Fully Connected Network

Group 6 - AI1804
FPT University - HCMC Campus

Tran Hoang Nam Tran Gia Hien Ngo Ha Giang
Nguyen Bao Han Bui Nhat Anh Nguyen Minh Quan

March 2024

Contents

Abstract

Fully Connected Network (FCN) is a pivotal model in the field of Deep Learning, extensively used across various applications such as image recognition, natural language processing, and sequential data prediction. This presentation delves into comprehending the structure and functionality of FCN, alongside exploring its applications and potential developments in diverse domains.Our research objectives encompass understanding the structure and operation of Fully Connected Network, investigating training methods and model fine-tuning for efficient FCN deployment, exploring FCN applications in fields like image classification and graph searching, and proposing methods to enhance FCN's performance and flexibility in specific applications.Through theoretical exploration and practical experimentation, this presentation elucidates Fully Connected Layer concepts, activation functions, forward and backward propagation methods, and loss functions. Furthermore, it delves into the comparison between FCN and Convolutional Neural Networks (CNNs) in terms of image processing efficiency and performance.The findings reveal the significance of Fully Connected Layers as a foundational element in deep learning architectures, characterized by their structural agnosticism and flexibility across diverse input data. While FCN exhibits proficient performance and ease of implementation, it also faces challenges such as overfitting and computational intensity. Hence, the presentation suggests strategies like regularization, optimization techniques, network architecture refinement, and data augmentation to mitigate these challenges and enhance FCN's efficacy.In conclusion, this research endeavors to provide insights into improving FCN's performance and applicability across real-world scenarios, facilitating a deeper understanding of its utilization and potential advancements in various domains.

1 Introduction

In recent years, deep learning models have emerged as powerful tools for solving complex problems across various domains, ranging from computer vision to natural language processing. Among these models, Fully Connected Networks (FCNs) have played a crucial role in advancing the capabilities of deep learning systems.

The foundation of FCNs lies in their ability to establish connections between every neuron in adjacent layers, allowing for comprehensive information processing and feature extraction. Initially popularized in image classification tasks, FCNs have since found applications in diverse fields, including but not limited to, speech recognition, financial forecasting, and medical diagnosis.

This paper aims to provide a comprehensive overview of FCNs, elucidating their architectural principles, training methodologies, and practical applications. Additionally, it seeks to explore the comparative advantages and limitations of FCNs compared to other deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

By delving into the intricacies of FCNs and their role in contemporary deep learning research, this paper endeavors to contribute to the broader discourse surrounding artificial intelligence and its potential implications for society.

2 Definition

A fully connected layer in neural networks applies a linear transformation to the input vector using a weights matrix, resulting in all possible connections between layers. This implies that each input of the input vector influences every output of the output vector.

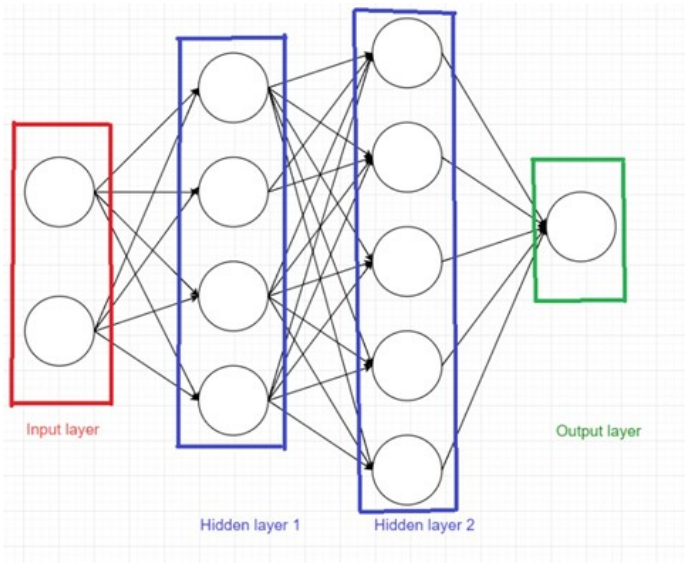

Figure 1: Illustration of a fully connected layer.

Neural networks comprise dependent non-linear functions, where each function consists of a neuron or perceptron. In fully connected layers, neurons apply a linear transformation to the input vector through a weights matrix, followed by a non-linear activation function f.

$$y_{jk}(x) = f\left(\sum_{i=1}^{n_H} \left(w_{jk}x_i + w_{j0}\right)\right)$$

Figure 2: Equation for the non-linear transformation in a fully connected layer.

This transformation involves taking the dot product between the weights matrix W and the

input vector x. While the bias term (W_0) can be added inside the non-linear function, it is disregarded here as it does not affect the output sizes or decision-making.

