Signature Detection using CNNs

Abdullah Masood Mughal.

FAST NUCES Islamabad Pakistan

i210822@nu.edu.pk

***Abstract*—This paper presents the development of a Convolutional Neural Network (CNN) model aimed at classifying and identifying signatures from various individuals. The study utilized a dataset consisting of 16 images, each featuring 12 rows containing 4 signatures per row, leading to a comprehensive collection of signature samples. The dataset was meticulously organized into training and testing sets, with labels formatted to indicate the specific image and row of origin. The preprocessing steps included converting images to grayscale, applying binary thresholding, and extracting signature contours to standardize input size at 128x128 pixels for the CNN. The architecture of the CNN incorporated multiple layers, including convolutional, pooling, and dense layers, and was trained with categorical cross-entropy loss using the Adam optimizer. Evaluation metrics indicated a training accuracy of approximately 75.12**

I. Introduction

In this research, we developed a Convolutional Neural Network (CNN) model to classify and identify signatures from different individuals using a structured image processing and training approach. The dataset comprised 16 images, each containing 12 rows, with 4 signatures per row, amounting to a diverse set of signature samples. These signatures were organized into training and testing sets, with each labeled in the format *imagenumber_rownumber*, indicating the specific image and row from which it was extracted. The model was trained to recognize these labeled signatures, and its performance was evaluated through various metrics, offering insights into its accuracy and effectiveness. This approach showcases the potential of deep learning for automating signature recognition and classification tasks.

II. Methodology

The dataset used in this project comprised 16 images, each containing 12 rows of signatures, with 4 signatures per row. To facilitate training and testing for signature classification, the dataset was preprocessed to extract individual signature samples from the images.

A. Preprocessing

A grayscale image was processed by first applying binary thresholding and morphological operations to enhance the signature contours. Each image was divided into rows and columns to isolate the signature boxes, and the largest contour within each box was extracted as the signature. These signatures were resized to 128x128 pixels to ensure uniform input for the CNN model. The processed signatures were then organized into a structured folder system, with separate directories for training and testing sets. Each signature was labeled in the format `image_number_row_number`, indicating the image and row of origin. This structured approach maintained consistency across the dataset and ensured that each signature could be easily tracked back to its source image and row.

B. Model Architecture

The CNN model consisted of multiple layers, including convolutional, pooling, and dense layers. The first layer used 32 filters with a 3×3 kernel for feature extraction, followed by a max-pooling layer and dropout for regularization. This pattern was repeated with 64 filters in the second convolutional layer. The model was flattened and passed through a fully connected (dense) layer with 128 units and ReLU activation before the final softmax layer. The output layer's size matched the number of unique label classes, making it capable of distinguishing between different individuals' signatures. The model was compiled using the Adam optimizer and categorical cross-entropy loss, suitable for multi-class classification tasks.

III. Prepare Your Paper Before Styling

In this section, we present some key points for understanding and applying the CNN Model for classification.

A. Abbreviations and Acronyms

- **CNN**: Convolutional Neural Network
- **ReLU**: Rectified Linear Unit
- **SGD**: Stochastic Gradient Descent
- **FC**: Fully Connected
- **FNN**: Feedforward Neural Network
- **Loss**: Loss Function
- **Epoch**: Full Pass Through the Training Dataset
- **Batch**: Subset of Training Samples

B. CNN Equations

- **Convolution Operation**:

$$(I * K)_{i,j} = \sum_m \sum_n I_{i+m,j+n} K_{m,n}$$

Applies a filter K over the input image I to produce feature maps.

- **Activation Function (ReLU)**:

$$A(x) = \max(0, x)$$

Introduces non-linearity by outputting the maximum of zero and the input x.

- **Pooling Operation (Max Pooling)**:

$$P_{i,j} = \max_{m,n} A_{2m+i,2n+j}$$

Reduces dimensionality by taking the maximum value from each pooling window.

- **Flattening**:

$$F = \text{flatten}(A)$$

Converts the multi-dimensional output of the last convolutional layer into a one-dimensional vector.

