Archaeology of Intelligent Machines

Transfer Learning in High-Dimensional Regression

21st of January, 2025

Abstract

Our project aims to compare transfer learning and fine-tuning using the **MTCNN** model for facial recognition. We evaluate the performance of the pre-trained version against a version with a manually fine-tuned P-Net layer on the same dataset. This analysis highlights differences in accuracy, runtime, and robustness, providing practical insights into the trade-offs between these approaches for domain-specific tasks.

Approach and Steps

Transfer Learning:
We use the pre-trained weights of the entire network to perform facial recognition tasks.

Fine-Tuning:
This involves retraining specific layers, such as the P-Net while the other layers are unchanged. This adapts the model to the target domain, improving performance.

Testing:
We evaluate the pre-trained model and the fine-tuned model on the same dataset. The evaluation process compares predictions to ground truth, classifies them, and calculates recall and precision for a summary of model performance.

Understanding the Data

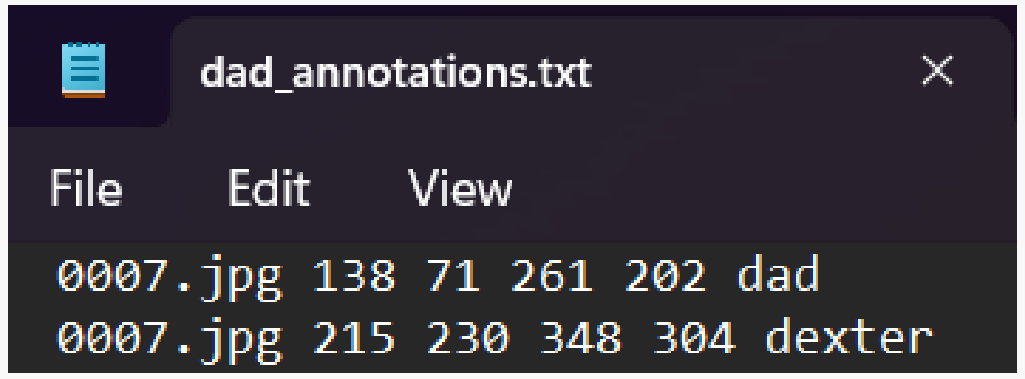
Images

We have a training dataset of 4,000 images with a 480x360 resolution.