Visualizing a fully connected layer in a neural network with an input size of nine and an output size of four:

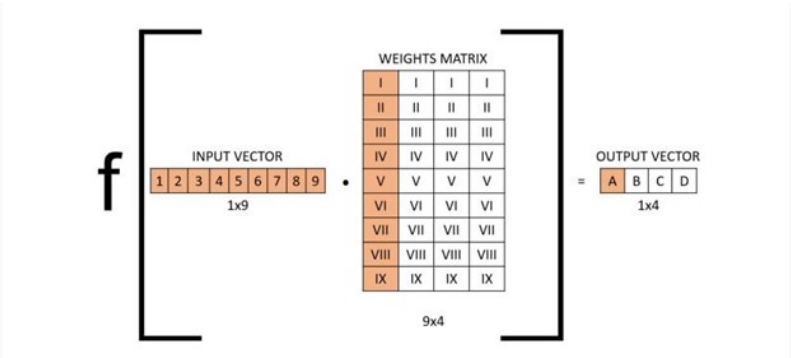

Figure 3: Visualization of the fully connected layer operation.

An alternative visualization of a fully connected layer:

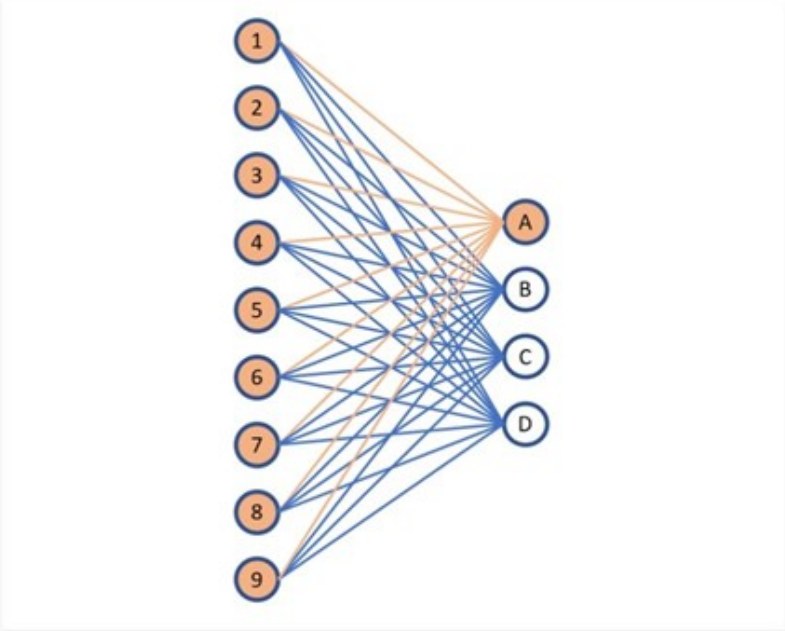

Figure 4: Alternate visualization of a fully connected layer.

This illustration underscores the "fully connected" nature of these layers, where every input influences every output, though not all weights affect all outputs. The orange lines denote connections from the first neuron, affecting output A exclusively.

Activation Function

- **Definition**: An activation function simulates the firing rate of a neuron's axon. In an artificial neural network, the activation function acts as a non-linear element at the output of neurons.
- **Necessity of Non-linearity**: Without non-linear activation functions, our neural network, despite having multiple layers, would effectively behave like a single linear layer.

- **Examples**: Sigmoid, Tanh, ReLU, Softmax.

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Transforms input to a value in the range (0, 1).
- Often used in the output layer of binary classification models.

Tanh

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- Transforms input to a value in the range (-1, 1).
- Retains non-linear properties like the sigmoid function but with an extended value range.

ReLU

$$\text{ReLU}(x) = \max(0, x)$$

- Simple and computationally efficient.
- Converts negative values to 0 while leaving positive values unchanged.
- Often used in hidden layers of deep learning models.

Softmax

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- Converts input to a probability distribution, suitable for multi-class classification tasks.
- Probabilities sum up to 1.

2.1 Forward Propagation

Concept: Forward Propagation is the process by which data is passed through a neural network from the input layer to the output layer without feedback connections. In this process, data is transmitted through each layer from the input layer to the output layer, and each layer performs a specific operation to transform the input.

Example: Consider a 3-layer model: Input Layer, Hidden layer, Output layer.

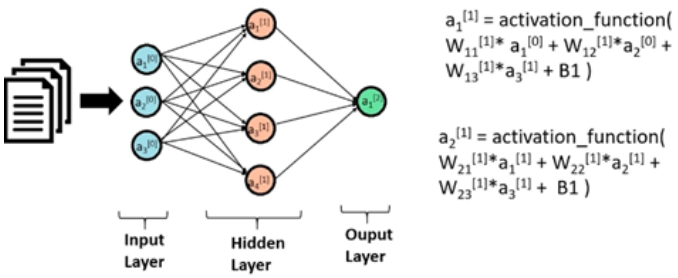

Figure 5: Forward Propagation

- Input: X (a vector or matrix representing the input data).
- Weights and Bias: Each layer has a set of weights (W) and bias (b).
- Weighted Sum and Bias Calculation: $Z = X \cdot W + b$.
- Activation Function: $A = \text{activation}(Z)$ (where activation is the activation function of that layer).

