Hands-on 2: Computer vision classification for cattle food grading with a hands-on exercise (Python) on Google Co-lab

Yalong Pi, Associate Research Scientist, Texas A&M Institute of Data Science

2025 ASAS, Sunday, July 6, 2025, 12:30 PM – 2:20 PM

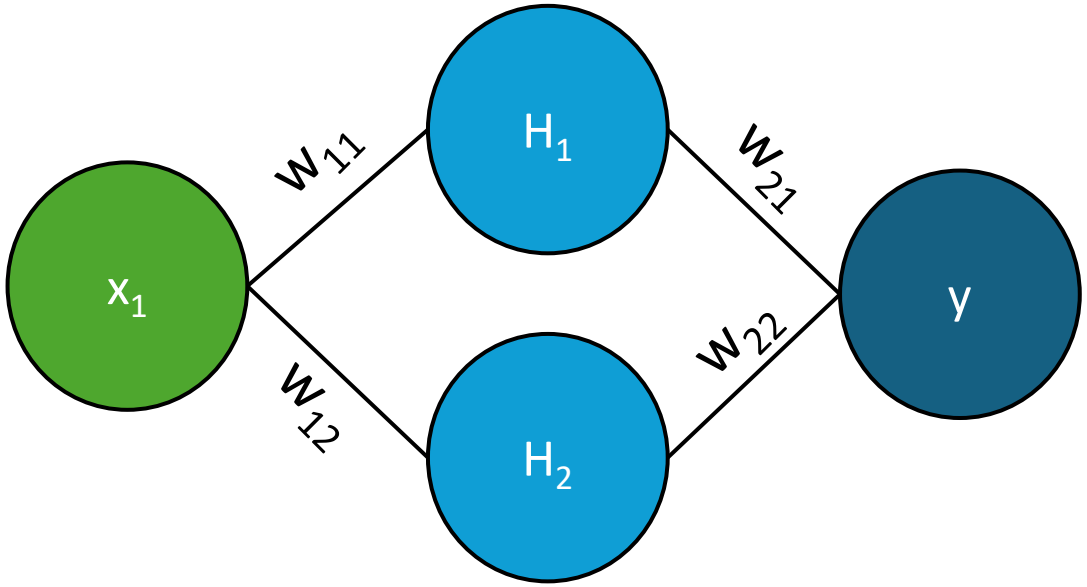
https://github.com/National-Animal-Nutrition-Program/2025ASAS_Pi

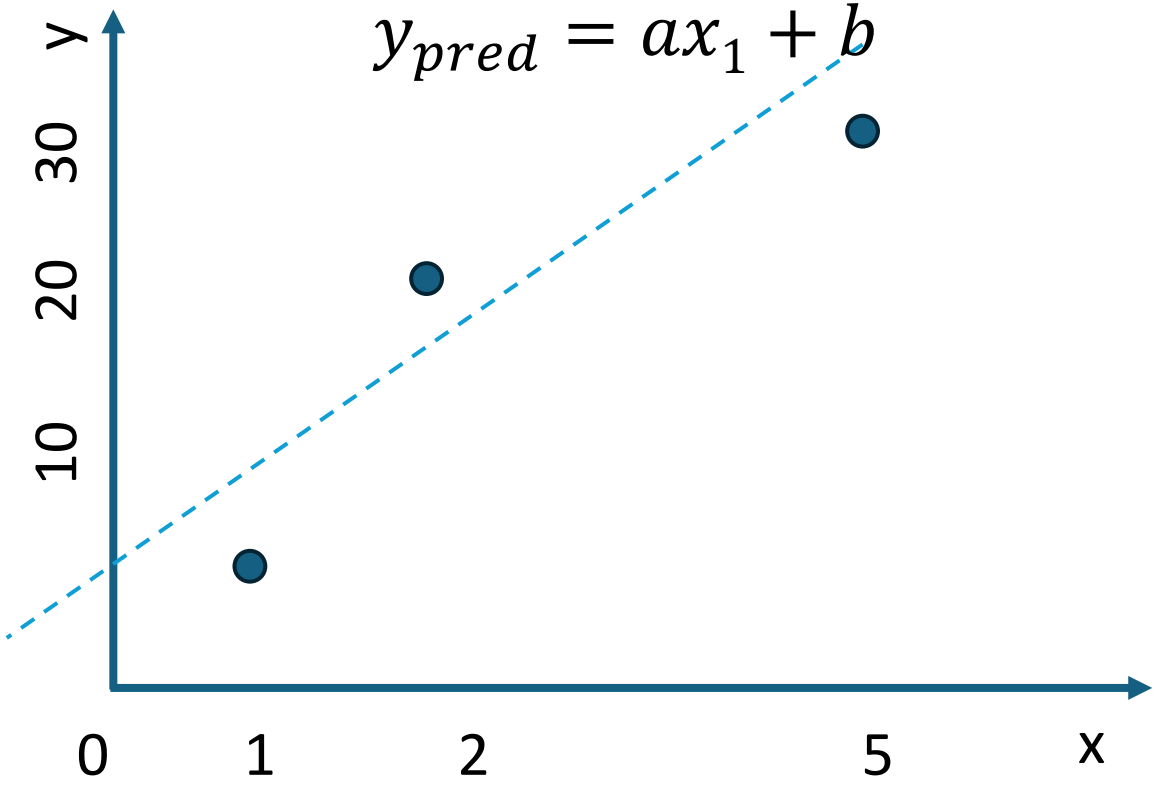
- Objective
- Data
- Simple neural network
- Convolutional network (pre-trained)
- Test on unseen data
- Result visualization

PI

Agenda

- Fundamentals of machine learning (1 hour)
  - Simple example
  - Training and test
  - Understand the results
  - metric
- Feed grading with computer vision part 1 (20 minutes)
- Feed grading with computer vision part 2 (20 minutes)





$$H_1 = x_1 * w_{11}$$
$$H_2 = x_1 * w_{12}$$
$$y_{pred} = x_1 * w_{11} * w_{21} + x_1 * w_{12} * w_{22}$$

x	y
1	10
2	20
5	30

$$y_{pred} = w_{21} * x_1 * w_{11} + w_{22} * x_1 * w_{12}$$

$$loss = (y_{pred} - y_{true})^2$$

$$loss = (y_{true} - x_1 * w_{11} * w_{21} + x_1 * w_{12} * w_{22})^2$$

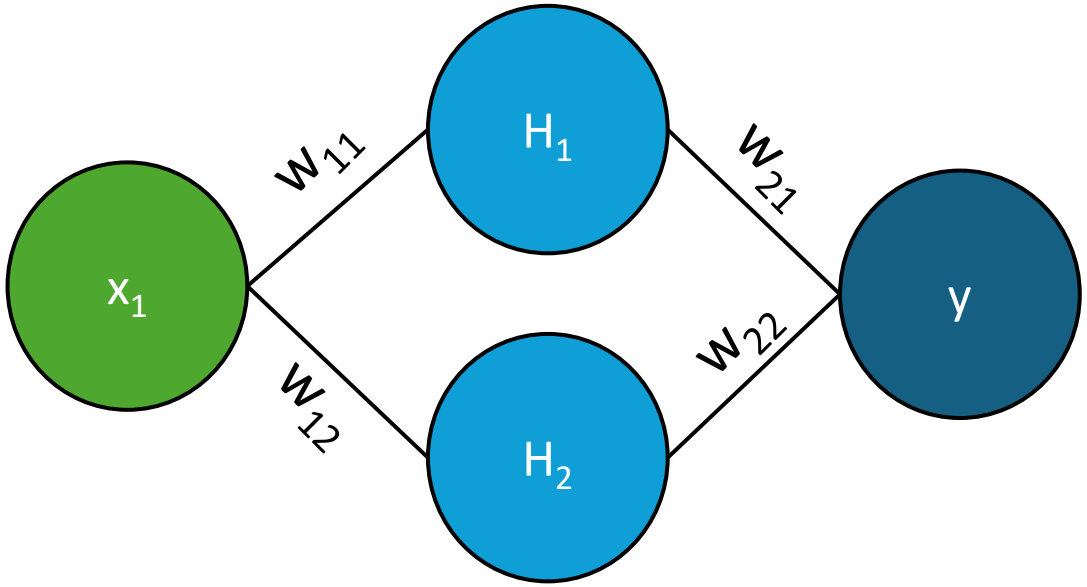
$$= \left(10 - (1 * 1 * 1 + 1 * 1 * w_{22})\right)^2$$

