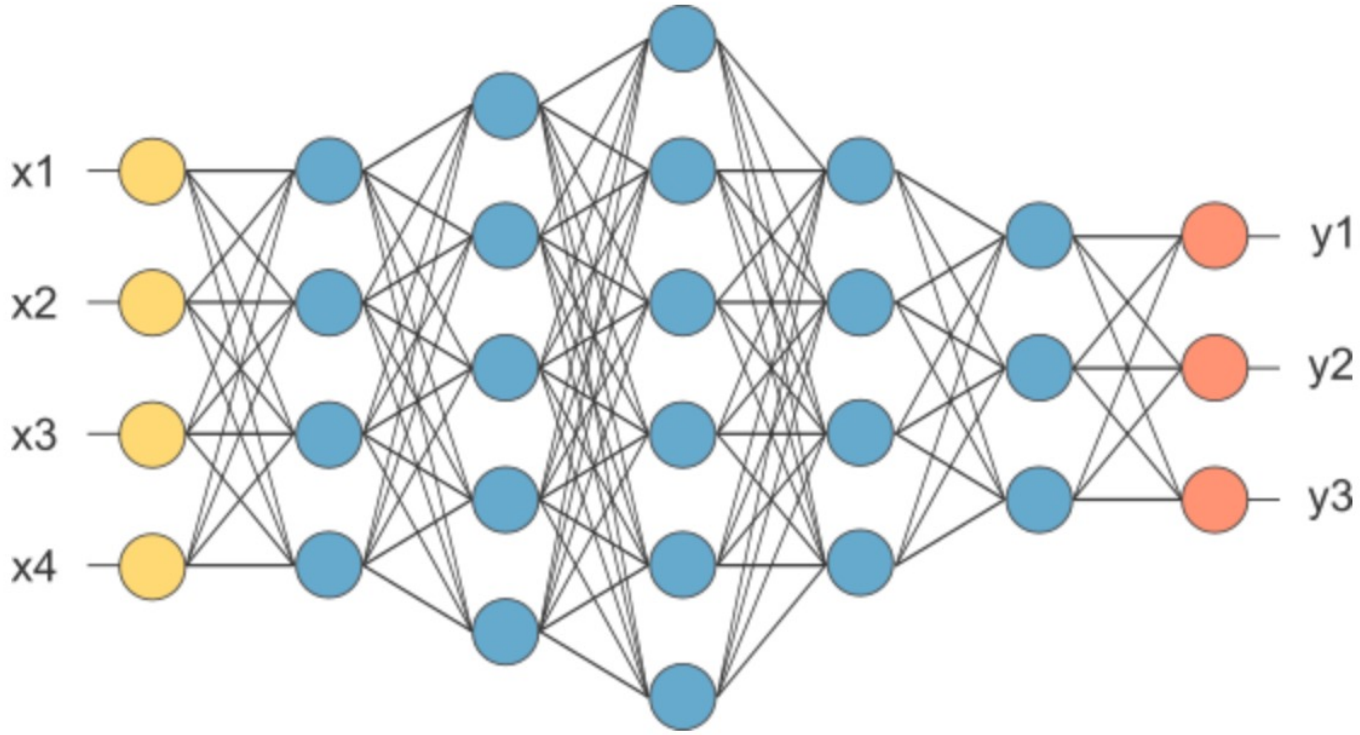
DEEP LEARNING FOR COMPUTER VISION

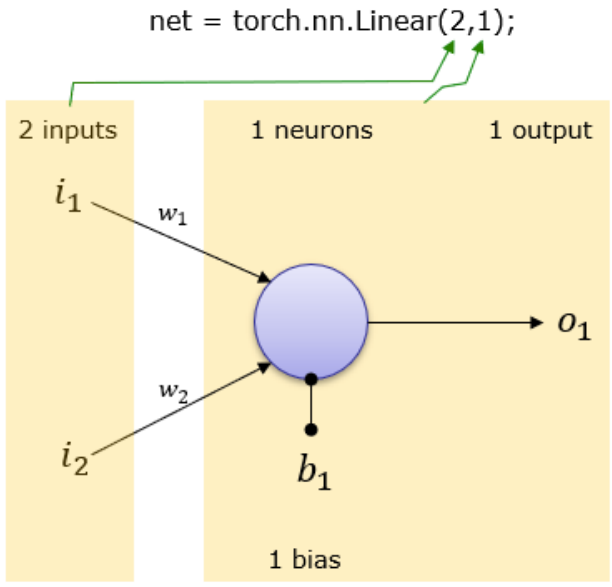
Week6