Annotations

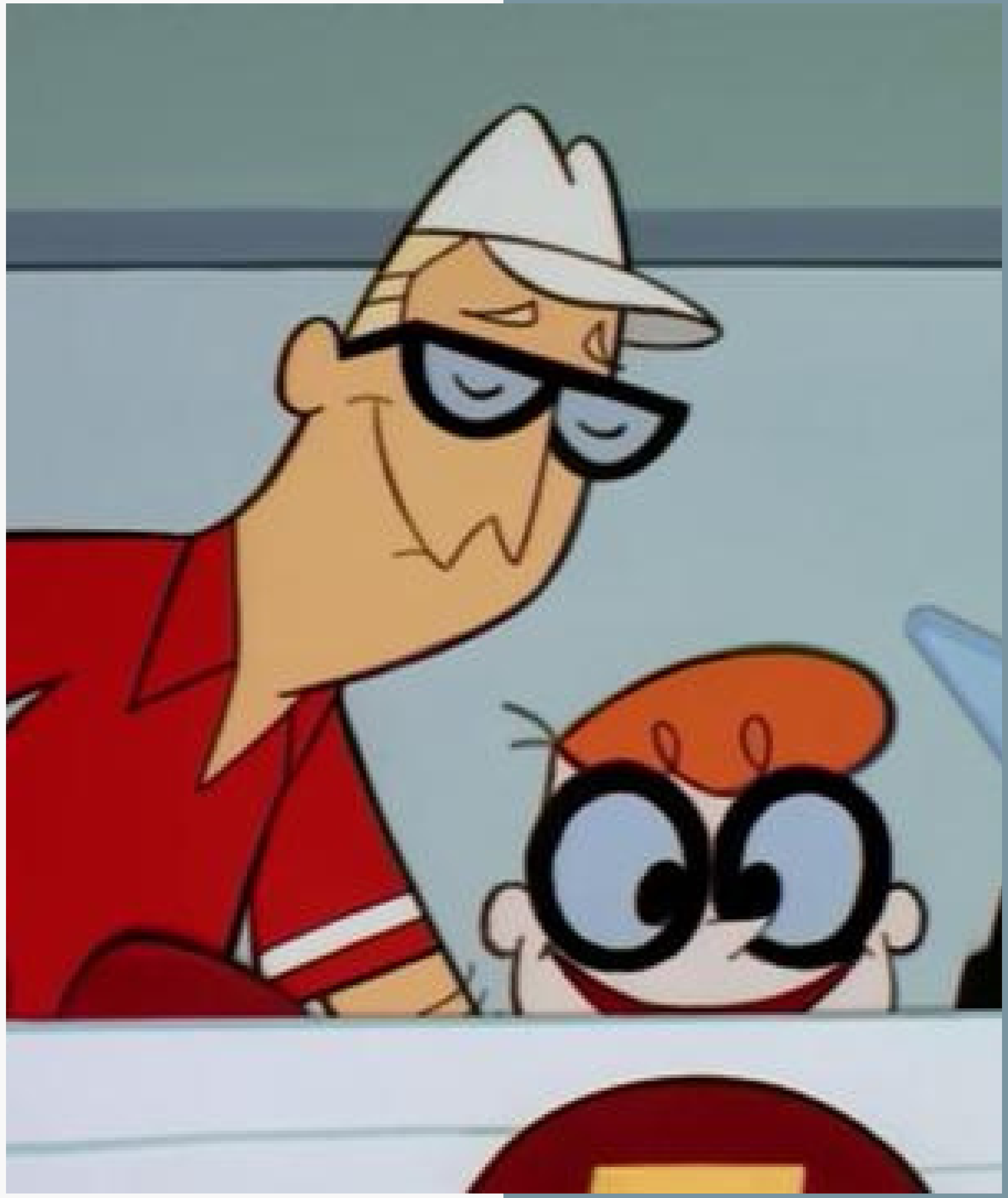
For each 1,000 images, we have an annotation file that contains the image file path, the bounding boxes (x_min, y_min, x_max, y_max) for the face and the character represented.

NOTE: there can be multiple faces in each image.