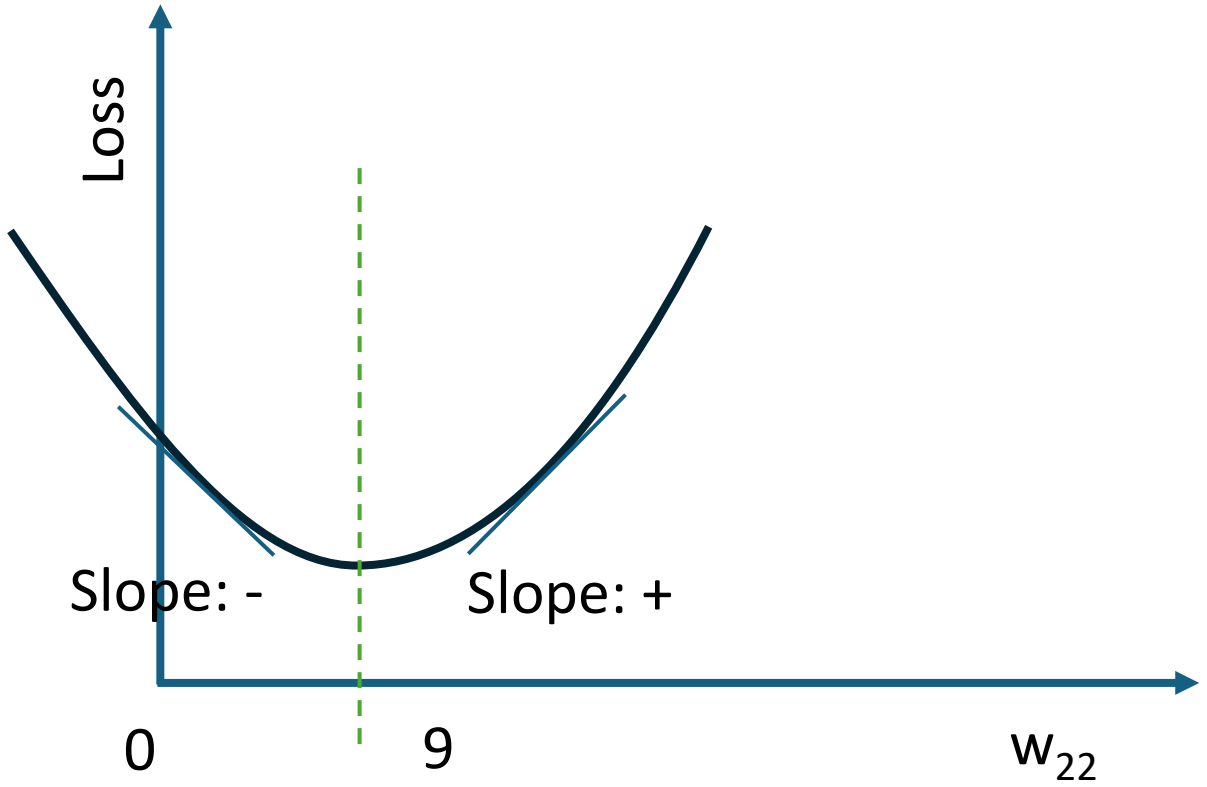
$$= \left(10 - (1 + w_{22})\right)^2$$

$$= (9 - w_{22})^2$$

$$= 91 - 18w_{22} + w_{22}^2$$

x	y
1	10
2	20
5	30



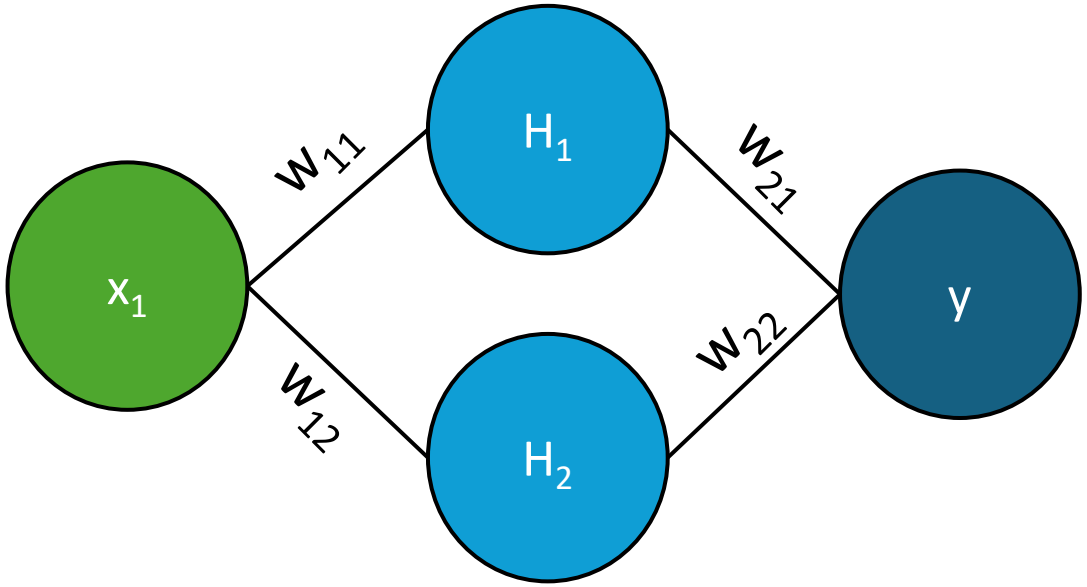


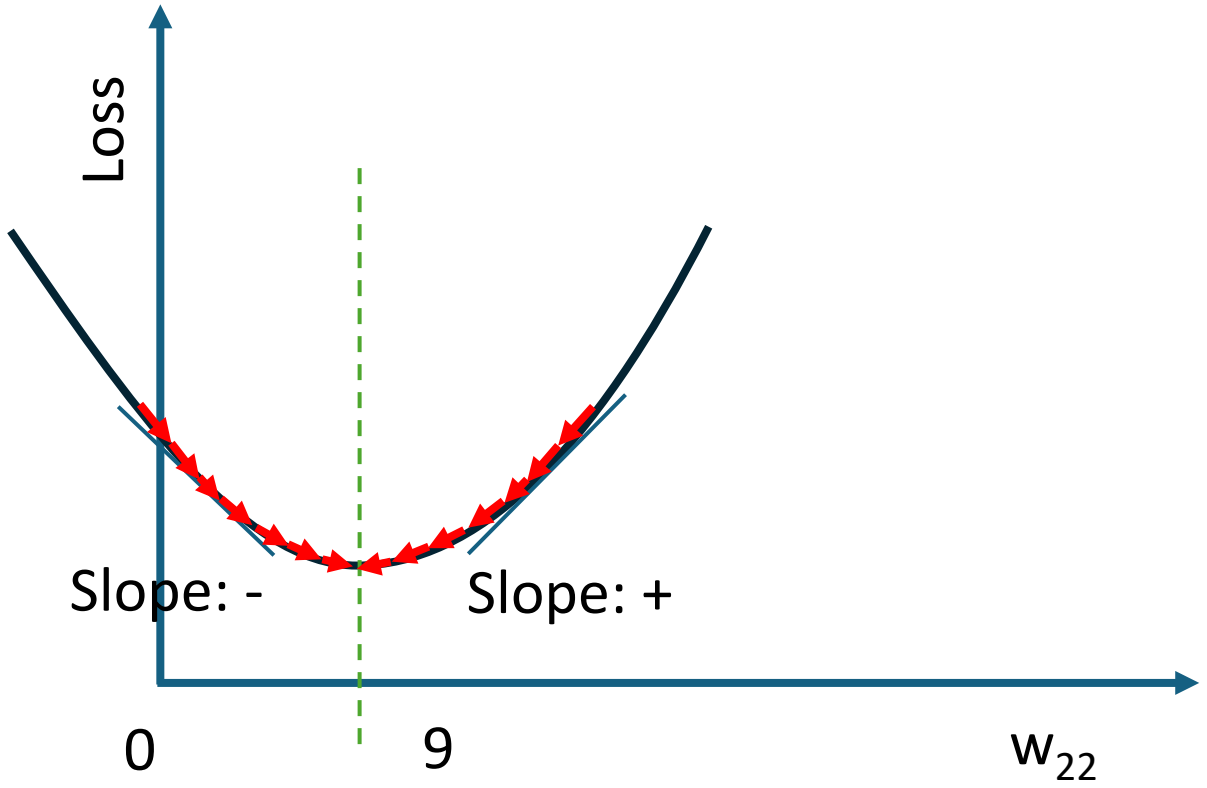
$$loss = 91 - 18w_{22} + w_{22}^2$$

$$slope_{22} = \frac{loss(w_{22} + \Delta) - loss(w_{22})}{\Delta}$$

$$= 2 * w_{22} - 18 = \frac{\partial loss}{\partial w_{22}}$$

x	y
1	10
2	20
5	30

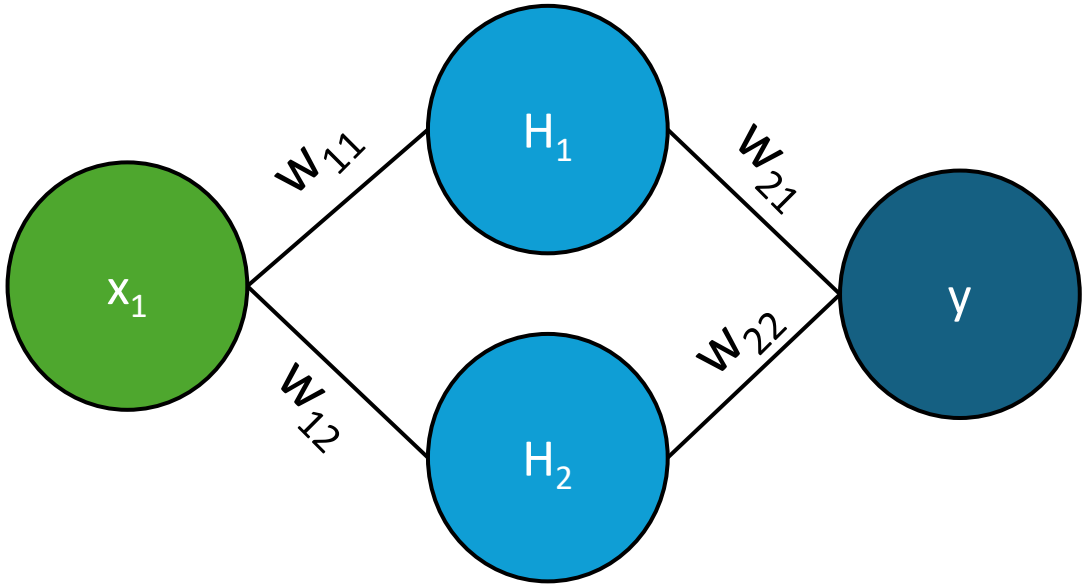


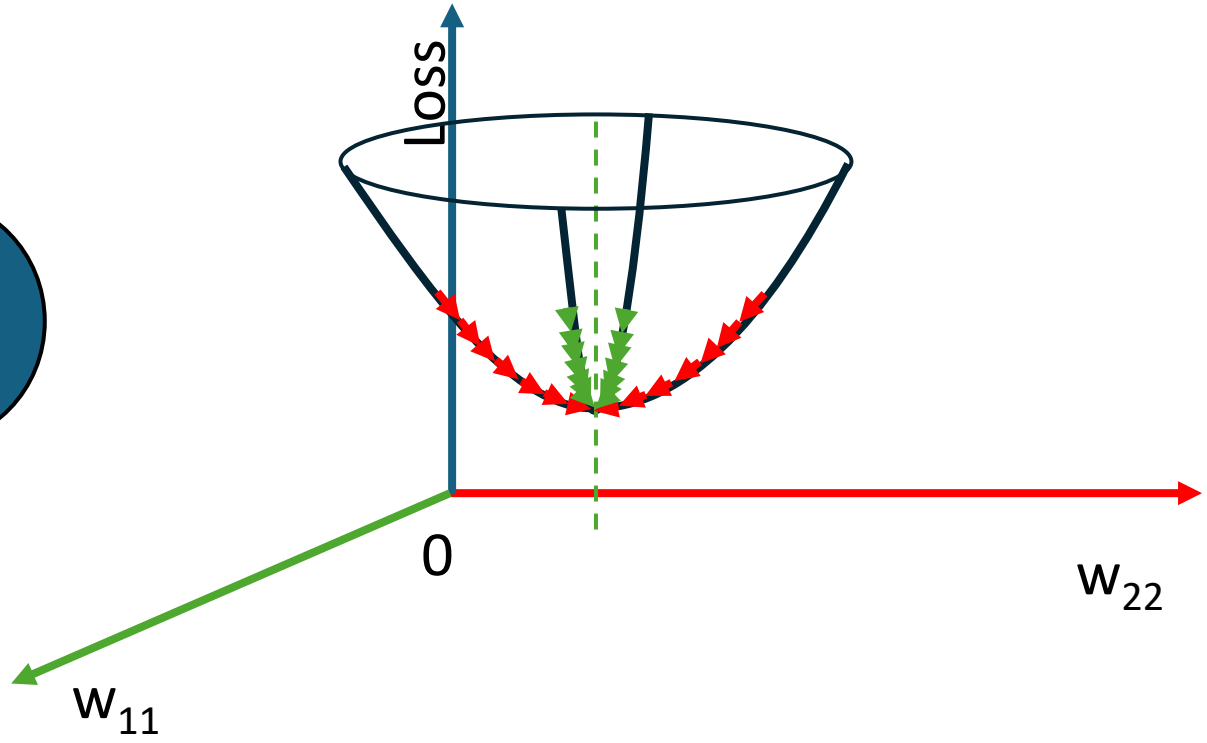


$$w_{22}(new) = w_{22}(old) - slope_{22} * LR$$

$LR\ is\ a\ very\ small\ learning\ rate$
$= 0.0001$

x	y
1	10
2	20
5	30

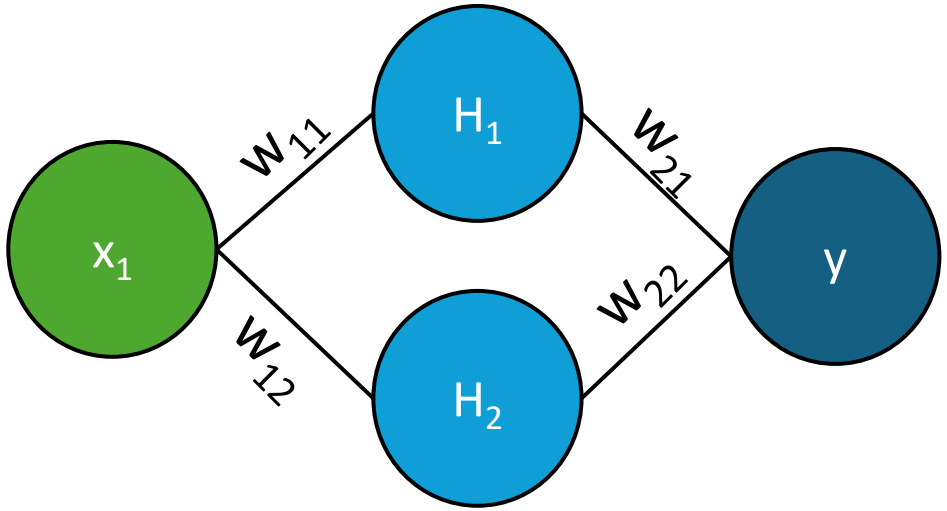


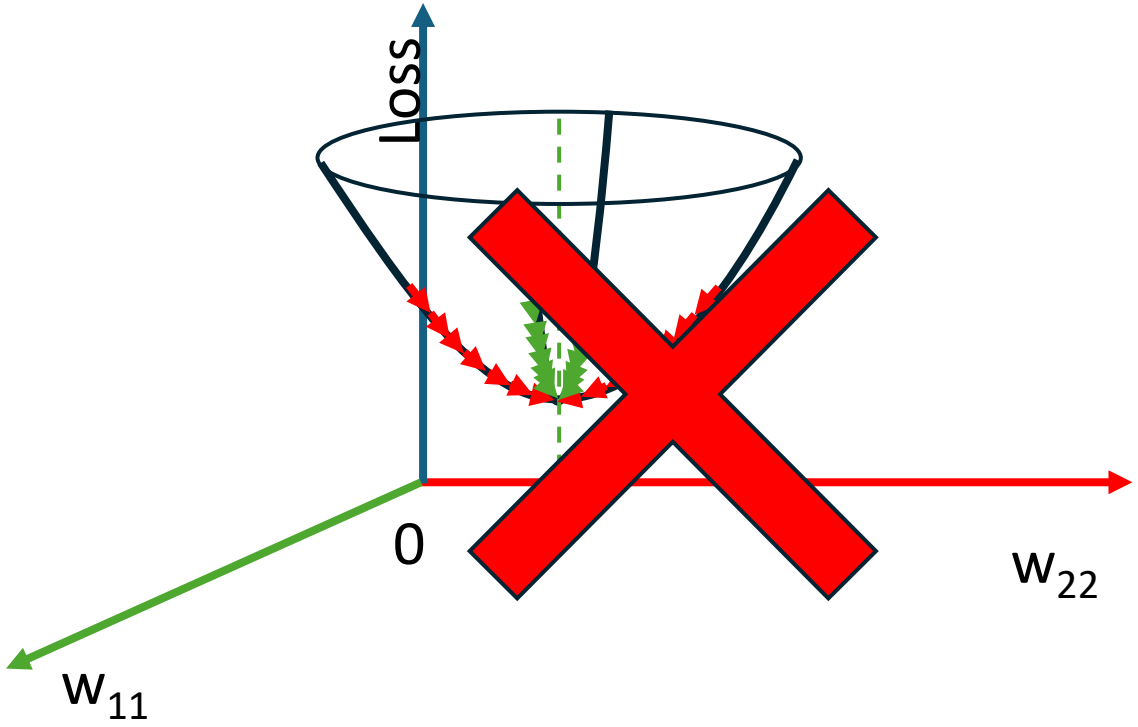


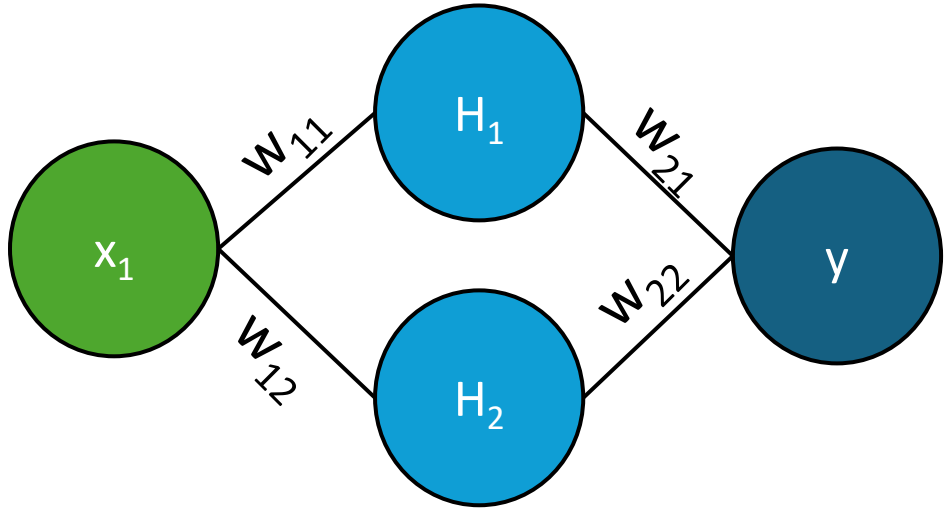
$$w_{22}(new) = w_{22}(old) - slope_{22} * LR$$
$$w_{11}(new) = w_{11}(old) - slope_{11} * LR$$

x	y
1	10
2	20
5	30

$$w_{22}(new) = w_{22}(old) - slope_{22} * LR$$
$$w_{11}(new) = w_{11}(old) - slope_{11} * LR$$
$$w_{12}(new) = w_{12}(old) - slope_{12} * LR$$
$$w_{21}(new) = w_{21}(old) - slope_{21} * LR$$







$$w_{22}(new) = w_{22}(old) - \frac{\partial loss}{\partial w_{22}} * LR$$

$$w_{11}(new) = w_{11}(old) - \frac{\partial loss}{\partial w_{11}} * LR$$

$$w_{12}(new) = w_{12}(old) - \frac{\partial loss}{\partial w_{12}} * LR$$

$$w_{21}(new) = w_{21}(old) - \frac{\partial loss}{\partial w_{21}} * LR$$

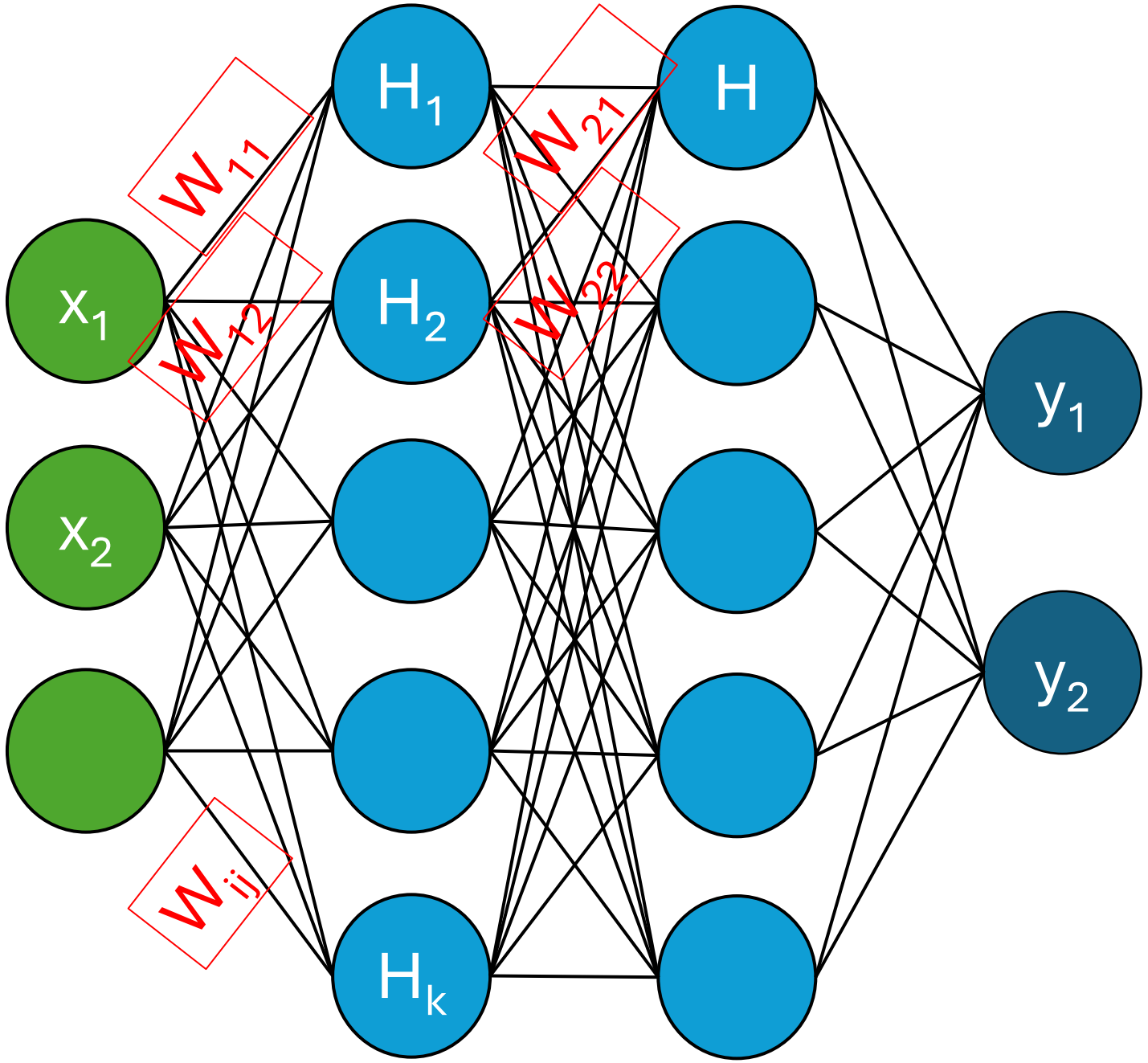
$$
\begin{cases}
w_{22}(new) = w_{22}(old) - \dfrac{\partial loss}{\partial w_{22}} * LR \\
w_{11}(new) = w_{11}(old) - \dfrac{\partial loss}{\partial w_{11}} * LR \\
w_{12}(new) = w_{12}(old) - \dfrac{\partial loss}{\partial w_{12}} * LR \\
w_{21}(new) = w_{21}(old) - \dfrac{\partial loss}{\partial w_{21}} * LR
\end{cases}
$$

$$
\overrightarrow{\boldsymbol{w}\,(\boldsymbol{new})} = \overrightarrow{\boldsymbol{w}\,(\boldsymbol{old})} - \boldsymbol{\nabla} * \boldsymbol{LR}
$$

