**Real-Time Object Detection and Classification of Face Masks: A Comprehensive Project Report**

**1. Objective**

The primary objective of this project is to develop a robust real-time object detection system capable of identifying and classifying face masks. This system leverages deep learning methodologies to achieve high accuracy in detecting three specific classes:

1. No Mask
2. Correctly Worn Mask
3. Incorrectly Worn Mask

The project employs convolutional neural networks (CNNs) integrated with object detection algorithms, including SSD (Single Shot Multibox Detector) and YOLO (You Only Look Once), to achieve precise object detection with bounding boxes and class labels.

**2. Dataset**

The dataset used for this project is the "Face Mask Detection" dataset, which is publicly available on Kaggle. It contains images annotated for the presence of face masks, correctly-worn masks, and incorrectly-worn masks. Each image includes bounding boxes and class labels, making it ideal for training focused detection models.

* **Dataset Link**: [Face Mask Detection on Kaggle](https://www.kaggle.com/datasets/andrewmvd/face-mask-detection)

**Dataset Characteristics:**

* Annotated images with bounding boxes and class labels
* A variety of lighting conditions and backgrounds
* Challenges include occlusions, clutter, and diverse face orientations

**3. Project Tasks**

**3.1. Dataset Exploration**

The dataset was explored to understand its structure, including:

* Class distribution across the three categories
* Image dimensions and resolutions
* The variety of lighting conditions and backgrounds

**3.2. Preprocessing**

To ensure compatibility with neural network models, the following preprocessing steps were implemented:

* **Resizing**: Standardized image dimensions to fit the model’s input requirements.
* **Normalization**: Pixel values were scaled to the range [0, 1] for consistent gradient updates during training.
* **Color Space Conversion**: Optional conversions to alternative color spaces (e.g., grayscale) to emphasize object features.
* **Augmentation**: Random rotations, flips, and brightness adjustments were applied to improve generalization.

**3.3. Model Training**

Two CNN-based object detectors were trained:

* **SSD (Single Shot Multibox Detector)**
* **YOLO (You Only Look Once)**

**1. Overview of SSD**

The Single Shot MultiBox Detector (SSD) is a popular convolutional neural network (CNN) designed for object detection. SSD achieves high-speed and accurate detection by predicting object categories and bounding box offsets directly from feature maps in a single forward pass. Unlike region-based approaches, SSD eliminates the need for regional proposals, making it suitable for real-time applications. Key features include:

**2. Goals**

1. Implement a robust SSD model for real-time object detection.
2. Optimize the network for speed and accuracy on custom datasets.
3. Mitigate overfitting and ensure generalization across various environments.
4. Evaluate model performance using relevant metrics and visualizations.

**3. Comparison to YOLO**

SSD and YOLO (You Only Look Once) are both real-time object detection architectures, but they differ in key aspects:

* Prediction Mechanism:
  + SSD uses multiple feature maps for detection at various scales, improving performance on smaller objects.
  + YOLO predicts bounding boxes and class probabilities from a single feature map.
* Speed and Accuracy:
  + YOLO is faster but may sacrifice accuracy, especially for small objects.
  + SSD offers a better trade-off between speed and accuracy.
* Complexity:
  + SSD’s use of default boxes and multi-scale features adds complexity compared to YOLO’s simpler architecture.

**4. Overfitting**

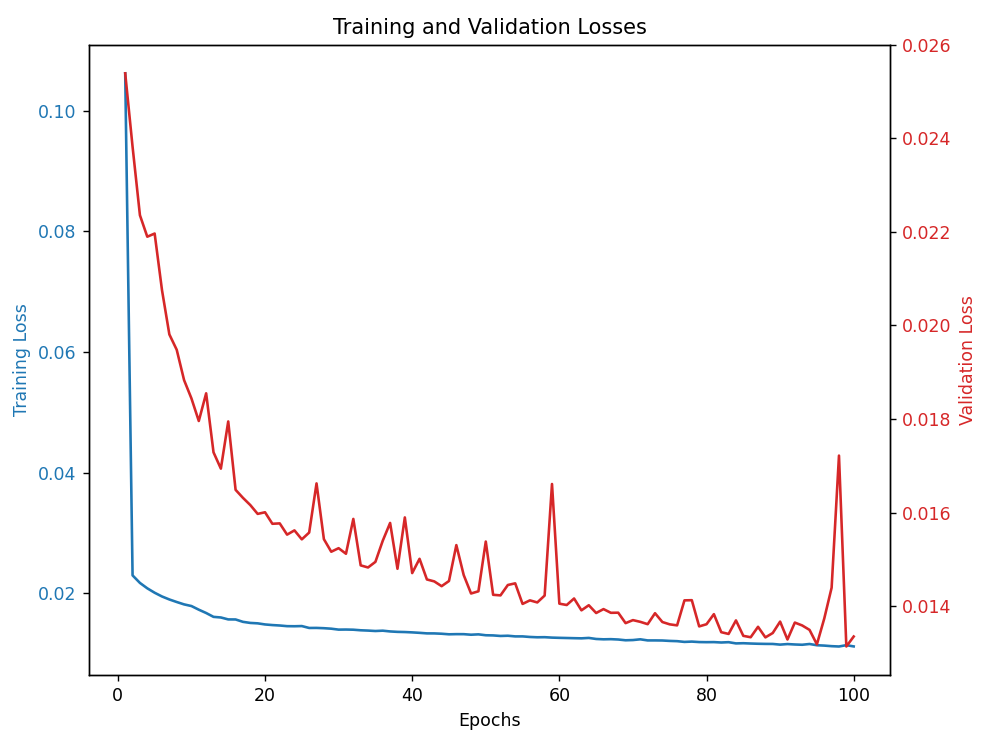
Overfitting occurs when the SSD model memorizes training data rather than generalizing. Strategies to address this include:

