**Drowsiness Detection Using Computer Vision with YOLOv11n**

**Project Overview**

This project involves the development of a **drowsiness detection system** using computer vision. A pre-trained **YOLOv11n** model is employed for this task, which is fine-tuned to detect states of alertness (drowsiness, awake, and yawning) from images. YOLOv11n, a variant of the YOLO family of object detection models, is known for its efficiency and precision, making it ideal for real-time applications in safety-critical environments such as driver monitoring systems or workplace fatigue detection.

**YOLO** (You Only Look Once) is one of the most popular object detection algorithms used in computer vision, known for its speed and efficiency. It is designed to detect and classify objects in images or videos in real-time by treating object detection as a single, unified problem. In contrast to earlier methods that used region proposals and multiple stages for classification and localization, YOLO processes an image in one go, predicting bounding boxes and class labels simultaneously.

**How YOLO Works**

**Grid Division:** The first step in YOLO involves dividing the input image into a grid of cells (often 7x7 or 13x13, depending on the input size and model version). Each cell in the grid is responsible for detecting objects whose center falls within it. This grid structure is critical as it enables the model to efficiently learn which objects appear in specific regions of an image.

1. **Bounding Box Prediction**: Each grid cell is tasked with predicting a fixed number of bounding boxes. A bounding box is defined by four parameters:
   * x: the x-coordinate of the box center relative to the grid cell.
   * y: the y-coordinate of the box center relative to the grid cell.
   * w: the width of the bounding box.
   * h: the height of the bounding box.

Additionally, each bounding box has a confidence score, which represents the model's certainty that an object exists within that box. The **confidence score** is calculated by multiplying two factors:

* + **Objectness score:** the probability that the bounding box contains an object.
  + **Intersection over Union (IoU):** how well the predicted bounding box overlaps with the ground truth box.

1. **Class Prediction:** In addition to bounding boxes, each grid cell predicts the probability of each object class being present within the bounding box. For example, if YOLO is trained on 80 object categories, each grid cell will predict a probability distribution over these 80 classes. This helps the model classify the detected object.
2. **Non-Maximum Suppression (NMS):** After the network outputs the predicted bounding boxes and their associated class probabilities, many of the predicted bounding boxes may overlap. To resolve this, YOLO uses Non-Maximum Suppression (NMS) to remove redundant bounding boxes, keeping only the ones with the highest confidence score. NMS ensures that each object is detected by only one bounding box and is classified correctly.

**YOLO Output for Predictions**

The output of YOLO's predictions consists of:

* **Bounding Boxes:** Each bounding box is represented by the coordinates (x, y, w, h) and a confidence score that indicates the probability of an object existing within that box.
* **Class Probabilities:** For each bounding box, YOLO provides a set of class probabilities, each representing the likelihood that the object belongs to a particular class (e.g., person, car, dog, etc.).

**Advantages of YOLO:**

* **Real-Time Detection**: YOLO's design allows it to make predictions in real time, making it suitable for applications that require fast object detection, such as autonomous vehicles or video surveillance.
* **Unified Architecture**: Since YOLO processes the entire image at once, it is computationally efficient and avoids the need for separate stages in object detection, unlike traditional methods.
* **Global Context**: YOLO looks at the entire image during training and testing, enabling it to make better predictions based on context and spatial relationships between objects.

**Pre-trained YOLOv11n Model and the COCO Dataset**

The **YOLOv11n** model used in this project is pre-trained on the **COCO dataset**, which provides a robust foundation for object detection tasks. Pre-training on a large and diverse dataset like COCO allows the model to learn basic features and patterns that are applicable across a wide range of images, making it easier to adapt to specialized tasks such as detecting drowsiness and other states of alertness.

**What is the COCO Dataset and Why is it Important for Computer Vision?** The **COCO dataset** (Common Objects in Context) is one of the most widely used datasets in the computer vision community. It contains over **330,000 images** annotated with detailed information about **80 object categories**. The dataset includes various object types that range from everyday items to animals and vehicles, making it versatile and suitable for a wide array of computer vision tasks, including object detection, segmentation, and image captioning.

These object categories, learned from COCO, enable YOLOv11n to detect a wide range of objects, making the model versatile and effective in **detecting various patterns and features in images.**

**Our Custom Dataset**

**Link:** [**Drowsiness detection for YOLOv8**](https://www.kaggle.com/datasets/cubeai/drowsiness-detection-for-yolov8)

While the **COCO dataset** served as the foundation for pre-training the model, the model was fine-tuned using a **custom dataset** designed to detect **drowsiness**, **wakefulness**, and **yawning**. This dataset consists of images captured in controlled environments where subjects either demonstrate alertness or display signs of drowsiness or yawning.