- **Fully Connected Layer**:

$$Z = W \cdot F + b$$

Computes the weighted sum of inputs F with weights W and bias b.

- **Softmax Function**:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Converts logits z into probabilities for each class.

C. Authors and Affiliations

All people involved in the project are acknowledged appropriately.

IV. Results

The performance of the CNN model was evaluated using training and validation metrics over multiple epochs. As depicted in Figure 1, the training loss consistently decreased while the training accuracy improved over the epochs. The training loss started at approximately 2.80 and converged to about 2.74, indicating effective learning. In contrast, the test loss peaked at around 5.20, suggesting challenges in generalization on unseen data. The model achieved a training accuracy of approximately 75.12% but struggled on the test set with an accuracy of only 9.94% (see Figure 2).

Further evaluation metrics revealed a precision of 9%, recall of 12%, and an F1 Score of 10% (refer to Figure 3), indicating that the model's performance was suboptimal on the test dataset, which consisted of 179 samples across 185 classes. These results highlight the need for additional strategies, such as data augmentation or hyperparameter tuning, to enhance the model's robustness and accuracy in classifying signatures accurately.

V. Training and Validation Metrics

- **Train Loss:** 2.80
- **Train Accuracy:** 75.12%
- **Test Loss:** 5.20
- **Test Accuracy:** 9.94%

The training and validation graphs illustrate these findings, with loss decreasing over epochs for the training dataset and a more erratic behavior for the test dataset, alongside the accuracy trends for both datasets.

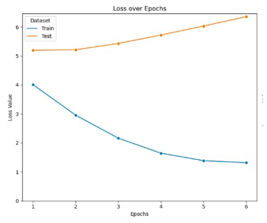


Fig. 1. Loss vs Epoch for Training and Test Set

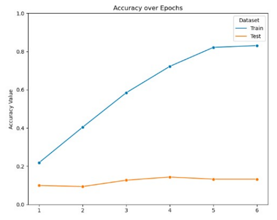
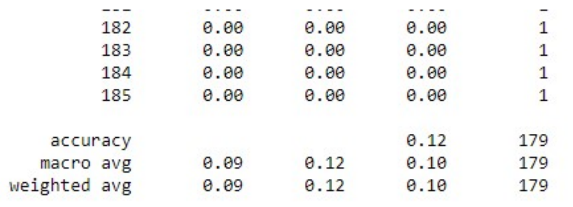


Fig. 2. Accuracy vs Epoch for Training and Test Set



```
         182       0.00      0.00      0.00         1
         183       0.00      0.00      0.00         1
         184       0.00      0.00      0.00         1
         185       0.00      0.00      0.00         1

    accuracy                           0.12       179
   macro avg       0.09      0.12      0.10       179
weighted avg       0.09      0.12      0.10       179
```

Fig. 3. Precision, Recall, F1 and Accuracy

Acknowledgment

The major headings are organized to offer clear insights into the construction of the CNN model and its training.

article hyperref

References

[1] Tiwari, A. (2020). *Classification of Signature and Text Images Using CNN and Deploying the Model on Google Cloud ML*. Towards Data Science. [Online]. Available: https://towardsdatascience.com/classification-of-signature-and-text-images-using-cnn-and-deploying-the-model-on-google-cloud-ml-30bf6f4e3207 [Accessed: 29-Sep-2024].

[2] Khan, A. (2021). *Image Classification Using CNN - 94% Accuracy*. Kaggle. [Online]. Available: https://www.kaggle.com/code/arbazkhan971/image-classification-using-cnn-94-accuracy [Accessed: 29-Sep-2024].

[3] Huang, M. (2020). *Image Recognition with CNNs: Improving Accuracy and Efficiency*. Medium. [Online]. Available: https://medium.com/@mitchhuang777/image-recognition-with-cnns-improving-accuracy-and-efficiency-dd347b636e0c [Accessed: 29-Sep-2024].