Confidence of the picture below without the patch

With Phyton Code

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture | Picture Name | Prediction | Confidence |
|  | Car | Truck  Car | 66.06  63.72 |
|  | Sheep | Sheep | 95.98 |
|  | Bottle | Bottle | 90.00 |
|  | Person | Person | 94.80 |
|  | Dog | Dog | 91.84 |

Phone Camera

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture | Picture Name | Prediction | Confidence |
|  | Car | Car | 70% |
|  | Sheep | Sheep | 66% |
|  | Bottle | Bottle | >80% |
|  | Human | Human | >90% |
|  | Dog | Dog | 80% |

Python Code

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture with Patch | Picture Name | Prediction | Confidence |
|  | 1 | Apple  Motorcycle  Apple | 55.69  53.39  34.62 |
|  | 2 | Bowl  Bear  Apple  Apple  Apple  Apple  Sheep | 74.46%  66.36%  61.09%  40.90%  31.64%  31.20%  27.33% |
|  | 3 | 1 Apple | 69.47 |
|  | Testing 4 | 1 Person | 94.84%  78.09%  56.05% |
|  | 5 | Cat  Apple  Bed | 92.61%  85.24%  40.05% |

Camera

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture with Patch | Picture Name | Prediction | Confidence |
| A blue car with a picture of fruit on the hood  Description automatically generated | 1 | Car | >60% |
| A close up of a sheep's face  Description automatically generated | 2 | Cat | 70% |
| A plastic bottle filled with fruit  Description automatically generated | 3 | Vase | >70% |
| A person with a fruit on her face  Description automatically generated | Testing 4 | Person | >90% |
| A close up of a mouse  Description automatically generated | 5 | Cat | >70% |

**Instructions**

**Using the Phyton Code**

1. The original picture is coded into the detection system to facilitate identifying and classifying its content. ​
2. Results such as prediction and confidence rate obtained from the detection system are recorded.​
3. When the patch was overlaid onto the original picture, analysis and results obtained from the detection system were recorded using the command line.
4. The detection results obtained before and after overlaying the patch on the original picture are analyzed. The impact of the original image and overlaid patch on the image was recorded and analyzed

**Using the Camera**

1. The original picture is placed into the front detection system using a phone to identify and classify its content. ​
2. Results such as prediction and confidence rate obtained from the detection system are recorded.​
3. When the patch was overlaid onto the original picture, analysis, and results obtained from the detection system were recorded using a phone.
4. The detection results obtained before and after overlaying the patch on the original picture are analyzed. The impact of the original image and overlaid patch on the image was recorded and analyzed

Experiment 1 Notes

**CONFIDENCE OF THE PICTURE BELOW WITH THE PATCH AND WITHOUT THE PATCH**

- We know that YOLOv8 segments the image into pieces.

- We know that when the disturbance from the patch is present in the image, the model gives an inaccurate prediction.

- I believe placing the patch in the model’s blind spot or where the model focuses the most to make the prediction would be effective.

- The adversarial patch works with the code rather than the camera because YOLOv8 segments the image with the patch in the code. When the image is segmented into pieces, the disturbance is contained within the grid of each segment. When this disturbance is present in the grid from the patch, and YOLOv8 analyzes each grid, if there is a significant amount of disturbance (perturbation) present, it will cause the model to be disturbed and affect the prediction, leading to inaccurate results.

- My next belief is that the patch doesn’t work with the camera because YOLOv8 is not segmenting the picture but analyzing it as a whole. It is not performing the same process as when the image is input into the code. Thus, the model will not catch the disturbance from the patch created by the patch in the image. Belief 1 helps to prove this theory. The confidence rate helps to prove this theory.

- Although the prediction is essential, the confidence rate is critical in determining whether adversarial patches work.

- the higher the confidence rate, the more accurate the model’s prediction. However, if the confidence rate is lower, the model can still predict but has low confidence, which the adversarial patch affects.

- Before the adversarial patches were placed, YOLOv8 had predicted what was contained in the image with a high confidence rate. However, when the patch was placed in the image, the model could still expect the same result but with a lower confidence rate.

- An adversarial patch introduces perturbations that interfere with the model’s ability to detect and classify objects accurately. These perturbations can affect the model’s feature recognition process.

- Lowered Confidence: Even if the model predicts the correct objects, the lower confidence indicates the patch has an effect. This decrease in confidence could be critical in scenarios where high certainty is required.

- Consistent Predictions: The fact that the model still predicts the correct objects suggests that the adversarial patch’s perturbations are not sufficient to fool the model entirely but are enough to cause uncertainty.