* **Regularization:** L1 loss and weight decay has been implemented in the code.
* **Early Stopping:** Overtraining a model has a tendency to cause overfitting. Early stopping detects when the learning of the model has halted and ends training early.,
* **Learning Rate Optimization:** By creating a scheduler, it can allow our code to learn slower as time passes, when learning fast is detrimental to our model.
* **Invalid Data Filtering:** While training, false data can cause the model to learn wrong. By filtering blatantly wrong data out we can cause our data to learn better. Negative Masking helps in this.

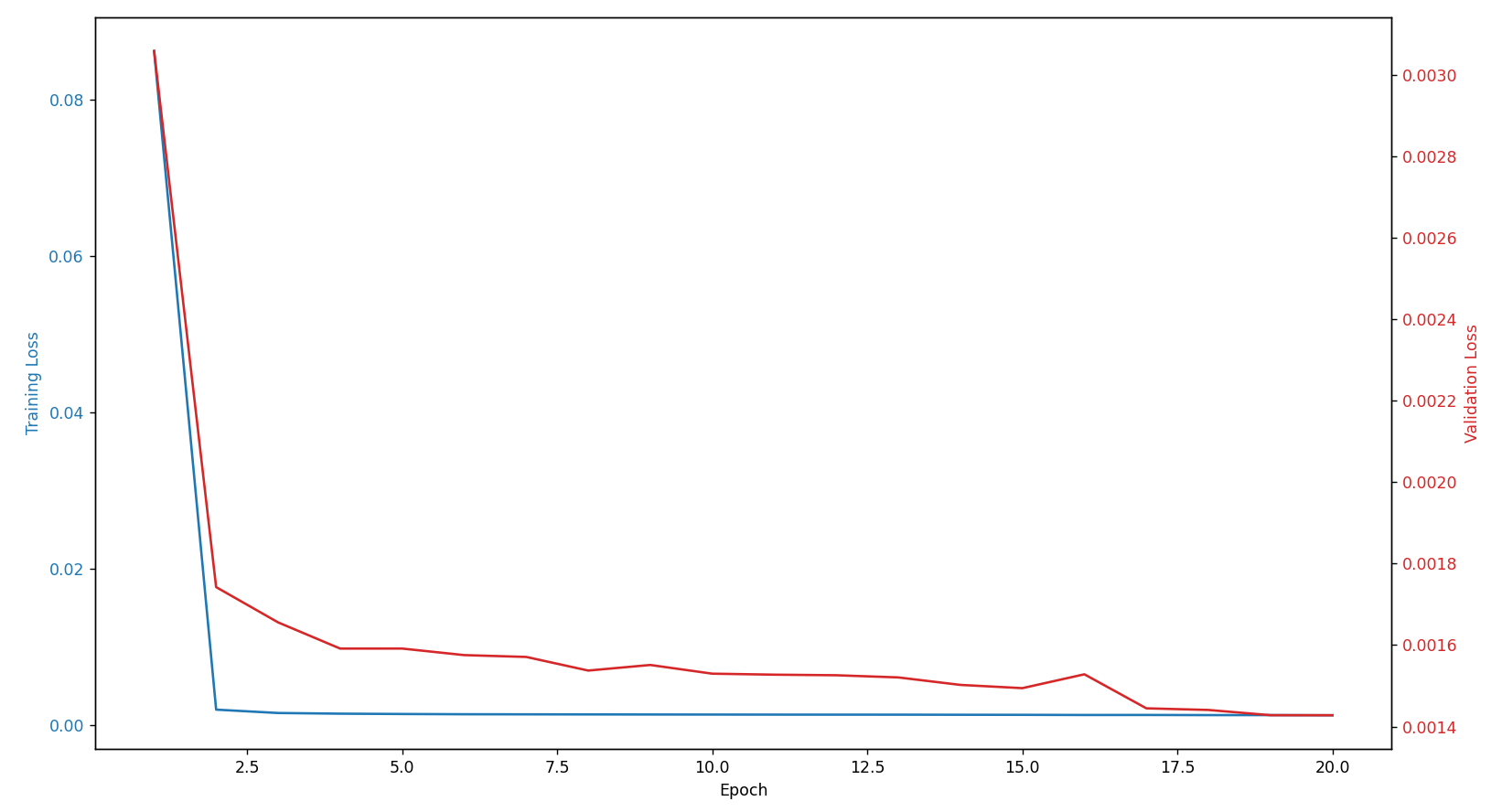
**5. Metrics and Graphs**

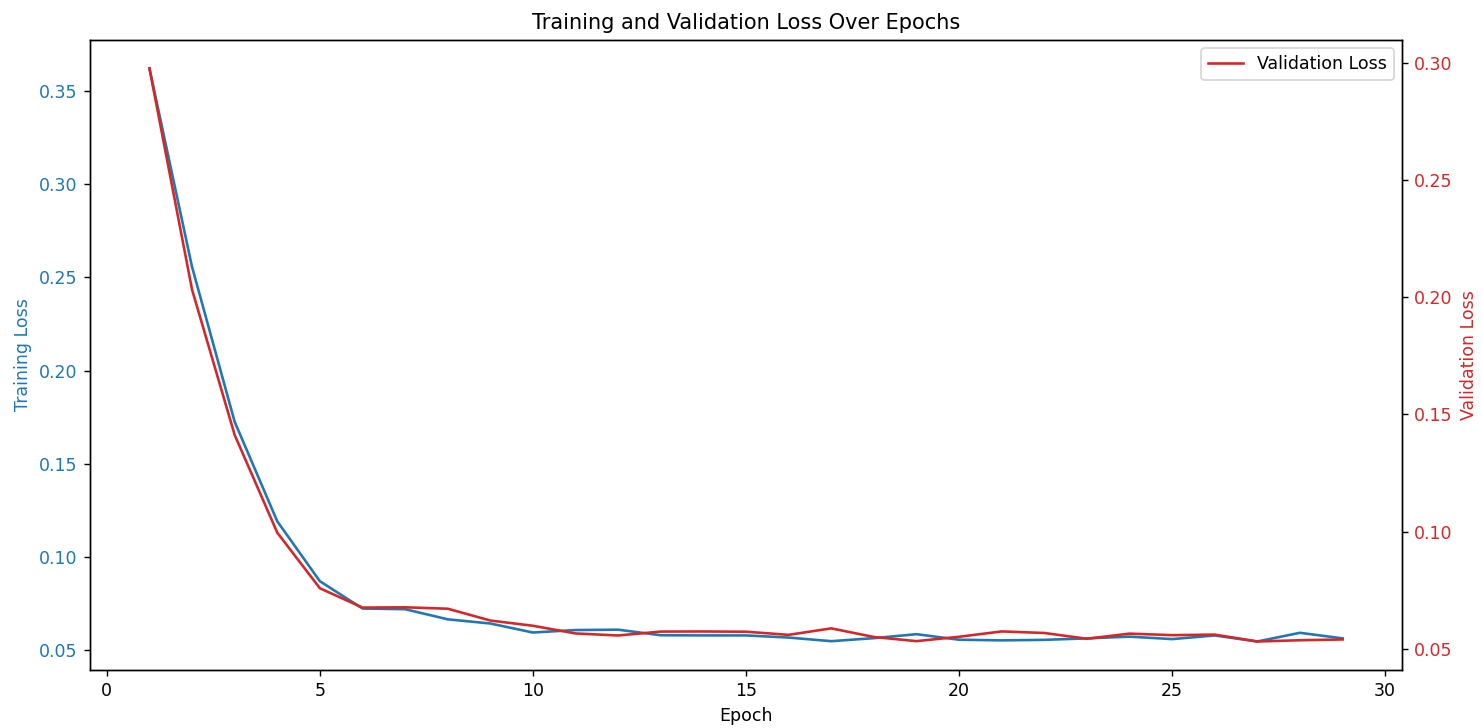
To evaluate SSD performance, key metrics include:

1. **Validation loss, Training loss and Time:** Time taken to process a single image, indicative of real-time capabilities.

At the start of the project overfitting was a very big issue. In this graph no overfitting prevention technique was used. This caused instability and overfitting to be prevalent

Some iterations later I added early stopping and L1 regularization. While the instability went away there was still a gap between training and validation loss.

****

Towards the end of the project I implemented schedulers, iou validation, negative masking and class weights. These techniques allowed my code to learn more smoothly and consistently. 

1. **Precision Recall F1:** Metrics that show how good our model is at giving reliable outputs:

**Metrics at the Metrics at the**

A screen shot of a computer

Description automatically generated**beginning of the project end of the project**

A screen shot of a computer

Description automatically generated

1. **IoU Thresholds:** Evaluates the localization accuracy of predicted bounding boxes. This is included in our evaluation function of the code. And adds its impact to validation loss.

**6. Conclusion**

The Single Shot MultiBox Detector (SSD) is a powerful framework for real-time object detection, balancing speed and accuracy through its multi-scale design. Addressing challenges such as overfitting and optimizing specific datasets ensures its effectiveness in practical applications.

