Artificial intelligence is the theory and development of computer systems capable of performing activities ordinarily carried out by humans, such as visual perception, speech recognition, decision-making, and language translation. Under the topic of adversarial patches, we have researched, continued to learn, and experimented with how adversarial patches affect the normal function of artificial intelligence models like YOLOv8. This exploration begins with a foundational understanding of adversarial patches and then progresses into experimental analysis, following articles such as *Adversarial Patch* by Tom B. Brown et al. Initially, we learned and understood what adversarial patches are. These patches are carefully designed digital inputs, resembling 3D image-like patches, that can be added to a scene. When a patch is added (for example, to a tabletop scene containing a banana and a notebook), it is meant to deceive or manipulate the model. This manipulation attack is termed an adversarial attack. The following paragraphs delve deeper into how adversarial patches can affect models.

After gaining a foundational understanding of adversarial patches, we explored and experimented with various research papers to enhance our comprehension of their impact on models. Specifically, we analyzed the *Adversarial Patch* research article by Tom B. Brown et al. and then conducted experiments that mirrored the methodologies outlined. The article illustrated the attack by demonstrating that when a carefully chosen input, such as a 3D image-like patch resembling a toaster, is silently placed into a scene (like a tabletop with a banana and notebook alongside a cow and dog), the model’s decision-making is manipulated, resulting in misclassification and inaccurate predictions. When the patch is introduced into the scene, it generates a disturbance known as perturbation. As the model analyzes the scene for classification and prediction, this disturbance spreads across the entire composition, manipulating the model and leading to misclassifications and inaccuracies. For example, when the model processed an image of a tabletop with a banana and a notebook, it confidently classified the scene as containing a banana. However, upon introducing the adversarial patch, the model was misled and incorrectly predicted the presence of a toaster. A similar experiment with a scene containing a cow and a dog yielded the same results, where the patch inaccurately altered the model’s predictions.

The results from the experiment shown in Table 1 and Table 2 illustrate the impact of adversarial patches on object detection for a cow and a dog. In Table 1, without the patch, the AI model accurately identifies the cow and dog (both >70%), showing relatively quick preprocessing and postprocessing speeds. However, Table 2, which includes the adversarial patch, shows significant misclassification. The model incorrectly identifies a dog as a cow with a reduced confidence of 62% and a person instead of a dog with 75% confidence. Additionally, inference and postprocessing speeds are slower, indicating a more challenging detection process when the adversarial patch is applied. These examples illustrate how the classifier struggled with conflicting signals, resulting in incorrect classifications. In one instance, a scene containing only a banana was misclassified as a toaster after introducing the adversarial patch despite no toaster being present. Similarly, another experiment involving a picture of a cow and a dog showed that the introduction of the patch caused the model to misclassify the scene. Although the model should have provided accurate predictions based on the actual objects in both cases, the adversarial patch led to incorrect classifications. The significance of this study lies in its demonstration of the detrimental impact of overlaid patches on model predictions. By showcasing how inherent and overlaid perturbations contribute to inaccuracies in detection systems, our findings underscore the critical importance of robust defense mechanisms against adversarial attacks. Additionally, we recognized that the size and position of the overlaid patch on the original image are key factors contributing to the perturbations that manipulate object detection systems.

**A table with numbers and text

Description automatically generated with medium confidence**

**Table1: The results when no adversarial patch was used in the scene.**

**A blue and white graph with text

Description automatically generated with medium confidence**

**Table 2: The results when the adversarial patch was used in the scene**

Having grasped the basics of adversarial patches, we sought a deeper understanding of how they impact AI models. This inquiry involved examining the specific effects of adversarial patches on the YOLOv8 object detection model by measuring changes in prediction accuracy and confidence levels and reading articles such as *An Improved YOLOv8 to Detect Moving Objects* by M. Safaldin et al. To achieve this, we placed patches in various locations within scenes and compared the results from images processed by the AI model with those displayed on the phone and shown to the detection camera. The process for both methods began with the detection system processing the image to identify and classify its content, recording predictions and confidence rates. After overlaying an adversarial patch, we captured the system results via the command line and mobile camera, allowing for a comparison of the impacts before and after the patch introduction. The edited image was then reintroduced for further classification using Python and the camera, with results documented to analyze system performance.

The graphs comparing object detection confidence levels between the phone camera and AI model, with and without adversarial patches, highlight key differences. Without the patch, the AI model consistently outperformed the phone camera, especially for objects like sheep (95.98%) and dogs (91.84%). However, the phone camera maintained a solid accuracy of 70% for car detection versus the AI model’s 66.06%. With the patch, the AI model’s performance declined significantly, resulting in misclassifications and even 0% detection for specific objects, while the phone camera remained more stable, mainly for cars and humans. This experiment underscores the AI model’s increased vulnerability to adversarial attacks. Moreover, this analysis shows that YOLOv8 divides images into four quadrants during execution, allowing adversarial patches to cause disruptions in specific segments. Each quadrant is processed separately, ensuring that perturbations retain their intended influence. In contrast, during real-time image capture, the model processes the entire image without segmentation, which dilutes the impact of the adversarial patch as perturbations are dispersed across the whole picture.

A group of green and yellow bars

Description automatically generated

**Graph 1: The confidence rate of scenes with/without the patch with different scenes.**

The YOLOv8 model processes images in stages, each affected by the adversarial patch. In the feature extraction stage (Backbone), convolutional layers receive perturbed input, with the patch introducing misleading features. Differences arise depending on the input method—Python code directly applies the patch to a high-quality image, making the perturbations more evident. In contrast, capturing the image via phone to laptop camera can distort the effect, either amplifying or diminishing the patch’s impact. Moving to the feature pyramid stage (Neck), feature maps are combined, and the adversarial patch influences this process, leading to incorrect feature representations. Python-applied patches tend to generate more effective misleading features, while camera distortions may interfere with perturbations, reducing their potency. The detection head is similarly affected, as perturbed feature maps alter bounding box predictions and class probabilities, often causing misidentifications or false detections. Python-applied patches maintain more substantial adversarial effects, but camera distortions can dilute these effects, leading to unpredictable errors. Finally, the patch misguides mask predictions in the segmentation head, resulting in incorrect object masks or the detection of nonexistent objects. The patch proves more effective when directly applied via Python, whereas camera distortions can affect its efficacy in misleading segmentation.

In conclusion, this analysis highlights the detrimental impact of overlaid patches on models. By demonstrating how inherent and overlaid perturbations contribute to inaccuracies in detection systems, our findings underscore the critical importance of robust defense mechanisms against adversarial attacks. Future work aims to test these patches on different models or develop a Conditional Generative Neural Architecture (cGAN) defense for real-time detection and neutralization of adversarial patches.

**References**

1. Brown, T. B., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (December 27, 2017). *Adversarial patch*. arXiv. <https://arxiv.org/abs/1712.09665>
2. Hofman, O., Giloni, A., Hayun, Y., Morikawa, I., Shimizu, T., Elovici, Y., & Shabtai, A. (2023, June 14). *X-Detect: Explainable adversarial patch detection for object detectors in retail*. arXiv. <https://arxiv.org/abs/2306.08422>
3. Lou, H., Duan, X., Guo, J., Liu, H., Gu, J., Bi, L., & Chen, H. (2023). DC-YOLOV8: Small-size object detection algorithm based on the camera sensor. *Electronics, 12*(10), 2323. <https://doi.org/10.3390/electronics12102323>
4. *An improved YOLOV8 to detect moving objects*. (n.d.). IEEE Xplore. <https://ieeexplore.ieee.org/document/10508365>