Practical Machine Learning

Neural Network Structure

Sven Mayer

Neuronal Network Structure

TensorFlow Syntax for a 2-Layer Model

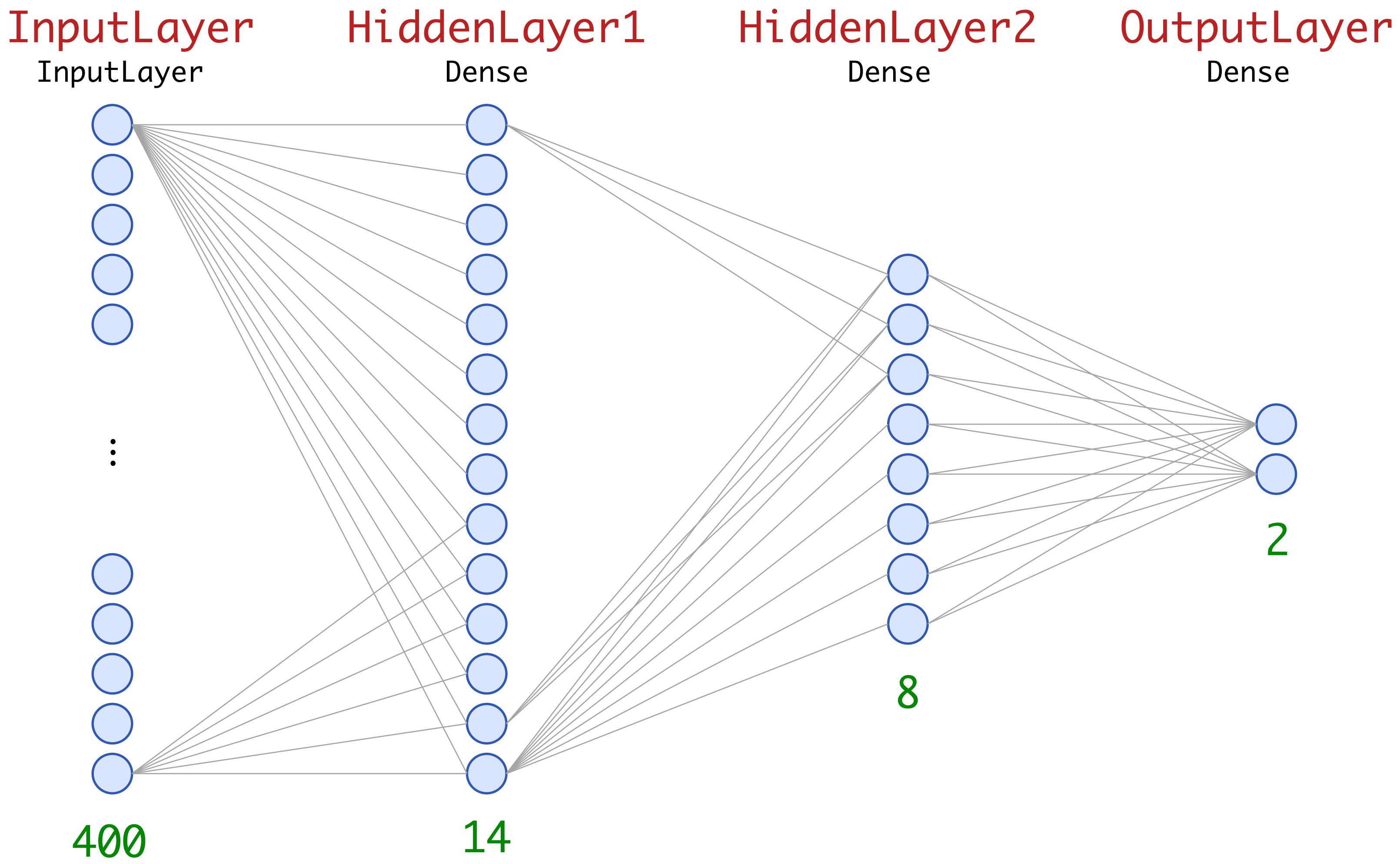
```
model = tf.keras.Sequential()

x = tf.keras.layers.InputLayer((400,), name =
"InputLayer")
model.add(x)

x = tf.keras.layers.Dense(14, name = "HiddenLayer1",
activation = 'relu')
model.add(x)

model.add(tf.keras.layers.Dense(8, name =
"HiddenLayer2", activation='relu'))

model.add(tf.keras.layers.Dense(2, name = "OutputLayer",
activation = 'softmax'))
```