- My question is whether the adversarial patch is having partial success where the model can still predict, but the prediction comes with low confidence. Does this prove that the adversarial patch is effective? Or do we need the prediction to be entirely inaccurate with a high confidence rate?

-- Even if the model predicts the correct objects, the lower confidence indicates the patch has an effect.

- The results prove that when using code, the adversarial patch introduces significant perturbations, leading to multiple predictions with varying confidence levels. This suggests partial success, as the model’s certainty is impacted and predicts various incorrect objects. While using the camera, the results show that the model predicts single objects with relatively high confidence, indicating less disturbance from the patch.

- The results suggest that the adversarial patch is effective, particularly when the image is processed through code where segmentation is involved. However, the patch is less effective when the image is analyzed as a whole, as seen in the camera results.

- Another question would be if the patch’s image quality and resolution affect the patch’s effectiveness in fooling the model.

- Notice that the model still gives a wrong prediction without the patch.

- The model is trained to analyze different objects using critical features.

- Factors that can influence behaviors are image analysis, environmental factors, image quality, and resolution.

- Learned about image quality and resolution, such as compression/editing, but it still didn’t affect the misclassification from Yolov8

- My question is how Image analysis can affect the effectiveness of an adversarial patch.

- When we know, we YOLOv8 segments the image with the patch in the code. When the image is segmented into pieces, the disturbance is contained within the grid of each segment. When this disturbance is present in the grid, and YOLOv8 analyzes each grid, if there is a significant document prediction, leading to inaccurate results. My next belief is that the patch doesn’t work with the camera because YOLOv8 is not segmenting the picture but analyzing it as a whole. It is not performing the same process as when the image is input into the code. Thus, the model will not catch the disturbance created by the patch in the image.

The final inference I will make is that Image analysis can affect the effectiveness of an adversarial patch. When YOLOv8 segments the image with the patch in the code. When the image is segmented into pieces, the disturbance is contained within the grid of each segment. When this disturbance is present in the grid, and YOLOv8 analyzes each grid, if there is a significant amount of disturbance (perturbation) present, it will cause the model to be disturbed and affect the prediction, leading to inaccurate results.

Image Resolution and Quality Edited

Code

|  |  |  |
| --- | --- | --- |
| Picture Name | Prediction | Confidence |
| 1 | Bus  Truck | 80.31  45.83 |
| 2 | 2 Apple | 30.54  26.33 |
| 3 | Person  Apple | 46.15  40.87 |
| Testing 4 | Bus  Apple | 45.34  33.24 |
| 5 | No Detection | No Detection |

Camera

|  |  |  |
| --- | --- | --- |
| Picture Name | Prediction | Confidence |
| 1 | Car | >60% |
| 2 | No detection | No detection |
| 3 | Bottle | 68 |
| Testing 4 | Person | >80% |
| 5 | Cat/Dog | 50% |

**Instructions**

**Using the Phyton Code**

1. The edited picture is coded into the detection system to facilitate identifying and classifying its content. ​
2. Results such as prediction and confidence rate obtained from the detection system are recorded.​

**Using the Camera**

1. The edited picture is placed into the front detection system using a phone to identify and classify its content. ​
2. Results such as prediction and confidence rate obtained from the detection system are recorded.​

**Experiment 2 Note**

* When the image resolution and quality of the image were altered, the adversarial patch worked with the code and then the camera. When using code, the image undergoes standard preprocessing steps like resizing and normalization, ensuring the adversarial patch retains its designed perturbations.
* In contrast, real-time image capture through a camera introduces additional preprocessing steps such as compression, noise, and lighting variations, potentially diminishing the patch’s effectiveness. With code, you control the image resolution and quality, maintaining consistent properties for the patch to function as intended.

**Final Note**

When using YOLOv8 with code, the model segments the picture into four quadrants. If an adversarial patch is located in one of these quadrants, it introduces a disturbance called perturbation in that quadrant. This segmentation allows YOLOv8 to process each quadrant separately, making the adversarial patch more effective in causing specific disruptions. The controlled preprocessing steps, consistent image quality, and resolution ensure that the perturbations from the patch retain their intended influence within the segmented quadrant.

However, YOLOv8 does not segment the image into quadrants when using a camera for real-time image capture. Instead, the entire image is processed as a whole. This lack of segmentation dilutes the impact of the adversarial patch, as the perturbations are not confined to specific segments. Additionally, variations in image quality and resolution introduced by real-time capture, such as compression, noise, lighting, and focus, further diminish the patch’s effectiveness. These inconsistencies make it challenging for the patch to maintain its designed perturbations, leading to reduced or inconsistent impact on the model’s predictions.