Our custom dataset is classified into three categories:

1. **Drowsiness**: Images depicting individuals exhibiting signs of tiredness or drowsiness.
2. **Awake**: Images of individuals who are alert and awake.
3. **Yawn**: Images where individuals are caught in the act of yawning.

By leveraging the pre-trained YOLOv11n model, we fine-tuned the model to accurately detect these three states, making it well-suited for real-time applications.

**The architecture** **of our model**

Here's a detailed breakdown of the architecture and the number of convolutional layers:

**Total Convolutional Layers**

The model contains a total of **88 convolutional layers**. These layers are distributed across the **backbone**, **neck**, and **detection head**.

**Breakdown of Convolutional Layers**

**1. Backbone**

The backbone is responsible for feature extraction. It consists of:

* **Initial Convolutions**: Standard convolutional layers for downsampling and feature extraction.
* **C3k2 Blocks**: Custom blocks with multiple convolutional layers for feature refinement.
* **Bottleneck Layers**: Used within C3k2 blocks to reduce computational complexity.

**Number of Convolutional Layers in Backbone**: **63**

**2. Neck**

The neck connects the backbone to the detection head and is responsible for feature fusion. It consists of:

* **Upsampling Layers**: To increase spatial resolution.
* **Concatenation Layers**: To combine features from different scales.
* **Additional C3k2 Blocks**: For further feature refinement.

**Number of Convolutional Layers in Neck**: **0** (The neck primarily uses upsampling and concatenation, which do not involve convolutional layers.)

**3. Detection Head**

The detection head is responsible for predicting bounding boxes, objectness scores, and class probabilities. It consists of:

* **Convolutional Layers**: For refining features and making predictions.
* **DFL (Dynamic Focus Layer)**: A specialized layer for bounding box regression.

**Total Convolutional Layers**

* **Backbone**: 63
* **Detection Head**: 25
* **Total**: **88**

In YOLO (You Only Look Once), the architecture is designed to perform both object detection and classification efficiently in a single forward pass. A key difference between YOLO and traditional object detection models is the use of **1x1 convolutional layers** instead of **fully connected layers** for predicting bounding boxes, objectness scores, and class probabilities. Here's an explanation of this design choice and how it works:

**1. 1x1 Convolutional Layers:**

* **Convolutional layers** are typically used in CNNs (Convolutional Neural Networks) to extract spatial features from images. The idea of using 1x1 convolutions is to perform feature extraction while maintaining spatial dimensions.
* A **1x1 convolutional layer** is a specialized type of convolution that uses a kernel of size 1x1 (i.e., each filter only looks at one pixel at a time).

**2. Why 1x1 Convolutions?**

* **Efficiency**: Using 1x1 convolutions helps reduce the computational complexity compared to fully connected layers, especially in deep networks. Fully connected layers have a large number of parameters because every node in the layer is connected to all neurons in the previous layer. 1x1 convolutions drastically reduce the number of parameters while still enabling rich feature extraction and prediction.
* **Spatial Hierarchy Preservation**: In typical convolutional networks, convolutional layers maintain spatial hierarchies (e.g., width and height). YOLO uses 1x1 convolutions to preserve this spatial hierarchy across the network. Instead of flattening the image into a vector (which would happen with fully connected layers), the spatial dimensions of the input image are maintained throughout the network. This allows the model to make predictions about bounding boxes and class probabilities at each pixel or grid cell, maintaining spatial coherence.

**3. How YOLO Uses 1x1 Convolutions for Prediction:**

* **Grid Cells**: YOLO divides the image into an S×SS \times SS×S grid, where each grid cell is responsible for detecting objects that fall within that cell. For each cell in the grid, the model predicts the following:
  + **Bounding Box Coordinates**: Each grid cell predicts the coordinates of the bounding box, represented by (x,y,w,h)(x, y, w, h)(x,y,w,h).
  + **Objectness Score**: The objectness score indicates the probability that a bounding box contains an object (as opposed to background).
  + **Class Probabilities**: For each grid cell, YOLO predicts a set of class probabilities, where each class represents a potential object (e.g., person, car, etc.).
* **1x1 Convolution for Prediction**:
  + After the initial convolutional layers extract features from the input image, the network uses 1x1 convolutions to predict the final bounding boxes, objectness scores, and class probabilities.
  + These 1x1 convolutional layers output a tensor with a shape of S×S×(B×5+C)S \times S \times (B \times 5 + C)S×S×(B×5+C), where:
    - S×SS \times SS×S is the grid size (the number of grid cells).
    - BBB is the number of bounding boxes predicted per grid cell (usually 1 or 2).
    - The number 5 refers to the 4 bounding box parameters (x,y,w,h)(x, y, w, h)(x,y,w,h) and the **objectness score**.
    - CCC is the number of classes (e.g., person, car, dog, etc.).
* The 1x1 convolution helps generate this output by producing feature maps that represent the predicted bounding boxes, class labels, and confidence scores for every cell in the grid. Each grid cell then outputs these predictions without flattening the image or using fully connected layers.

