

# MACHINE LEARNING IN THE BROWSER

# LIAN LI



# LIAN LI





▶ Lian Li



June 8, 2015 ·

If I was a computer,..I would look like this,...



## Lian Li PC-Q33B schwarz

Das sehr kompakte Mini Tower-Gehäuse Lian Li PC-Q33B hat bereits einen vorinstallierten 120-mm-Lüfter. Das PC-Q33B bietet Platz für zwei 3,5-Festplatten, drei 2,5-Festplatten oder SSDs sowie ein Mini-ITX- oder Mini-DTX-Mainboard. Dieses

ALTERNATE.DE

2 Likes 1 Comment





▶ Lian Li



June 8, 2015 ·

If I was a computer,..I would look like this,...



## Lian Li PC-Q33B schwarz

Das sehr kompakte Mini Tower-Gehäuse Lian Li PC-Q33B hat bereits einen vorinstallierten 120-mm-Lüfter. Das PC-Q33B bietet Platz für zwei 3,5-Festplatten, drei 2,5-Festplatten oder SSDs sowie ein Mini-ITX- oder Mini-DTX-Mainboard. Dieses

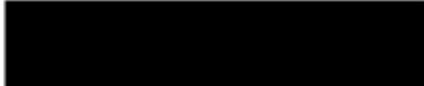
ALTERNATE.DE

2 Likes 1 Comment



You and





▶ Lian Li



June 8, 2015 ·

If I was a computer,..I would look like this,...



## Lian Li PC-Q33B schwarz

Das sehr kompakte Mini Tower-Gehäuse Lian Li PC-Q33B hat bereits einen vorinstallierten 120-mm-Lüfter. Das PC-Q33B bietet Platz für zwei 3,5-Festplatten, drei 2,5-Festplatten oder SSDs sowie ein Mini-ITX- oder Mini-DTX-Mainboard. Dieses

ALTERNATE.DE

2 Likes 1 Comment



You and



Like · Reply · 2y

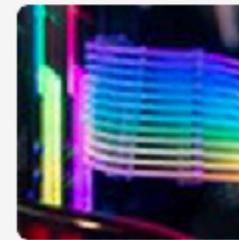


@Lian Li <https://tweakers.net/nieuws/139365/lian-li-voorziet-24-en-8-pinsvoedingskabels-van-fiberdraden-met-rgb-leds.html>

 Tweakers

### Lian Li voorziet 24- en 8-pinsvoedingskabels van fiberdraden met rgb-leds

Lian Li demonstreert op de Computex voedingskabels met rgb-verlichting. De kabels geven zelf geen licht, maar zijn voorzien van een opzetstukje met leds en doorzichtige fiberdraden waar het licht door schijnt.



12:57 PM

is that our Lian Li? or someone else?

1:10 PM

<http://www.lian-li.com/> 🤖

# LIAN LI



**@CHIMNEY42**



**CHIMNEY42**

**MASTODON.SOCIAL**

**@CHIMNEY42**

**LIAN LI**

**SOFTWARE ENGINEER  
@ CONTAINER SOLUTIONS**

**@CHIMNEY42**

**LIAN LI**

**MACHINE LEARNING  
ENTHUSIAST**

**@CHIMNEY42**

# MACHINE LEARNING

**Expert Systems**

**Machine Learning**

➔ Input  
→

Expert Systems

Machine Learning





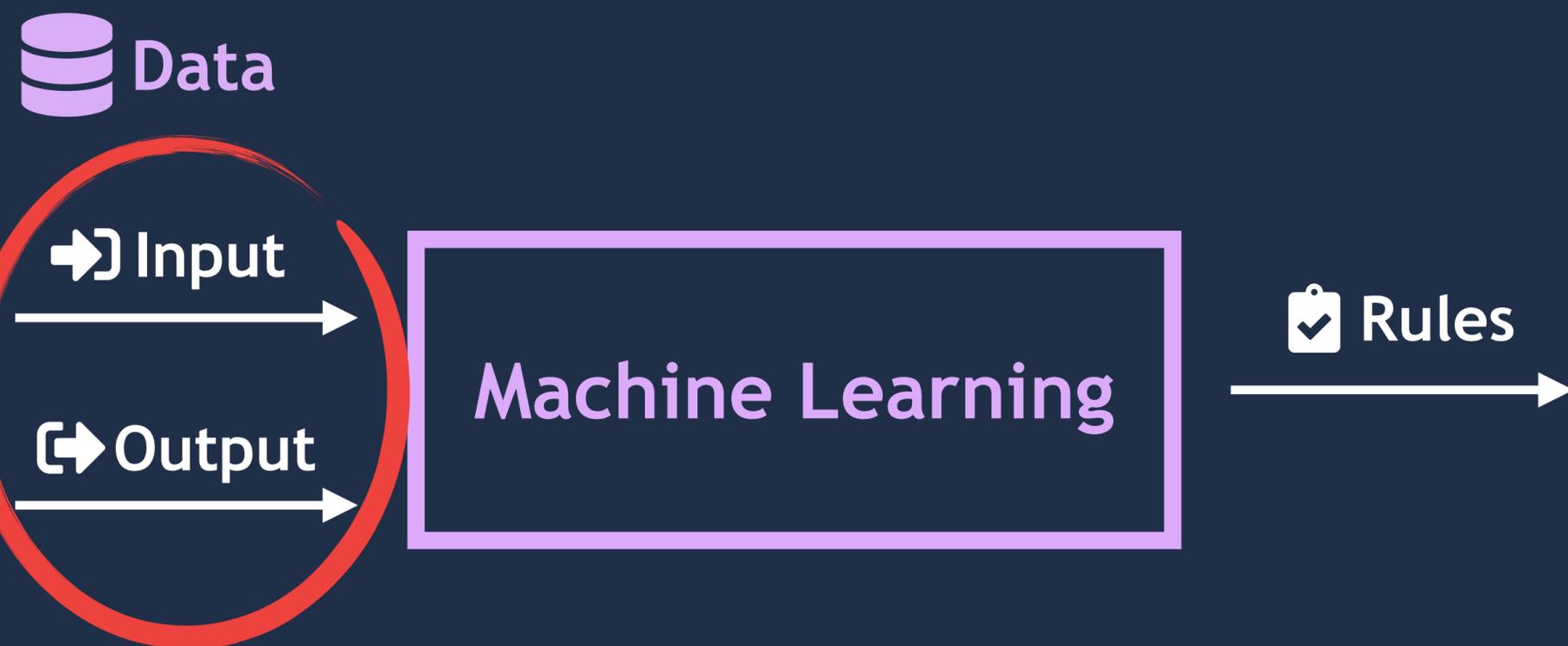
Machine Learning











# MACHINE LEARNING

# MACHINE LEARNING

Experience

# MACHINE LEARNING

Experience

(# of datapoints)

# MACHINE LEARNING

Experience

(# of datapoints)

Repetition

# MACHINE LEARNING

Experience

(# of datapoints)

Repetition

(# of training iterations)

# CLIENT-SIDE ML

# CLIENT-SIDE ML

Speed

# CLIENT-SIDE ML

Speed

Availability

# CLIENT-SIDE ML

Speed

Availability

Security

[@CHIMNEY42](#)

# COMPLEMENTARY COLOURS

[@CHIMNEY42](#)

**Input**

**Input**

**Function**

**Input**

**Function**

**Output**

Input



Function

Output

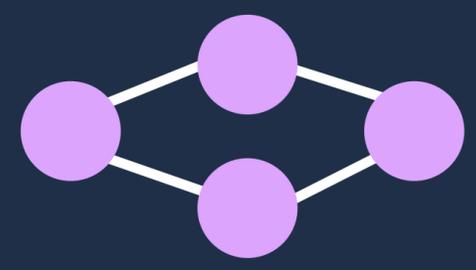
Features

Input



Features

Function



Network

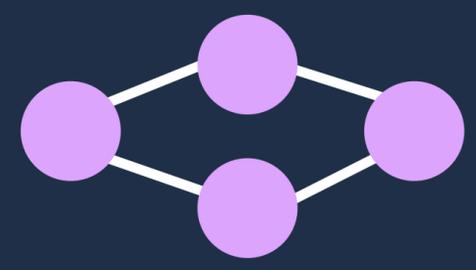
Output

Input



Features

Function



Network

Output



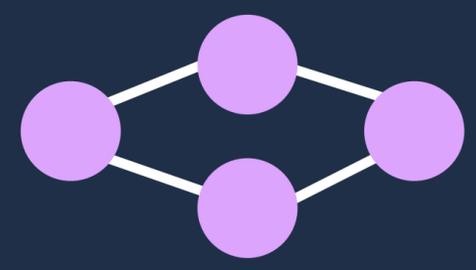
Prediction

Input



Features

Function



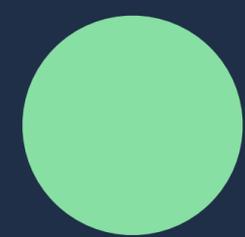
Network

Output



Prediction

Desired  
output

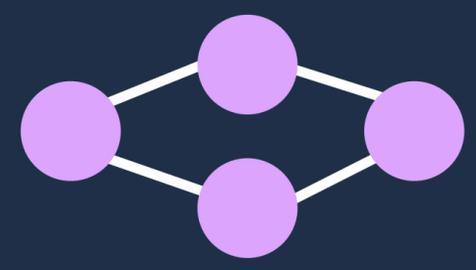


Input



Features

Function



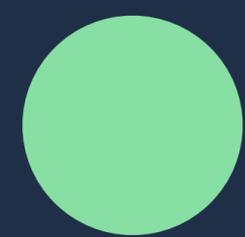
Network

Output



Prediction

Desired  
output



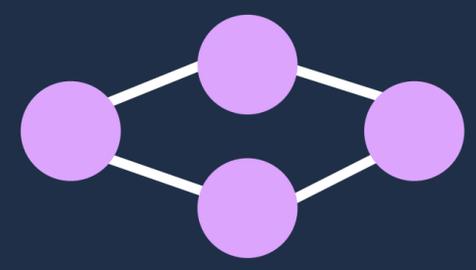
Target

Input



Features

Function



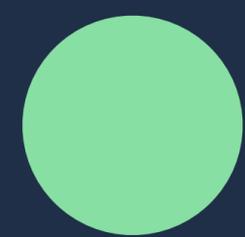
Network

Output

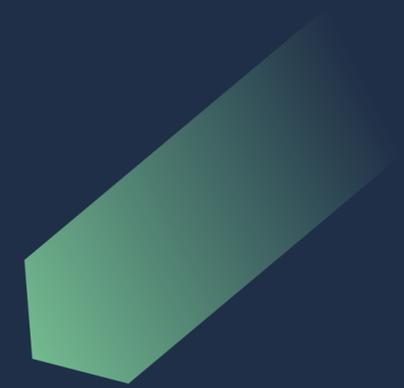
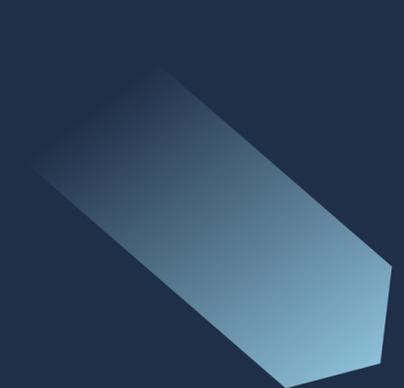


Prediction

Desired output



Target

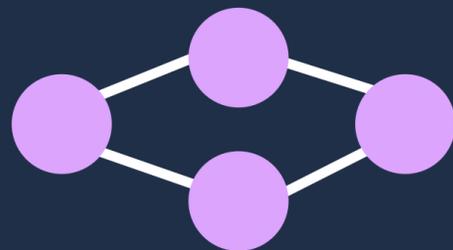


Input



Features

Function



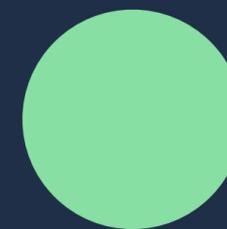
Network

Output



Prediction

Desired  
output



Target



Data



```
1 import * as tf from "@tensorflow/tfjs";
```

```
1 import * as tf from "@tensorflow/tfjs";  
2  
3 const generateData = (size) => {
```

```
22 };
```

```
1 import * as tf from "@tensorflow/tfjs";  
2  
3 const generateData = (size) => {  
4   const features = [];  
5   const targets = [];
```

```
22 };
```

```
1 import * as tf from "@tensorflow/tfjs";
2
3 const generateData = (size) => {
4   const features = [];
5   const targets = [];
6
7   for(let i = 0; i < size; i++) {
```

```
22   };
```

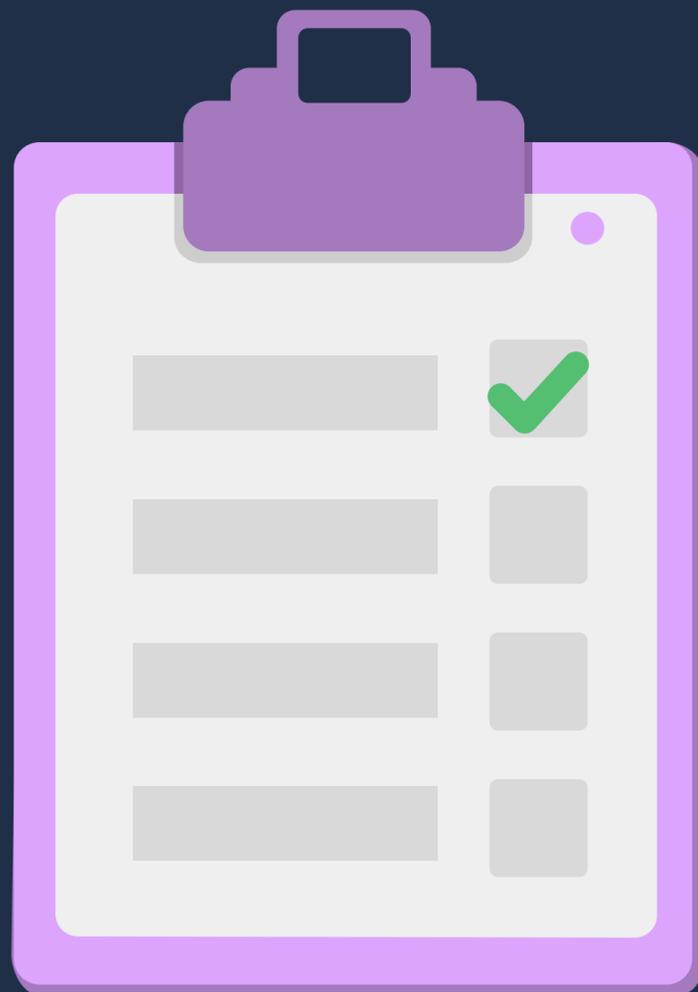
```
1 import * as tf from "@tensorflow/tfjs";
2
3 const generateData = (size) => {
4   const features = [];
5   const targets = [];
6
7   for(let i = 0; i < size; i++) {
8     const inputColor = colorHelper.randomColorArray(); // e.g. [0, 127, 255]
9     const targetColor = colorHelper.computeComplementaryColor(inputColor); // e.g. [255, 127, 0]
10
11
12
13
14
15
16
17
18
19
20
21
22   };
```

```
1 import * as tf from "@tensorflow/tfjs";
2
3 const generateData = (size) => {
4   const features = [];
5   const targets = [];
6
7   for(let i = 0; i < size; i++) {
8     const inputColor = colorHelper.randomColorArray(); // e.g. [0, 127, 255]
9     const targetColor = colorHelper.computeComplementaryColor(inputColor); // e.g. [255, 127, 0]
10
11     features.push(normalize(inputColor)); // [0, 0.5, 1]
12     targets.push(normalize(targetColor)); // [1, 0.5, 0]
13   }
14
15   return { features, targets };
16 }
17
18 // Example usage:
19 const { features, targets } = generateData(100);
20
21 // Convert to tensors
22 };
```

```
1 import * as tf from "@tensorflow/tfjs";
2
3 const generateData = (size) => {
4   const features = [];
5   const targets = [];
6
7   for(let i = 0; i < size; i++) {
8     const inputColor = colorHelper.randomColorArray(); // e.g. [0, 127, 255]
9     const targetColor = colorHelper.computeComplementaryColor(inputColor); // e.g. [255, 127, 0]
10
11     features.push(normalize(inputColor)); // [0, 0.5, 1]
12     targets.push(normalize(targetColor)); // [1, 0.5, 0]
13   }
14
15   const featureShape = [size, features[0].length] // 2-dimensional array
16   const targetShape = [size, targets[0].length] // 2-dimensional array
17
18   return {
19     features: tf.tensor(features, featureShape),
20     targets: tf.tensor(targets, targetShape)
21   };
22 };
```