The size and placement of the patch are other details to pay attention to. I realized that the bigger the patch was, the more yolov8 could detect what was in the patch, as the patch was found in one of the grids where object detection could analyze the patch. The patch could be realized where the detection should not be able to detect the patch. The patch should be sized to fit in a grid when the detection segments make a disturbance (perturbation). This disturbance caused Yolov8 to make an inaccurate prediction. When the patch was placed in a fully dense area, that patch was collected in the grid when the yolo segmented the picture.

**Conclusion:**

Overall, significant factors such as image processing of the image, quality, and resolution affect the effectiveness of an adversarial patch overlaid on an image. When using YOLOv8 with code, the model segments the image into quadrants, allowing the adversarial patch’s perturbations to be confined to specific areas and ensuring the patch maintains its designed influence.

**Discussion**

This controlled preprocessing and consistent image characteristics make the patch more effective in disrupting the model’s predictions. In contrast, real-time image capture via camera can alter the patch’s appearance and diminish its effectiveness. In a camera version, the picture is not segmented, and the turbulence (perturbation) that would be produced and stuck in a grid if the patch was located in the grid is spread through the picture and not captured in a quadrant as it would be using the code. The absence of segmentation in camera-captured images spreads the perturbations across the entire image, reducing their impact. Therefore, maintaining high image quality and resolution and using segmentation techniques is crucial for ensuring the effectiveness of adversarial patches. Additionally, the placement of the patch is vital; its effectiveness increases when positioned in a densely populated area within a segmented quadrant. Therefore, maintaining high image quality and resolution, using segmentation techniques, and strategically placing the patch are essential for maximizing the effectiveness of adversarial patches.

***Structure of Yolov8***

Lou H, Duan X, Guo J, Liu H, Gu J, Bi L, Chen H. DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor. Electronics. 2023; 12(10):2323. https://doi.org/10.3390/electronics12102323

A screenshot of a computer

Description automatically generated

**YOLOv8 has been perfect in all aspects, but some problems in identifying small objects in complex scenes still need to be solved. The reasons for the inaccurate detection of small objects are analyzed as follows: (a) When the neural network performs feature extraction, large-size objects mislead small-size objects, and the features extracted at a deep level lack a lot of small-size object information, which leads to the neglect of small objects in the whole learning process, so the detection effect is poor. (b) Compared with standard size, small-size objects are more easily overlapped by other objects and are easily partially blocked by other-size objects, making it difficult to distinguish and locate in an image.** The importance of the patch is small and not big

***Process of Yolov8 when using the Phyton Code and Camera***

**YOLOv8 Segmentation Overview**

* **Backbone Network:** The backbone is a convolutional neural network (CNN) that extracts features from the input image. Standard backbones include architectures like CSPDarknet or variations of ResNet, EfficientNet, etc.
* **Neck:** The neck is used to generate feature pyramids. This helps the model detect objects at different scales. Standard components in the neck include the Feature Pyramid Network (FPN) or Path Aggregation Network (PAN).
* **Head:** In YOLOv8 for segmentation, the head has two main branches:
* **Detection Head:** Predicts bounding boxes, objectness scores, and class probabilities.
* **Segmentation Head:** Predicts pixel-wise masks for each detected object.

**Detailed Segmentation Process in YOLOv8**

**Feature Extraction (Backbone):**

The input image is passed through a series of convolutional layers.

These layers extract hierarchical features at different levels of abstraction.

The output of the backbone is a set of feature maps at various resolutions.

**Feature Pyramid (Neck):**

The neck combines feature maps from different stages of the backbone.

It generates a pyramid of feature maps that help detect objects at multiple scales.

Techniques like FPN or PAN are used to combine these features effectively.

**Detection Head:**

The detection head processes the feature maps to predict bounding boxes.

For each bounding box, it predicts the objectness score and class probabilities.

This involves regression (for bounding box coordinates) and classification (for object categories).

**Segmentation Head:**

The segmentation head predicts masks for each detected object parallel to the detection head.

This typically involves sampling the feature maps to match the resolution of the input image.

Convolutional layers predict a binary mask for each object, indicating the pixels that belong to that object.

