A Brief Introduction to Deep Learning in GW Detection

Chia-Jui Chou

National Chiao Tung University, Taiwan

2019 TGWG Mini-Workshop on Gravitational Wave Data Analysis

Outline

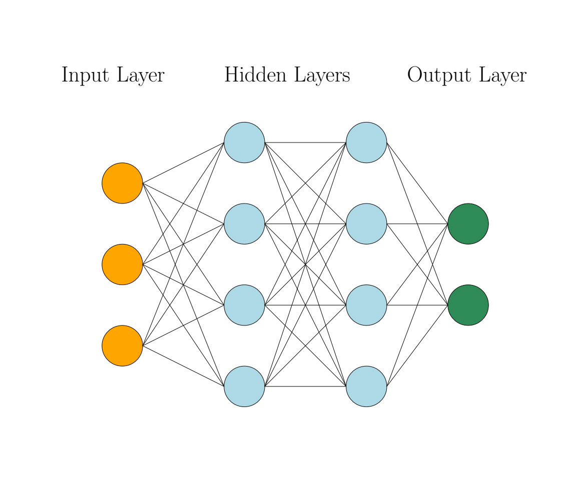
Deep Learning Using Neuron Network

- Transform an input vector to an output vector.
- Linear model: $y_i = W_{ij}x_j + b_i$, W_{ij}, b_i are parameters.
- Non-linear model: $y_i = F(W_{ij}x_j + b_i)$, F is activation function.

Commonly used activation functions:

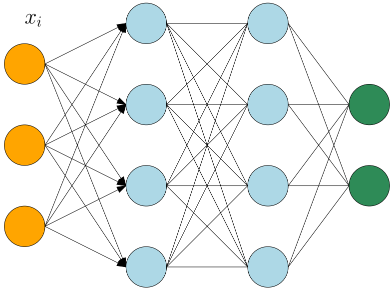
- step function
- Sigmoid: $F(x) = \frac{1}{1+e^{-x}}$
- tanh: $F(x) = \tanh(x)$
- Rectified Linear Unit (relu): $F(x) = \max(0, x)$

Deep Learning Using Neuron Network

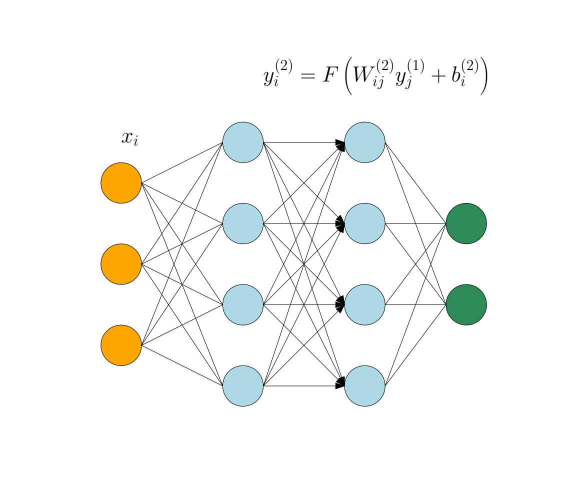


Deep Learning Using Neuron Network

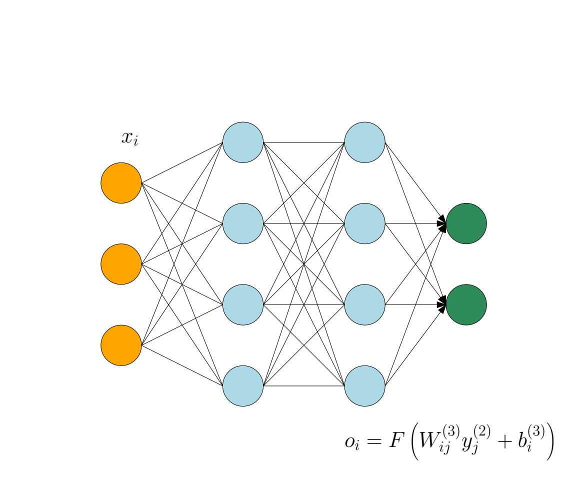
$$y_i^{(1)} = F\left(W_{ij}^{(1)} x_j + b_i^{(1)}\right)$$