For each neuron H:

$$H = \sum x_k \, w_{ij} + b$$

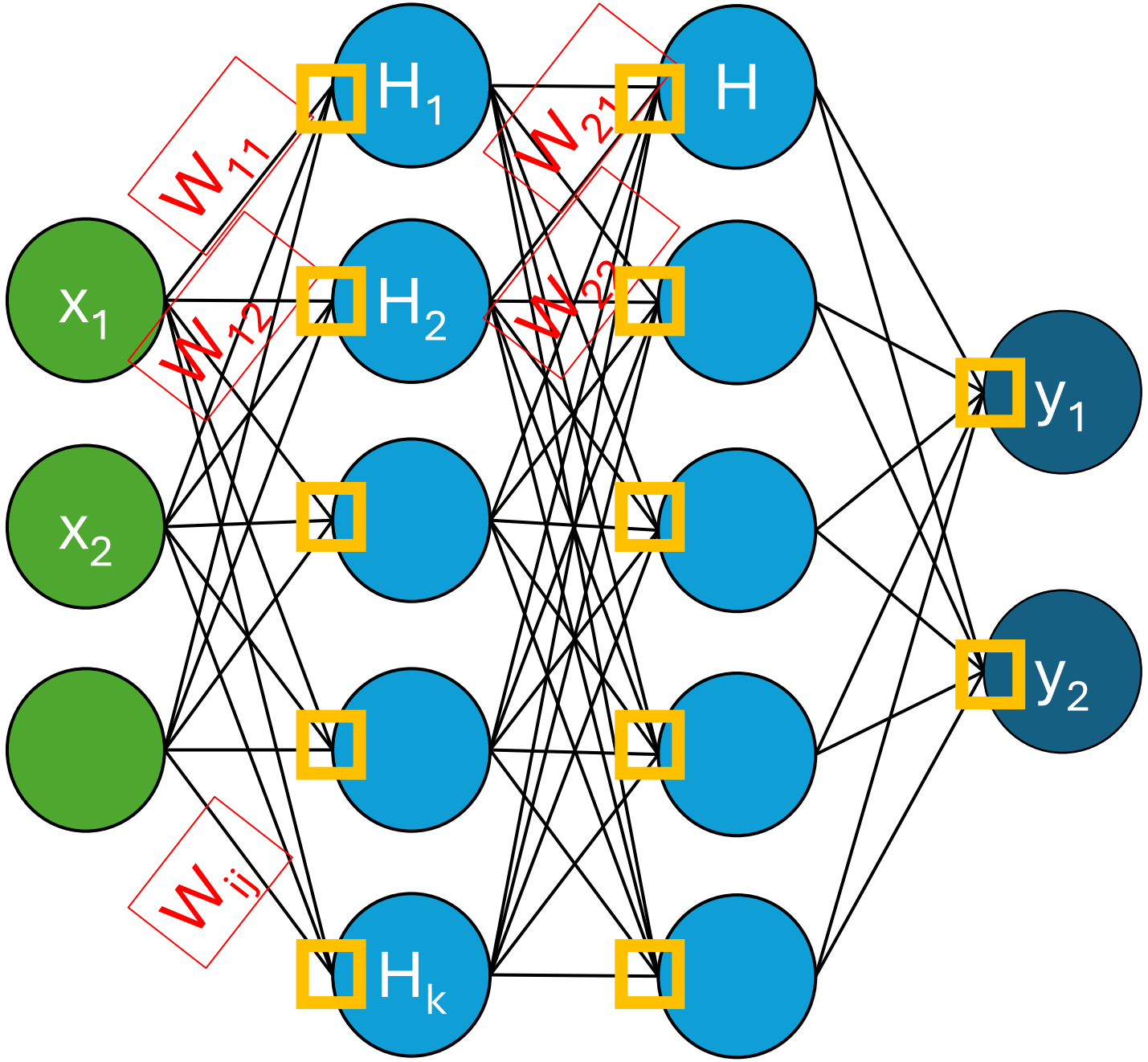
- w_{ij} is the weight
- b is the bias
- A DNN has millions of weights and biases

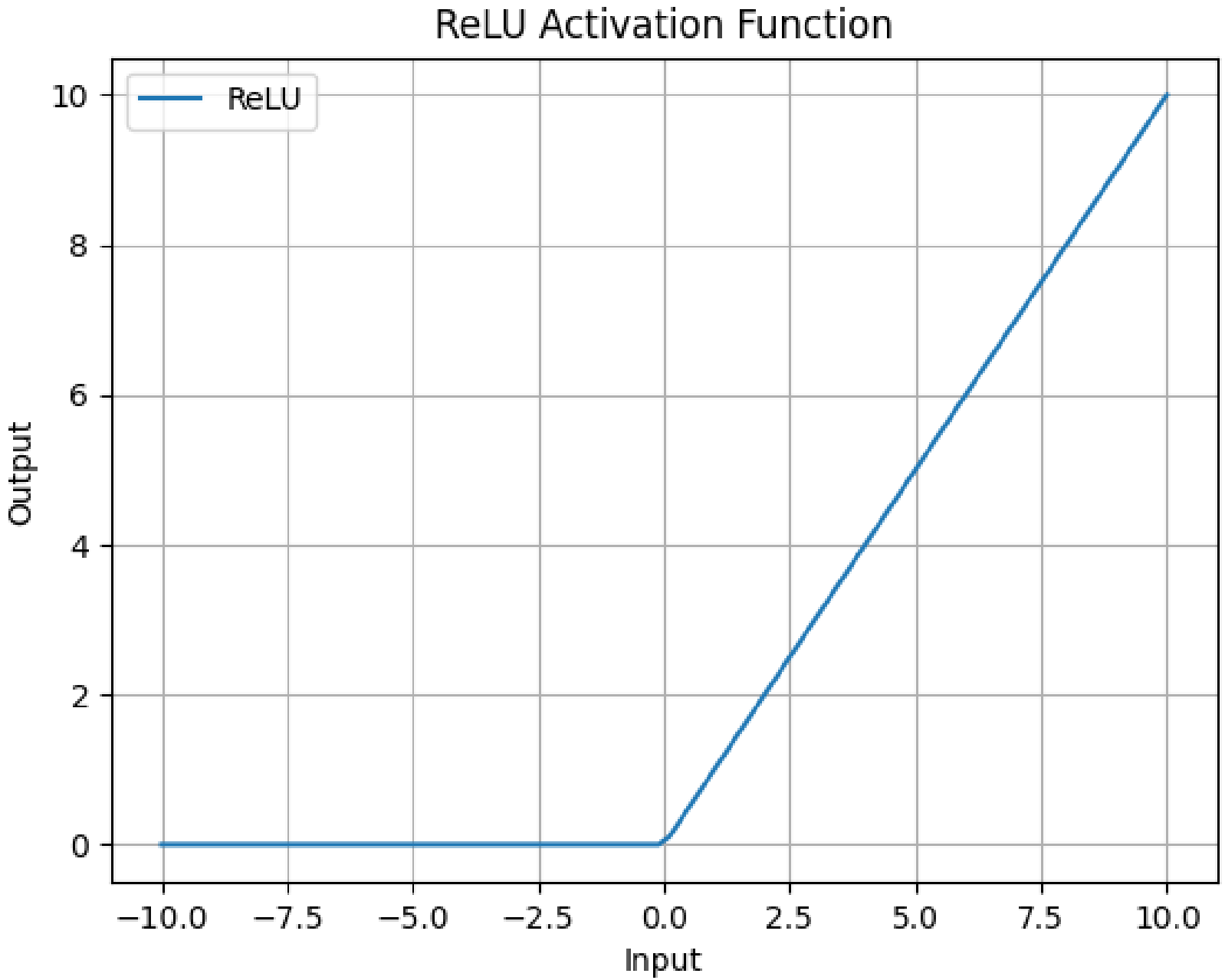


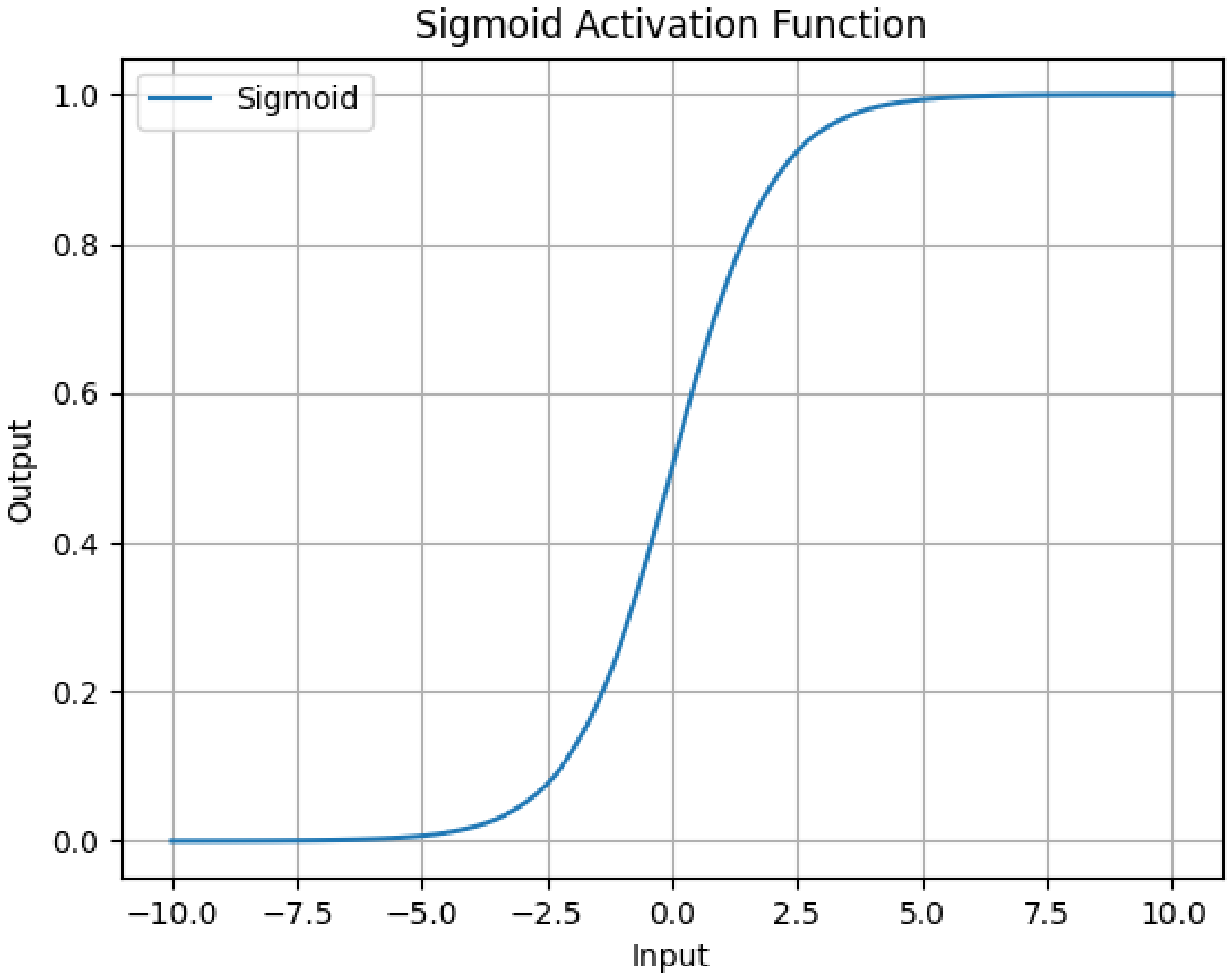
For each neuron H:

$$H = f(\sum x_k w_{ij} + b)$$

- f is the activation function
- w_{ij} is the weight
- b is the bias
- A DNN has millions of weights and biases