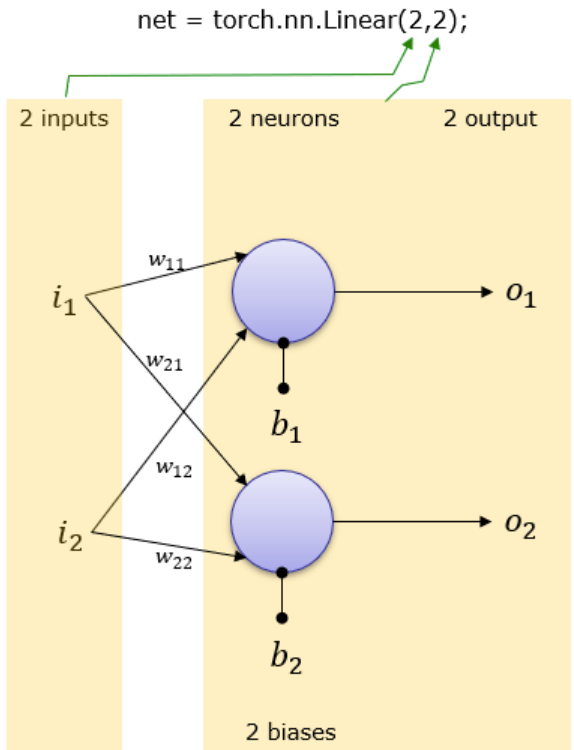
Dr. Tuchsanai. PloySuwan

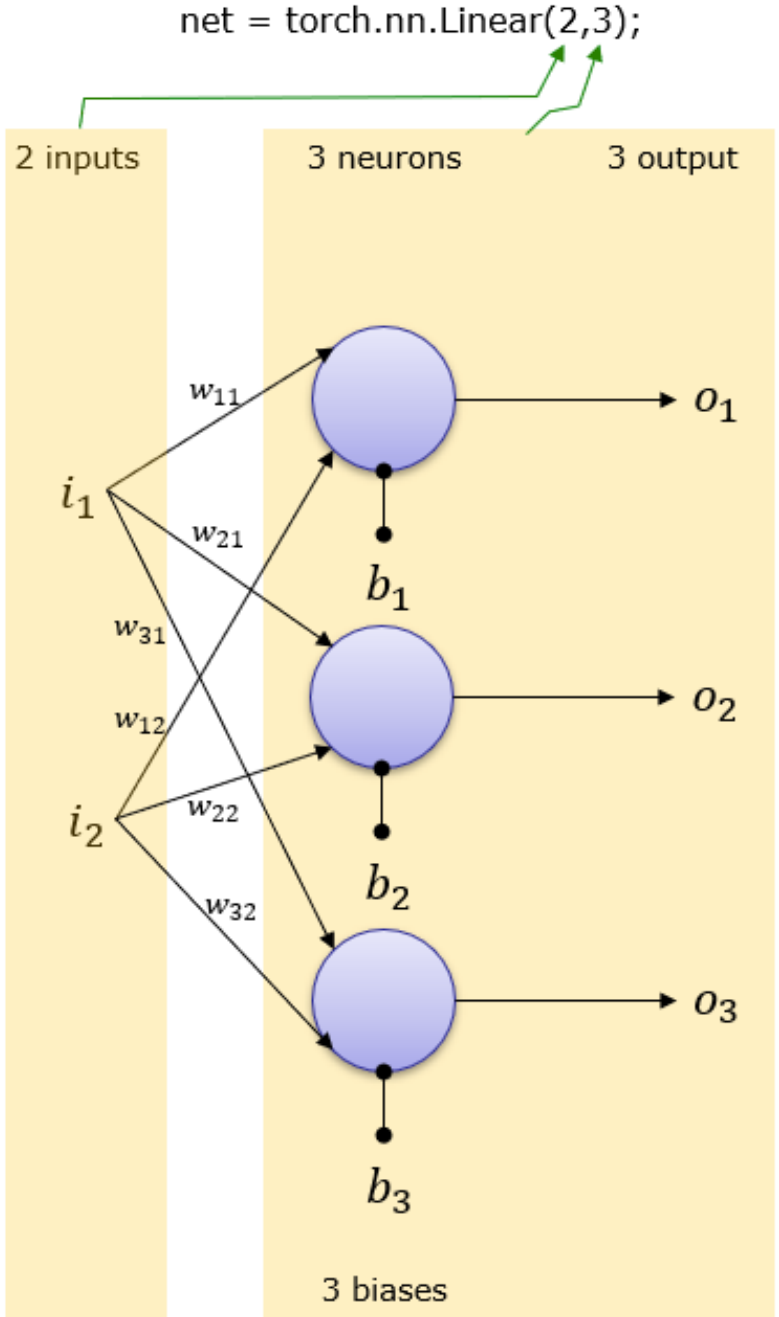
