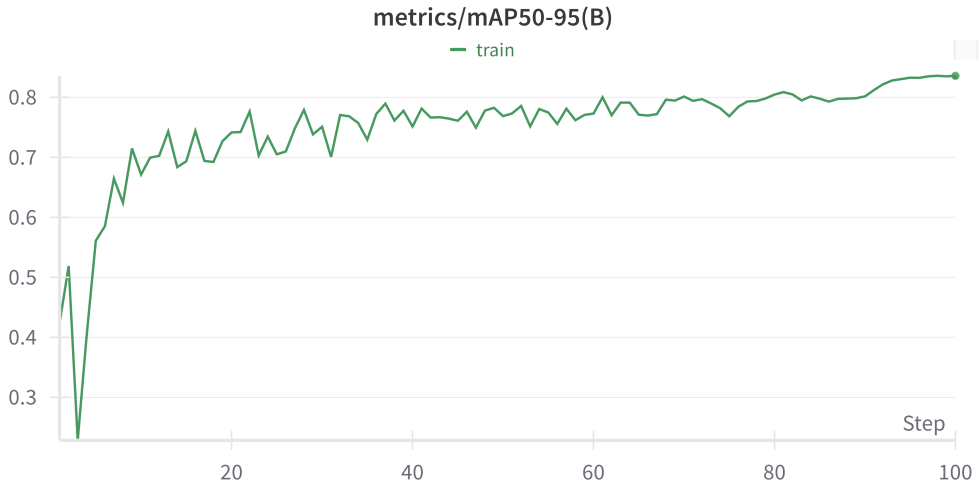

Figure 2. YOLO11 mAP50-95.

3.2. Roc Curve

From Figures 3–5, PR_CNN and PD_CNN demonstrated superior performance, with AUC values approaching 1.00 across all categories, indicating exceptionally high classification accuracy. In contrast, the basic CNN models, such as R_CNN and S_CNN, struggled to differentiate certain categories, particularly Class B, resulting in lower AUC values. This outcome suggests challenges in accurately identifying instances of this class. Overall, models trained with MediaPipe feature data outperformed those trained without such data, particularly in the case of basic CNN models. Although the pre-trained models exhibited improvement after integrating feature data, the extent of this enhancement was relatively limited. The challenges associated with Class B may be due to image distortion caused by the shooting angle and inconsistent task postures. Further adjustments and optimizations are necessary to address these issues.

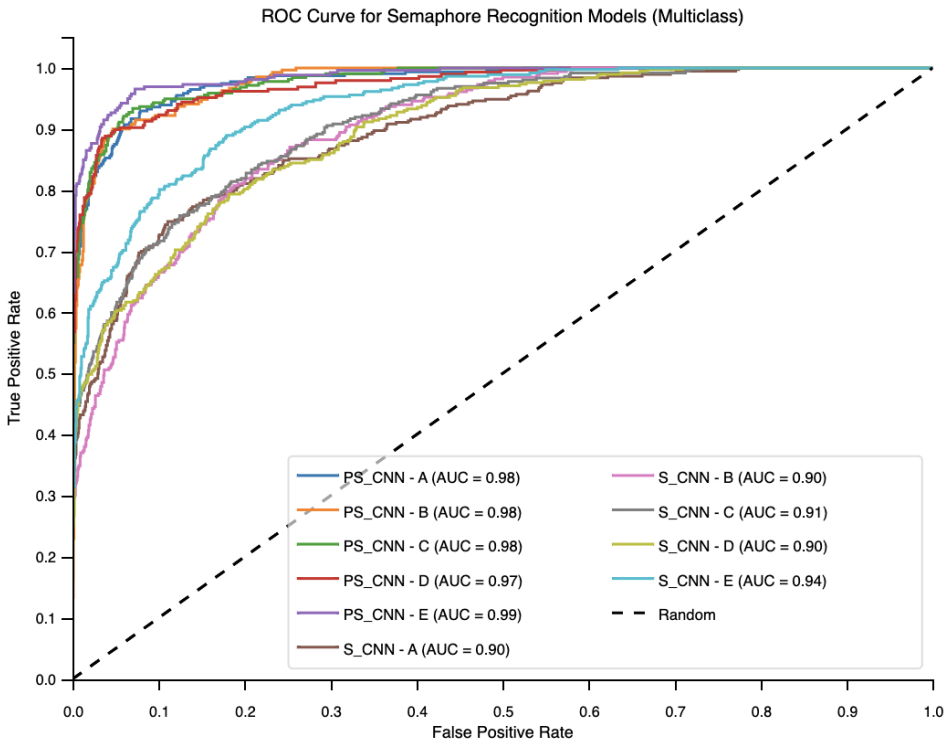

Figure 3. S_CNN and PS_CNN ROC curve.