Layers to Pick from

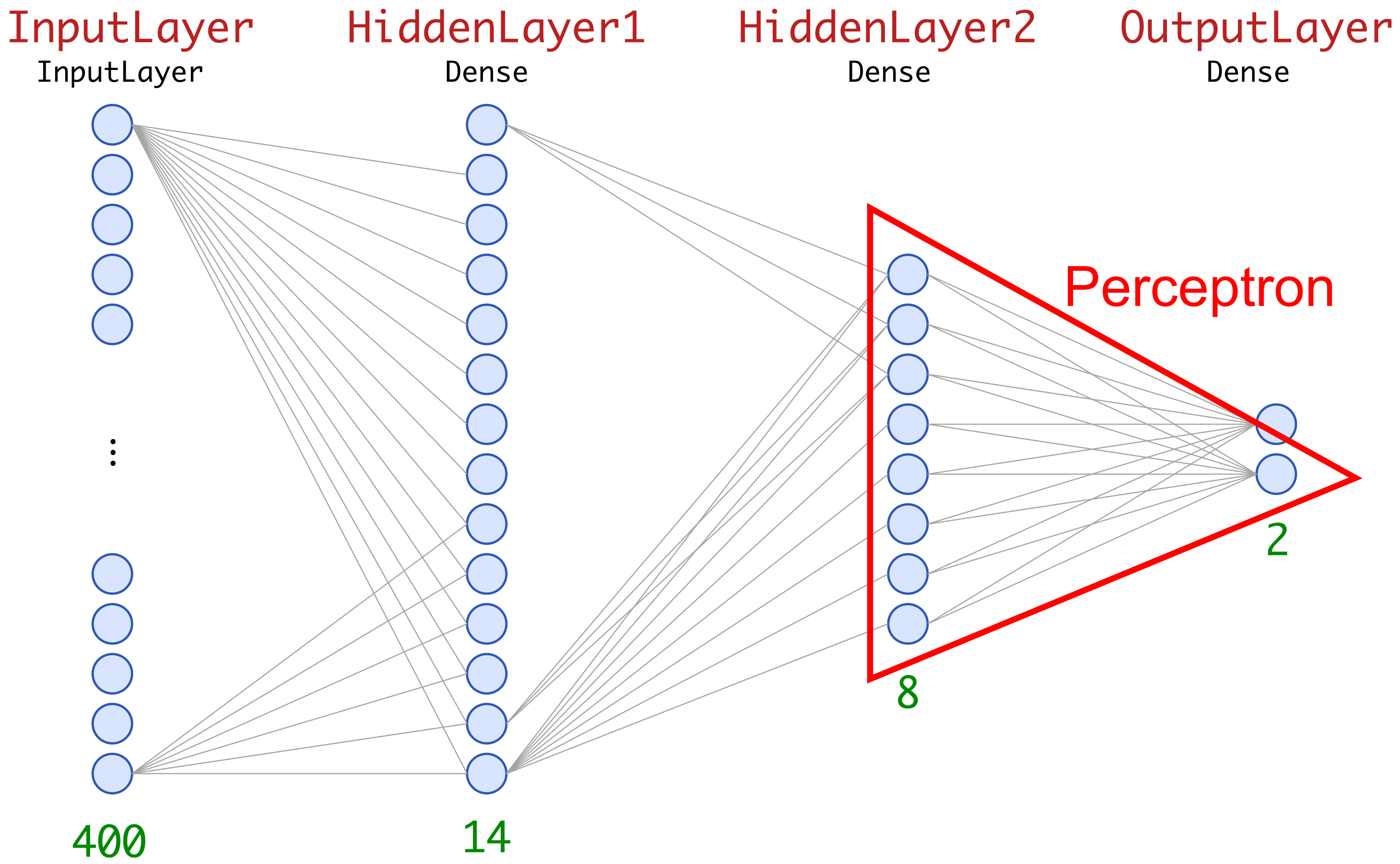
`tf.keras.layers.*`

- AbstractRNNCell
- Activation
- ActivityRegularization
- Add
- AdditiveAttention
- AlphaDropout
- Attention
- Average
- AveragePooling1D
- AveragePooling2D
- AveragePooling3D
- AvgPool1D
- AvgPool2D
- AvgPool3D
- BatchNormalization
- Bidirectional
- Concatenate
- Conv1D
- Conv1DTranspose
- **Conv2D**
- Conv2DTranspose
- Conv3D
- Conv3DTranspose
- ConvLSTM2D
- Convolution1D
- Convolution1DTranspose
- Convolution2D
- Convolution2DTranspose
- Convolution3D
- Convolution3DTranspose
- Cropping1D
- Cropping2D
- Cropping3D
- Dense
- DenseFeatures
- DepthwiseConv2D
- Dot
- Dropout
- ELU
- Embedding
- Flatten
- GRU
- GRUCell
- GaussianDropout
- GaussianNoise
- GlobalAveragePooling1D
- GlobalAveragePooling2D
- GlobalAveragePooling3D
- GlobalAvgPool1D
- GlobalAvgPool2D
- GlobalAvgPool3D
- GlobalMaxPool1D
