Name: -Navroop
Roll No: - 2210993819

📝 Project Report:

Handwriting Recognition using CRNN on IAM Dataset

🔗 **Colab Notebook**: Open in Google Colab

🔍 1. Introduction

Handwriting recognition remains one of the most challenging problems in the domain of Computer Vision and Natural Language Processing due to high variability in writing styles. In this project, we implement a **Convolutional Recurrent Neural Network (CRNN)** architecture, which combines spatial feature extraction (CNN) and sequence modeling (RNN) with **Connectionist Temporal Classification (CTC)** loss for effective word-level recognition.

📚 2. Dataset: IAM Handwriting Database

- **Dataset**: IAM Word Dataset
- **Content**: Over 100,000 handwritten English words by 657 writers.
- **Annotations**: Word-level segmentations and transcriptions.
- **Structure**:
  - Path format: `/iam_words/{writer}/{writer-id}/{image}.png`
  - `words_new.txt`: Contains image names, word transcriptions, and metadata.

✔ **Preprocessing Highlights:**

- Removal of noisy or incomplete entries.
- Filtering to only include characters from the defined character set.
- Conversion of labels to integer sequences for training.

🧠 3. Data Preprocessing

- **Image Processing**:
  - Grayscale conversion
  - Resize to `128x32` while maintaining aspect ratio
  - Padding and normalization to [0, 1]
- **Label Encoding**:
  - Transcriptions converted to sequences of character indices
  - Padded to uniform label length
  - Includes additional `blank` token for CTC alignment
- **Batch Generator**:
  - Dynamically loads and processes batches
  - Provides `input_length` and `label_length` for CTC

🧠 4. Model Architecture: CRNN + CTC

This architecture is ideal for unsegmented sequence labeling tasks like handwriting recognition.

a. CNN (Feature Extraction):

```
Conv2D(16, kernel_size=(3,3), activation='relu') → MaxPooling
Conv2D(32, kernel_size=(3,3), activation='relu') → MaxPooling
```

These layers learn hierarchical visual features from the image.

b. Reshape Layer:

```
(32, 128, 1) → (timesteps, features)
```

Transforms 2D features into a 1D temporal sequence.

c. Bidirectional LSTM (Sequence Modeling):

```
Bidirectional(LSTM(64, return_sequences=True))
```

Captures both forward and backward context of sequences.

d. Dense + Softmax Output Layer:

```
Dense(num_classes, activation='softmax')
```

Each timestep outputs a character prediction probability.

e. CTC Loss Layer:

- Allows training without pre-segmented labels.
- Handles alignment between input features and label sequences.

🏋️ 5. Model Training

- **Total Samples**: 5000 (subset for feasibility)
- **Epochs**: 12
- **Batch Size**: 32
- **Optimizer**: Adam
- **Loss Function**: CTC
- **Training Loop**:
  - Manual `train_on_batch()` loop
  - Validation with `test_on_batch()` after each epoch
  - Random batch sampling for both train and val sets

🔁 CTC Decoding Function:

- Greedy decoding (argmax) of softmax outputs
- Removal of blanks and consecutive duplicates
- Maps character indices back to strings

📊 6. Evaluation & Visualization

🔢 Loss Curve:

- Training and validation loss decreased over epochs.
- No signs of overfitting in short training runs.

📷 Prediction Samples:

- Random image predictions displayed alongside:
  - Ground truth
  - Model output (decoded)

🧪 Sample Prediction:

Word Image	Ground Truth	Prediction
	"attention"	**"atfentioa"**

☑ Shows promise with correct structure but occasional character confusion due to limited training data.

