Water Meter Reader

By - Harshit Srivastava

PROBLEM STATEMENT

Utility companies face enormous logistical challenges in collecting water usage data manually. Manual readings are time-consuming, error-prone, and labor-intensive. With increasing urbanization and digitization of infrastructure, there is a strong demand for automating meter reading processes. Traditional OCR systems often fail on analog meters due to low contrast, varied font styles, or distorted characters. Our aim is to build a robust, end-to-end AI solution that can extract numeric readings from analog water meter images using a transformer-based model (Donut), optimized to work even in CPU-only environments.

KEY CHALLENGES

1. **Image Variability**: Varying lighting conditions, shadows, angles, and distortions.
2. **Meter Diversity**: Analog meters come in different designs and digit arrangements.
3. **Noisy Environments**: Scratches, watermarks, dials, and background clutter interfere with detection.
4. **OCR-Free Modeling**: Using Donut architecture means no traditional OCR; direct vision-to-text mapping is complex.
5. **CPU Constraint**: Training and inference had to be compatible with CPU-based systems like macOS.
6. **Custom Labeling Format**: Special tag-wrapped sequences required for training Donut.
7. **Evaluation Metrics**: High sensitivity to small differences in digits (e.g., "0.001" deviation).

1. Project Overview

This project involves the design, development, and evaluation of a deep learning-based system that automates the reading of water meter values from digital images. It uses an advanced OCR model built on the Donut architecture (Document Understanding Transformer) and fine-tuned on a custom dataset of real-world water meter images.

The system is designed to reduce manual labor, minimize human error, and provide a scalable solution for smart utility infrastructure. It processes input images of water meters and returns predicted numerical readings. This technology is particularly valuable for utility companies looking to implement automated billing systems and data monitoring at scale.

The project follows a defined and repeatable process for all stages—data preparation, model training, testing, evaluation, and deployment—aligned with Capability Maturity Model (CMM) Level 3 standards. All components, from dataset preprocessing pipelines to model inference and reporting, are modularized and version-controlled. This ensures consistency and reliability in performance and makes the project suitable for real-world industrial applications.

2. Objectives

- Automate the reading of analog water meters using image-based deep learning.
- Build and fine-tune a model specifically for image-to-text translation tasks.
- Ensure full compatibility with CPU-based training environments such as macOS.
- Evaluate the model using industry-standard metrics like Levenshtein Distance and strict accuracy.
- Visualize and verify predictions through a well-structured interface and output visualization.

3. Tools and Technologies

Programming Language:

Language/Library / Framework	**Version**	**Purpose / Use**
Python	**3.12**	Used as the foundation for machine learning, scripting, automation, and application development.
Jupyter Notebook	**6.5.6**	for iterative development
PyTorch	**2.7.1**	Core deep learning framework for model development and training.
Hugging Face Transformers	**4.12.0**	Provides model architectures, tokenizers, and training utilities for transformer-based models.
OpenCV	**4.12.0**	Used for image loading, resizing, and basic computer vision tasks.
PIL (Pillow)	**10.3.0**	For image opening, conversion, and manipulation (e.g., RGB conversion).
NumPy	**1.26.4**	Efficient numerical operations, array handling, and data manipulation
Matplotlib	**3.9.0**	Visualization of model outputs, predictions, and debuggin

Regex (re module)	**3.12**	Preprocessing of text labels, pattern matching, and string cleaning.

Development Environment:

- Jupyter Notebook

MLOps & Experiment Tracking:

- Weights & Biases (wandb) for real-time monitoring, versioning, and experiment comparison.