**Segmentation Process in The Experiment**

 **Feature Extraction (Backbone)**:

* **Process**:
  + **Convolutional Layers**: The input image, now containing an adversarial patch, is passed through convolutional layers. The patch is designed to perturb the feature extraction process by introducing misleading features.
* **Input Method Differences**:
  + **Python Code**: The adversarial patch is directly applied to a high-quality image. The convolutional layers receive apparent perturbations designed to mislead the model.
  + **Phone to Laptop Camera**: Distortions from capturing the image can either amplify or diminish the effect of the adversarial patch, depending on the nature of the distortion.

 **Feature Pyramid (Neck)**:

* **Process**:
  + **Combining Feature Maps**: The neck combines feature maps, which now include the influence of the adversarial patch. The patch aims to affect the feature combination process, leading to incorrect feature representations.
* **Input Method Differences**:
  + **Python Code**: The perturbations introduced by the patch are combined, potentially leading to more effective misleading features.
  + **Phone to Laptop Camera**: Distortions can interfere with how perturbations are combined, possibly reducing the adversarial patch’s effectiveness.

 **Detection Head**:

* **Process**:
  + **Bounding Box Predictions**: The detection head processes perturbed feature maps. The adversarial patch aims to alter the bounding box predictions, objectness scores, and class probabilities.
  + **Effect**: The patch can cause the model to misidentify, miss, or detect non-existent objects.
* **Input Method Differences**:
  + **Python Code**: The detection head understands the patch’s perturbations, leading to the intended adversarial effects.
  + **Phone to Laptop Camera**: The patch’s effect can be diluted or altered due to distortions, potentially leading to unpredictable detection errors.

 **Segmentation Head**:

* **Process**:
  + **Mask Prediction**: The segmentation head now deals with perturbed feature maps. The adversarial patch attempts to mislead the mask predictions, causing incorrect object masks.
  + **Effect**: This can lead to masks that do not correctly outline objects or that indicate the presence of non-existent objects.
* **Input Method Differences**:
  + **Python Code**: The apparent perturbations from the patch can mislead the segmentation head effectively.
  + **Phone to Laptop Camera**: The patch’s effectiveness in misleading the segmentation head can vary due to image distortions.

Task

1. Understand and find information about the algorithm/structure of how Yolov8 and how Yolov8 segments image
2. Summarize the experiment and the notes I took, including number 1 above.
3. Make my adversarial patch with the help of ChatGPT
4. The following experiment we want to do is add an adversarial patch in a real-time video; remember, most of our experiments use a static image.
5. Look at the different segmentation or image analyses
6. Chart with the results with the code and phyton
7. Choose and Explain What Method I choose
8. Draw a Method Plan
9. Look uo for article based on the method I choose
10. Create Code For This Phase
11. Try out the Segmentation Process with the method you choose/Write Notes
12. Write a Summary of the Segmentation Process Experiment
13. Fix table and graph
14. Learn how to fool yolov8 properly
15. Clean up stuff properly

Without Patch

With Patch

Using Code

The experiment aimed to test the effectiveness of an adversarial patch when it is overlaid on images of different qualities and resolutions, as cameras have different image quality and resolutions. This approach provides insights into the model’s robustness against adversarial attacks under various conditions.

Algorithms

Citations

1. Lou H, Duan X, Guo J, Liu H, Gu J, Bi L, Chen H. DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor. Electronics. 2023; 12(10):2323. https://doi.org/10.3390/electronics12102323
2. M. Safaldin, N. Zaghden and M. Mejdoub, “An Improved YOLOv8 to Detect Moving Objects,” in IEEE Access, vol. 12, pp. 59782-59806, 2024, doi: 10.1109/ACCESS.2024.3393835.keywords: {YOLO;Feature extraction;Real-time systems;Detectors;Computer architecture;Object recognition;Task analysis;Deep learning;Deep learning;localization;object detection;segmentation;YOLO},
3. A. Sahithi, B. S. Teja, C. V. Shastry, C. Venugopal and C. Rajyalakshmi, “Enhancing Object Detection and Tracking from Surveillance Video Camera Using YOLOv8,” 2023 International Conference on Recent Advances in Information Technology for Sustainable Development (ICRAIS), Manipal, India, 2023, pp. 228-233, doi: 10.1109/ICRAIS59684.2023.10367122. keywords: {YOLO;Adaptation models;Training data;Object detection;Computer architecture;Streaming media;Video surveillance;CNN (Convolutional Neural Networks);YOLOv8 algorithm;Deep Learning;Detection;Classification;Segmentation;tracking},
4. Al Jaberi, S. M., Patel, A., & AL-Masri, A. N. (2023). Object tracking and detection techniques under Gann threats: A systemic review. *Applied Soft Computing*, *139*, 110224. https://doi.org/10.1016/j.asoc.2023.110224