2.2 Backward Propagation

Concept: Backpropagation is a method used in the training process of neural networks to adjust the network weights based on the error between the predicted output and the actual value. This process helps the model learn how to optimize weights to minimize prediction errors.

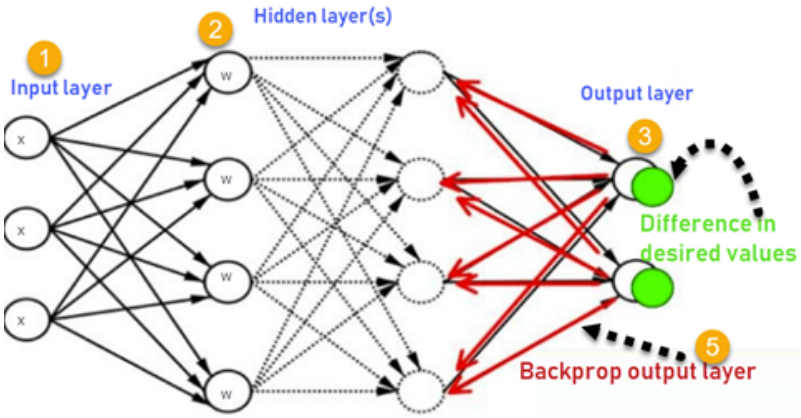

Figure 6: Backward Propagation

Steps:

- Forward Propagation:
  - Compute the output of the neural network from the input layer to the output layer.
- Compute Loss Function Gradient:
  - Calculate the gradient of the loss function with respect to the output of the last layer.
- Backward Propagation:
  - Backpropagate the gradients through the network to compute the gradient of the loss function with respect to the weights and biases of each layer.
- Update Weights and Bias:
  - Use gradient descent or other optimization methods to update the weights and biases based on the computed gradients.
- Repeat:
  - Repeat the forward and backward propagation process through multiple training iterations to improve the model.

2.3 Loss Function

A loss function evaluates the difference between the predicted output and the actual value. The choice of loss function is typically based on the type of problem.

Some Loss Functions

Cross Entropy: Cross-entropy is calculated based on the distance between the predicted probability distribution and the actual probability distribution of the classes. It measures the similarity between the predicted distribution and the actual distribution, while producing a loss value that can be used to adjust the weights during training.

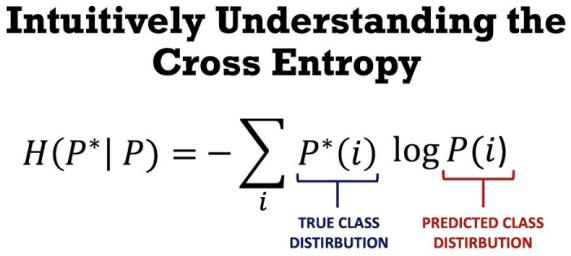

Figure 7: Cross Entropy Loss

It's important to note that when a model predicts accurately (high probability for the correct class), the cross-entropy value approaches 0. Conversely, when the model predicts inaccurately, the cross-entropy value increases. Therefore, the goal of the training process is to minimize the cross-entropy value to improve the model's prediction quality.

Mean Square Error: Mean Squared Error (MSE) is a type of loss function commonly used in regression problems, where the goal is to predict a continuous numerical value. MSE measures the difference between the predicted value and the actual value, providing a measure of the average error.

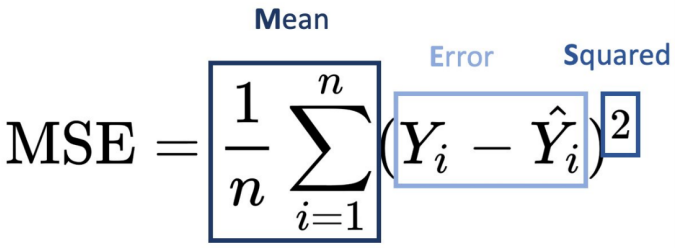

Figure 8: Mean Square Error Loss

MSE computes the average of the squared differences between the actual and predicted values. A MSE value of 0 indicates that the model perfectly fits the training data. The higher the MSE value, the greater the bias of the model towards the data.

3 Research Model

Below are some suggestions on how to use Fully Connected Network (FCN) in various fields:

3.1 Image Classification

FCN can be used as part of a Convolutional Neural Network (CNN) for image classification. In a CNN, the first layers are usually convolutional layers, designed to extract features from images, while the last layers are often Fully Connected Layers, used to classify the extracted features. A research direction could be to investigate how to optimize the number and size of FC layers to improve model performance.