- **Fully Connected Layer**

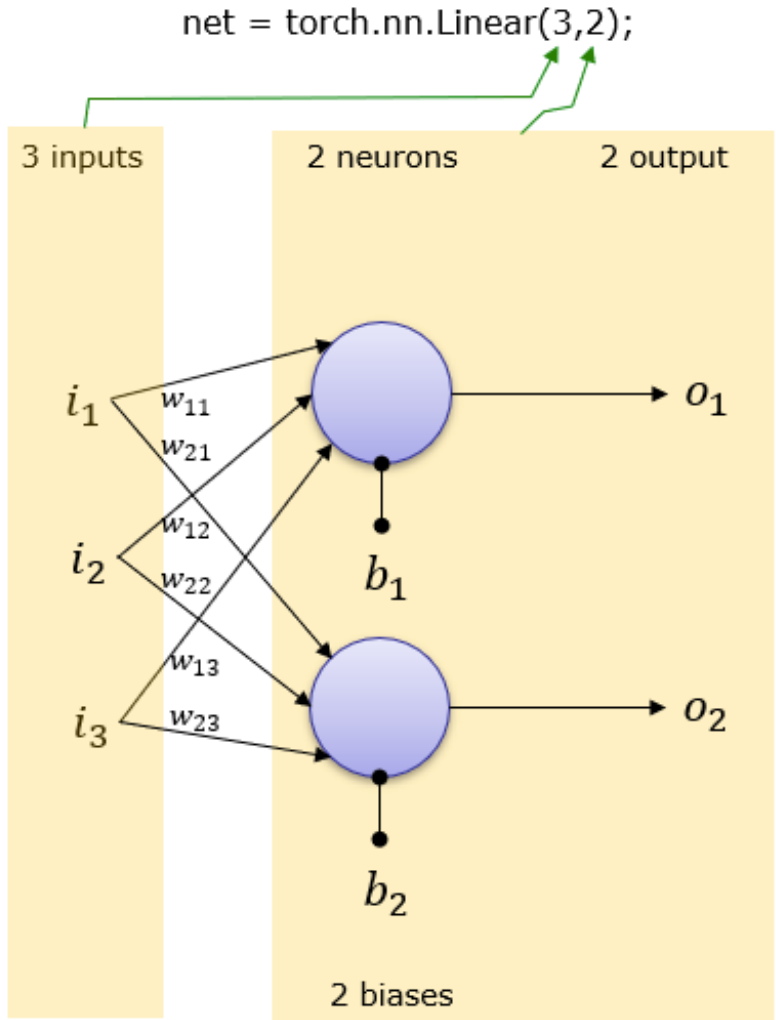
The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.