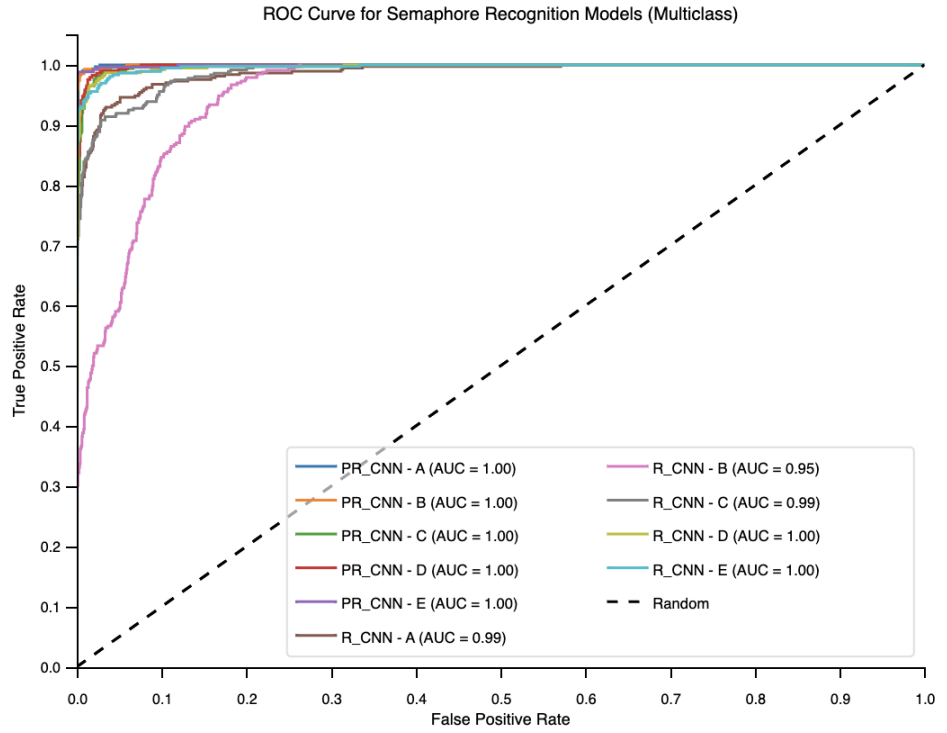

Figure 4. R_CNN and PR_CNN ROC curve.

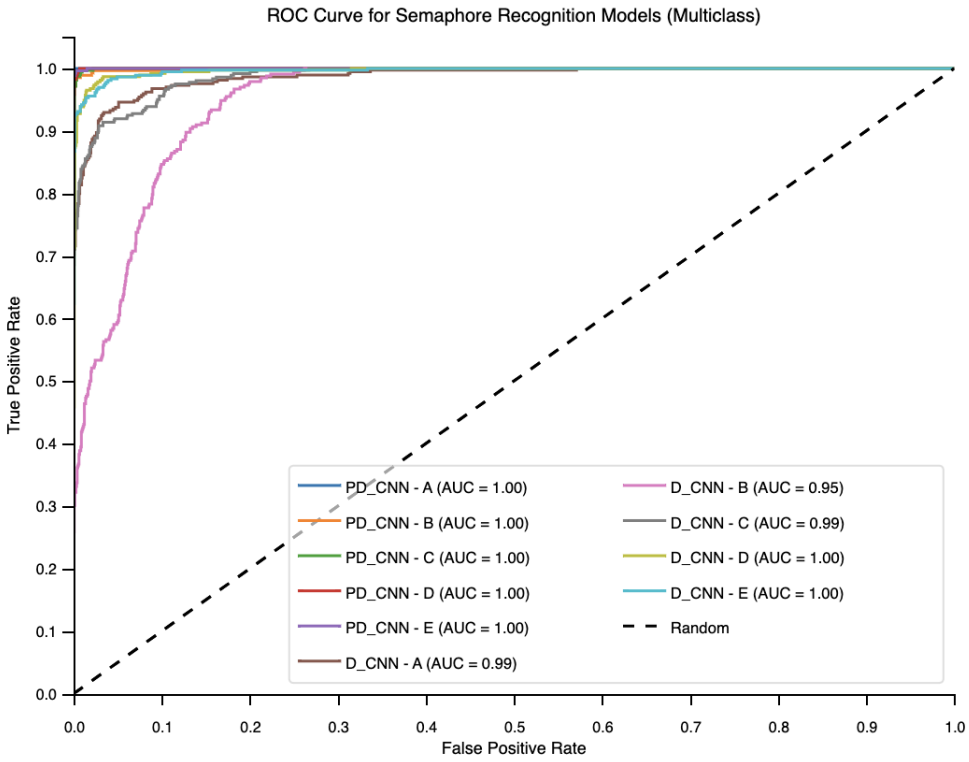

Figure 5. D_CNN and PD_CNN ROC curve.

3.3. Confusion Matrix

From the confusion matrix of Figures 6 and 7, it can be seen that the S_CNN and PS_CNN models demonstrate a high confusion rate in classifying Class B and Class C gestures. Specifically, the S_CNN model frequently misclassifies gestures from these classes. Class B gestures are often incorrectly classified as Class A or Class C, while Class C gestures are mistaken for Class B. This misclassification arises mainly due to the visual similarity between these two gesture classes, influenced by the angle of the photos. The minimal variations in arm angles between these gestures make it difficult for the model to precisely differentiate the pixel data in the images.

By introducing the pose features extracted by MediaPipe, the PS_CNN model shows a significant improvement in the classification of Class B and Class C gestures. MediaPipe extracts geometric location information of the arms, such as the relative coordinates of the shoulders, elbows, and wrists. These coordinates capture key posture variations that are difficult to detect through image pixel data alone. As a result, the PS_CNN model can utilize these pose characteristics to distinguish between Class B and Class C gestures, thereby reducing confusion between the two.