**Overview of YOLO**

The "You Only Look Once" (YOLO) framework is a state-of-the-art object detection algorithm designed for speed and simplicity. YOLO predicts bounding boxes and class probabilities directly from a single feature map in one forward pass, making it highly efficient for real-time applications. Key features include:

1. **Unified Architecture**: YOLO frames object detection as a single regression problem, combining classification and localization tasks.
2. **High Speed**: By processing the entire image at once, YOLO achieves real-time detection speeds, even on standard hardware.
3. **Global Context Awareness**: YOLO evaluates the entire image context, reducing false positives from local ambiguities.

**Goals**

1. Implement a real-time YOLO model for mask detection with high-speed inference.
2. Achieve competitive performance metrics, focusing on precision and recall.
3. Enhance generalization across varied conditions by minimizing overfitting.
4. Evaluate and visualize results, comparing YOLO's strengths and weaknesses to SSD.

**Addressing Overfitting in YOLO**

YOLO is prone to overfitting, especially when trained on smaller datasets or without regularization. To address this, the following strategies were employed:

1. **Regularization**:

Applied dropout layers to prevent over-reliance on specific features.

Used L2 weight regularization to penalize overly complex models.

1. **Early Stopping**:
   * Monitored validation loss to terminate training when improvements plateaued.
2. **Learning Rate Optimization**:
   * Implemented a learning rate scheduler to gradually reduce the learning rate, promoting finer adjustments during later epochs.
3. **Data Augmentation**:
   * Used techniques such as random cropping, flipping, and brightness variations to expand the effective training dataset.

**Metrics and Graphs**

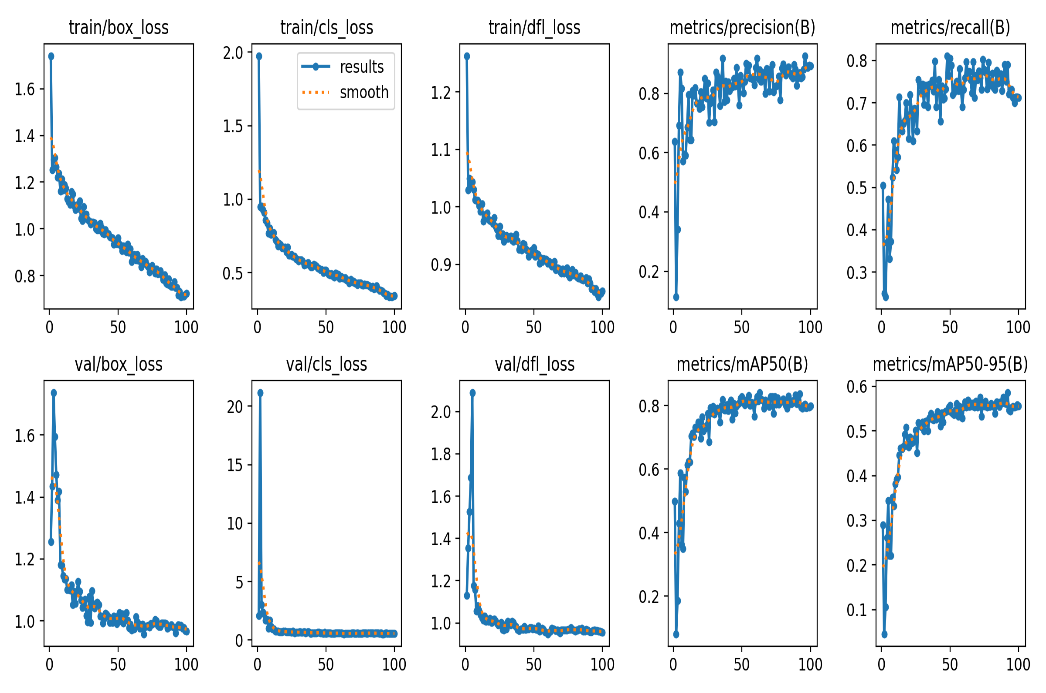
Key metrics were used to evaluate YOLO's performance:

1. **Training and Validation Loss**:

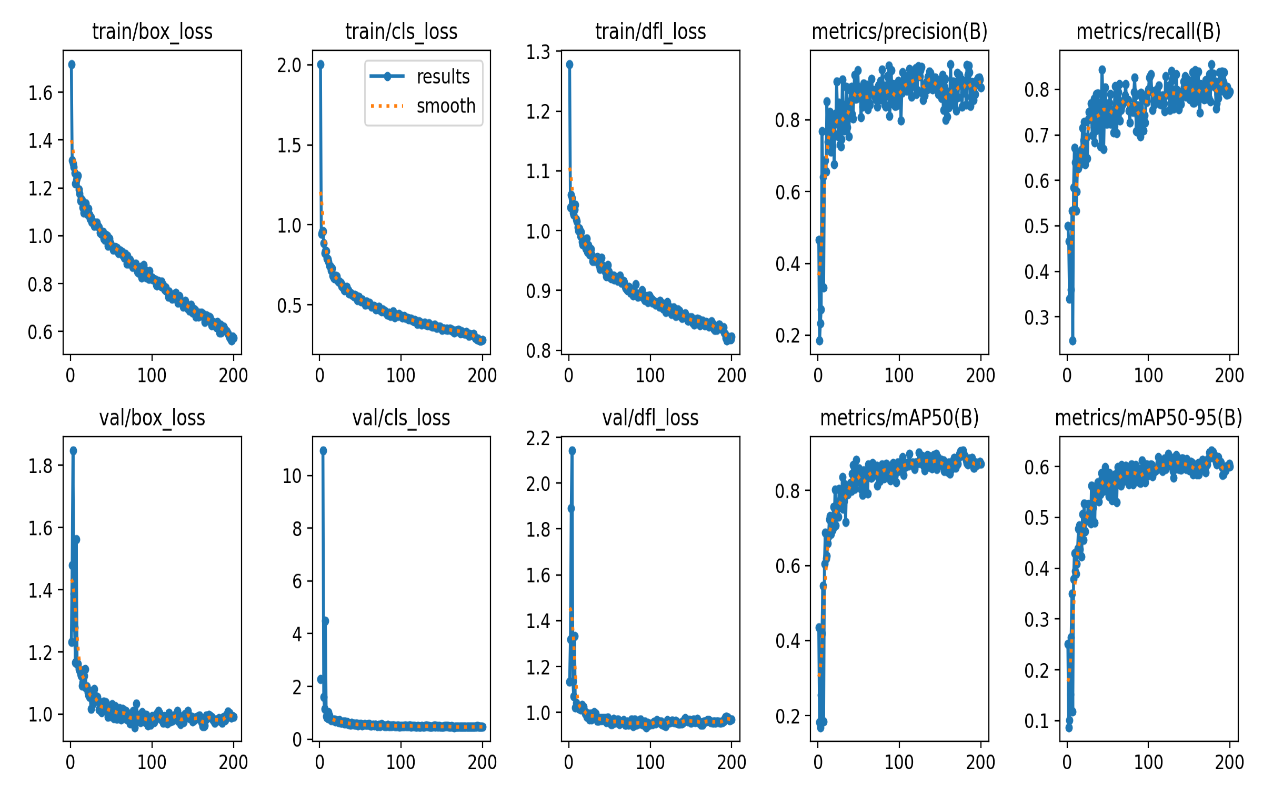
Graphs revealed initial instability, which was mitigated by regularization and learning rate adjustments.

Improvements in training consistency were observed after applying augmentation techniques.

First Training (100 epochs):



3. Training (200 epochs + augmentation):



1. **Precision, Recall, and F1 Scores**:

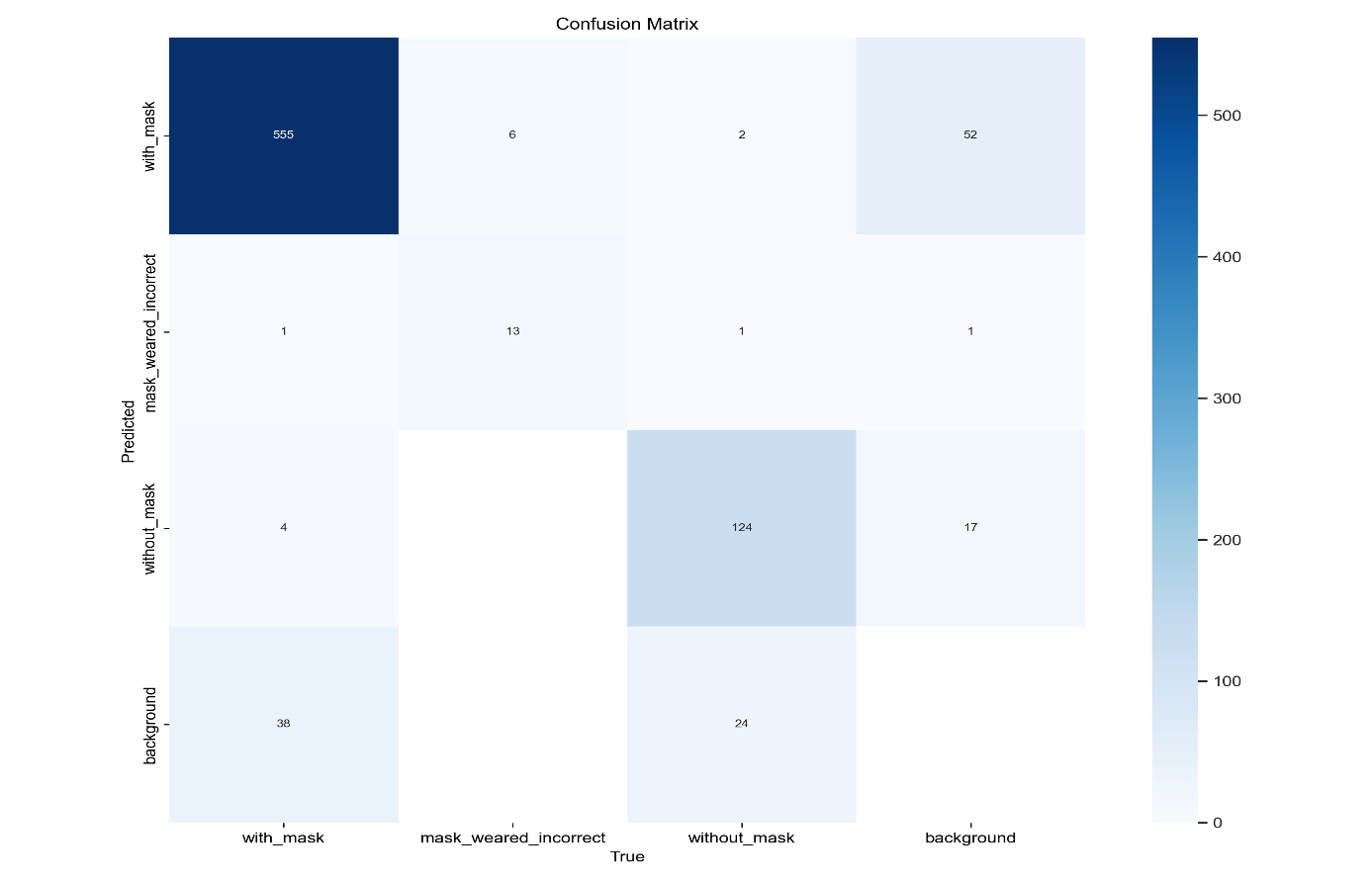
Metrics demonstrated significant improvement over time, especially in distinguishing between "No Mask" and "Incorrectly Worn Mask" classes.

