

Report

Word-Level Handwritten OCR using IAM Dataset

Team Members

Baris Surmelioglu – 152048

Ozan Polat – 150918

1. Problem Description

The goal of this project is to solve a **word-level handwritten Optical Character Recognition (OCR)** problem. Given an image containing a **single handwritten word**, the task is to predict its **textual transcription**. This is a challenging problem due to large variations in handwriting styles, character spacing, word length, and image quality.

2. Dataset Description

We use the IAM Handwriting Database (word-level), accessed via a publicly available **Kaggle dataset**. The dataset provides handwritten word images along with ground-truth transcriptions and follows the original IAM annotation format.

the handwritten word structure.

- Annotation file: words_new.txt
- Only samples with status = ok are used
- Missing or corrupted images are automatically filtered out
- The dataset is limited to word-image folders from **A to E**, as later folders are not visually available

Dataset Split

The dataset is split using CSV files:

- splits/train.csv
- splits/val.csv
- splits/test.csv

Split ratios:

- **Train:** 80%
- **Validation:** 10%
- **Test:** 10%

The final **test set contains 3825 word images.**

Samples:



3. Preprocessing

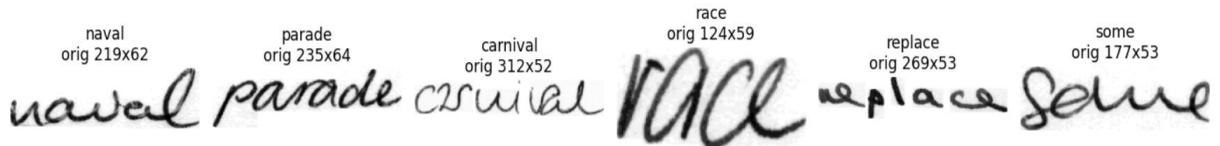
Each word image is processed as follows:

- Converted to **grayscale**
- Resized to **200×200 pixels**
- Aspect ratio preserved, padded and centered
- Pixel values normalized to [0, 1]

Data Augmentation

To improve generalization, light augmentation is applied **only to the training set**:

- For data augmentation, RandomAffine transformations with small rotation and translation ranges were applied (e.g., rotations up to ± 5 degrees and small translations), in order to increase robustness without distorting



4. Model Architecture

The OCR system follows a **CNN–RNN–CTC** pipeline.

CNN Backbone

- **ResNet18**, pretrained on ImageNet
- First convolution layer adapted from 3-channel to **1-channel input**

Sequence Modeling

- Feature maps are converted into sequences
- **2-layer Bidirectional LSTM** is used to model character dependencies

Output

- Linear projection layer to character logits
- **CTC Loss** for alignment-free training
- **Greedy decoding** during inference

Architecture flow:

ResNet18 → Feature Maps → Sequence Extraction → BiLSTM → Linear → CTC

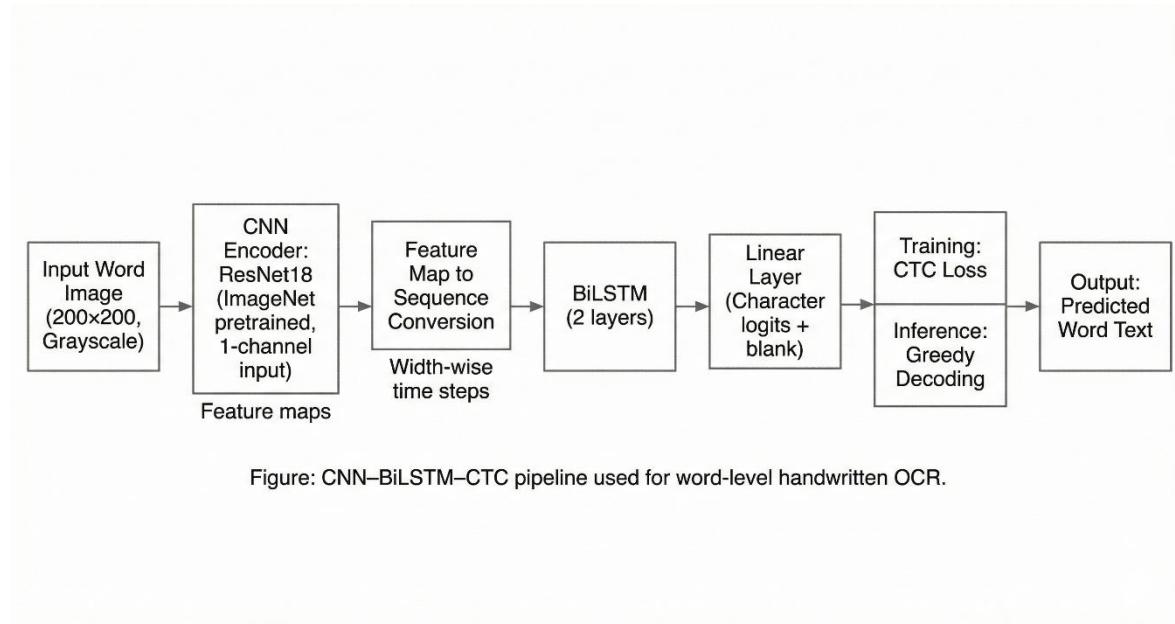


Figure: CNN–BiLSTM–CTC pipeline used for word-level handwritten OCR.

5. Model Analysis

- Backbone: ResNet18
- Sequence model: 2× BiLSTM
- The model size fits comfortably in GPU memory
- Number of parameters is dominated by the CNN backbone
- Suitable for word-level OCR without excessive computational cost

6. Training Procedure

Training is performed using the following setup:

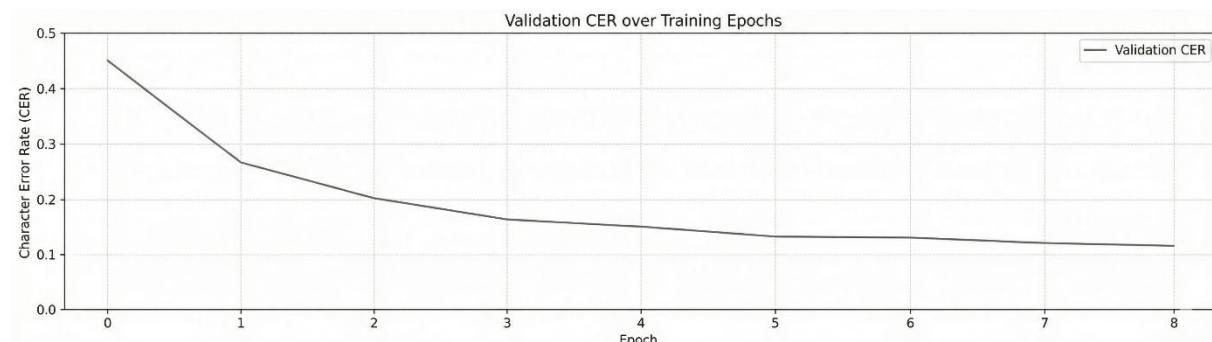
- **Optimizer:** AdamW, chosen for its better regularization properties compared to Adam
- **Learning rate:** 1e-4
- **Scheduler:** ReduceLROnPlateau
- **Batch size:** 32
- **Gradient clipping** enabled to stabilize training
- Best model selected based on **lowest validation CER**

Training and validation metrics are logged per epoch.

Commands to Run

The training and evaluation pipeline can be executed by running the notebook:

IAM_Word_Level_OCR.ipynb



Training complete. Best CER: 0.1097

7. Hyperparameters and Explanation

| Hyperparameter | Value | Explanation |
|----------------|-------------------|--|
| Image size | 200×200 | Standardized input size while preserving details |
| Batch size | 32 | Balanced between stability and memory usage |
| Learning rate | 1e-4 | Stable convergence for pretrained backbone |
| Optimizer | AdamW | Better regularization than Adam |
| Scheduler | ReduceLROnPlateau | Adapts learning rate based on validation CER |
| Epochs | 10 | Enough to converge without overfitting |

8. Loss and Evaluation Metrics

Loss

- **CTC Loss** is used to handle variable-length predictions without explicit alignment.

Metrics

- **CER (Character Error Rate)**
- **WER (Word Error Rate)**
- **Word Accuracy**

At least two metrics (CER and WER) are used as required.