To further reduce classification errors, the optimization methods can be applied: First, data augmentation techniques are applied to generate more Class B and C gesture images under varying angles and lighting conditions, enhancing the model's generalization ability for these gesture types. Second, increasing the diversity of gesture features especially by incorporating more detailed finger position information can help the model recognize finer details and improve overall accuracy.

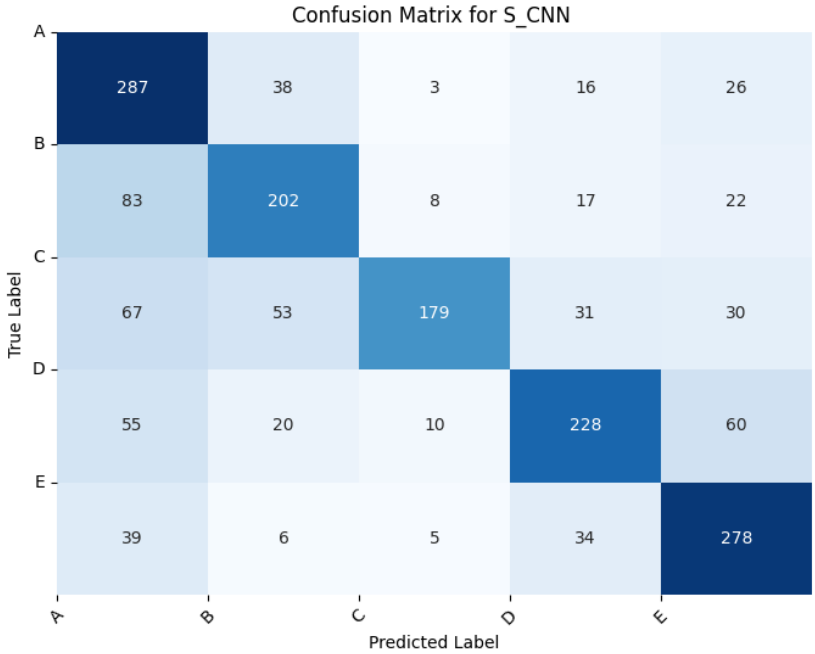

Figure 6. Confusion matrix for S_CNN.

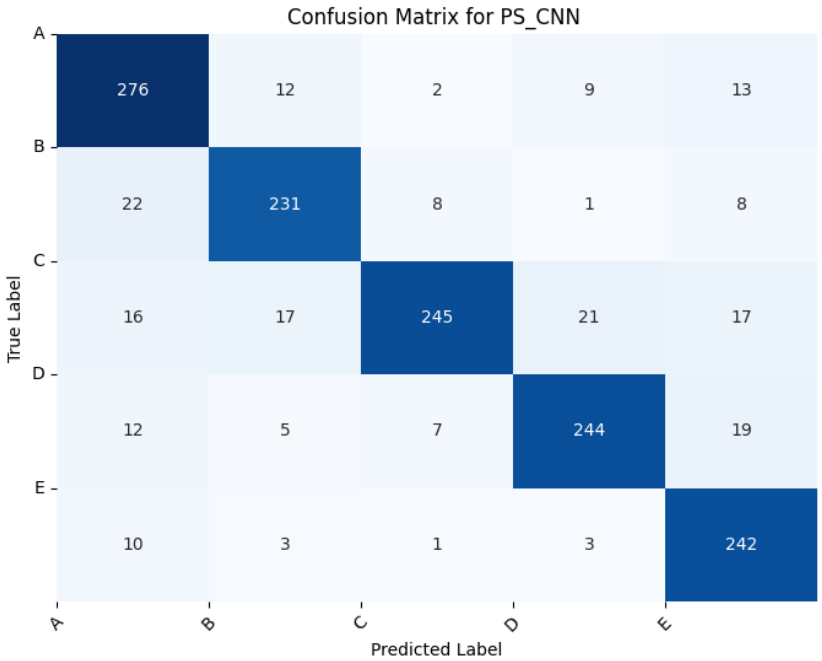

Figure 7. Confusion matrix for PS_CNN.

From Figures 8 and 9, the confusion matrix analyses of R_CNN and PR_CNN show that the PR_CNN model performs significantly better than the R_CNN model in classification tasks, especially for gestures of type B and type C. In the R_CNN model, gestures of type B are often misclassified as A or C, and similar confusion exists for gestures of type C. This result suggests that the R_CNN model relies primarily on image pixel features, making it difficult to discern subtle differences between gestures of type B and type C that are similar due to visual angles.

