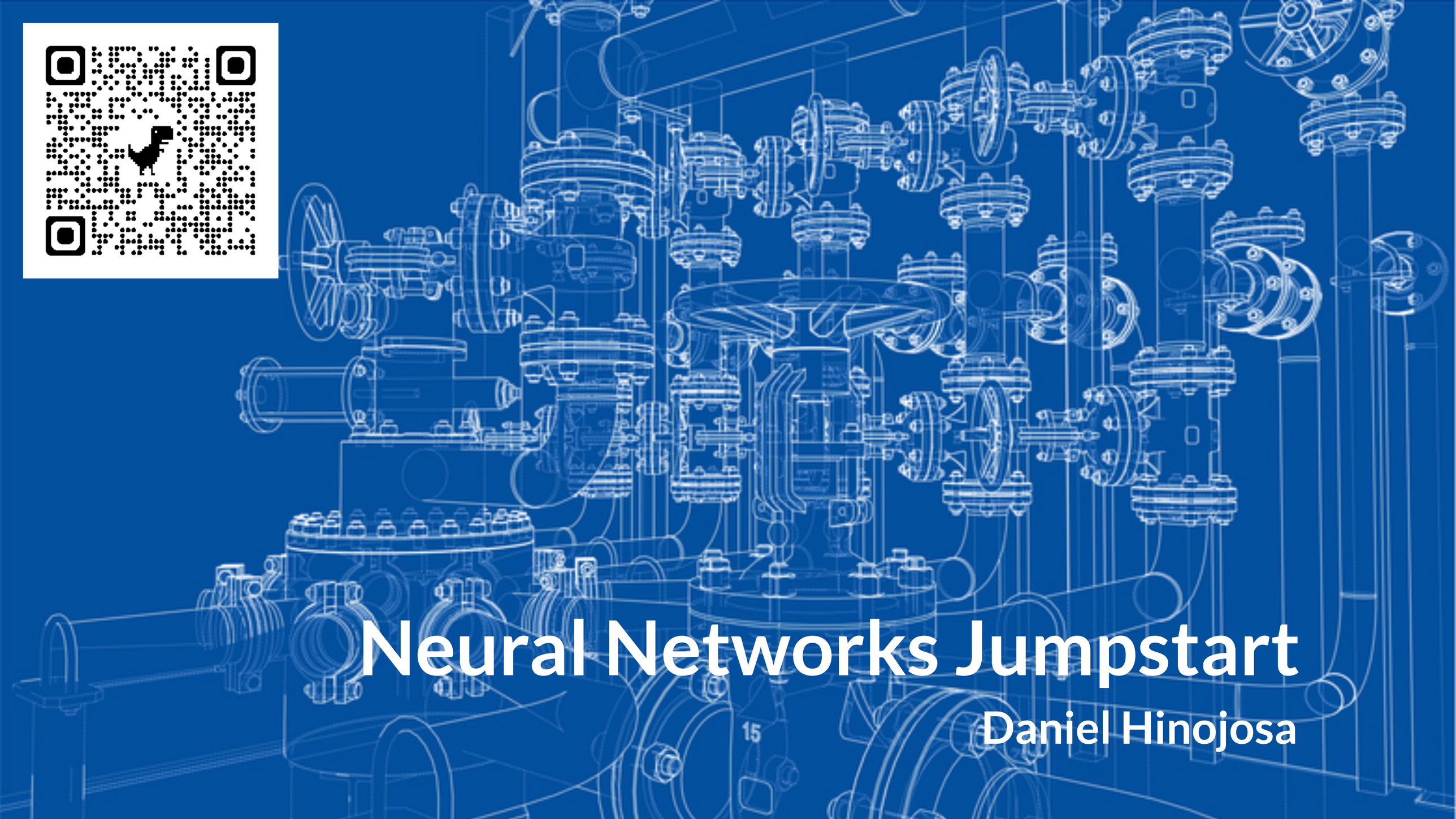
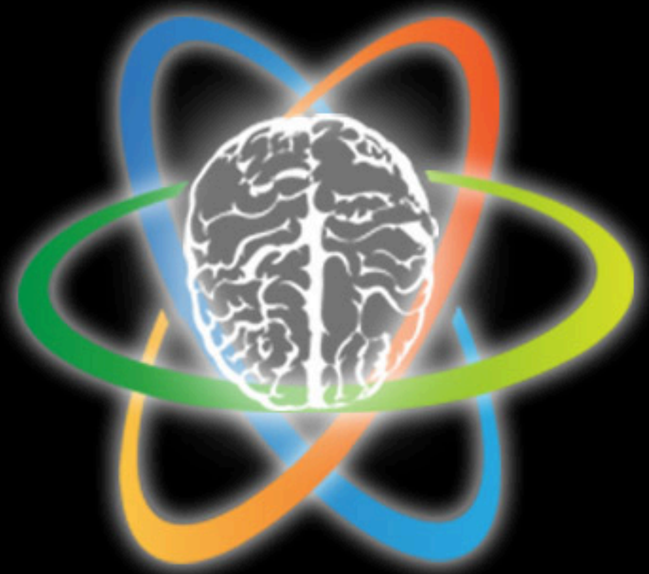


Neural Networks Jumpstart

Daniel Hinojosa


NFJS



ÜberConf

THE ULTIMATE CONFERENCE FOR SOFTWARE DEVELOPERS AND ARCHITECTS

July 15 - 18, 2025 — Denver

Save $250 - Register by March 31st

Gateway Software Symposium
April 4 - 5, 2025

Central Ohio Software Symposium
April 11 - 12, 2025

New England Software Symposium
May 2 - 3, 2025

CODE REMIX
Summit Miami | May 12 - 14, 2025

Great Lakes Software Symposium
May 16 - 17, 2025

ÜberConf
July 15 - 18, 2025

Central Iowa Software Symposium
September 5 - 6, 2025

Twin Cities Software Symposium
September 12 - 13, 2025

Northern Virginia Software Symposium
September 19 - 20, 2025

Pacific Northwest Software Symposium
October 10 - 11, 2025

TECHLEADER SUMMIT
December 3 - 5, 2025

ARCHCONF
December 8 - 11, 2025

Each NFJS conference registration now comes bundled with an NFJS Virtual Workshop Subscription!

Applications

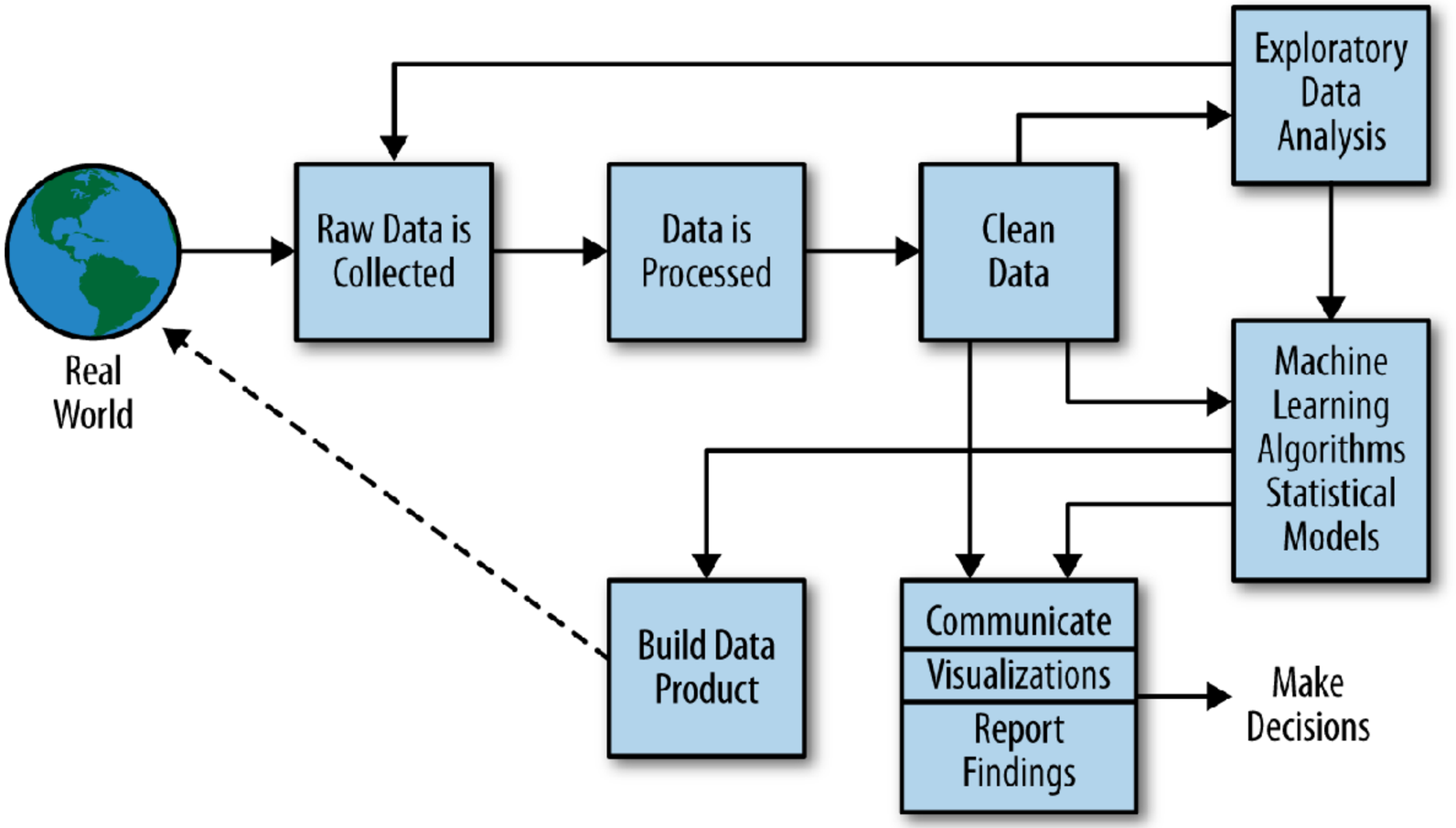
Fraud Detection

Financial Approvals

Picture Recognition

Your Idea Here

Process of Machine Learning



Doing Data Science By Cathy O'Neil, Rachel Schutt

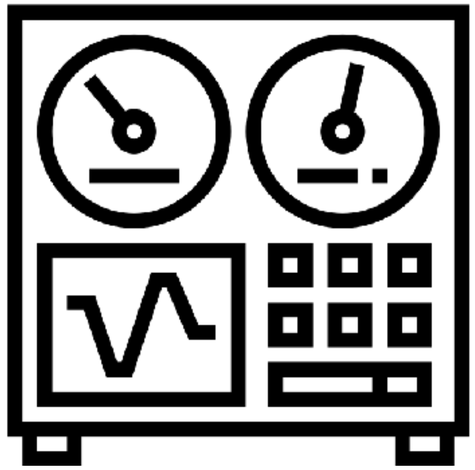
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

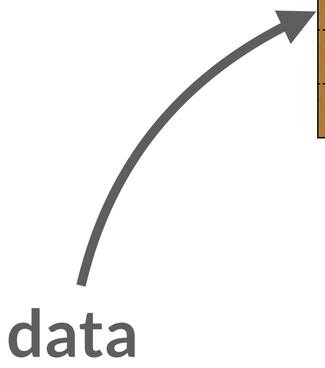
testing

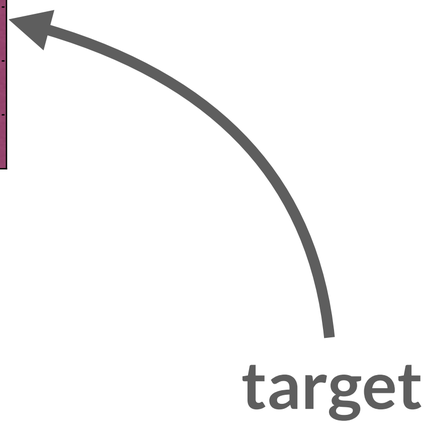
10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1

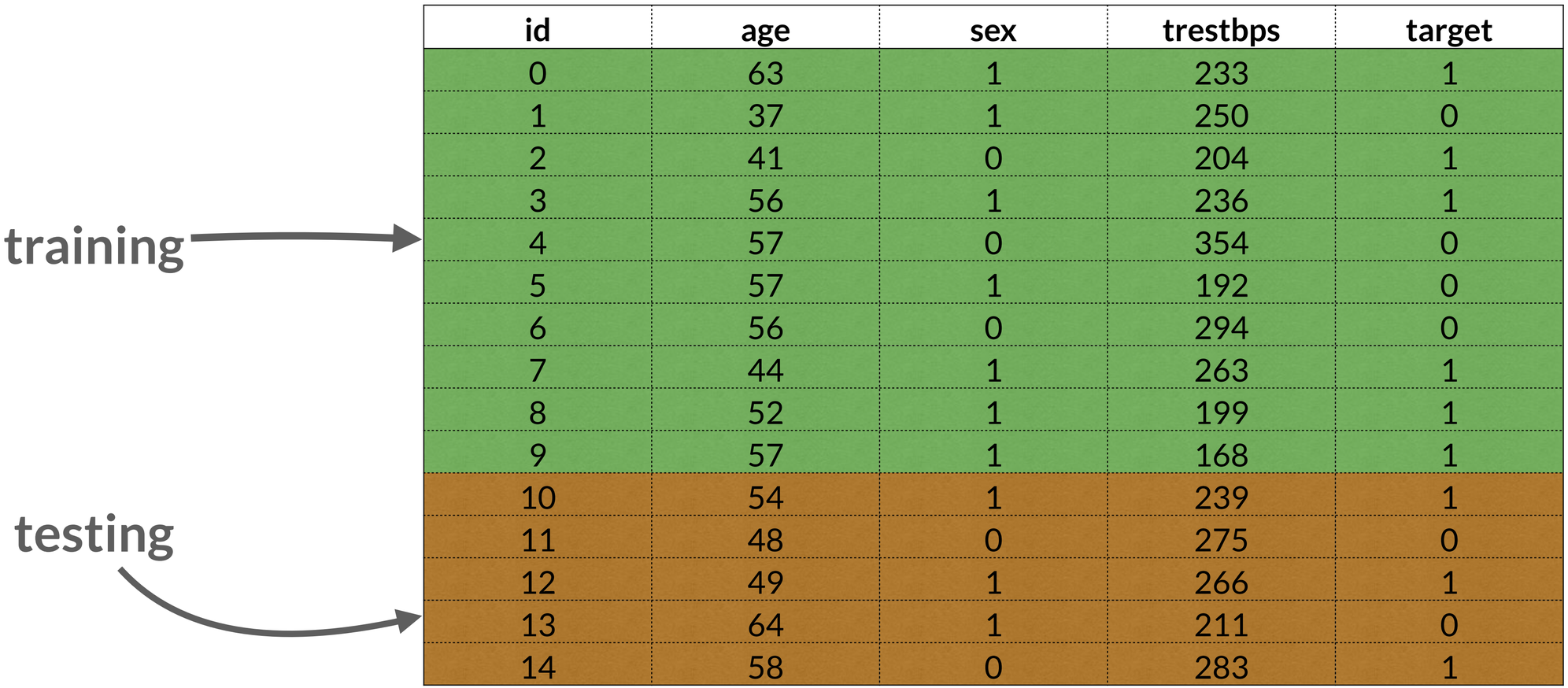
training phase



id	age	sex	trestbps	risk
0	20	1	145	1
1	45	1	130	1
2	43	0	130	0
3	33	1	150	0







id	age	sex	trestbps	target
0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1
10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1

id	age	sex	trestbps	target
0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1
10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1

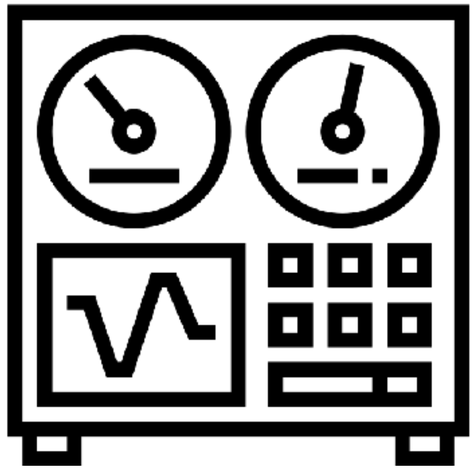
random distribution between training and testing

training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1



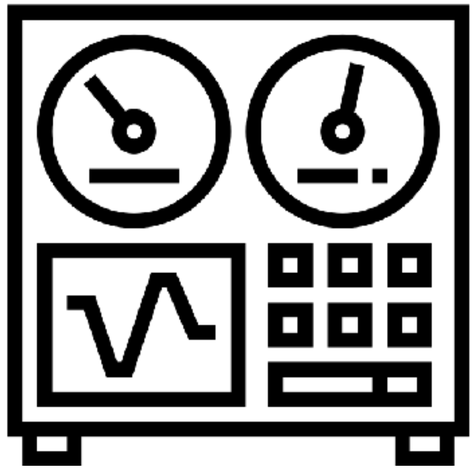
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1

model



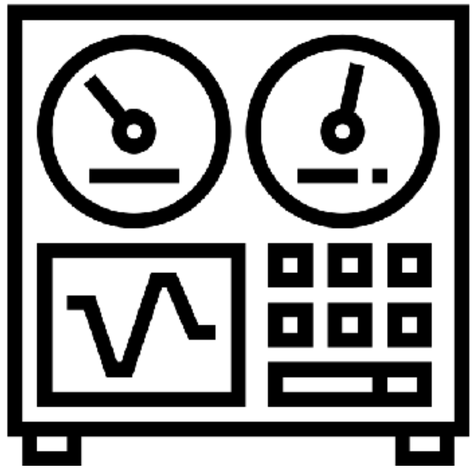
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1

training phase

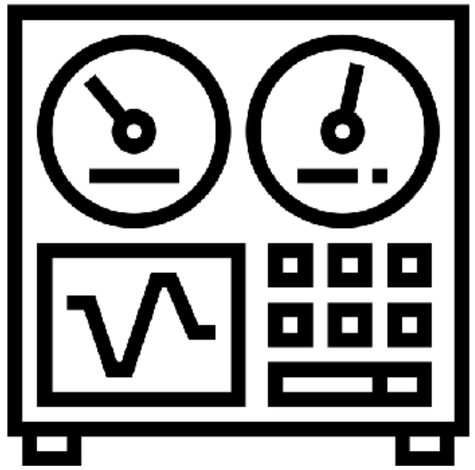


training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239	1
11	48	0	275	0
12	49	1	266	1
13	64	1	211	0
14	58	0	283	1



training

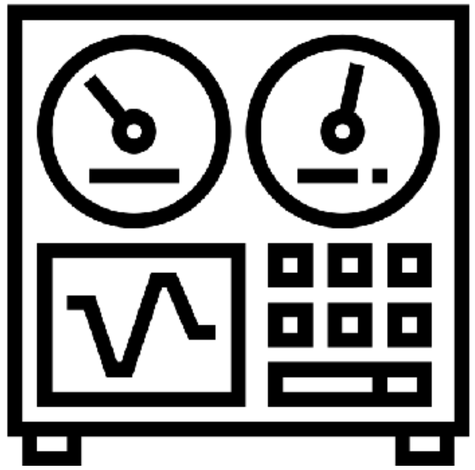
0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239
11	48	0	275
12	49	1	266
13	64	1	211
14	58	0	283

actual

1
0
1
0
1



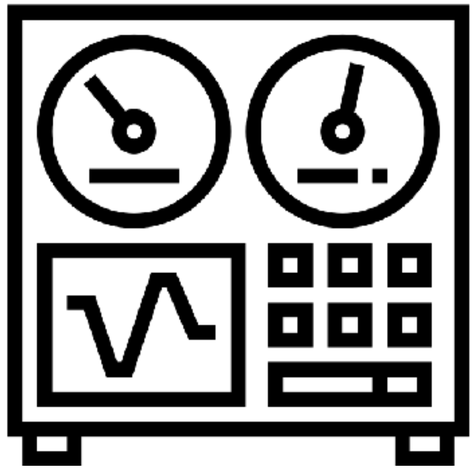
reserve the actual later for verification

training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239
11	48	0	275
12	49	1	266
13	64	1	211
14	58	0	283



testing phase

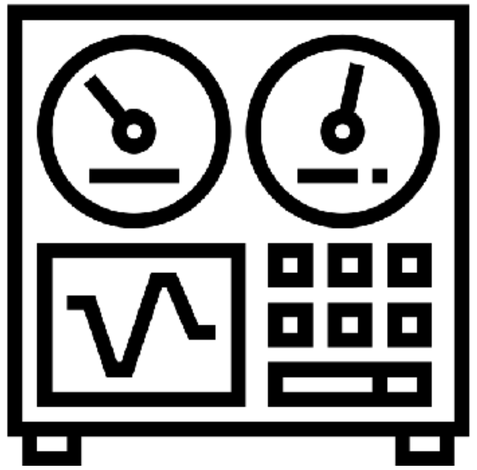
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239
11	48	0	275
12	49	1	266
13	64	1	211
14	58	0	283