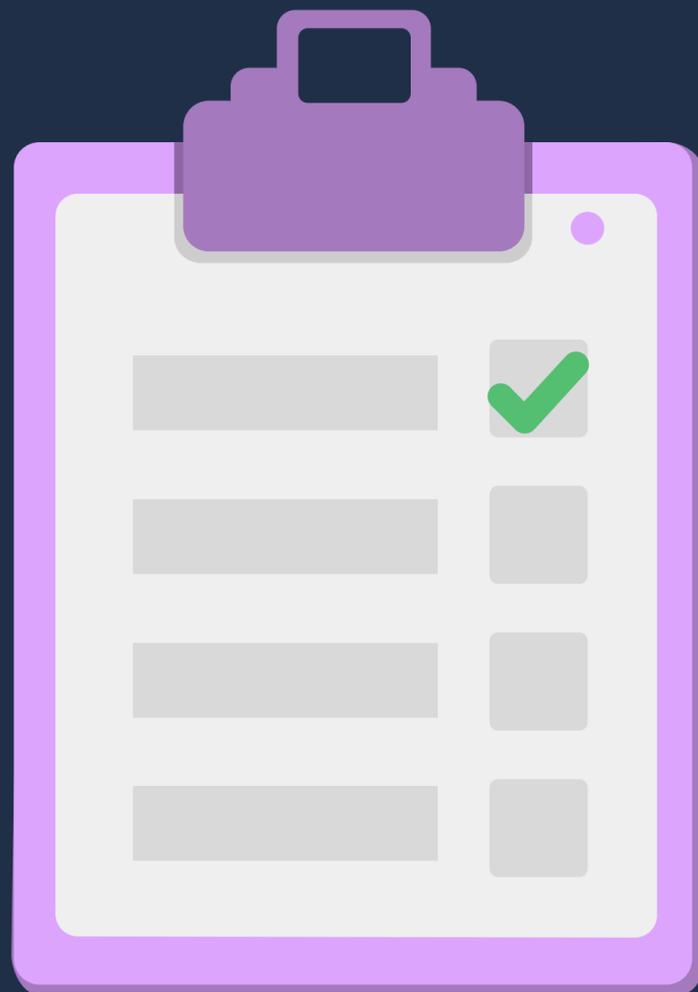
```
1 import * as tf from "@tensorflow/tfjs";
2
3 const generateData = (size) => {
4   const features = [];
5   const targets = [];
6
7   for(let i = 0; i < size; i++) {
8     const inputColor = colorHelper.randomColorArray(); // e.g. [0, 127, 255]
9     const targetColor = colorHelper.computeComplementaryColor(inputColor); // e.g. [255, 127, 0]
10
11     features.push(normalize(inputColor)); // [0, 0.5, 1]
12     targets.push(normalize(targetColor)); // [1, 0.5, 0]
13   }
14
15   const featureShape = [size, features[0].length] // 2-dimensional array
16   const targetShape = [size, targets[0].length] // 2-dimensional array
17
18   return {
19     features: tf.tensor2d(features, featureShape),
20     target: tf.tensor2d(targets, targetShape)
21   };
22 };
```

# CHECKLIST

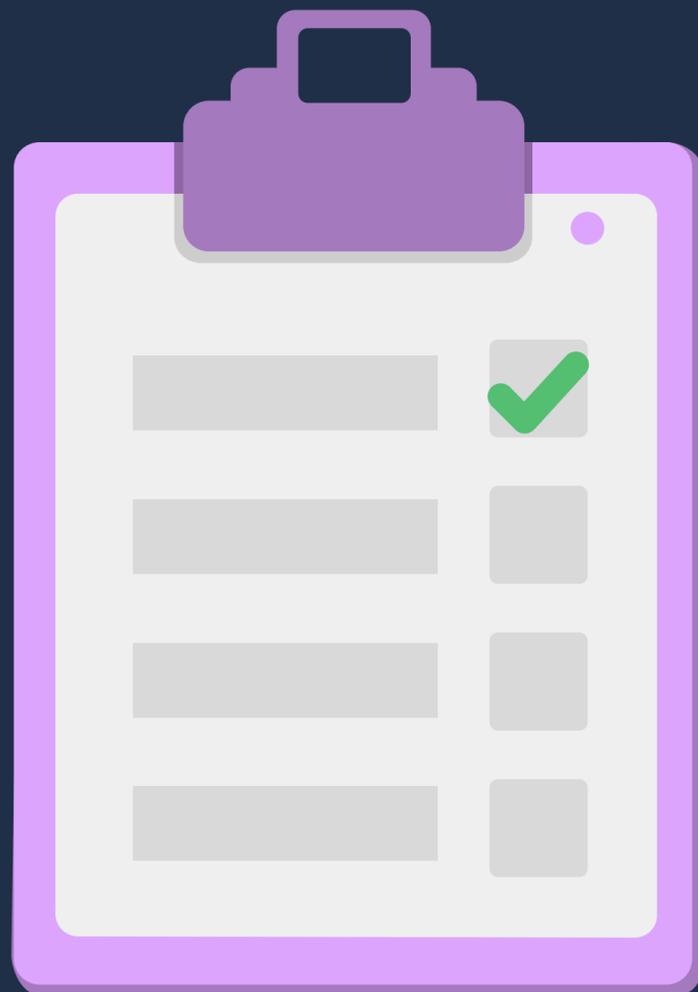


# CHECKLIST

✓ Data



# CHECKLIST



✓ Data

Network

# CHECKLIST

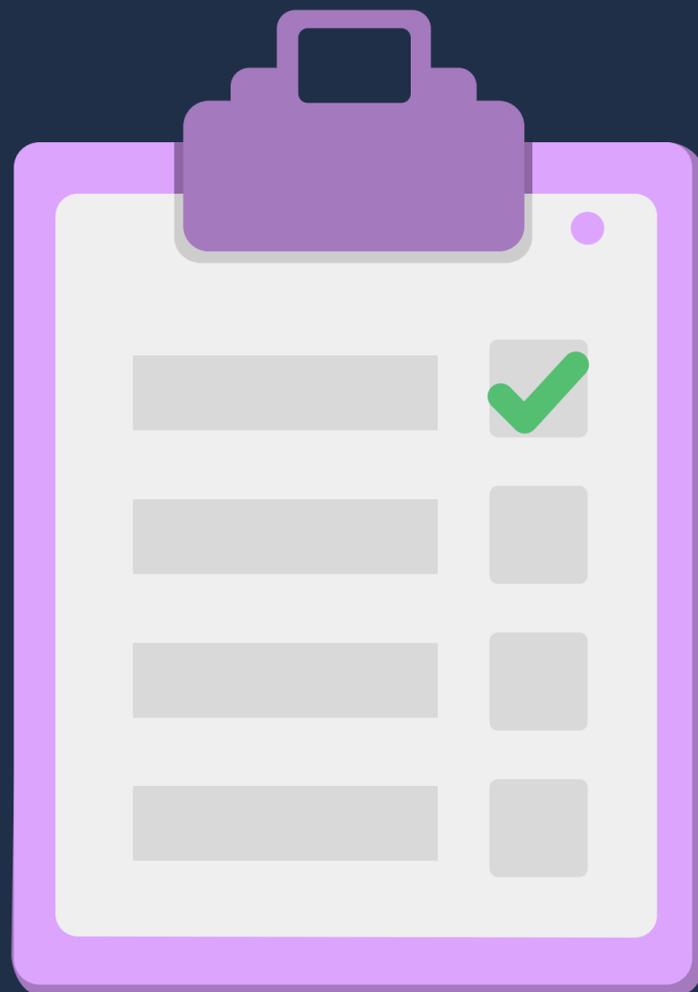


✓ Data

Network

???

# CHECKLIST



Data

Network

???

Profit!