- GlobalMaxPool2D
- GlobalMaxPool3D
- GlobalMaxPooling1D
- GlobalMaxPooling2D
- GlobalMaxPooling3D
- InputLayer
- InputSpec
- **LSTM**
- LSTMCell
- Lambda
- Layer
- LayerNormalization
- LeakyReLU
- LocallyConnected1D
- LocallyConnected2D
- Masking
- MaxPool1D
- **MaxPool2D**
- MaxPool3D
- MaxPooling1D
- MaxPooling2D
- MaxPooling3D
- Maximum
- Minimum
- MultiHeadAttention
- Multiply
- PReLU
- Permute
- **RNN**
- **ReLU**
- RepeatVector
- Reshape
- SeparableConv1D
- SeparableConv2D
- SeparableConvolution1D
- SeparableConvolution2D
- SimpleRNN
- SimpleRNNCell
- Softmax
- SpatialDropout1D
- SpatialDropout2D
- SpatialDropout3D
- StackedRNNCells
- Subtract
- ThresholdedReLU
- TimeDistributed
- UpSampling1D
- UpSampling2D
- UpSampling3D
- Wrapper
- ZeroPadding1D
- ZeroPadding2D
- ZeroPadding3D

Neuronal Network Structure



Neuronal Network Structure



Neuronal Networks

What can be trained?