!!!!!!!



generated result

1
0
1
1
1

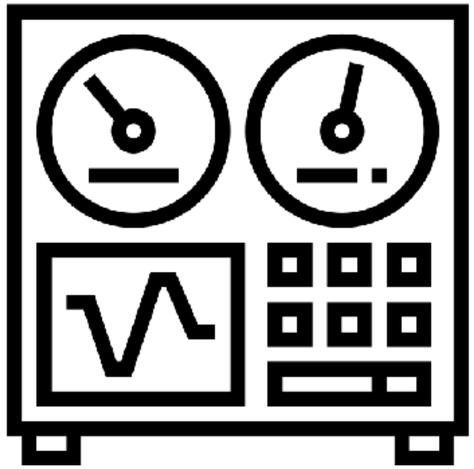
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239
11	48	0	275
12	49	1	266
13	64	1	211
14	58	0	283

!!!!!!!



generated result

1
0
1
1
1

actual

1
0
1
0
1

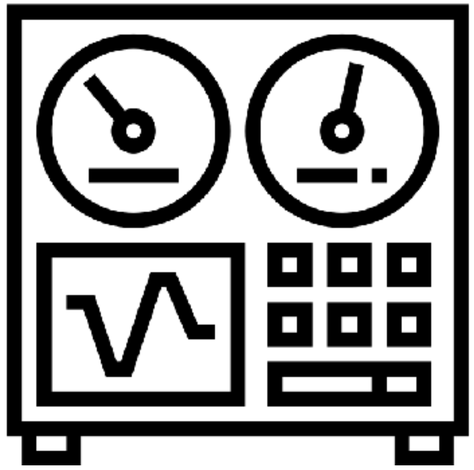
training

0	63	1	233	1
1	37	1	250	0
2	41	0	204	1
3	56	1	236	1
4	57	0	354	0
5	57	1	192	0
6	56	0	294	0
7	44	1	263	1
8	52	1	199	1
9	57	1	168	1

testing

10	54	1	239
11	48	0	275
12	49	1	266
13	64	1	211
14	58	0	283

!!!!!!!



generated result

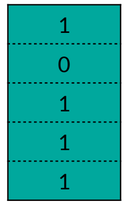
1
0
1
1
1

actual

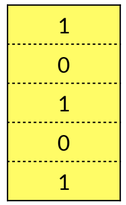
1
0
1
0
1

How did we do?

generated result



actual

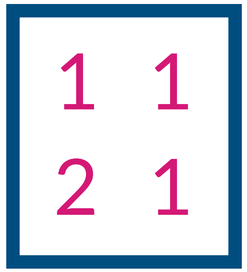


How did we do?

Tensor

Scalar	Vector	Matrix
Rank-0	Rank-1	Rank-2

1

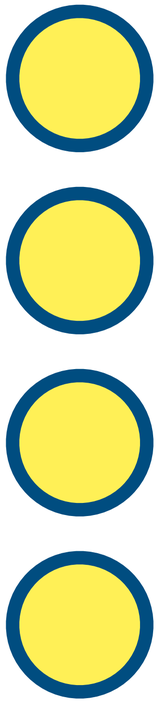


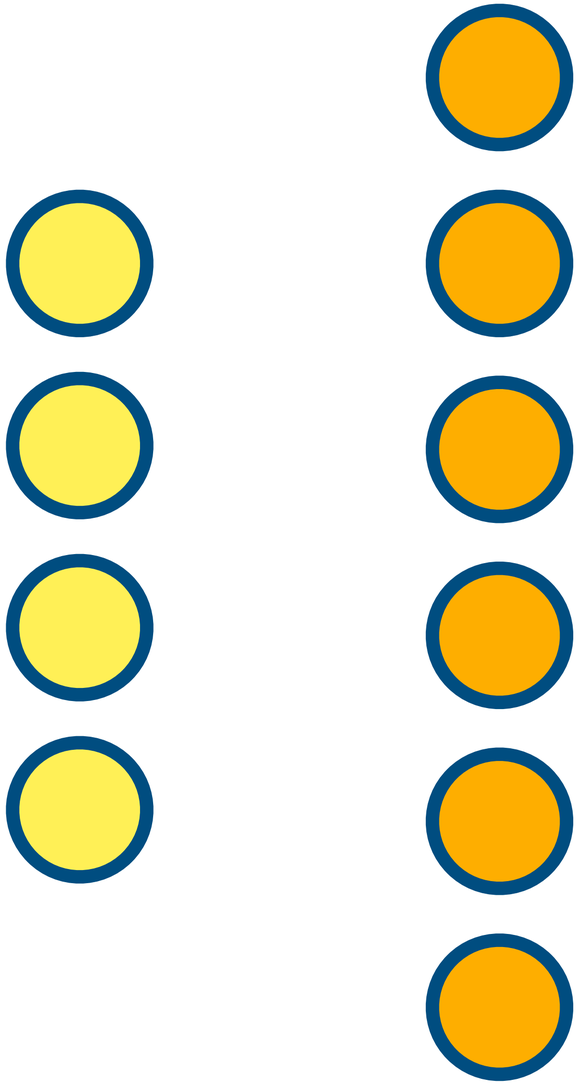
A tensor is a mathematical entity that lives in a structure and interacts with other mathematical entities. If one *transforms* the other entities in the structure in a regular way, then the tensor *must obey a related transformation rule.*

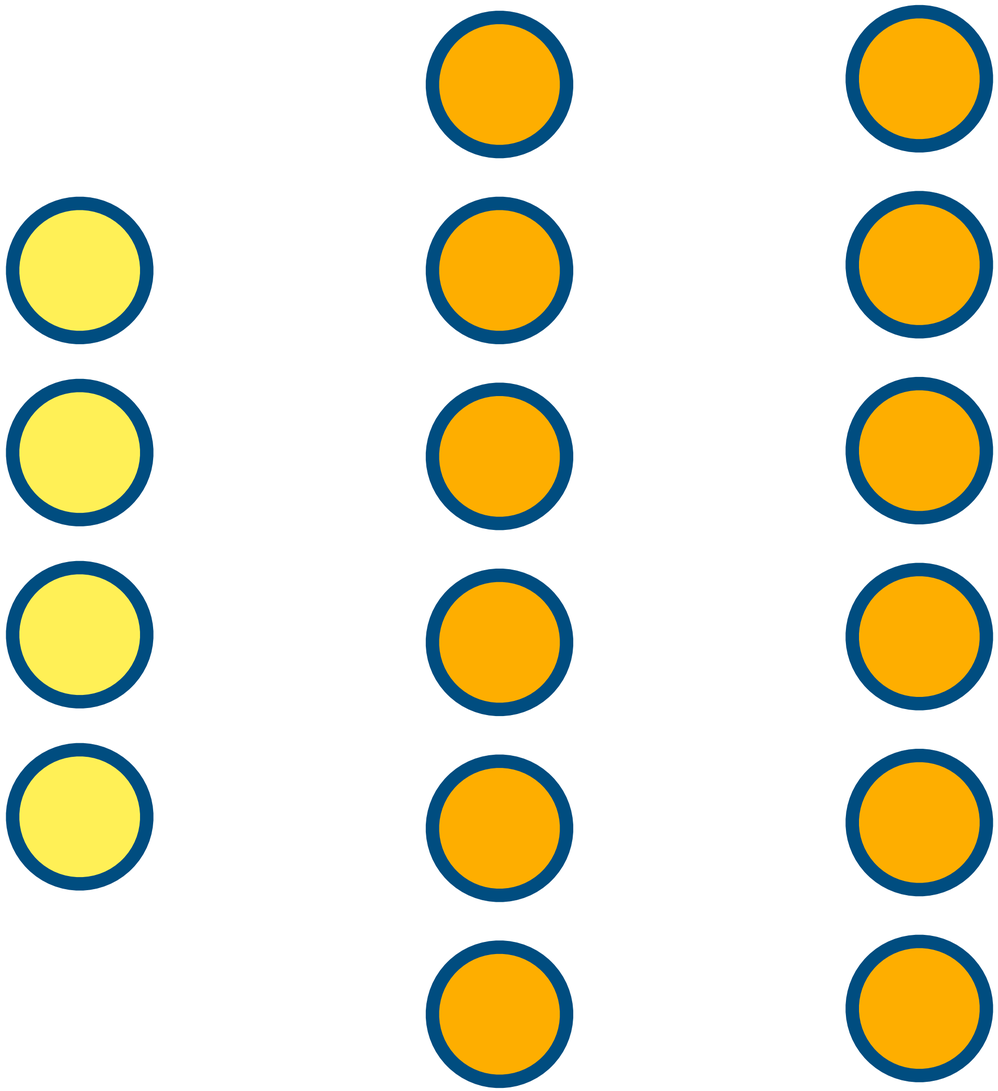
8 9	7 8 9	6 8 9	**1**	**5**	**2**	**3**	4 6 7	6 7
3	2 5 7 8 9	1 2 6 8 9	4 6 7 9	6 7 8 9	4 6 8 9	1 2 4 5 7	4 5 6 7	2 5 6 7
1 2 5	2 5 7	**4**	3 6 7	3 6 7	3 6	1 2 5 7	**8**	**9**
2 4 8	2 3 4 8	**5**	2 3 4 6	2 3 6	**7**	2 4 8	**9**	**1**
6	2 4 9	**7**	**8**	1 2 9	1 4 5 9	2 4 5	4 5	**3**
2 4 8 9	**1**	2 8 9	2 3 4 5 6 9	2 3 6 9	3 4 5 6 9	2 4 5 7 8	4 5 6 7	2 5 6 7 8
2 5 8	**6**	2 8	2 3 5 7	**4**	3 5 8	**9**	**1**	5 7 8
1 5 8 9	5 8 9	**3**	5 7 9	1 7 8 9	1 5 8 9	**6**	**2**	**4**
7	2 4 5 8 9	1 2 8 9	2 3 5 6 9	1 2 3 6 8 9	1 3 5 6 8 9	5 8	3 5	5 8

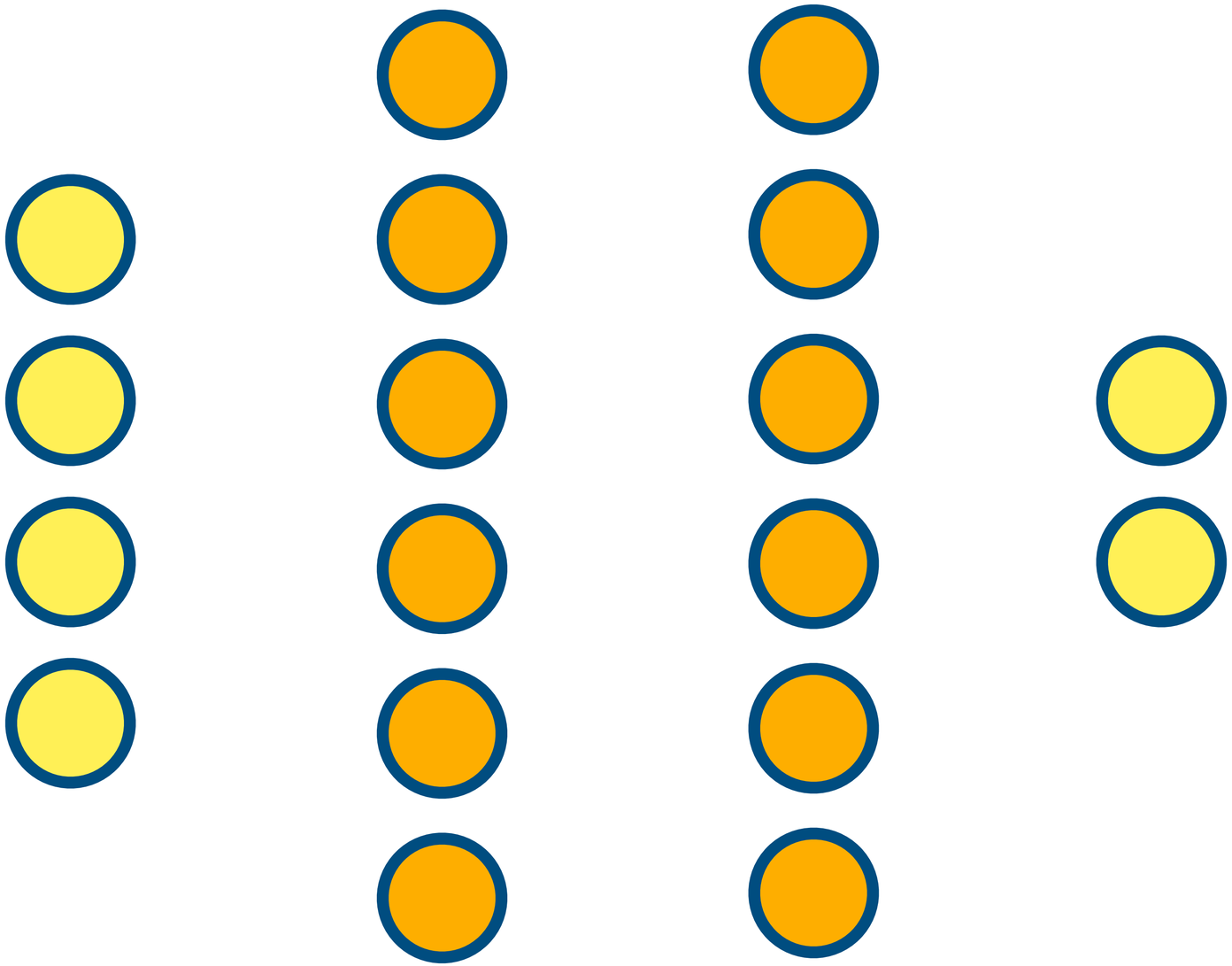
A 2-D Tensor?

