
Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Project Overview

This project implements an **image-based smart meter classification system** using **Deep Learning (CNNs)**.

Given an input image of an electricity meter, the system predicts the **meter brand/type** with a confidence score.

The model is trained and evaluated on **38 different meter classes**, achieving ~94% accuracy on unseen test images.

A professional **Streamlit dashboard** is also provided for real-time inference.

Problem Statement

Electricity meters from different manufacturers often have **visually similar designs**, making manual identification slow and error-prone.

The goal of this project is to:

- Automatically identify the **meter type** from an image
- Handle variations such as **blur, zoom, angle, and lighting**
- Provide **confidence-aware predictions** suitable for real-world deployment

Solution Approach

Model Architecture

- Backbone - ResNet-18
- Framework - PyTorch
- Clustering - HDBSCAN
- Input Size - 224×224 RGB images
- Output - 38-meter classes
- Loss Function - Cross Entropy Loss
- Optimizer - Adam
- Output - SoftMax over 38 known meter classes
- Dashboard - Streamlit

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Training Strategy

A two-phase training strategy was used:

Phase 1 – Classifier Training

ResNet backbone frozen

Only the final classification layer trained

Phase 2 – Fine Tuning

Last ResNet block unfrozen

End-to-end fine tuning with lower learning rate

This approach significantly **improved generalization and reduced overfitting.**

Dataset Details

Dataset Design & Splitting Strategy

Total Images - 1,975

Total Classes - 38

Directory structure

data/processed/

 └── train/

 └── val/

 └── test/

Folder names represent **ground truth labels.**

- Finally Automated **image-wise inference audit** on test data instead of manually checking image after image to know the model accuracy.
- Detailed CSV report generated for inspection

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Class-wise splitting logic (important)

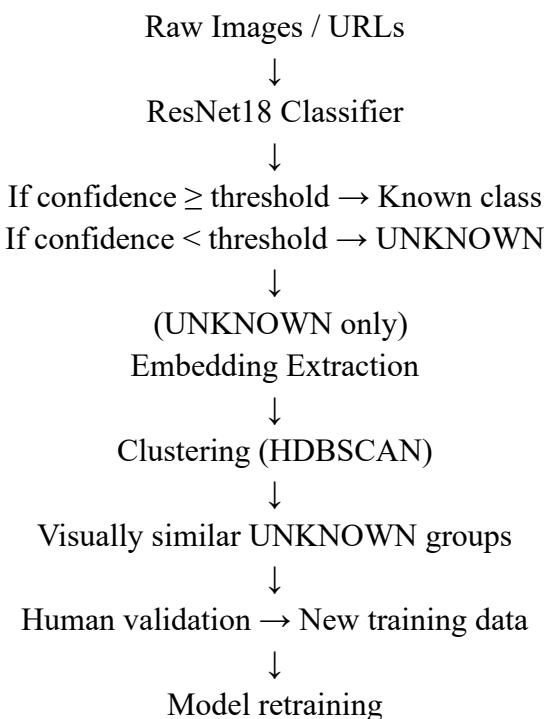
Because meter data is **highly imbalanced**, we used **adaptive splits**:

Images per class	Split Strategy
Very few images	Train only
Medium count	Train + Test
Large count	Train + Val + Test

This avoids:

- Overfitting small classes
 - Artificially inflating accuracy
 - Starving rare meters during training
-

High-Level Architecture



Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Confidence Threshold & UNKNOWN Logic

Final decision rule:

If Top-1 confidence $\geq 50\%$ \rightarrow Accept prediction

If Top-1 confidence $< 50\%$ \rightarrow Mark as UNKNOWN

This ensures:

- High precision for known meters
- Safe fallback for unseen meters
- No forced misclassification

UNKNOWN + Top-3 + Embedding Strategy (VERBATIM SECTION)

UNKNOWN does not mean “no information”

Even when an image is marked UNKNOWN (Top-1 confidence $<$ threshold):

- We still compute SoftMax
- We still store Top-3 predictions + confidences
- We do not discard model knowledge

So UNKNOWN = “*model is unsure, not blind.*”

What metadata we keep for UNKNOWN images

For every UNKNOWN image we store:

- image_url
- top1_class, top1_confidence
- top2_class, top2_confidence
- top3_class, top3_confidence
- final label = UNKNOWN

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

This allows:

- Human review
 - Pattern analysis (e.g., “mostly looks like LT2 / GENUS1”)
 - Safer decision-making downstream
-

