Recognition of Sign Language Digits using Convolutional and Fully Connected Neural Networks

João Nuno Valente
Dept. of Mechanical Engineering
University of Aveiro
Aveiro, Portugal
jnvalente@ua.pt

***Abstract*—This report details the implementation and evaluation of two neural network architectures, Convolutional Neural Networks (CNNs) and Fully Connected Neural Networks (FCNNs), for the recognition of sign language digits (0-9). The project used a dataset of 2062 images, preprocessed by resizing to 64 × 64 pixels and converting to grayscale, then split into training and testing sets. The CNN model demonstrated superior performance, achieving approximately 92.5% accuracy, compared to the FCNN's 87.4%. A visualization tool was developed to illustrate the transformation of input images through the CNN layers, providing insights into feature extraction.**

***Index Terms*—Sign Language Recognition, CNN, FCNN, Deep Learning, Image Classification.**

I. Introduction

Sign languages are essential for communication among individuals who are deaf or hard of hearing. Automating the recognition of sign language gestures can significantly increase accessibility and communication. This project focuses on the recognition of sign language digits (0-9) using deep learning techniques. The goal is to develop and compare two distinct neural network architectures, Convolutional Neural Networks (CNNs) and Fully Connected Neural Networks (FCNNs), to assess their effectiveness in this task. Additionally, a visualization tool was developed to provide insights into the internal workings of the CNN model, particularly the feature extraction process. All code used in this project can be found in the GitHub repository [1].

II. Dataset

The dataset used in this project comprises 2062 images of hand gestures, each representing a digit from 0 to 9, as sourced from [2]. These images underwent preprocessing to prepare them for the neural network models. Initially, each image was 100×100 pixels. To reduce computational demands and increase training efficiency, the images were resized to 64×64 pixels. Additionally, they were converted to grayscale, simplifying the input data by removing color information.

A. Label Correction

It was observed that the original dataset had label inconsistencies. These were manually corrected to ensure accurate training and evaluation of the models.

III. Methodology

A. Convolutional Neural Network (CNN)

CNNs are particularly effective for image classification tasks due to their ability to automatically learn spatial hierarchies of features. They use convolutional layers to extract local features, pooling layers to reduce dimensionality, and fully connected layers for classification.

Architecture

The CNN architecture includes:

- Input layer: 64×64 grayscale images.
- Convolutional layers: Three convolutional layers with 32, 64, and 128 filters, respectively, each with a kernel size of 3×3 and Rectified Linear Unit (ReLU) activation. Convolutional layers use filters (kernels) to slide over the input image, detecting features like edges and textures.
- Three-pooling layers: Two max-pooling layers with a pool size of 2×2. Max-pooling reduces the spatial dimensions of the feature maps, making the network more robust to small variations in the input.
- Fully connected layers: One fully connected layer with 128 neurons and ReLU activation.
- Dropout layer: A dropout rate of 0.5. Dropout is used after the fully connected layer to randomly set a fraction of the neurons to 0 during training, helping to prevent overfitting.
- Output layer: Predicts the digit (0-9) using softmax activation. Softmax converts the output into a probability distribution over the classes.

Training

The CNN was trained using the *rmsprop* optimizer with a learning rate of 0.001 and 20 epochs. The *rmsprop* optimizer adapts the learning rate for each parameter, which can lead to faster convergence. A batch size of 32 was used, which balances computational efficiency and training stability.

B. Fully Connected Neural Network (FCNN)

FCNNs consist of fully connected layers where each neuron in one layer is connected to every neuron in the next layer. They are less effective for image classification compared to CNNs because they do not explicitly model spatial relationships.

Architecture

The FCNN architecture includes:

- Input layer: Flattened 64×64 grayscale images. Flattening converts the 2D image into a 1D vector.
- Two fully connected layers: Each layer has 128 neurons with ReLU activation.
- Output layer: Predicts the digit (0-9) using softmax activation.

Training

The FCNN was trained using the *adam* optimizer with a learning rate of 0.001 and 50 epochs. The *adam* optimizer combines the benefits of adaptive learning rates and momentum. A batch size of 32 was used.

IV. Results

In this section, the performance results for both the CNN and the FCNN models is presented. The evaluation focuses primarily on the accuracy achieved by each model on the test dataset, providing a comparison of their effectiveness in sign language digit recognition.