Neural Networks









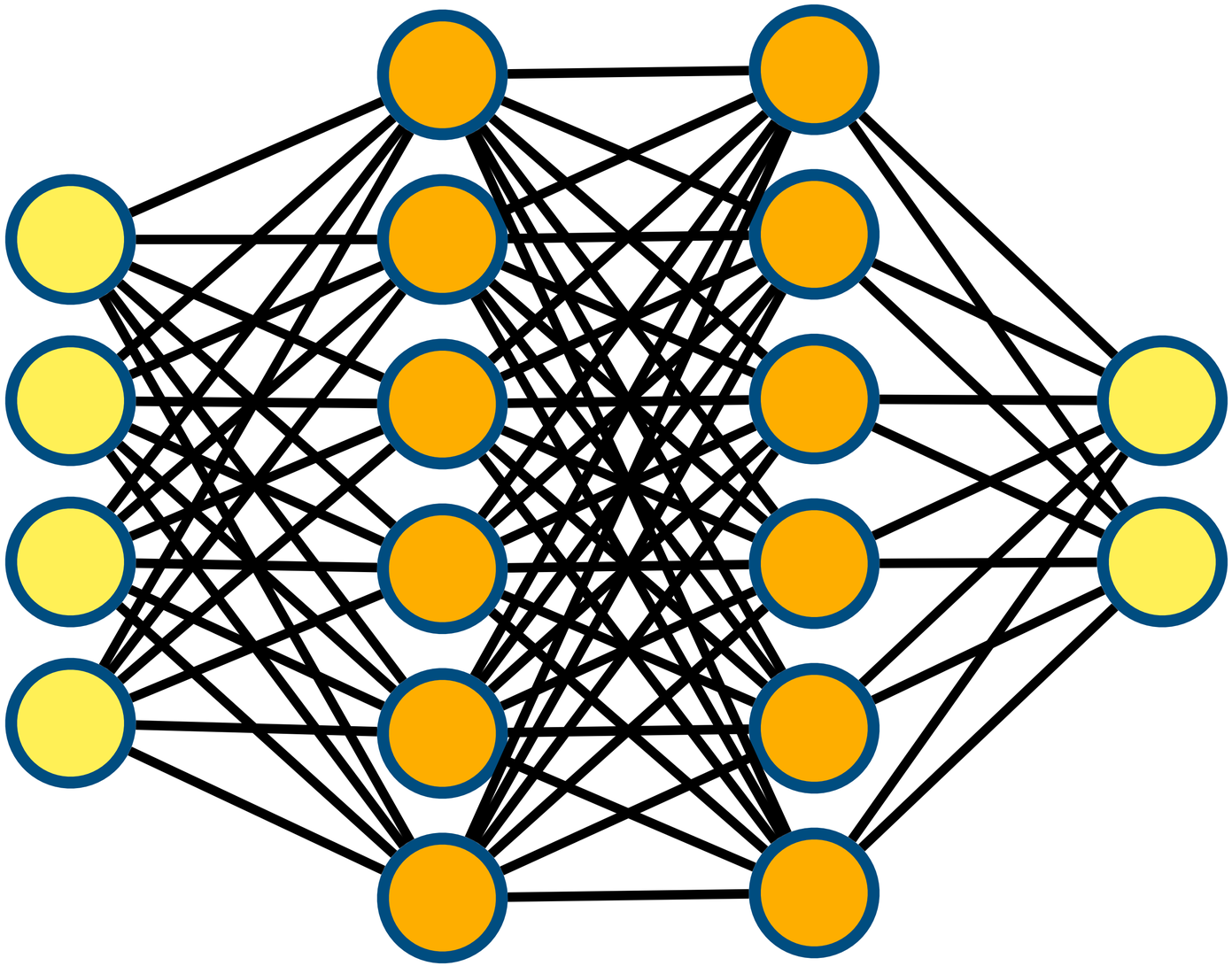


Table of Data

Temperature (°C)	Humidity (%)	Dew Point (°C)	Cloud Cover (%)	Rain (Label: Y)
25	90	20	90	1 (Rain)
30	50	15	40	0 (No Rain)
22	85	18	95	1 (Rain)
28	60	17	60	0 (No Rain)
26	78	19	85	1 (Rain)
31	55	16	35	0 (No Rain)
24	90	21	98	1 (Rain)
29	65	18	70	0 (No Rain)
23	88	20	92	1 (Rain)
27	58	17	50	0 (No Rain)
30	80	20	90	1 (Rain)
22	50	15	40	0 (No Rain)
28	85	18	95	1 (Rain)
26	60	17	60	0 (No Rain)
31	78	19	85	1 (Rain)
24	55	16	35	0 (No Rain)
29	90	21	98	1 (Rain)
23	65	18	70	0 (No Rain)
27	88	20	92	1 (Rain)
30	58	17	50	0 (No Rain)
22	80	20	90	1 (Rain)
28	50	15	40	0 (No Rain)
26	85	18	95	1 (Rain)
31	60	17	60	0 (No Rain)
24	78	19	85	1 (Rain)
29	55	16	35	0 (No Rain)
23	90	21	98	1 (Rain)
27	65	18	70	0 (No Rain)
30	88	20	92	1 (Rain)
22	58	17	50	0 (No Rain)
28	80	20	90	1 (Rain)
26	50	15	40	0 (No Rain)
31	85	18	95	1 (Rain)
24	60	17	60	0 (No Rain)
29	78	19	85	1 (Rain)
23	55	16	35	0 (No Rain)
27	90	21	98	1 (Rain)
30	65	18	70	0 (No Rain)
22	88	20	92	1 (Rain)
28	58	17	50	0 (No Rain)
26	80	20	90	1 (Rain)
31	50	15	40	0 (No Rain)
24	85	18	95	1 (Rain)
29	60	17	60	0 (No Rain)
23	78	19	85	1 (Rain)
27	55	16	35	0 (No Rain)
30	90	21	98	1 (Rain)
22	65	18	70	0 (No Rain)
28	88	20	92	1 (Rain)
26	58	17	50	0 (No Rain)
31	80	20	90	1 (Rain)
24	50	15	40	0 (No Rain)
29	85	18	95	1 (Rain)
23	60	17	60	0 (No Rain)
27	78	19	85	1 (Rain)
23	55	16	35	0 (No Rain)
23	90	21	98	1 (Rain)
23	65	18	70	0 (No Rain)
23	88	20	92	1 (Rain)

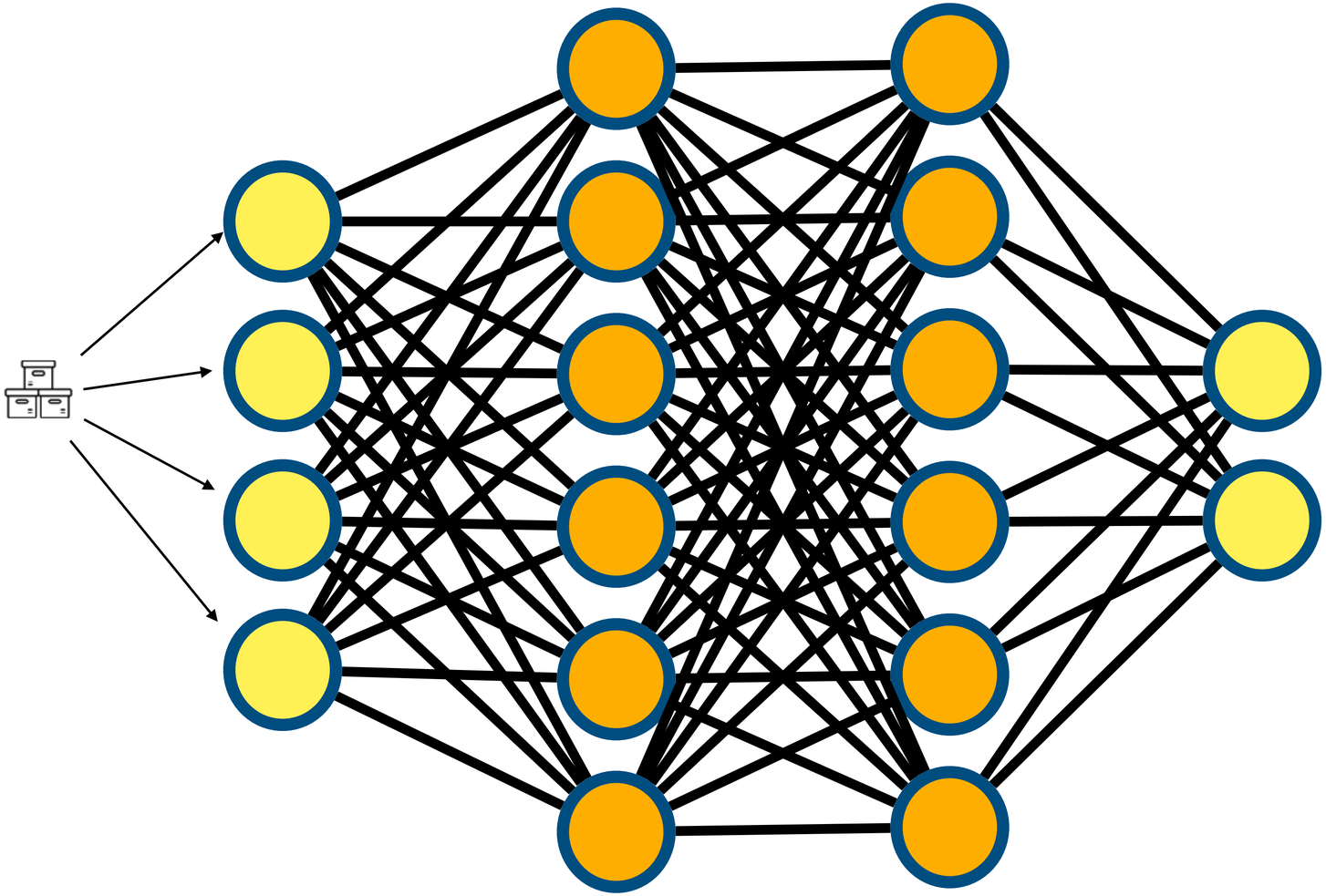
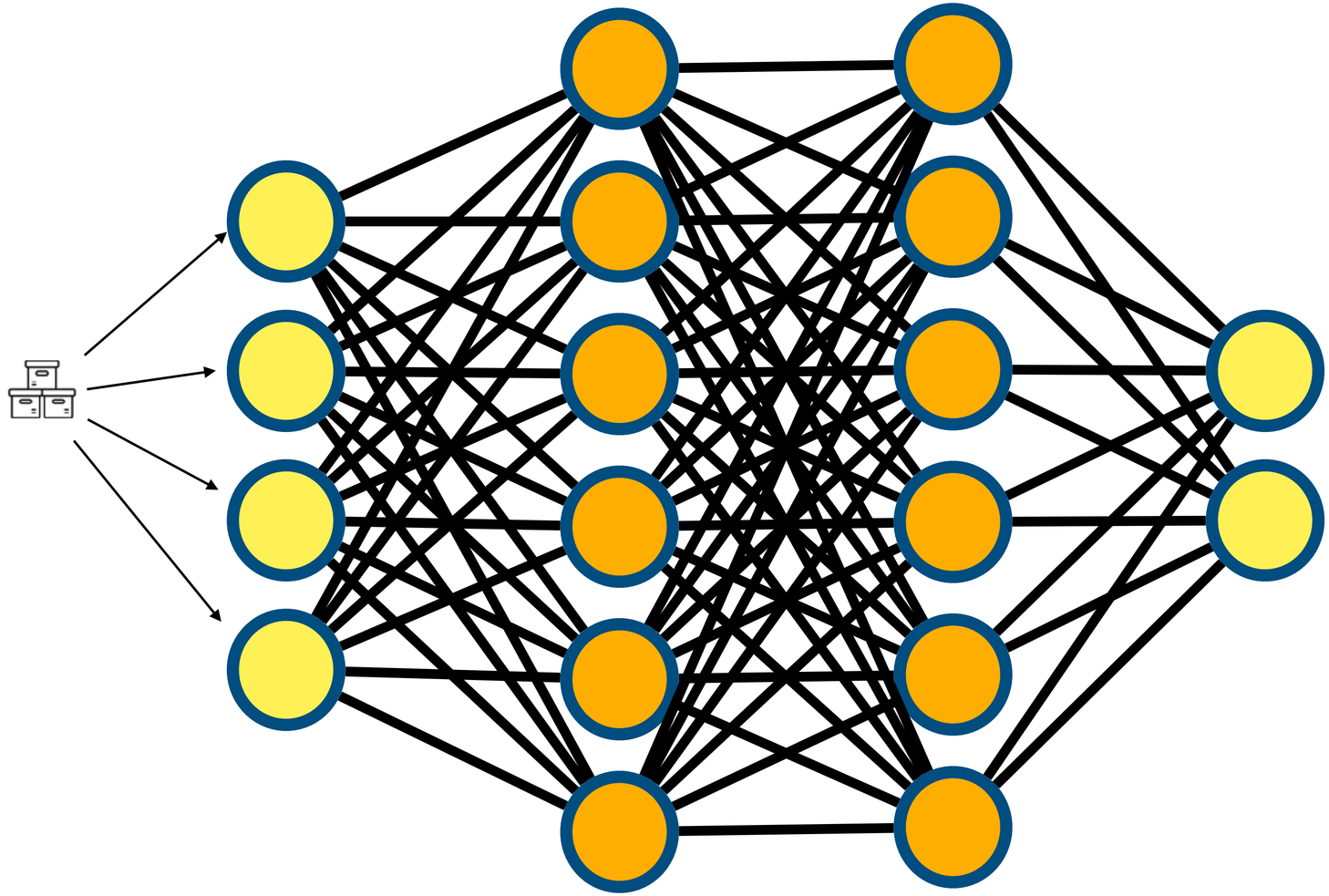


Table of Data

Temperature (°C)	Humidity (%)	Dew Point (°C)	Cloud Cover (%)	Rain (Label: Y)
25	90	20	90	1 (Rain)
30	50	15	40	0 (No Rain)
22	85	18	95	1 (Rain)
28	60	17	60	0 (No Rain)
26	78	19	85	1 (Rain)
31	55	16	35	0 (No Rain)
24	90	21	98	1 (Rain)
29	65	18	70	0 (No Rain)
23	88	20	92	1 (Rain)
27	58	17	50	0 (No Rain)
30	80	20	90	1 (Rain)
22	50	15	40	0 (No Rain)
28	85	18	95	1 (Rain)
26	60	17	60	0 (No Rain)
31	78	19	85	1 (Rain)
24	55	16	35	0 (No Rain)
29	90	21	98	1 (Rain)
23	65	18	70	0 (No Rain)
27	88	20	92	1 (Rain)
26	58	17	50	0 (No Rain)
22	80	20	90	1 (Rain)
28	50	15	40	0 (No Rain)
26	85	18	95	1 (Rain)
31	60	17	60	0 (No Rain)
24	78	19	85	1 (Rain)
29	55	16	35	0 (No Rain)
23	90	21	98	1 (Rain)
27	65	18	70	0 (No Rain)
30	88	20	92	1 (Rain)
22	58	17	50	0 (No Rain)
28	80	20	90	1 (Rain)
26	50	15	40	0 (No Rain)
31	85	18	95	1 (Rain)
24	60	17	60	0 (No Rain)
29	78	19	85	1 (Rain)
23	55	16	35	0 (No Rain)
27	90	21	98	1 (Rain)
30	65	18	70	0 (No Rain)
22	88	20	92	1 (Rain)
28	58	17	50	0 (No Rain)
26	80	20	90	1 (Rain)
31	50	15	40	0 (No Rain)
24	85	18	95	1 (Rain)
29	60	17	60	0 (No Rain)
23	78	19	85	1 (Rain)
27	55	16	35	0 (No Rain)
30	90	21	98	1 (Rain)
22	65	18	70	0 (No Rain)
28	88	20	92	1 (Rain)
26	58	17	50	0 (No Rain)
31	80	20	90	1 (Rain)
24	50	15	40	0 (No Rain)
29	85	18	95	1 (Rain)
23	60	17	60	0 (No Rain)
27	78	19	85	1 (Rain)
23	55	16	35	0 (No Rain)
23	90	21	98	1 (Rain)
23	65	18	70	0 (No Rain)
23	88	20	92	1 (Rain)



🌦️ Example Batch of Data (10 Observations)

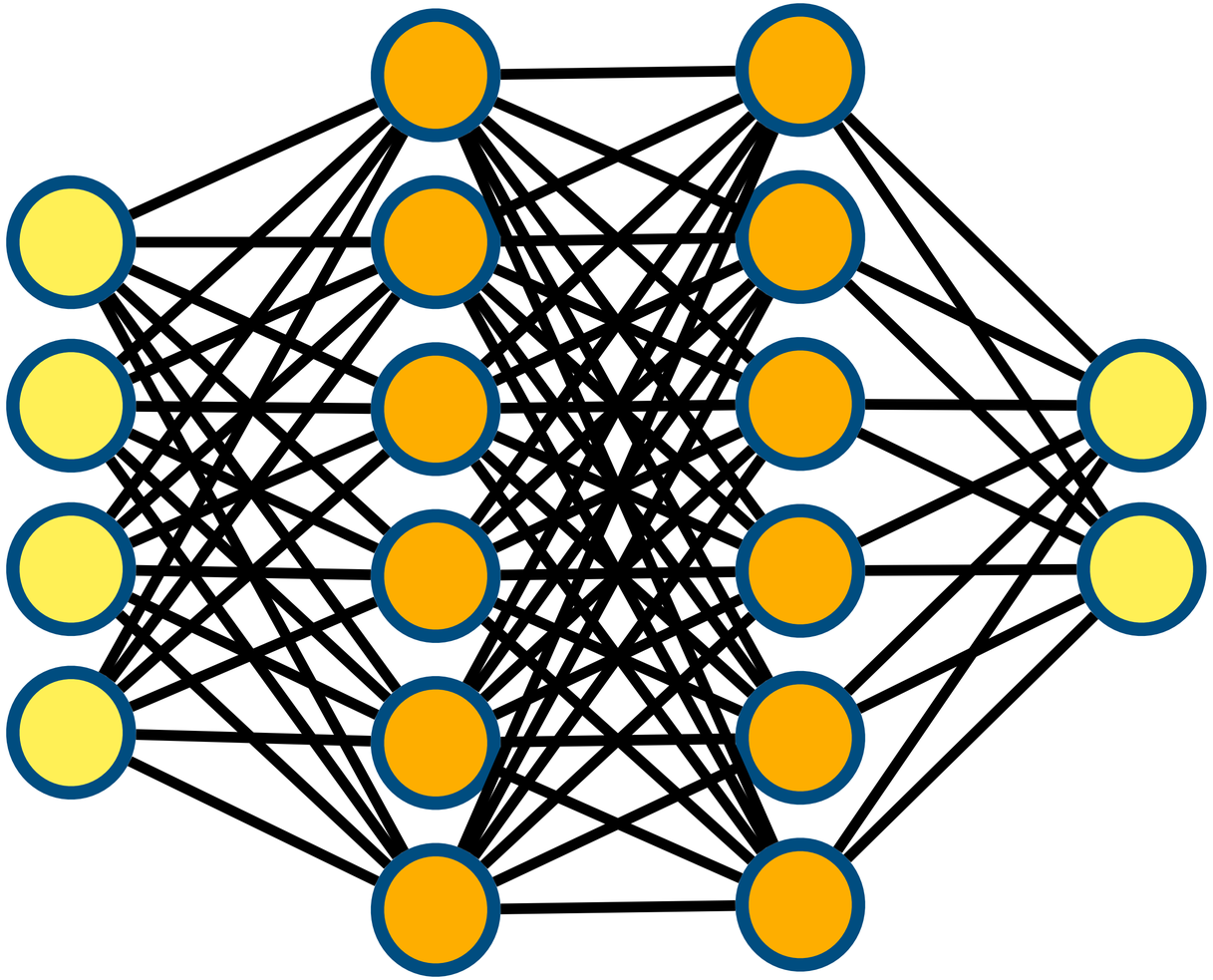
Temperature (°C)	Humidity (%)	Dew Point	Cloud Cover	Rain (Label: Y)
25	80	20	90	1 (Rain)
30	50	15	40	0 (No Rain)
22	85	18	95	1 (Rain)
28	60	17	60	0 (No Rain)
26	78	19	85	1 (Rain)
31	55	16	35	0 (No Rain)
24	90	21	98	1 (Rain)
29	65	18	70	0 (No Rain)
23	88	20	92	1 (Rain)
27	58	17	50	0 (No Rain)

