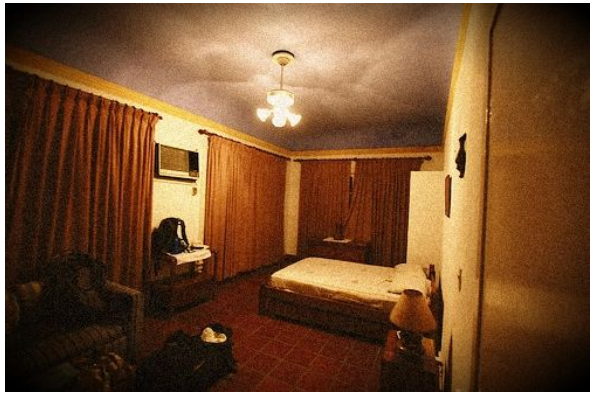
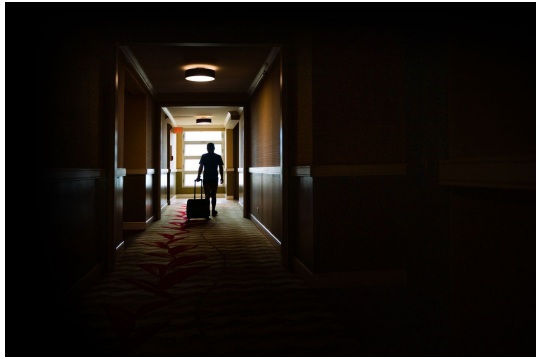
# PD 1.3 - Phase I Project Demo (Team)

Sneha, Heron, Zinnia, Faisal, Aiah

# Business Understanding



- The human trafficking crisis is a global issue affecting millions, often hidden in hotel rooms
- A common challenge for law enforcement is identifying victim locations from limited visual evidence is time-consuming and error-prone.
- Thousands of hotels with unique room designs make manual identification inefficient, which brings the need for automation
- Project Goal: Develop an AI model to match hotel room images to specific hotels.
- This will enhance law enforcement response time, improve victim rescue success, and demonstrate AI's role in tackling human rights issues



# Baseline Model: Random Assignment

- Establish a performance benchmark
- Extract full list of hotel IDs from dataset
- For each test image, randomly assign 5 hotel IDs as predictions
- Submit random assignment predictions to Kaggle to be evaluated against the leaderboard.
- Random model will likely have an extremely low MAP@5 score

[5]:
```
hotel_df = pd.read_csv("./hotels-50k/dataset/hotel_info.csv")
display(hotel_df.head())
```

| | hotel_id | hotel_name | chain_id | latitude | longitude |
|---|---|---|---|---|---|
| 0 | 391 | Extended Stay America - Fairbanks - Old Airpor... | 72 | 64.83538 | -147.82330 |
| 1 | 392 | Hilton Hangzhou Qiandao Lake Resort | 3 | 29.60819 | 119.07290 |
| 2 | 393 | Taj Lands End | -1 | 19.04391 | 72.81879 |
| 3 | 395 | Cambridge Suites Hotel Sydney | -1 | 46.13663 | -60.19551 |
| 4 | 396 | Tamanu Beach | 14 | -18.84213 | -159.78794 |

# EDA and Metrics

```
print("Image count:", len(data_df))
print("Hotel count:", len(data_df["hotel_id"].
print("Chain count:", len(data_df["chain_id"].
```

```
Image count: 1124215
Hotel count: 50000
Chain count: 93
```

- 1.4 million images across thousands of hotels.
- Hotel images labeled with hotel ids.
- Unlabeled images to test dataset.
- Significant variation in hotel room designs.
- Class imbalance as some hotels have more images than others.
- Need for data augmentation and better feature extraction.
- Expect MAP@5 score expected to be low
- This serves more as a lower-bound benchmark for model improvement.