# TENSORFLOW JS

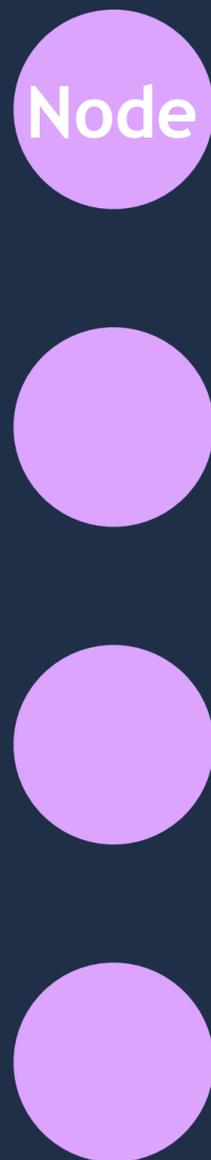
# NEURAL NETWORK

# NEURAL NETWORK



# NEURAL NETWORK

Node



# NEURAL NETWORK

Tensor

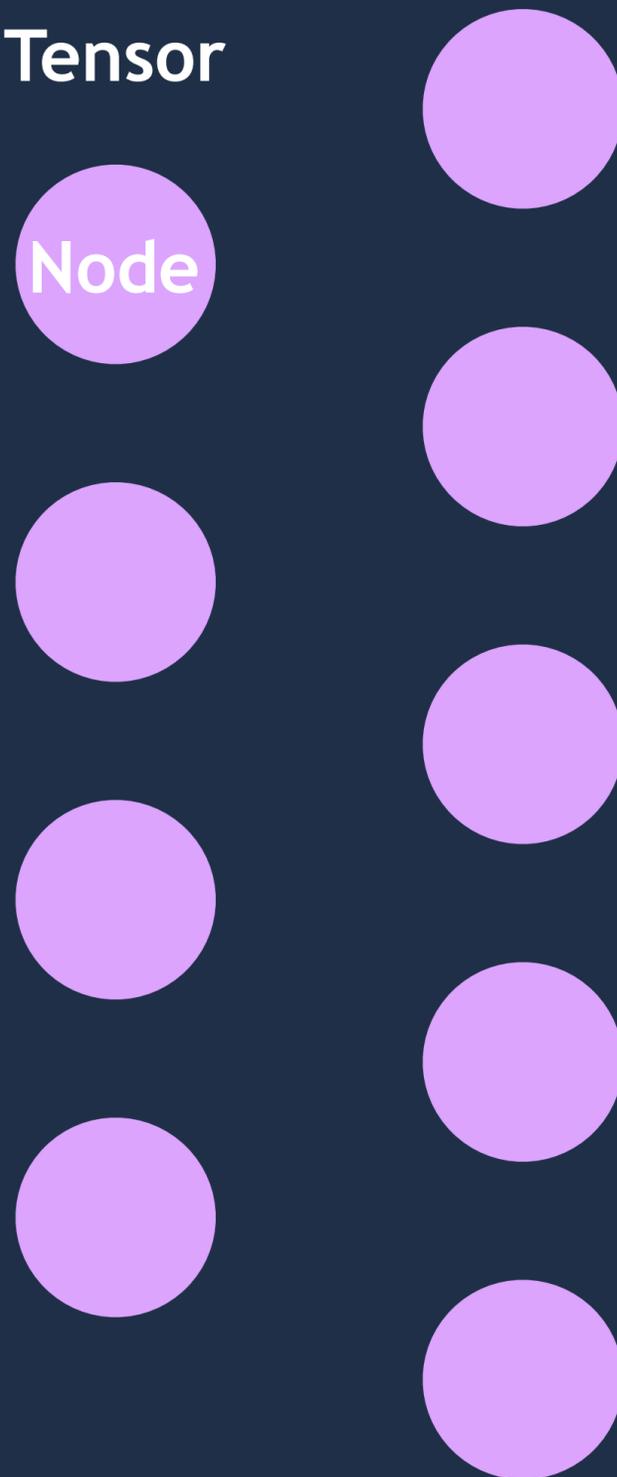
Node



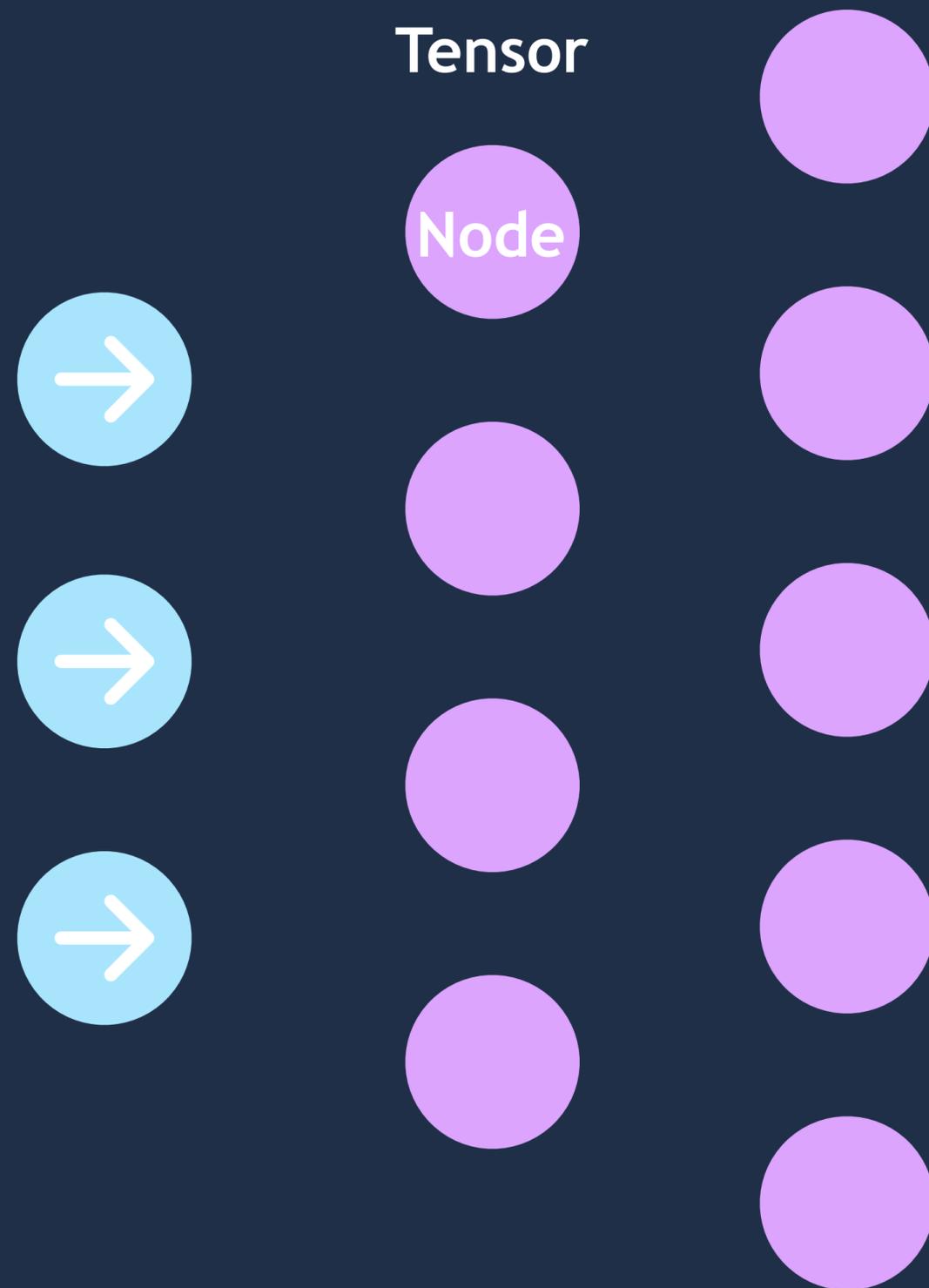
# NEURAL NETWORK

Tensor

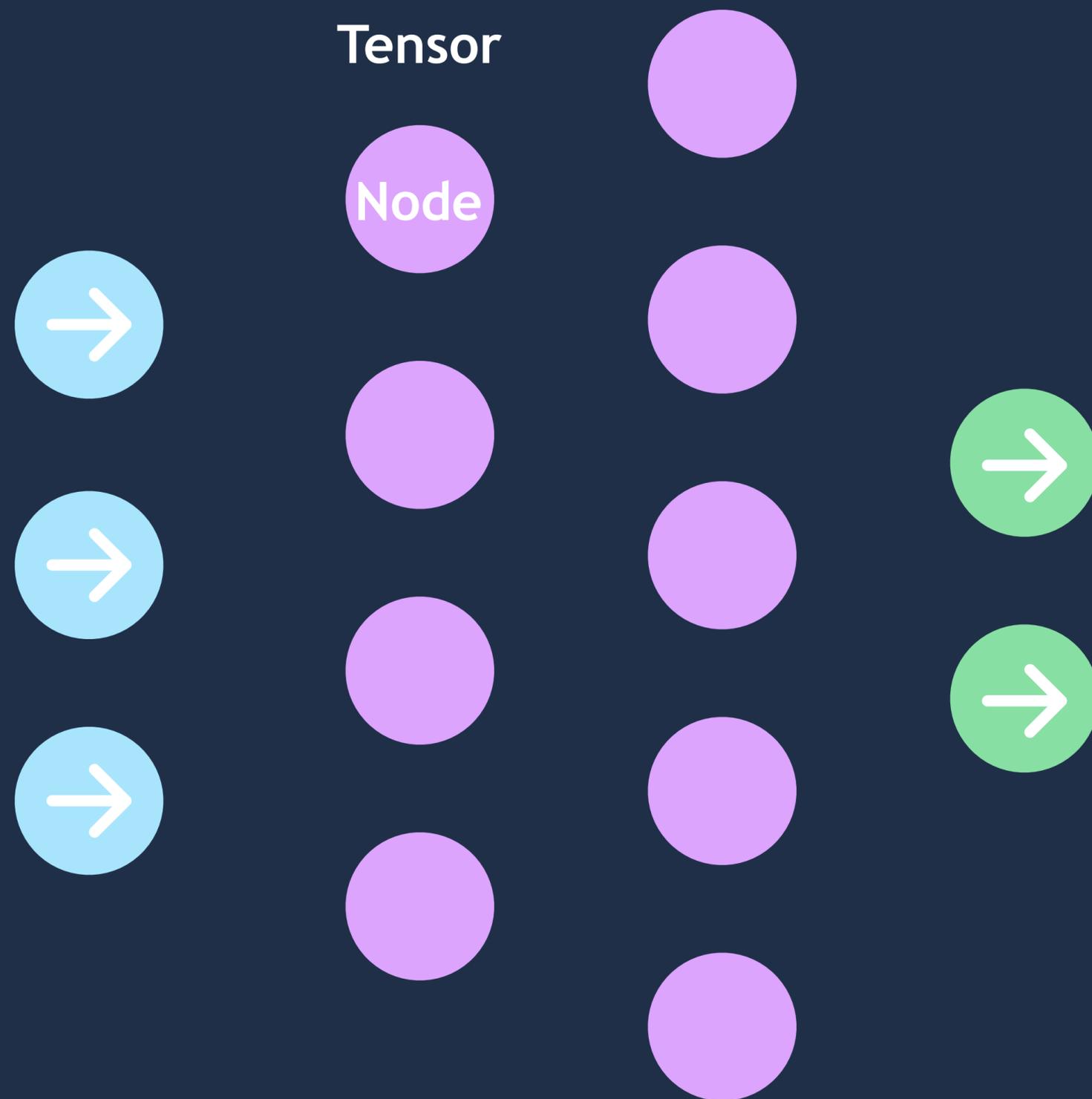
Node



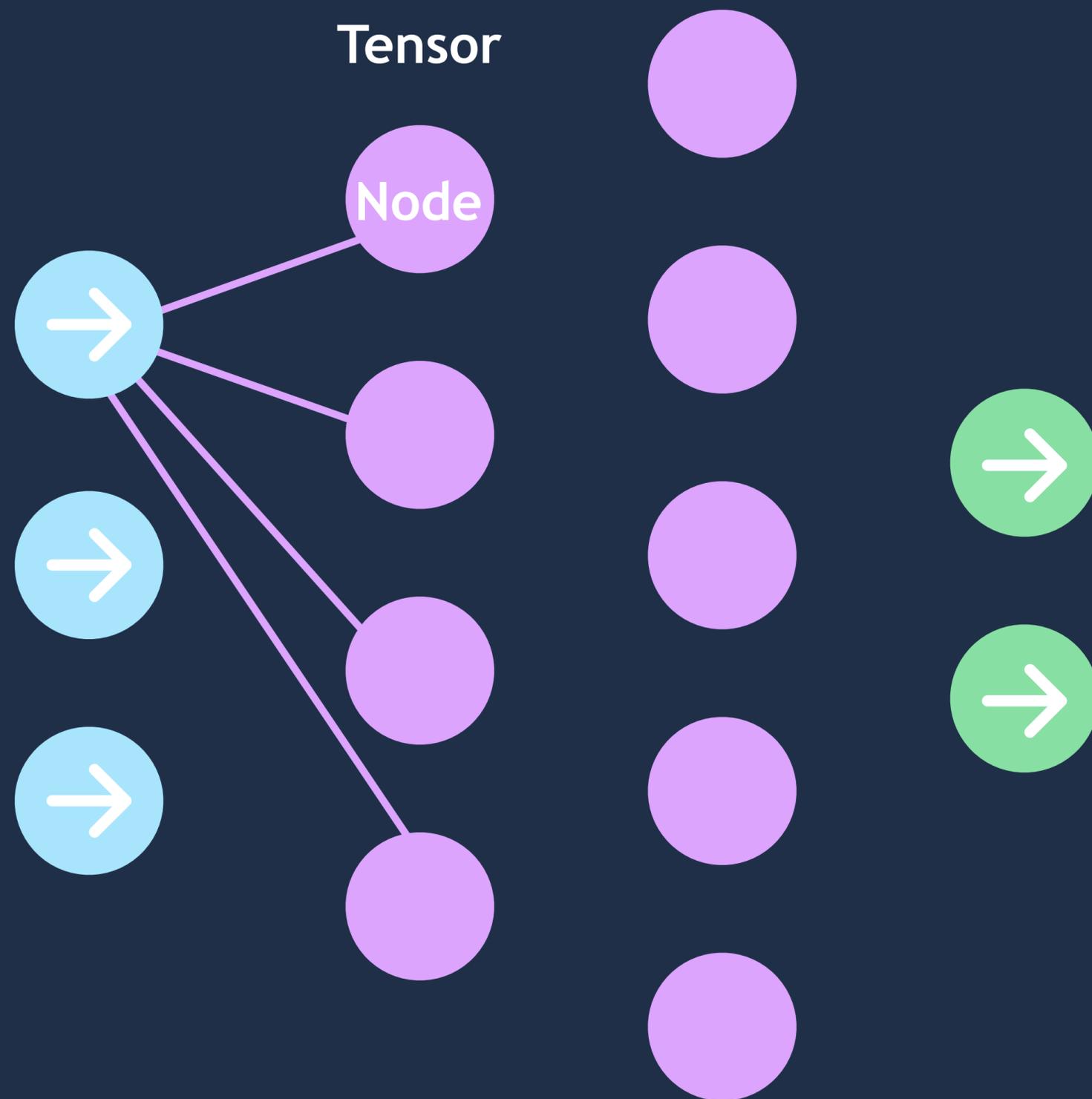
# NEURAL NETWORK



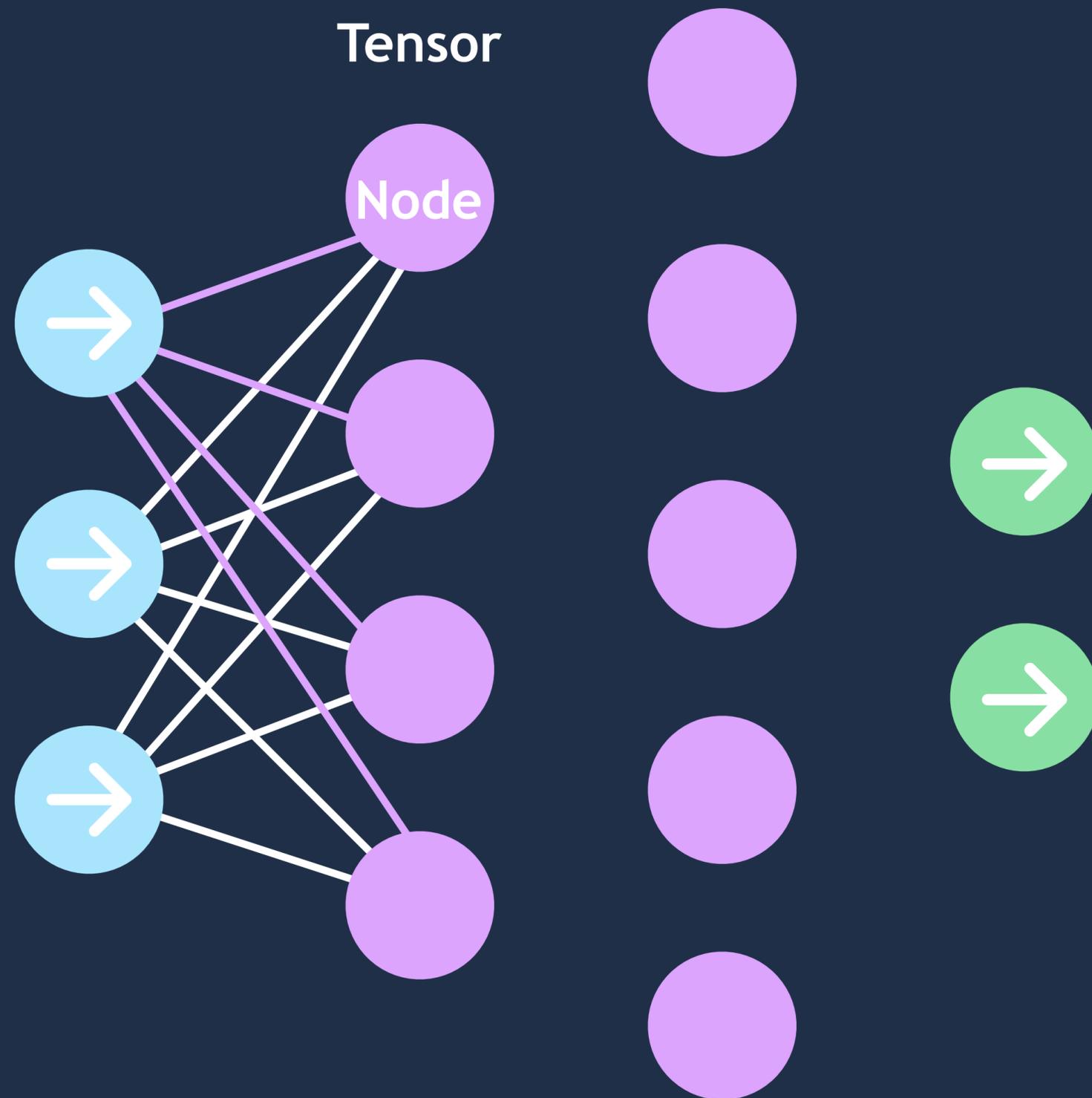
# NEURAL NETWORK



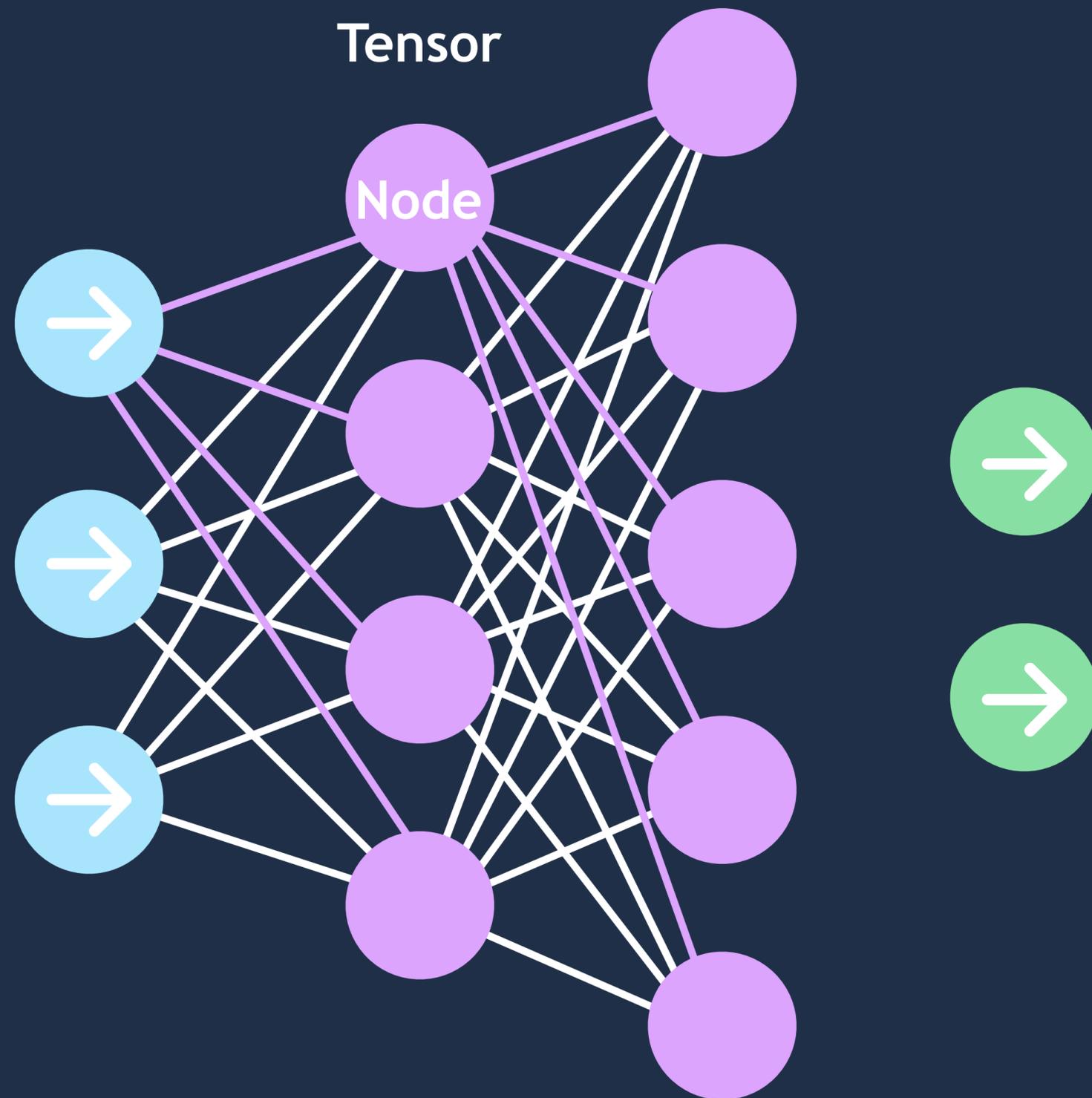
# NEURAL NETWORK



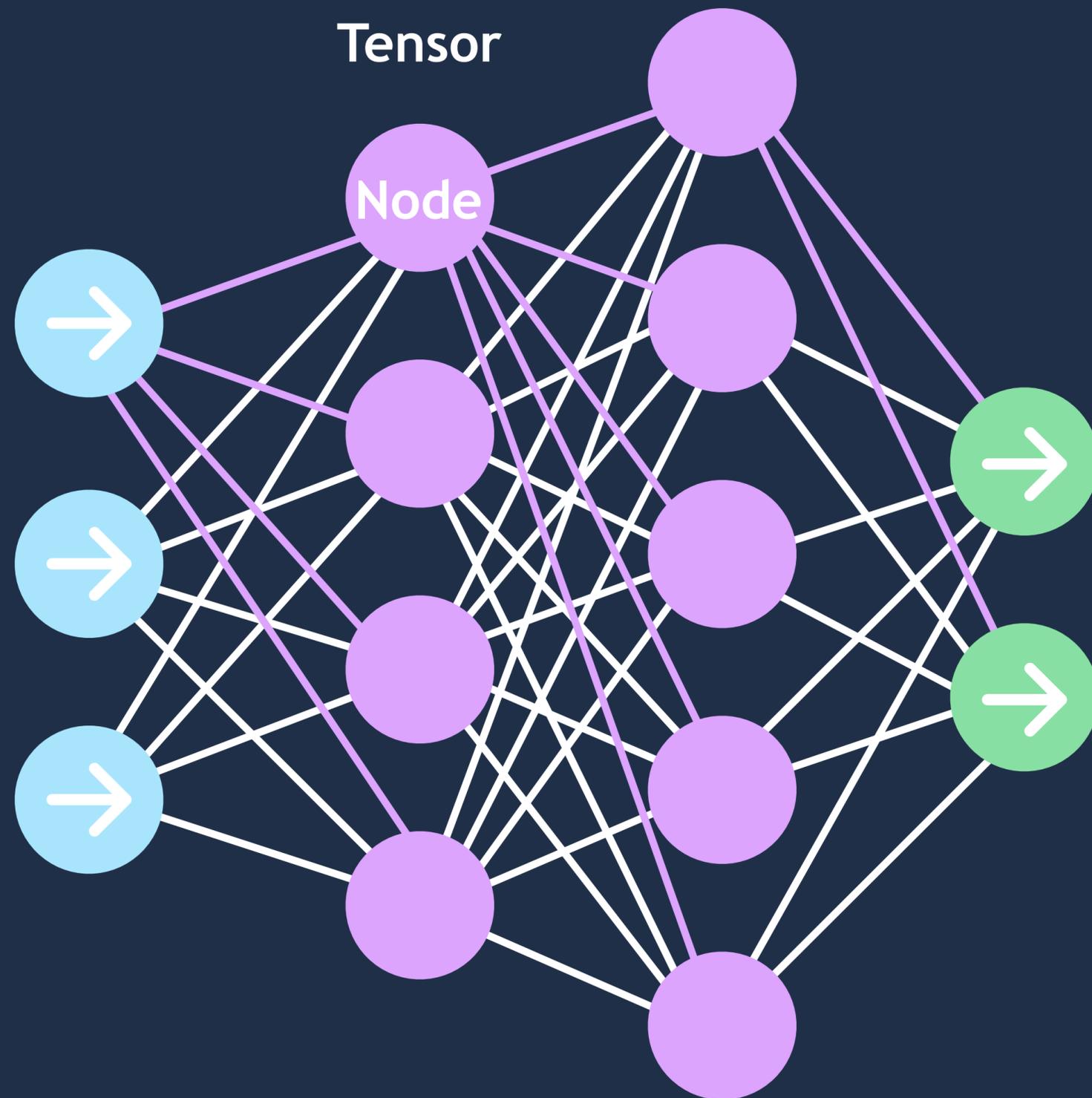
# NEURAL NETWORK



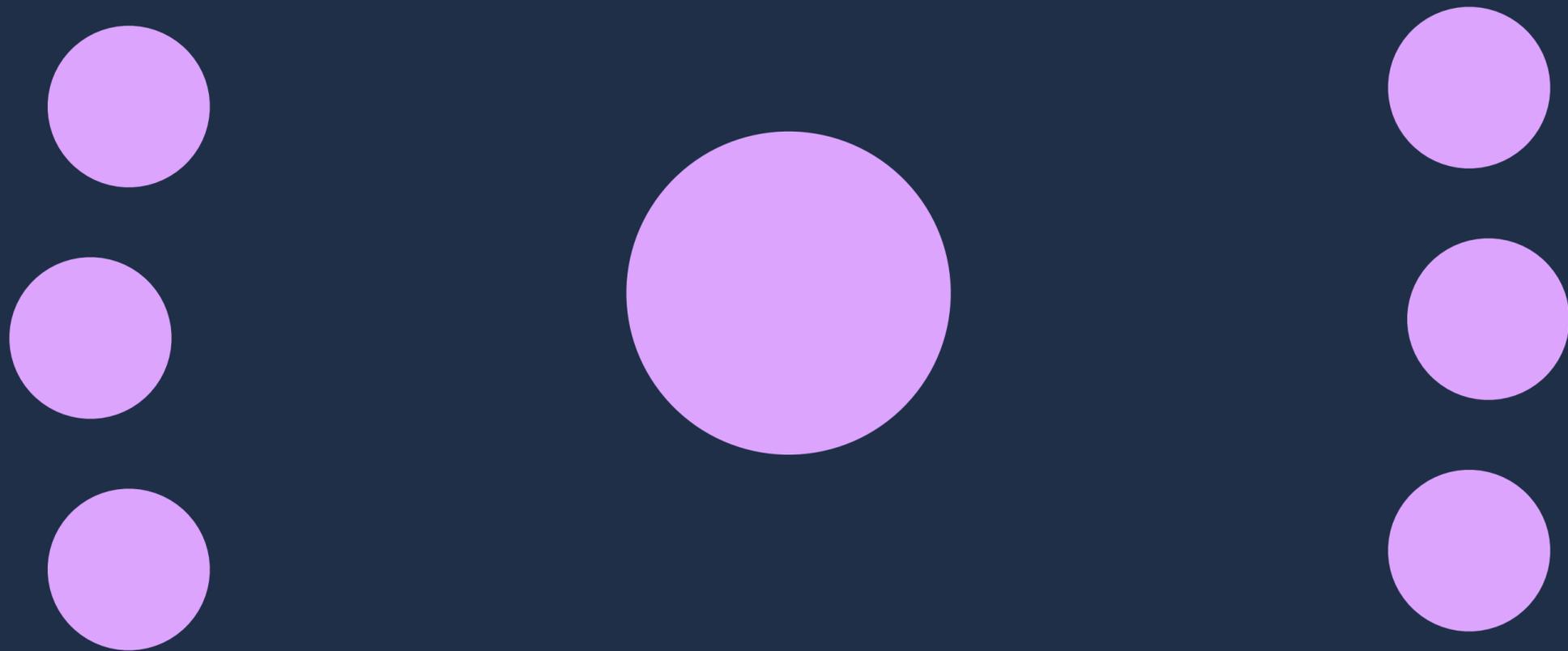
# NEURAL NETWORK



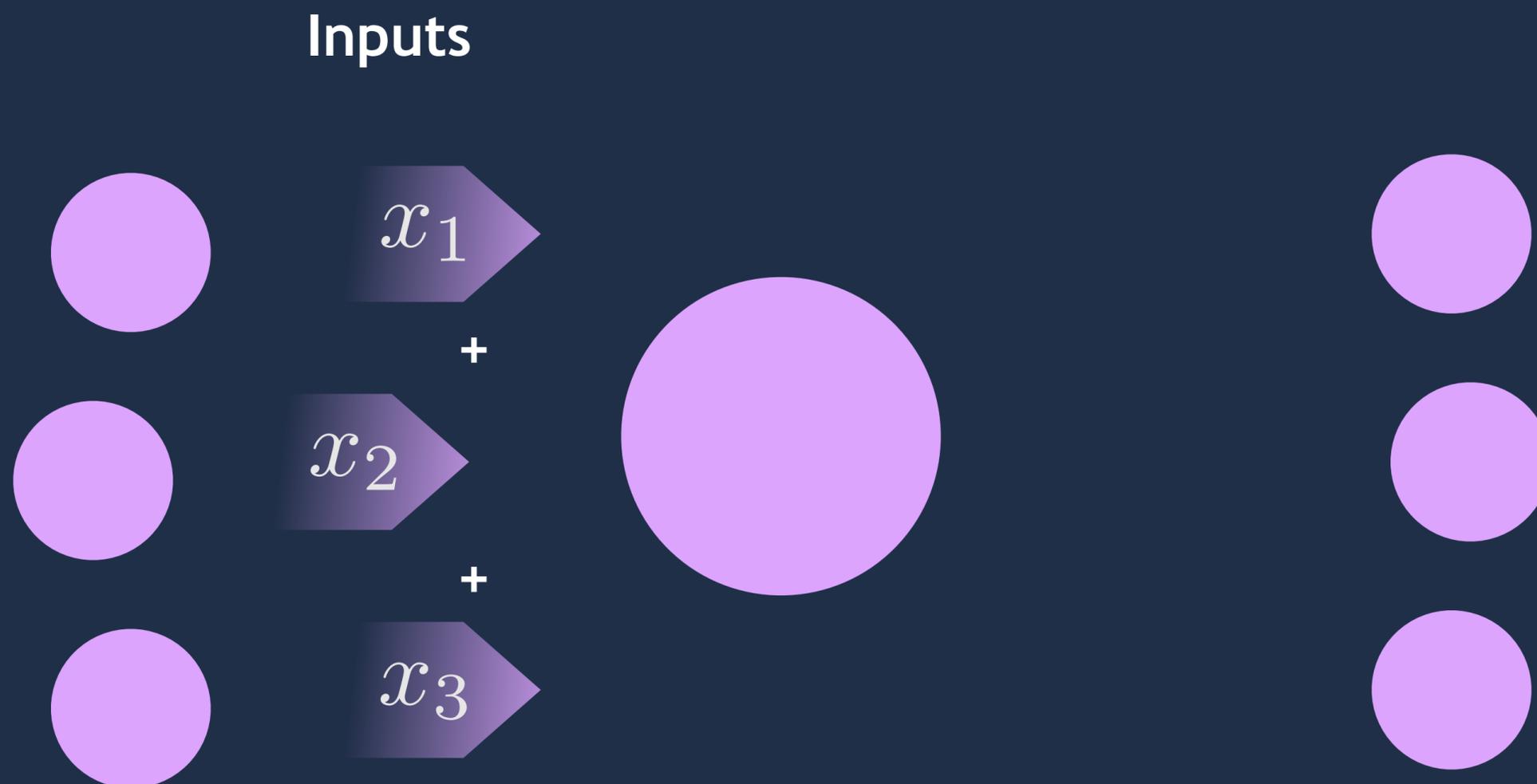
# NEURAL NETWORK



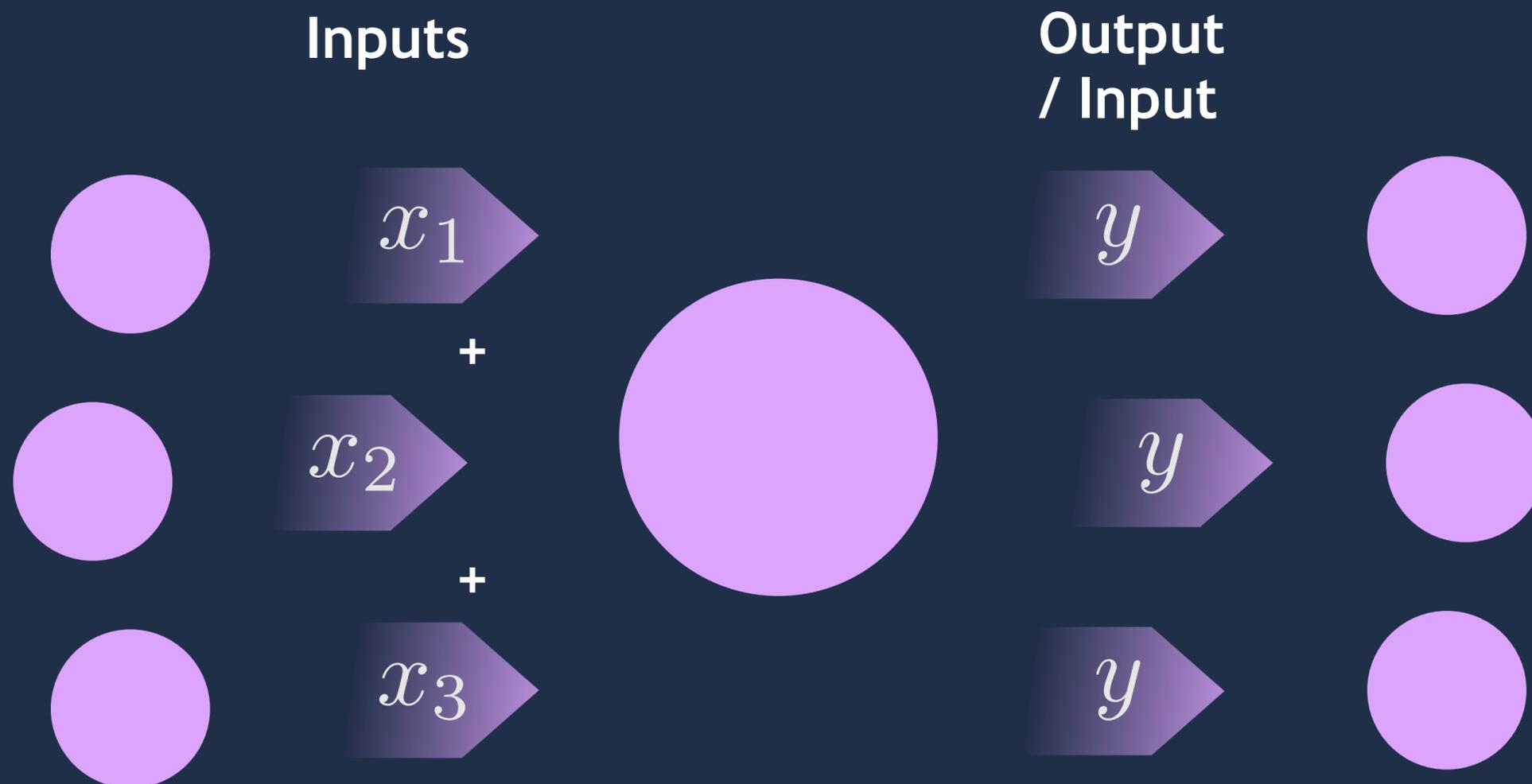
# NODE



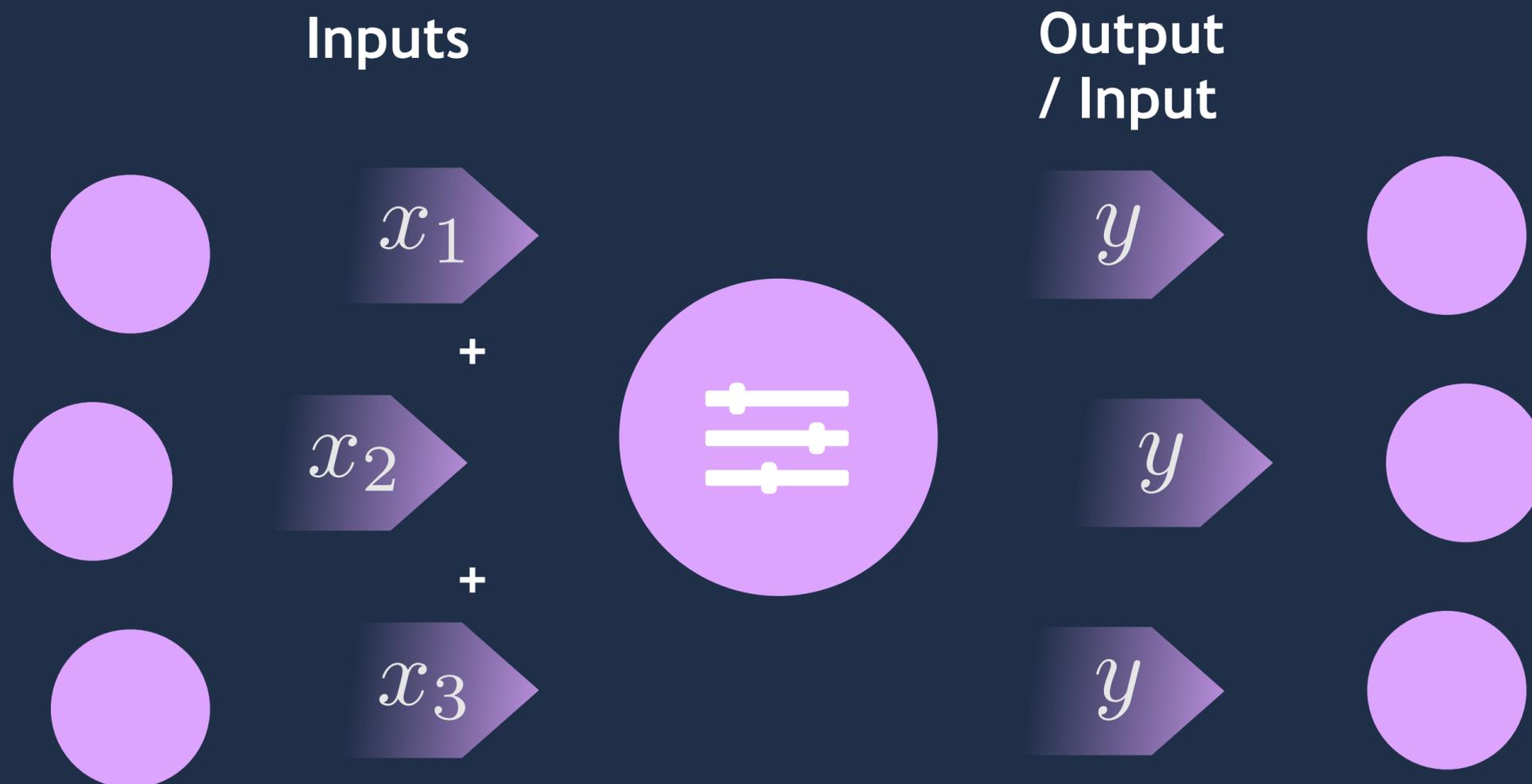
# NODE



# NODE



# NODE



# WEIGHTS

# WEIGHTS

Property of node

# WEIGHTS

Property of node

One for each input

# WEIGHTS

Property of node

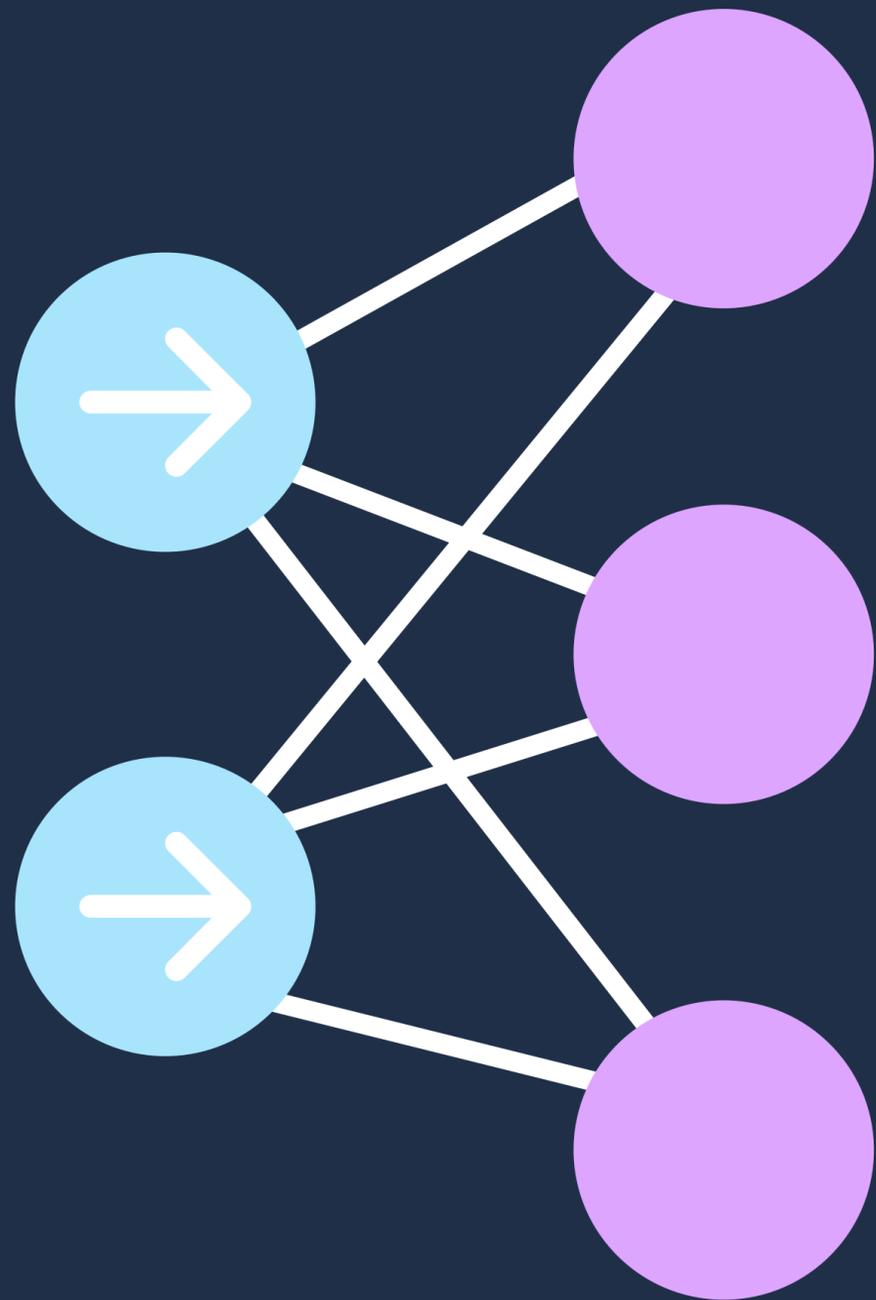
One for each input

Model

# DENSE LAYER

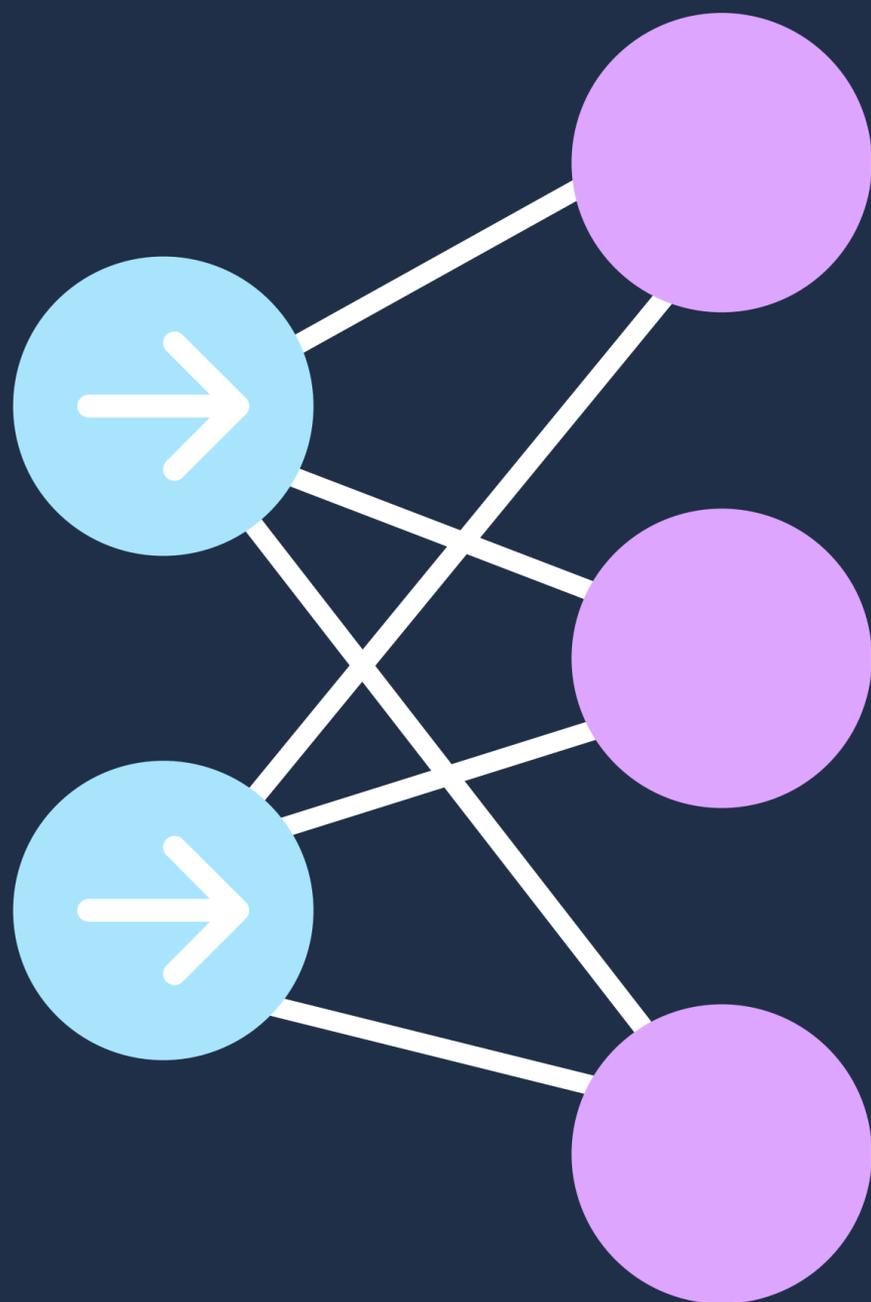


# DENSE LAYER

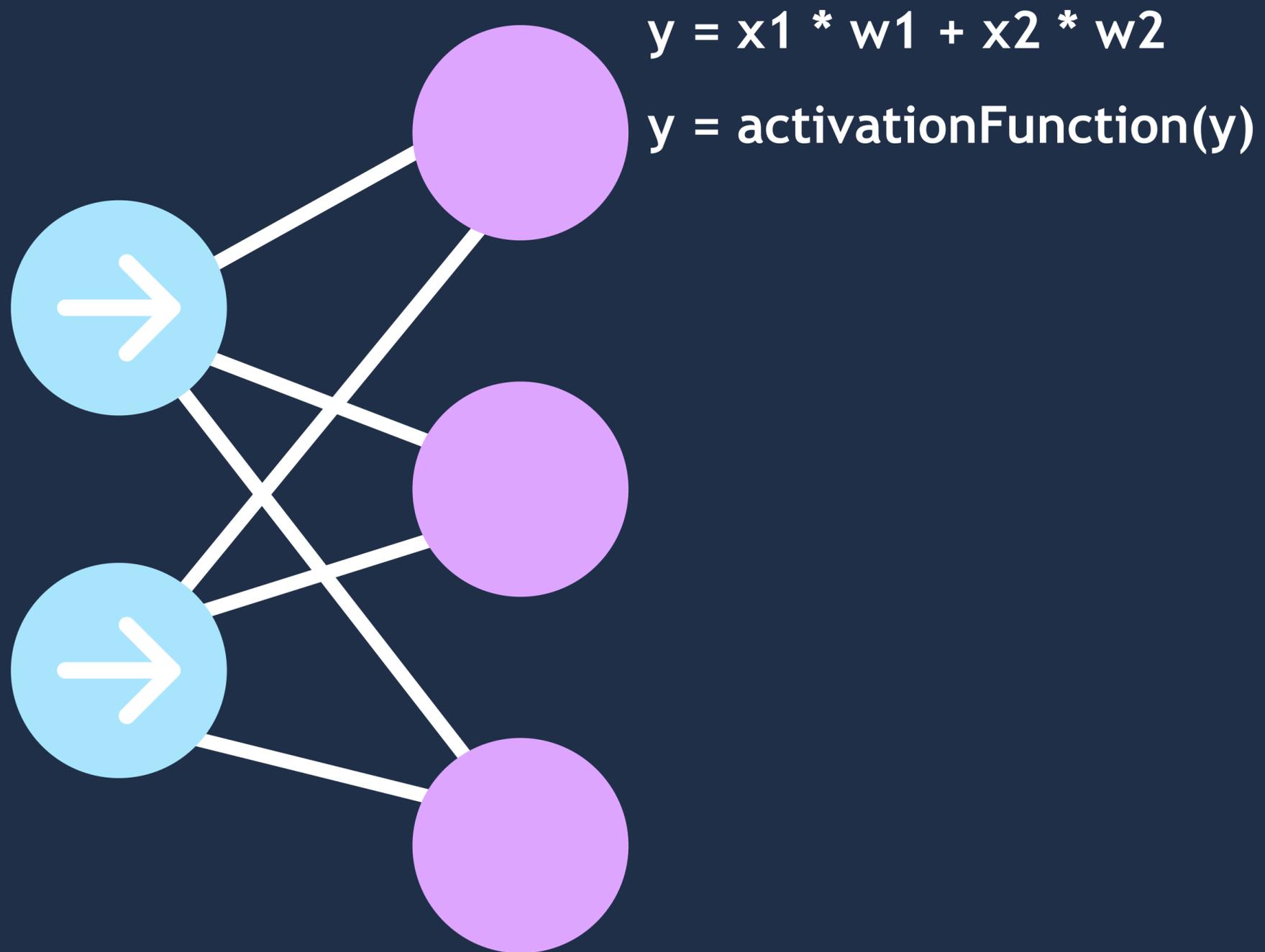


# DENSE LAYER

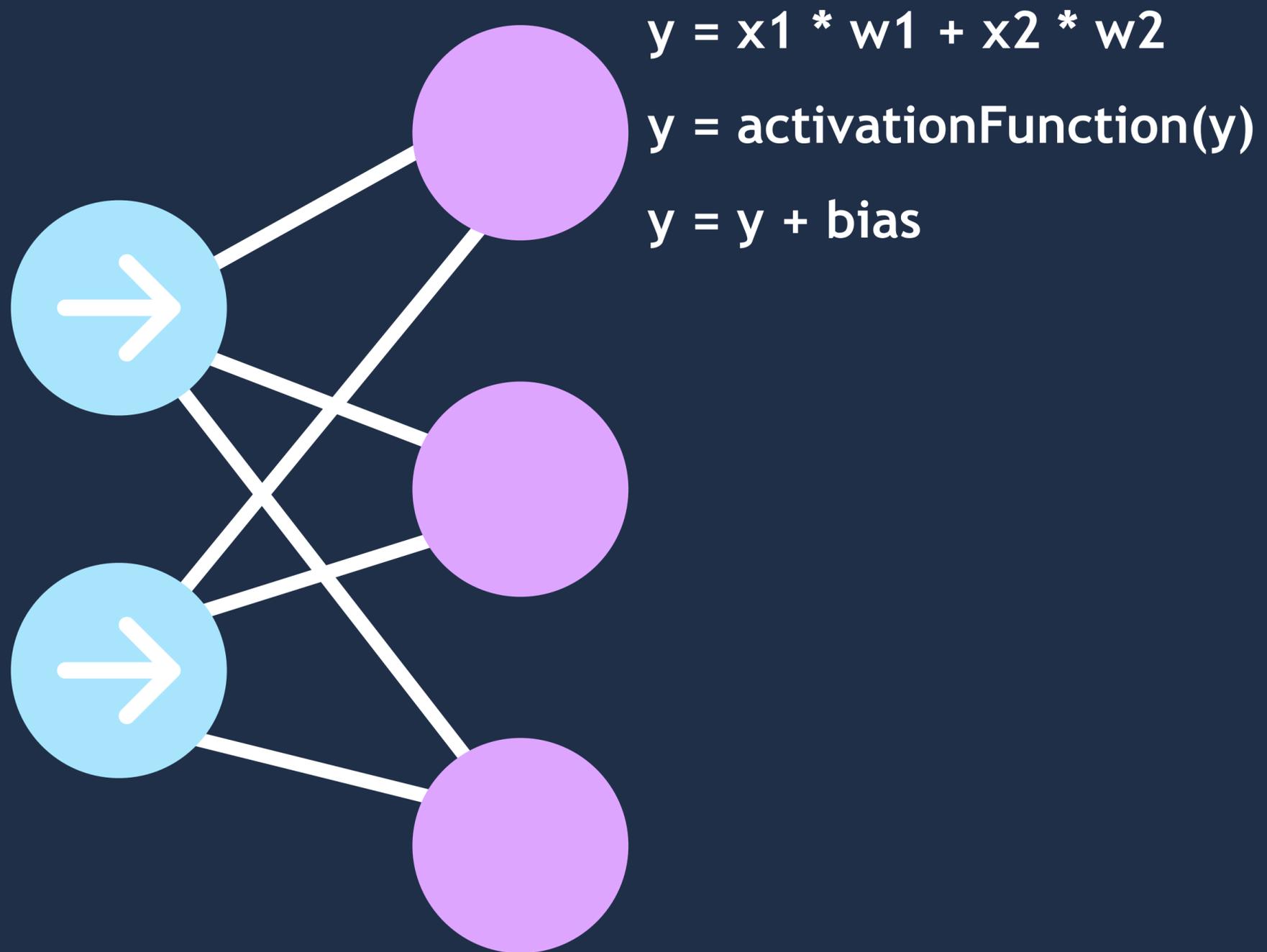
$$y = x1 * w1 + x2 * w2$$



