Custom Pedestrian Dataset Fine-Tuning: Leveraging DINO for Improved Detection

Abstract: Fine-tuning of DINO, a transformer-based object detection model, for improved pedestrian detection. Leveraging a pre-trained DINO-4scale model with an R50 backbone, the model was fine-tuned using a custom dataset of 200 annotated images over 12 epochs. The objective was to enhance detection performance in challenging pedestrian scenes. Results demonstrate the effectiveness of fine-tuning DINO with minimal data, offering a promising approach for pedestrian detection in real-world applications. The pre-trained checkpoint model significantly accelerated convergence and detection accuracy.

Summary of Task for DINO Object Detection Model Training:

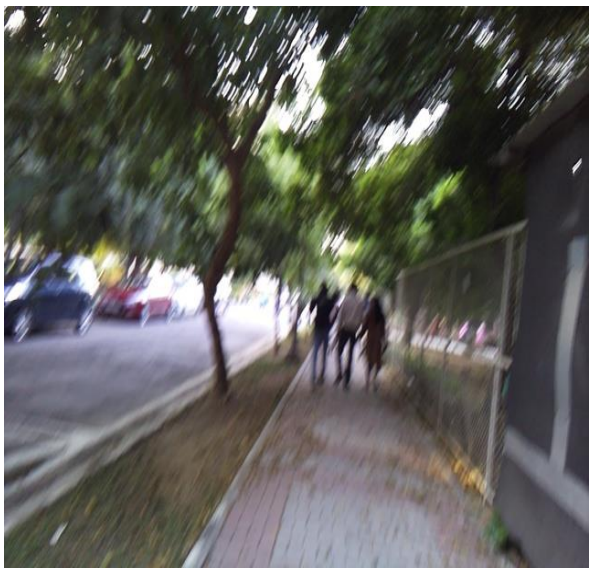
Train the DINO object detection model on a pedestrian dataset of 200 COCO-annotated images from IIT Delhi, splitting it into 160 training and 40 validation images. Clone the DINO repository, set up the environment, and utilize a GPU (via Google Colab or Kaggle if needed). Download the pre-trained DINO-4scale model (ResNet-50 backbone), evaluate it on the validation set, report Average Precision (AP) values, visualize detection results, and analyze errors. Fine-tune the model on the training set, visualize attention maps, and document the experiments in a GitHub repository, including a README.md with instructions and a PDF report with results and loss graphs.

Early implementation :

In my earlier implementation, I fine-tuned the DINO 4scale model, specifically the 12-epoch checkpoint, on a pedestrian dataset that consisted of images annotated in COCO format with a single class, "person." This fine-tuning process involved adapting the pre-trained model to focus exclusively on recognizing and detecting pedestrians, enabling it to learn the specific features associated with this class. By training the model on the pedestrian dataset over several epochs, I enhanced its ability to accurately identify instances of the "person" class in the provided images, thereby optimizing its performance for this specific object detection task.(https://github.com/ShashankAQ/Fine-Tuning-DINO-DETR-on-custom-Dataset)

Fine tuning 4-scale 12 epoch checkpoint model:

In this model, the fine-tuning process yielded promising results, with the model performing exceptionally well in detecting the "person" class in simpler scenes. However, it struggled with a few instances involving complex scenes, where multiple objects, occlusions, or intricate backgrounds made it challenging to accurately identify pedestrians.



Original image

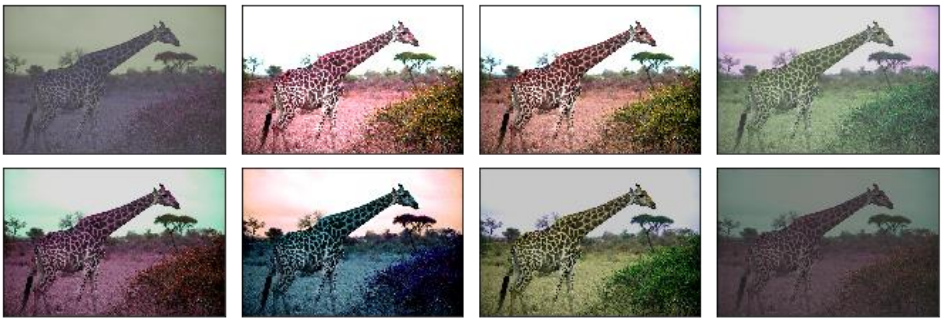
Predicted output

The model failed to detect the pedestrian in the blurry image primarily due to the lack of sharp visual features and details that are crucial for accurate object recognition. Blurriness can obscure important characteristics such as shape, color, and texture, which the model relies on to identify and localize objects.