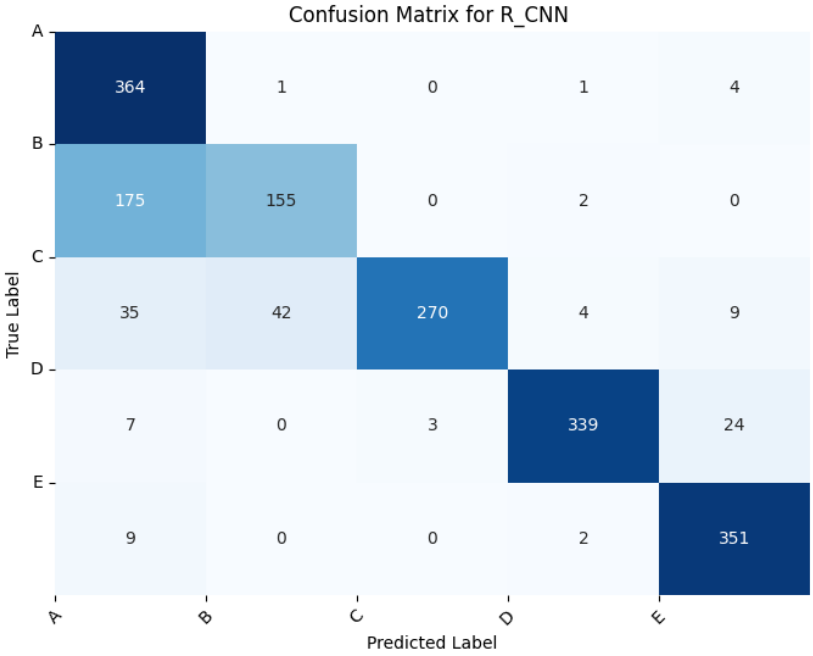

Figure 8. Confusion matrix for R_CNN.

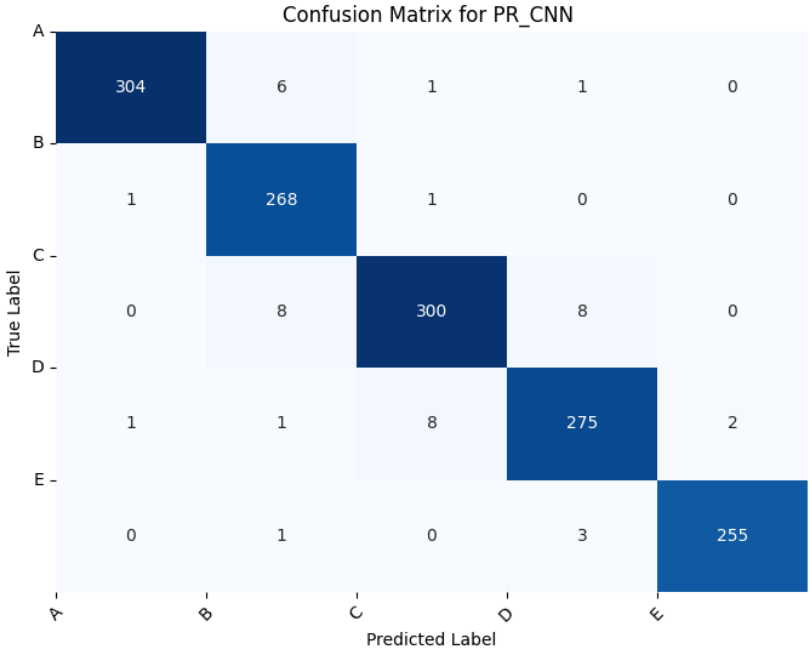

Figure 9. Confusion matrix for PR_CNN.

The PR_CNN model effectively improves the ability to distinguish these gestures by introducing MediaPipe posture features. The coordinate information of shoulder, elbow, and wrist extracted by MediaPipe provides the model with additional geometric information, allowing the PR_CNN model to better understand the posture and position changes of the arm. Compared to relying solely on image features, posture features improve the ability of the model to distinguish between B and C gestures, reducing misclassification between them. For example, in the PR_CNN model, the number of misclassifications of B and C gestures is greatly reduced, which shows that posture information can effectively make up for the shortcomings of image features and improve the classification accuracy of the model.

In addition, the classification performance of PR_CNN in other categories has also been significantly improved, especially when dealing with complex gestures. By combining visual features and posture features, the model's overall perception of gestures is significantly enhanced. Combining multimodal features enhances both classification accuracy and the robustness of the model in complex backgrounds.

From Figures 10 and 11, by comparing the confusion matrices of D_CNN and PD_CNN, it is obvious that the PD_CNN model performs significantly better than the D_CNN model in the gesture classification task, especially in the classification of Class B and Class C gestures. In the D_CNN model, Class B gestures have a higher misclassification rate and are often misclassified as Class A or Class C, while Class C gestures also have a higher misclassification rate, especially when being mistaken for Class B. This confusion phenomenon shows that the D_CNN model only relies on image pixel features and is difficult to effectively distinguish when processing these gestures with similar visual features.

The PD_CNN model with MediaPipe posture features significantly improves the classification accuracy of the model by providing additional geometric position information. The coordinates of shoulder, elbow, and wrist extracted by MediaPipe provide the model with accurate arm posture information, allowing the model to better capture the angle and position changes of the arm between different gestures. Compared with D_CNN, PD_CNN significantly reduces the confusion between Class B and Class C by using these posture features. For example, the PD_CNN model reduced the number of misclassifications of gestures in category B from 20 to 0, and the number of misclassifications of gestures in category C from 56 to 2, showing the key role of posture features in distinguishing complex gestures.