A. CNN Performance

The CNN model achieved an accuracy of approximately 92.5% on the test set, demonstrating its effectiveness in capturing spatial features relevant to sign language digit recognition.

In Figure 1, the evolution of the performance of the CNN model on the training and test datasets across epochs can be seen.

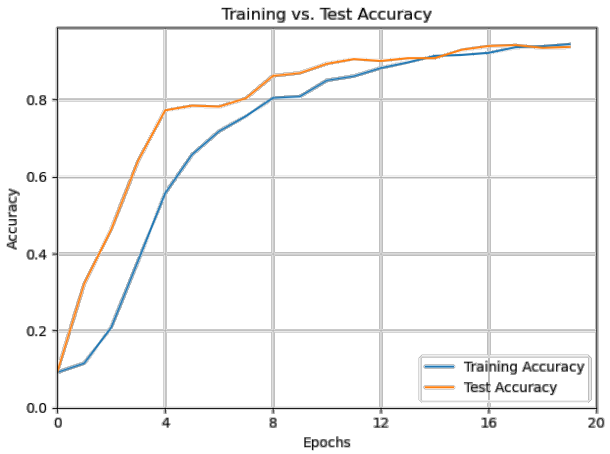

Fig. 1. Performance of the CNN model on the training and test datasets

B. FCNN Performance

The FCNN model achieved an accuracy of approximately 87.4% on the test set. Figure 2 illustrates the progression of the FCNN model's performance on both the training and test datasets throughout the epochs.

Fig. 2. Performance of the FCNN model on the training and test datasets

V. CNN Visualization

A visualization tool was developed to illustrate the transformation of input images through the CNN layers. This tool provides insights into the feature extraction process and the activation of neurons in different layers. It can be seen in Figure 3.

The visualization process involves passing an input image through the trained CNN and recording the activations of each convolutional layer. These activations are then visualized as feature maps, showing how the CNN transforms the input

Fig. 3. Visualization of the feature extraction process in the CNN, showing the transformation of an input image across different layers.

image through its layers. These maps represent the features extracted by the CNN, such as edges, textures, and more complex patterns. The visualization helps in understanding how the network progressively learns more abstract representations of the input.

VI. Challenges and Discussion

During the project, several challenges were encountered, including:

- Determining the optimal architecture configuration.
- Understanding the impact of batch size and number of epochs.
- Debugging and correcting label inconsistencies in the dataset.

A. Architecture Configuration

Finding the best architecture configuration was an iterative process. Different combinations of convolutional layers, pooling layers, and fully connected layers were tested. The goal was to find a balance between model complexity and performance.

B. Batch Size and Epochs

The impact of batch size and the number of epochs was also explored. Different batch sizes and epoch numbers were tested to find the optimal values. It was observed that the model's performance was sensitive to these parameters. The number of epochs was adjusted to prevent overfitting.

C. Label Inconsistencies

The label inconsistencies in the dataset were corrected to ensure accurate training and evaluation of the models. This involved manually inspecting the dataset and correcting the labels.

VII. Usefull Resources

For understanding the fundamental concepts of neural networks, I found two resources particularly helpful. First, the YouTube playlist by *3Blue1Brown* [3] provides a visually engaging introduction to the topic, explaining the basic building blocks and mechanisms with clear and intuitive animations. Second, the comprehensive online book by *Michael Nielsen* [4] offers a deeper dive into the mathematics behind neural networks, covering the theory, algorithms, and practical applications in detail.

VIII. Conclusion

This project demonstrated the effectiveness of CNNs and FCNNs for sign language digit recognition. The CNN model outperformed the FCNN model, achieving higher accuracy. The visualization tool provided valuable insights into the CNN's operation, highlighting the importance of spatial feature extraction. The results suggest that CNNs are well-suited for image classification tasks like sign language recognition, where spatial relationships between pixels are crucial.

References

[1] João Nuno Valente, "Recognition of Sign Language Digits," GitHub. https://github.com/joaonunovalente/Recognition-of-Sign-Language-Digits

[2] Arda Mavi, "Sign Language Digits Dataset," Kaggle, 2018. https://www.kaggle.com/datasets/ardamavi/sign-language-digits-dataset

[3] 3Blue1Brown, "But what is a neural network?," YouTube playlist. https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

[4] Michael Nielsen, "Neural Networks and Deep Learning," Determination Press, 2015. http://neuralnetworksanddeeplearning.com/