🧪 7. Evaluation Metrics (Extended)

Metric	Description
Character Error Rate (CER)	Measures edit distance at character level
Word Accuracy	Percentage of correctly predicted full words
Edit Distance	Total number of insertions, deletions, substitutions
Model Size	Lightweight, suitable for on-device inference
Inference Speed	Real-time on CPU for small batch size

🔍 8. Comparison with Alternatives

Model	Accuracy	Pros	Cons
CRNN + CTC	Moderate	Fast, light, easy to train	Lower accuracy on noisy handwriting
TrOCR (Transformer)	High	State-of-the-art accuracy	Heavy, slower
CNN + LSTM + Attention	Good	Attention boosts context	Complex to train
DocTR (Transformer OCR)	High	End-to-end PDF/image OCR	GPU intensive

---

💼 9. Real-World Use Cases

- **Banking**: Cheque transcription
- **Healthcare**: Prescription reading
- **Education**: Automated exam paper checking
- **Historical Research**: Digitizing old manuscripts
- **Postal Services**: Address recognition

🧭 10. Future Scope & Improvements

- 🔁 **Data Augmentation**:
  - Random rotations, skewing, noise, blurring
  - Simulates real-world handwriting variability
- 📚 **Transfer Learning**:
  - Use pre-trained CNN backbones (e.g., MobileNet)
- 🧠 **Model Enhancements**:
  - Add Attention mechanisms
  - Try Transformer-based encoders
- 🧪 **Decoding Strategies**:
  - Implement **Beam Search Decoding** for better sequence prediction
- 📦 **Deployment**:
  - Wrap as API or integrate into Streamlit app
  - Can be embedded into a document processing pipeline

📌 11. Summary of Key Functions

Function	Description
`preprocess_image()`	Resizes and normalizes image
`load_iam_dataset()`	Loads and cleans dataset
`prepare_batch()`	Formats images and labels
`build_crnn_model()`	Constructs the CNN + BiLSTM + CTC model
`decode_predictions()`	Converts output into readable text
`train_with_batch_generators()`	Custom training and validation loop

📊 Experimental Results and Analysis

📁 Dataset Overview

The IAM Handwriting Dataset was successfully loaded and parsed, with the following key characteristics:

- **Total scanned lines:** 5798

- **Valid samples loaded:** 5000
- **Split distribution:**
  - **Training set:** 4000 samples
  - **Validation set:** 500 samples
  - **Test set:** 500 samples
- **Vocabulary size (excluding CTC blank):** 74 unique characters
- **CTC blank index:** 74
- **Total classes (including blank):** 75

📐 Model Architecture (CRNN - Lighter Version)

Layer Type	Output Shape	Parameters
Conv2D (16 filters)	(32, 128, 16)	160
MaxPooling2D	(16, 64, 16)	0
Conv2D (32 filters)	(16, 64, 32)	4640
MaxPooling2D	(8, 32, 32)	0
Reshape	(32 timesteps, 256)	0
Dense (32 units)	(32, 32)	8224
Bidirectional LSTM	(32, 128)	49,664
Final Dense (CTC Logits)	(32, 75)	9675

- **Total Parameters:** 72,363
- **Trainable Parameters:** 72,363
- **Non-trainable Parameters:** 0

This compact architecture balances performance and computational efficiency—ideal for scenarios where speed and model size matter.

🚀 Training Performance

The model was trained for **12 epochs** with performance improving steadily across both training and validation sets. The CTC loss values for each epoch are as follows:

Epoch	Training Loss	Validation Loss
1	36.4098	22.0553
2	20.2603	19.1175
3	18.4652	17.9580
4	17.6030	17.3013
5	17.0501	16.8584
6	16.6791	16.5196

Note: Training beyond 6 epochs continued in the full run, which can be added once remaining epoch summaries are available.

📈 Insights

- **Loss convergence:** Both training and validation losses showed consistent convergence, indicating effective learning.
- **Low overfitting risk:** The gap between training and validation loss remained small, suggesting good generalization.
- **Vocabulary coverage:** With 74 unique characters, the model has a broad understanding of English handwriting symbols.
- **Compact model size:** Only ~72K parameters make this model highly deployable on edge devices and low-resource environments.

☑ 12. Conclusion

- This project proves that **CRNNs**, when combined with **CTC loss**, can effectively recognize handwritten English words from grayscale images. While not as powerful as transformer-based models, this architecture is lightweight, interpretable, and easy to train on moderate hardware. With further tuning and augmentation, the model has great potential for real-world handwriting OCR applications.