3.2 Graph Searching

FCN can be used to model relationships between nodes in a graph. A research direction could be to explore the use of FCN for efficient graph searching. A specific example is using FCN in Graph Neural Networks (GNNs) to model spatio-temporal relationships in multivariate time series data.

4 The Importance of Fully Connected Layers

Fully Connected Layers play a crucial role as the foundation for many deep learning architectures. They exhibit a structure-agnostic nature, meaning they do not require any specific assumptions about the input.

For instance, a typical fully connected neural network can be utilized for image classification, where the input data consists of pixel images. This network does not need prior knowledge of the specific height or width of the images before training. Instead, it can accommodate images of varying pixel counts and sizes, adapting automatically to their dimensions.

This structure-agnostic characteristic makes neural networks flexible and capable of handling various types of input data without the need for structural modifications. This is particularly useful in scenarios where the same model is desired for multiple tasks or when input data may vary without necessitating changes to the model architecture.

By understanding Fully Connected Layers, we gain a profound insight into the inner workings of neural networks.

5 Comparison Between Fully Connected and Convolutional Networks

Explain why Convolutional Networks perform better than Fully Connected Networks:

- **Ability to learn low-level features:** CNNs use convolutional layers to learn low-level features from image data, such as edges, corners, and textures. Convolutional layers help the model automatically learn important features of the image without the need to flatten the data first.
- **Reduction of the number of parameters:** Convolutional layers reduce the number of parameters to be learned by using shared filters/kernels. This helps reduce the risk of overfitting, especially when dealing with limited training data.
- **Invariance to translation and scale:** CNNs can learn features that are invariant to translation and scale, helping the model generalize better.
- **Localized nature of convolutional layers:** Convolutional layers allow the model to focus on local regions of the image, making it well-suited for image recognition tasks.
- **Efficiency with image data:** Image data often contains spatially and temporally continuous features, which convolutional layers are very effective at processing.

In contrast, FCNs use fully connected layers, which do not leverage the spatial structure of image data and are more prone to overfitting, especially with limited data.

6 Research Results and Recommendations

6.1 Research Results on Fully Connected Networks

- **Network performance:** Fully Connected Networks typically have the ability to learn complex models from data and achieve good performance on various recognition and classification tasks.
- **Ease of deployment:** Simple architecture, easy to deploy, and train.
- **Overfitting:** Prone to overfitting, especially with large datasets and complex models.
- **Computational cost:** Demands significant computational resources, especially with large networks.
- **Scalability:** Limited scalability with large datasets and complex problems.

6.2 Options for Improving Research Directions

- **Regularization:** Use regularization techniques such as dropout, L2 regularization to reduce overfitting and improve model generalization.
- **Optimization:** Use efficient optimization algorithms such as Adam, RMSProp for faster convergence and to avoid falling into local optima.
- **Network architecture:** Optimize network architecture by experimenting with the number of layers, number of neurons, and batch size to find the most suitable structure for the specific problem.
- **Activation functions:** Use activation functions like ReLU or Leaky ReLU instead of sigmoid or tanh to avoid vanishing gradient issues and speed up convergence.
- **Data augmentation:** Apply data augmentation techniques such as rotation, zooming, shifting to expand the dataset and reduce the risk of overfitting.
- **Learning rate control:** Adjust the learning rate to ensure stable training process and faster convergence.
- **Computational optimization:** Use distributed computing techniques, GPU/TPU to reduce training time and computational costs.
- **Combination with other networks:** Combine Fully Connected Networks with other network architectures such as CNNs, RNNs to leverage the advantages of each architecture and improve performance.
- **Comparative evaluation:** Compare the performance of Fully Connected Networks with other methods and network architectures to evaluate its flexibility and effectiveness on specific datasets and problems.

This synthesis can help researchers better understand how to improve performance and apply Fully Connected Networks in practical applications.

7 Acknowledgements

We would like to express our heartfelt gratitude to everyone who has contributed to this research project. We owe a debt of gratitude to our mentors and advisors for their invaluable guidance and unwavering support throughout this journey. Their expertise and encouragement have been instrumental in shaping our understanding of Fully Connected Networks.

We would also like to extend our sincere appreciation to our fellow classmates and colleagues for their collaboration, insightful discussions, and constructive feedback. Their input has been crucial in refining our ideas and improving the quality of our work.

Furthermore, we are immensely grateful to our families and friends for their constant encouragement and understanding. Their unwavering support has been a constant source of motivation and inspiration.

This project would not have been possible without the support and encouragement of all those mentioned above. We are truly grateful for their contributions and dedication.

8 References

1. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep Learning*. MIT Press.
2. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
3. Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.
4. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.