Chosen Algorithm: Convolution Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms specifically designed for processing grid-like data, such as images. CNNs are highly effective at automatically detecting spatial hierarchies in images, thanks to their convolutional layers, which apply filters to input data and preserve the spatial relationship between pixels. This ability to learn and extract meaningful features from raw pixel data without the need for manual feature engineering makes CNNs particularly well-suited for tasks like image classification, object detection, and segmentation. In the case of YOLOv9, CNNs are leveraged to perform end-to-end object detection, where the model simultaneously predicts bounding boxes and class labels for multiple objects within an image. We chose CNNs for YOLOv9 due to their proven performance in computer vision tasks, particularly for real-time detection. The architecture of YOLOv9 builds upon the core strengths of CNNs but enhances them with additional layers and techniques that improve detection accuracy and speed. The model is designed to predict object locations and classifications in a single pass through the network, making it incredibly fast compared to traditional object detection methods. We implemented CNN-based architectures in YOLOv9 by employing various convolutional layers, batch normalization, and activation functions like ReLU to extract features from input images. The outputs from these layers are then processed by fully connected layers that predict bounding box coordinates, object confidence scores, and class probabilities. Additionally, YOLOv9 uses multi-scale detection to handle objects of various sizes effectively, further refining the model's ability to identify and localize objects across diverse scenes. This structure allows YOLOv9 to offer high performance in terms of both accuracy and computational efficiency, making it ideal for real-time applications.

Implementation:

- Model Training:
    - YOLOv9n (nano version) was used for faster training with limited computational resources.
    - The model was trained using the Ultralytics YOLOv9 library with custom configuration files.
    - Hyperparameters like learning rate, batch size, and image size (640x640) were adjusted to optimize performance.

    - Data augmentation was enabled within the YOLOv9 pipeline itself during training.

- Testing and Inference:

    - After training, real-time testing was performed using webcam feeds and separate test datasets.
    - Predicted bounding boxes and class labels were visualized and saved.

Model Performance

- Accuracy:

    - Achieved high precision (0.9375) and recall(0.86948) across validation and test datasets, with mAP50 scores consistently above 0.85.

- Loss Curves:

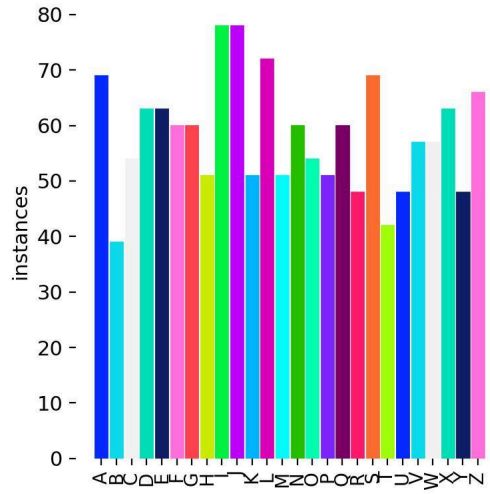
    - Loss curves for box loss, objectness loss, and classification loss showed steady convergence without overfitting.

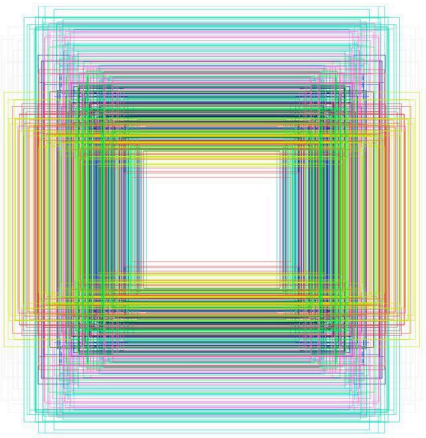
    - Early stopping was applied to avoid unnecessary training epochs when validation loss plateaued.