# Available Model Repositories for PyTorch & CNNs

1. **TIMM (Torch Image Models) – Best for diverse, state-of-the-art models**
2. **Torchvision – Best for official, standard CNNs**
3. **MMClassification – Best for research and customization**

| Feature | TIMM | Torchvision | MMClassification |
|---|---|---|---|
| CNNs Available | Wide Selection | Standard Models | Many CNNs |
| Vision Transformers | Yes | No | Yes |

## Why TIMM is the Best Option?

- Largest selection of pre-trained models (CNNs + Transformers)
- Supports modern architectures (EfficientNet, RegNet, ResNet)
- Optimized for fast training & inference
- Easy to use with flexible APIs
- Well-maintained and frequently updated

# Optimizing the Current Baseline Model (ResNet-34) & Selecting the Best Alternative

## Recommended CNN Architectures:

- ResNeXt-101 / Res2Net – Best for fine-grained recognition, strong feature extraction, handles high variance in hotel interiors
- EfficientNet-B3 to B5 / RegNetY – Best for scalability, maintains accuracy with efficiency, ideal for large-scale search systems
- MobileNetV3-Large / EfficientNet-Lite0-4 – Best for real-time inference and mobile deployment, optimized for low-power devices

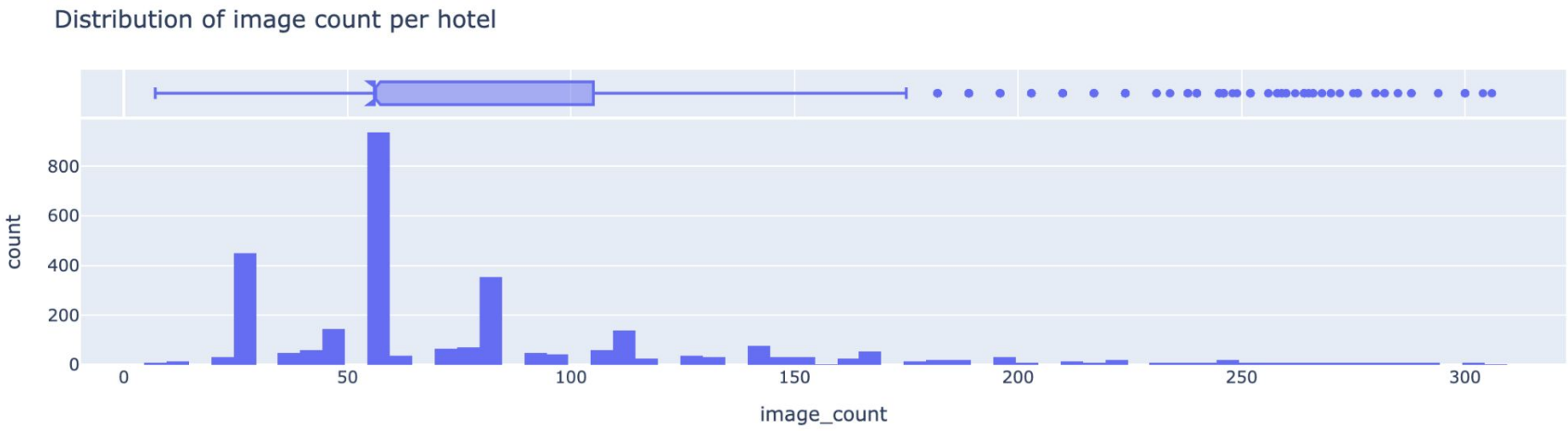
# Optimizing the Current Image Processing Library (Albumentations)