# DENSE LAYER



# DENSE LAYER



```
1  const setupNetwork = () => {
```

```
1  import * as tf from '@tensorflow/tfjs';
```

```
21 }
```

```
1  const setupNetwork = () => {  
2    const model = createModel();
```

```
1  import * as tf from '@tensorflow/tfjs';
```

```
21 }
```

```
1  const setupNetwork = () => {  
2    const model = createModel();
```

```
1  import * as tf from '@tensorflow/tfjs';  
2  
3  const createModel = () => {  
4    return tf.sequential();  
5  }
```

```
21 }
```

```
1  const setupNetwork = () => {  
2    const model = createModel();  
3    addDenseLayer(model, {  
4      units: 3,  
5      inputDim: 3  
6    }); // input tensor
```

```
1  import * as tf from '@tensorflow/tfjs';  
2  
3  const createModel = () => {  
4    return tf.sequential();  
5  }
```

```
21 }
```

```
1  const setupNetwork = () => {
2    const model = createModel();
3    addDenseLayer(model, {
4      units: 3,
5      inputDim: 3
6    }); // input tensor
```

```
1  import * as tf from '@tensorflow/tfjs';
2
3  const createModel = () => {
4    return tf.sequential();
5  }
6
7  const addDenseLayer = (model, config) => {
8    model.add(tf.layers.dense(config));