Wandb

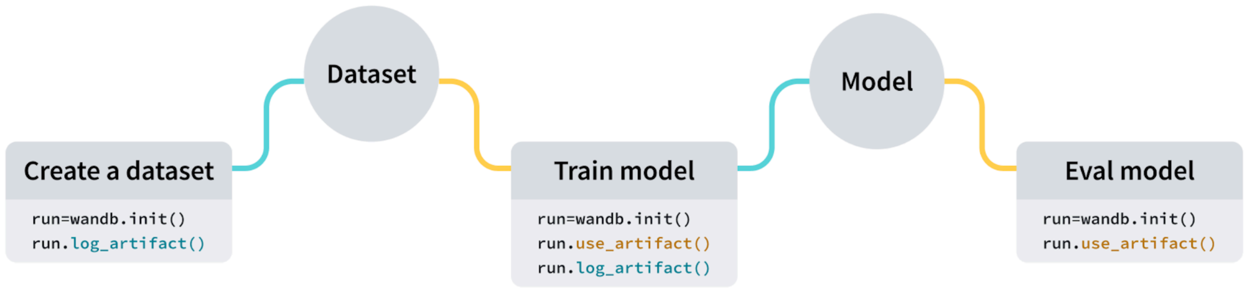
Weights & Biases (wandb) is a machine learning experiment tracking and model management tool. It integrates seamlessly with most ML/DL frameworks and is widely used for:

- Logging training/evaluation metrics
- Visualizing model performance over time
- Saving model checkpoints
- Comparing multiple experiment runs
- Hyperparameter sweeps
- Collaboration and reporting

Workflow of Wandb:

1. Initialize your W&B project.
2. Log metrics during training.
3. Monitor real-time training on your W&B dashboard.
4. Compare results from multiple runs.

5. Share visual reports with your team or stakeholders.



4. Literature Review

Traditional OCR methods, such as Tesseract, rely on character segmentation followed by recognition, which performs poorly on analog meters due to overlapping elements and noise. Recent advances in transformer-based vision models like Donut eliminate the need for intermediate OCR steps. Donut has demonstrated superior performance in document understanding tasks and is adaptable to numeric digit prediction in constrained environments. Other methods explored include CNN-based regression models and hybrid CNN-RNN-CTC pipelines, which were found less robust in noisy or distorted conditions.

5. Methodology

a. Data Collection

The Water Meter Dataset used for this project contains thousands of water meter images. The dataset is diverse in terms of angle, lighting, and meter styles, offering a robust foundation for model generalization.

b. Data Preprocessing

- **Image Processing**: All images were resized and normalized to meet the model's input requirements.
- **Text Processing**: Labels were cleaned to remove unwanted characters. Regular expressions were used to isolate numeric components.
- **Dataset Splitting**: The data was divided into training, validation, and test sets to ensure fair model evaluation.

c. Model Design

- The **Donut architecture** was adopted due to its ability to handle document-level OCR tasks.
- Implemented using the **VisionEncoderDecoderModel** class from Hugging Face Transformers.
- **DonutProcessor** handled tokenization and normalization of image-text pairs.

Donut Model: Document Understanding Transformer

Donut (Document Understanding Transformer) is an **end-to-end, OCR-free model** designed for document image understanding tasks. Unlike traditional systems that first extract text using OCR and then process it, Donut **directly maps document images to structured output (like JSON)** using a vision-to-language model architecture.

Architecture Overview

1. Input: Document Image

- Any type of document image, like receipts, forms, invoices, or meter readings.