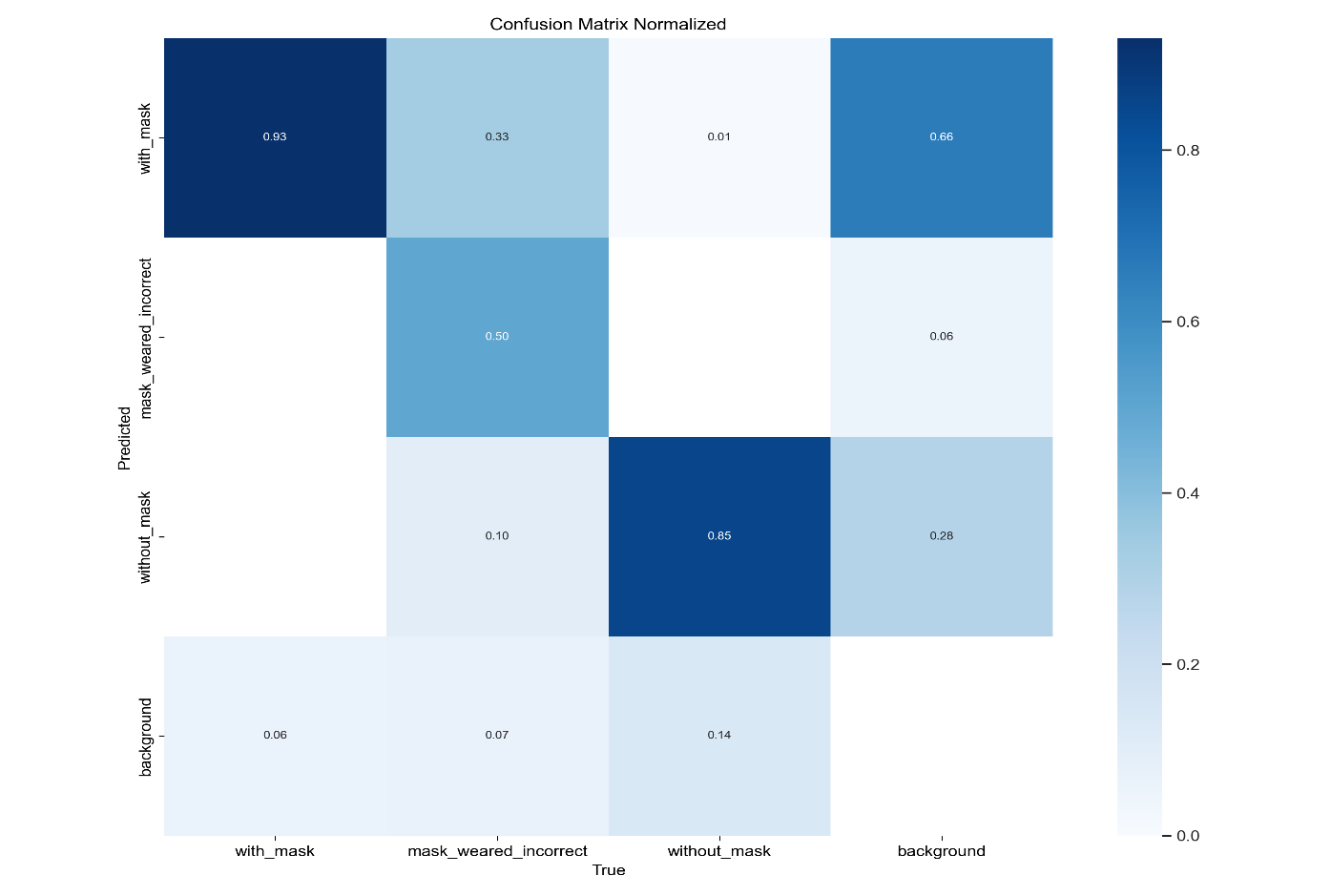
1. **IoU Validation**:

* Evaluated the accuracy of bounding box predictions, integrated into the loss function to optimize localization.