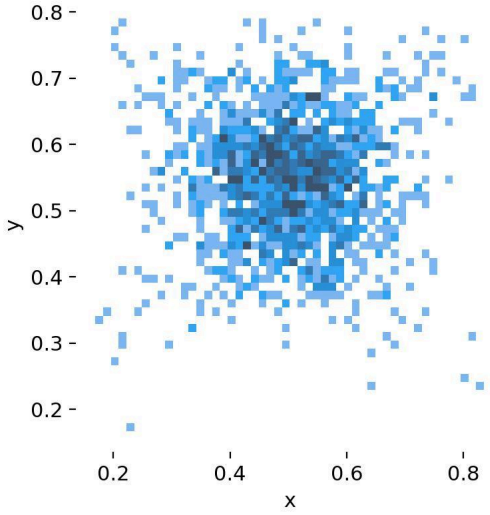
- Hyperparameter Comparison:

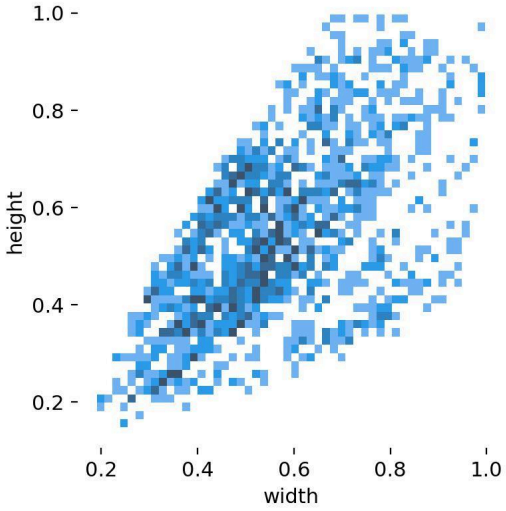
    - Lower learning rates (0.001) resulted in smoother convergence but required longer training time.
    - Data augmentation increased model robustness by reducing validation loss variance.
    - Increasing image size from 224 x 224 to 640 x 640 boosted detection precision by capturing finer hand details.