In addition, PD_CNN also performed more stably on other gesture categories, especially in the classification tasks of categories D and E, with almost no obvious misclassification. Compared with D_CNN, the overall classification performance of PD_CNN is more robust, especially in complex gesture scenes, the combination of posture features makes the model more advantageous in capturing changes in arm movement. This shows that multimodal feature fusion effectively improves the accuracy and robustness of the model, especially when dealing with gestures that are visually difficult.

From Table 2, **YOLO11** performs well on the test set, with category B being misclassified as category C only once. In contrast, PD_CNN shows slightly more confusion, especially with category A often being confused with category B and category C being misclassified as other adjacent categories. The higher number of misclassifications compared to **YOLO11** suggests that PD_CNN faces more challenges when dealing with categories that have overlapping or similar visual features. **YOLO11** demonstrates better generalization ability and less confusion between categories, making it more reliable for the flag classification scenarios.

Table 2. Confusion matrix for YOLO11 and PD_CNN.

	YOLO11					**PD_CNN**				
	A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
A	354	0	0	0	0	295	14	1	2	0
B	0	302	0	0	0	0	267	3	0	0
C	0	1	342	0	0	1	2	312	1	0
D	0	0	0	340	0	0	0	2	285	0
E	0	0	0	0	309	0	0	1	3	255

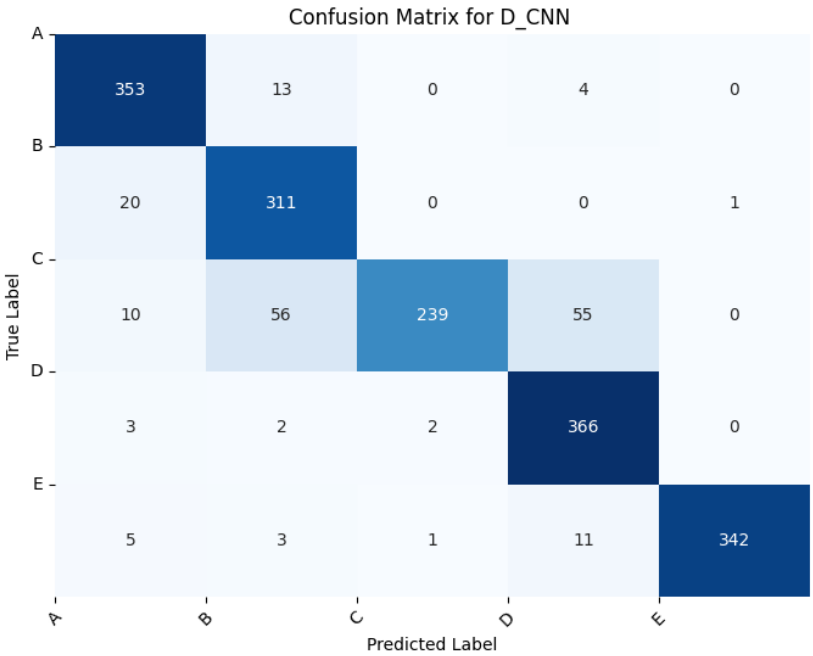

Figure 10. Confusion matrix for D_CNN.

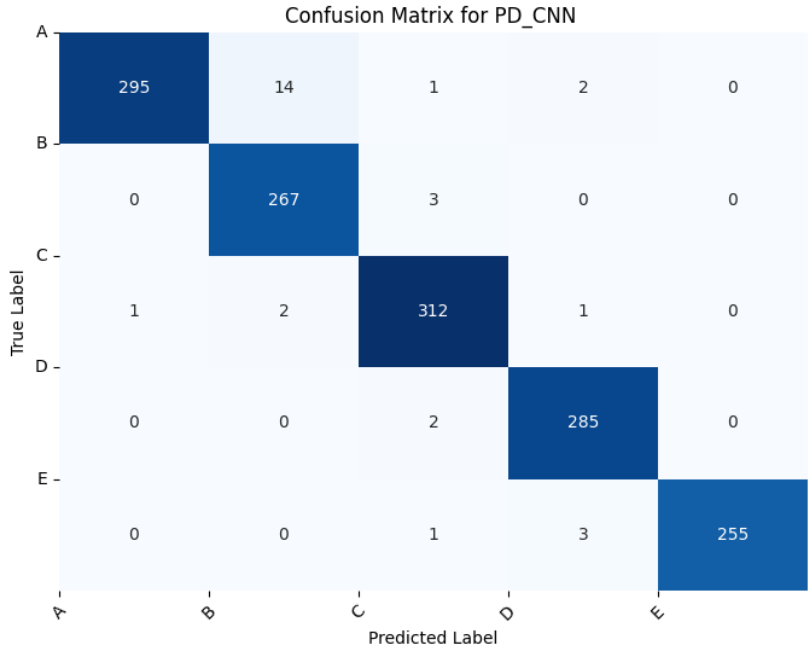

Figure 11. Confusion matrix for PD_CNN.

The ablation experiment on the YOLO11 model reveals key insights into the influence of epochs, batch size, data enhancement, and mosaic augmentation on model performance.

From Table 3, when the epochs are set to 100, the model achieves an accuracy and precision of 0.9994, with mAP@50 reaching 0.9914. The performance difference between 50 and 100 epochs is marginal. However, when the epochs are reduced to 30, a significant drop in performance is observed. Thus, 50 epochs are sufficient for training this model, as increasing to 80 or 100 yields only minor improvements, whereas fewer epochs (e.g., 30) result in underfitting.