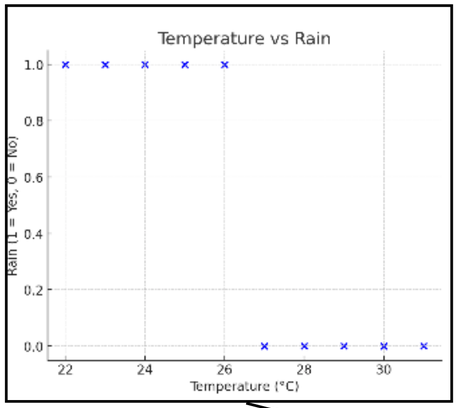
Temp: [25, 30, 22, 28, 26, 31, 24, 29, 23, 27]

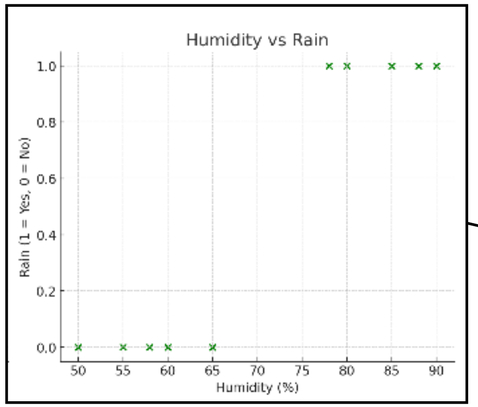
Humidity [80, 50, 85, 60, 78, 55, 90, 65, 88, 58]

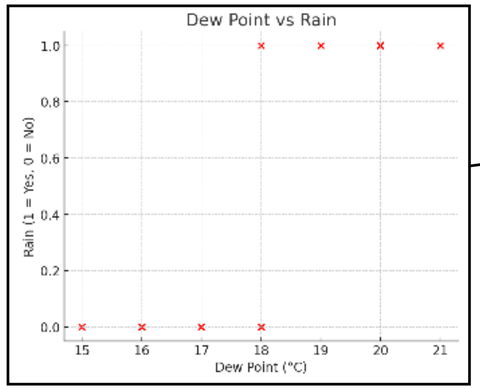
Dew Point [20, 15, 18, 17, 19, 16, 21, 18, 20, 17]

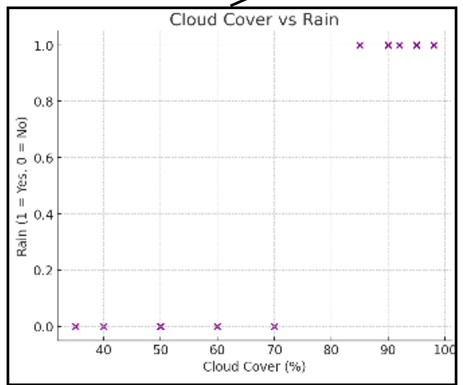
Cloud Cover [90, 40, 95, 60, 85, 35, 98, 70, 92, 50]

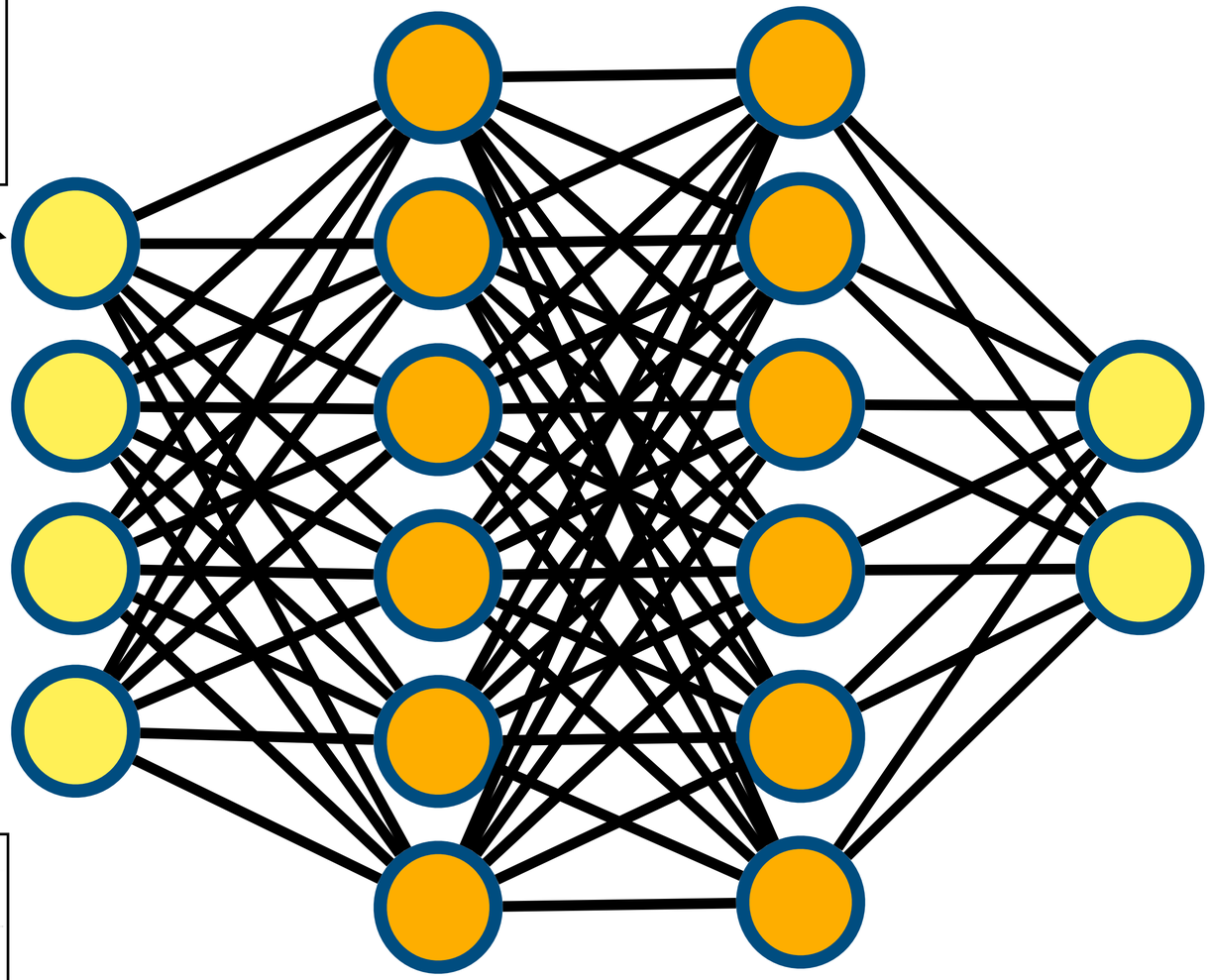


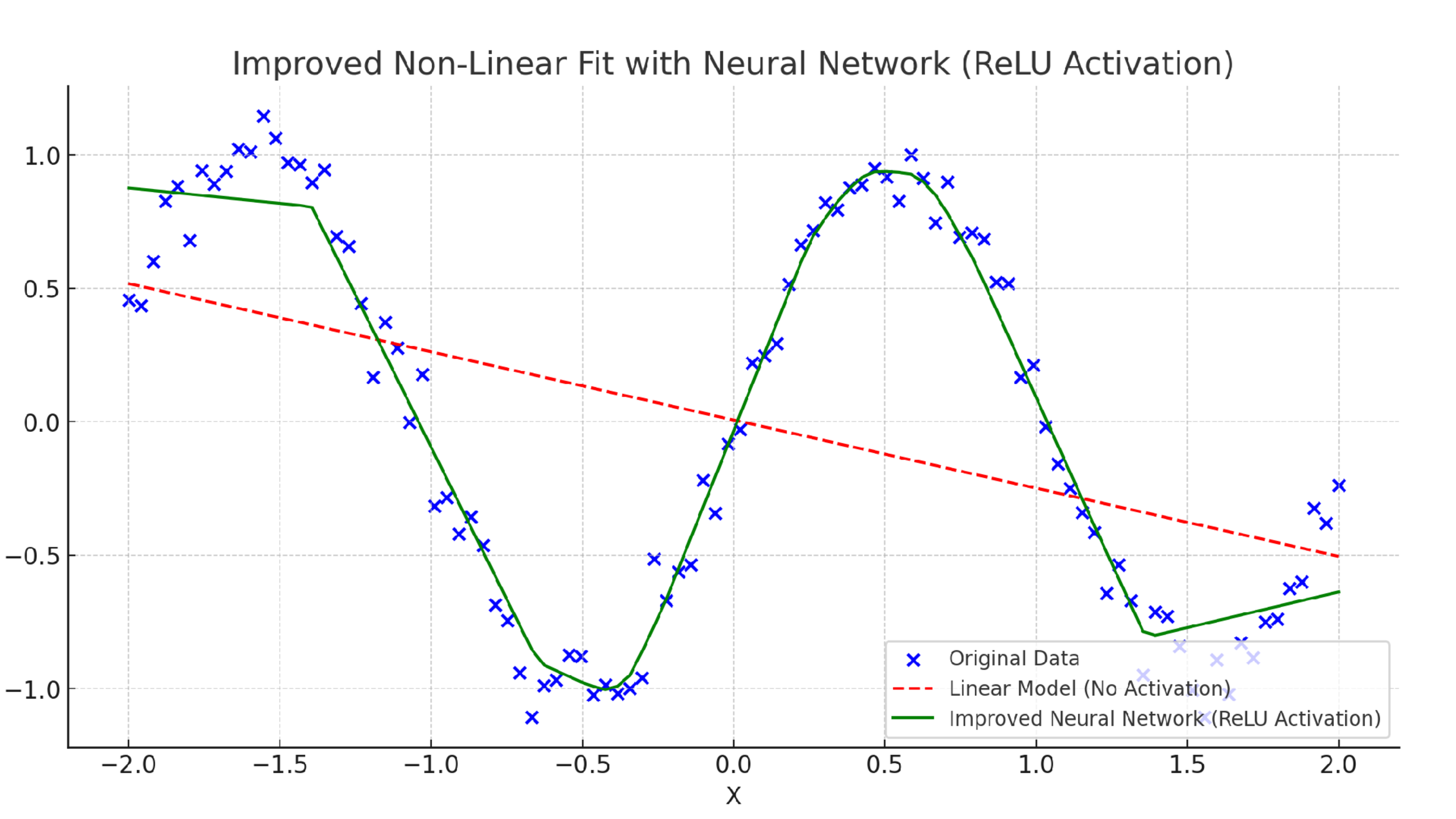

Temperature vs Rain
Rain (1 = Yes, 0 = No)
Temperature (°C)


Humidity vs Rain
Rain (1 = Yes, 0 = No)
Humidity (%)

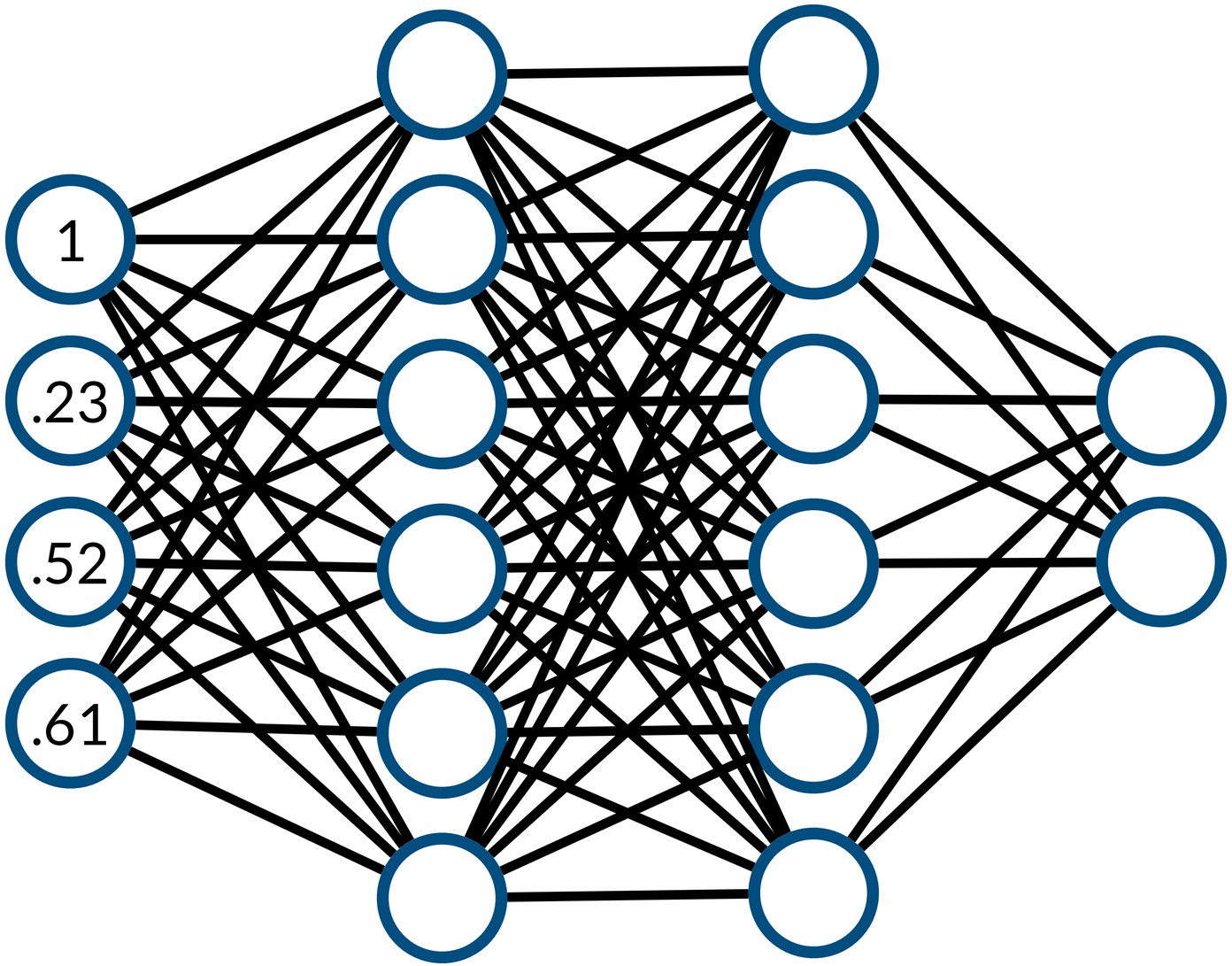

Dew Point vs Rain
Rain (1 = Yes, 0 = No)
Dew Point (°C)

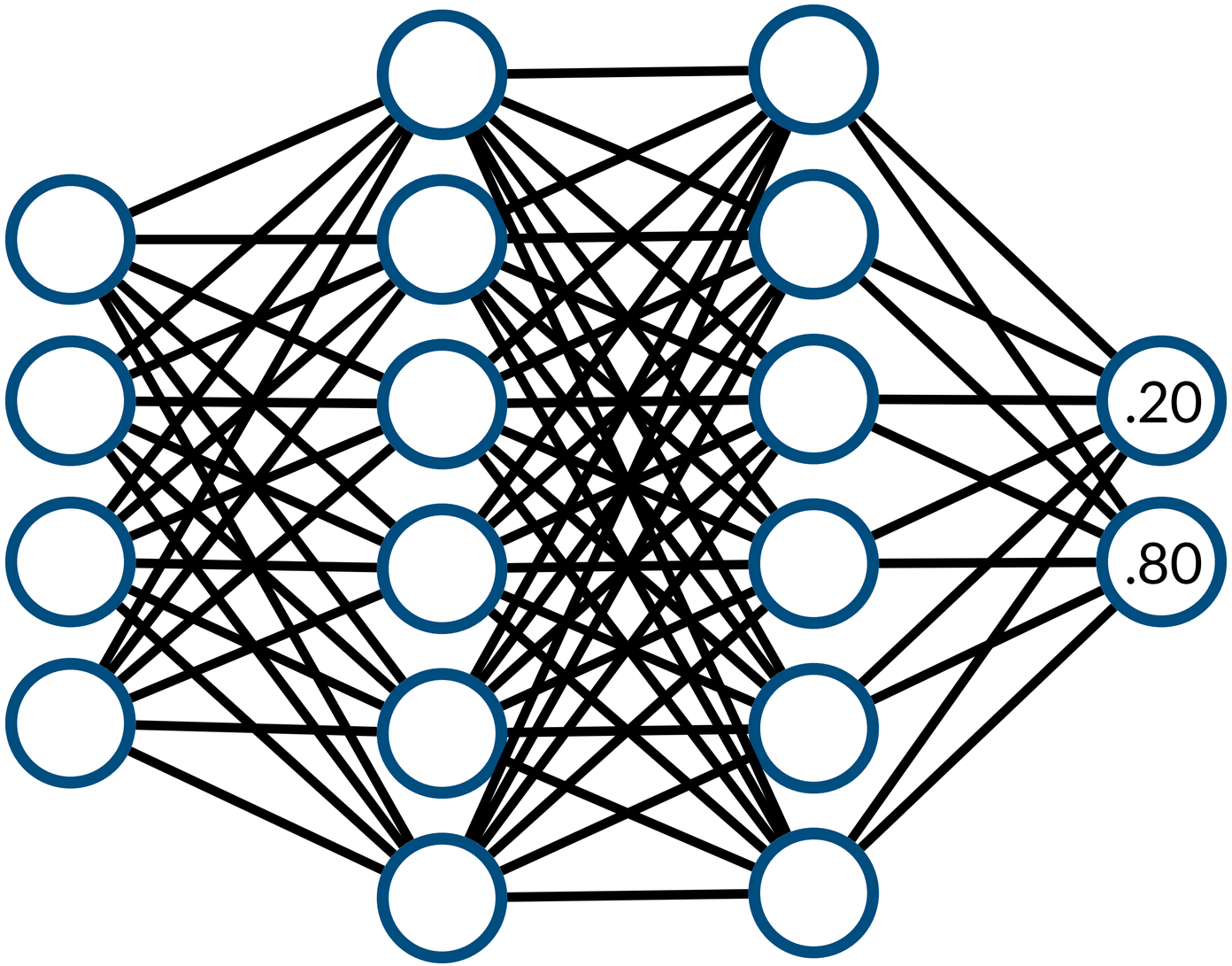

Cloud Cover vs Rain
Rain (1 = Yes, 0 = No)
Cloud Cover (%)