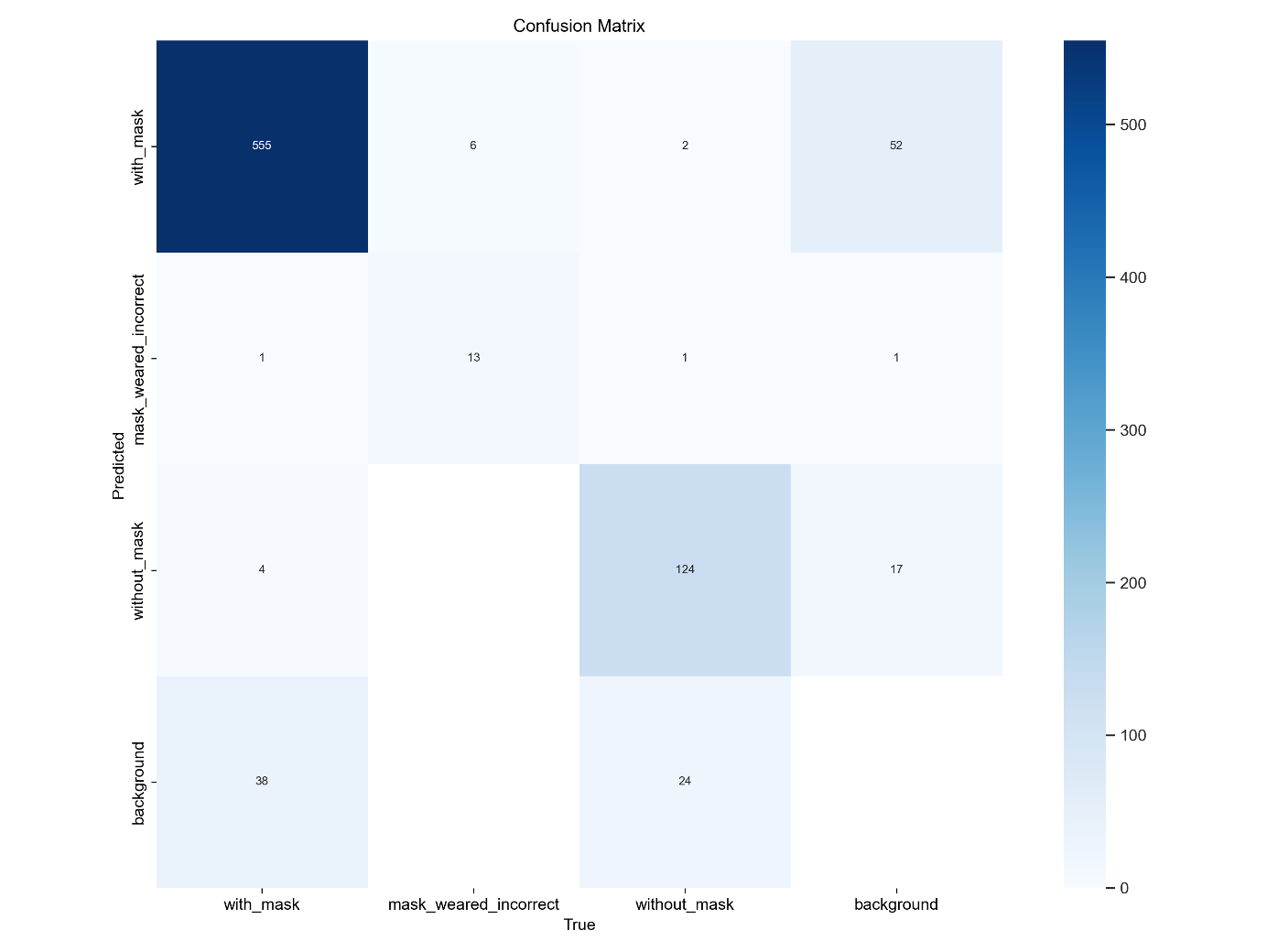
1. Confusion Matrix

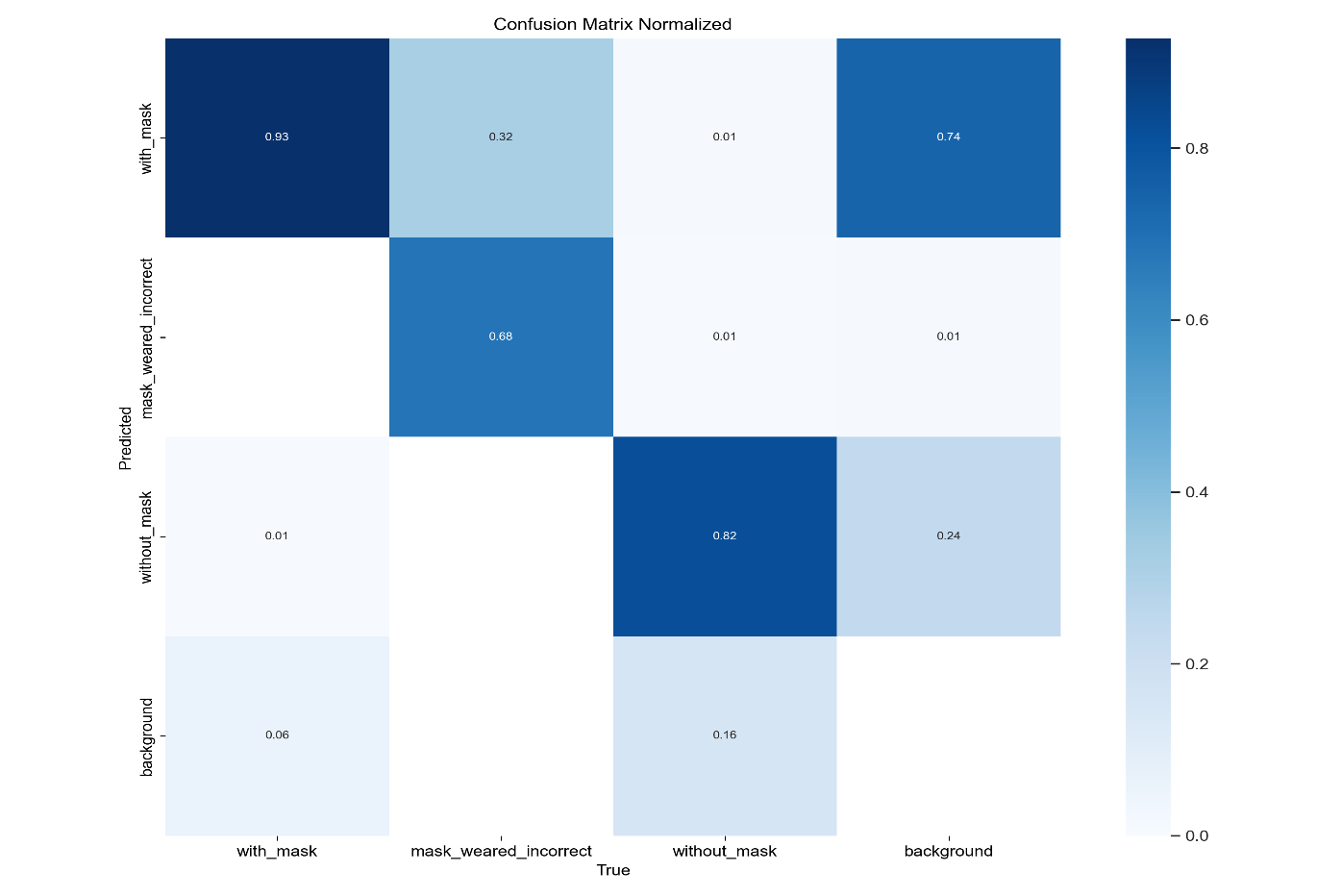
First Training (100 epochs):





Confusion Matrix 3. Training (200 epochs + augmentation):





**YOLO Development Phase**

We used YOLO v11 for training and converted the dataset to the YOLO bounding box and labeling format. The dataset was split into three subsets: training, validation, and test. Initially, we trained the YOLO model for 100 epochs. While the results seemed promising at first, testing the model using live webcam feed revealed issues, particularly with the masked\_worn\_incorrectly class. Additionally, the model struggled in low-light conditions.

Upon reviewing the metrics and confusion matrix, we observed that only 50% of the masked\_worn\_incorrectly class was correctly identified, with frequent misclassifications as the with\_mask class. To address this, we analyzed the class distribution across the dataset splits and noticed that the validation set had more than enough examples of the masked\_worn\_incorrectly class. Consequently, we moved some of these examples to the training set.

To improve performance under different lighting conditions, we added the RandomBrightnessContrast augmentation. After incorporating these updates, we retrained the model. The new model performed significantly better, especially for the masked\_worn\_incorrectly class. Testing with live webcam feed also confirmed this improvement.

Since there were no signs of overfitting, we decided to extend training to 200 epochs. After this round, the best-performing model corresponded to the 116th epoch, which slightly outperformed the previous model. Out of curiosity, we also trained the model for 500 epochs. However, YOLO stopped training around 250 epochs as there were no noticeable improvements, and the best-performing epoch remained the 116th.

We also used images from daily life, including some from the COVID era with masks worn correctly, incorrectly, or not at all, along with images containing no humans. The initial model mistakenly identified some non-human images, such as skies with varying brightness and contrast, as no\_mask instances. It also struggled to differentiate between the masked\_worn\_correctly and masked\_worn\_incorrectly classes. However, after retraining, every image was predicted correctly, and the model showed substantial improvement in its classification accuracy.

**3. Training model’s evaluation result on test set (YOLO uses validation set when training)**

**Precision: 0.9508054297403129**

**Recall: 0.8218425270684026**

**mAP@0.5: 0.8968360637738604**

**mAP@0.5:0.95: 0.648751271052077**

**Mean IoU: 0.8484409162521269**

**Filtered IoU list length 292**