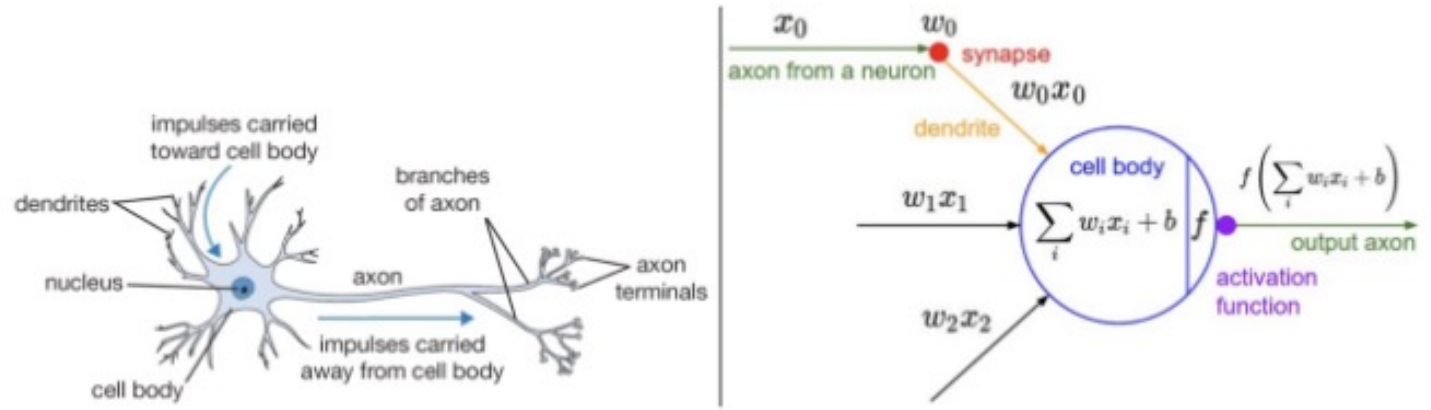

net = torch.nn.Linear(2,1);
2 inputs
1 neurons
1 output
i_1
w_1
o_1
w_2
i_2
b_1
1 bias


net = torch.nn.Linear(2,2);
2 inputs
2 neurons
2 output
w_{11}
i_1
o_1
w_{21}
b_1
w_{12}
i_2
o_2
w_{22}
b_2
2 biases


net = torch.nn.Linear(2,3);
2 inputs
3 neurons
3 output
w_{11}
i_1
o_1
w_{21}
b_1
w_{31}
w_{12}
o_2
i_2
w_{22}
b_2
w_{32}
o_3
b_3
3 biases

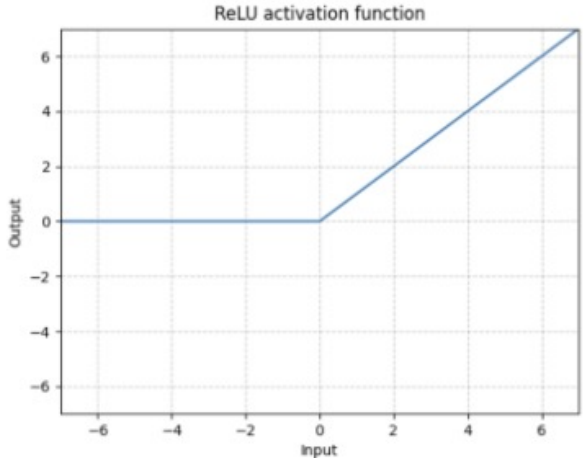

net = torch.nn.Linear(3,2);
3 inputs
2 neurons
2 output
i_1
w_{11}
w_{21}
o_1
w_{12}
i_2
b_1
w_{22}
w_{13}
o_2
i_3
w_{23}
b_2
2 biases