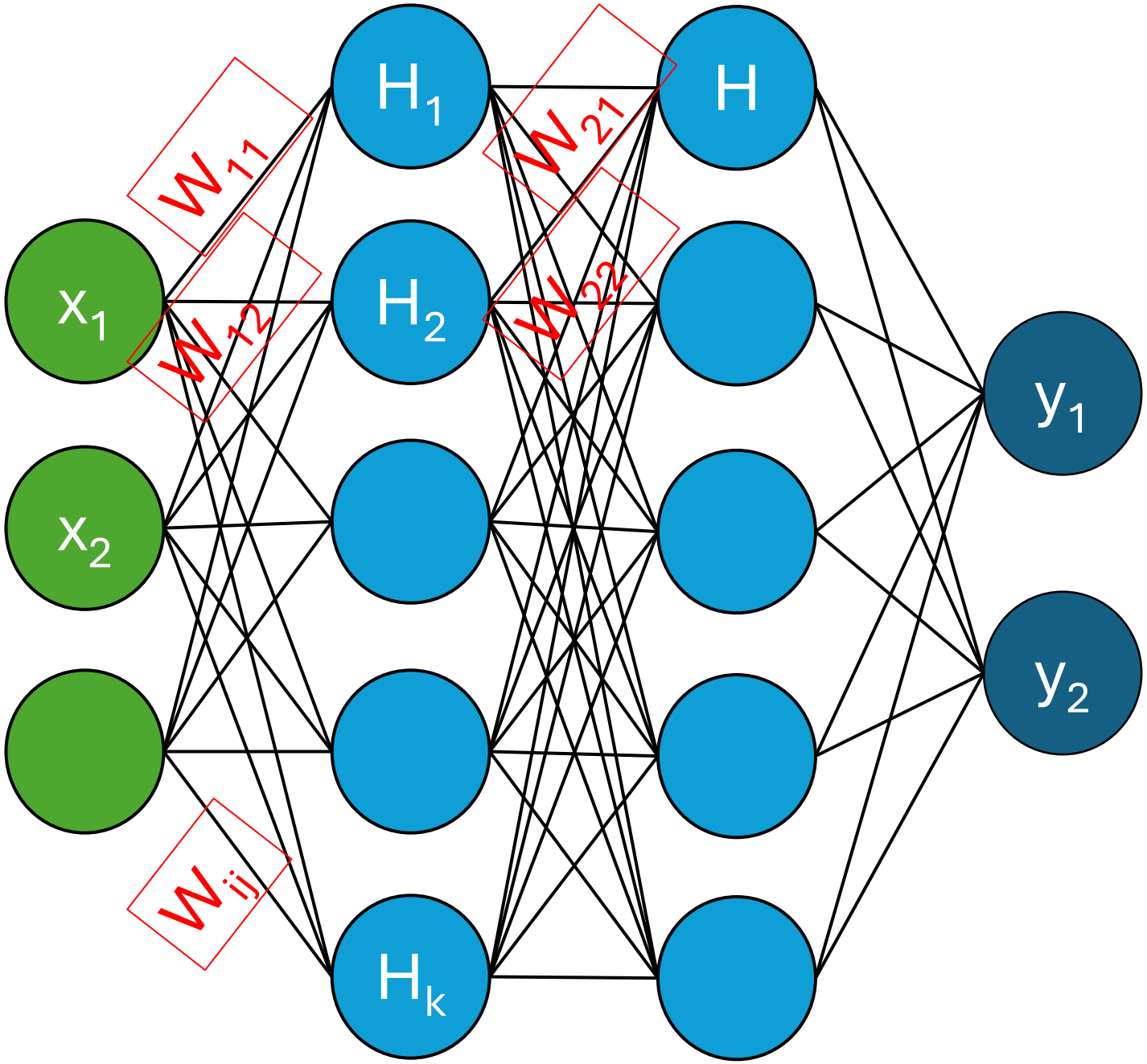



ReLU Activation Function
ReLU
Output
Input
10
8
6
4
2
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Sigmoid Activation Function
Sigmoid
1.0
0.8
0.6
0.4
0.2
0.0
Output
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Input

Output types

- Regression
- Classification
- Feature extraction
- Combination



y_{pred} y_{true}

$\begin{bmatrix} 110 \\ 5.1 \end{bmatrix}$ $\begin{bmatrix} 150 \\ 7.1 \end{bmatrix}$

Regression loss

- Mean absolute error

$$MAE = \frac{1}{n}|y_{pred} - y_{true}|$$

- Mean square error

$$MSE = \frac{1}{n}\left(y_{pred} - y_{true}\right)^2$$

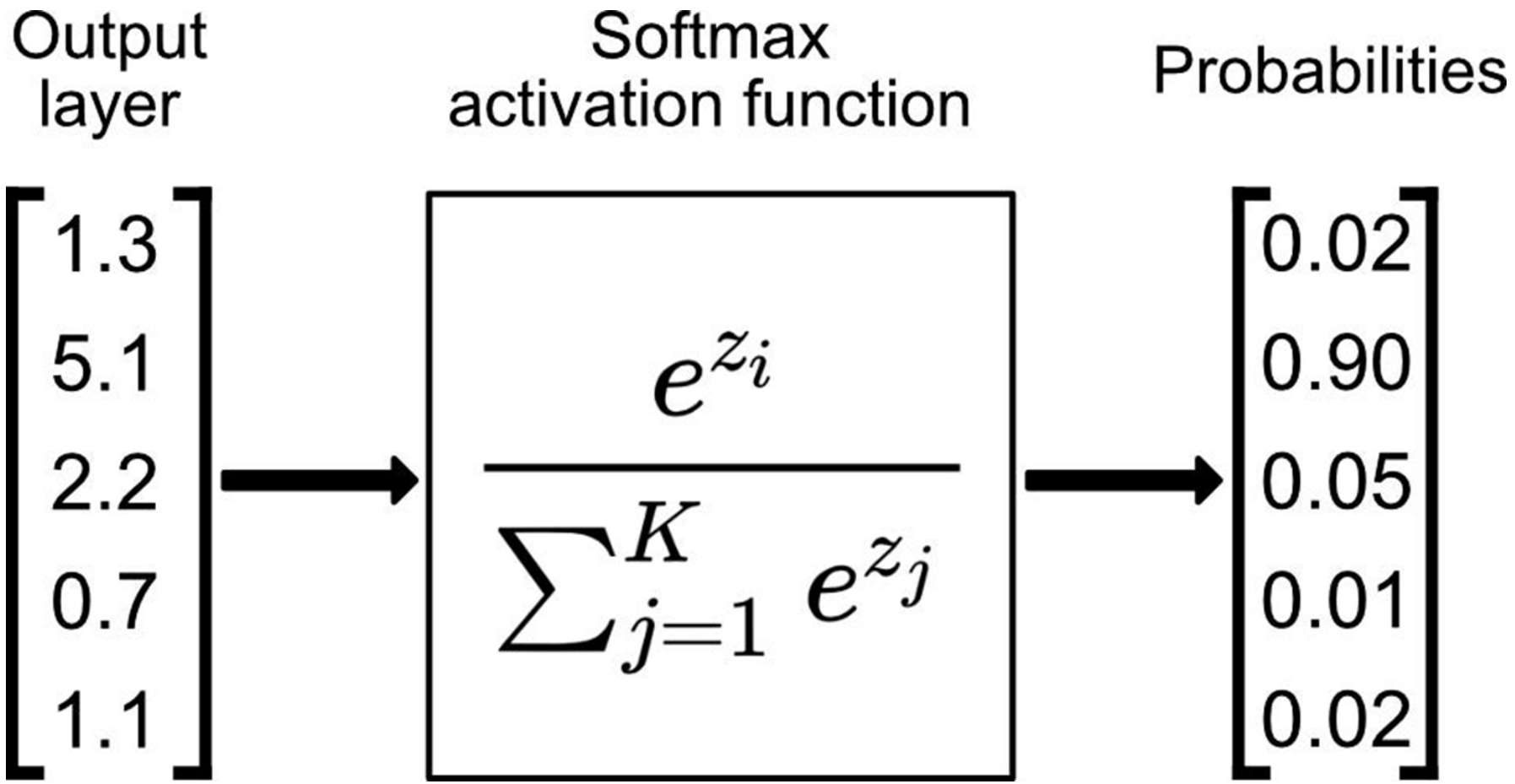
One hot encoding for y_{true}

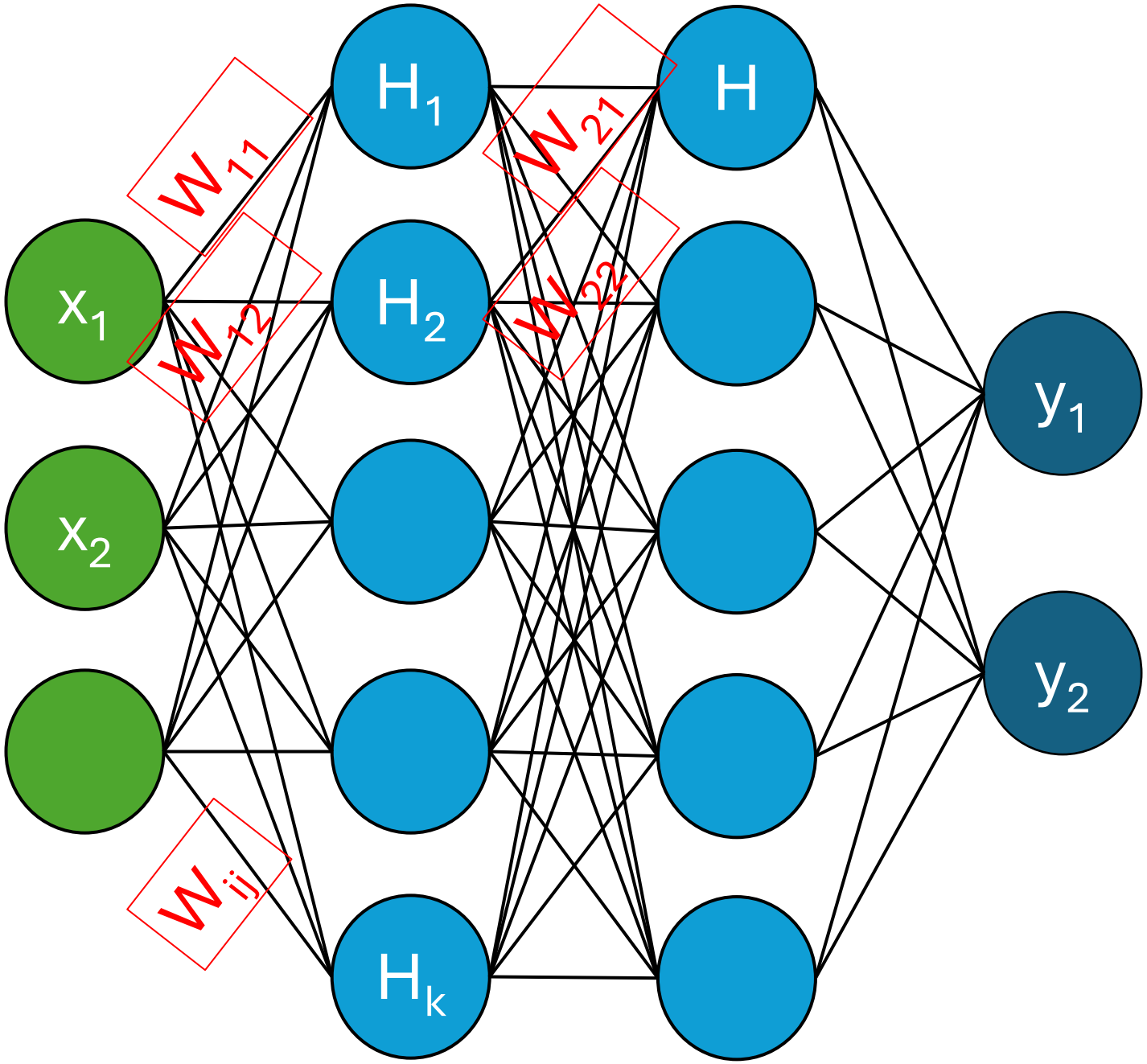
Chicken
Banana

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Softmax for y_pred





y_{pred} y_{true}

$\begin{bmatrix} 0.97 \\ 0.03 \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

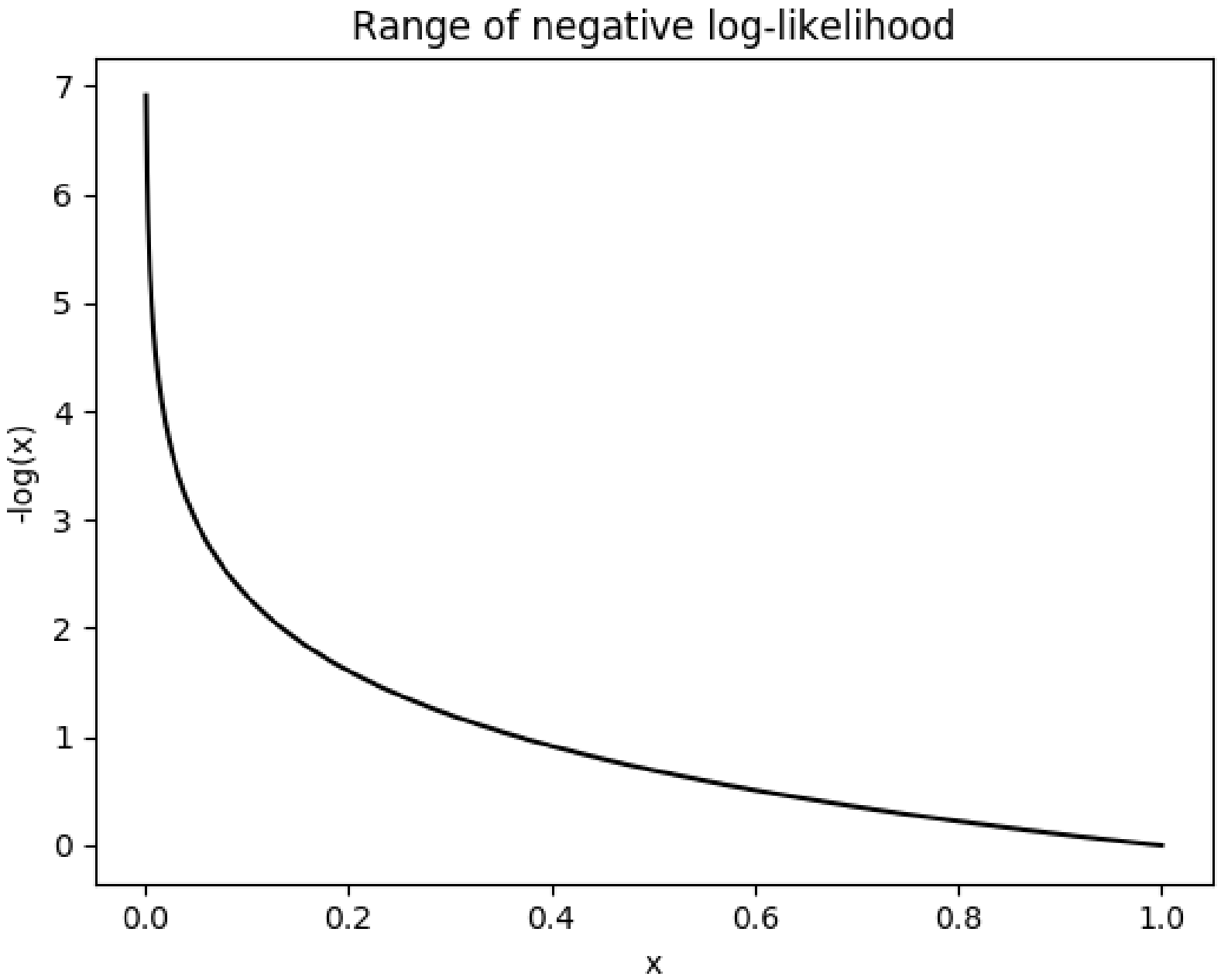
Cross entropy loss for classification

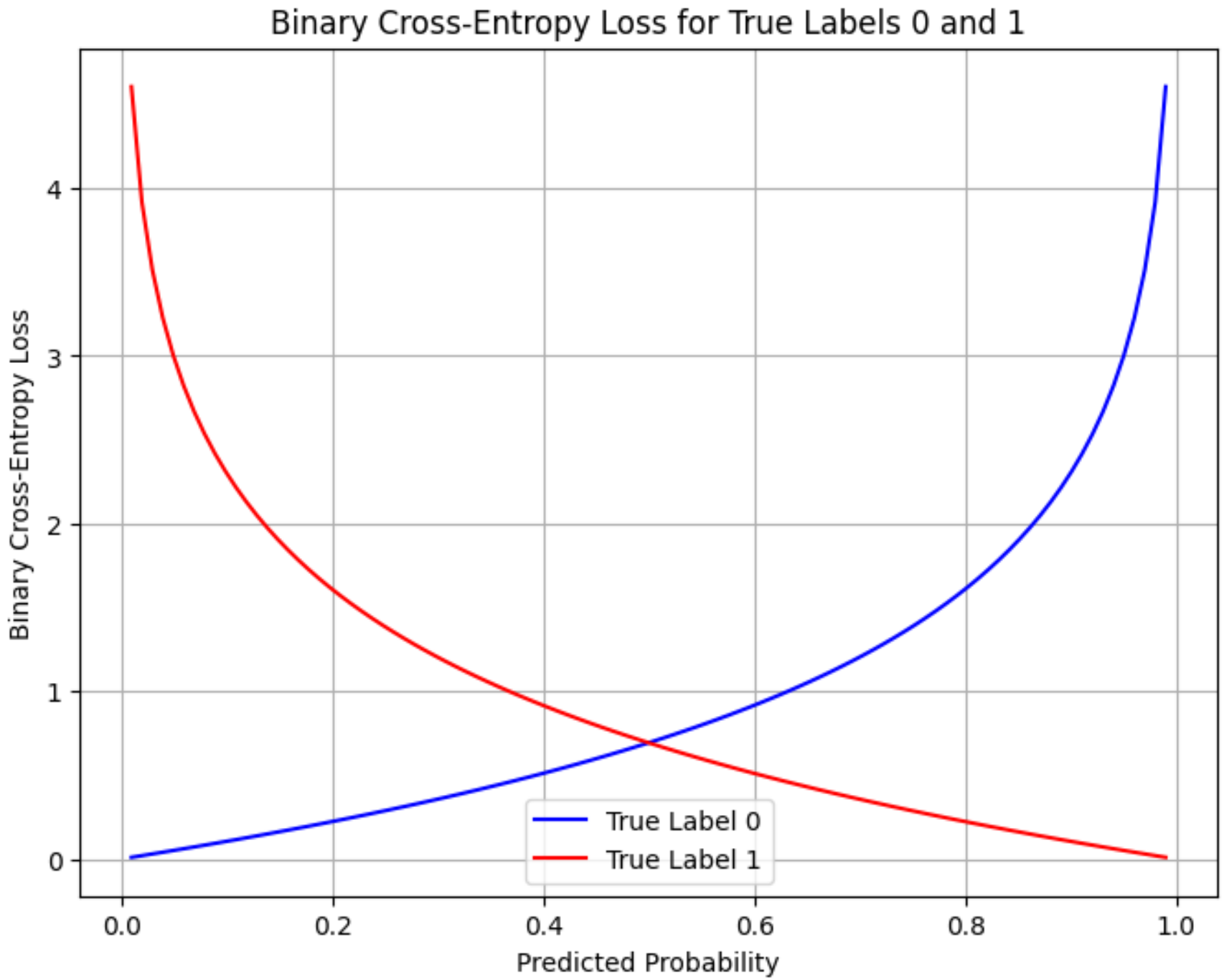
- Binary cross entropy
  - After softmax

