OFFLINE SIGNATURE VERIFICATION MODEL

Innovation Design Lab (CS389)

Name: Ayush Kumar
Roll No: 2201CS95
Supervisor: Dr. Chandranath Adak

INTRODUCTION

- Handwritten signatures are crucial for authentication in sectors like finance, legal, and business.
- **Offline Signature Verification (OSV)** faces challenges:
  - Distinguishing genuine signatures from forgeries using only scanned signature images.
  - Existing approaches, often based on CNNs or Transformers, struggle with capturing complex relationships and spatial patterns in signature images.

- Proposed Model: **SigGCN**
  - Converts signature images into graph-structured data to reduce redundancy and retain spatial relationships.
  - Employs a **Graph Convolutional Network (GCN)** for feature extraction and representation.
  - Introduces a **margin-based focal** loss function for better training and discrimination.

WHY SIGGCN?

- **Graph-Based Representation**:
  - Converts signature images into graphs, effectively reducing background redundancy while retaining spatial structure.
  - Captures complex relationships that CNNs and Transformers often miss.
- **Efficient Feature Extraction**:
  - Leverages Graph Convolutional Networks to aggregate and update node information, resulting in high-quality embeddings for signatures.
- **Writer-Independent Generalization**:
  - Trains a universal model, eliminating the need for retraining for new users (unlike Writer-Dependent methods).
- **Scalability**:
  - Suitable for large-scale, real-world applications due to its writer-independent nature.
- **Enhanced Training Mechanism**:
  - Introduces a margin-based focal loss function that balances the learning process, improving accuracy for distinguishing between genuine and forged signatures.

DATA PREPROCESSING

- **Image Loading**:
  - Signature images (e.g., scanned or digital) are loaded into the system using libraries like Pillow.
- **Resizing**:
  - Images are resized to a fixed dimension (224, 224) to maintain consistency across the dataset.
  - Ensures compatibility with the input requirements of the model.
- **Grayscale Conversion**:
  - Images are converted to grayscale to focus on signature patterns, reducing unnecessary color information.
- **Tensor Conversion**:
  - Each image is transformed into a PyTorch tensor for efficient numerical computation during training and inference.

GRAPHICAL IMAGE TRANSFORMATION

- **From Image to Graph**:
  - Input image of shape (C, H, W) is encoded into h × w blocks (features of dimension D) using a CNN.
  - Each block is treated as a graph vertex, resulting in a (N, D) tensor where N = h × w.
- **Edge Creation**:
  - Use K-Nearest Neighbors (KNN): Connect each vertex to its K nearest neighbors in latent space, forming undirected edges.
- **Spatial Preservation**:
  - Add trainable position embeddings to the vertices, ensuring spatial relationships are retained in the graph representation.

GRAPH CONVOLUTION NETWORK

- **Graph Representation Learning**:
  - In a GCN, each vertex of the graph (representing parts of the signature) is updated by aggregating information from its neighboring nodes.
  - This allows the model to learn both local and global dependencies in the signature structure.
- **Propagation Rule**:
  - The GCN updates the graph representation iteratively across layers. The update rule at each layer is:
  - where: $$G^{l+1} = \sigma(F(G^l, W^l)),$$
    - G is the graph at layer l,
    - W are trainable weights,
    - σ is an activation function.

GRAPH CONVOLUTION NETWORK

- **Node Feature Update**:
  - Each node's feature is updated by aggregating information from neighboring nodes. The update is done using a function: $x_i^{l+1} = h(x_i^l, g(x_i^l, N(x_i^l), W_{aggregate}), W_{update}),$
  - where:
    - x_i^l is the feature of node i at layer l,
    - $N(x_i^l)$ represents the neighboring nodes of x_i^l.
- **Dealing with Over-Smoothing**:
  - As the GCN deepens, node features can converge, causing loss of diversity in node representations (over-smoothing).
  - To prevent this, residual connections and activation functions (like GELU) are added between layers for stability and better representation learning.