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

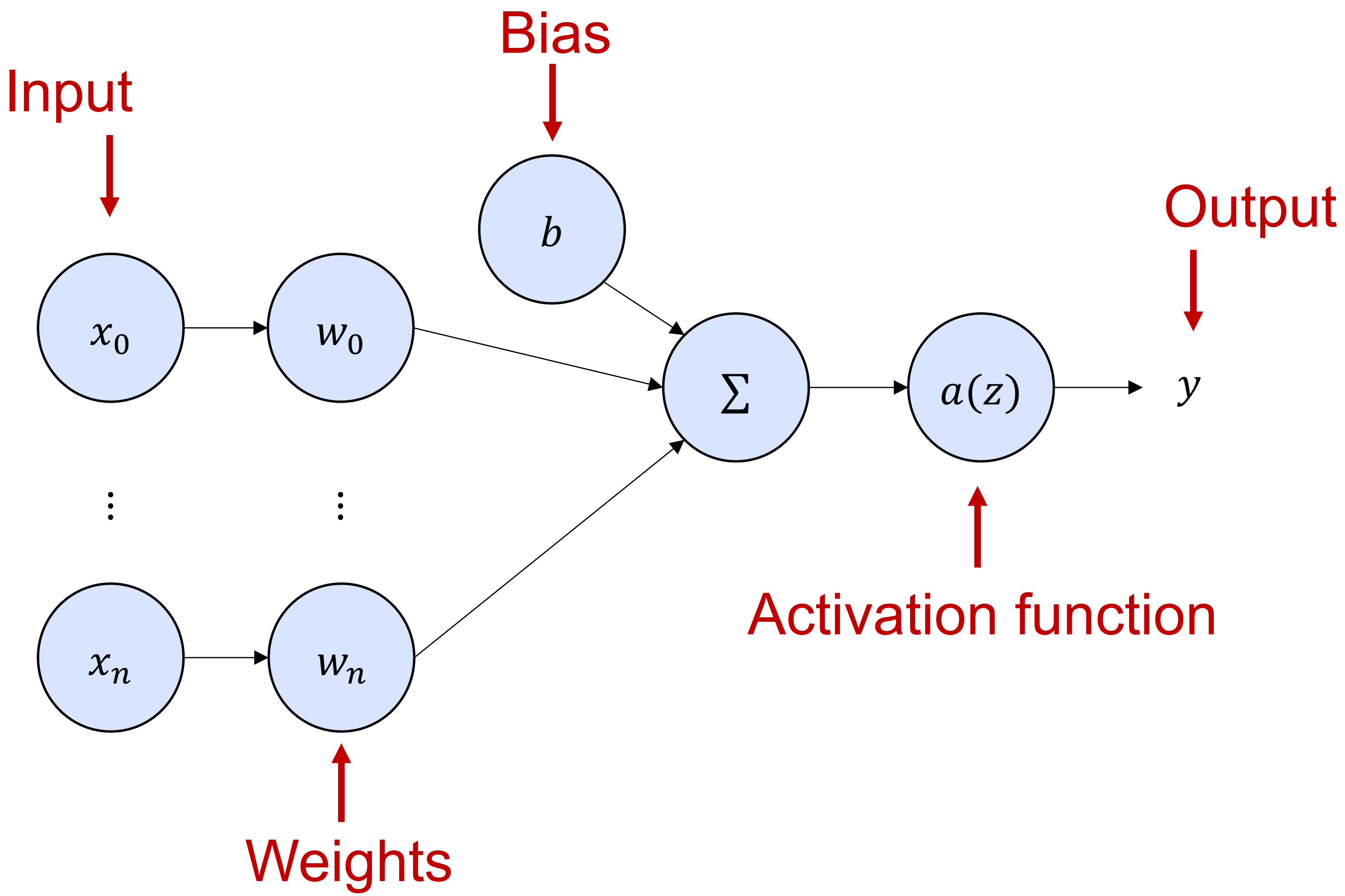
If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental

and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an

Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65, no. 6 (1958): 386. DOI: https://psycnet.apa.org/doi/10.1037/h0042519

What is a Perceptron?

Single-Layer Perceptron



What can be trained?

Single-Layer Perceptron

Input Output

$$f(x) = y$$

Model

What can be trained?

Single-Layer Perceptron

Input

$$f(x) = f\left(\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}\right) = a(x \cdot w + b) = y$$

Model

Output

What can be trained?

Single-Layer Perceptron

Input ↓ Weights ↓ Bias ↓

$$f(x) = f\left(\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}\right) = a(x \cdot w + b) = y$$

↑ Model ↑ Activation function ↑ Output

What can be trained?

Single-Layer Perceptron

Weights

Input

Bias

$$f(x) = f\left(\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}\right) = a(x \cdot w + b) = y$$