Embeddings are used only for similarity, not labelling

From UNKNOWN images:

- We extract **embedding vectors** using the CNN backbone
- These embeddings capture **visual similarity**, not class labels
- No classifier bias is introduced here

This is the **correct separation of concerns**:

- Classifier → *what do I think this is?*
 - Embeddings → *what looks similar to what?*
-

Clustering UNKNOWN images = automated segregation

Using embeddings + HDBSCAN:

- UNKNOWN images are grouped into **subfolders**
- Each subfolder contains **visually similar meter images**
- Results:
 - $\text{cluster_id} = -1 \rightarrow$ noise / junk / bad images
 - $\text{cluster_id} \geq 0 \rightarrow$ **new meter candidates**

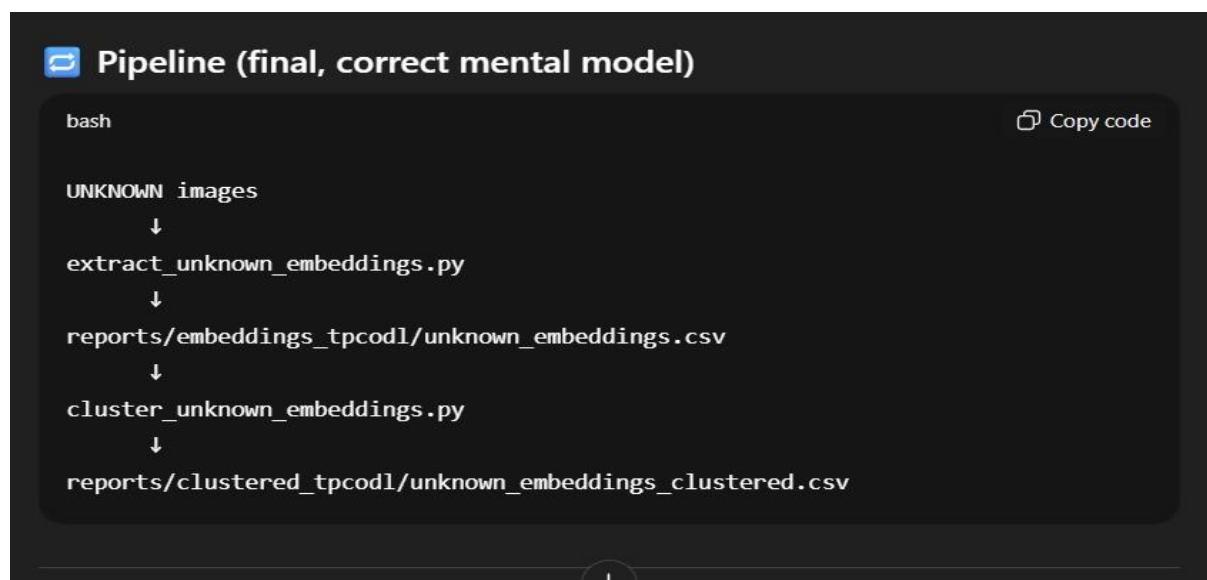
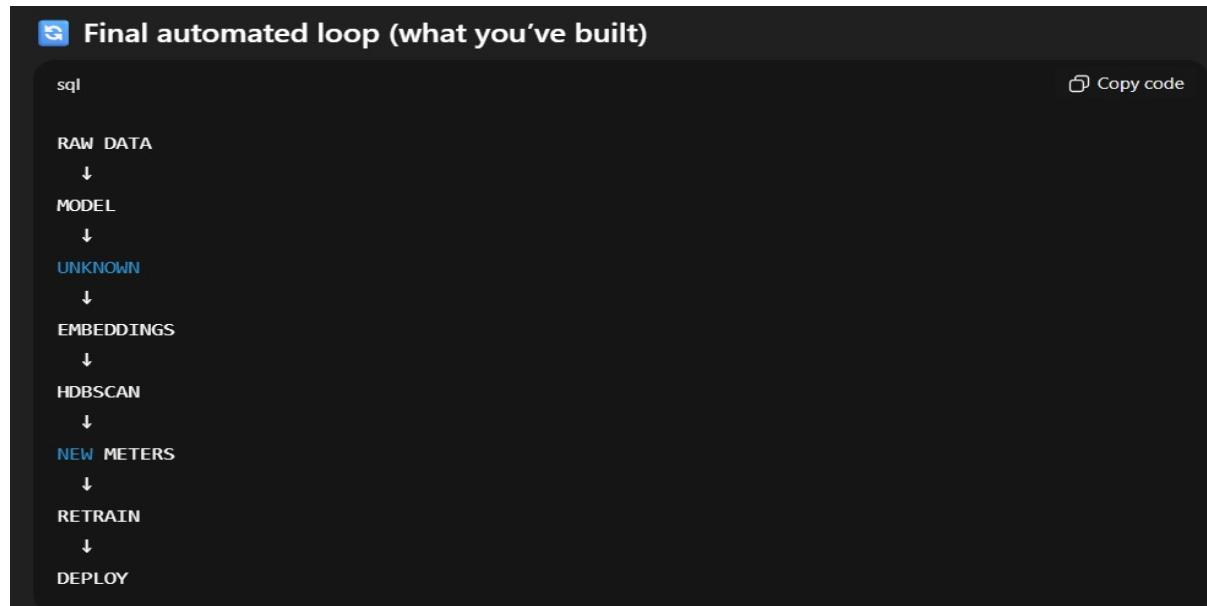
So:

UNKNOWN → clusters → similar images grouped together

This replaces **manual image-by-image segregation**.

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline



Why this works in production

- Keeps system safe
- Reduces manual effort drastically
- Enables fast dataset growth
- Supports continuous learning

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Project Structure

METER CLASSIFICATION/

```
├── src/
│   ├── train.py
│   ├── evaluate.py
│   ├── predict.py
│   ├── url_predict_batch.py
│   ├── extract_embeddings.py
│   ├── extract_unknown_embeddings.py
│   ├── cluster_unknown_embeddings.py
│   └── dashboard.py

└── data/
    ├── processed/
    │   ├── train/
    │   ├── test/
    │   └── val/
    └── batchtest/
        └── *.xlsx

    └── models/
        └── final_meter_model.pth

    └── reports/
        ├── embeddings_tpcdl/
        │   └── unknown_embeddings.csv
        ├── metadata_tpcdl/
        │   └── unknown_metadata.csv
        └── clustered_tpcdl/
            ├── unknown_clustered.csv
            ├── cluster_summary.csv
            └── clusters/
                ├── cluster_0/
                ├── cluster_1/
                └── ...

    └── requirements.txt
    └── README.md
```

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

File-by-File Purpose

train.py

- Trains the ResNet18 classifier
- Handles class imbalance
- Saves final model

evaluate.py

- Evaluates on test data
- Generates accuracy, classification report, confusion matrix

predict.py

- Single image inference
- Outputs prediction + confidence + Top-3

url_predict_batch.py

- Takes Excel with image URLs
- Downloads images
- Runs inference
- Applies UNKNOWN logic
- Saves CSV output

extract_embeddings.py

- Extracts embeddings for **known** images (optional analysis)

extract_unknown_embeddings.py

- Extracts embeddings only for UNKNOWN images
- Saves:
 - Embedding vectors
 - Metadata separately

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

cluster_unknown_embeddings.py

- Normalizes embeddings
- Runs HDBSCAN clustering
- Generates:
 - Clustered CSV
 - Summary CSV
 - Cluster folders

dashboard.py

- Streamlit UI
- Single image inference
- Batch results visualization
- Class-wise image browsing

Dashboard (Real-Time Inference)

A Streamlit-based dashboard allows:

Uploading a new meter image

Viewing predicted class and confidence

Inspecting top-3 predictions

Flagging low-confidence predictions

Smart Meter Classification System

Deep Learning + UNKNOWN Discovery + Clustering Pipeline

Execution Order (IMPORTANT)

Environment

```
.\.venv\Scripts\Activate.ps1
```

Training

```
python src/train.py
```

Evaluation

```
python src/evaluate.py
```

Single Image Test

```
python src/predict.py --image path/to/image.jpg
```

Batch URL Inference

```
python src/url_predict_batch.py
```

UNKNOWN Embedding Extraction

```
python src/extract_unknown_embeddings.py
```

Clustering UNKNOWN Images

```
python src/cluster_unknown_embeddings.py
```

Dashboard

```
streamlit run src/dashboard.py
```

This project is an **end-to-end deep learning system** for:

- Classifying electricity meter images into known meter types
- Safely handling **unseen / new meter types**
- Automatically **grouping unknown meters** for faster dataset expansion
- Supporting **single-image inference, batch URL inference, and visual dashboards**
- Solves the problem of Manual segregation which is slow and error-prone