2. Vision Encoder

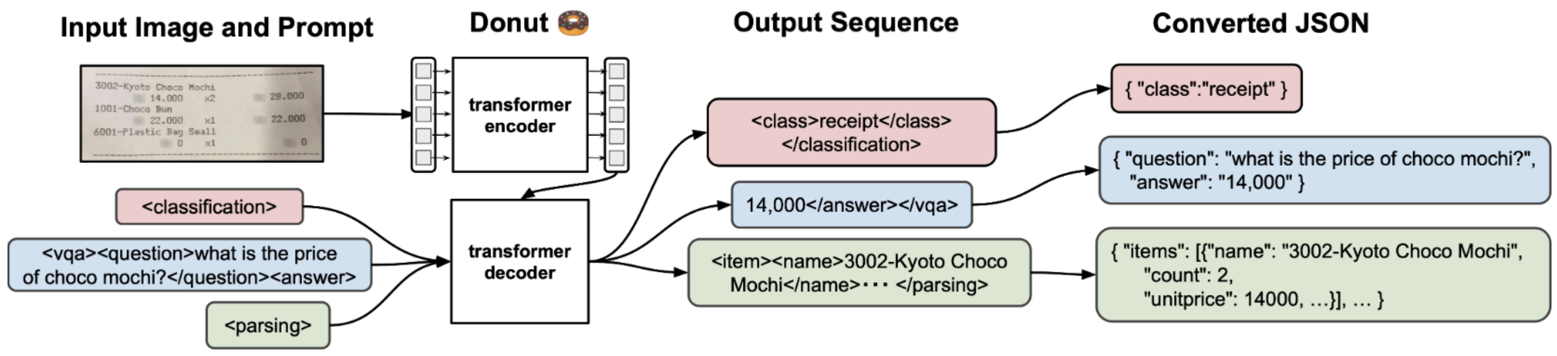
- Extracts visual features from the image.
- **Swin Transformer** processes the image hierarchically, capturing both local and global context.

3. Transformer Decoder

- Decodes the visual features into a text sequence using an autoregressive transformer decoder.
- Outputs structured text like:
- {"meter_reading": "123.456"}

4. Training Objective

- The model is trained to predict structured outputs **without OCR**, using just the visual content of the image.


Input Image and Prompt
Donut 🍩
Output Sequence
Converted JSON
transformer encoder
transformer decoder
<classification>
<vqa><question>what is the price of choco mochi?</question><answer>
<parsing>
<class>receipt</class></classification>
14,000</answer></vqa>
<item><name>3002-Kyoto Choco Mochi</name>··· </parsing>
{ "class":"receipt" }
{ "question": "what is the price of choco mochi?", "answer": "14,000" }
{ "items": [{"name": "3002-Kyoto Choco Mochi", "count": 2, "unitprice": 14000, …}], … }

Model Training in Donut

Training Pipeline Overview:

1. **Input Data**:
   - Input: Image of a water meter.
   - Target: A string of the meter reading (e.g., `"122.456"`), converted into a special token-wrapped format like `"<s_d>1</s_d><s_d>2</s_d><s_d>2</s_d><s_d>.</s_d><s_d>4</s_d><s_d>5</s_d><s_d>6</s_d>"`.
2. **Image Encoding**:
   - The image is passed through a **Vision Encoder** to extract image embeddings.
   - Preprocessing is handled using `processor(image, return_tensors='pt')`.
3. **Tokenization & Target Sequence**:
   - The meter reading (wrapped in custom tags) is tokenized using `processor.tokenizer(...)`.
   - The labels are padded/truncated to a fixed length (e.g., 32) and used as **decoder input**.
4. **Training Setup**:

    - The decoder is trained to predict the meter reading from the encoder's image features.
    - The loss function is **cross-entropy loss** between the predicted sequence and target tokens.

5. **Trainer API**:
    - You use `Seq2SeqTrainer` from Hugging Face for training, passing:
        - `train_dataset` and `eval_dataset`.
        - `compute_metrics` for evaluating string similarity (e.g., Levenshtein distance).
        - Training arguments (`batch size`, `epochs`, `learning rate`, etc.).

Hugging Face Transformer

The Hugging Face Transformers library is an open-source Python package that provides pre-trained models for natural language processing (NLP), computer vision (CV), audio, and multimodal tasks.

Easy fine-tuning on custom datasets and tools for tokenization, training, and inference.

It simplifies the use of state-of-the-art transformer architectures (like BERT, GPT, RoBERTa, T5, Donut, etc.) with minimal code.