Improved Non-Linear Fit with Neural Network (ReLU Activation)
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
X
Original Data
Linear Model (No Activation)
Improved Neural Network (ReLU Activation)

Activation Function


1
.23
.52
.61

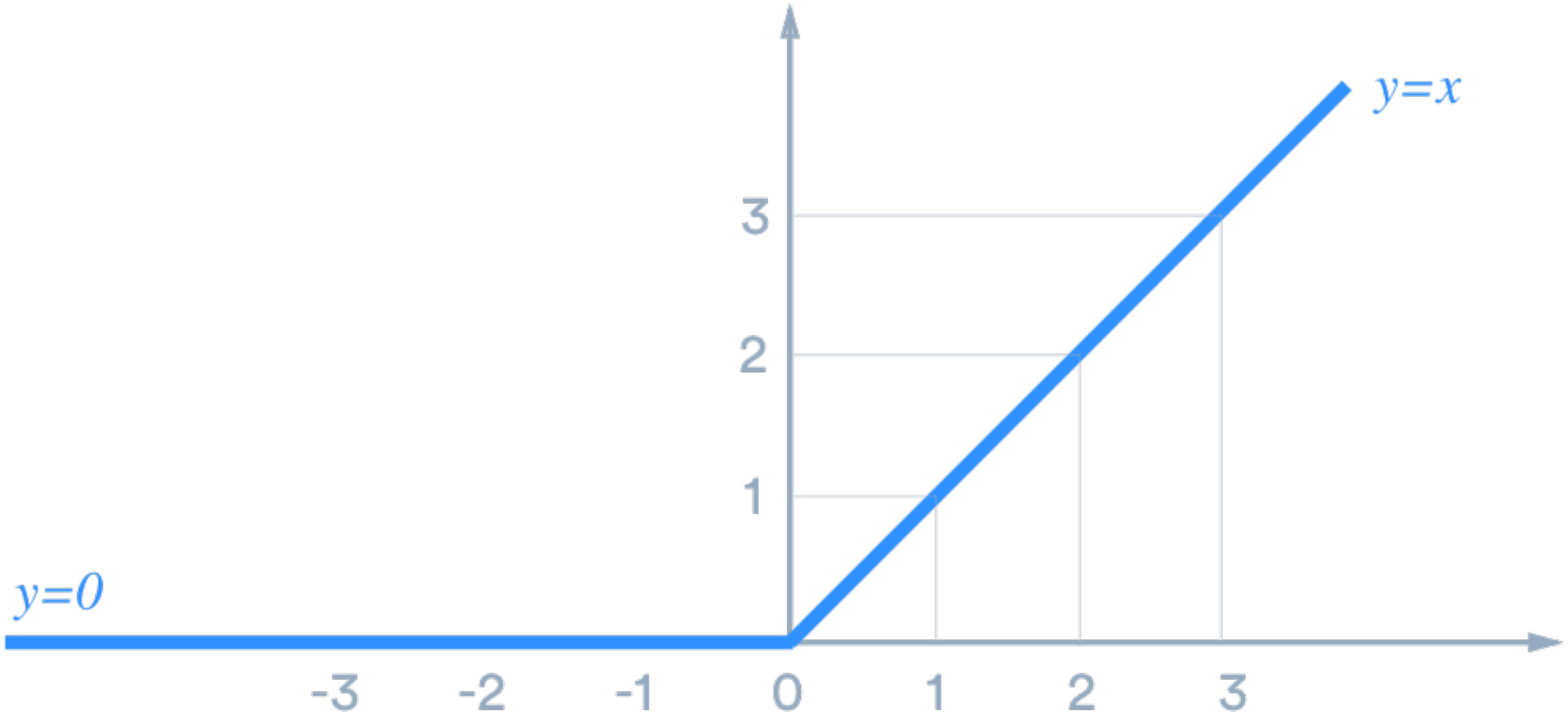

.20
.80

Activation Function

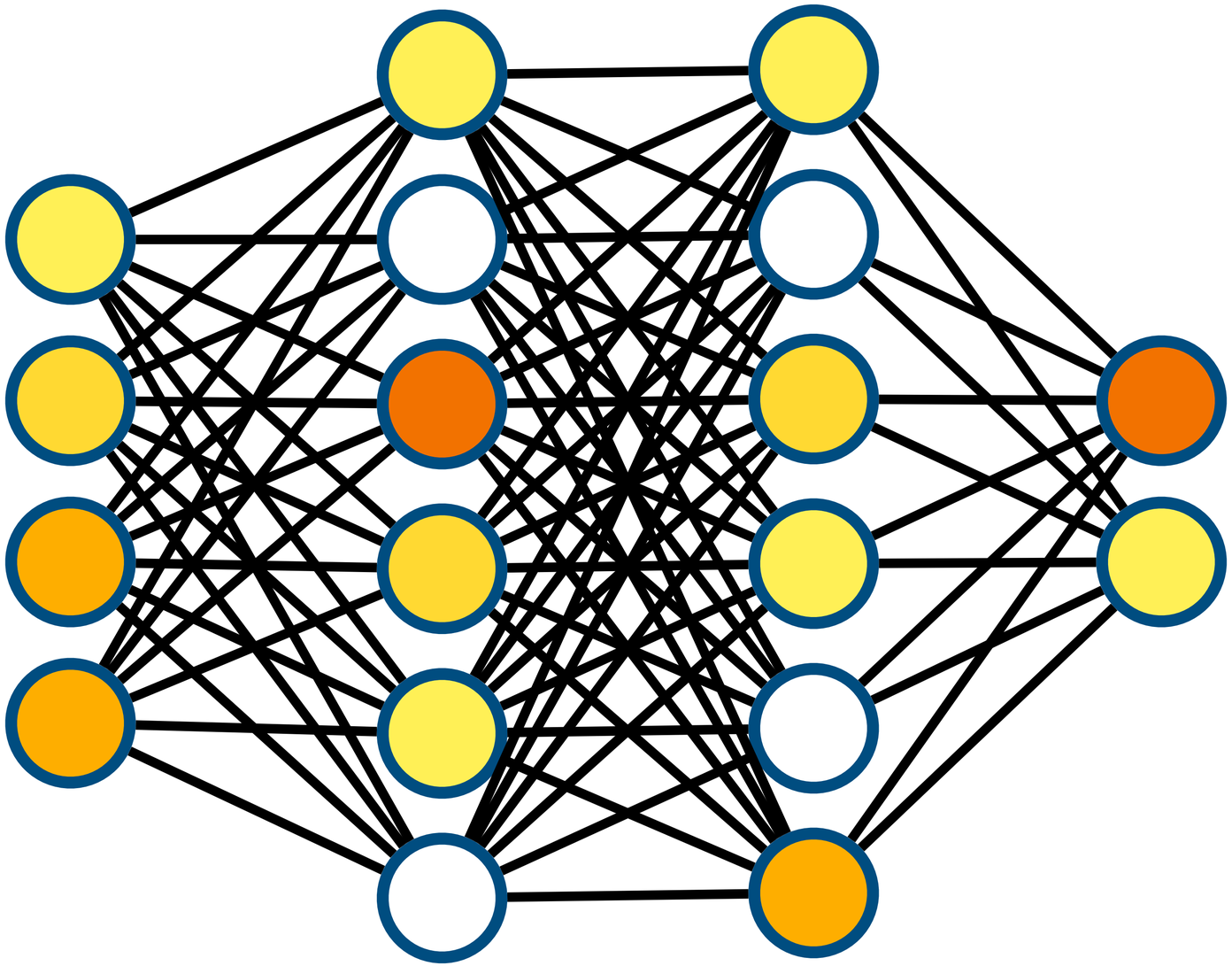
- **Definition** - A non-linear function which decides if the output of a neuron should be propagated or allowed to continue forward
- Without activation functions, neural networks will not be able to solve non-linear problems, without it, it is the same a linear regression

- Functions include:
  - Sigmoid (final - candidate)
  - `tanh`
  - `relu`
  - `selu`
  - leaky `relu`
  - `softmax` (final)

Rectified Linear Unit



https://medium.com/@sonish.sivarajkumar/relu-most-popular-activation-function-for-deep-neural-networks-10160af37dda



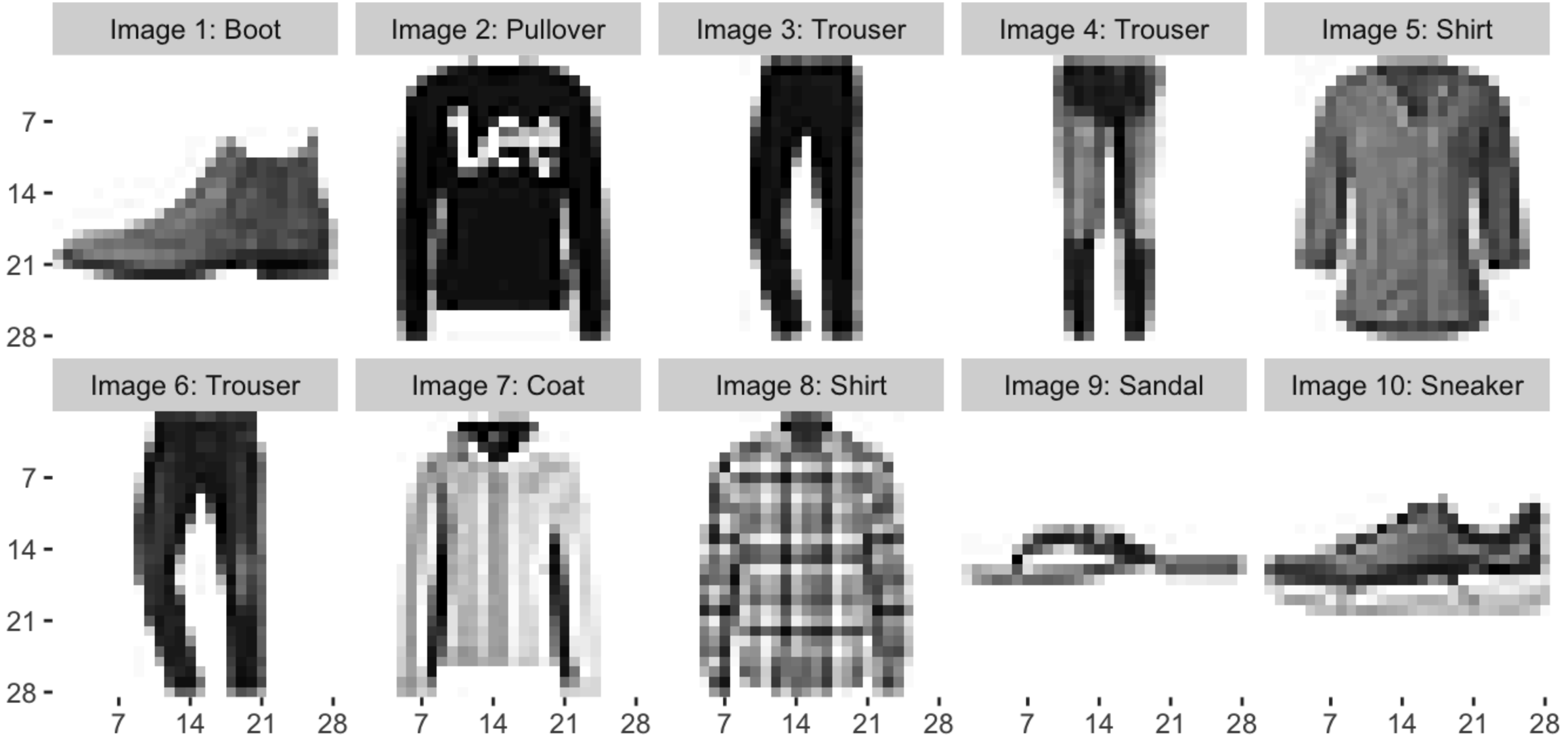


Image 1: Boot
Image 2: Pullover
Image 3: Trouser
Image 4: Trouser
Image 5: Shirt
Image 6: Trouser
Image 7: Coat
Image 8: Shirt
Image 9: Sandal
Image 10: Sneaker
7
14
21
28