Deep Learning Using Neuron Network

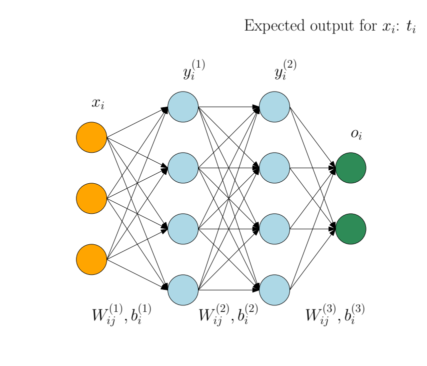


Deep Learning Using Neuron Network



Backpropagation

- Loss Function: $E_{\text{total}} = \sum_i \frac{1}{2}(o_i - t_i)^2$
- Parameters to be adjusted: $W_{ij}^{(a)}, b_i^{(a)}$
- Optimizer: Gradient Descent, for example.

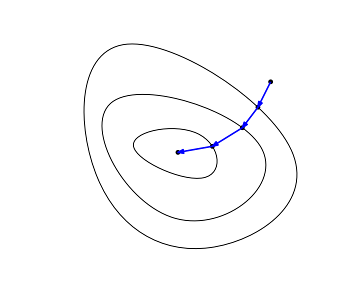


Backpropagation: Gradient Descent

- The change of loss function when varying $W_{ij}^{(3)}$:

$$\frac{\partial E_{\text{total}}}{\partial W_{ij}^{(3)}} = \frac{\partial E_{\text{total}}}{\partial o_k} \frac{\partial o_k}{\partial F\left(y_i^{(2)}\right)} \frac{\partial y_i^{(2)}}{\partial W_{ij}^{(3)}}.$$

- $W_{ij}^{(3)\prime} = W_{ij}^{(3)} + \eta \cdot \frac{\partial E_{\text{total}}}{\partial W_{ij}^{(3)}}.$

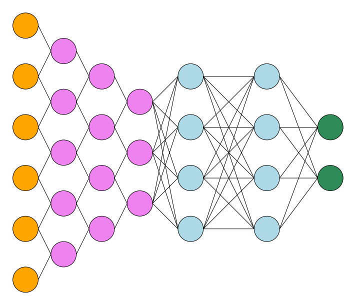


Training the Model

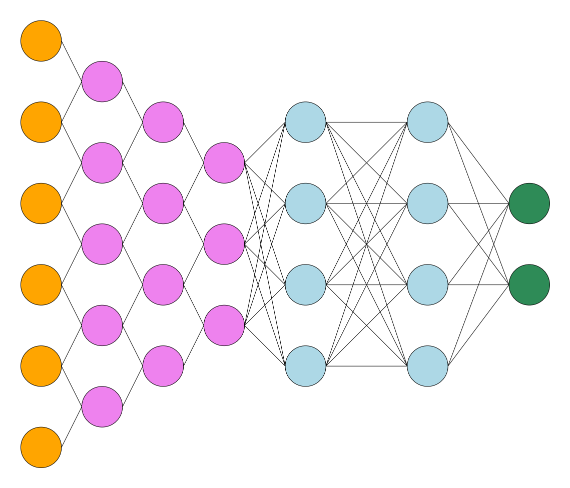
1. Prepare labeled training dataset.
2. Initialize the hyperparameters.
3. Feed forward the training dataset.
4. Adjust the parameters to optimize the objective function (minimize the loss function).