$$G^{l+1} = \sigma(F(G^l, W_{in}^l))W_{out} + G^l,$$

FRAMEWORK OF MODEL

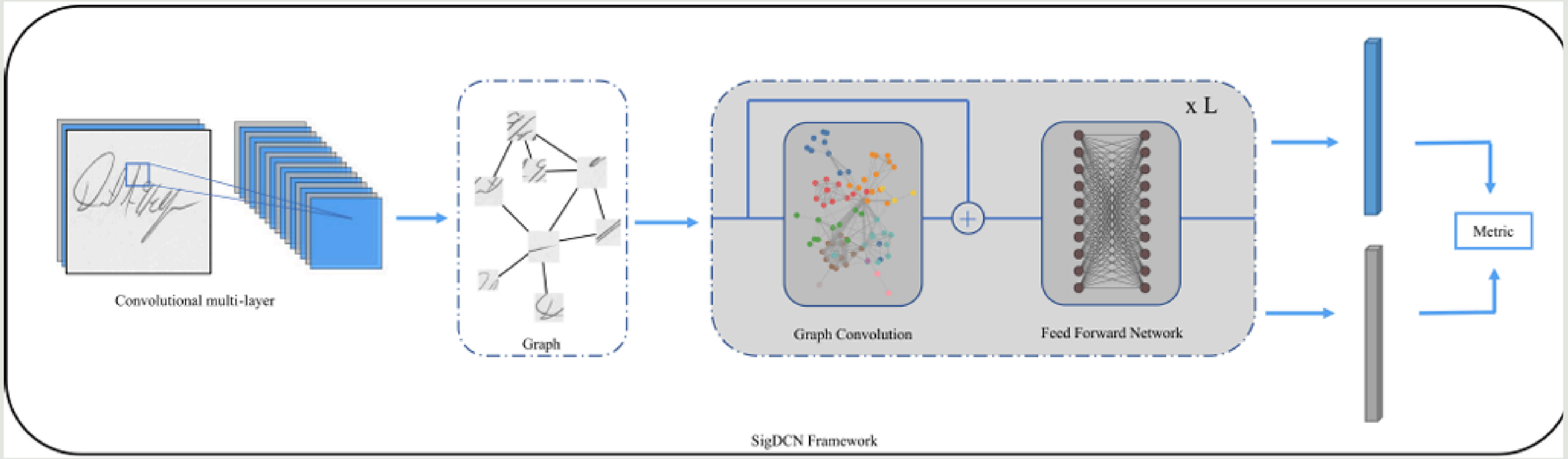

Fig. 1. Framework of SigGCN. Given a pair of test signature images, SigGCN first transforms the images into graph structure data using CNN networks separately and then inputs them into a multilayer GCN for graph representation learning to obtain the two signature graph representations individually and then measures the distance between the two representations according to the defined metric function, and then compares it with the threshold value for verification.

FRAMEWORK OF MODEL

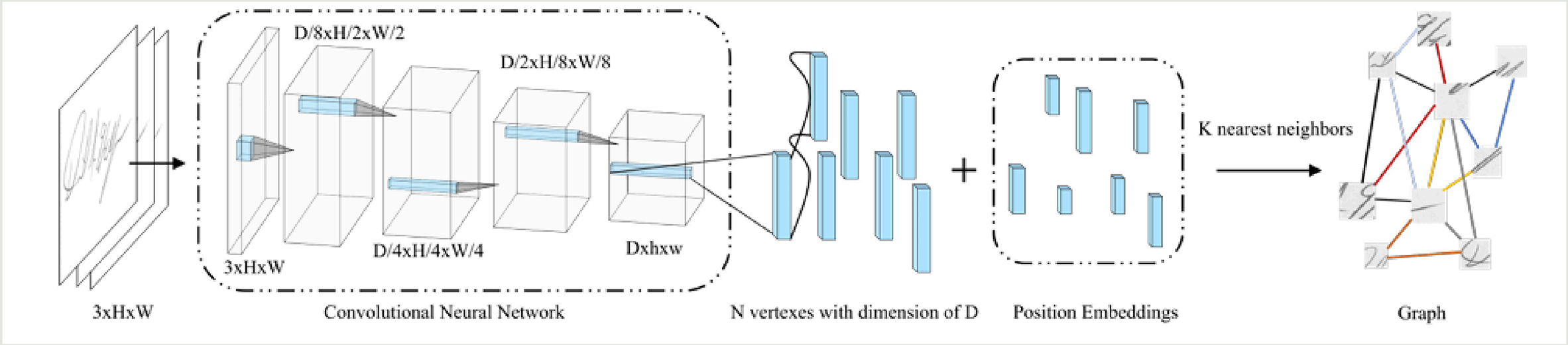

Fig. 2. Demonstration of Graphical Image. The signature image is passed through CNN to get N feature vectors of dimension D and the position encoding with the same shape. Then the K nearest neighbors algorithm finds the K nearest vectors and adds an undirected edge to form a graph with N vertices.