$$BCE = -\sum y_{true} * \log(y_{pred}) + (1 - y_{true}) * \log(1 - y_{pred})$$

- Multi class entropy
  - After softmax

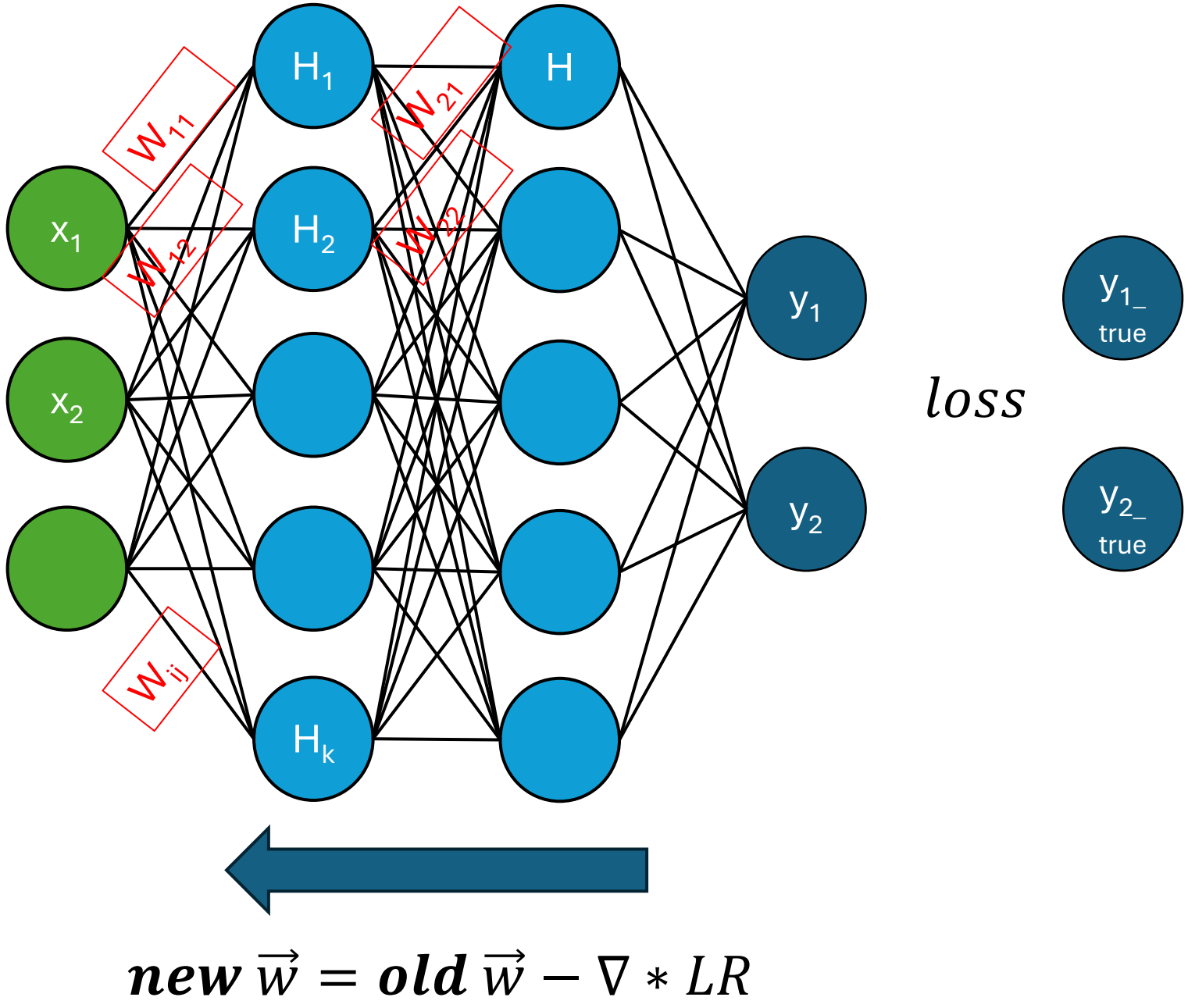
$$CE = -\sum y_{true} * \log(y_{pred})$$


Range of negative log-likelihood
-log(x)
x
0
1
2
3
4
5
6
7
0.0
0.2
0.4
0.6
0.8
1.0


Binary Cross-Entropy Loss for True Labels 0 and 1
Binary Cross-Entropy Loss
Predicted Probability
True Label 0
True Label 1
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0

Weight update

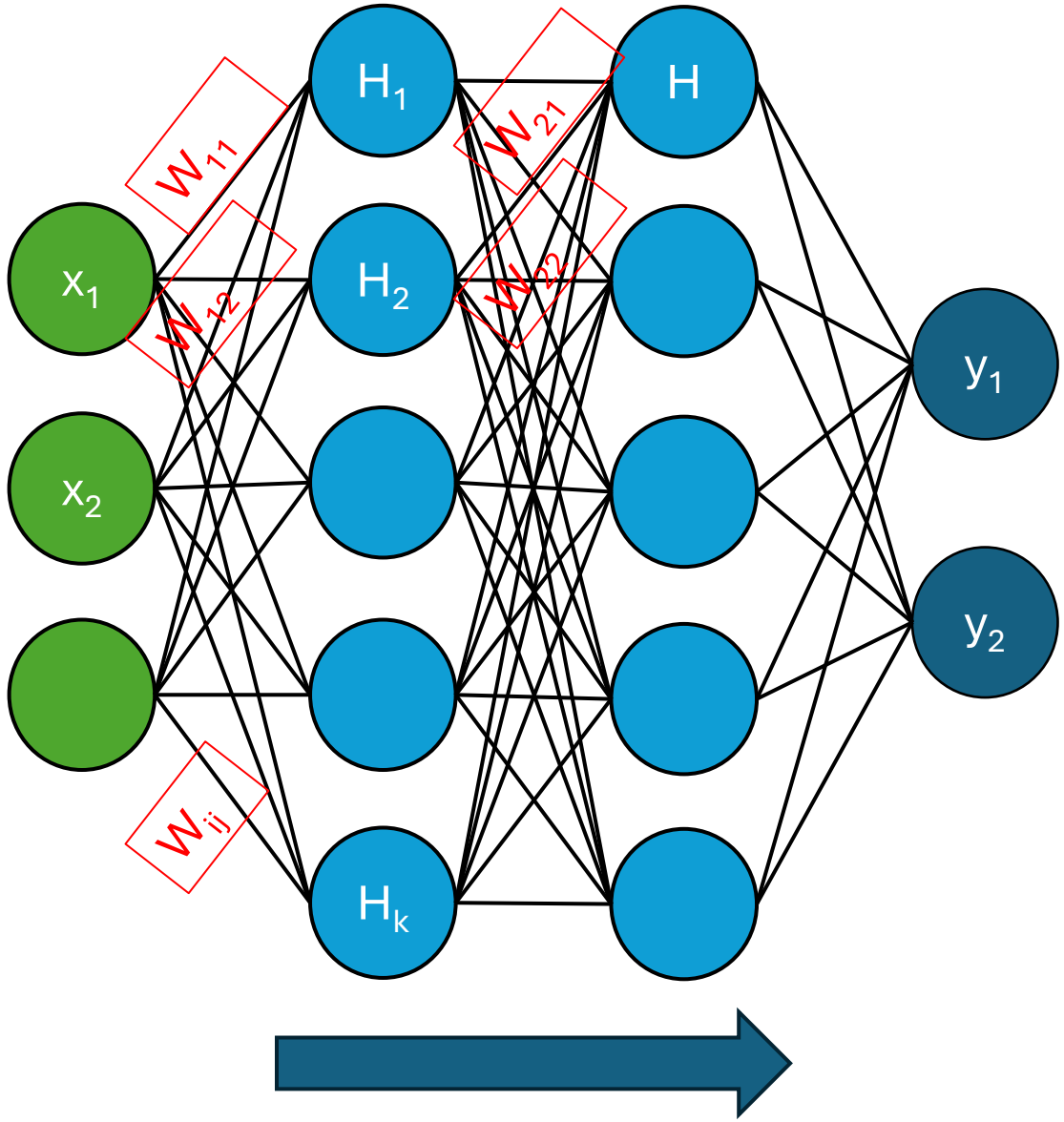
- Learn
- Fit
- Train
- Back propagation



$$\boldsymbol{new}\ \vec{w} = \boldsymbol{old}\ \vec{w} - \nabla * LR$$

Use the model

- Forward pass
- Inference
- Test
- Validation
- Deployment



$$NN(\vec{x}) = \vec{y}$$

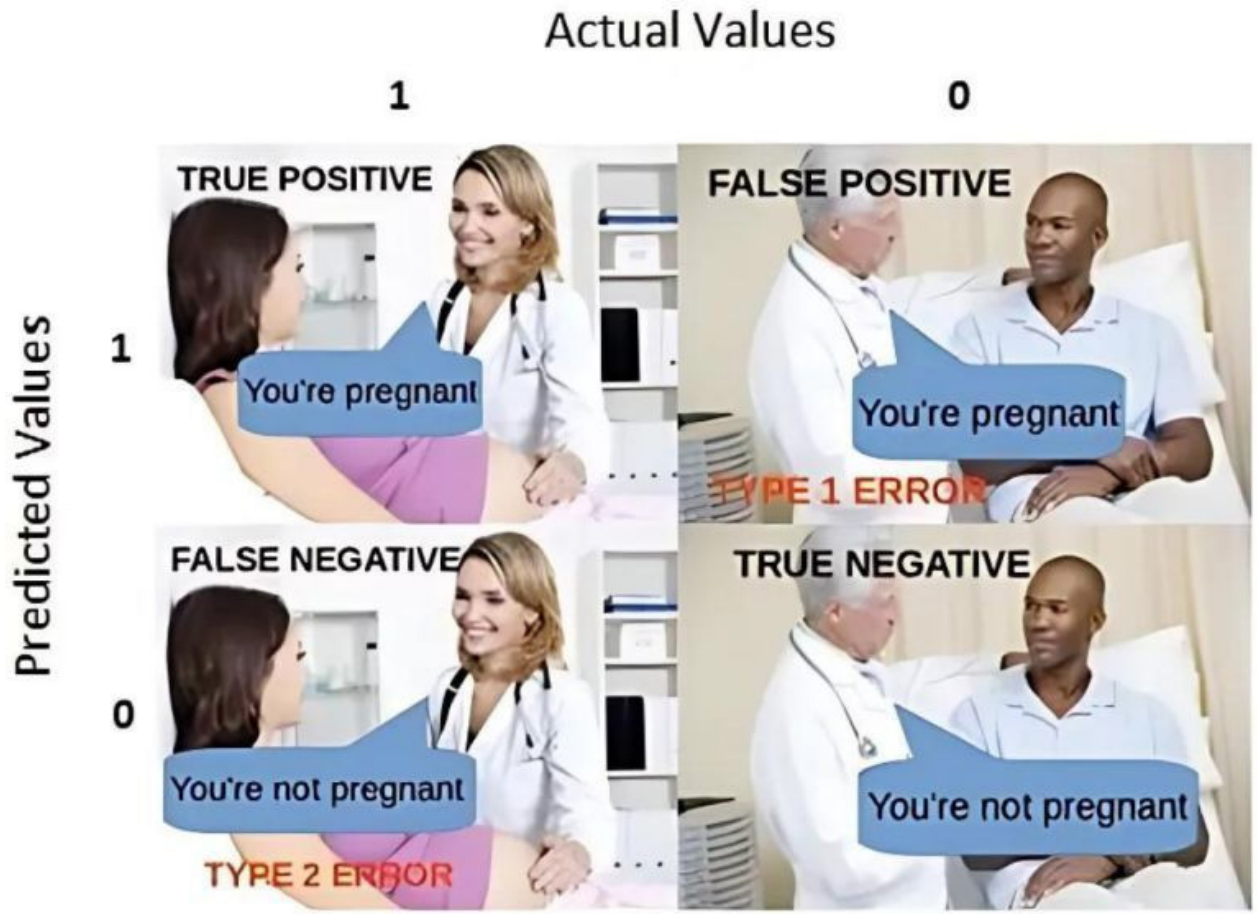
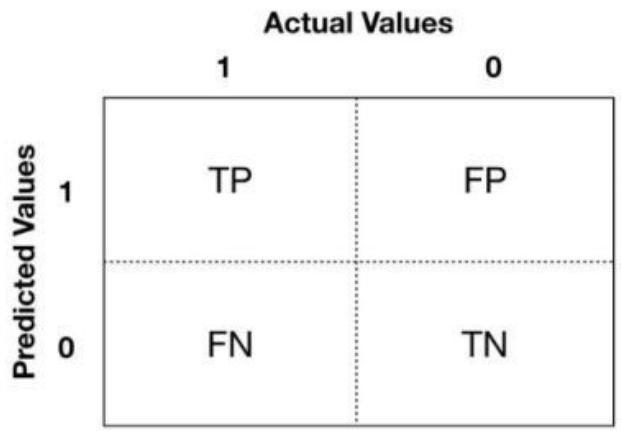