```
0007.jpg 138 71 261 202 dad
0007.jpg 215 230 348 304 dexter
```

Annotations for the respective image



Example of Image from the Dataset

Methodology for Transfer Learning

Model Loading:

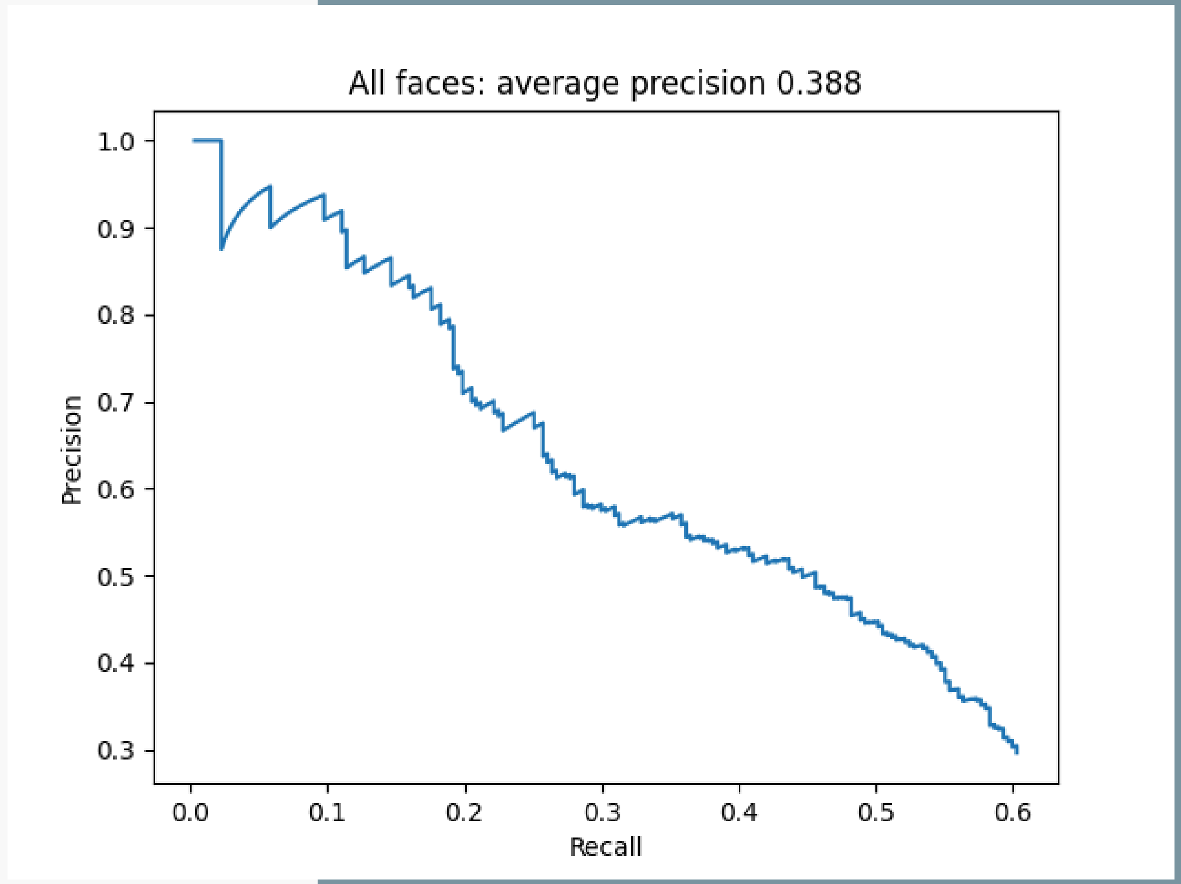
- Initializing the MTCNN model.