## Image Augmentation Libraries used with TIMM

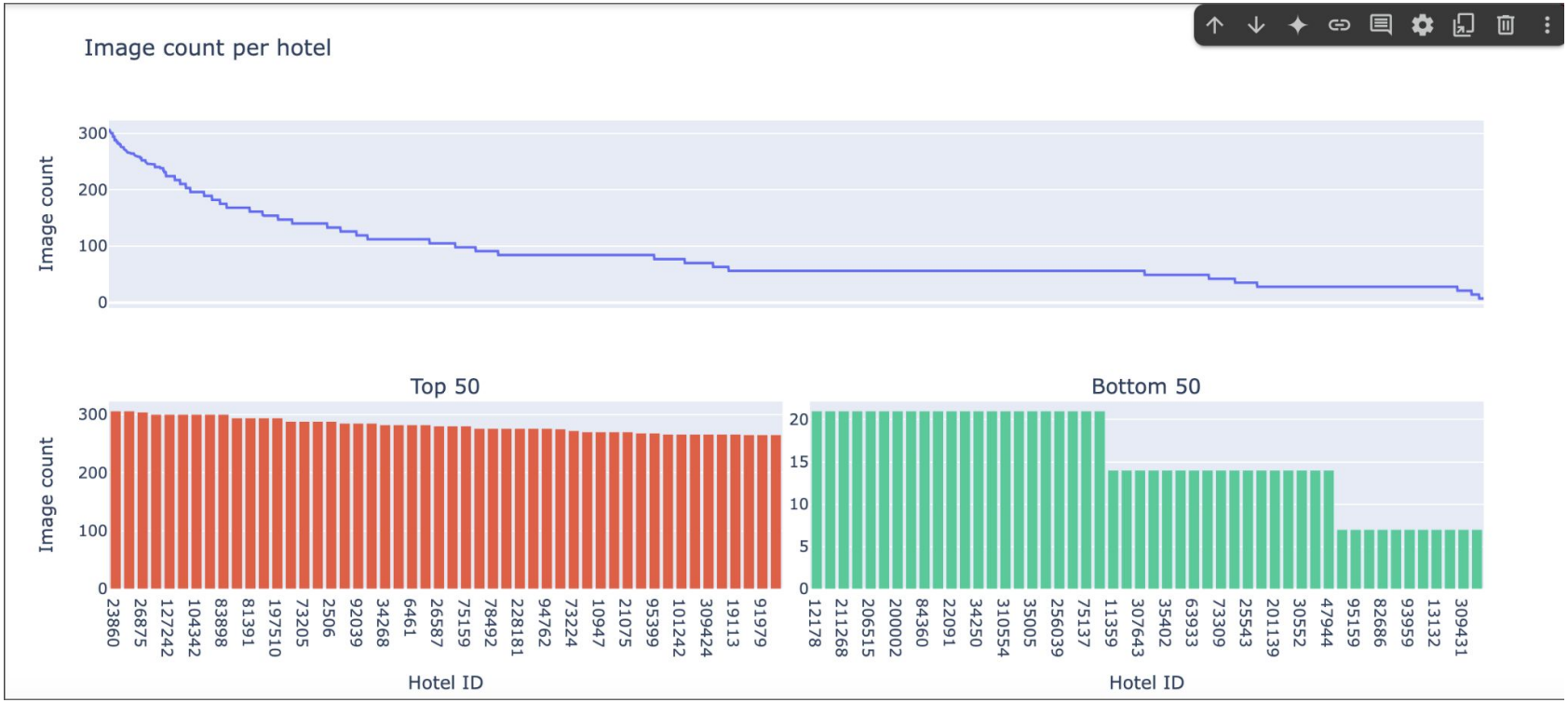
- **Albumentations** – Best for **fast, high-performance augmentations**, optimized for deep learning.
- **Torchvision Transforms** – Best for **PyTorch-native** augmentations, well-integrated but lacks key augmentations.
- **imgaug** – Best for **handling complex image distortions**, including adversarial transformations.