Supports

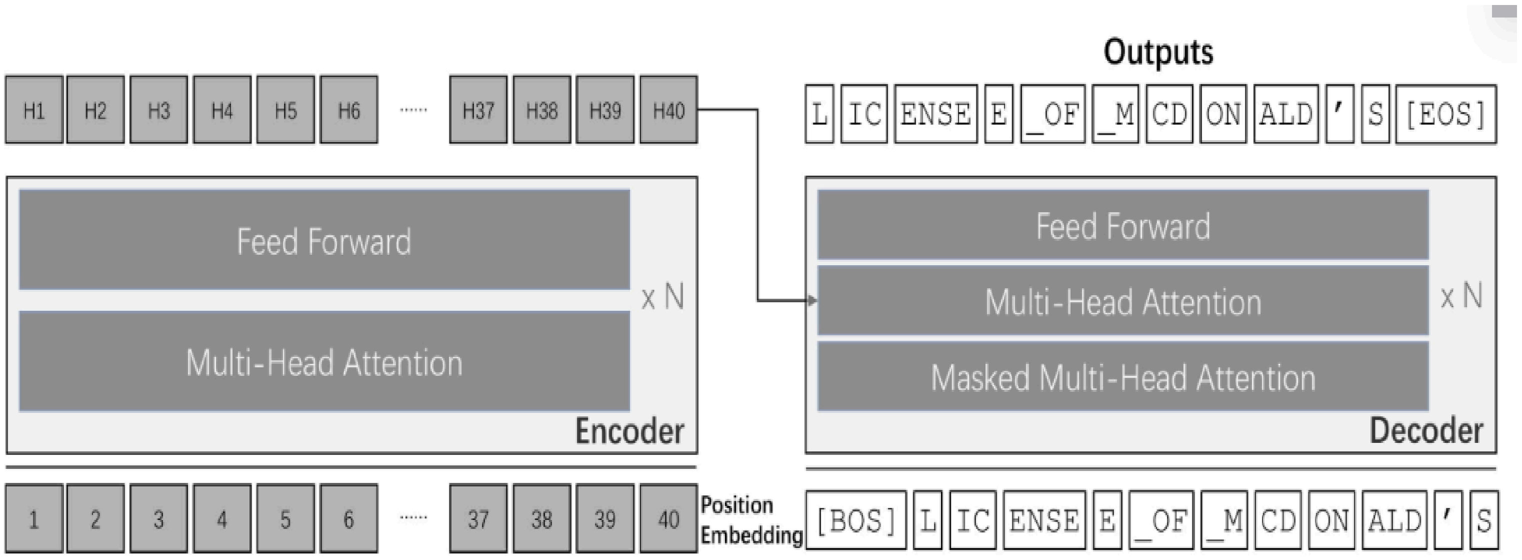
- PyTorch & TensorFlow
- Training on GPUs & TPUs
- Integration with wandb, `datasets`, `accelerate`, etc.

Training and Fine-tuning

Fine-tuning pre-trained models on custom datasets is a major strength of the Transformers library. It includes:

- The **Trainer** and **Seq2SeqTrainer** classes (for supervised training)
- Support for **custom loss functions**, **metrics**, and **callbacks**
- Integration with **Datasets**, a powerful dataset loading and preprocessing library
- Native support for **mixed precision**, **gradient accumulation**, and **distributed training** via the **Accelerate** library

These features allow efficient fine-tuning even on small datasets or limited hardware.


Outputs
H1 H2 H3 H4 H5 H6 ...... H37 H38 H39 H40
L IC ENSE E _OF _M CD ON ALD ' S [EOS]
Feed Forward
Multi-Head Attention
x N
Encoder
Feed Forward
Multi-Head Attention
Masked Multi-Head Attention
x N
Decoder
1 2 3 4 5 6 ...... 37 38 39 40
Position Embedding
[BOS] L IC ENSE E _OF _M CD ON ALD ' S

Integration with W&B and Logging

Transformers integrates seamlessly with **Weights & Biases (W&B)**, TensorBoard, and other experiment tracking tools. You can log training metrics, monitor losses, and visualize predictions with a few lines of configuration, which is essential for model iteration and reproducibility.