Algorithms

**1. Improving Detection of Small-Size Objects in Object Detection Algorithms**

This detection algorithm significantly improves the detection of small-size objects while ensuring the detection quality of normal-size objects. The key components are:

* **MDC Module**: Uses depth separable convolution, Maxpool, and a 3x3 convolution with stride 2 for downsampling. This approach minimizes information loss during downsampling and preserves more contextual information.
* **Improved Feature Fusion**: Enhances shallow and deep information, retaining critical information during feature extraction. This helps in accurate object positioning and reduces misdetections caused by large-size objects.
* **DC Module**: Consists of depth-wise separable convolution and a 3x3 convolution. This module is stacked and fused, forming a new network structure that replaces the C2f module in front of the detection head. It increases the depth and resolution without significant computational cost, capturing more contextual information and improving detection accuracy, especially for overlapping objects.

**2. Enhanced Object Detection**

This section outlines a procedure for improving object detection using advanced techniques:

* **Advanced Data Augmentation**
  + **Input**: Original dataset
  + **Output**: Augmented dataset
  + **Procedure**: Introduce the advanced augmentation function to apply advanced data augmentation techniques.
  + **Return**: Augmented dataset
* **Advanced Backbone Network**
  + **Input**: Input images
  + **Output**: Predicted bounding boxes
  + **Procedure**: Employ EnhancedYOLOv8Architecture function to define Enhanced YOLO-V8. Use a sophisticated network like ResNet to capture intricate details.
  + **Return**: Predicted bounding boxes
* **Fine-Tuning**
  + **Input**: pre-trained model, Augmented dataset
  + **Output**: Fine-tuned model
  + **Procedure**: Perform fine-tuning with neural\_network.fine\_tune on the augmented data. Adapt the model to the specific dataset and task.
  + **Return**: Fine-tuned model
* **Advanced Post-Processing**
  + **Input**: Predicted bounding boxes
  + **Output**: Refined bounding boxes
  + **Procedure**: Apply advanced\_postprocess techniques like Soft-NMS. Soft-NMS improves object detection accuracy.
  + **Return**: Refined bounding boxes

**3. Enhanced YOLO-V8 Object Detection Function**

This function describes the process of using Enhanced YOLOv8 for object detection:

The function EnhancedYOLOv8(image) begins by preprocessing the image and applying advanced augmentation techniques. It then initializes the Enhanced YOLOv8 architecture, optionally loading pre-trained weights if available. The model is fine-tuned on the augmented image. The neural network performs a forward pass to produce detection results, which are refined using advanced post-processing techniques.

**4. Enhanced YOLOv8Architecture Function**

This function defines the architecture for Enhanced YOLOv8. EnhancedYOLOv8Architecture ensures the neural network architecture by calling another function, DefineEnhancedArchitecture, which specifies the layers and structure of the Enhanced YOLOv8 network. The defined architecture is then returned for use in the detection process.

**5. Enhanced Architecture Function**

This function outlines the steps to build the enhanced network layers for the YOLOv8 architecture. The function DefineEnhancedArchitecture constructs the layers and configurations needed for the Enhanced YOLOv8 network. The built architecture, designed to capture detailed features, is then returned.

**6. Advanced Post-Processing Function**

This function describes the post-processing of raw detection results. The function advanced post process (detection\_results) applies advanced post-processing techniques (e.g., Soft-NMS) to the raw detection results. The refined detection results are then returned, enhancing the overall performance of the object detection model.

**7. Recommendation by Saeed Matar Al Jaberi et al.**

Saeed Matar Al Jaberi and colleagues recommend using Generative Adversarial Neural Networks (GANN). In GANN, a generator creates realistic data, while a discriminator distinguishes between accurate and generated data. This adversarial process improves the quality of generated samples, providing high-quality synthetic data for training. This approach enhances model robustness and helps in scenarios where accurate data is limited or imbalanced

***Explanation/ Description of Each Algorithm***

### 1. Improving Detection of Small-Size Objects in Object Detection Algorithms

This method enhances small-size object detection while maintaining accuracy for normal-size objects. The MDC module uses advanced convolution techniques to keep more information during downsampling. Improved feature fusion combines shallow and deep features to retain critical information better, ensuring accurate object positioning and fewer misdetections. The DC module, which includes stacked depth-wise separable convolutions, replaces the C2f module, increasing depth and resolution without adding much computational cost. This helps capture more contextual information and improves detection accuracy, especially for overlapping objects.