Image Credit: Linkedin.com

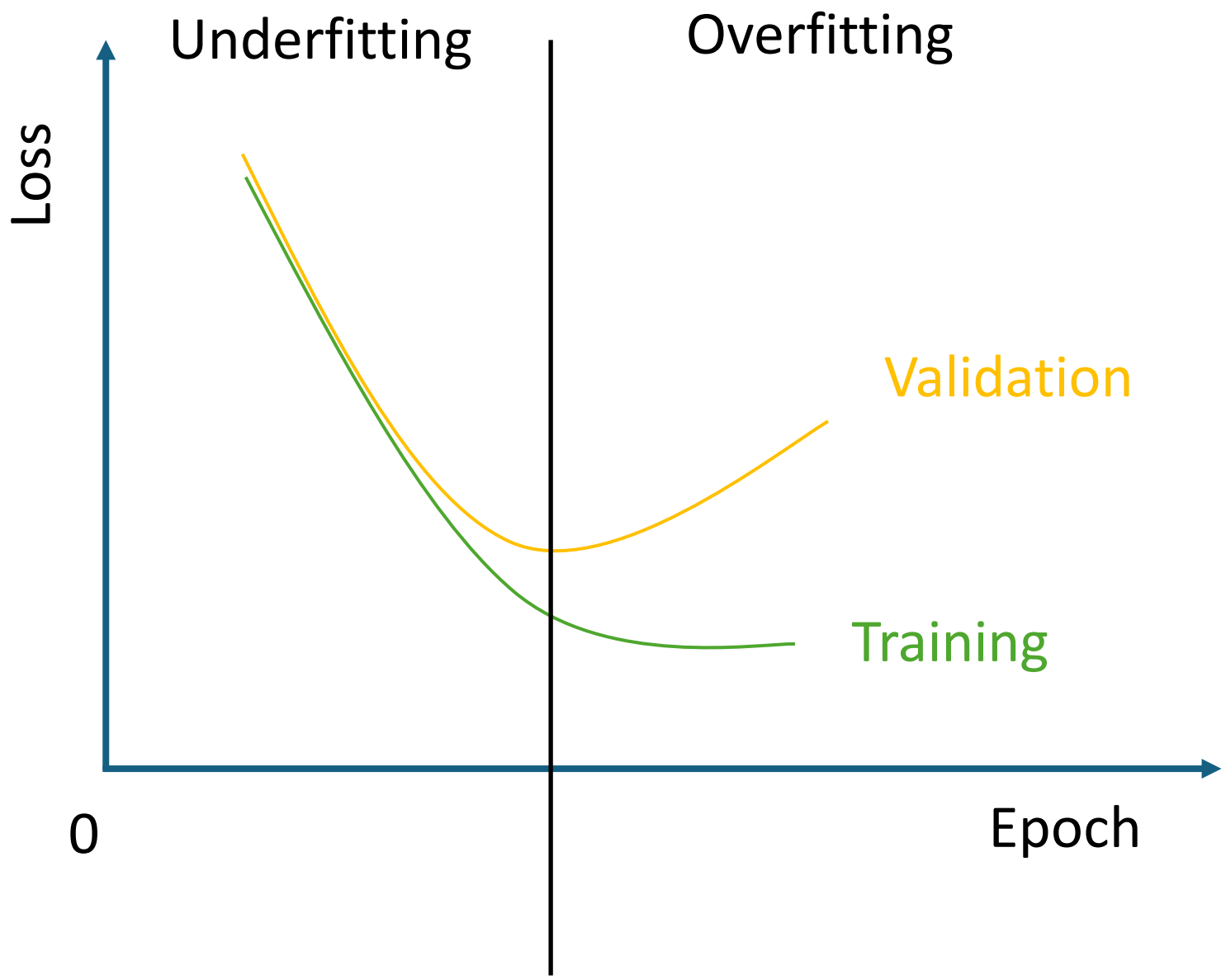


$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

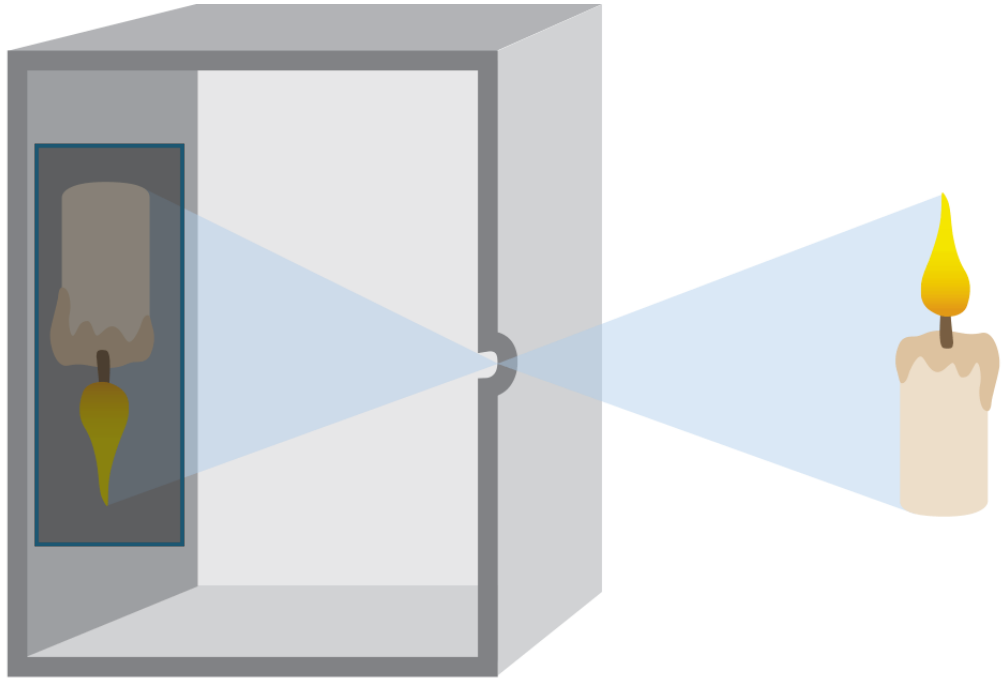
$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

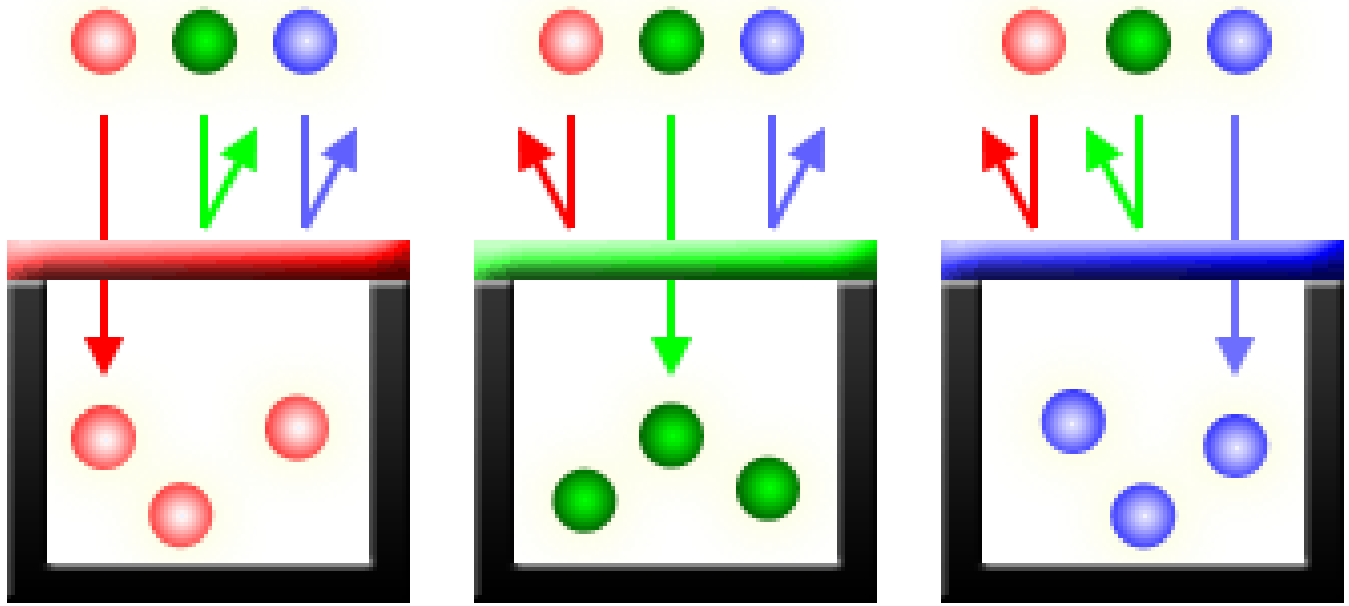

Underfitting
Overfitting
Loss
Validation
Training
0
Epoch

Computer vision

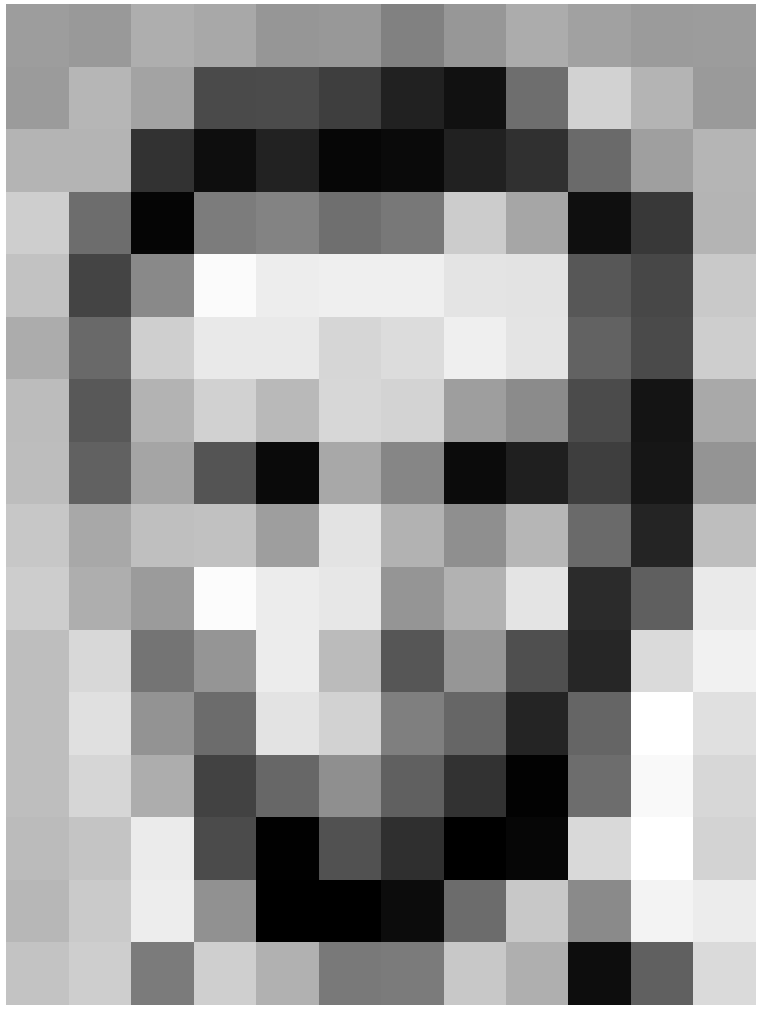

X000URJVYF
MADE IN CHINA

- ❑Pinhole principle
- ❑Traditional film
- ❑Digital sensors (CCD and CMOS)
- ❑Red Green Blue (RGB) channels





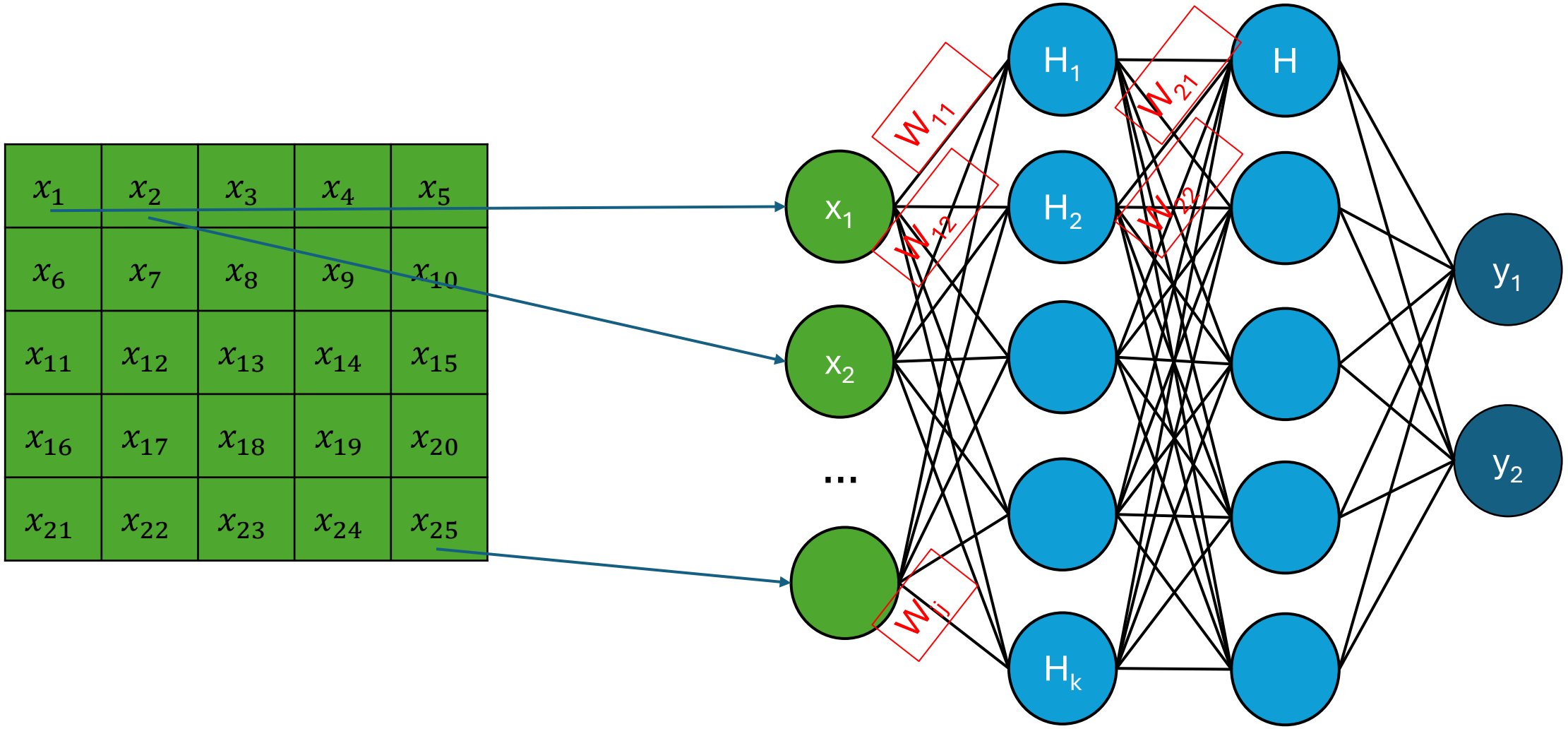
Grey Scale



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Image of Abraham Lincoln as a matrix of pixel values. Image Credit: Thomas Smits


x_1 x_2 x_3 x_4 x_5
x_6 x_7 x_8 x_9 x_{10}
x_{11} x_{12} x_{13} x_{14} x_{15}
x_{16} x_{17} x_{18} x_{19} x_{20}
x_{21} x_{22} x_{23} x_{24} x_{25}
x1
x2
...
H1
H2
Hk
H
W11
W12
Wij
W21
W22
y1
y2

Howdy | 25 SPRING ICPE 642 6 | computer_vision_for_fe | Central Authentication | 24A ICPE 639 601/701 | -log(x) | CrossEntropyLoss — P

colab.research.google.com/github/TAMIDSpiyalong/Decision-Science-for-Sustainable-Livestock-Systems/blob/main/computer_vision_for_feed_grading.ipynb#scrollTo=7NkWA-0kVou8

Howdy | Home - Workday | My Drive - Google... | SciENcv - Home | Research.gov - Hom... | Welcome | Login.gov | Indect IVIS | Natural Language P... | Faculty Job Board | E&M Supplemental... | My Benefit Coverage

computer_vision_for_feed_grading.ipynb | Save in GitHub to keep changes

File Edit View Insert Runtime Tools Help

Share | Gemini | Y

Commands | + Code | + Text | Copy to Drive | T4 | RAM | Disk

Files

Analyze your files with code written by Gemini | Upload

..
.config
RANDOM_PICTURES_TO_VERI...
mixture
S00
S05
S10
S20
S30
S40
sample_data
Confusing Matrix.pdf
RANDOM_PICTURES_TO_VERIF...
best_model.pth
mixture.zip
precision_recall_table.csv

Disk 74.28 GB available

Load Images

torchvision.datasets.ImageFolder is a PyTorch class that simplifies loading image datasets by organizing them into a directory structure where each subdirectory represents a class or category of images. In practice, it is more complicated but doable with care.