9  }
10
```

```
21 }
```

```
1  const setupNetwork = () => {
2    const model = createModel();
3    addDenseLayer(model, {
4      units: 3,
5      inputDim: 3
6    }); // input tensor
7    addDenseLayer(model,
8      {units: 64, useBias: true}
9    );
10   addDenseLayer(model,
11     {units: 32, useBias: true}
12   );
13   addDenseLayer(model,
14     {units: 16, useBias: true}
15   );
```

```
21 }
```

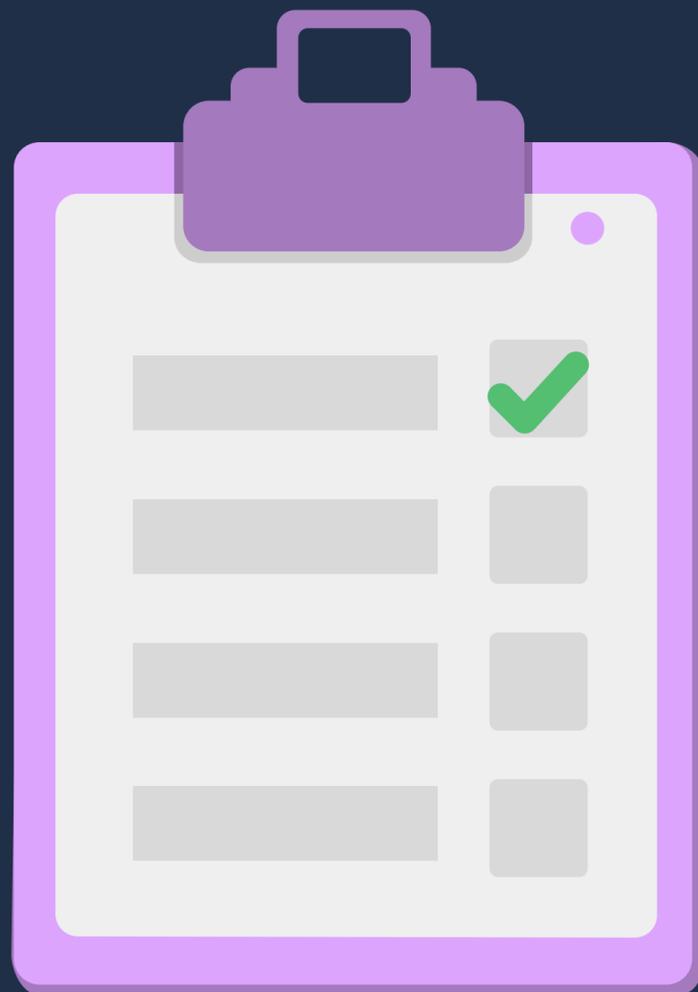
```
1  import * as tf from '@tensorflow/tfjs';
2
3  const createModel = () => {
4    return tf.sequential();
5  }
6
7  const addDenseLayer = (model, config) => {
8    model.add(tf.layers.dense(config));
9  }
10
```

```
1  const setupNetwork = () => {
2    const model = createModel();
3    addDenseLayer(model, {
4      units: 3,
5      inputDim: 3
6    }); // input tensor
7    addDenseLayer(model,
8      {units: 64, useBias: true}
9    );
10   addDenseLayer(model,
11     {units: 32, useBias: true}
12   );
13   addDenseLayer(model,
14     {units: 16, useBias: true}
15   );
16   addDenseLayer(model, {
17     units: 3,
18     activation: "relu",
19     useBias: true
20   }); //output layer
21 }
```