Table 3. YOLO11 ablation experiments results.

Ep	Batch	Mos	Augm	Acc	Prec	Rec	F1	mAP50	mAP50-95
100	16	1	1	0.9994	0.9994	0.9994	0.9994	0.9914	0.8348
80	16	1	1	0.9912	0.9912	0.9912	0.9912	0.9912	0.8353
50	16	1	1	0.9988	0.9988	0.9988	0.9988	0.9901	0.8305
30	16	1	1	0.8816	0.8841	0.8816	0.8816	0.8411	0.5911
50	16	1	0	0.8044	0.8393	0.8044	0.8101	0.7603	0.5453
50	16	0	1	0.9951	0.9952	0.9951	0.9951	0.9904	0.8318
50	32	1	1	0.9939	0.9942	0.9939	0.9939	0.9888	0.8329
50	8	1	1	0.9971	0.9971	0.9972	0.9971	0.9893	0.8261

In the experiments on batch size, a batch size of 16 achieves the best performance, although batch sizes of 8 and 32 result in only slight declines. This indicates that batch size has a relatively small impact on the performance of the model in this experiment.

The mosaic augmentation method, which combines four images into one to include multiple targets, is typically beneficial for detecting small targets. In this task, enabling mosaic results in only minor improvements. This limited effect is likely because the flag targets are relatively large, reducing the benefit of mosaic augmentation.

For general data enhancement, operations such as rotation and brightness adjustment are applied to create augmented datasets. The results show that training on the enhanced dataset achieves significantly better performance, with accuracy dropping from 0.9988 to 0.8044 and mAP@50 declining notably when enhancements are disabled. This highlights that data augmentation is crucial for improving model generalization and ensuring data diversity.

From these four groups of experiments, it can be concluded that for the flag classification task, 50 epochs are sufficient, and a batch size of 16 yields the best performance. The mosaic data augmentation function has minimal impact on the model, likely due to the relatively large target size. However, the absence of general data augmentation results in a significant performance decline, underscoring its importance in ensuring data diversity and enhancing the model's generalization ability.

Additionally, while the best mAP@50-95 remains above 83%, it has not improved significantly. Unlike mAP@50, which reflects target detection accuracy, mAP@50-95 imposes stricter requirements on both detection and box positioning. The relatively limited improvement suggests that errors in annotation range or background noise may hinder precise localization. Addressing these issues will be the focus of future optimization efforts.

In conclusion, this ablation experiment effectively identified the role of data enhancement and the optimal number of epochs. Future work will involve fine-tuning hyperparameters, exploring different learning rates, loss functions, and optimizers to further improve the performance of the model.

4. Discussion

This project aimed to compare the performance of multiple CNN models in flag signal recognition tasks to prepare for the subsequent real-time flag signal recognition system.

4.1. Is YOLO11 Better than Other CNN Models in Terms of Overall Indicators?

In the flag classification task, CNN models exhibit varying performance levels. The basic S_CNN model performs significantly worse than the other two models, largely due to its simpler architecture, fewer parameters, and a limited amount of training data. Among the pre-trained models, while ResNet18 demonstrates strong performance, it remains inferior to DenseNet121 in terms of stability and accuracy. The densely connected architecture and

deeper layers of DenseNet121 allow each layer to more effectively receive outputs from previous layers. This structure greatly enhances information flow and improves feature reuse. Additionally, the augmentation through image enhancement techniques further improves the capacity of the model. Most importantly, the inclusion of posture features extracted by MediaPipe provides additional inputs to DenseNet121. This enables the model to achieve high accuracy and strong generalization, even without large labeled datasets. As a result, DenseNet121 demonstrates superior performance and adaptability in the flag classification task.

During **YOLO11** training, MediaPipe serves as an auxiliary tool to extract postures and generate annotation files. This approach maximizes the data extracted by MediaPipe, significantly enhancing the model classification ability, resulting in excellent performance on the test set. In real-time detection, **YOLO11** can also predict flag gestures, although the current accuracy is not higher. Nevertheless, it is evident that **YOLO11** demonstrates strong real-time detection and classification capabilities.

4.2. Does the Posture Extraction Feature of MediaPipe Help the Task of Flag Signal Recognition?

The comparison of six models across three groups revealed that CNN models applying MediaPipe features achieved superior performance. This is demonstrated by improvements in key evaluation metrics for the basic model. Although the pre-trained models show only modest improvements in these metrics, the confusion matrices indicate that models using MediaPipe maintain strong classification accuracy and stability. This finding indicates that the flag classification task can succeed without manually labeled data. The integration of posture features extracted via MediaPipe substantially enhances model performance.

The integration of posture features extracted by MediaPipe greatly improved model performance. Therefore, this method provides an effective solution for flag recognition training. In addition, the reliable posture recognition ability of MediaPipe can also assist with image annotation data, significantly reducing work time. In subsequent flag recognition research, it can help to annotate data and can also assist in judging flag images through posture data, improving the accuracy of system predictions.