### 2. Enhanced Object Detection Workflow

This workflow uses advanced techniques to improve object detection. First, advanced data augmentation techniques enhance the original dataset, making the model more robust. Next, a sophisticated backbone network like ResNet captures detailed features. The model is then fine-tuned using the augmented dataset to adapt to specific tasks. Finally, advanced post-processing techniques like Soft-NMS refine predicted bounding boxes, improving accuracy by reducing duplicate detections.

### 3. Enhanced YOLO-V8 Object Detection Function

This function outlines how to use the Enhanced YOLOv8 for object detection. It starts by applying advanced augmentation to the input image. The Enhanced YOLOv8 architecture is then loaded, possibly with pre-trained weights. If available, the model is fine-tuned on the augmented image. The neural network performs a forward pass to produce detection results, which are refined using advanced post-processing techniques.

### 4. EnhancedYOLOv8Architecture Function

This function defines the Enhanced YOLOv8 network architecture. It is called another function that specifies the layers and structure of the network. Optimized for better performance, the defined architecture is then returned for use in the detection process.

### 5. Enhanced Architecture Function

This function builds the specific layers for the Enhanced YOLOv8 network. It constructs the necessary components, such as convolutional and pooling layers, to create a robust network structure. The completed architecture, designed to capture detailed features, is then returned.

### 6. Advanced Post-Processing Function

This function refines raw detection results from the neural network. Advanced techniques like Soft-NMS are applied to improve accuracy by reducing duplicate detections. The refined detection results are then returned, enhancing the overall performance of the object detection model.

### 7. Recommendation by Saeed Matar Al Jaberi et al.

Saeed Matar Al Jaberi and colleagues recommend using Generative Adversarial Neural Networks (GANN). In GANN, a generator creates realistic data, while a discriminator distinguishes between accurate and generated data. This adversarial process improves the quality of generated samples, providing high-quality synthetic data for training. This approach enhances model robustness and helps in scenarios where accurate data is limited or imbalanced.

Simpler Terms

Improving Detection of Small-Size Objects in Object Detection Algorithms

This method helps computers find small objects in pictures more effectively. It uses a special tool called the MDC Module that makes images smaller while keeping important details intact. Imagine shrinking a picture but still being able to see the important parts clearly. The method also combines different layers of information from the picture to get a better understanding, kind of like layering transparent sheets with different details to see the full image. Another part, the DC Module, builds a stronger network by stacking and merging smaller tools, helping the computer to see overlapping objects better without needing much extra computing power.

Enhanced Object Detection

This method improves how well computers can find objects in pictures by using several advanced techniques. First, it creates new versions of the training images by making changes like flipping, rotating, or changing colors. This is like giving the computer more varied practice images. Then, it uses a smart core part of the model, like ResNet, which can capture fine details in the images. Next, it fine-tunes an already trained model with the new practice images to make it even better for specific tasks. Finally, it uses advanced techniques to refine the results, making the detection boxes around objects more accurate.

Enhanced YOLO-V8 Object Detection Function

This function is a step-by-step guide for using an improved version of the YOLOv8 model to find objects in images. It starts by preparing the image with special techniques and then sets up the enhanced YOLOv8 model. If available, it loads previously trained settings. The model is then fine-tuned with new images to improve its performance. After the model processes the image to find objects, the results are refined using advanced techniques to make them more accurate.

Enhanced YOLOv8Architecture Function

This function defines how to build the enhanced YOLOv8 model. It calls another function that specifies the details of the network, such as what layers and structures are needed. Think of it as setting up a blueprint for constructing a detailed and effective model that can find objects accurately in images.

Enhanced Architecture Function

This function is all about building the network layers for the YOLOv8 model. It lays out the steps and configurations needed to capture detailed features from images. The resulting architecture is like a well-designed machine that can efficiently and accurately detect objects in pictures.

Advanced Post-Processing Function

This function takes the initial results from the object detection model and improves them. It uses advanced techniques like Soft-NMS to refine the detection results, making the boxes around the detected objects more precise. This final step ensures that the overall performance of the object detection model is as accurate and reliable as possible.

Recommendation by Saeed Matar Al Jaberi et al.

Saeed Matar Al Jaberi and his colleagues suggest using a technique called Generative Adversarial Neural Networks (GANN). In GANN, there are two parts: one part creates realistic-looking data, and the other part tries to tell if the data is real or fake. This competition helps the first part get really good at creating high-quality fake data. This high-quality fake data can be used to train the model, making it more robust and effective, especially when there isn't enough real data available.