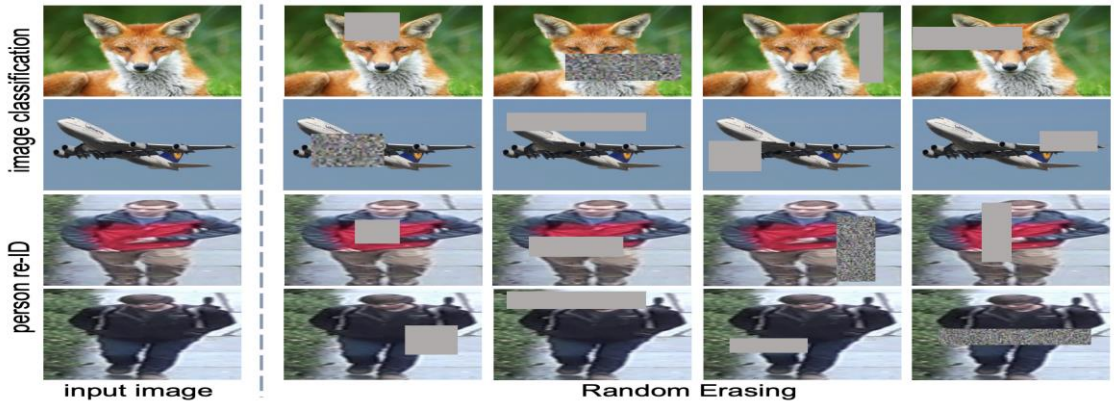
To improve the model's accuracy while working with a constrained dataset of only 160 training images, I implemented additional augmentation techniques to enhance the complexity and variability of the training data.

Gaussian Blur: This technique intentionally increases blurriness in some images, helping the model learn to recognize pedestrians in scenarios where the input images may be unclear or out of focus. By exposing the model to blurred images during training, it becomes more adept at handling similar challenges during inference.

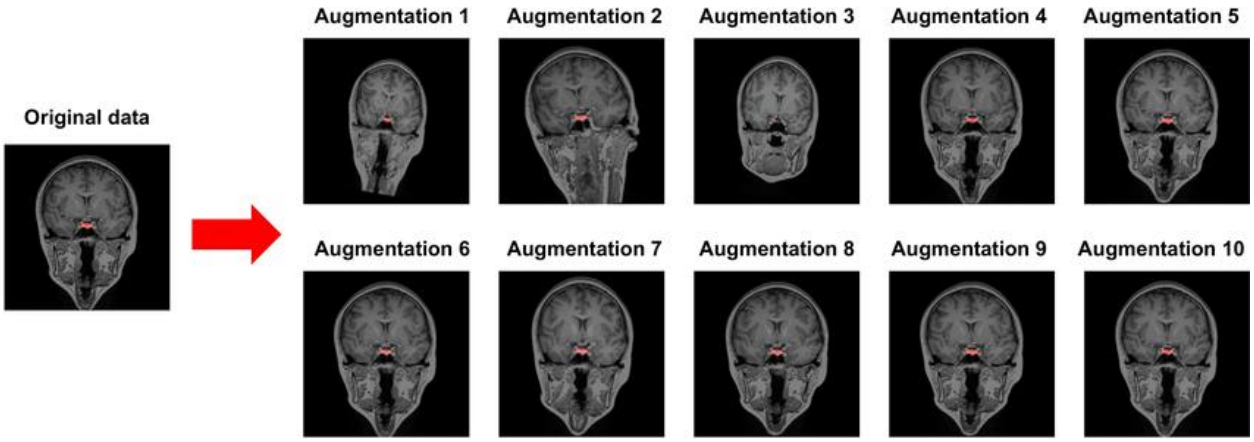
Color Jitter: This augmentation modifies the brightness, contrast, saturation, and hue of images, effectively simulating varying lighting conditions. This helps the model generalize better in diverse environments, ensuring robust performance even when the lighting is less than ideal.