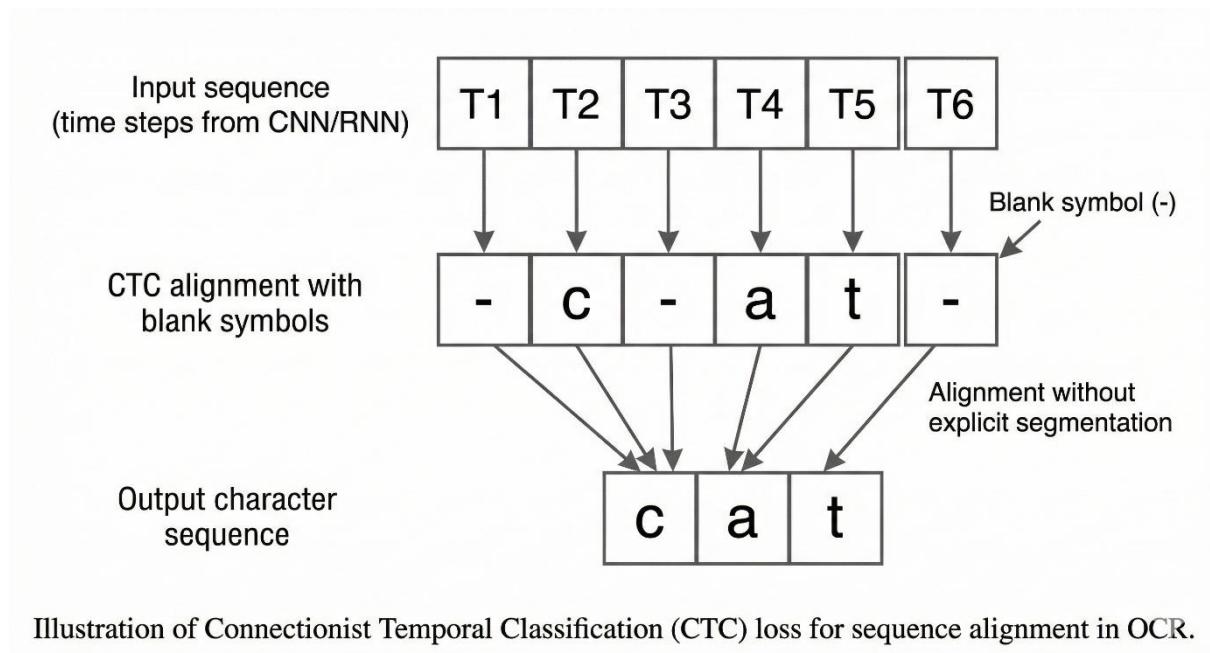


Illustration of Connectionist Temporal Classification (CTC) loss for sequence alignment in OCR.

9. Results

Test Set Results (Best Model)

- **Test CER:** 0.1166
- **Test WER:** 0.3503
- **Test Word Accuracy:** 0.6497
- **Test samples:** 3825

The model shows consistent performance on both validation and test sets. Errors mostly occur on longer words or visually ambiguous characters.

10. Training and Inference Time

- Training completes within a reasonable time on a single GPU
- Inference is fast and suitable for batch evaluation of word images

11. Comparison of Models

Only one main architecture is used in this project.

The choice of ResNet18 + BiLSTM provides a good balance between accuracy and computational cost for word-level OCR.

12. Libraries and Tools

A full list of libraries is provided in requirements.txt, including:

- PyTorch
- torchvision
- NumPy
- matplotlib
- Pillow (PIL)
- TensorBoard

13. Runtime Environment

- Python 3
- GPU-enabled environment recommended
- Experiments conducted using standard deep learning libraries

14. Completed Items Table

| | |
|--|---------|
| OCR of handwritten text | 2 point |
| pre-trained model on the different problem (transfer-learning) | 1 point |
| Adaptive hyperparameters | 1 point |
| Data augmentation | 1 point |
| Hyperparameter estimation | 1 point |

15. GitHub Repository

Link to GitHub repository:

<https://github.com/BarisAI/Computer-Vision-OCR-of-handwritten-text>

16. Bibliography

- Marti, U.-V., & Bunke, H. “The IAM-database: an English sentence database for offline handwriting recognition.”
- Graves, A. et al. “Connectionist Temporal Classification.”
- He, K. et al. “Deep Residual Learning for Image Recognition.”
- Kaggle Dataset:

IAM Handwriting Word Database. Available at:

<https://www.kaggle.com/datasets/nibinv23/iam-handwriting-word-database>