**Method of Choice to Use the Generative Adversarial Network (Option 7)**

A generative Adversarial Network is a machine learning model in which two neural networks compete using deep learning methods to become accurate in their predictions. There are many different types of GAN networks. Some of these networks are Vanilla GAN, Conditional GAN, and other types of networks. The network that will be used in the segments section will be the Conditional GAN. Conditional generative adversarial network, or can for short, is a type of GAN that involves the conditional generation of images by a generator model. It is a type of GAN model where a condition is implemented to get the output.

***Pros of CGAN***

1. **Controlled Generation**: The primary advantage of cGANs is the ability to steer the generative process by providing additional information. This enables users to generate specific instances or data variations based on the conditions.
2. **Contextual Realism**: Conditional information enhances the realism of generated data by incorporating context. This makes the generated samples more coherent and aligned with the specified conditions.
3. **Multi-Modal Generation**: can facilitate the generation of multiple data modes within the same distribution. This is particularly useful in tasks where diverse outputs are desired.

***Cons of CGAN***

1. **Data Availability**: cGANs use annotated or labeled data for conditioning. Obtaining such data can be challenging and time-consuming, limiting the applicability of cGANs in specific domains.
2. **Mode Collapse**: Like traditional GANs, cGANs can suffer from mode collapse, where the generator produces a limited variety of samples regardless of the input conditions.

**Segmentation Phase Using Conditional GAN for Adversarial Patch Mitigation**

**Objective:**

The segmentation phase aims to divide the image into smaller segments and analyze each using a Conditional GAN (cGAN) to detect anomalies, indicating potential adversarial patches.

**Process:**

1. **Image Division**:
   * **Technique**: The image is divided into smaller segments using sliding windows, grid-based, or region-based segmentation.
   * **Segment Size**: The size of each segment is determined based on the image resolution and required level of detail. Smaller segments provide finer analysis but increase the computational load.
2. **Conditional GAN Analysis**:
   * **Generator and Discriminator**:
     + **Generator**: The generator attempts to produce realistic segments based on a given condition.
     + **Discriminator**: The discriminator evaluates whether each segment (real or generated) matches the expected pattern under the given condition.
   * **Conditioning**:
     + **Condition**: Specific features or labels (such as object type, expected pattern, or normal behavior) are used as conditions.
     + **Application**: Each segment is analyzed by the cGAN, where the generator produces segments conditioned on expected features and the discriminator checks for deviations.
3. **Detection of Deviations**:
   * **Training**:
     + **Normal Data**: The cGAN is trained on a dataset of standard images without adversarial patches, learning the distribution of regular segments.
   * **Analysis**:
     + **Deviation Detection**: Each segment is compared to the expected pattern during analysis. Significant deviations are flagged as potential anomalies, indicating adversarial patches.
4. **Comparison Between Edited and Real Images**:
   * **Comparison Process**:
     + **Edited Image**: Each segment from the edited image (potentially containing adversarial patches) is compared with segments from a corresponding actual image.
     + **Real Image**: The actual image is the reference, containing the average, expected patterns without adversarial modifications.
   * **Difference Calculation**:
     + **Metrics**: Use similarity metrics like Structural Similarity Index (SSIM) or Mean Squared Error (MSE) to quantify differences between corresponding segments.
     + **Thresholds**: Establish thresholds for these metrics to determine significant deviations, indicating potential adversarial patches.

**Detailed Steps:**

1. **Image Segmentation**:
   * **Step 1**: Divide the image into smaller segments.
   * **Example**: For a 1024x1024 image, you might divide it into 64x64 segments, resulting in 256 segments.
2. **Applying Conditional GAN**:
   * **Step 2**: For each segment, apply the GC analysis.
   * **Conditioning Information**: Use prior knowledge or features as conditions, such as object class or expected texture.
   * **Generator**: Generate segments based on the condition.
   * **Discriminator**: Evaluate real and generated segments to identify deviations.
3. **Training the can**:
   * **Step 3**: Train the cGAN on typical images to learn the distribution of regular segments.
   * **Normal Data**: Ensure the training dataset is diverse and representative of normal conditions.
4. **Analyzing Segments**:
   * **Step 4**: Analyze each segment with the trained cGAN.
   * **Deviation Detection**: Identify segments that deviate significantly from the expected pattern. These deviations indicate potential adversarial patches.
5. **Flagging Anomalies**:
   * **Step 5**: Flag segments with detected deviations for further processing in the isolation and blocking phases.
6. **Comparing Segments**:
   * **Step 6**: Compare each segment from the edited image with the corresponding segment from the actual image.
   * **Step 7**: Calculate similarity metrics (e.g., SSIM, MSE) for each segment pair.
   * **Step 8**: Flag segments with similarity metrics below the established threshold as potential adversarial patches.