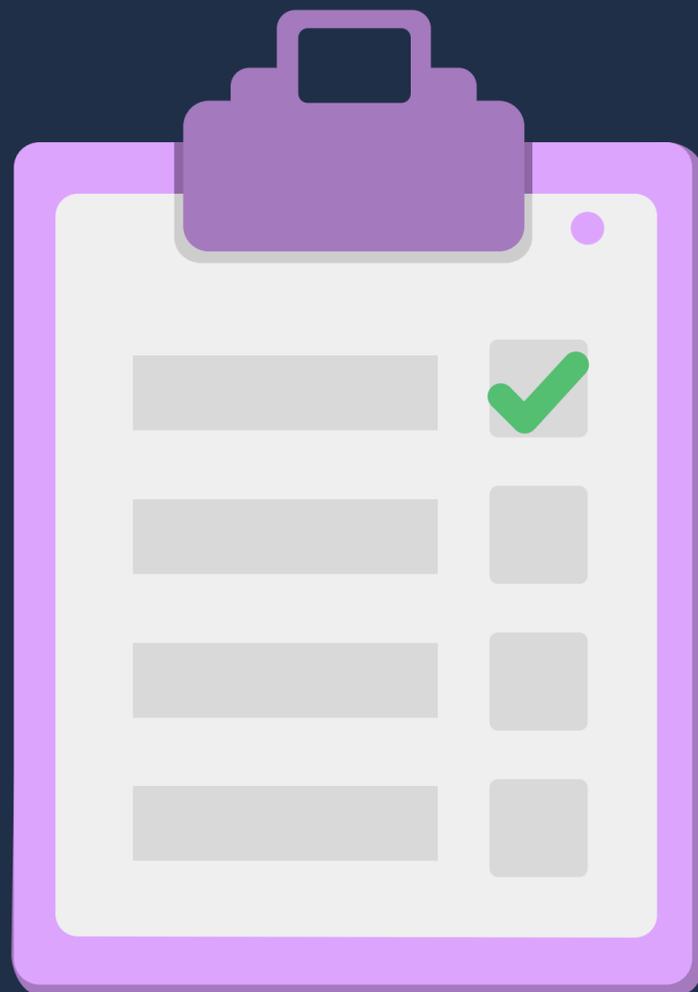
```
1  import * as tf from '@tensorflow/tfjs';
2
3  const createModel = () => {
4    return tf.sequential();
5  }
6
7  const addDenseLayer = (model, config) => {
8    model.add(tf.layers.dense(config));
9  }
10
```

# CHECKLIST

✓ Data



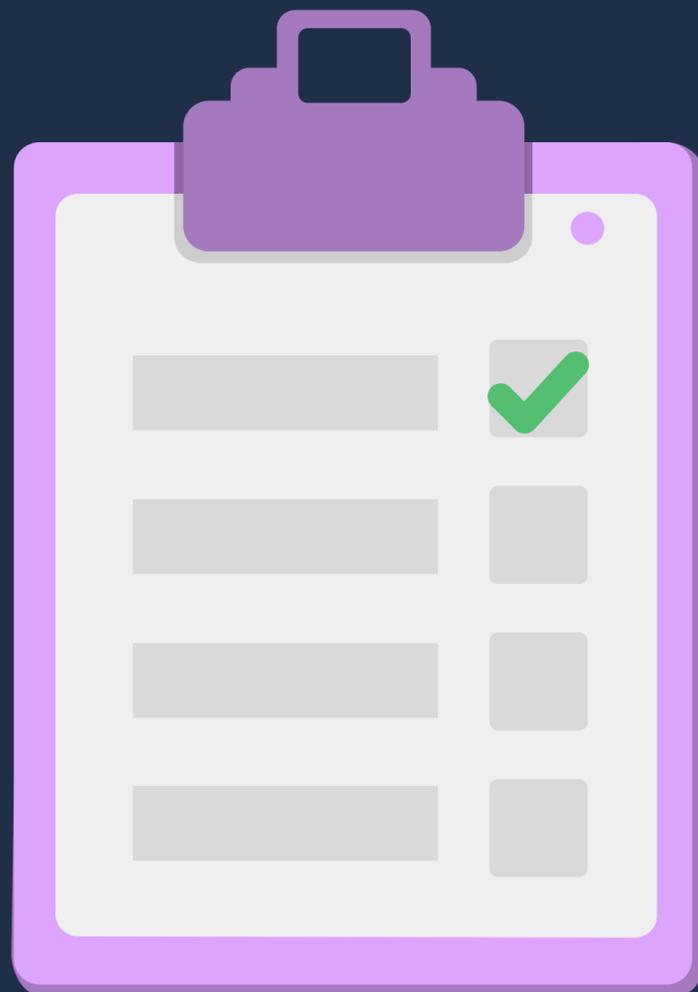
# CHECKLIST



✓ Data

✓ Network

# CHECKLIST

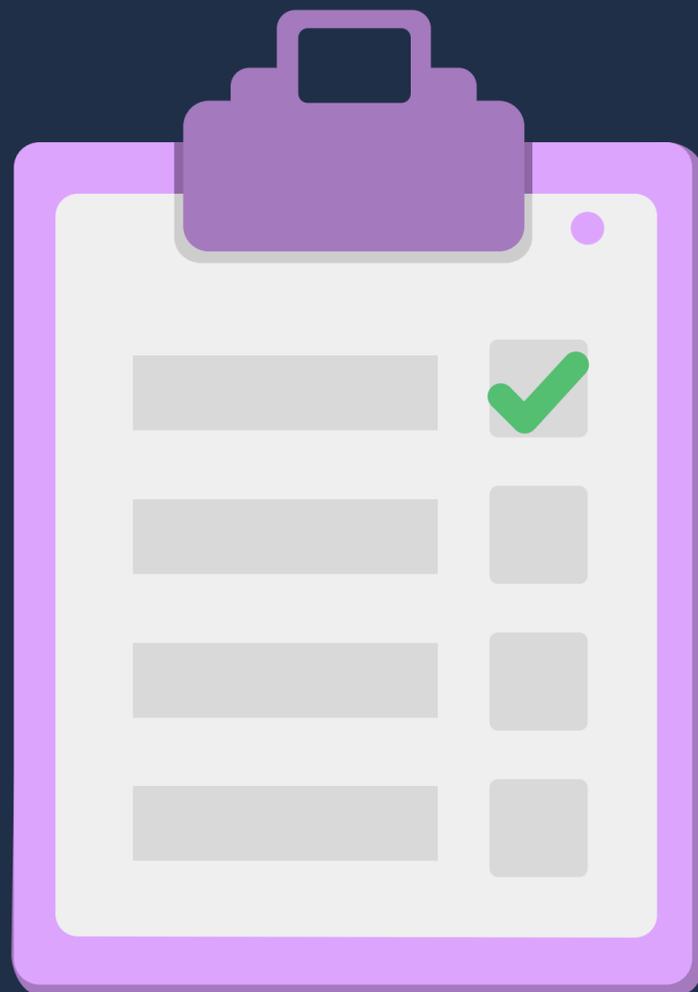


✓ Data

✓ Network

???

# CHECKLIST



✓ Data

✓ Network

???

Profit!