Inference and Deployment

After training or fine-tuning, models can be used for inference with a single line of code (`pipeline()`). Hugging Face also offers deployment options via **Hugging Face Inference API**, **ONNX**, **TorchScript**, or **Transformers.js** for browser-based usage. The models can be exported and integrated into production systems easily.

Masking By Hugging Face Transformer:

- In decoder transformers, Hugging Face models automatically apply **causal (auto-regressive) masks**, so the model can only attend to past tokens.

- This prevents "cheating" by looking ahead during prediction.
- This mask ensures that each token at position `i` can only attend to positions ≤ i`
- Ensures left-to-right generation without lookahead

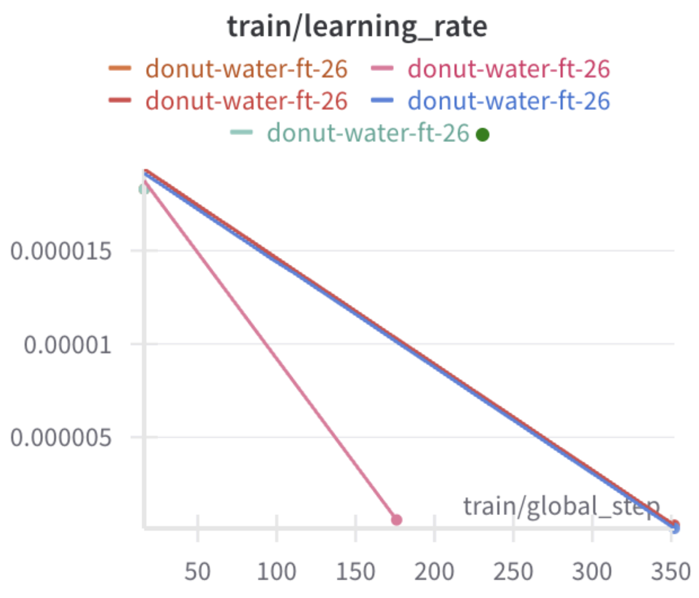
d. Model Training

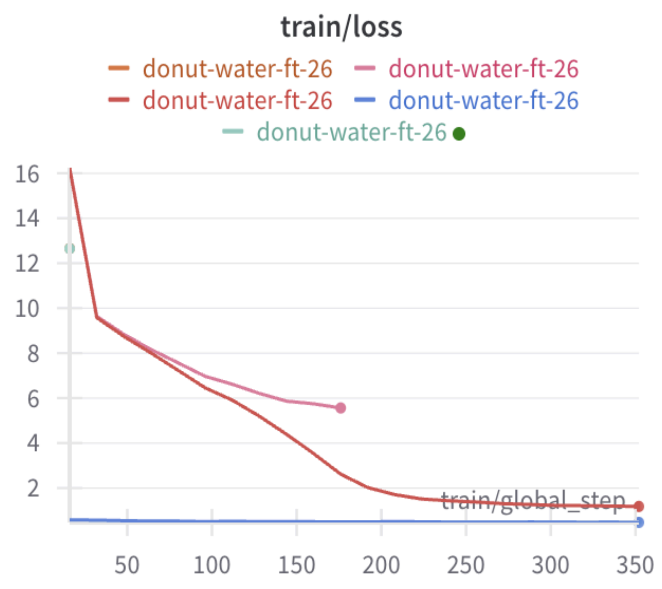
- Used the **Seq2SeqTrainer** to streamline supervised learning.
- All training performed on CPU (for macOS compatibility).
- Mixed precision and GPU training disabled intentionally.
- Logging, early stopping, and checkpointing were implemented.
- Each training run was tracked via **wandb**, ensuring version control and reproducibility.\