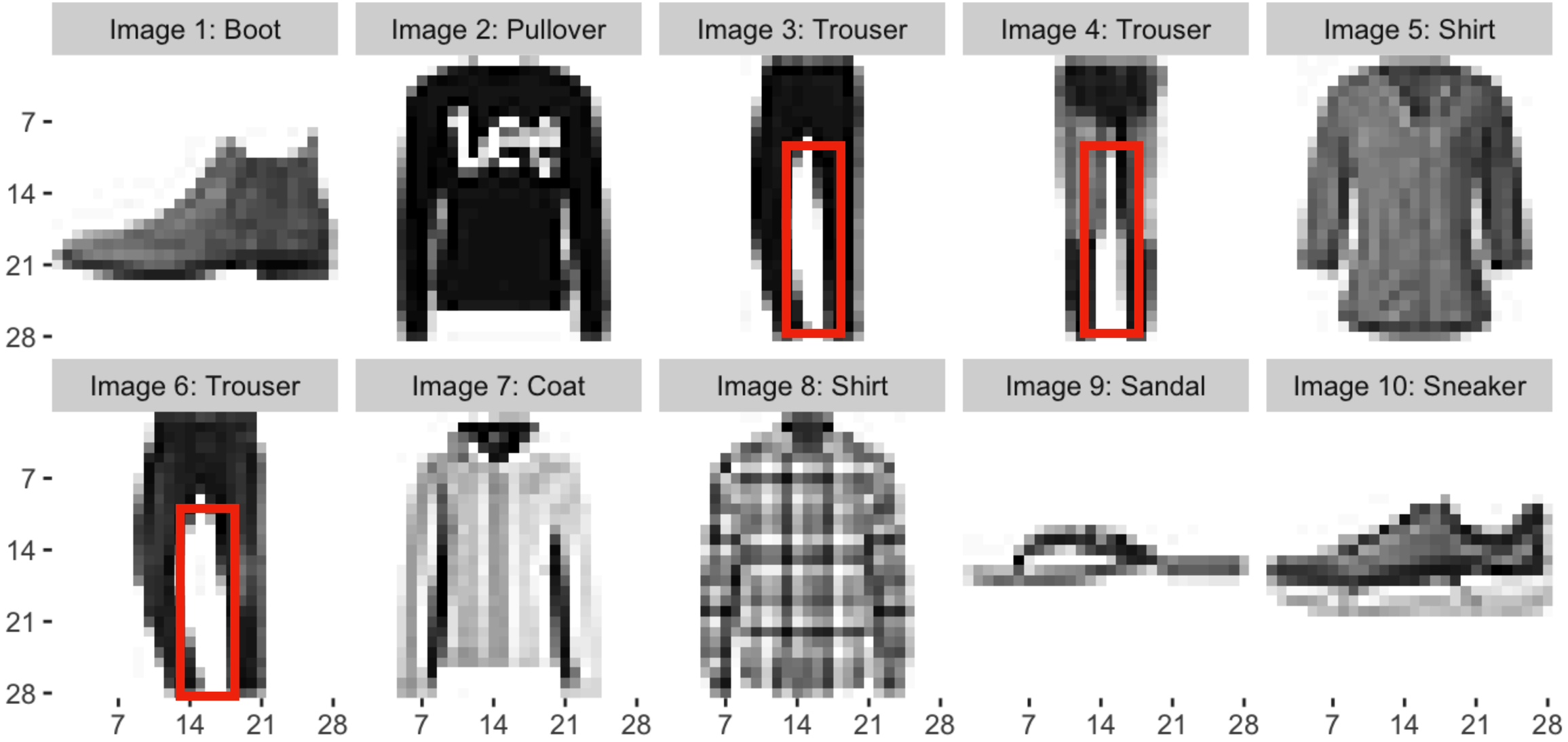
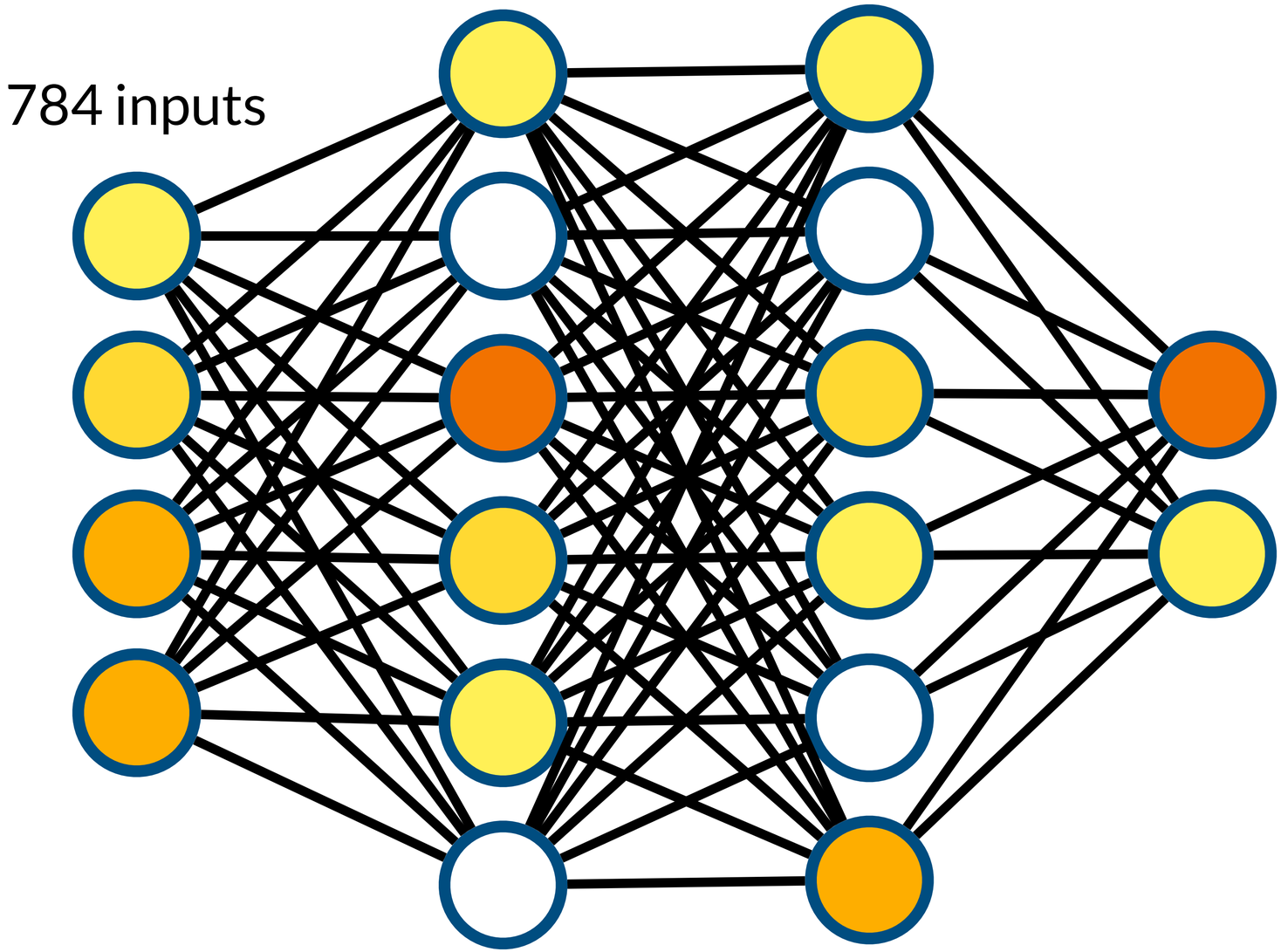
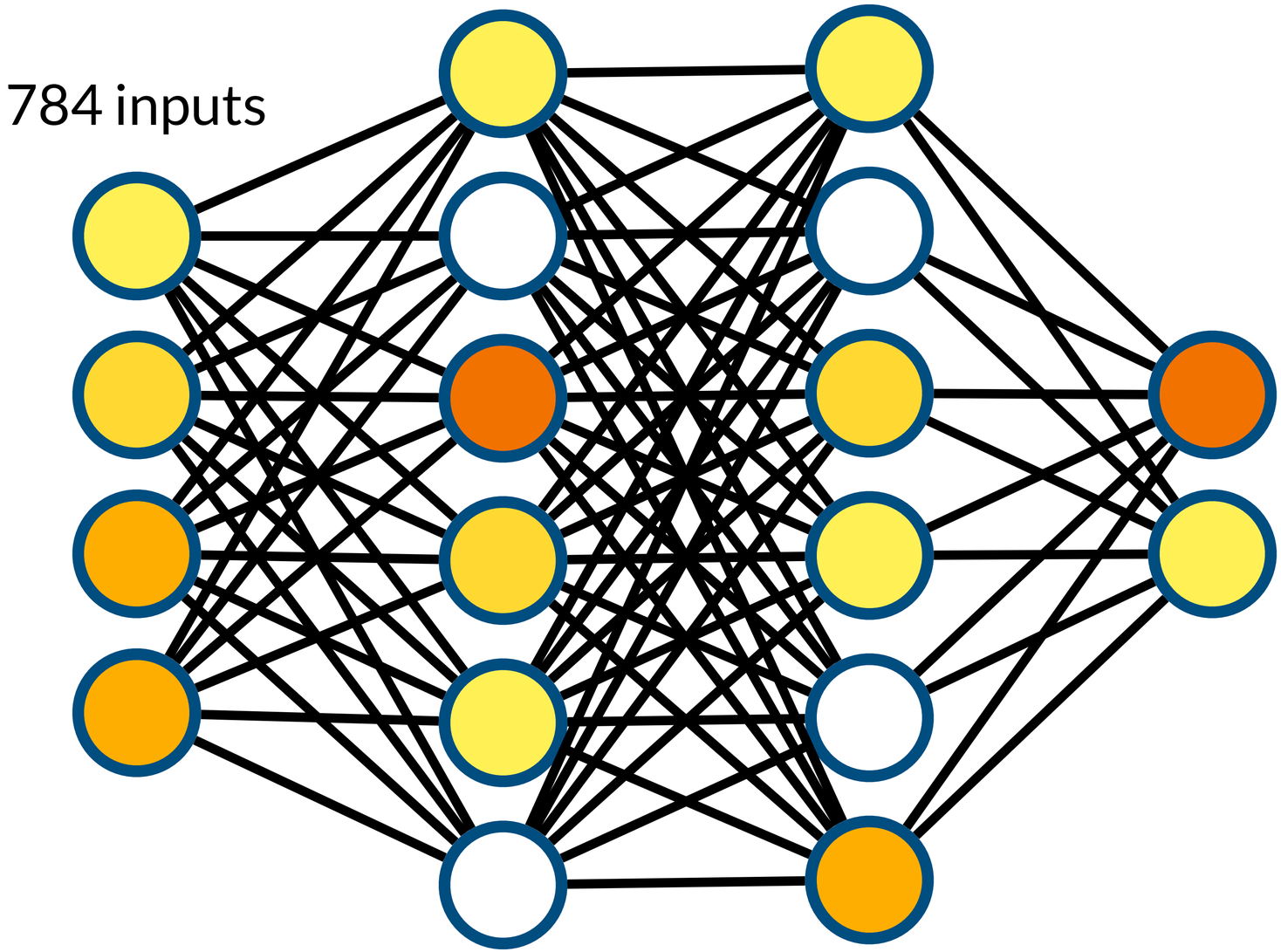
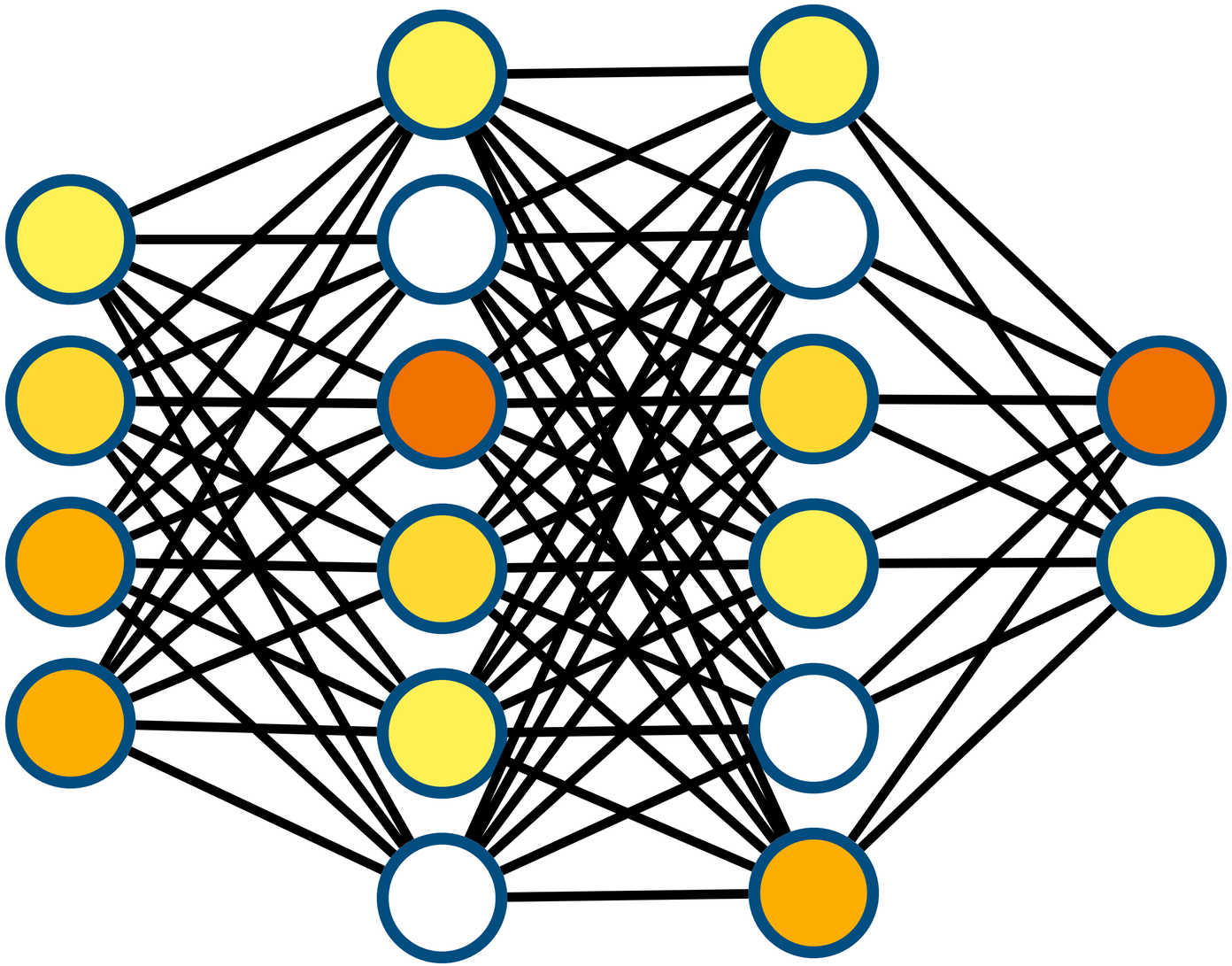
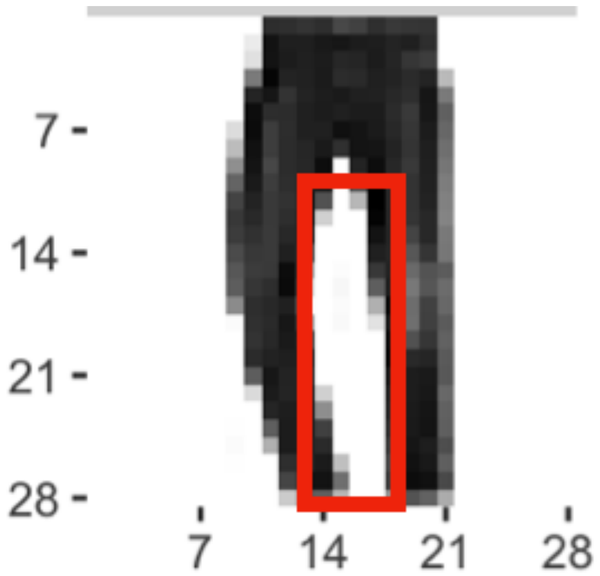


Image 1: Boot
Image 2: Pullover
Image 3: Trouser
Image 4: Trouser
Image 5: Shirt
Image 6: Trouser
Image 7: Coat
Image 8: Shirt
Image 9: Sandal
Image 10: Sneaker
7
14
21
28


784 inputs


784 inputs

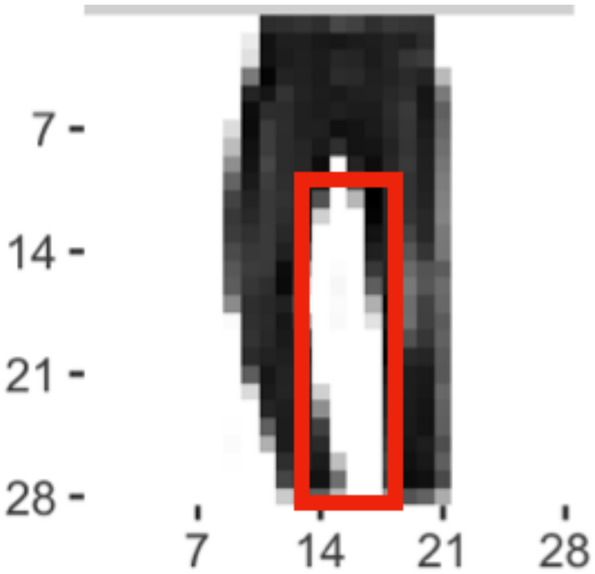




values = $(a1, a2, a3 \ldots a784)$

weighted values = $(w1 * a1 + w2 * a2, \ldots, w784 * a784)$

The bias is a correctional shift



values = $(a1, a2, a3 \ldots a784)$

weighted values = $(w1 * a1 + w2 * a2, \ldots, w784 * a784 \textbf{ - bias})$

through an activation function = $f(w1 * a1 + w2 * a2, \ldots, w784 * a784 \textbf{ - bias})$

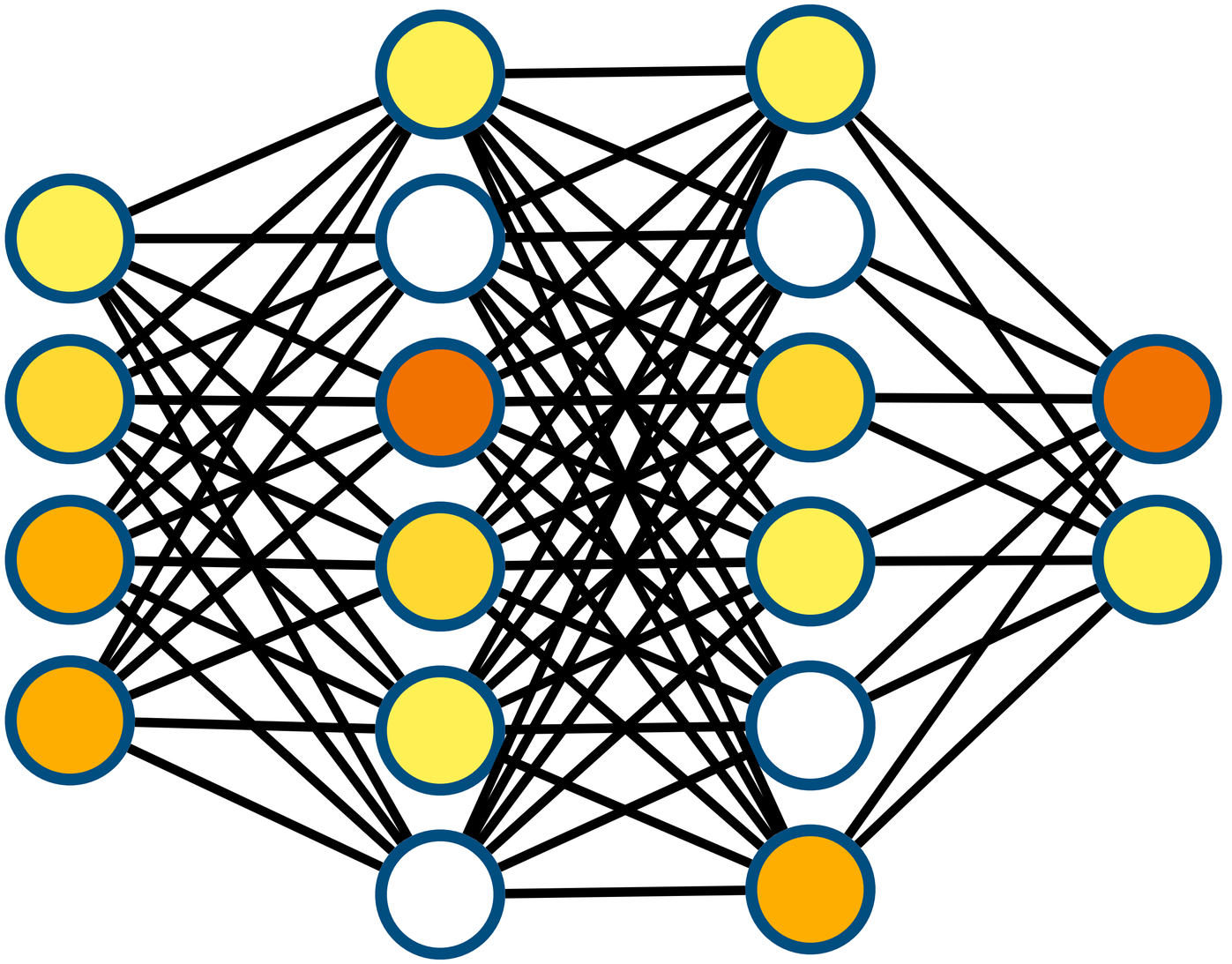
$$f(w1 * a1 + w2 * a2, ..., w784 * a784 - \mathbf{bias})$$

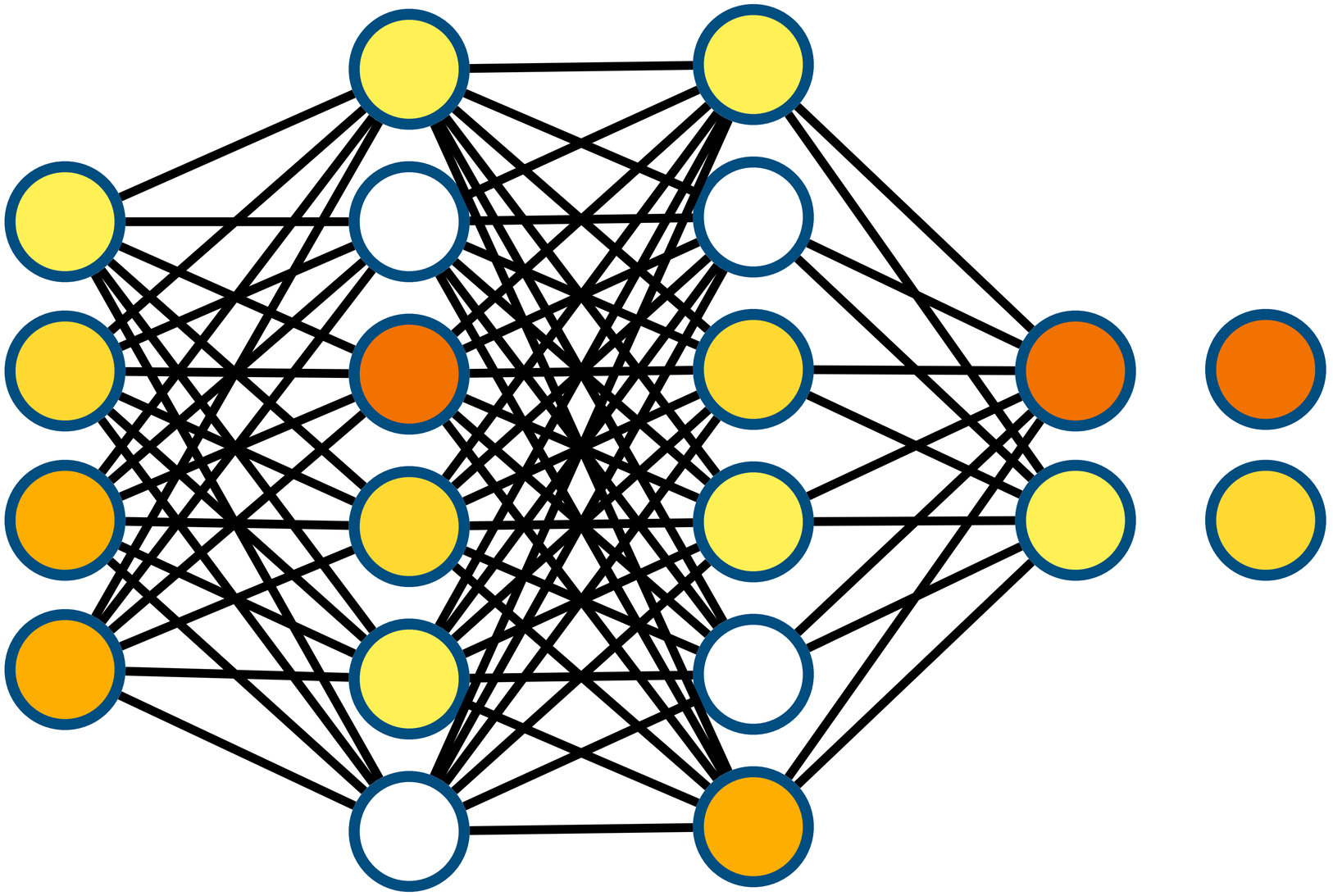
$$f(\mathbf{W} + \mathbf{b})$$

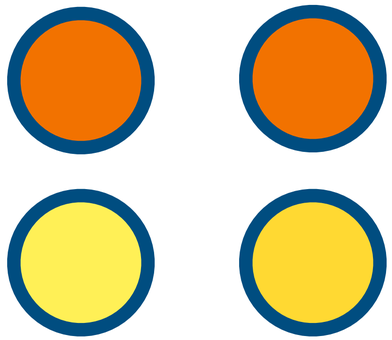
$$ReLu(w1 * a1 + w2 * a2, ..., w784 * a784 - \mathbf{bias})$$