Transforming Output into the Required Format:

- Get model's prediction.
- Transform them into the desired format for evaluation:
  - image file path, bounding boxes and confidence score must be stored.
  - append each of them to a numpy file.

Evaluation:

- Run the evaluation code, check the average precision on a 200 images test sample.



Actual Representation of Model Performance

Methodology for Fine-Tuning

Data preprocessing:

- Rescale the initial images to 12x12px in order to properly run the model.
- Append to X the image file paths and the bounding box for each face but also some random patches that contains no-faces.

Layers Freezing:

- Freeze both RNet (Refine Network) and ONet (Output Network) layers to preserve their original parameters.

Training PNet Layer:

- Create a custom PNet (Proposal Network) layer:
  - Convolutional Layers: These layers scan the image to find important features, like shapes or edges.
  - Activation Functions (PReLU): Process the output from the convolutional layers.
  - Pooling: This step simplifies the image, focusing on the most important features while reducing details.
  - Output Layers: These predict the bounding boxes.

Methodology for Fine-Tuning

Training PNet Layer:

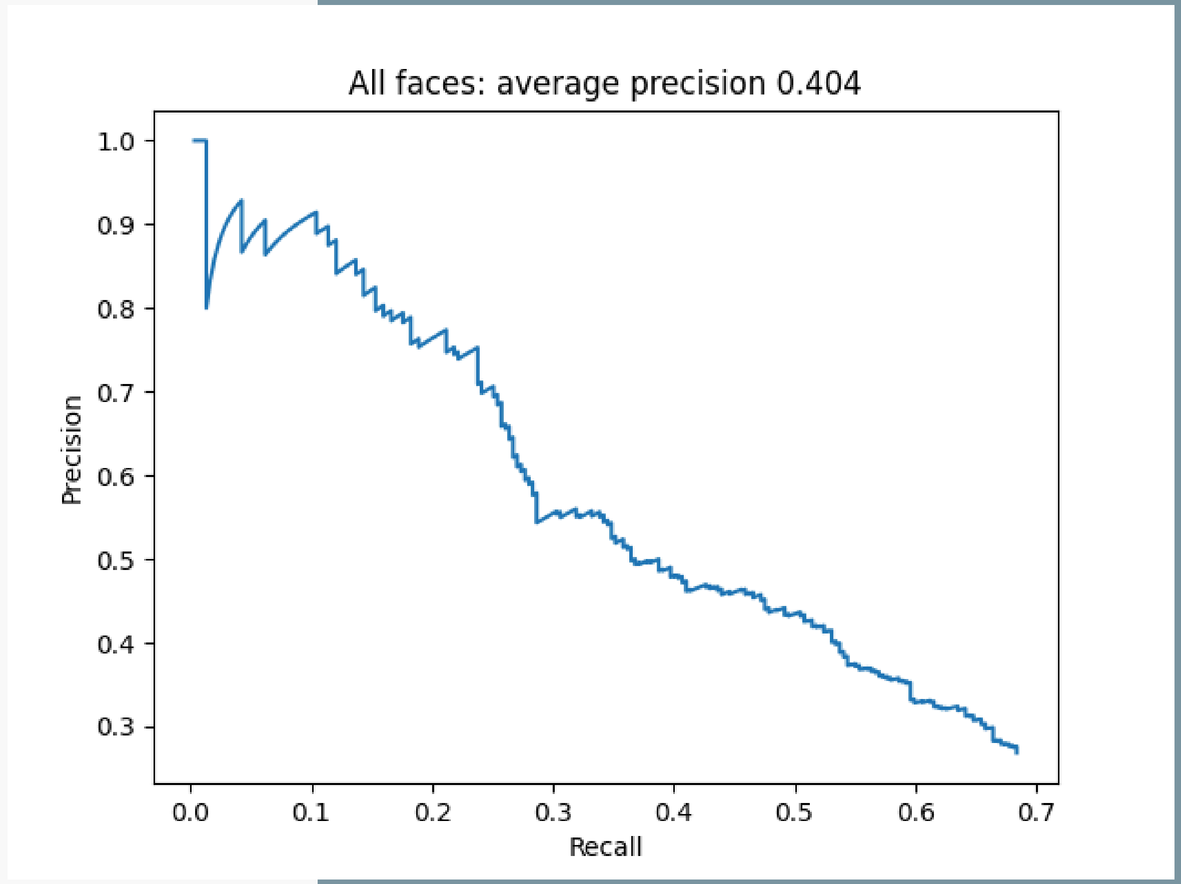
- Train the layer:
    - Loss Functions: Define the classification loss for detecting faces and the bounding box regression loss.
    - Training Loop: For each epoch, the model:
        - Forwards the batch of images through the PNet.
        - Computes the classification and regression losses.
        - Adjusts the model's weights by using the calculated losses.

Transforming Output into the Required Format:

- Get model's prediction.
- Transform them into the desired format for evaluation:
    - image file path, bounding boxes and confidence score must be stored.
    - append each of them to a numpy file.

Evaluation:

- Run the evaluation code, check the average precision on a 200 images test sample.



Actual Representation of Model Performance

Methodology for Testing

Load the Ground Truth:

- Load the ground truth files with annotations.