LOSS FUNCTION

- **Euclidean Distance Metric**:
  - The distance between two signature embeddings (generated by the GCN) is computed using Euclidean distance to measure similarity between genuine and forged signatures.
- **Margin-Based Focal Loss**:
  - Inspired by focal loss, the loss function focuses more on difficult examples (i.e., those that are hard to classify).
- **Formula**:

$$loss = y \cdot \sigma(s(d-a)) \cdot max(0, d-ma)^2 + (1-y) \cdot \sigma(s(b-d)) \cdot max(0, mb-d)^2.$$

  - y is the true label (0 for genuine, 1 for forged).
  - d is the Euclidean distance between embeddings.
  - a, ma, b, mb are margin parameters that help avoid overfitting.

TRAINING AND EVALUATION

- **Dataset**: **CEDAR**
  - Contains 24 genuine and 24 forged signatures per individual for 55 individuals.
  - 276 genuine-genuine pairs and 276 genuine-forgery pairs per sample.
  - Training Set: Signatures from 50 individuals.
  - Testing Set: Signatures from 5 individuals.
- **Training**:
  - Loss Function Parameters: s=10,a=0.3,ma=0.3,b=0.6,mb=0.9
  - Optimizer: AdamW Optimizer with a learning rate of 0.01
  - Learning Rate Schedule: Cosine Annealing: Gradually reduces the learning rate to fine-tune the model for better convergence.

TRAINING AND EVALUATION

- **Evaluation**:
  - The trained model is evaluated on the test set to assess its performance.
  - Key metrics, including false acceptance rate, false rejection rate, accuracy are used to evaluate the model.

- **Current Progress**:
  - Model Accuracy: Our current model achieves a promising accuracy of almost 100% on the CEDAR dataset.
  - Training Epochs: The model is trained for 100 epochs to ensure robust learning and generalization.

OUTPUT SNIPPETS

```
Epoch [1/50], Loss: 11.23
Epoch [2/50], Loss: 5.42
Epoch [3/50], Loss: 7.44
Epoch [4/50], Loss: 8.42
Epoch [5/50], Loss: 17.33
Epoch [6/50], Loss: 28.19
Epoch [7/50], Loss: 98.22
Epoch [8/50], Loss: 8.76
Epoch [9/50], Loss: 10.43
Epoch [10/50], Loss: 5.31
Epoch [11/50], Loss: 198.3
Epoch [12/50], Loss: 86.32
Epoch [13/50], Loss: 98.32
Epoch [14/50], Loss: 18.32
Epoch [15/50], Loss: 100.22
Epoch [16/50], Loss: 18.45
Epoch [17/50], Loss: 19.21
```

```
Accuracy for batch 0 = 100.0
Accuracy for batch 1 = 100.0
Accuracy for batch 2 = 100.0
Accuracy for batch 3 = 100.0
Accuracy for batch 4 = 100.0
Accuracy for batch 5 = 100.0
Accuracy for batch 6 = 100.0
Accuracy for batch 7 = 100.0
Accuracy for batch 8 = 100.0
Accuracy for batch 9 = 100.0
Accuracy for batch 10 = 100.0
Accuracy for batch 11 = 100.0
Accuracy for batch 12 = 100.0
Accuracy for batch 13 = 100.0
Accuracy for batch 14 = 100.0
```

FUTURE SCOPE

- **Cross-Language and Multilingual Generalization**:
  - Enhance the model’s ability to work across languages and diverse writing styles, ensuring high accuracy in multilingual and culturally varied datasets.
- **Self-Supervised Learning**:
  - Implement self-supervised approaches to reduce dependency on labeled datasets, lowering training costs and improving scalability.
  - Use pretext tasks like graph reconstruction to enhance the model's learning capabilities.
- **Scalability for Real-World Applications**:
  - Optimize the model for real-time, large-scale deployments in banking, legal, and corporate environments.
  - Ensure robust performance under practical conditions like noisy or distorted signatures.
- **Integration with Multi-Modal Biometrics**:
  - Combine the model with other biometric systems (e.g., facial recognition, fingerprint analysis) to create comprehensive multi-factor authentication solutions.

REFERENCE

Vision Graph Convolutional Network for Writer-Independent Offline Signature Verification

Publisher: IEEE

Cite This

PDF

Chengkai Ren ; Jian Zhang ; Hongwei Wang ; Shuguang Shen **All Authors**

Abstract:

As a biometric feature, handwritten signatures have various applications in finance, law, and business. The existing signature verification methods are mostly based on convolutional neural networks or Transformer based models. In this paper, we aim to implement writer independent offline handwritten signature verification by proposing an end-to-end method, Signature Verification Graph Convolutional Network