[4] 0s
```
images = torchvision.datasets.ImageFolder('mixture/')
```

[32] 0s
```
images
```
```
Dataset ImageFolder
    Number of datapoints: 183
    Root location: mixture/
    StandardTransform
Transform: Compose(
               Resize(size=(128, 128), interpolation=bilinear, max_size=None, antialias=True)
               ToTensor()
               Grayscale(num_output_channels=1)
           )
```

0s
```
images[4][1]
```
```
0
```

0s
```
print(np.array(images[4][0]).shape)
```
```
(450, 500, 3)
```

Image Transformation

PyTorch transformations are crucial for preparing datasets before training machine learning models. They ensure standardization and normalization, which scale data points uniformly to prevent features with large ranges from dominating the training process. Transformations also enable data augmentation through techniques like rotation, flipping, and cropping, helping prevent overfitting and improve model

0s completed at 2:22 PM

2:23 PM 3/27/2025

Howdy | 25 SPRING ICPE 642 6 | computer_vision_for_fe | Central Authentication | 24A ICPE 639 601/701 | -log(x) | CrossEntropyLoss — P

colab.research.google.com/github/TAMIDSpiyalong/Decision-Science-for-Sustainable-Livestock-Systems/blob/main/computer_vision_for_feed_grading.ipynb#scrollTo=69ziB2mQawVO

Howdy | Home - Workday | My Drive - Google... | SciENcv - Home | Research.gov - Hom... | Welcome | Login.gov | Indect IVIS | Natural Language P... | Faculty Job Board | E&M Supplemental... | My Benefit Coverage

computer_vision_for_feed_grading.ipynb | Save in GitHub to keep changes

File Edit View Insert Runtime Tools Help

Share | Gemini | Y

Commands | + Code | + Text | Copy to Drive | T4 | RAM | Disk

Files

Analyze your files with code written by Gemini | Upload

- ..
- .config
- RANDOM_PICTURES_TO_VERI...
- mixture
  - S00
  - S05
  - S10
  - S20
  - S30
  - S40
- sample_data
- Confusing Matrix.pdf
- RANDOM_PICTURES_TO_VERIF...
- best_model.pth
- mixture.zip
- precision_recall_table.csv

Disk 74.28 GB available

Simple Neural Network

```python
import torch.nn as nn

class SimpleModel(nn.Module):
    def __init__(self):
        super(SimpleModel, self).__init__()
        self.fc1 = nn.Linear(16384, 1000)
        self.fc2 = nn.Linear(1000, 100)
        self.fc3 = nn.Linear(100, 6)    # Output layer is the same dimension as the classes

    def forward(self, x):
        x = x.view(-1, 16384)  # Flatten the input
        x = torch.relu(self.fc1(x))  # Activation function for hidden layer
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

model = SimpleModel().to(device)

# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.0001, momentum=0.9)
```

Loading...

Training

0s completed at 2:22 PM

2:33 PM 3/27/2025

Howdy | 25 SPRING ICPE 642 6 | computer_vision_for | Central Authentication | 24A ICPE 639 601/701 | -log(x) | CrossEntropyLoss — P

colab.research.google.com/github/TAMIDSpiyalong/Decision-Science-for-Sustainable-Livestock-Systems/blob/main/computer_vision_for_feed_grading.ipynb#scrollTo=ZUN0hlhNflQQ

Howdy | Home - Workday | My Drive - Google... | SciENcv - Home | Research.gov - Hom... | Welcome | Login.gov | Indect IVIS | Natural Language P... | Faculty Job Board | E&M Supplemental... | My Benefit Coverage

computer_vision_for_feed_grading.ipynb | Save in GitHub to keep changes

File Edit View Insert Runtime Tools Help

Share | Gemini | Y

Commands | + Code | + Text | Copy to Drive | T4 | RAM | Disk

Files

Analyze your files with code written by Gemini | Upload

..
.config
RANDOM_PICTURES_TO_VERI...
mixture
sample_data
Confusing Matrix.pdf
RANDOM_PICTURES_TO_VERIF...
best_model.pth
mixture.zip
precision_recall_table.csv

Disk 74.28 GB available

Training

39s

```python
since = time.time()

best_acc = 0.0
num_epochs=50

for epoch in range(num_epochs):
    print('-' * 10)

    model.train()  # Set model to training mode
    train_running_loss = 0.0
    train_running_corrects = 0

    for inputs, labels in train_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        # backward + optimize only if in training phase
        loss.backward()
        optimizer.step()

```

Connected to Python 3 Google Compute Engine backend (GPU)

2:41 PM 3/27/2025

computer_vision_for_feed_grading.ipynb Save in GitHub to keep changes

File Edit View Insert Runtime Tools Help

Share Gemini

Commands | + Code | + Text | Copy to Drive T4 RAM Disk

Table of contents

Train a Model with Pretrained Weights

We need to modigy the original pre-defined model becuase our input channel is only one. Also the output classes of the original resnet is 1000 whereas we only want 6 classes.

```
1
model = torchvision.models.resnet18(pretrained=True)

num_ftrs = model.fc.in_features
# The last layer should have len(class_names neruons)
model.fc = nn.Linear(num_ftrs, len(class_names))

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)

model.conv1 = nn.Conv2d(1, 64, kernel_size=7, stride=2, padding=3,bias=False)

model = model.to(device)

```

```
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be rem
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are dep
  warnings.warn(msg)
```

[18]
```
Start coding or generate with AI.
```

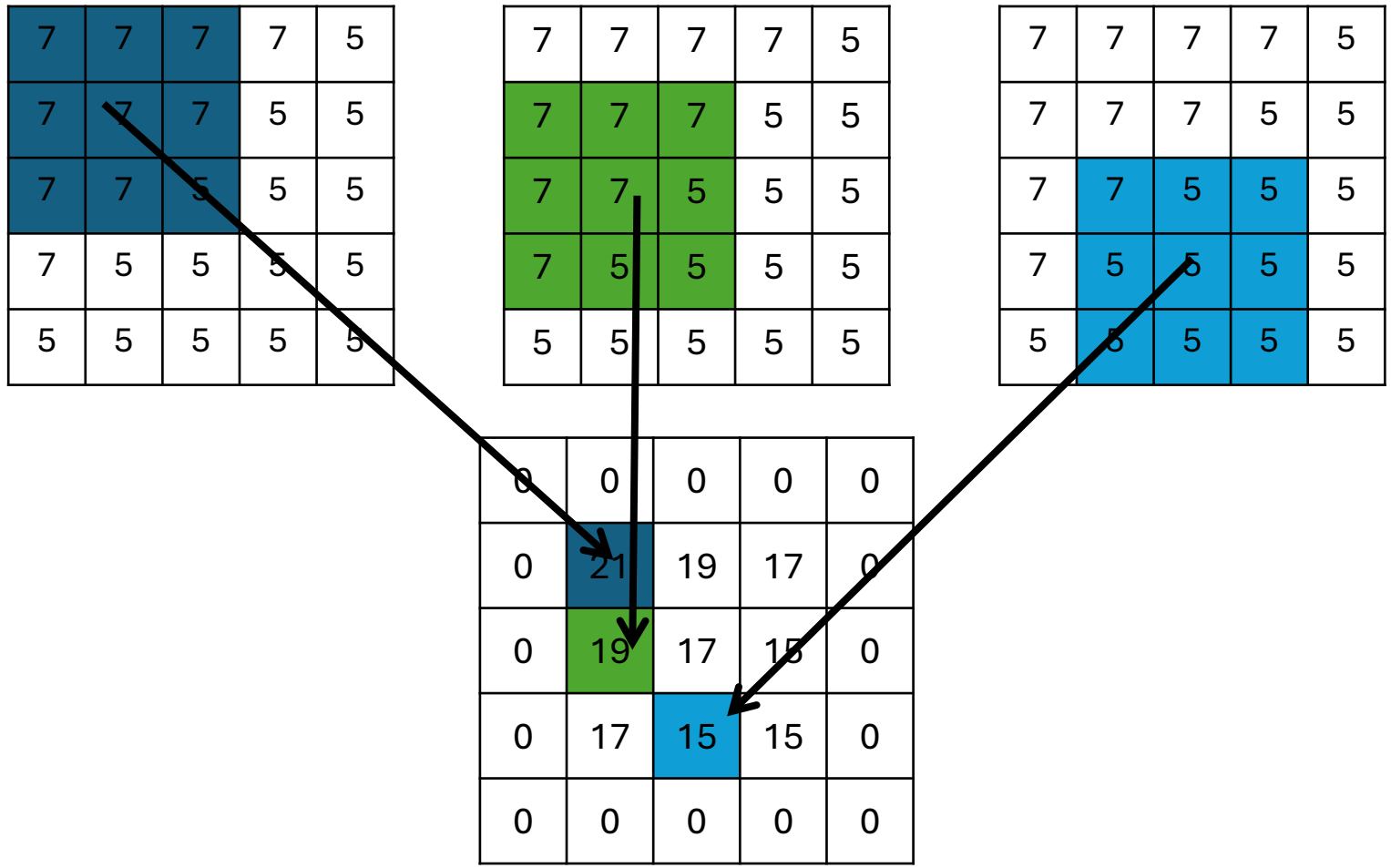
[19]
```
since = time.time()
```

0s completed at 2:50 PM

2:50 PM 3/27/2025

Convolution Computation

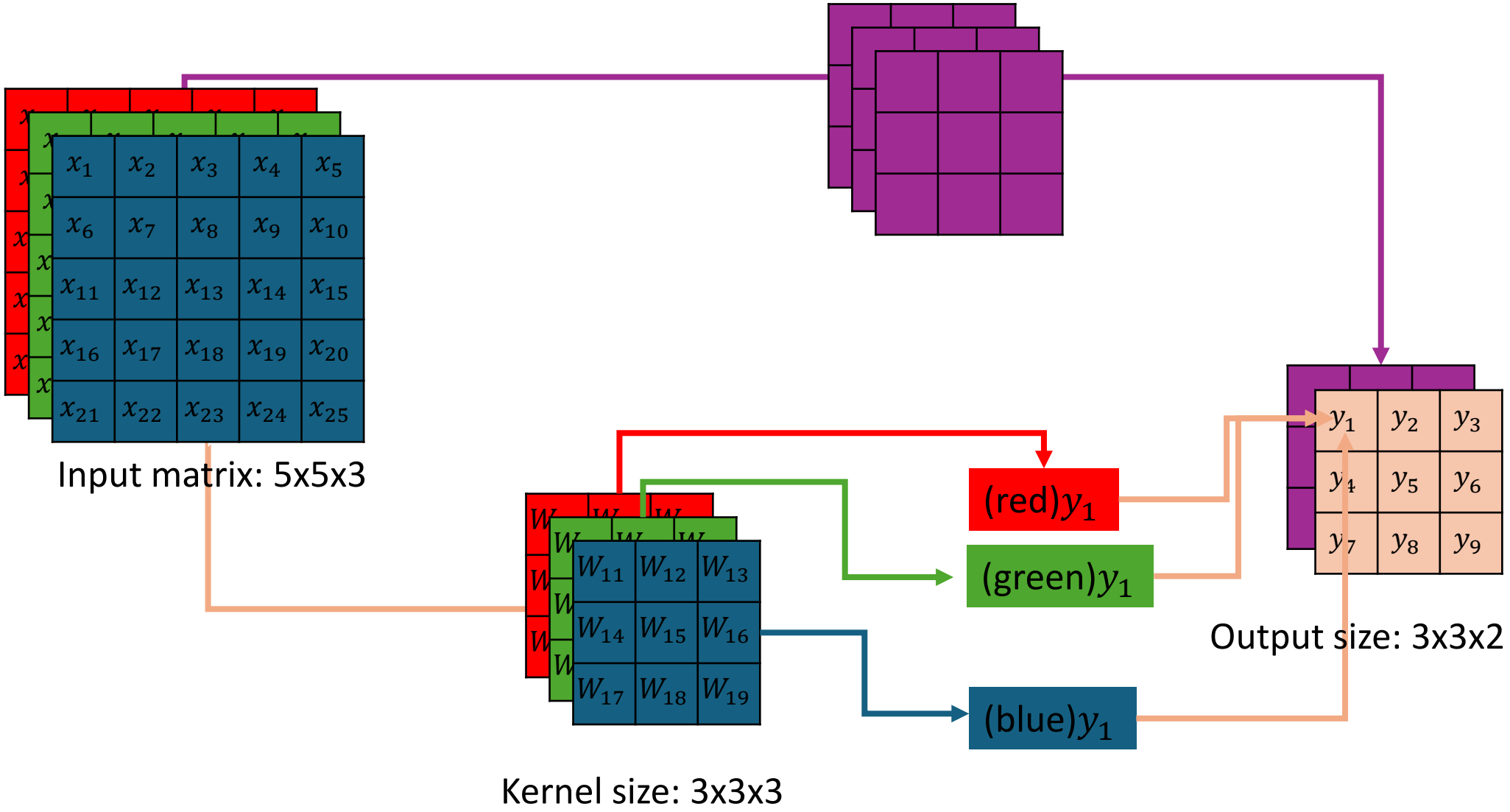
$$K = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$