# TRAINING

# TRAINING

# TRAINING

Find metric

# TRAINING

Find metric

Change something

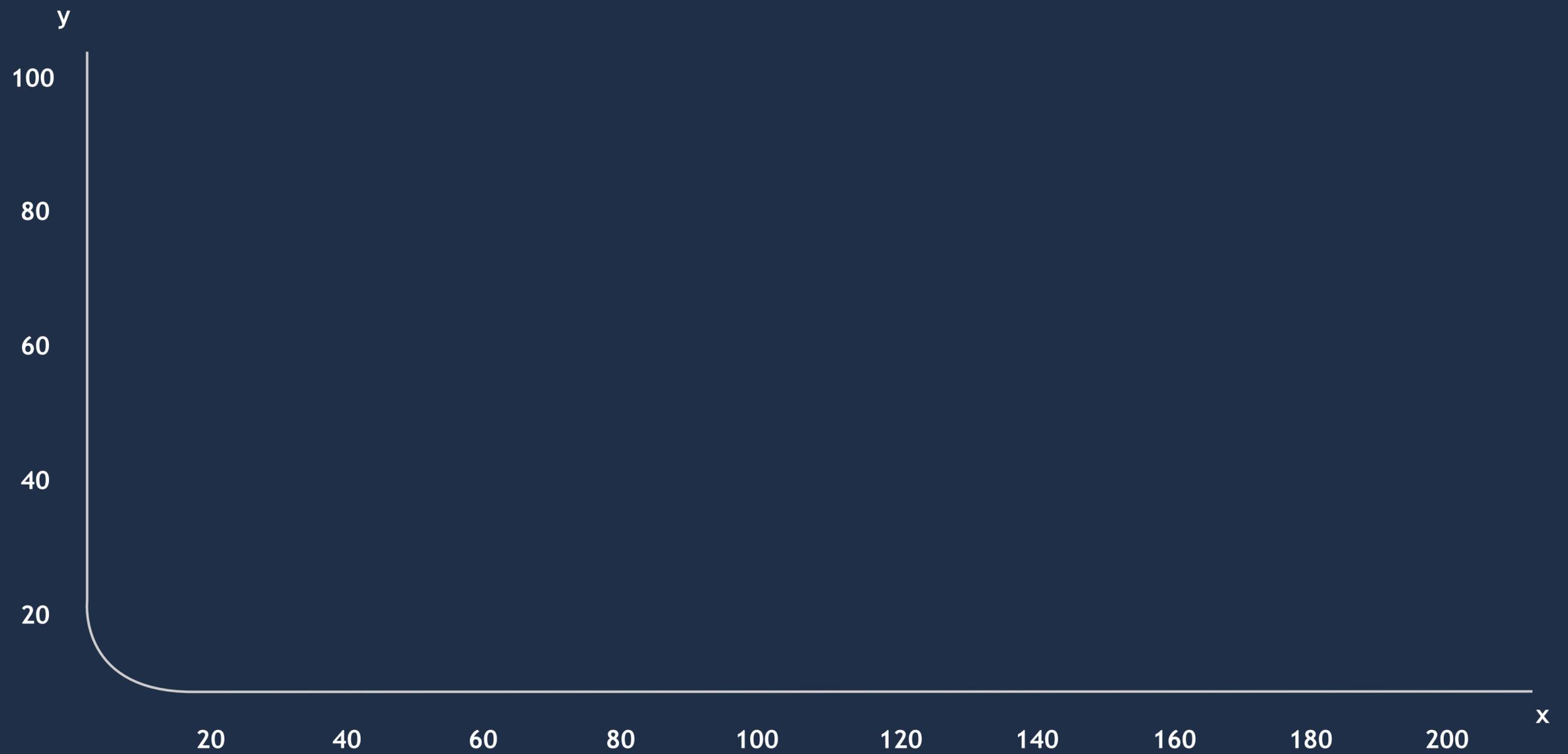
# TRAINING

Find metric

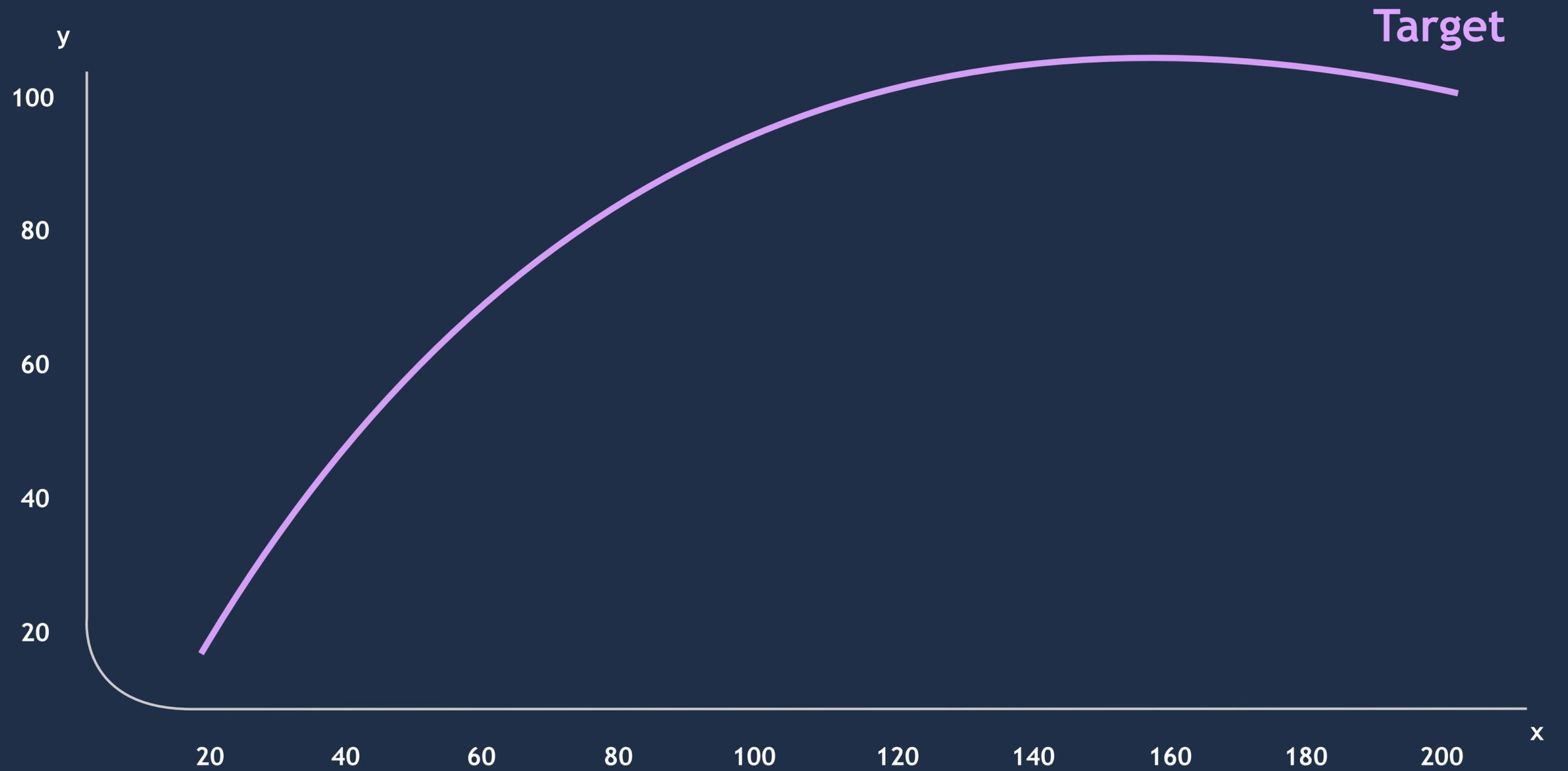
Change something

Check metric

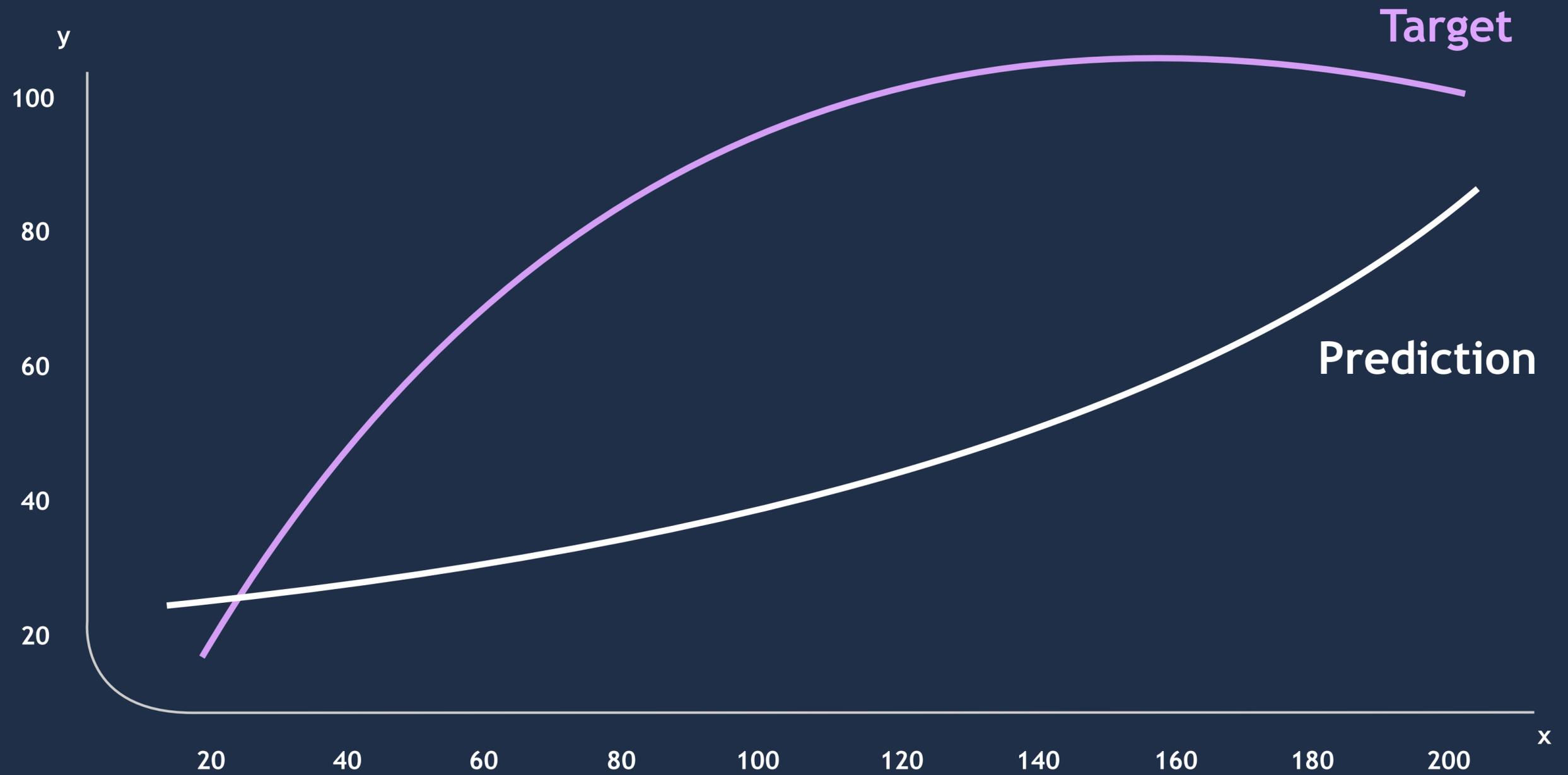
# COST/LOSS



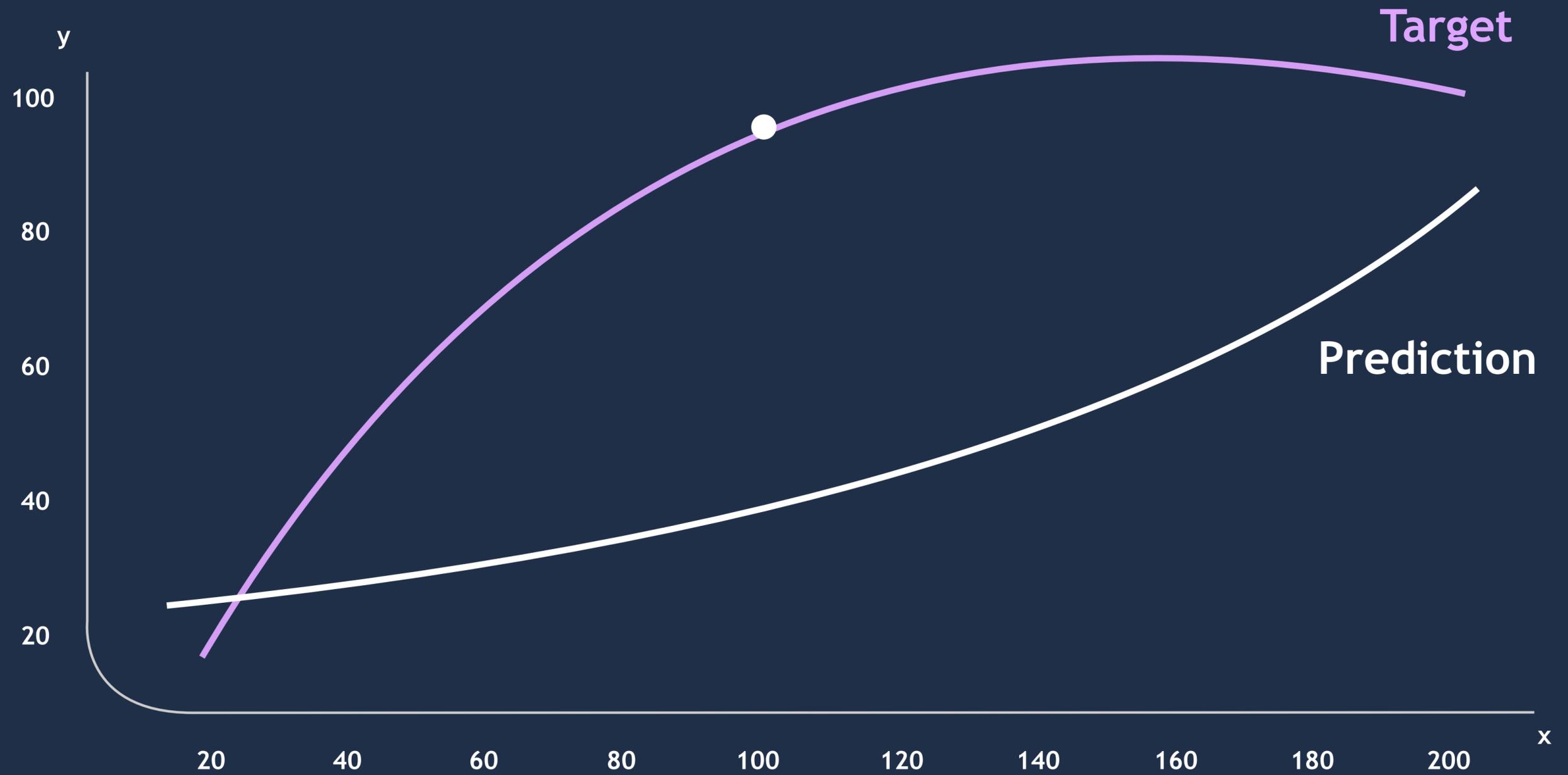
# COST/LOSS



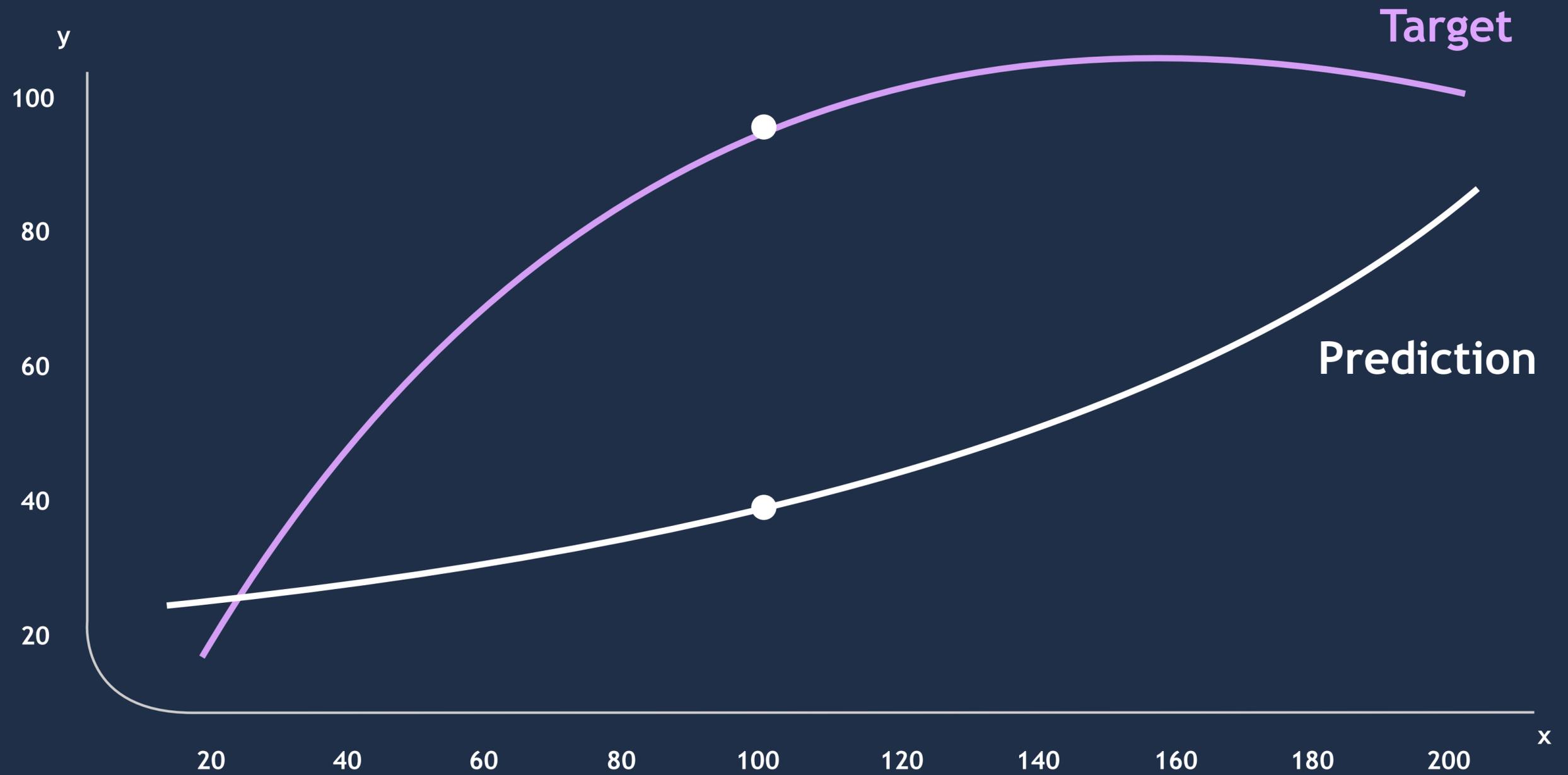