Activation functions in Neural Networks



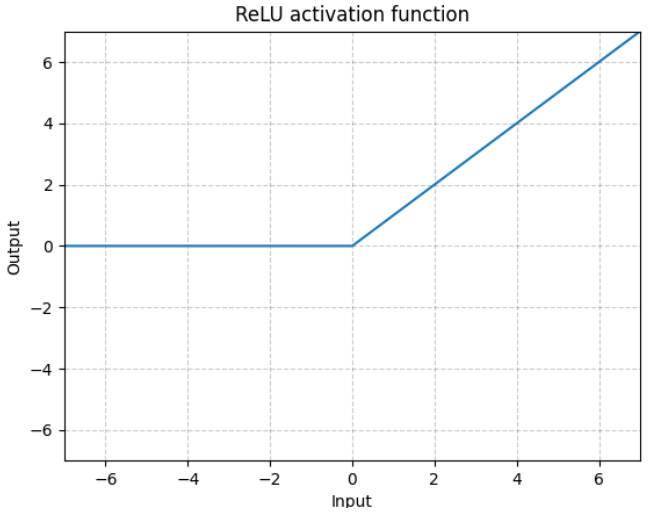
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

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



torch.nn.ReLU() -

$$\text{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$



$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + \exp(-x)}$$

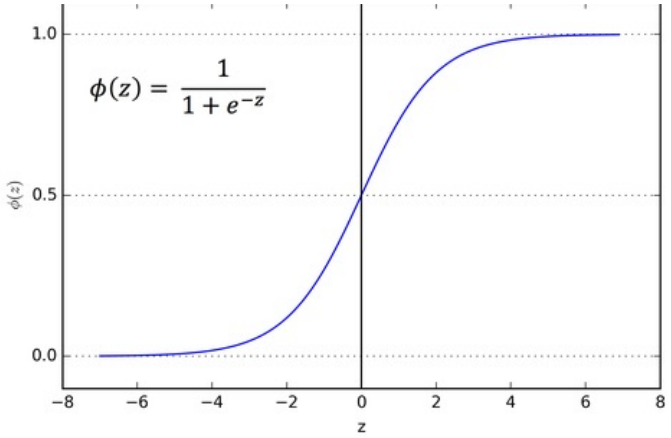
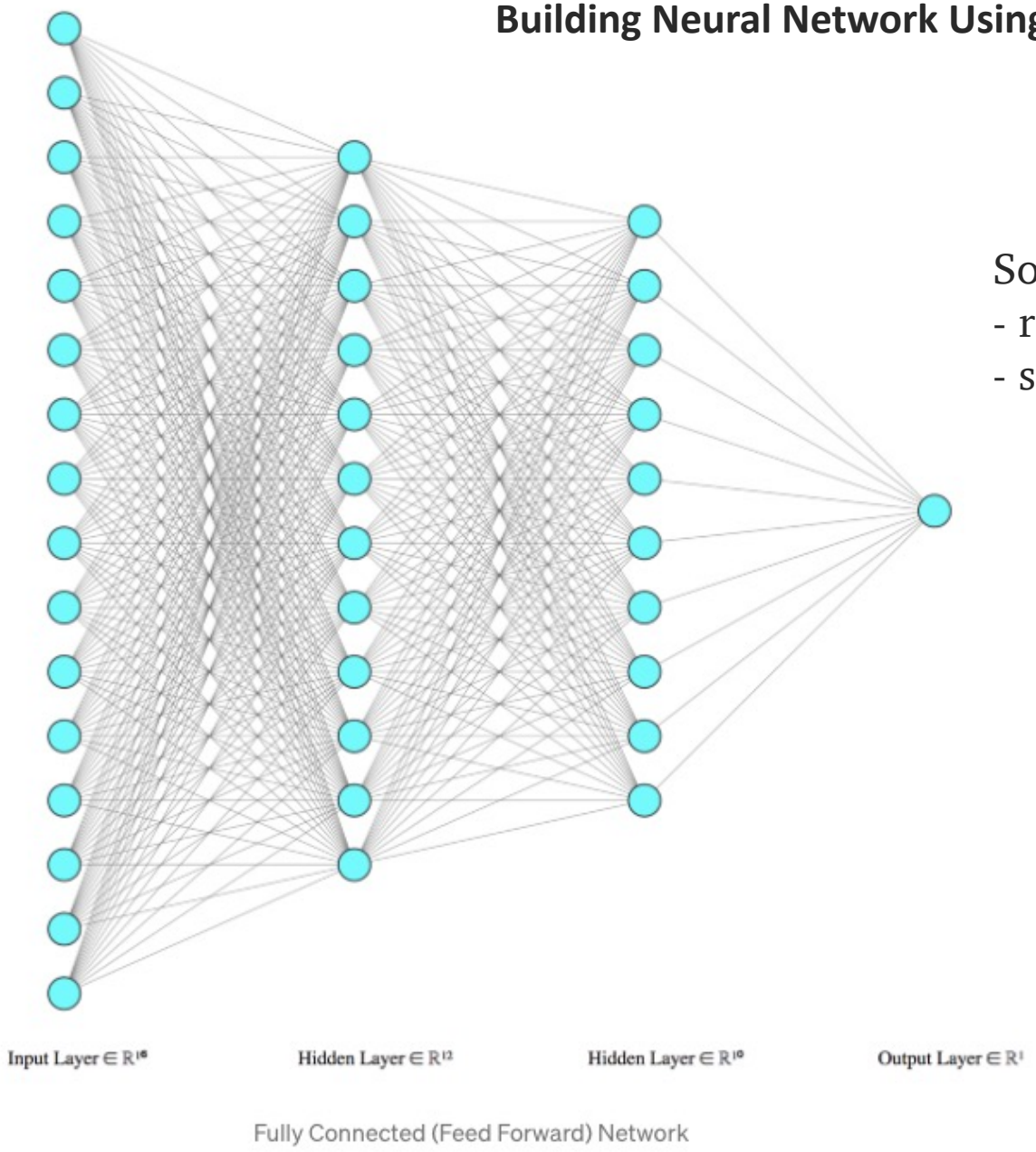