Convolutional Neuron Network

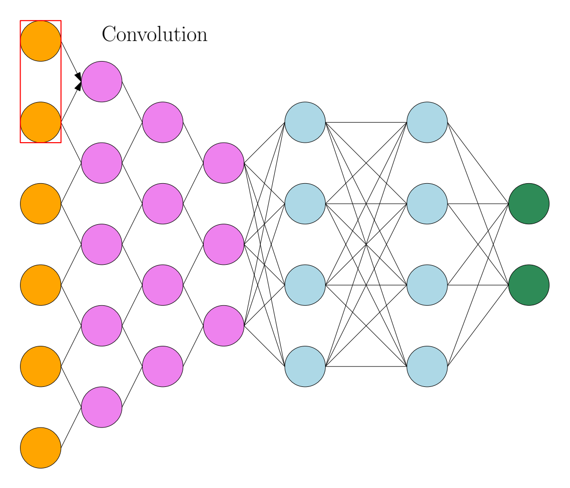
- Convolution Operation: Filter out special pattern
- Pooling: Mean Pooling, Max Pooling, Sum Pooling
- Fully-connected layers



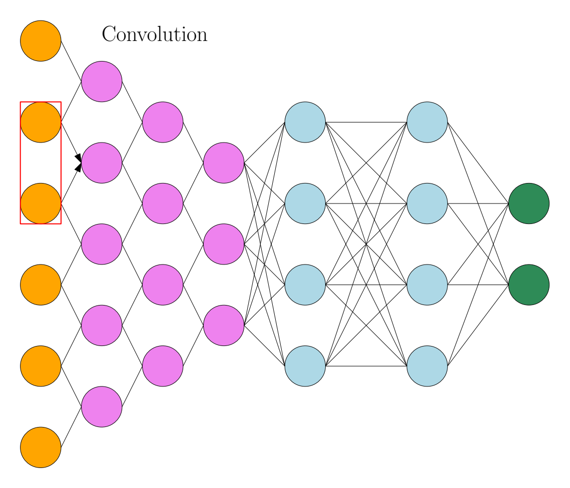
Convolutional Neuron Network



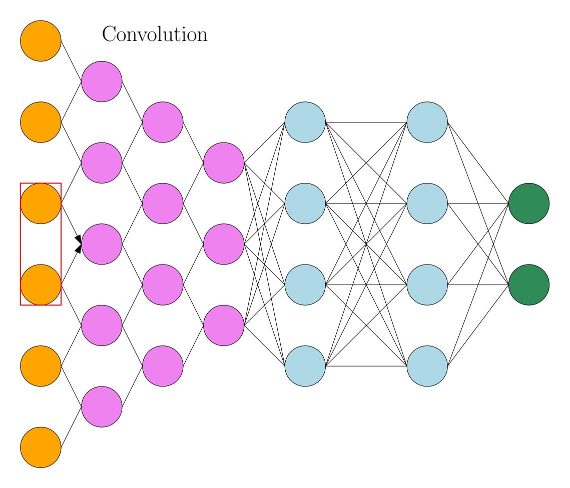
Convolutional Neuron Network



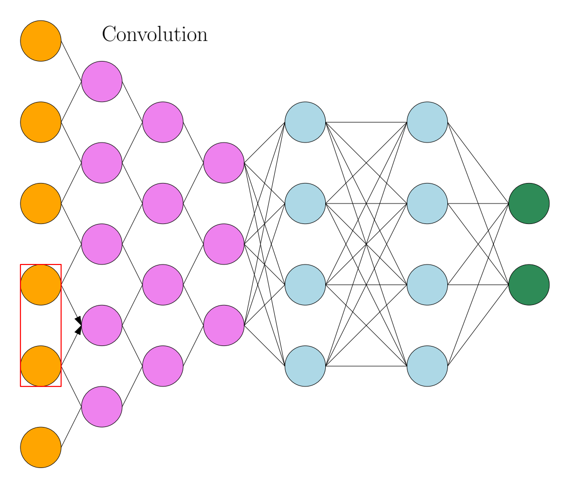
Convolutional Neuron Network



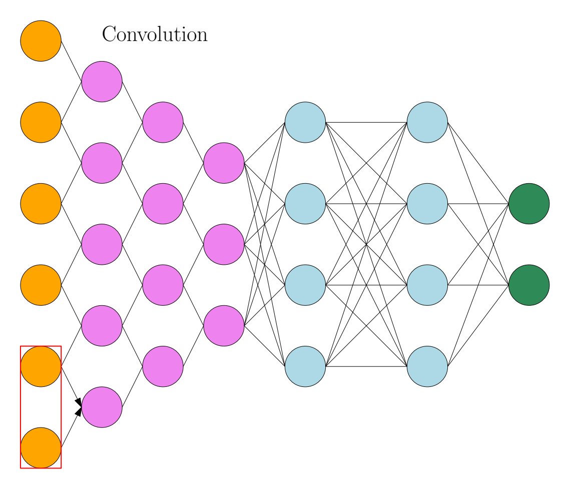
Convolutional Neuron Network



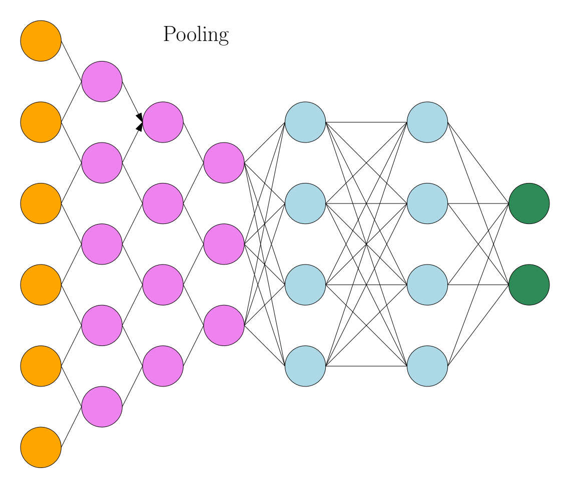
Convolutional Neuron Network



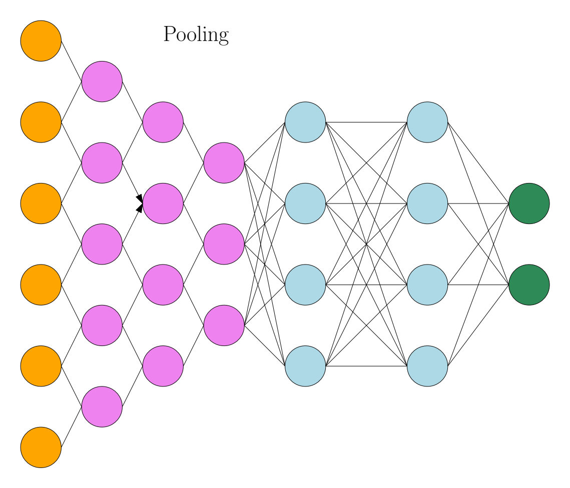
Convolutional Neuron Network



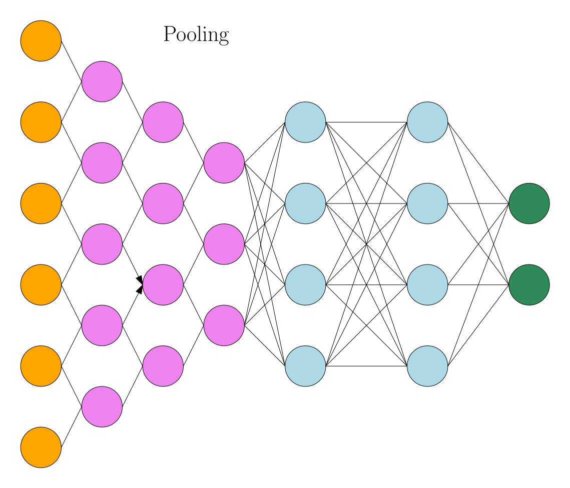
Convolutional Neuron Network