**4. Advantages of Using 1x1 Convolutions:**

* **Reduces Computational Cost**: Since fully connected layers can have a large number of parameters (especially in deeper networks), using 1x1 convolutions reduces the number of computations and parameters, making the network more efficient.
* **Better Spatial Awareness**: By retaining the spatial dimensions of the input and using 1x1 convolutions, YOLO is better able to make spatially-aware predictions about the location and class of objects in the image.
* **Real-Time Detection**: The use of 1x1 convolutions makes YOLO a faster, more real-time detector compared to traditional object detection models like R-CNN or Fast R-CNN, which use region proposal networks (RPNs) and fully connected layers.

**YOLOv11n Training on the Custom Dataset**

The YOLOv11n model was fine-tuned on the custom dataset using the following Hyper-parameters:

* **Batch Size**: 128
* **Epochs**: 100
* **Learning Rate**: “auto” ( Adapted during training based on performance )
* **Freeze**: 11 ( *Freeze first 120 layers* )
* **Patience**: 10
* **Dropout**: 0.1

Total number of **freezed convolutional layers:** 40

**Data augmentation techniques**, such as flipping, rotation, and scaling, were applied to enhance the model's robustness and ability to generalize to new, unseen images.

**EarlyStopping:** Training stopped early as no improvement observed in last 10 epochs. Best results observed at epoch 52, best model saved as best.pt.

62 epochs completed in 2.308 hours.

**Model Training and Results**

**A screenshot of a computer

AI-generated content may be incorrect.**

The following evaluation results were on a test data:

**Evaluation Metrics:**

1. **Overall Results**:
   * **Precision (P)**: 0.963
   * **Recall (R)**: 0.94
   * **mAP50**: 0.984
   * **mAP50-95**: 0.87
2. **Class-wise Results**:
   * **Drowsiness**:
     + Precision: 0.961
     + Recall: 0.915
     + mAP50: 0.98
     + mAP50-95: 0.891
   * **Awake**:
     + Precision: 0.975
     + Recall: 0.955
     + mAP50: 0.99
     + mAP50-95: 0.913
   * **Yawn**:
     + Precision: 0.951
     + Recall: 0.951
     + mAP50: 0.983
     + mAP50-95: 0.807
3. **Inference Speed**:
   * **Preprocess**: 0.2ms per image
   * **Inference**: 1.9ms per image
   * **Postprocess**: 1.2ms per image

**Val\_batch1\_pred**

A collage of people with different facial expressions

AI-generated content may be incorrect.

**Metrics**

A group of graphs showing the value of a number of data

AI-generated content may be incorrect.

**confusion\_matrix**

**A blue squares with white text

AI-generated content may be incorrect.**

**Key Insights:**

* The **precision** of the model is high across all classes, demonstrating its ability to accurately detect drowsiness, wakefulness, and yawning.
* The **recall** for the awake and drowsiness classes is also high, meaning the model is capable of identifying these states effectively.
* The **mAP50** is exceptional for awake (0.99) and drowsiness (0.98), showcasing the model’s ability to detect these conditions with minimal false positives.
* However, **mAP50-95** reveals that the model struggles with the yawning class, suggesting room for improvement, particularly for detecting subtle variations in yawning gestures.
* **Inference speed** is quick, making the model suitable for real-time applications in driver safety systems and workplace monitoring.

**Model Optimizations:**

For further improvements, the following optimization strategies could be applied:

1. **Fine-tune Hyperparameters**: Experimenting with different learning rates, batch sizes, and epochs could enhance performance, particularly for the yawning class.
2. **Increase Data Diversity**: More diverse examples of yawning and drowsiness, including variations in lighting and posture, could improve the model’s ability to generalize.
3. **Ensemble Approaches**: Combining multiple models or employing advanced techniques such as transfer learning might boost recall and precision for harder-to-detect cases, such as yawning.