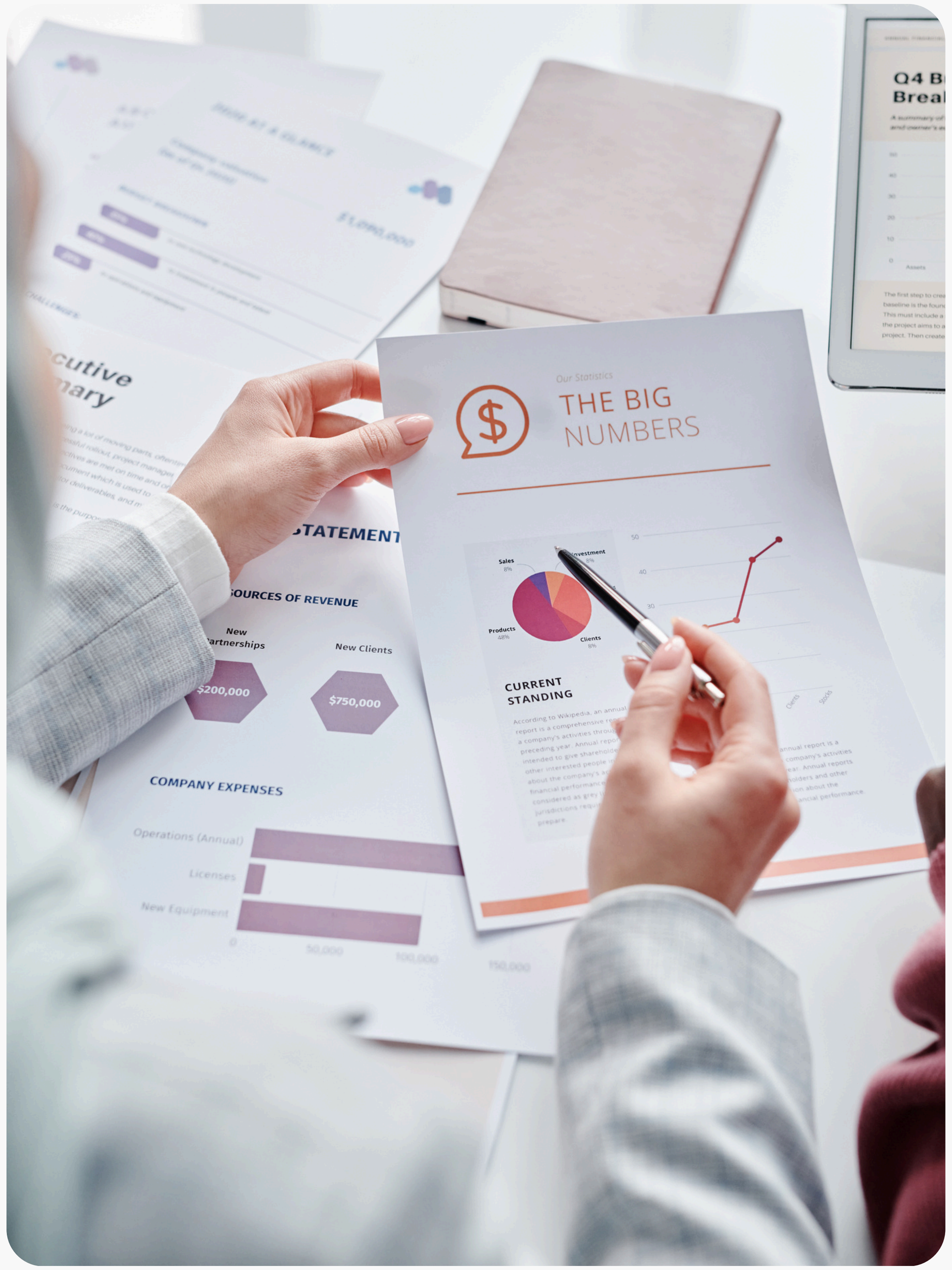
Load the Predictions:

- Get model's prediction.
- Load the numpy files with annotations.

Compare:

- Run the evaluation code:
    - Computes the IoU, which is the area of overlap between two bounding boxes divided by the area of their union. This is used to measure how much two bounding boxes overlap.
    - If the maximum IoU exceeds a threshold (0.3 in this case), and the ground truth box has not been matched to another detection, it is considered a true positive. Otherwise, it is a false positive.
    - It also calculates cumulative true positives and false positives.
    - Finally, recall and precision are calculated.
- Plot Precision-Recall Curve.

Expected Outcomes



Pre-trained MTCNN model:

- Expected to be more accurate due to extensive training on a larger, diverse dataset.
- Better performance than the Fine-tuned model.
- Slower execution.

Fine-tuned MTCNN model:

- Anticipated to have improved specificity for the given dataset after fine-tuning.
- Expectation that the PNet layer is easy to retrain, allowing for better adaptation to specific data characteristics.
- Potential slight performance drop due to limited training data, but improved detection on specialized tasks.

Real Outcomes

Speed

- The pre-trained MTCNN was definitely faster than the fine-tuned version of it.
- 6-7 times faster (including the training).

Ease of Use

- Difficulties in fine-tuning the model using a custom training dataset.
- Numerous bugs while training the fine-tuned version.

Performance

- 4.1% better score on the fine-tuned version.

Conclusion

The pre-trained MTCNN model demonstrated superior speed, making it more efficient for quick deployments. However, the fine-tuned model showed a slight performance improvement, suggesting better adaptation to the specific dataset. The challenges in fine-tuning highlight the need for robust debugging and potential adjustments in the training process. Overall, the choice between pre-trained and fine-tuned models will depend on the specific requirements of speed versus performance for the task at hand.