Model

Activation function

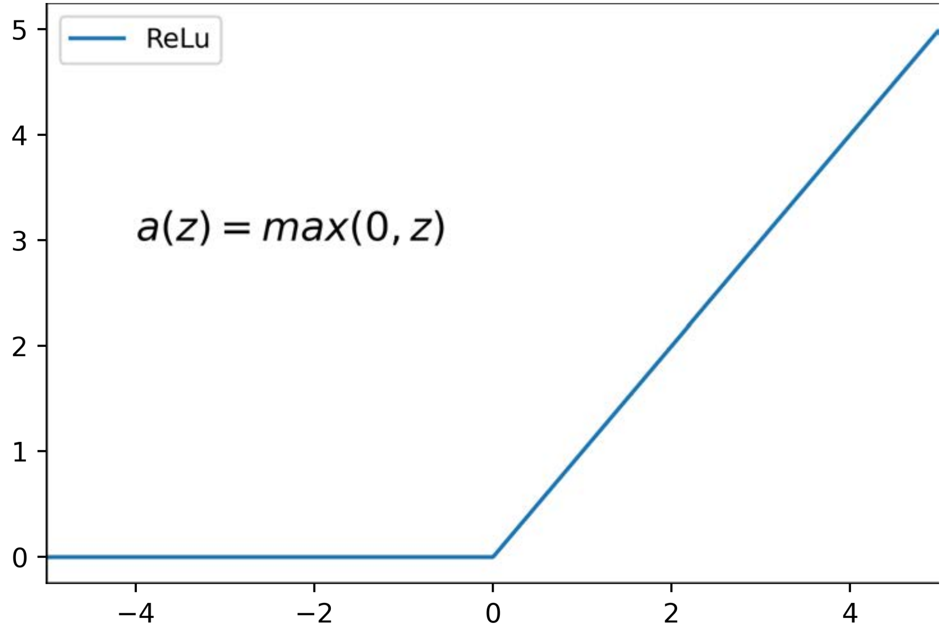
Output

$$= a\left(\sum_{i=0}^{n} x_i w_i + b\right)$$

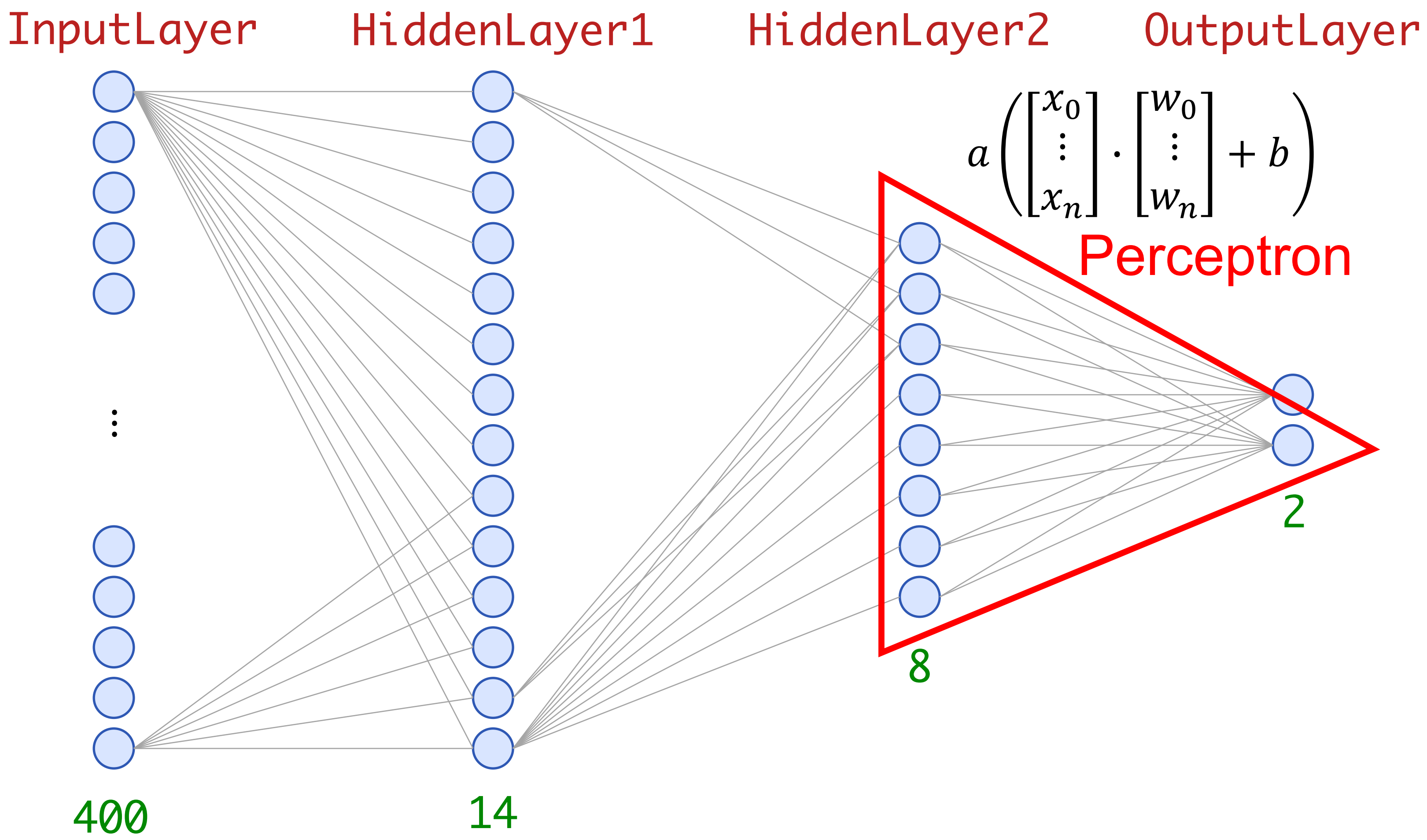
Activation Function

Rectified Linear Unit (ReLU)

$$a(z) = \max(0, z) = \begin{cases} z & if\ z > 0 \\ 0 & otherwise \end{cases} = \begin{cases} z & if\ x \cdot w + b > 0 \\ 0 & otherwise \end{cases}$$