4.3. Error Analysis

The analysis of the ROC curve and confusion matrix revealed a higher propensity for misclassification in categories B and C. This issue can be attributed primarily to image distortion caused by the shooting angle. Specifically, in category B, the correct posture requires the right hand to be parallel to the ground and the left hand perpendicular to the ground, whereas in category C, the correct posture involves the right hand forming a 45-degree angle to the upper right and the left hand remaining perpendicular to the ground. When the camera is positioned at a left-front or right-front angle relative to the subject, and elevated above the subject, it becomes challenging to visually distinguish the right arm's angle. This setup also introduces postural deformation, leading to deviations from the standard movements. Additionally, handheld operation of the camera introduces motion-induced blurring, further affecting image quality.

Postural deformation similarly impacts the classification performance for categories C and D, particularly in models other than YOLO11. Furthermore, the horizontal flipping of images exacerbates confusion between categories C and E. These findings highlight the critical importance of ensuring the quality of the dataset and optimizing the structural processing of images. Future research should focus on addressing these challenges through improved data acquisition protocols, enhanced preprocessing strategies, and ensuring the stability of the dataset. Additionally, disabling the model's automatic image enhancement parameters is recommended to maintain consistency and reliability in the data.

4.4. Limitations and Future Work

The current experimental results indicate that YOLO11 is well suited for flag signal recognition tasks. Its bounding box annotation method enables the model to efficiently learn target features. Based on the current findings, YOLO11 and MediaPipe have been identified as the core technologies for subsequent research. However, several challenges must be addressed to enhance the model's practical applicability.

Firstly, the current model has not been trained on the full set of 26 letters, nor does it cover specific international flag signal actions, such as those representing numbers, "start", "end", or "help". Future research should expand the training dataset to include these signals while ensuring the model maintains real-time detection capabilities.

Secondly, future work will focus on integrating the trained model with a large language model. This integration aims to enable flag signal videos to be converted into text descriptions, which can then generate corresponding flag signal images. Ultimately, these modules will be combined to develop a web-based system that facilitates quick understanding and learning of flag signals for users. This holistic approach will significantly enhance the accessibility and usability of flag signal recognition technology.

5. Conclusions

This study aimed to evaluate the performance of various CNN models in flag classification in order to find the best solution for subsequent research. The compared models include YOLO11, ResNet18, DenseNet121, and basic CNN.

Among them, the YOLO11 model shows excellent performance, and its accuracy and precision are better than the other models. It can also be seen in the confusion matrix that the YOLO11 model has a higher classification accuracy in the prediction results of the test set. It proves that the YOLO11 model is more suitable for flag classification tasks than other CNN models in the experiment.

In addition, the experiment innovatively introduces MediaPipe posture data for joint training. The analysis of ROC curves and confusion matrices shows that the model with MediaPipe posture features is always better than the model without such features. MediaPipe plays a key role in extracting human posture features. By detecting key points such as shoulders, elbows, and wrists, it provides additional geometric information, which greatly improves the model's classification accuracy for flags. Models integrating MediaPipe features show higher stability and accuracy, especially when classifying complex gestures. Therefore, in the flag signal training task, the introduction of key posture data and image data for joint training has an effect on improving the performance of the model.

In the experimental results, it can be seen that the model assisted by posture data training has improved in both indicators and classification. At the same time, in the labeling work of YOLO11 training data, with the help of MediaPipe's posture data coordinates, the flag signal part range in the image can be preliminarily delineated, and then manually fine-tuned, which greatly saves the time of YOLO11 training data labeling.

Therefore, through research, it can be found that YOLO11 and MediaPipe can be used together for flag signal recognition tasks. Not only can MediaPipe be used to improve efficiency in the data processing stage, but also the dimension of data can be increased in the training stage to help the model notice the human posture in the flag signal task. Later, posture can also be used in the prediction stage to help verify the prediction results of the model and improve the accuracy and stability of the output results.

Through ablation experiments, we confirmed the significant contribution of data augmentation operations to enhancing data diversity and improving the model's effectiveness. Additionally, the experiments helped identify optimal training parameters, thereby increasing training efficiency. The results also highlighted the importance of focusing on

dataset diversity and ensuring the accuracy of data annotations. In future research, we will continue to utilize ablation experiments to refine and optimize the model further.

This experiment has several limitations. First, it imposes strict requirements on the posture and camera positioning of the subjects. Abnormal angles or excessive shooting distances may lead to MediaPipe's inability to accurately detect the coordinates of human body postures. Such data require manual screening, annotation, or, in some cases, exclusion. This loss of data reduces the diversity of the training samples, potentially impacting the model's generalization capability. Additionally, the experimental settings lack sufficient variation in the scenes of flag signal actions and subject appearances. While the model demonstrates strong performance on the test set, its application in real-world scenarios yields less satisfactory results due to the limited diversity of the training environment.

The future work goal will be to expand to a complete set of 26 letters, use YOLO11 to achieve real-time flag signal recognition, and establish an interactive conversion system for flag signal actions and natural language. It can be used in scenarios where flags are used, such as sailing, sports, or flag language teaching, to help users understand and learn flag language more easily.