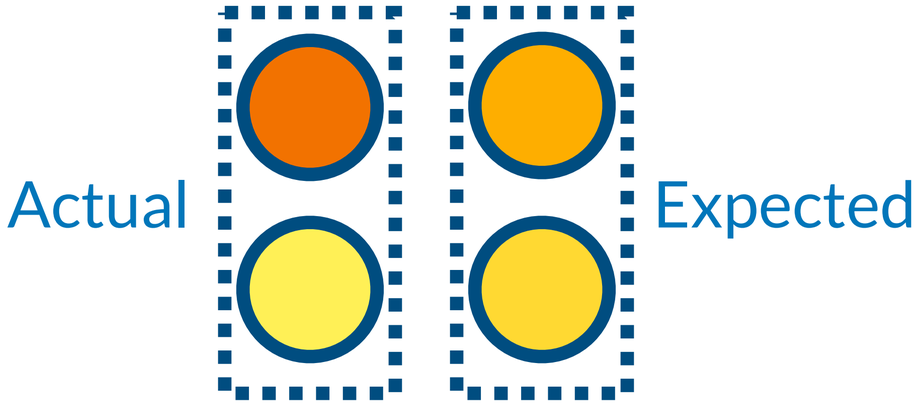
$$ReLu(\mathbf{W} + \mathbf{b})$$

Cost Function



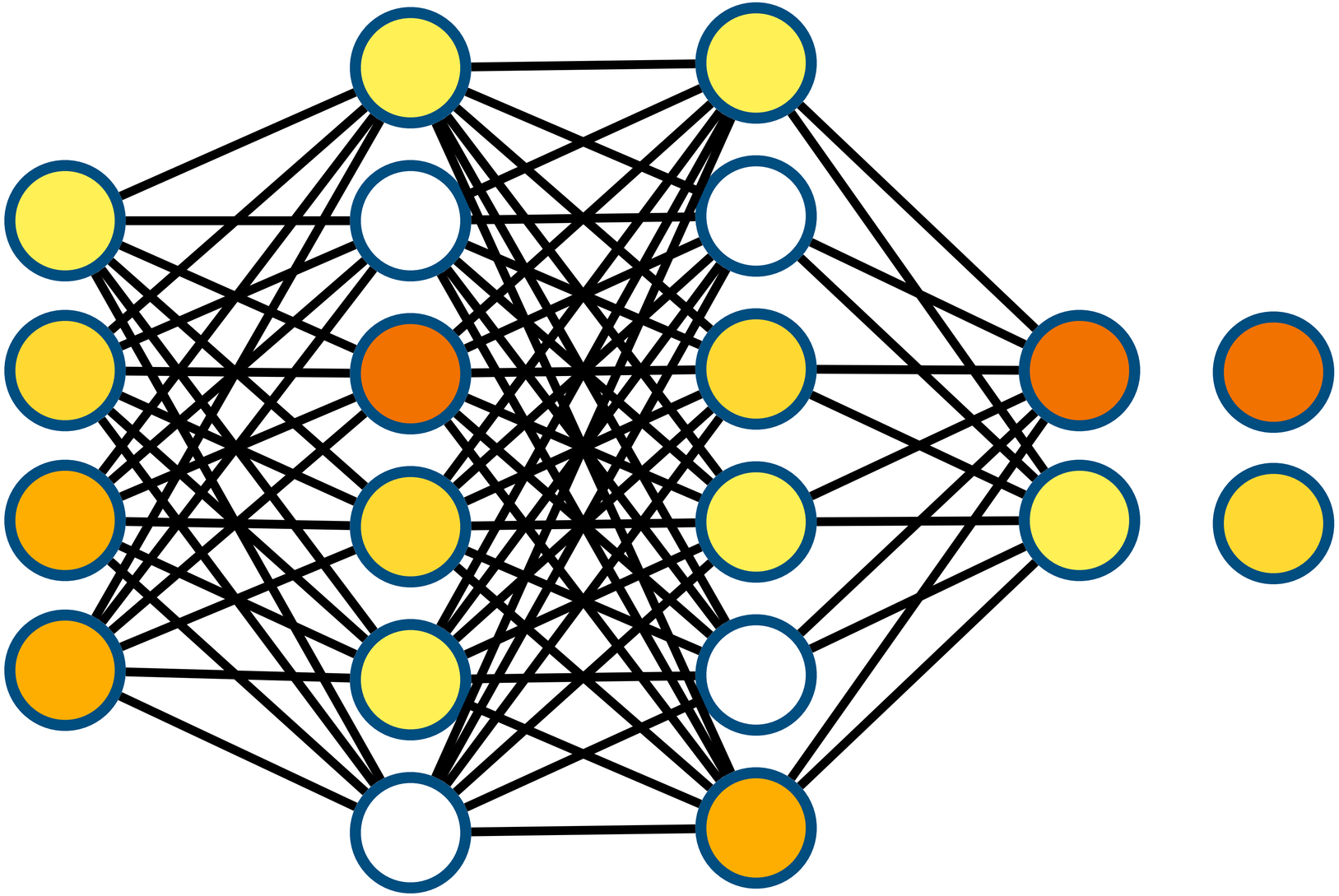




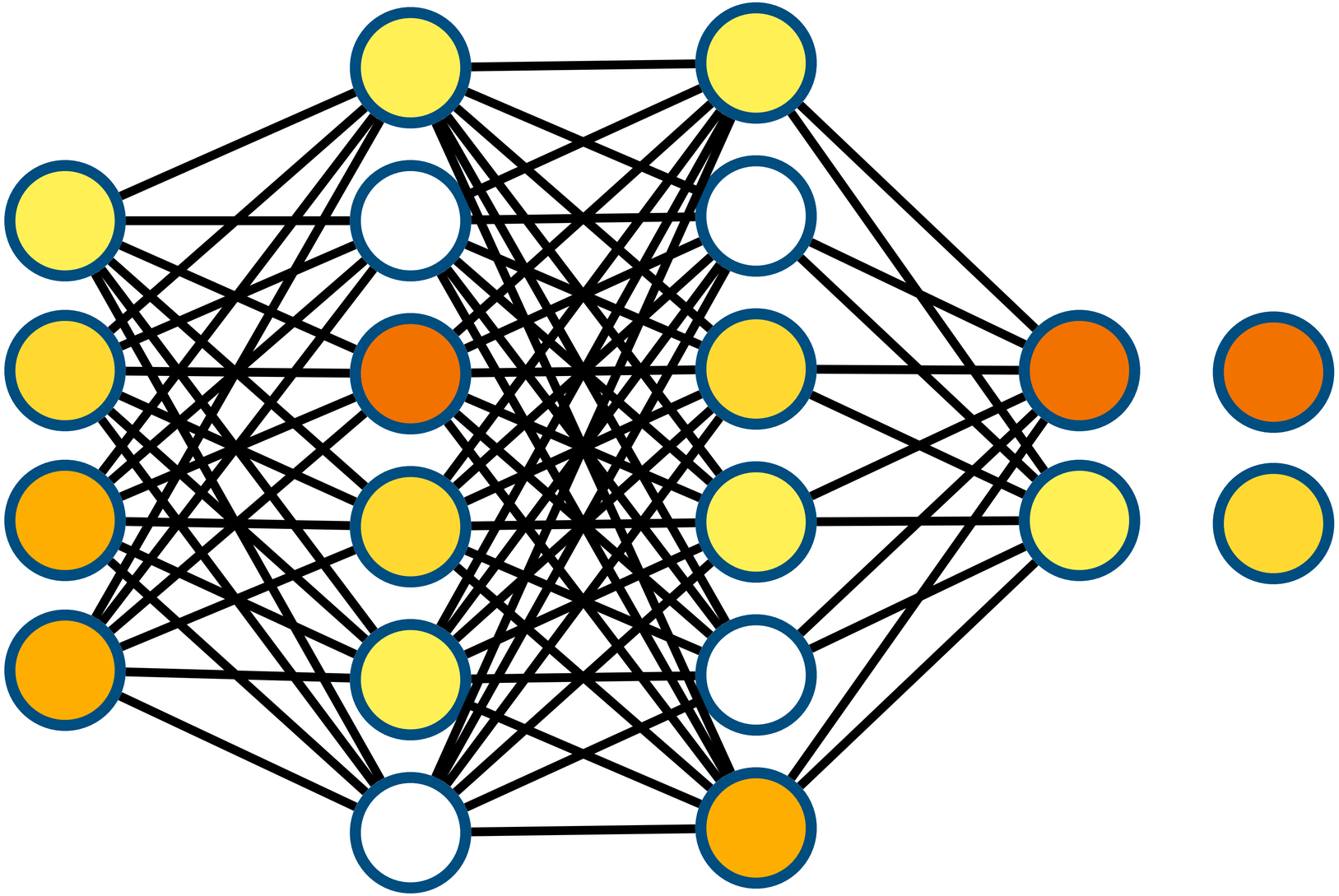


The "cost" is what is the difference: (actual - expected)^2

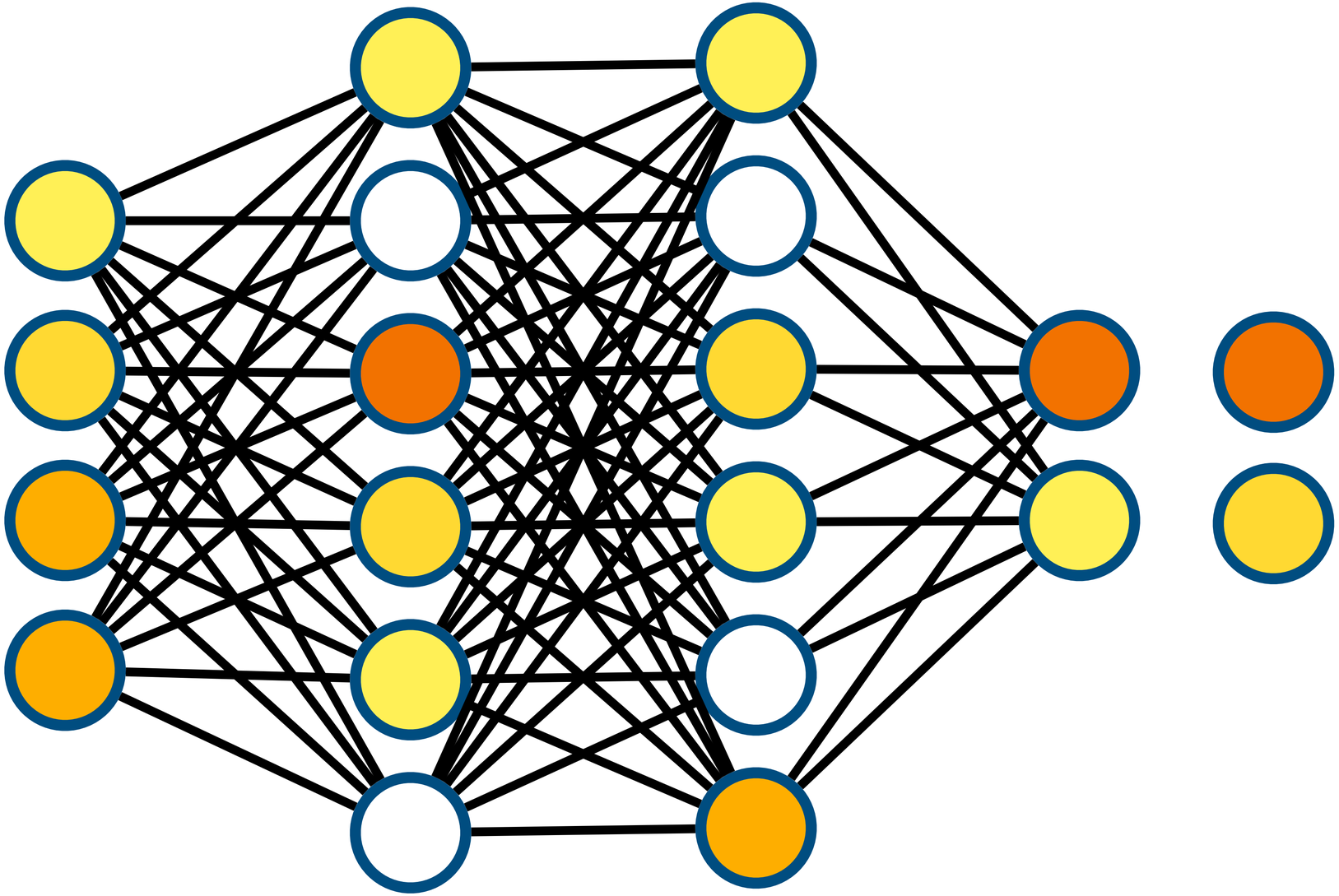
Backpropagation



Goal: How can we minimize the cost function?



Go back and "tweak" each of the neurons and measure the response



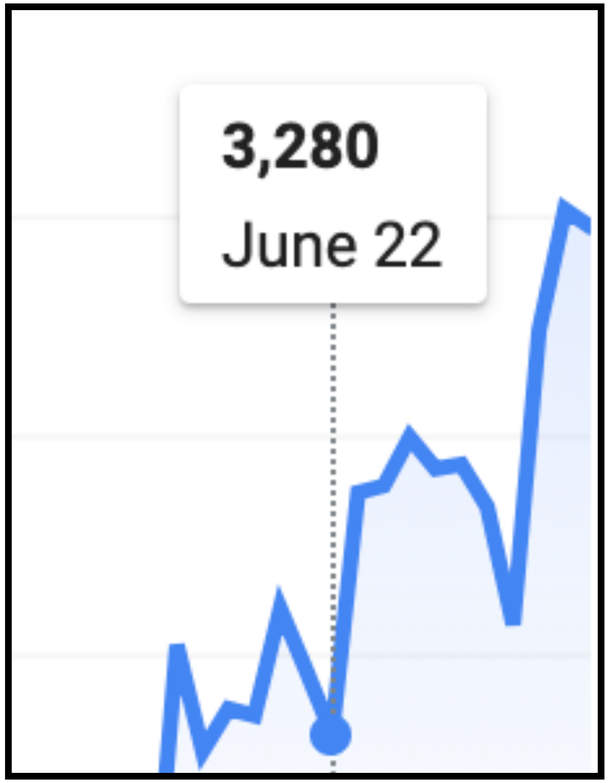
The Neural Network will determine which is more sensitive with small adjustments