**Example Workflow:**

1. **Input Image**: A 1024x1024 image is divided into 64x64 segments.
2. **Segment Analysis**: Each segment is analyzed using the GC.
   * **Condition**: The cGAN is conditioned on the expected class (e.g., a clear sky, tree texture).
   * **Generator**: Generates segments resembling the expected condition.
   * **Discriminator**: Evaluates if segments match the condition.
3. **Anomaly Detection**: Segments significantly deviate from the expected pattern are flagged.
   * **Example**: A segment with an unexpected pattern in a clear sky image might indicate an adversarial patch.
4. **Comparing Segments**:
   * **Edited Image**: Analyze segments from an edited image suspected to contain adversarial patches.
   * **Real Image**: Compare each edited segment with the corresponding segment from an actual image.
   * **Metrics Calculation**: Calculate SSIM or MSE for each segment pair.
   * **Flagging**: Segments with significant differences are flagged as potential adversarial patches.

Redo of the Experiment (Detection was able to detect the patch)

Python Code

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture with Patch | Picture Name | Prediction | Confidence |
|  | Blue Car | Car  Truck | 72.66  54.32 |
|  | Sheep | Sheep  Bear | 91.27%  83.32% |
|  | Bottle | Person | 76.53% |
|  | Human | Human | 93.48% |
|  | Dog | Dog | 95.79% |

Camera

|  |  |  |  |
| --- | --- | --- | --- |
| Original Picture with Patch | Picture Name | Prediction | Confidence |
|  | Blue Car | Car | 83% |
|  | Sheep | No Detection | No Detection |
|  | Bottle | Bottle | 70% |
|  | Human | Human | 58% |
|  | Dog | Cat | 60% |

Without Patch

A graph of a number of people

Description automatically generated with medium confidence

With Patch

A graph of a number of people

Description automatically generated

ReNote

* Notes from the previous experiment still stands
* The image that returned with the two predictions showed that the model was effective. **Blue Car Image:** The patch caused the model to predict the blue car as a truck with a confidence of 72.66% and as a car with a confidence of 54.32%. This shows that the patch confused the model about the object’s identity. **Sheep Image:** The patch caused the model to predict the sheep as a bear with a high confidence of 91.27%, while the confidence for the correct prediction (sheep) was 83.32%. This indicates that the patch significantly distorted the model’s object perception.
* Static segmentation in the code might allow the patch to disrupt the model's feature extraction more consistently, while dynamic segmentation in the camera feed can mitigate this effect due to environmental variability and continuous re-segmentation
* Static vs Dynamic:

**Segmentation Approach:**

1. **Real-Time Camera Feeds:**
   * **Continuous Processing:** YOLOv8 continuously captures frames from the camera feed and processes each frame independently in real-time.
   * **Dynamic Input:** Frames from the camera are dynamic and continuously changing, requiring the model to adapt to real-world conditions such as varying lighting, motion, and object positions.
   * **Immediate Feedback:** Results are displayed or acted upon immediately after processing each frame, allowing for real-time monitoring and response.
2. **Python Code with Static Images:**
   * **Batch Processing:** YOLOv8 processes static images one at a time from disk or memory.
   * **Controlled Input:** Static images provide controlled and consistent conditions for testing and evaluation, without the variability introduced by real-time camera feeds.
   * **Delayed Feedback:** Results are typically stored and analyzed after processing each image, allowing for detailed examination and comparison.

**Processing Manner:**

1. **Real-Time Camera Feeds:**
   * **Continuous Loop:** The process involves a continuous loop that captures, preprocesses, segments, and analyzes frames in real-time.
   * **Immediate Display:** Detection results are immediately displayed on the screen, providing instant feedback on detected objects and their positions in the current frame.
2. **Python Code with Static Images:**
   * **Sequential Processing:** Images are processed sequentially, with each image undergoing preprocessing, segmentation, and analysis in a batch manner.
   * **Delayed Display:** Detection results are typically aggregated and displayed or analyzed after all images in the batch have been processed, depending on the implementation.

**Key Differences Summarized:**

* **Real-Time Camera Feeds:** Involve continuous, real-time processing of dynamically changing frames, adapting to immediate environmental changes. Results are displayed instantly, offering live feedback on object detection.
* **Python Code with Static Images:** Involves batch processing of static images, providing controlled and repeatable conditions for testing. Results are typically reviewed after processing, enabling detailed analysis and comparison.