a

# Data Visualization



# Data Visualization



# Data Visualization

**Current Code Framework**

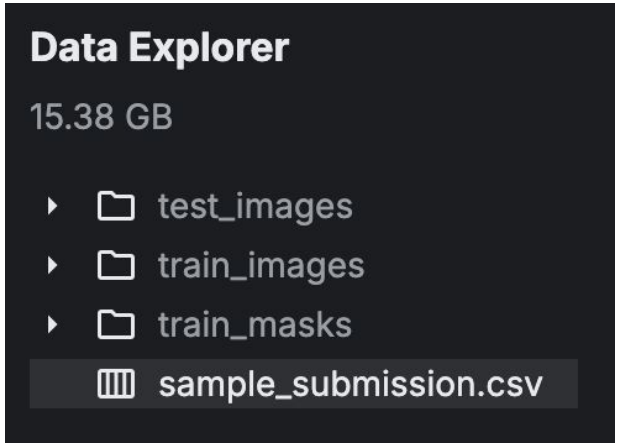
- **Loads the dataset** (train.csv) containing image IDs and hotel IDs.
- **Checks for actual image files** in the storage directory to ensure only valid images are used.
- **Filters out missing or invalid data** (removes rows where image_id or hotel_id is missing).
- **Randomly selects hotels** and displays a sample of room images for each.
- **Visualizes images in a grid format** to help verify data quality before training the AI model.

**Why This Feature Matters**

- **Ensures clean and valid data** by removing missing entries.
- **Speeds up AI training** by eliminating non-existent images.
- **Provides a visual check** to confirm images match their respective hotels.
- **Improves model accuracy** by ensuring only properly labeled images are used.

# Model Development

- First issue was storage
  - The competition has 44.7k files for analysis that was about 15 GB of data, which was too much for our computers to handle
- Fixed issue by:
  - Investing in extra google drive storage for easy access with Google Colab
  - Getting external pen drives to store data
  - Used Kaggle interface
  - Creating a smaller dataset for local use and testing



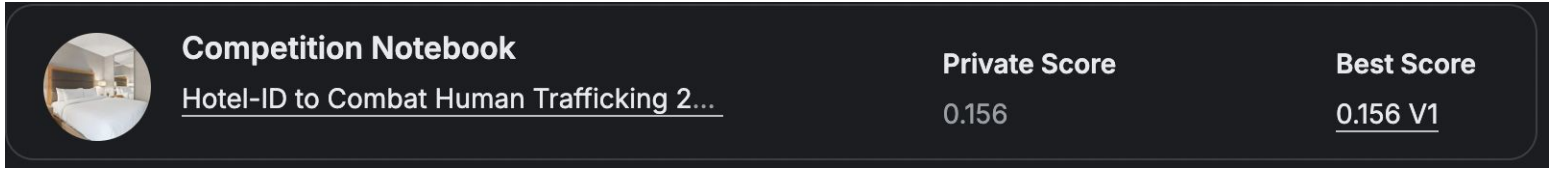
# Model Development

- Once storage was resolved, there were problems with the initial EfficientNet model
  - Hugging face
  - “AttributeError: module 'torch' has no attribute 'frombuffer'”
- Needed to either upgrade torch to 2.0.1, or downgrade transformers to 4.29.2 and safetensors to 0.3.0
  - Kaggle interface does not allow upgrade to 2.0.1, so downgrade was tried but failed to remove error
  - Currently in process of moving from Kaggle interface to Google Colab in order to upgrade torch
  - Will continue to work on EfficinetNet model and tweak in order to be able to run model

# Performance Evaluation of Current Model

- Baseline submission of random assignment model expected MAP@5 (~0.001-0.005).
- Position on the leaderboard near the bottom.

- With the ResNet-34 Model we saw improvement in MAP@5 (0.156).
- Position in leaderboard should move above other completely random models.



# Future Plans

- Get EfficientNet model to run
  - Get model off of Kaggle interface and use one of the local storage methods to properly upgrade or downgrade packages in the model to get resolve current errors
- Add similarity model and training/testing set from EfficientNet model starter to ResNet model starter to see if we can improve current notebook score
- Run a few improvement strategies in parallel
  - Start pulling techniques from higher scoring models on Kaggle to further improve notebook
  - Change the base model repository to see if there is improvement
  - See if changes to image augmentation allow for improvement