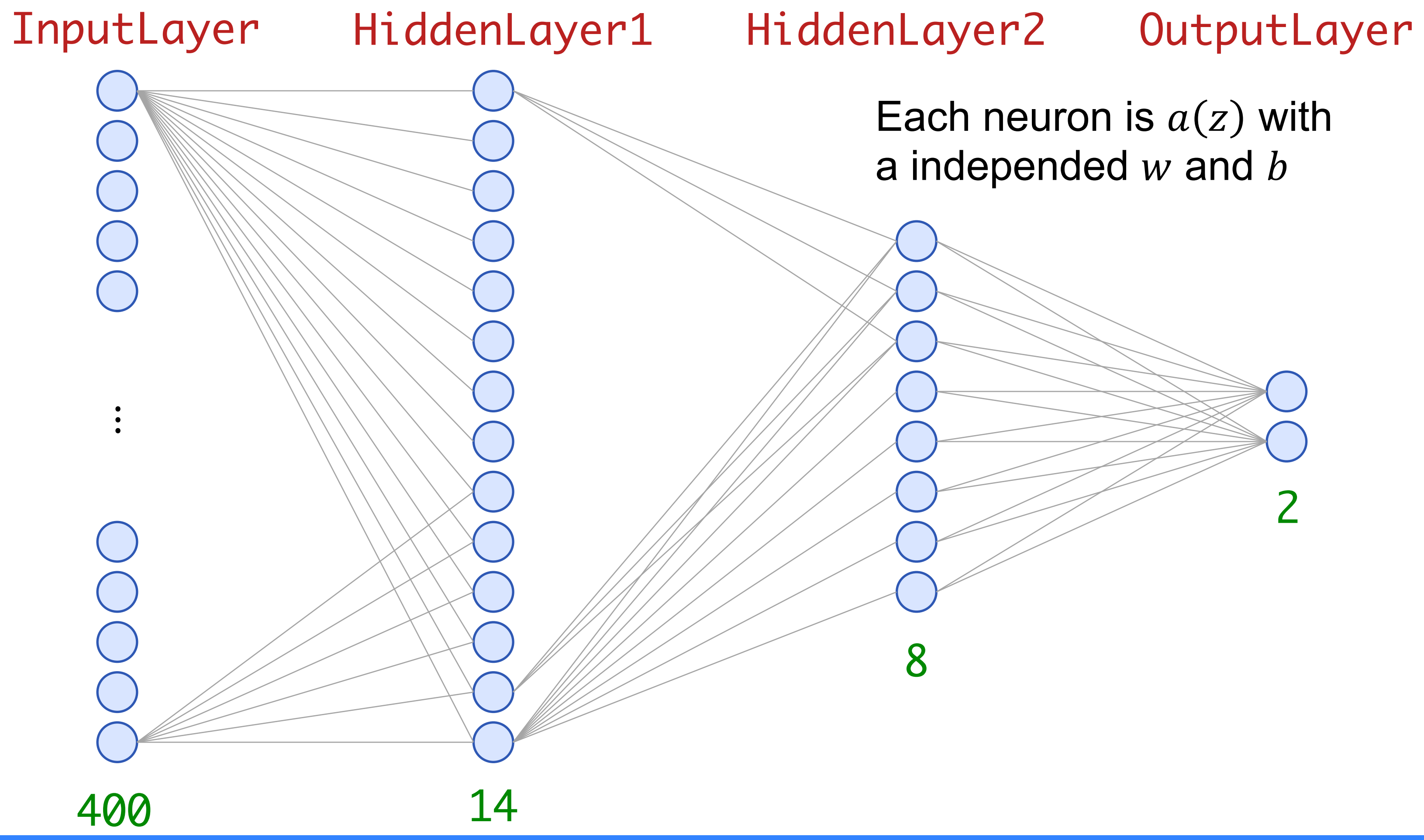


Neuronal Network Structure



Sven Mayer

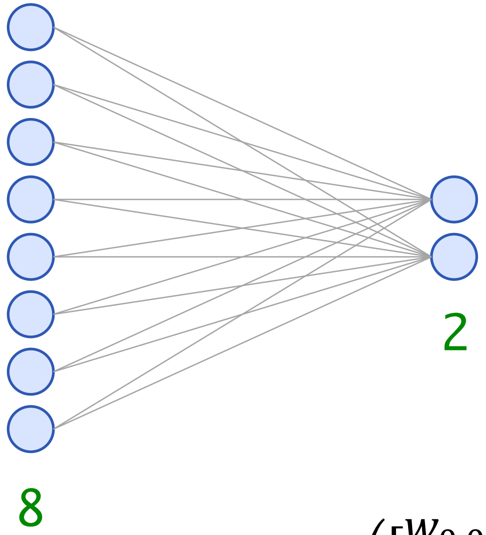
Neuronal Network Structure



Each neuron is $a(z)$ with
a independed w and b

Combining Perceptions

Why is it all about fast matrix multiplication?



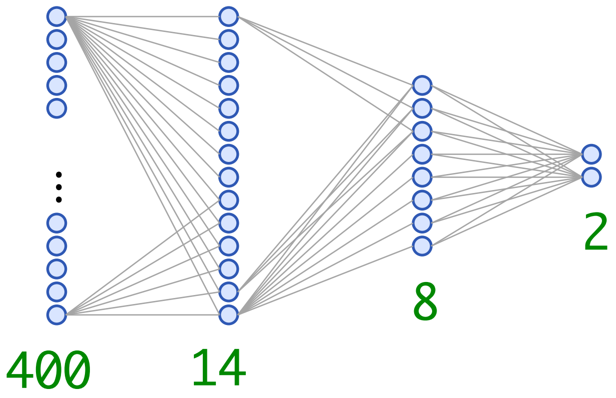
$$a(x \cdot w_0 + b_0) = y_0$$

$$a(x \cdot w_1 + b_1) = y_1$$

$$a\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,6} & w_{0,7} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,6} & w_{0,7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$a\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,m} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,m} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

Parameter



- Layers 400, 14, 8, and 2
  - => weights: 400 * 14 + 14 * 8 + 8 * 2 = 5,728
  - => biases: 14 + 8 + 2 = 24
  - => trainable parameter: 5,728 + 24 = 5,752

- Trainable parameters can raise fast
  - Layers 400, 100, 40, 2 => Parameter: 44,222
  - (one model from the walkthough)

Conclusion
Neural Network Structure

- Perceptron
- Weights
- Biases
- Combining multiple Perceptron
- Trainable Parameter
- Activation Function (e.g. ReLu)

License