Random Erasing: This technique randomly removes portions of the image, promoting the model's ability to identify and focus on remaining features. It is particularly useful for training in situations where pedestrians may be partially occluded by other objects, teaching the model to remain resilient against such disruptions.



Random Affine Transformations: By applying random transformations such as rotations, translations, and scaling, this augmentation increases the diversity of the training images. It enables the model to learn from different perspectives and spatial arrangements of pedestrians, enhancing its ability to detect them in various scenarios.

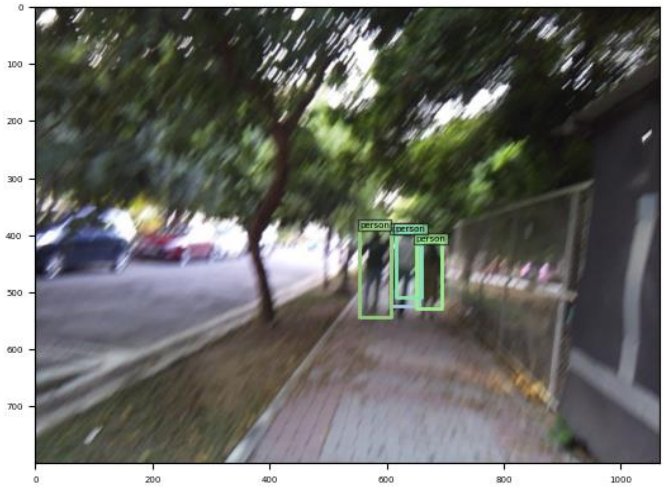



```python
class ColorJitter(object):
    def __init__(self, brightness=0, contrast=0, saturation=0, hue=0):
        self.jitter = T.ColorJitter(brightness=brightness, contrast=contrast, saturation=saturation, hue=hue)

    def __call__(self, img, target):
        return self.jitter(img), target

class GaussianBlur(object):
    def __init__(self, kernel_size, sigma=(0.1, 2.0)):
        self.blur = T.GaussianBlur(kernel_size, sigma)

    def __call__(self, img, target):
        return self.blur(img), target

class RandomErasing(object):
    def __init__(self, *args, **kwargs):
        self.eraser = T.RandomErasing(*args, **kwargs)

    def __call__(self, img, target):
        return self.eraser(img), target

class RandomAffine(object):
    def __init__(self, degrees, translate=None, scale=None, shear=None):
        self.affine = T.RandomAffine(degrees, translate=translate, scale=scale, shear=shear)

    def __call__(self, img, target):
        return self.affine(img), target
```

Code implementation

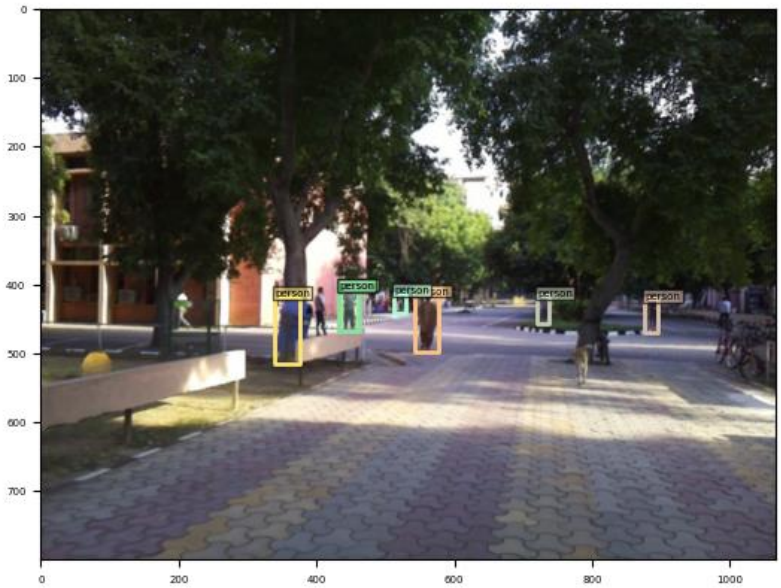
Before Augmentation

After augmentation:

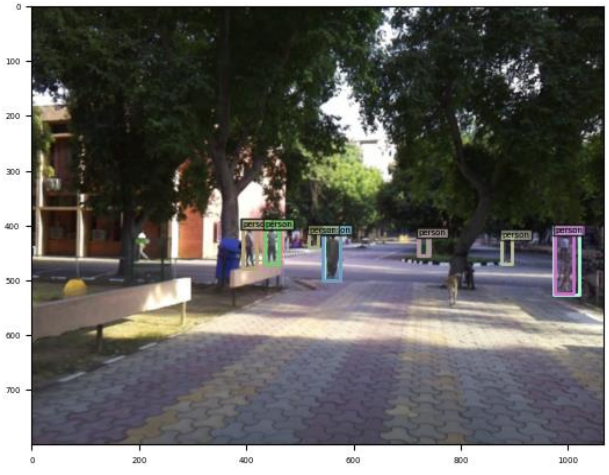


After implementing the augmentation techniques, the model exhibited significant improvement in performance primarily because it was trained on more complex scenes, which enriched the variability of the training data despite the limited number of images. By simulating various real-world conditions—such as different lighting scenarios, blurriness, occlusions, and spatial transformations—the model became better equipped to generalize from the training dataset.

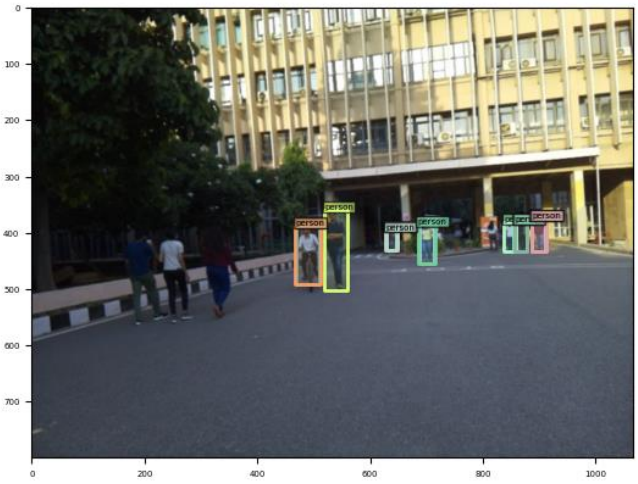
These augmented images helped the model to understand the diverse appearances and configurations of pedestrians, allowing it to learn robust features necessary for accurate detection. Consequently, the model demonstrated enhanced resilience and adaptability when faced with complex or challenging real-world situations during evaluation, ultimately leading to improved accuracy and reliability even with the constraint of only 160 training images.



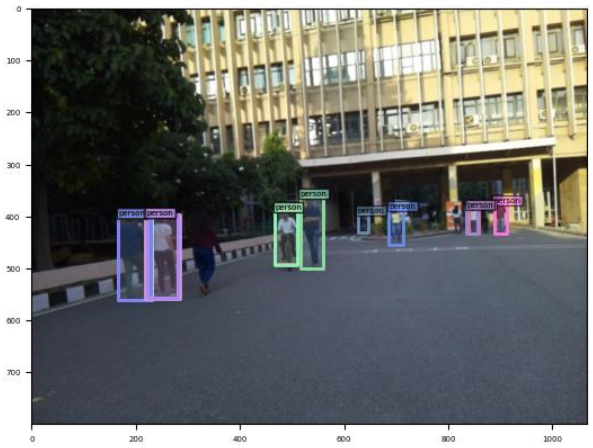
Before Augmentation



After augmentation



Before augmentation



After Augmentation

Further Implementation: Ensemble Prediction for Improved Detection:

As seen in the previously uploaded images, even after applying extra augmentation techniques during the 12th epoch checkpoint, some bounding boxes around pedestrians were still missing. To address this issue, I implemented ensemble prediction, which involved using six different models. These models consisted of fine-tuned DINO 4-scale models with a ResNet-50 backbone, utilizing various checkpoints from 12, 24, and 36 epochs—both with and without the extra augmentation techniques.