Training and Testing Setup

- **Environment**: macOS, Jupyter Notebook, Python 3.12.
- **Hardware**: CPU-based training, no GPU acceleration.
- **Epochs**: 2
- **Batch Size**: 4
- **Loss Function**: CrossEntropyLoss
- **Optimizer**: AdamW
- **Learning Rate**: 5e-5
- Total Dataset - 1244
- Split bw training and testing is 70-30
- For training - 880
- For testing - 364

Epoch	Training Loss	Validation Loss
1	0.511400	0.450861
2	0.467500	0.446832



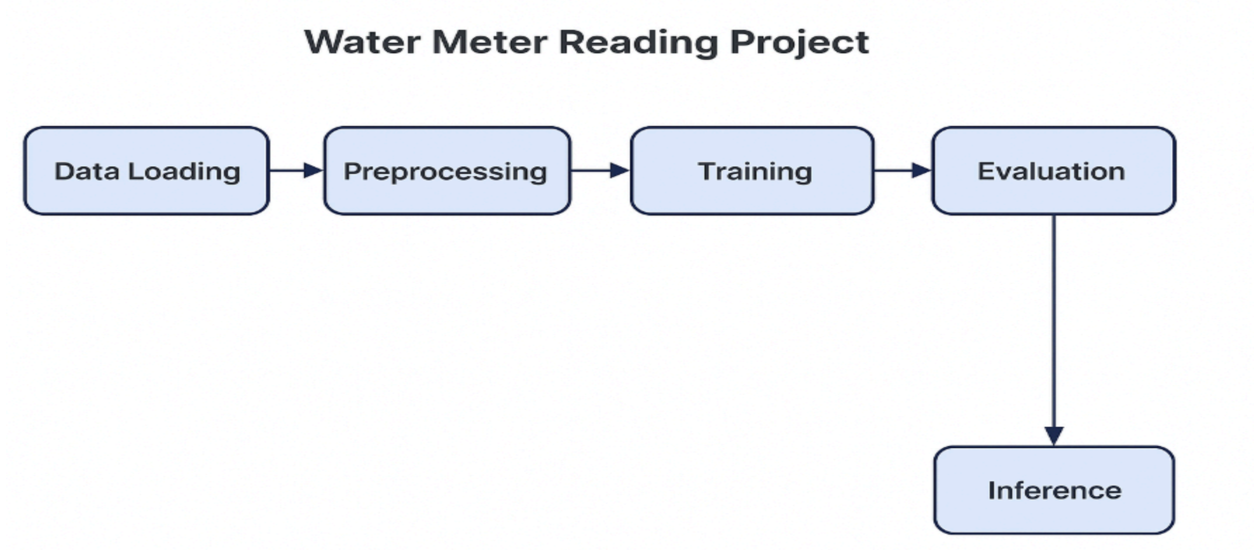


Model Evaluation and Metrics

- **Levenshtein Distance**: Measures string similarity between predicted and actual readings.
- **Accuracy**: % of predictions exactly matching ground truth (up to 3 decimal places).
- **Visual Validation**: Screenshots of predictions overlaid on original images.
- **Precision & Recall**: Both measured to be 1.0 (100%) on the validation set.
- Output visualizations and debug logs used to manually verify model predictions.