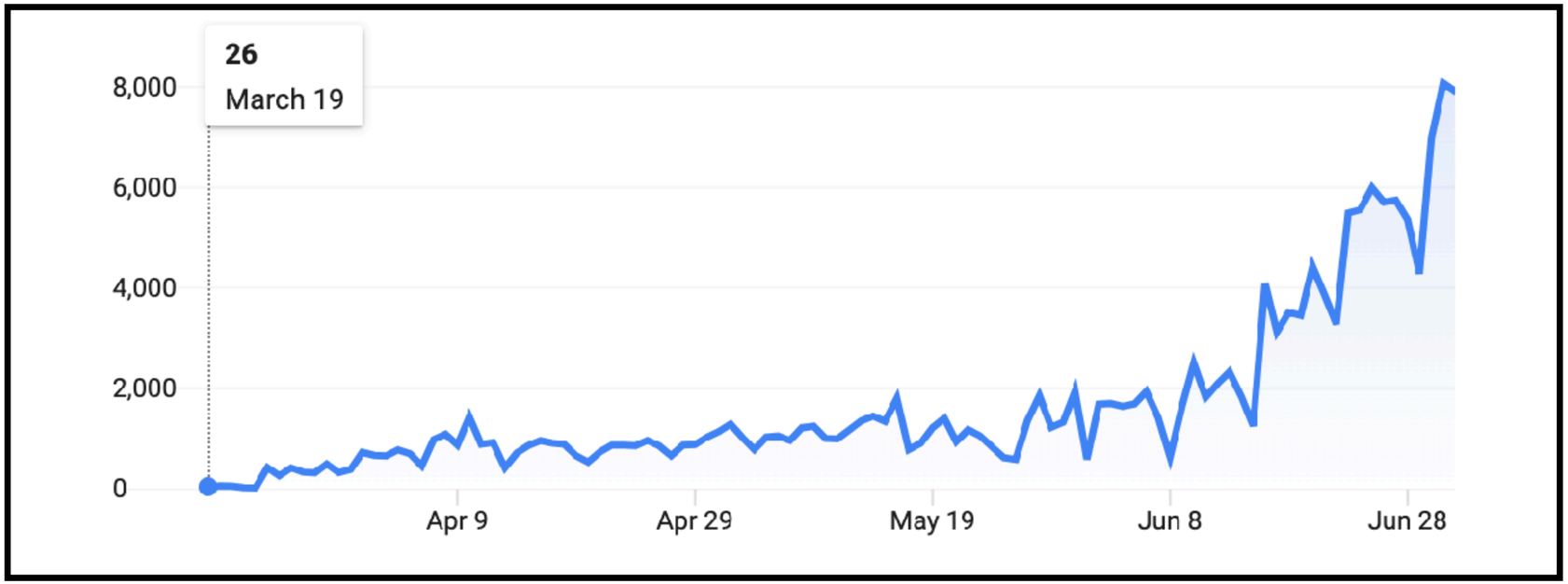

ASSIGN
SOLO
ON
PF
EQ
AUX 3-4
1-2 MUTE
3-4 MUTE

Gradient Descent

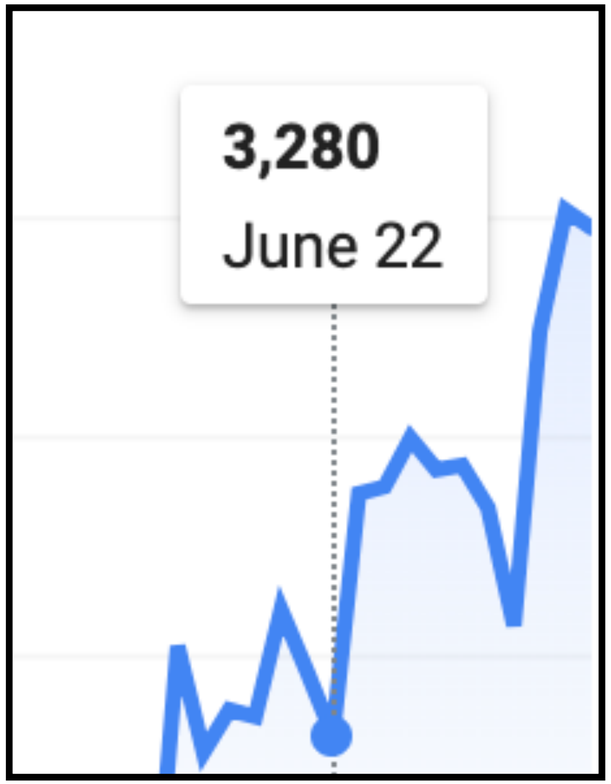
Local Minima/Global Minima

Local Minima vs Global Minima



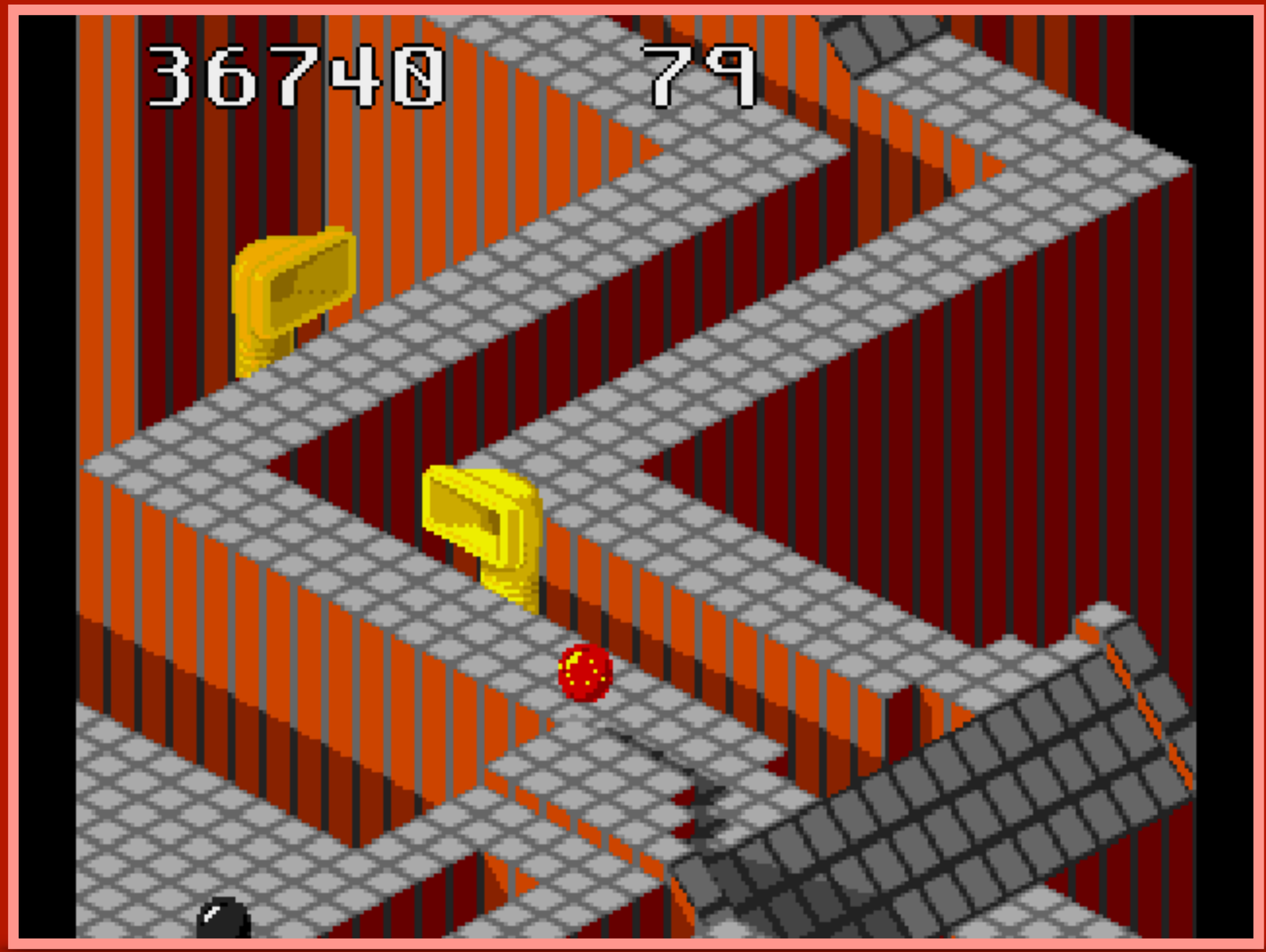


Local Minima vs Global Minima

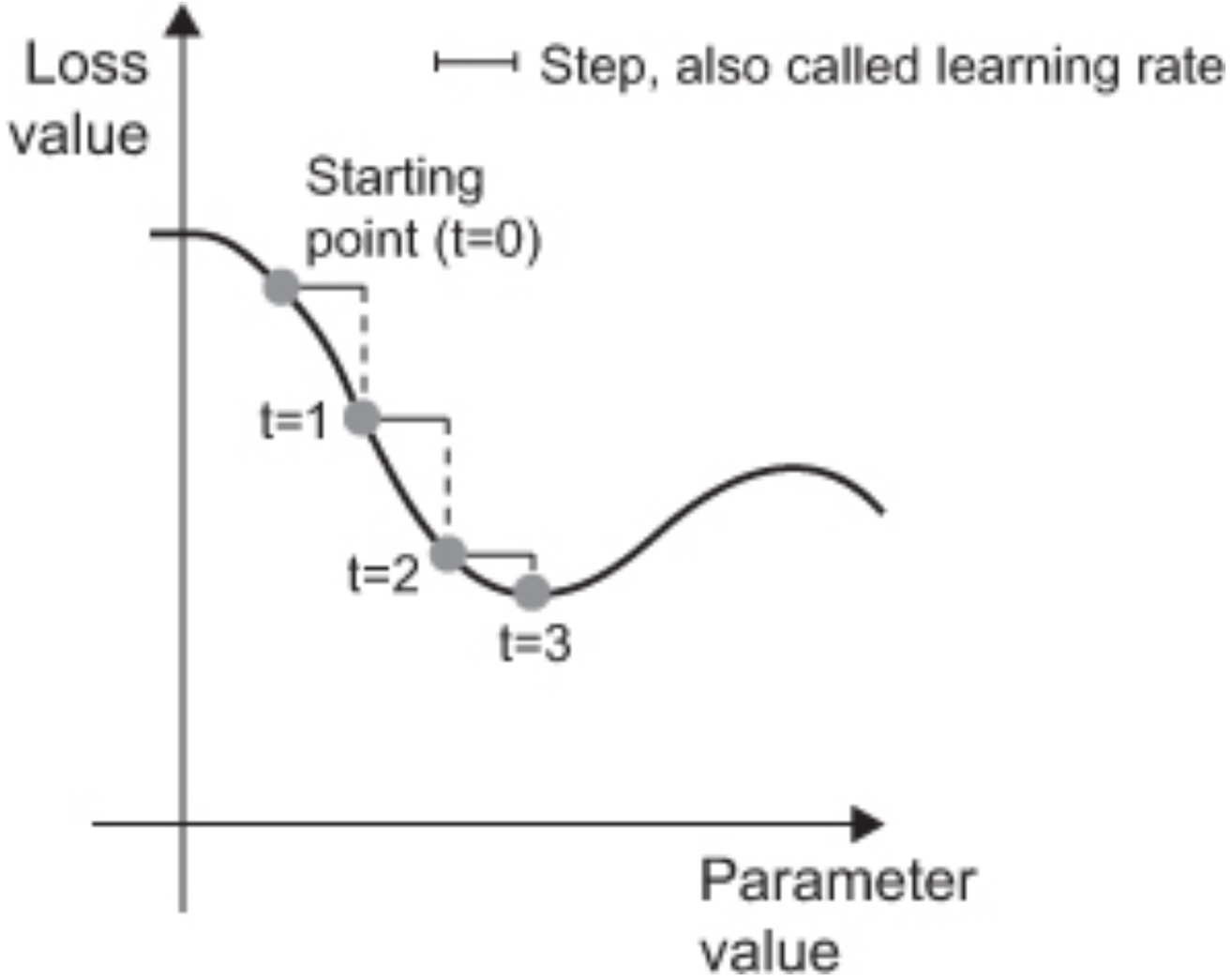


Definition - We pass the same dataset multiple times to the network in order to find optimal weights; one training session.

The More the Merrier - with just a few epochs you can reach only a local minimum, but with more epochs, you can reach a global minimum or at least a better local minimum

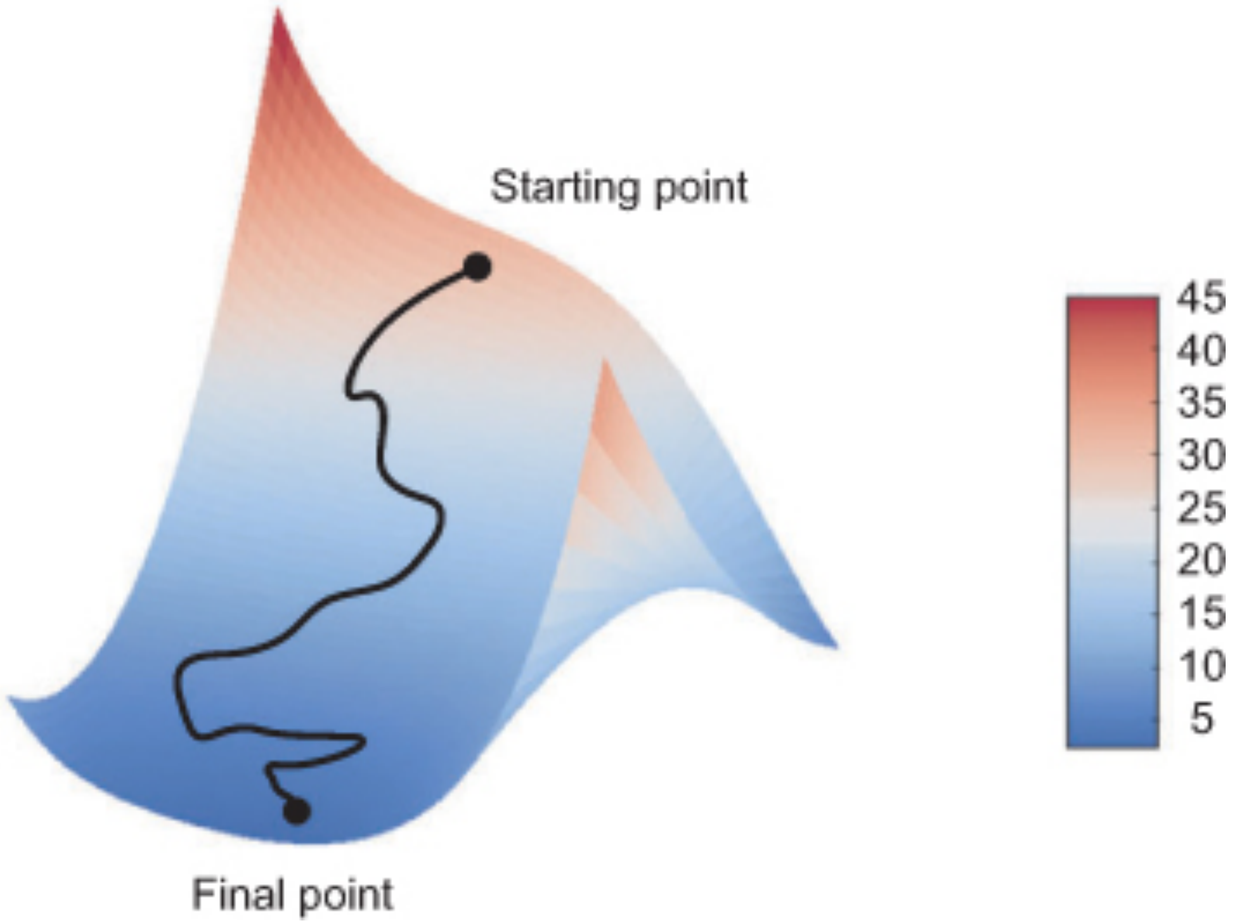

36740
79

Back to Gradient Descent



Each Epoch we are learning and trying to lower our cost function

Deep Learning with Python by Francois Chollet



3d gradient descent, finding the lowest cost function

Deep Learning with Python by Francois Chollet

Demo: Jupyter Lab & TensorFlow

Other Neural Networks Patterns

How Many Neurons?

- Determining the height, width, of a neural networks depends on the *complexity* of what you are trying to solve, the *amount of data* that you have, and the *resources* that you have.

  - Adding more hidden layers increases the model's ability to **learn complex hierarchical features**

  - **Increasing the number** of neurons in each layer helps the network capture more features at each level.

Other Uses for Neural Networks

Neural Network Types

Neural Networks are the backbone of many AI applications across different fields like **computer vision, language processing, healthcare, and more**. Let’s explore the key types and their applications.

Convolutional Neural Networks

- CNNs excel at image recognition, classification, and object detection
- Think of rod and cones in your eyes detecting a pattern, and think of the complexity of being able to see light vs. dark
- Applications:
  - Facial recognition (e.g., unlocking your phone)
  - Medical imaging (e.g., detecting tumors in X-rays)
  - Autonomous vehicles (e.g., recognizing pedestrians and road signs)

Recurrent Neural Networks

- RNNs process data where order matters, like time series or language.
- State function of the input feeds back to 2nd layer so we can capture sequences of words: *We have seen this before*
- Applications:
  - Speech recognition (e.g., Siri, Google Assistant)
  - Language translation (e.g., Google Translate)
  - Stock price prediction

Natural Language Processing

- Understanding and Generating Human Language
- This of how complex language is, “The book is on the table”. Later “It is brown”, what does “it” mean?
- Applications
  - Powered by: RNNs, Transformers, and Attention Mechanisms.
  - Chatbots (e.g., virtual assistants)
  - Sentiment analysis (e.g., detecting emotions in reviews)
  - Text generation (e.g., GPT models writing text)

Generative Adversarial Networks

- GANs have two networks (generator + discriminator) that **create realistic data** from scratch.
- Deepfake generations
- Art and music creation (e.g., AI-generated paintings)
- Enhancing low-res images (e.g., image upscaling)

Autoencoders

- Neural Network designed to compress data (encoding) and then reconstruct data to original
- Applications:
  - Data Compression and Anomaly Detection
  - Learn to compress and reconstruct data efficiently
  - Dimensionality reduction (e.g., simplifying complex datasets)
  - Denoising images

Reinforcement Learning

- Decision making and control but with complex scenario.
- Train agents to make decisions by rewarding desirable outcomes. It does so by assigning credit: *What is the value by taking a step forward?*
- ***When teaching a teenager to drive, the teenager has a semblance of what not to do, like not drive off a cliff. ML doesn't know that.***
- Applications:
  - Game AI (e.g., AlphaGo beating human players)
  - Robotics (e.g., robots learning to walk)
  - Self-driving cars (e.g., navigating dynamic environments)

What do Neural Networks have to do with LLMs?

Neural Networks and LLMS

- Neural Networks are the Foundation of LLMs: LLMs (Large Language Models) are built on deep neural networks—specifically, architectures like Transformers.
- Neural networks enable LLMs to learn patterns in language, grammar, and context from massive datasets.
- Scale and Complexity: LLMs have billions of parameters and require massive computing power to process and learn from large datasets.
- Models like ChatGPT analyze vast amounts of data to capture intricate patterns in language, enabling advanced capabilities like context understanding and text generation.

How Neural Networks Power LLMS

- LLMs use deep architectures with millions or billions of parameters.
- Each layer of the neural network refines its understanding of the data.
- Transformers rely on self-attention mechanisms to understand relationships between words in a sentence, regardless of their position.
- This structure allows LLMs to handle complex language tasks like translation, summarization, and text generation.

Transformers?

- Instead of processing words one by one (like older models), **self-attention** lets the model weigh the importance of all words in a sentence simultaneously.

- Example: In the sentence “The cat sat on the mat because it was tired,” the model uses self-attention to figure out that “it” refers to “the cat.”

https://medium.com/@sharanharsoor/mastering-transformers-a-comprehensive-guide-to-transformer-architecture-questions-7694e52ba949

Transformers

- Large Scale Modeling, allows use to parallel to larger scale to Process and generate complex human language using attention mechanisms.
- Attention Mechanisms are relationships of words and tokens
- Applications:
  - Algorithms like GPT-4, BERT (*which is not an LLM, but a Transformer Architecture*)
  - Machine translation (e.g., Google Translate)
  - Summarization and Q&A Systems

So, wait, what is difference between ML and AI?

Ai is about replicating human intelligence

ML is about learning from your data

Conclusion

Thank You

- Email: dhinojosa@evolutionnext.com
- Github: https://www.github.com/dhinojosa
- Linked In: http://www.linkedin.com/in/dhevolutionnext