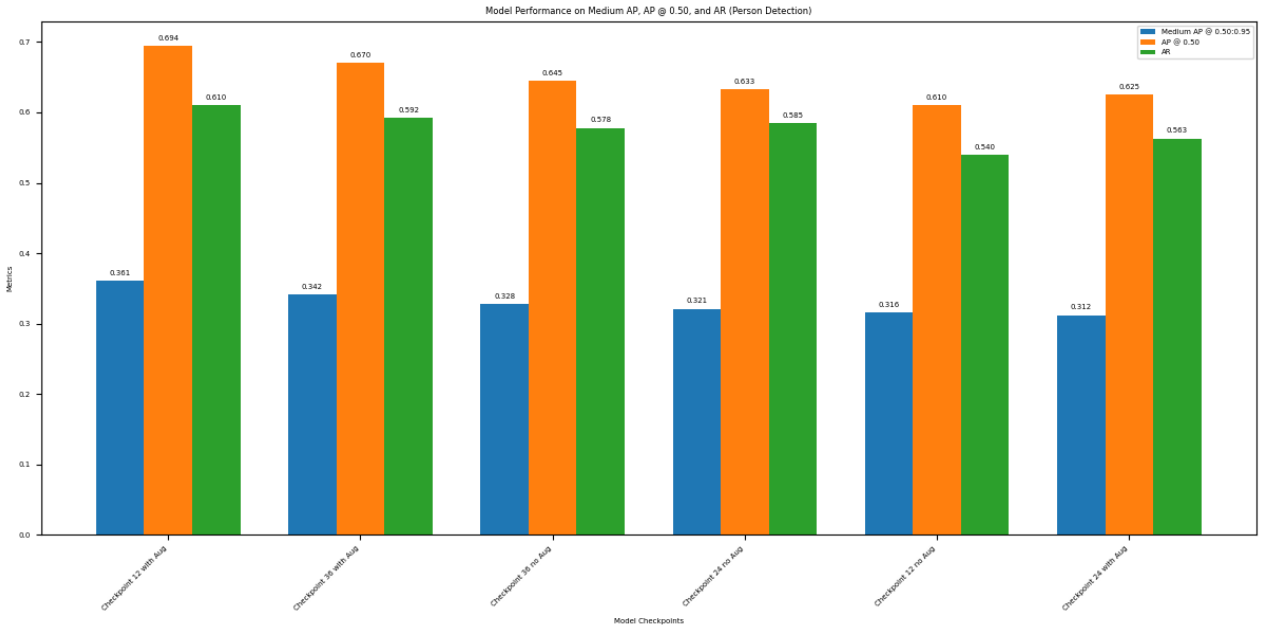
The reason for using multiple models in this ensemble approach is that each model captures different aspects of the data and learns unique features based on its training process. By combining the predictions of these models, the ensemble can leverage their collective strengths, leading to more accurate and robust detections. This method helps to reduce the likelihood of missing detections, as one model might succeed where another fails. Overall, using an ensemble of models enhances the reliability of pedestrian detection in images, improving overall performance on the task.

Evaluation of the checkpoints on the validation set to find the best models:

To evaluate the performance of the ensemble prediction, I fine-tuned the models and performed evaluations on each of the six checkpoints from the validation set. Among these, the best-performing models were the 12-epoch checkpoint with augmentation, the 36-epoch checkpoint with augmentation, and the 36-epoch checkpoint without augmentation.

In contrast, the 24-epoch checkpoints, both with and without augmentation, exhibited issues with hallucinations, where the model incorrectly drew multiple bounding boxes around the same person. This indicates that while some models excelled in accurately detecting pedestrians, others struggled, highlighting the importance of carefully selecting and combining models in the ensemble to achieve optimal detection performance. The evaluation results demonstrate that the combination of different training epochs and augmentation strategies can significantly influence the model's effectiveness in object detection tasks.

Model	Loss	AP @ 0.50:0.95 (Overall)	AP @ 0.50 (Overall)	AR (Overall)
Checkpoint 12 with Aug	6.22	0.316	0.662	0.580
Checkpoint 36 with Aug	6.69	0.292	0.615	0.568
Checkpoint 36 no Aug	6.46	0.290	0.593	0.561
Checkpoint 24 no Aug	6.42	0.283	0.591	0.586
Checkpoint 12 no Aug	6.86	0.276	0.543	0.520
Checkpoint 24 with Aug	6.93	0.279	0.570	0.543