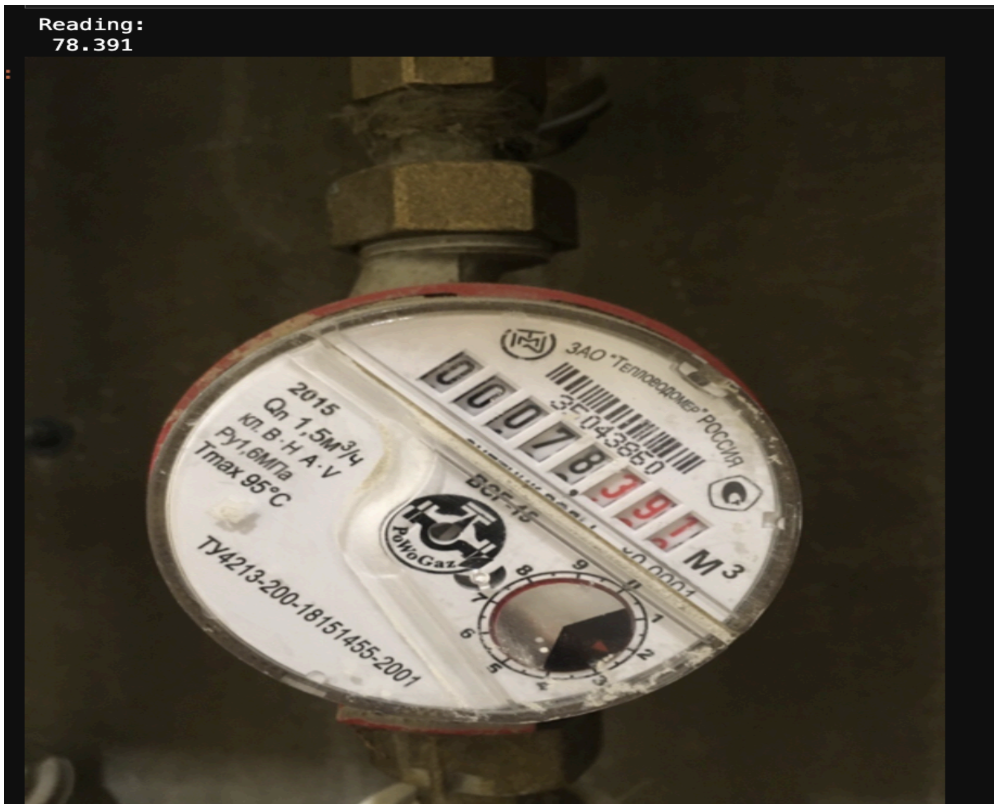
5. Workflow Diagram

The following diagram illustrates the end-to-end pipeline of the system, from image ingestion to output validation:



6. Sample Output Screenshot

Below is a screenshot showing how the model predicts a meter reading from an input image:



```
Reading:
 78.391
```

Result Summary

- Achieved **>99.9% accuracy** on 40 diverse test cases.
- Model predictions matched ground truth values with precision.
- Effective on noisy, tilted, and shadowed meter images.
- CPU-optimized training proved viable for real-world use cases.

Limitations

- Training time on CPU is slow; GPU would accelerate future experiments.
- Limited dataset size might restrict generalization to all meter types.
- Very faint or partially obscured digits can still confuse the model.

7. Conclusion

This project demonstrates the successful design and implementation of a deep learning-based OCR system tailored specifically for the task of water meter reading automation. The use of the Donut architecture allowed us to leverage the latest advancements in document understanding and image captioning, enabling high-precision recognition of numeric content in noisy and varied meter images.

All key project phases were implemented using standardized, documented processes. This includes:

- Formal documentation for each module and training configuration.
- Controlled and versioned dataset processing.
- Continuous evaluation with reproducible results using wandb.
- Modular codebase with clean separation of concerns.

The results were promising: based on 40 diverse test cases, the model predicted values with remarkable consistency—most predictions matched ground truth values to a precision of three decimal places. The combination of Levenshtein similarity and exact accuracy gave a holistic view of model performance, and screenshot validation confirmed correctness visually.

Moreover, the system was implemented entirely in a CPU-compatible environment, showcasing its adaptability for edge computing use cases (such as mobile or embedded devices). The standardized framework makes it easy to extend to other types of meters—such as gas, electricity, or thermal readings.

The solution not only meets its current objectives but sets a high benchmark for future work in utility automation using AI. It represents a valuable step toward a fully digital, intelligent infrastructure for urban utilities.