Author Contributions: Conceptualization, Y.H. and W.Y.; methodology, Y.H.; software, Y.H.; validation, Y.H.; formal analysis, Y.H.; investigation, Y.H.; resources, Y.H. and W.Y.; data collection, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H. and W.Y.; visualization, Y.H.; supervision, W.Y.; project administration W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research has no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding authors due to personal privacy concerns.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rachmad, A.; Fuad, M. A geometry-based algorithm for semaphore gesture recognition using skeleton images. *J. Theor. Appl. Inf. Technol.* **2015**, *81*, 102–107. https://doi.org/10.48550/arXiv.1608.06993.
2. Zhao, Q.; Li, Y.; Yang, N.; Yang, Y.; Zhu, M. Semaphore flag signaling recognition using a convolutional neural network. In Proceedings of the IEEE International Conference on Signal and Image Processing (ICSIP), Beijing, China, 13–15 August 2016; pp. 466–470. https://doi.org/10.1109/SIPROCESS.2016.7888306.
3. Li, W.; Yang, Y.; Wang, M.; Zhang, L. Convolutional neural network architecture for semaphore secognition. In Proceedings of the IEEE 14th International Conference on Automation Science and Engineering (CASE), Beijing, China, 23–25 November 2018; pp. 559–562.
4. Rachmad, A.; Fuad, M. Skeleton-based semaphore gesture recognition using geometrical algorithm. In Proceedings of the International Conference on Electrical Engineering and Computer Science (EECSI), Malang, Indonesia, 16–18 October 2018; pp. 219–224.
5. Putra, L.S.A.; Sumarno, L.; Gunawan, V.A. The recognition of semaphore letter code using Haar wavelet and Euclidean function. In Proceedings of the International Conference on Electrical Engineering and Computer Science (EECSI), Malang, Indonesia, 16–18 October 2018; pp. 759–763. https://doi.org/10.1109/EECSI.2018.8752707.
6. Li, K.; Wang, S.; Zhang, X.; Xu, Y.; Xu, W.; Tu, Z. Pose pecognition with cascade transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 1944–1953. https://doi.org/10.1109/CVPR46437.2021.00198.
7. Ur Rehman, M.; Ahmed, F.; Khan, M.A.; Tariq, U.; Alfouzan, F.A.; Alzahrani, N.M.; Ahmad, J. Dynamic hand gesture recognition using 3D CNN and LSTM networks. *Comput. Mater. Contin.* **2021**, *70*, 4675–4690. https://doi.org/10.32604/cmc.2022.019586.
8. Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Deveci, M.; Parmar, M. A review of convolutional neural networks in computer vision. *Artif. Intell. Rev.* **2024**, *57*, 99. https://doi.org/10.1007/s10462-024-10721-6.
9. Nair, V.; Hinton, G.E. Improving restricted Boltzmann machines using rectified linear units. In Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel, 21–24 June 2010; pp. 807–814.

10. Aruna, M.; Kaneriya, G.; Jain, P. Leveraging pre-trained ResNet-18 with transfer learning for yoga posture classification. In Proceedings of the 2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON), Bengaluru, India, 9–10 August 2024; pp. 1–5. https://doi.org/10.1109/NMITCON62075.2024.10699235.
11. He, K.; Zhang, X.; Ren, S.; Sun, J. Residual learning framework for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
12. Xue, Y.; Wang, W.; Fang, M.; Guo, Z.; Ning, K.; Wang, K. Research on a High-Efficiency Goat Individual Recognition Method Based on Machine Vision. *Animals* **2024**, *14*, 3509. https://doi.org/10.3390/ani14233509.
13. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
14. Dabwan, B.A.; Jadhav, M.E.; Gadkari, A.S.; Ali, Y.A.; Almula, S.M.; Ismil, O.A.; Mohammad, A.A. Hand gesture classification for individuals with disabilities using the DenseNet121 model. In Proceedings of the 2024 International Conference on Advancements in Power, Communication and Intelligent Systems (APCI), Kannur, India, 21–22 June 2024; pp. 1–5. https://doi.org/10.1109/APCI61480.2024.10616504.
15. Jeon, Y.J.; Park, S.; Lee, H.; Kim, H.Y.; Jung, D.H. Deep Learning-Based Model for Effective Classification of Ziziphus jujuba Using RGB Images. *AgriEngineering* **2024**, *6*, 4604–4619. https://doi.org/10.3390/agriengineering6040263.
16. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.
17. Sapkota, R.; Meng, Z.; Karkee, M. Synthetic meets authentic: Leveraging LLM generated datasets for YOLO11 and YOLOv10-based apple detection through machine vision sensors. *Smart Agric. Technol.* **2024**, *9*, 100614. https://doi.org/10.1016/j.atech.2024.100614.
18. Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.-L.; Yong, M.G.; Lee, J.; et al. MediaPipe: A Framework for building multimodal machine learning pipelines. *arXiv* **2020**, arXiv:1906.08172.
19. Sánchez-Vicinaiz, T.J.; Camacho-Pérez, E.; Castillo-Atoche, A.A.; Cruz-Fernandez, M.; García-Martínez, J.R.; Rodríguez-Reséndiz, J. MediaPipe frame and convolutional neural networks-based fingerspelling detection in Mexican sign language. *Technologies* **2024**, *12*, 124. https://doi.org/10.3390/technologies12080124.
20. Chen, J. DCM-GIFT: An Android malware dynamic classification method based on gray-scale image and feature-selection tree. *Inf. Softw. Technol.* **2024**, *176*, 107560. https://doi.org/10.1016/j.infsof.2024.107560.