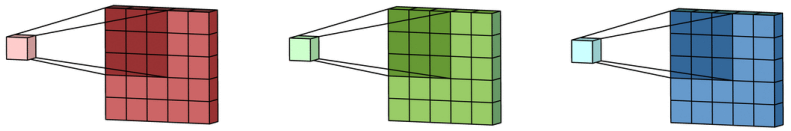
Kernel at position 1

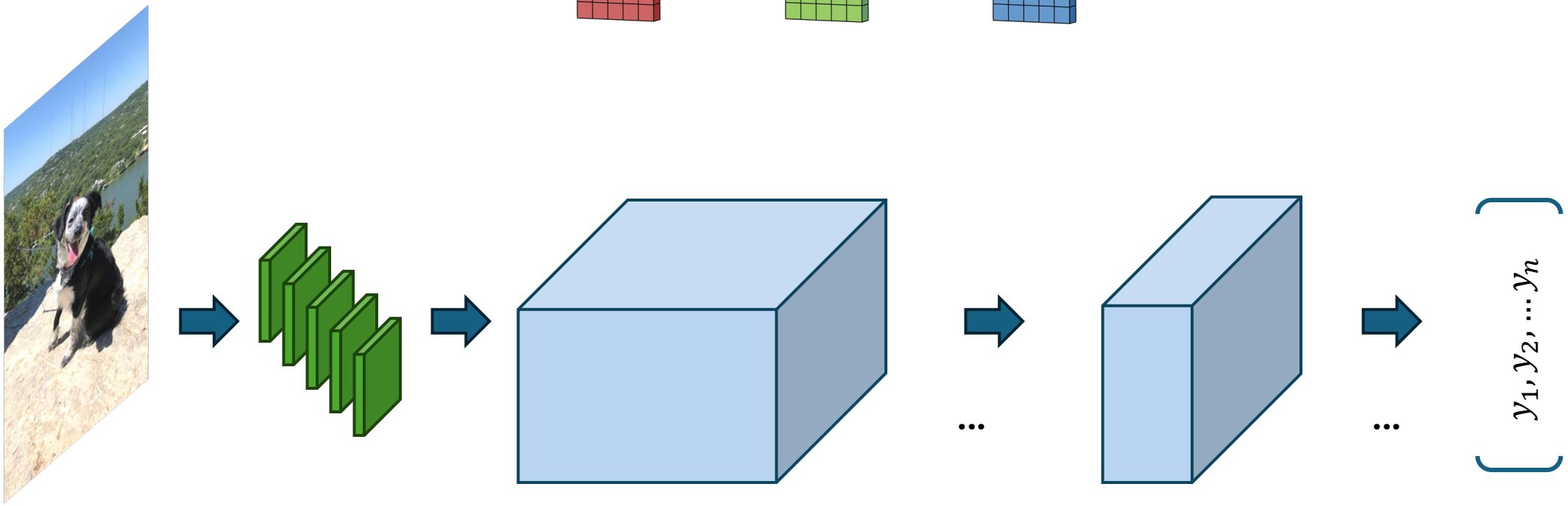
Kernel at position 2

Kernel at position n


x_1 x_2 x_3 x_4 x_5
x_6 x_7 x_8 x_9 x_{10}
x_{11} x_{12} x_{13} x_{14} x_{15}
x_{16} x_{17} x_{18} x_{19} x_{20}
x_{21} x_{22} x_{23} x_{24} x_{25}
Input matrix: 5x5x3
W_{11} W_{12} W_{13}
W_{14} W_{15} W_{16}
W_{17} W_{18} W_{19}
Kernel size: 3x3x3
(red)y_1
(green)y_1
(blue)y_1
y_1 y_2 y_3
y_4 y_5 y_6
y_7 y_8 y_9
Output size: 3x3x2

Convolutional Nueral Network





RGB Input (128x128x3)

50 filters (3x3x3)

Tensor (128x128x50)

k filters (3x3x50)

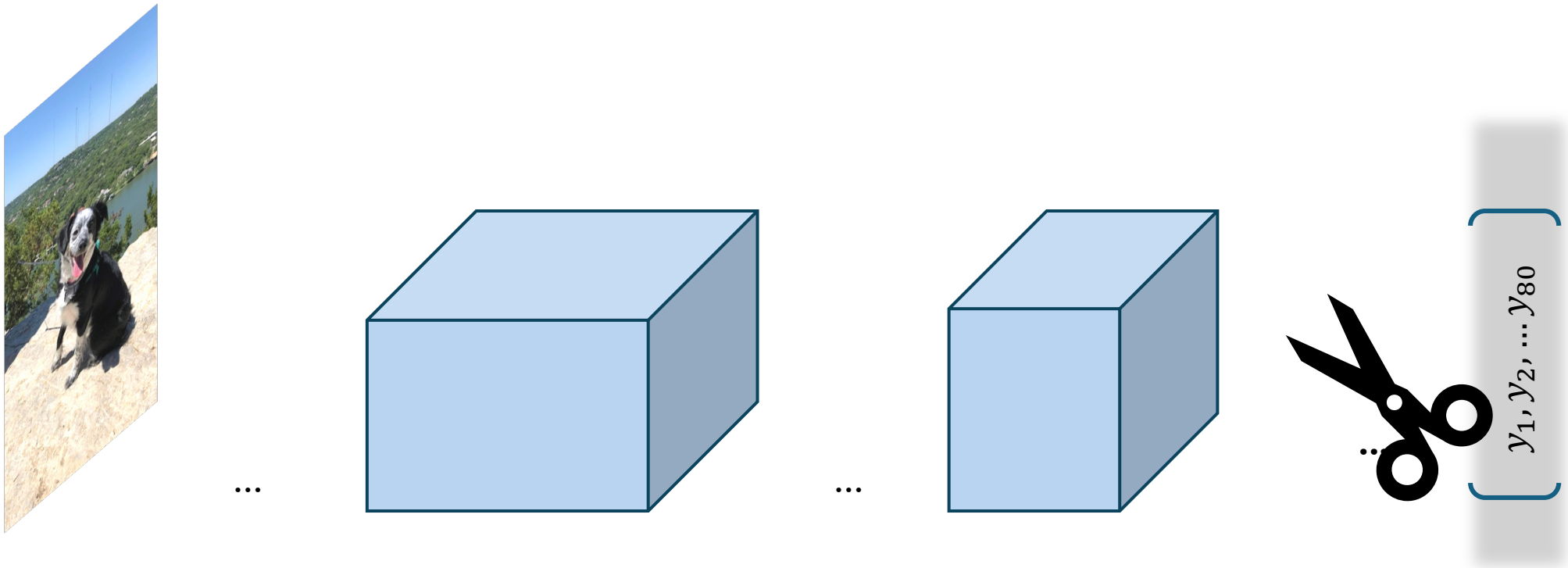
...

Output (1000 classes)

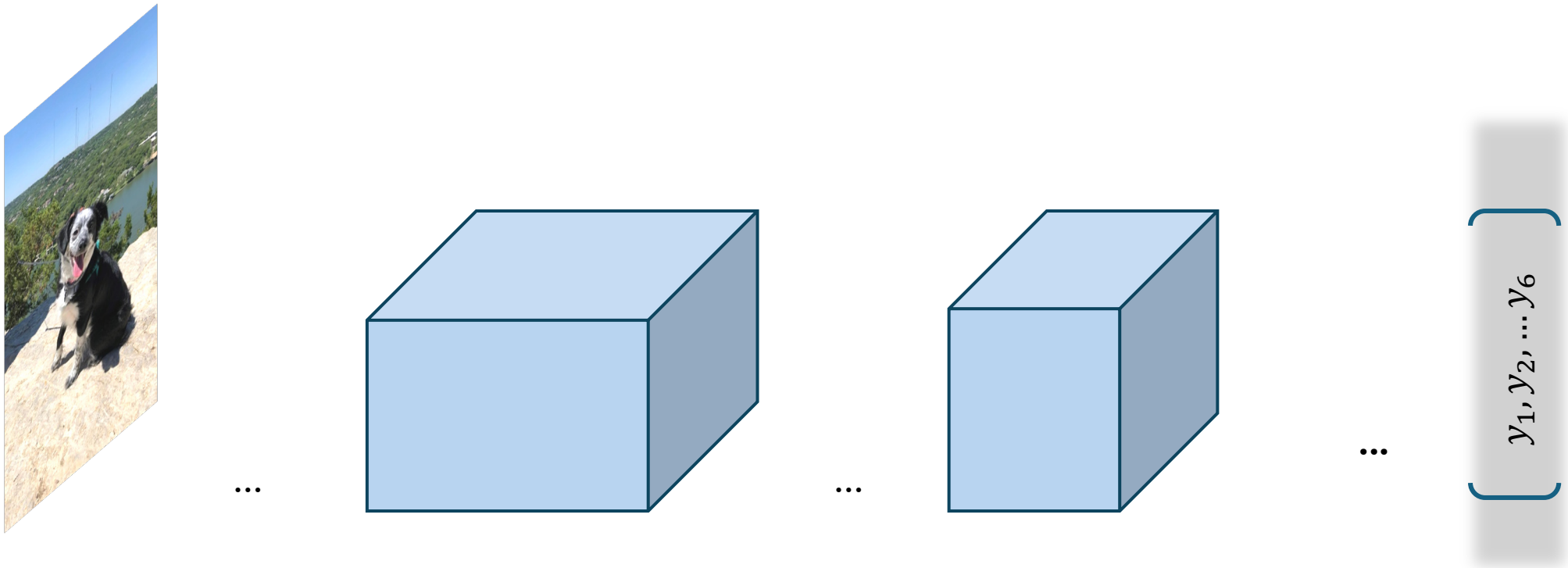
Large dataset

- COCO: https://cocodataset.org/#home (80 classes)
- ImageNet (1000 classes with 1.3M labeled images)
- Common Crawl (~100TB+)

Transfer learning

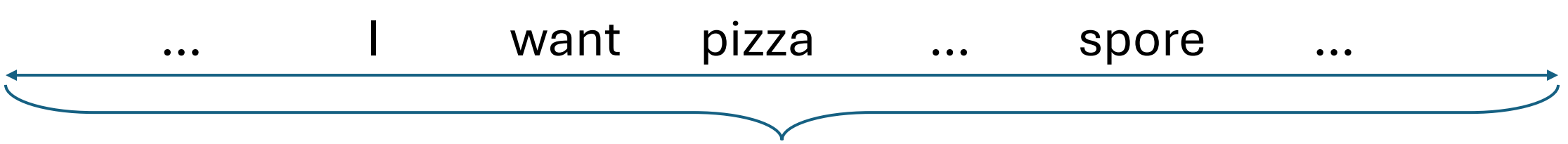


Transfer learning



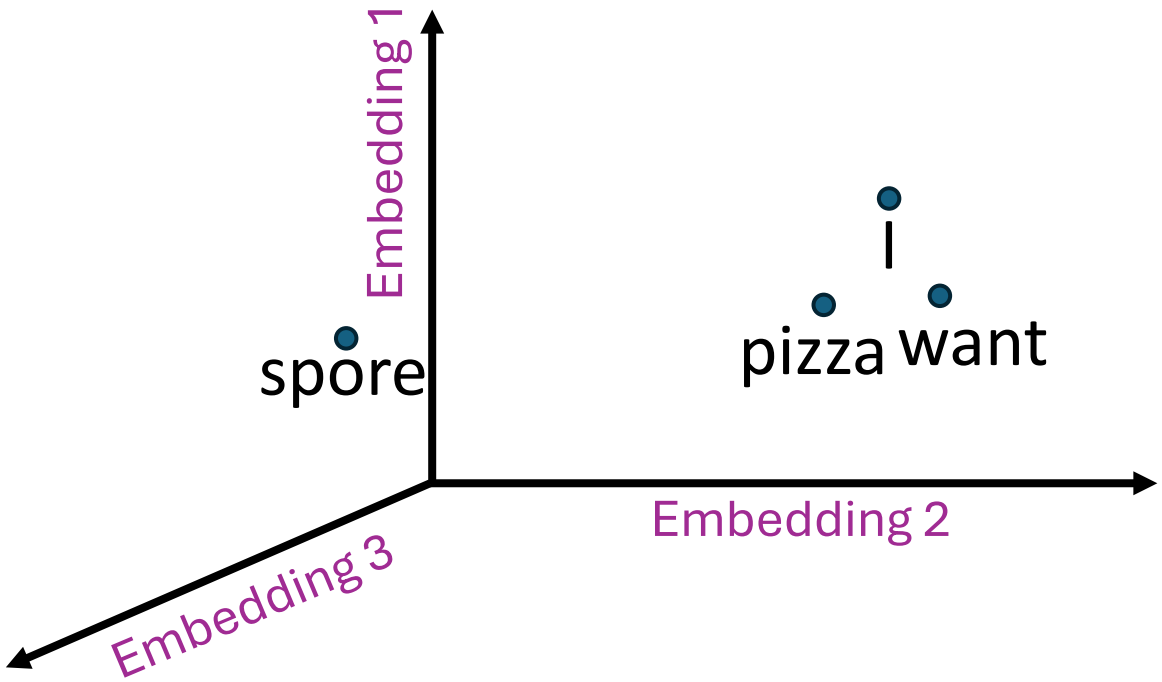
Transformer

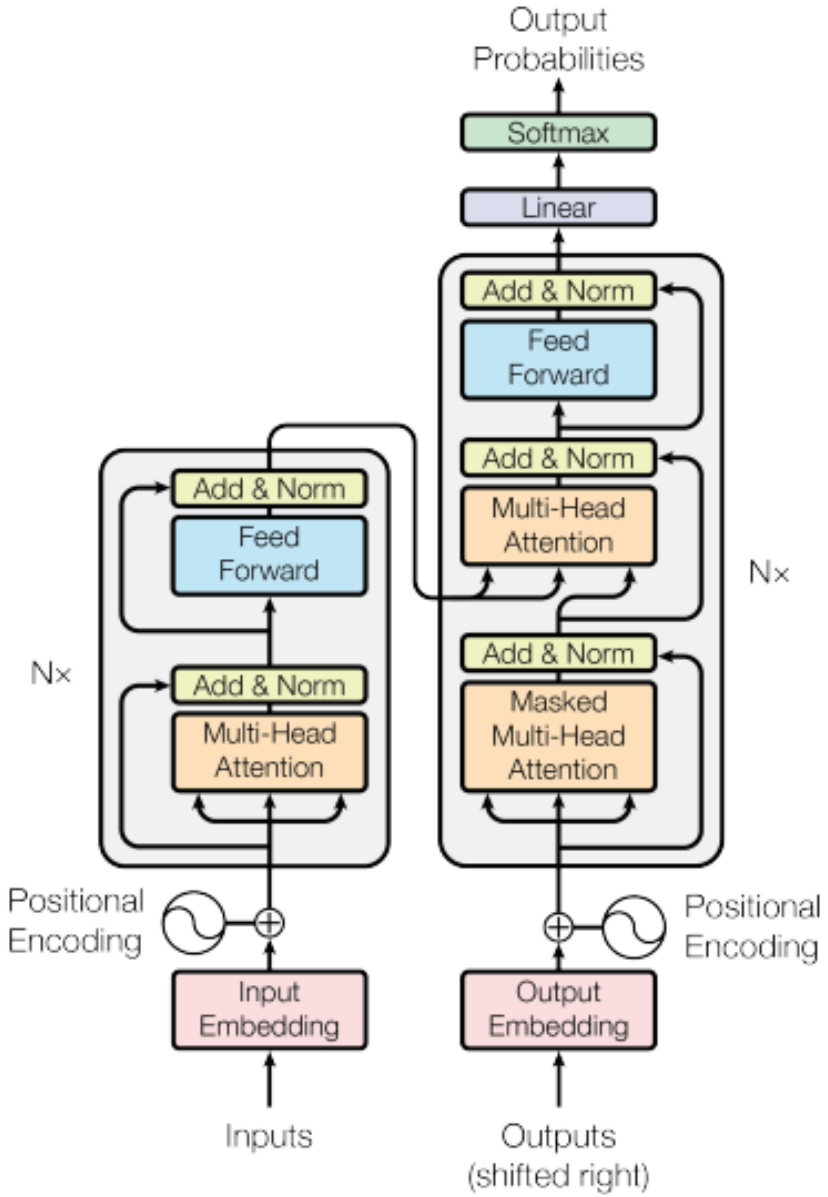
- GPT: Generative Pre-trained Transformer
- Tokenization: https://tiktokenizer.vercel.app/
- Vectorization: https://projector.tensorflow.org/



Total Text Corpus

Vocabulary	Embedding 1	Embedding 2	Embedding 3	...	Embedding n
...	...	...	...	...	...
I	e_{i1}	e_{i2}	e_{i3}	...	e_{in}
want	e_{w1}	e_{w2}	e_{w3}	...	e_{wn}
pizza	e_{p1}	e_{p2}	e_{p3}	...	e_{pn}
...	...	...	...	...	...
spore	e_{s1}	e_{s2}	e_{s3}	...	e_{sn}
...	...	...	...	...	...





Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

GPT 3

- Parameters 175 B
- Dataset 45T
- 96 attention heads
- 2048 token size
- Learn from their chief scientist: https://www.youtube.com/watch?v=kCc8FmEb1nY