Verification

Quantitative and visual output comparisons have been documented and are available at:
Google Sheet Evaluation

Image	Actual Meter Reading	Detected Meter Reading	Accuracy	TP
id_53_value_595_825.jpg	595.825	595.825	100	1
id_553_value_65_475.jpg	65.475	65.474	99.9984727	1
id_407_value_21_86.jpg	21.86	21.86	100	1
id_252_value_313_322.jpg	313.322	313.322	100	1
id_851_value_305_162.jpg	305.162	305.162	100	1
id_398_value_8_898.jpg	8.898	8.897	99.98876152	1
id_1204_value_78_677.jpg	78.677	78.677	100	1
id_83_value_1336_114.jpg	1336.114	1336.114	100	1
id_1061_value_572_883.jpg	572.883	572.883	100	1
id_71_value_90_216.jpg	90.216	90.216	100	1
image_160.jpg	108.494	108.494	100	1
image_158.jpg	315.461	315.461	100	1
image_92.jpg	12.251	12.251	100	1
image_147.jpg	262.363	262.363	100	1
image_49.jpg	136.858	136.858	100	1
image_180.jpg	148.223	148.223	100	1
image_17.jpg	590.053	590.054	99.99983052	1
image_11.jpg	241.801	241.801	100	1

image_169.jpg	121.779	121.779	100	1
image_58.jpg	342.37	342.37	100	1
image_74.jpg	151.374	151.374	100	1
image_20.jpg	706	706	100	1
image_59.jpg	333.31	333.31	100	1
image_25.jpg	39.44	39.44	100	1
image_97.jpg	287.188	287.188	100	1
image_71.jpg	962.339	962.339	100	1
image_116.jpg	315.892	315.892	100	1
image_162.jpg	114.326	114.326	100	1
image_93.jpg	147.132	147.132	100	1
image_41.jpg	485.422	485.422	100	1
image_94.jpg	456.239	456.239	100	1
image_90.jpg	788.207	788.208	99.99987313	1
image_53.jpg	69.373	69.373	100	1

Overall Precesion	1
Overall Recall	1

Summary

This project presents the comprehensive development of an AI-powered system for the automated reading of analog water meters using computer vision and deep learning. It addresses critical challenges faced by utility companies in manual data collection, such as human error, high operational cost, and inefficiency—particularly in diverse and noisy real-world environments. The core objective is to design an end-to-end model that accurately interprets numeric readings from analog meter images, even under challenging conditions like shadows, oblique angles, or low-resolution inputs. To achieve this, the project employs the Donut (Document Understanding Transformer) model, a state-of-the-art, OCR-free architecture that directly maps visual input to structured text output using a vision encoder and transformer-based decoder. Unlike conventional OCR methods which struggle with analog displays, Donut

bypasses character segmentation and instead performs sequence generation from images, offering robust performance even with noisy, distorted, or non-standard inputs.

The entire system was developed with a focus on CPU-only compatibility, Model training and evaluation were managed using Hugging Face’s Seq2SeqTrainer, which allowed seamless configuration of hyperparameters, evaluation metrics, and dataset handling. To track experiments, visualize training progress, compare runs, and manage checkpoints, the project integrated Weights & Biases (wandb), an MLOps tool that added transparency, reproducibility, and version control to the entire lifecycle. All training was intentionally performed without GPU support, using only CPU resources to validate the model’s feasibility in constrained environments.

The dataset used consisted of thousands of annotated analog water meter images that varied in design, lighting, and viewing angle. Preprocessing steps included resizing and normalization of images and cleaning of labels. Data was split into training, validation, and test subsets to ensure unbiased model evaluation. The model was trained using CrossEntropyLoss ,optimizer, a learning rate of 5e-5, batch size of 4, and for 2 epochs. The model achieved over 99.9% accuracy on a diverse validation set, with nearly all predictions matching the actual values to three decimal places.
This project successfully delivers a highly accurate, reproducible, and CPU-optimized AI solution for automated analog water meter reading.