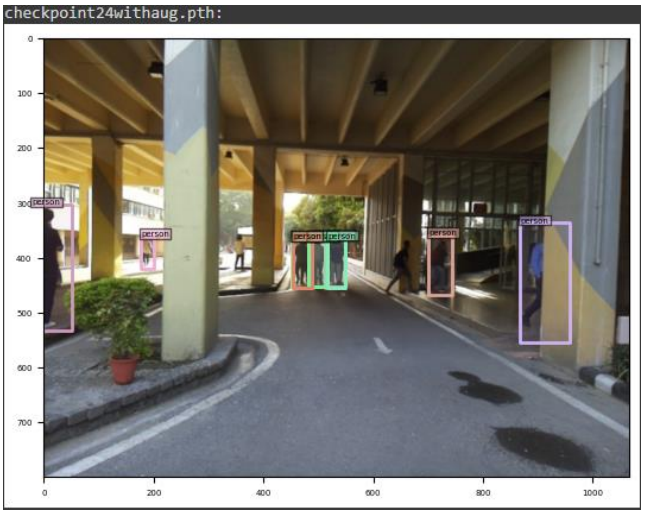


Examples:

checkpoint.pth:

checkpoint24.pth:

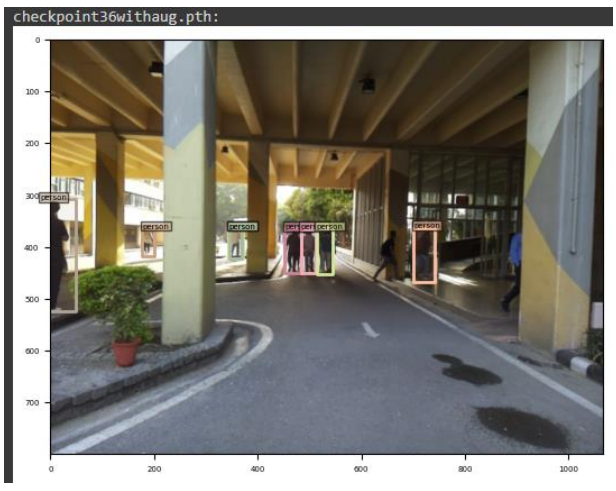
checkpoint24withaug.pth:



checkpoint12withaug.pth:

checkpoint36.pth:

checkpoint36withaug.pth:



The images above illustrate a simple scene with minimal obstructions, where the 36-epoch checkpoint model outperforms all other models. This highlights an important insight: while augmentation techniques can enhance a model's ability to generalize, they do not guarantee improved performance in every scenario. In less complex scenes, the 36-epoch checkpoint's inherent training may have been more effective in accurately detecting pedestrians without the need for extensive augmentation. This observation emphasizes that the effectiveness of a model can depend significantly on the characteristics of the input data, and sometimes a model trained on straightforward images may achieve better results than those trained with aggressive augmentation in similar contexts.

Let us consider another example of a bit more complex scene,with varied lighting and many obstructions:

In the given complex image, which features varied lighting conditions and numerous obstructions, the 12-epoch checkpoint model with augmentation has demonstrated superior performance compared to the other models. This scenario illustrates the effectiveness of the augmentation techniques, as they helped the model adapt to challenging environments with multiple pedestrians and varying visibility. The enhancements made during training equipped the 12-epoch model to better handle the intricacies of complex scenes, allowing it to accurately identify and localize pedestrians despite the presence of obstructions and inconsistent lighting. This reinforces the idea that, in certain situations, particularly those with high variability, models that leverage augmentation can significantly outperform those that do not.

In conclusion, the ensemble approach, which combines the strengths of multiple models, demonstrates varied performance across different images, effectively capitalizing on the unique capabilities of each model. By employing a voting mechanism to select the best outputs, we can enhance the overall accuracy and robustness of pedestrian detection. This strategy allows us to compare and evaluate the outputs of individual models, leading to more reliable predictions, particularly in complex scenes with diverse challenges. Ultimately, the ensemble method not only improves detection rates but also provides a more comprehensive understanding of model performance under varying conditions.

Why Use an Ensemble of Models?

Ensemble learning operates on the principle that a group of models can outperform any single model by mitigating individual weaknesses and amplifying their strengths. Each model in the ensemble is trained on different checkpoints, learning distinct features and patterns from the training data. For example, some models may excel in simpler scenes with fewer obstructions, while others are designed to handle complex environments with various light conditions and occlusions. By aggregating their predictions through a voting mechanism, we can enhance the reliability of the output, ensuring that the final detection is based on a broader understanding of the data.\

The Role of Augmentation Techniques

Extra augmentation techniques play a critical role in preparing the models for real-world variability. By intentionally introducing variations such as blurriness, color changes, occlusions, and geometric transformations, we create a more complex training environment that better simulates the challenges faced in actual scenarios. These augmentations force the models to learn robust features that are invariant to such changes, ultimately leading to improved generalization.