Fig: Sigmoid Function

`torch.nn.Sigmoid`

Building Neural Network Using PyTorch



So this is a Fully Connected 16x12x10x1 Neural Network -
- relu activations in hidden layers,
- sigmoid activation in output layer.

Three Ways to Build a Neural Network in PyTorch

```python
import torch
import torch.nn.functional as F
from torch import nn

# define the network class
class MyNetwork(nn.Module):
    def __init__(self):
        # call constructor from superclass
        super().__init__()

        # define network layers
        self.fc1 = nn.Linear(16, 12)
        self.fc2 = nn.Linear(12, 10)
        self.fc3 = nn.Linear(10, 1)

    def forward(self, x):
        # define forward pass
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = torch.sigmoid(self.fc3(x))
        return x

# instantiate the model
model = MyNetwork()
```

```python
from torch import nn

# define model architecture
model = nn.Sequential(
    nn.Linear(16, 12),
    nn.ReLU(),
    nn.Linear(12, 10),
    nn.ReLU(),
    nn.Linear(10, 1),
    nn.Sigmoid()
)

# print model architecture
print(model)
```

```python
from torch import nn
from collections import OrderedDict

# define model architecture
model = nn.Sequential(OrderedDict([
    ('fc1', nn.Linear(16, 12)),
    ('relu1', nn.ReLU()),
    ('fc2', nn.Linear(12, 10)),
    ('relu2', nn.ReLU()),
    ('fc3', nn.Linear(10, 1)),
    ('sigmoid', nn.Sigmoid())
]))

# print model architecture
print(model)
```

```python
import torch
from torch import nn

class MyNetwork2(nn.Module):
    def __init__(self):
        super().__init__()

        # define the layers
        self.layers = nn.Sequential(
            nn.Linear(16, 12),
            nn.ReLU(),
            nn.Linear(12, 10),
            nn.ReLU(),
            nn.Linear(10, 1)
        )

    def forward(self, x):
        # forward pass
        x = torch.sigmoid(self.layers(x))
        return x

# instantiate the model
model = MyNetwork2()

# print model architecture
print(model)
```