End-to-End Computer Vision Pipeline Report (Amazon Rekognition Custom Labels)

1. Problem Statement and Dataset Description

The objective of this project was to build a binary image classifier using Amazon Rekognition Custom Labels to distinguish between cats and dogs. A total of 40 cleaned images (20 cats, 20 dogs) were used for training and testing, alongside 12 additional unseen images (6 per class) for model evaluation.

Images were collected from public-domain sources and manually checked for quality, size (≥300x300), and correct file format (JPEG). An extra effort was made to remove transparency, CMYK color space, and metadata that may have previously caused training failures.

2. Labeling Method Used

All image labels were assigned via the **AWS Console** using the Custom Labels UI.

Initially, we attempted to create datasets, assign labels, and trigger training **entirely via Python (boto3)**. However, we encountered repeated issues such as:

- create_dataset_entries not available in our region,
- Manifest file errors: “too many invalid rows”,
- Inability to access the internal manifest used by Rekognition unless training succeeded.

As a result, we switched to labeling via the **Console UI**, which enabled us to assign labels directly on each image and visually confirm dataset health before training.

Key screenshots of this process are included in the appendix.

3. Training Configuration

- **Model Version ARN**: arn:aws:rekognition:...:project/CatDogClassifier-1/version/CatDogClassifier-1.2025-05-12T15.53.10/...
- **Training method**: Console-triggered
- **Training time**: ~11 minutes

- **Instance type (default)**: Managed by Rekognition (no explicit configuration)
- **Estimated cost**: Within free tier (below 10 training hours and 4 inference hours)

4. Quantitative Metrics and Qualitative Error Analysis

The model was evaluated using 12 unseen images:

Metric	Value
Overall Accuracy	**91.67%**
Cat Accuracy	6/6
Dog Accuracy	5/6 (1 Unknown)

We used rekognition.detect_custom_labels() to evaluate each test image. Predictions were stored and visualized using a confusion matrix. One dog image was not classified, returning Unknown – likely due to low model confidence or deviation from the training distribution.

5. Key Takeaways and Challenges

1. **Rekognition is highly sensitive to image formatting**:
   Even seemingly valid .jpg files failed silently due to metadata, transparency, or CMYK color space issues.
2. **Error feedback is minimal**:
   Console and boto3 often report vague messages like “too many invalid rows” without showing which rows or why.
3. **Manifest visibility is restricted**:
   If training fails, there is no way to download or inspect the generated manifest, making debugging extremely difficult.
4. **Hybrid approaches work best**:
   The most reliable pipeline combined code-based image preprocessing with manual labeling in the console.

6. Recommended Next Steps

- Use stricter image cleaning (e.g., force RGB, resize, strip EXIF) before uploading any future dataset.
- If building a larger pipeline, consider generating and managing .manifest manually, bypassing the console altogether.

- Explore custom thresholds or multiple predictions using confidence scores (not just top-1 class) for edge cases.