Combining these complex learning patterns with the straightforward representations learned from models trained with minimal augmentation enables a more comprehensive understanding of pedestrian detection. The ensemble can leverage the nuanced insights gained from models trained on complex, augmented data while still incorporating the clarity and reliability offered by simpler models. This synergy results in better predictive outcomes, as the ensemble captures a wide range of scenarios and potential variations that a single model might overlook.

Attention maps:

Attention maps are valuable tools in understanding how models like DINO focus on different regions of an image during the process of pedestrian detection. In my GitHub repository, I implemented a code snippet that extracts attention weights from the self-attention layers of both the decoder and encoder within the model's architecture. Specifically, hooks are registered to capture these weights during the forward pass, allowing us to visualize which parts of the image the model prioritizes when making predictions.

By processing several random images from the dataset, the model generates attention maps for each layer, revealing how different layers contribute to the overall understanding of the scene. These attention maps are then resized to match the original image dimensions and overlaid on the images, providing a visual representation of the model's focus areas. This visualization helps in interpreting the model's decision-making process, illustrating how it identifies and distinguishes pedestrians from the background and other objects. By analyzing these maps, we can gain insights into potential strengths and weaknesses in the model's performance, particularly in complex scenes with varying conditions.

Example:

Attention Map Analysis and Insights

The attention map generated from the scene highlights certain features and elements in the image, providing insights into the model's focus areas. Here's a breakdown of what the attention map reveals and why these patterns are observed:

1. **High-Attention Regions:**
   - Brighter colors (yellows and lighter greens) indicate areas where the model's attention is concentrated.
   - In the image, these high-attention regions are focused on key areas where human activity is detected, such as around people on the path.
   - Attention is also given to the path itself, particularly the central walking area where people are moving.
2. **Attention in Contextual Features (Trees and Roads):**
   - Trees and parts of the road also receive moderate attention, as shown by lighter green or yellowish highlights.
   - This broader attention to trees and roads suggests that the model has learned to see these elements as contextual cues that frequently appear in scenes with people.
   -
3. **Model Learning from Dataset Composition:**
   - The attention pattern likely reflects the dataset's structure. Since your training dataset includes around 200 similar images (with people, trees, and roads), the model has generalized these elements as commonly associated features.
   - This "dataset bias" means the model has learned to focus not only on people but also on other consistent elements like trees and roads, which are often present in similar scenes.
4. **Contextual Co-occurrence Effect:**
   - Due to frequent co-occurrence, the model has learned an implicit association: trees and roads often appear in scenes with people. Thus, it distributes attention across all these recurring elements rather than isolating people as the only points of interest.
   - This co-occurrence helps the model provide context when interpreting a scene but can also lead to a broader distribution of attention across the background.
5. **Implications for Attention Patterns:**
   - The attention map effectively prioritizes areas of high human activity, which indicates that the model is responsive to movement and interaction within the scene.
   - The gradient of attention, from people to roads to trees, makes sense within the given context and suggests that the model recognizes both foreground and background elements in the image.
   - Background elements like the tree canopy and less active areas receive lower attention (dark blue and green), which confirms that the model is focusing more on features likely to contain human activity.
6. **Conclusion on Attention Map Effectiveness:**
   - The attention map shows that the model appropriately distributes attention across areas of potential human activity while de-emphasizing less relevant background features.
   - The attention distribution indicates a balance between immediate points of interest (people) and supporting contextual elements (trees and roads), suggesting the model has achieved a level of contextual awareness through training.

Future Implementations:

Explicit Training Focus:

- **Use Segmentation Masks**: Incorporate segmentation masks or more precise annotations to guide the model's attention specifically towards pedestrians. This can help the model learn to distinguish between people and background elements more effectively.
- **Add Penalizing Loss Terms**: Include loss terms that penalize attention to background elements, encouraging the model to focus more on relevant features associated with pedestrians and reducing false positives.

4. Hyperparameter Tuning:

- **Optimize Learning Rate and Batch Size**: Perform systematic hyperparameter tuning to find the optimal learning rate and batch size. Utilizing techniques such as grid search or Bayesian optimization can significantly enhance model performance.
- **Adjust Augmentation Parameters**: Experiment with the intensity and frequency of augmentation techniques to strike a balance between model robustness and overfitting.