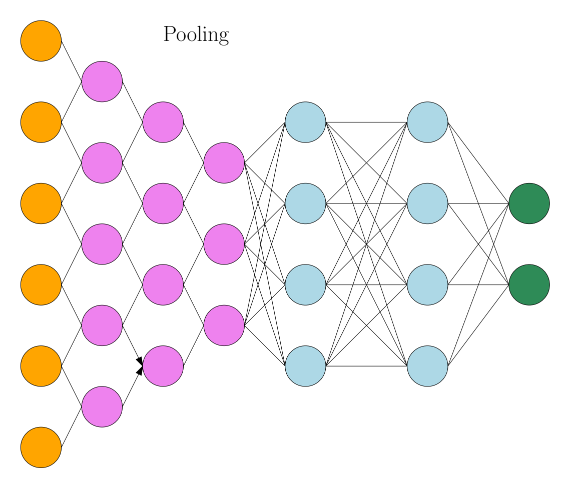
Convolutional Neuron Network



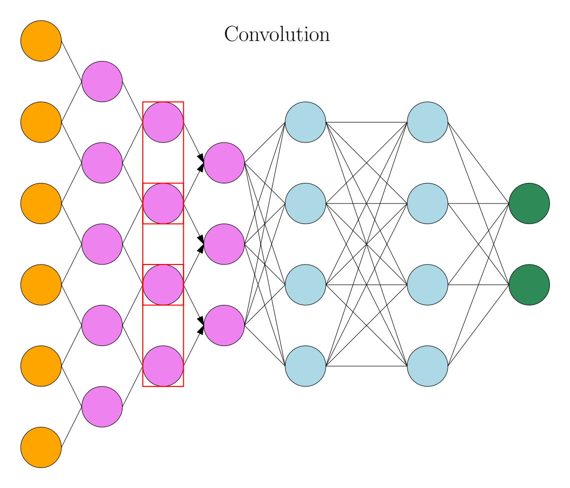
Convolutional Neuron Network



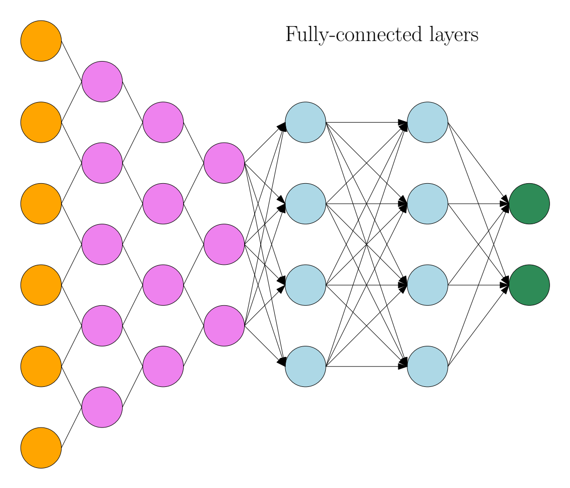
Convolutional Neuron Network



Convolutional Neuron Network



Convolutional Neuron Network



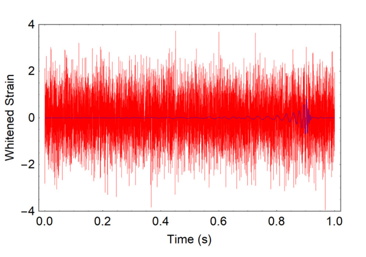
CNN for GW Detection: Deep Filtering

The advantages of Deep Filtering:

- Saving computing resources comparing to Match Filtering

Reference: arXiv: 1711.03121:

The input data is the real time strain data, for a 1-second-long input with sampling rate of 8192Hz, the input is a vector with 8192 components.



CNN for GW Detection: Deep Filtering

	Input	vector (size: 8192)
1	Reshape	matrix (size: 1×8192)
2	Convolution	matrix (size: 64×8177)
3	Pooling	matrix (size: 64×2044)
4	ReLU	matrix (size: 64×2044)
5	Convolution	matrix (size: 128×2014)
6	Pooling	matrix (size: 128×503)
7	ReLU	matrix (size: 128×503)
8	Convolution	matrix (size: 256×473)
9	Pooling	matrix (size: 256×118)
10	ReLU	matrix (size: 256×118)
11	Convolution	matrix (size: 512×56)
12	Pooling	matrix (size: 512×14)
13	ReLU	matrix (size: 512×14)
14	Flatten	vector (size: 7168)
15	Linear Layer	vector (size: 128)
16	ReLU	vector (size: 128)
17	Linear Layer	vector (size: 64)
18	ReLU	vector (size: 64)
19	Linear Layer	vector (size: 2)
	Output	vector (size: 2)

Example

- Demonstration using tensorflow, on Google Colab and Google Drive.
- Codes from Dr. Chun-Yu Lin
- More details to be discussed...