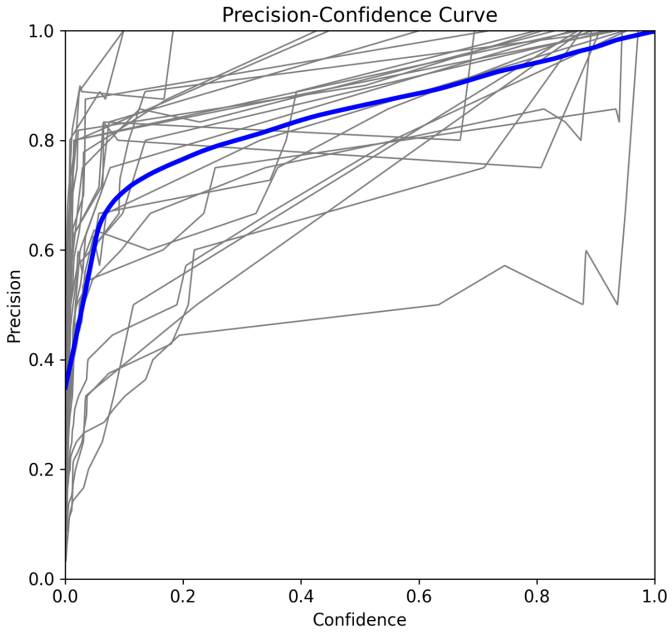
Appendix

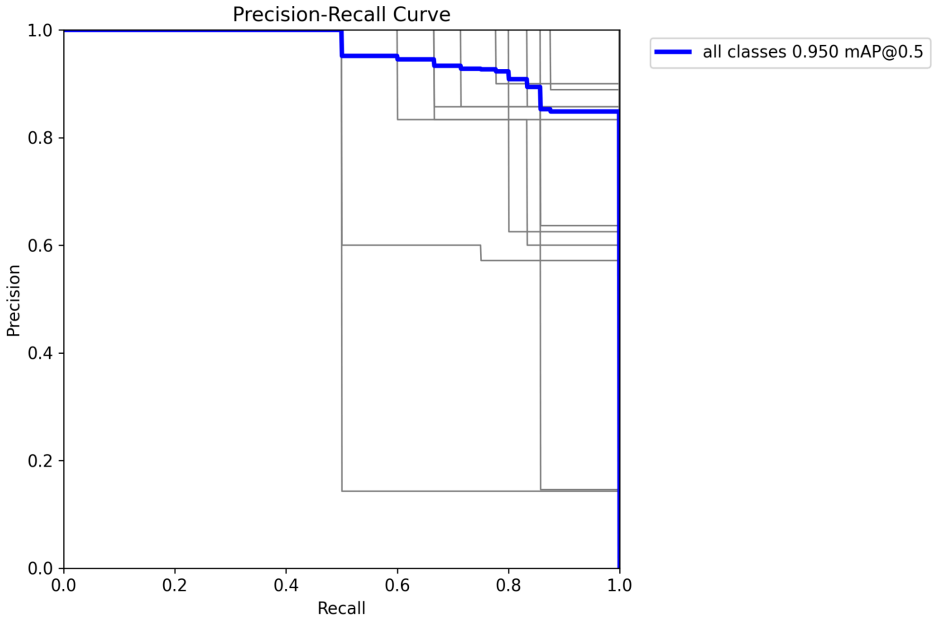

instances
80
70
60
50
40
30
20
10
0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

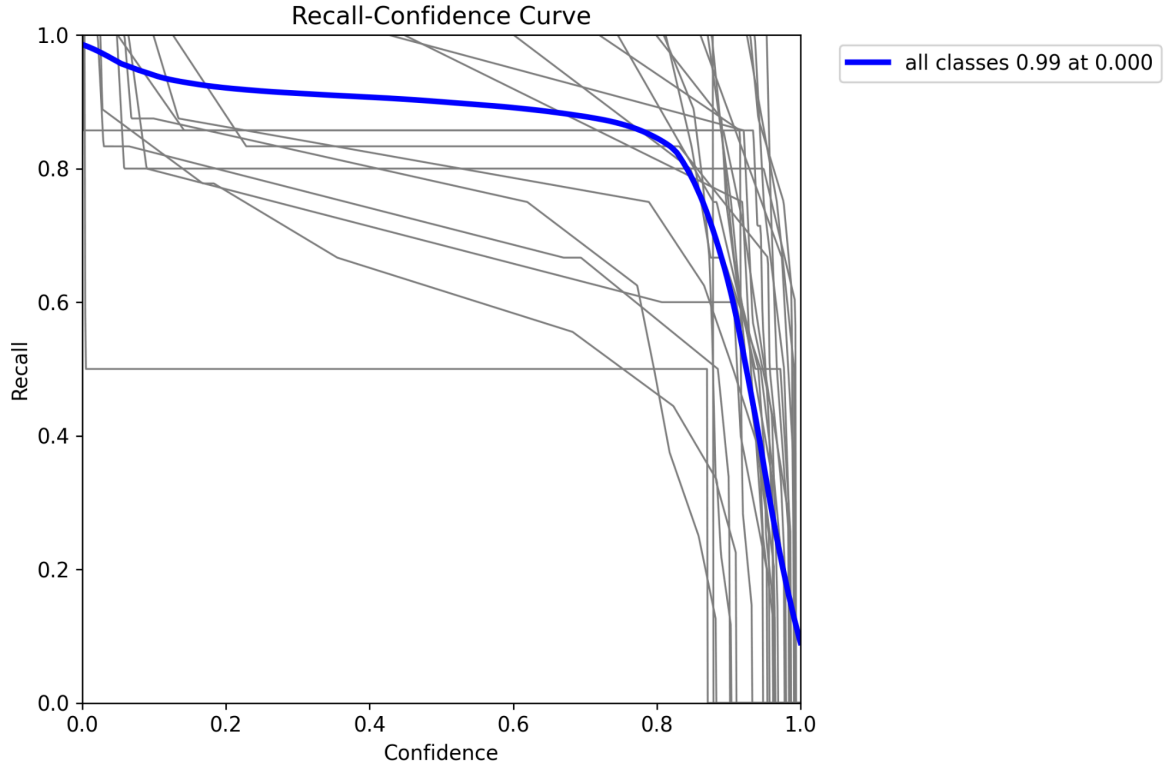



y
x
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.4
0.6
0.8


height
width
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0


Precision-Confidence Curve
all classes 1.00 at 1.000
Precision
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve
all classes 0.950 mAP@0.5
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Recall-Confidence Curve
all classes 0.99 at 0.000
Recall
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0