# COST/LOSS



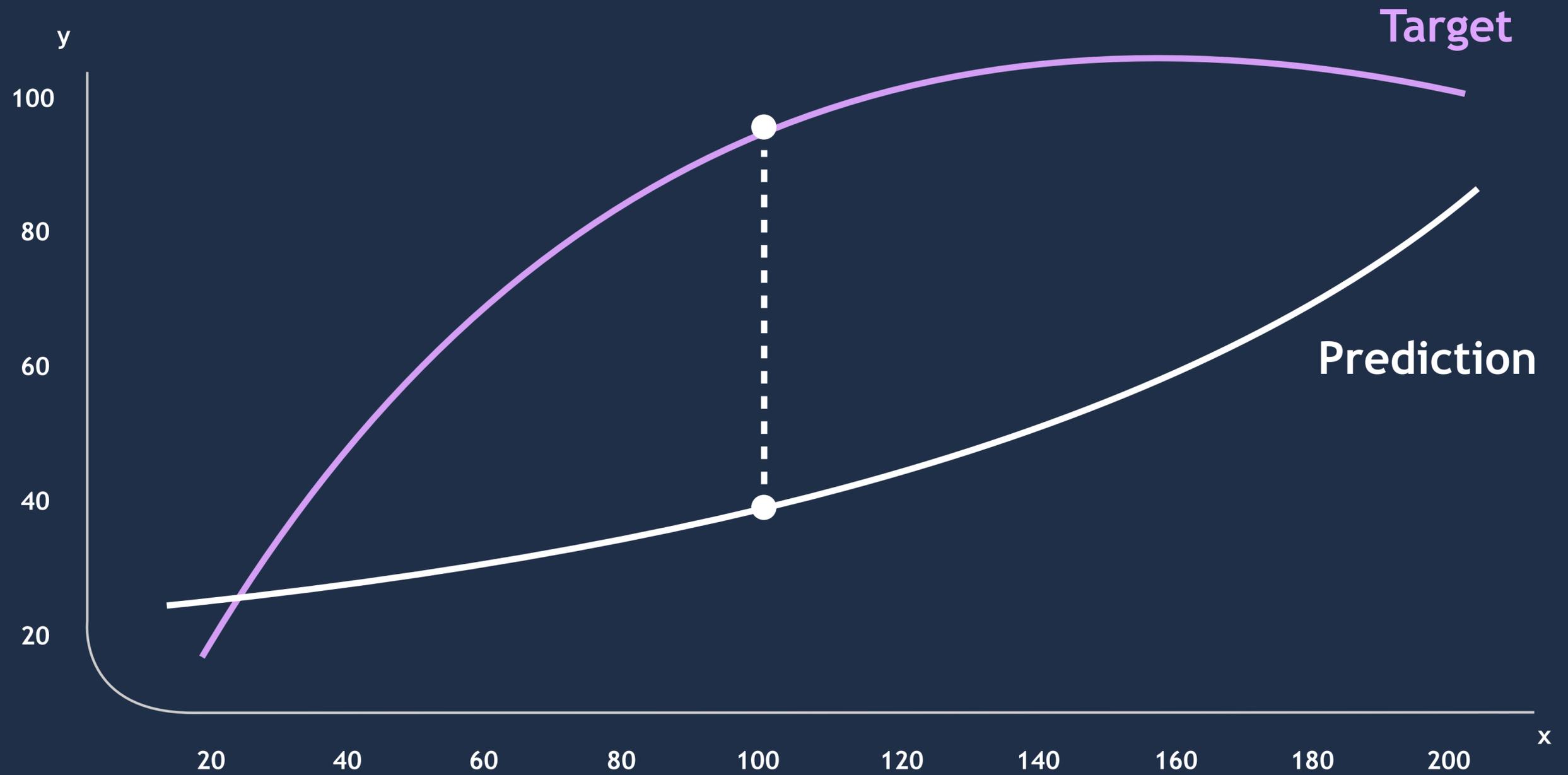
# COST/LOSS



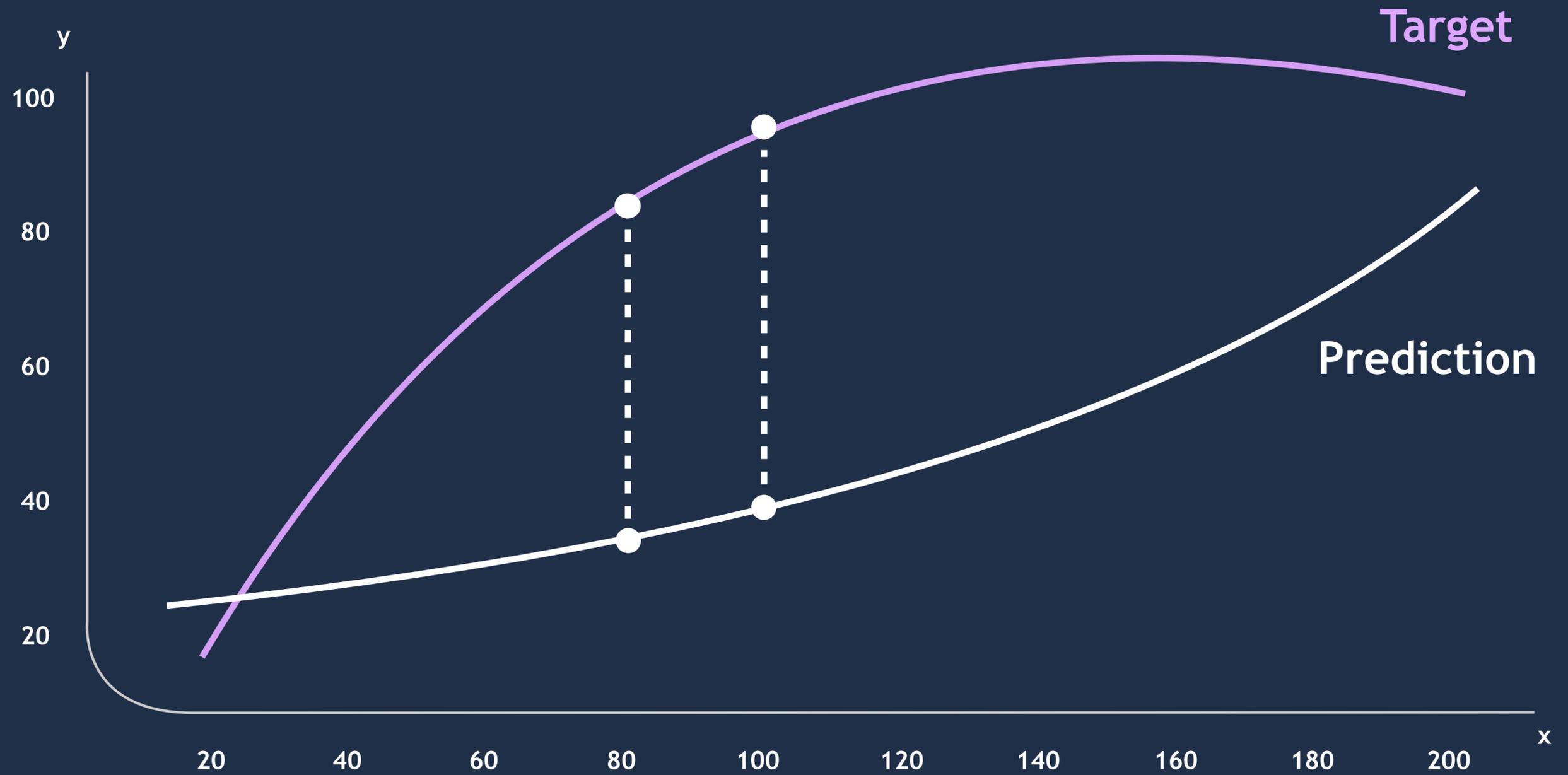
# COST/LOSS



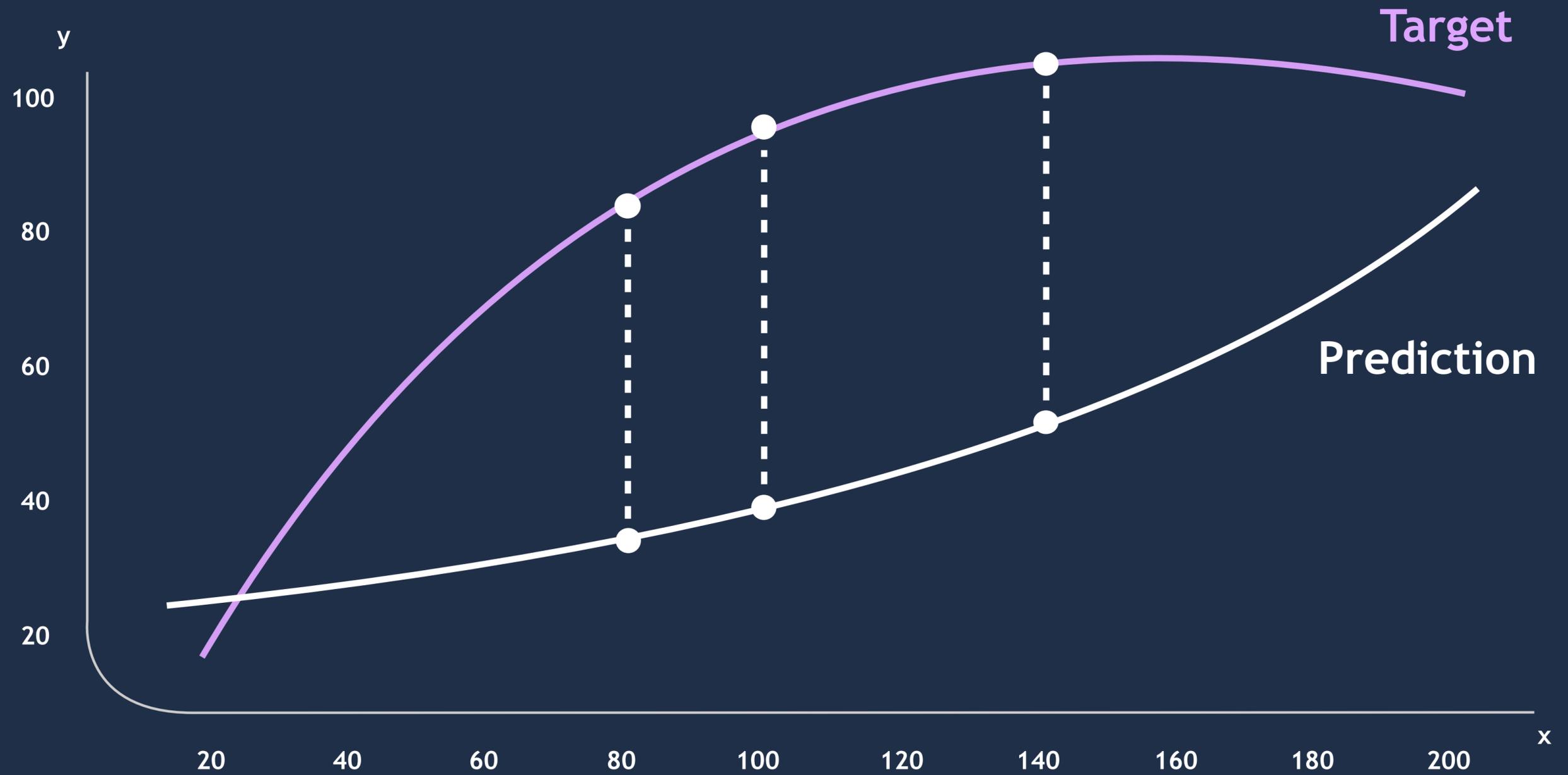
# COST/LOSS



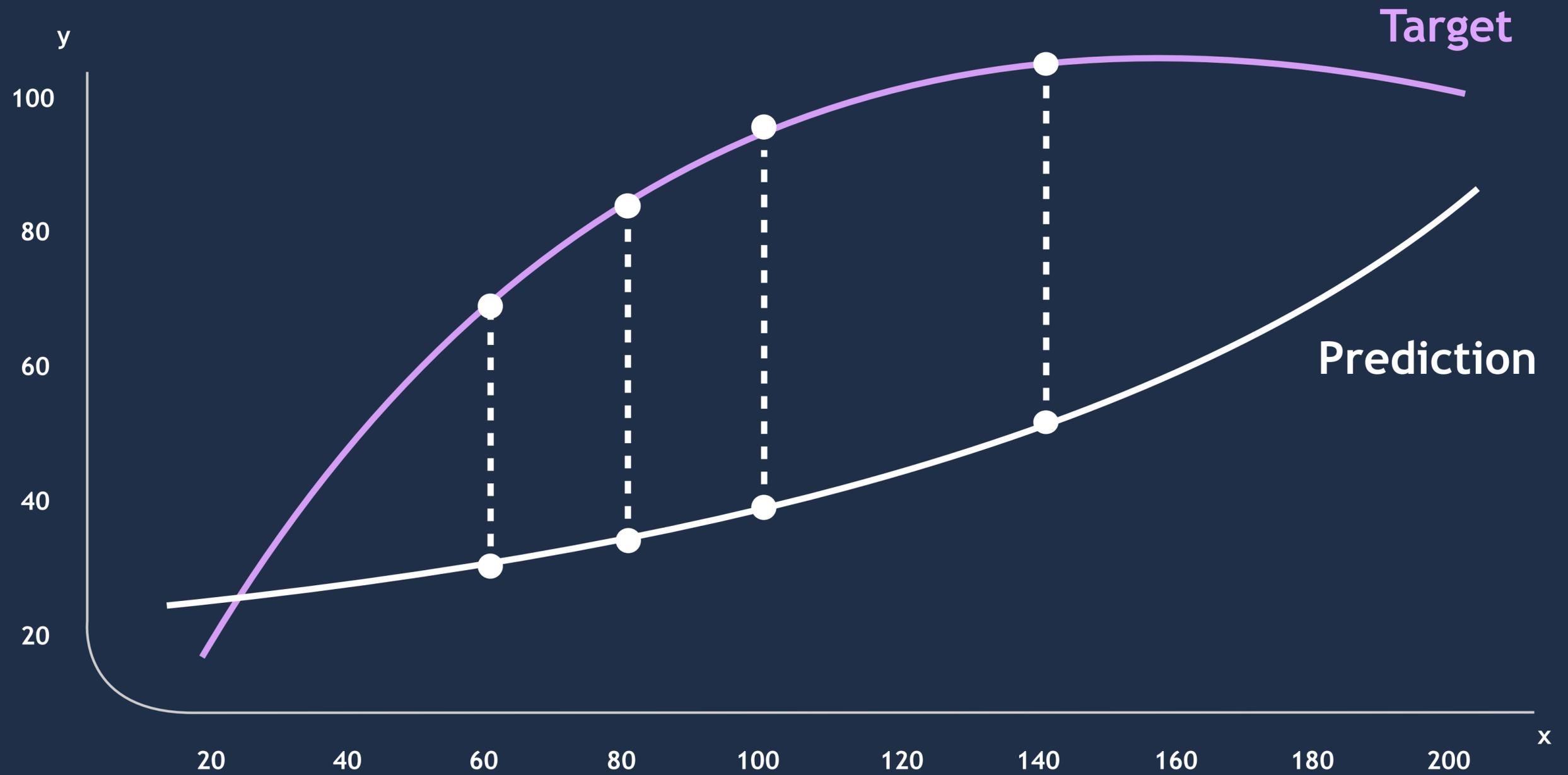
# COST/LOSS



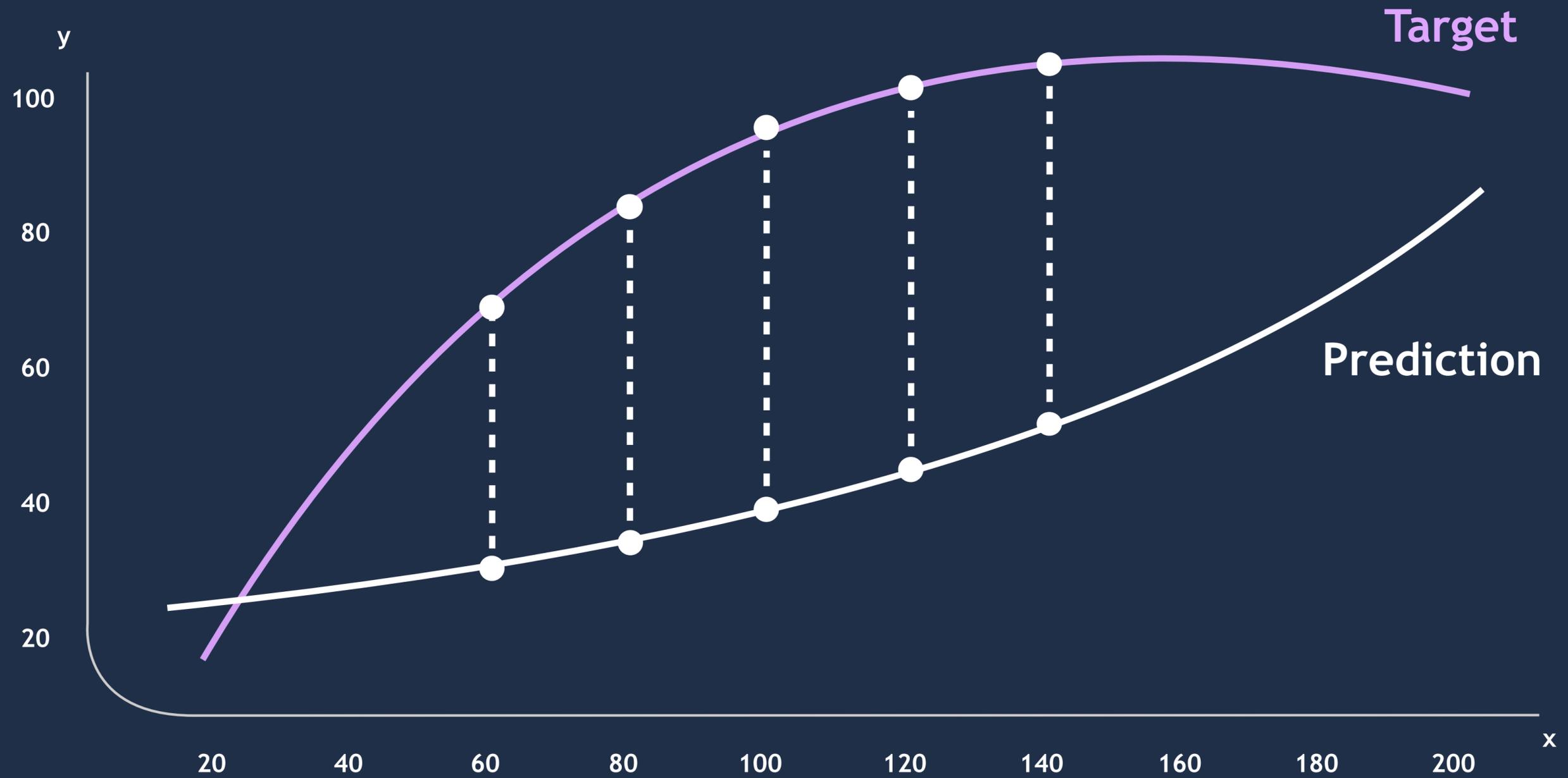
# COST/LOSS



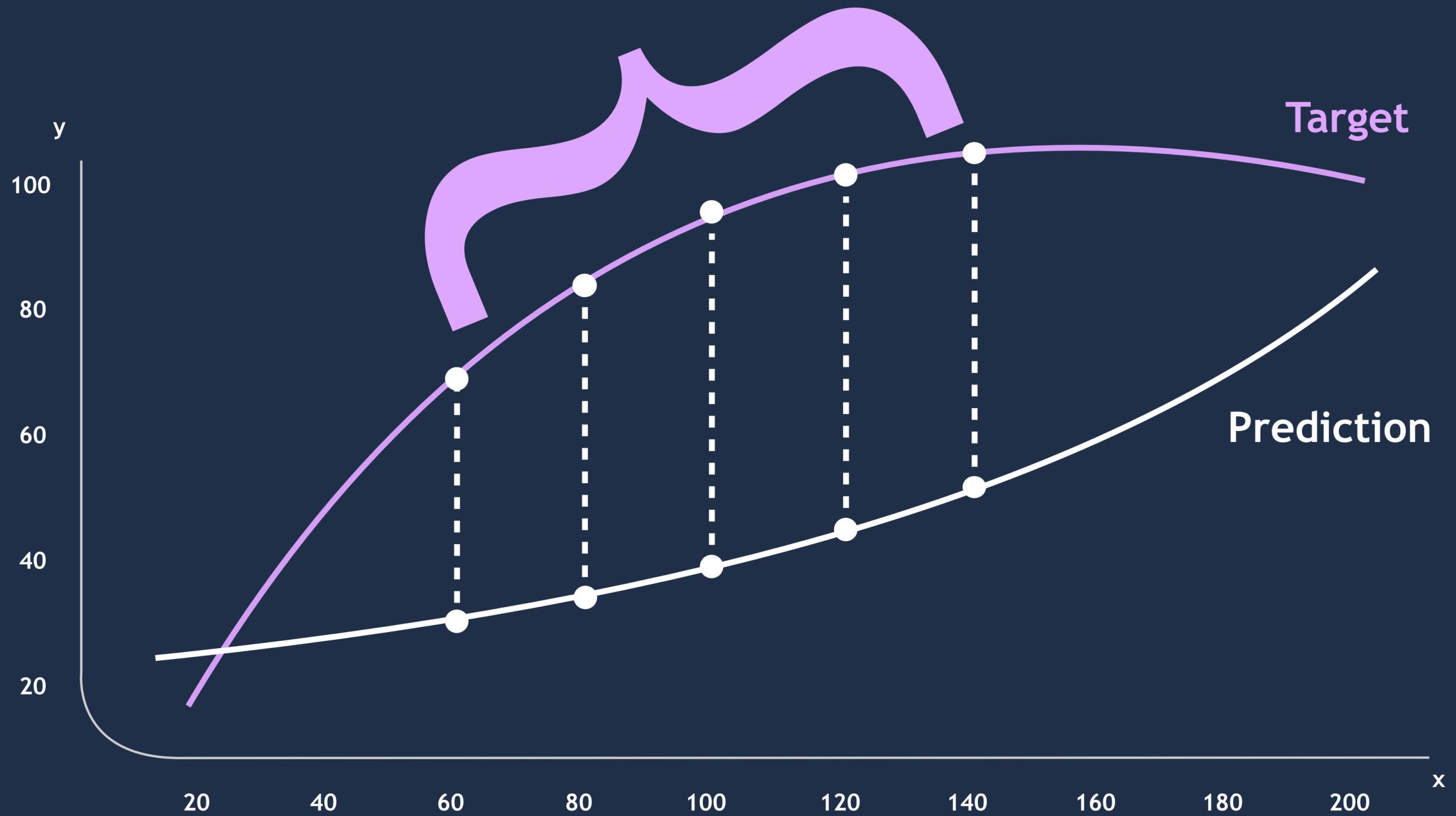
# COST/LOSS



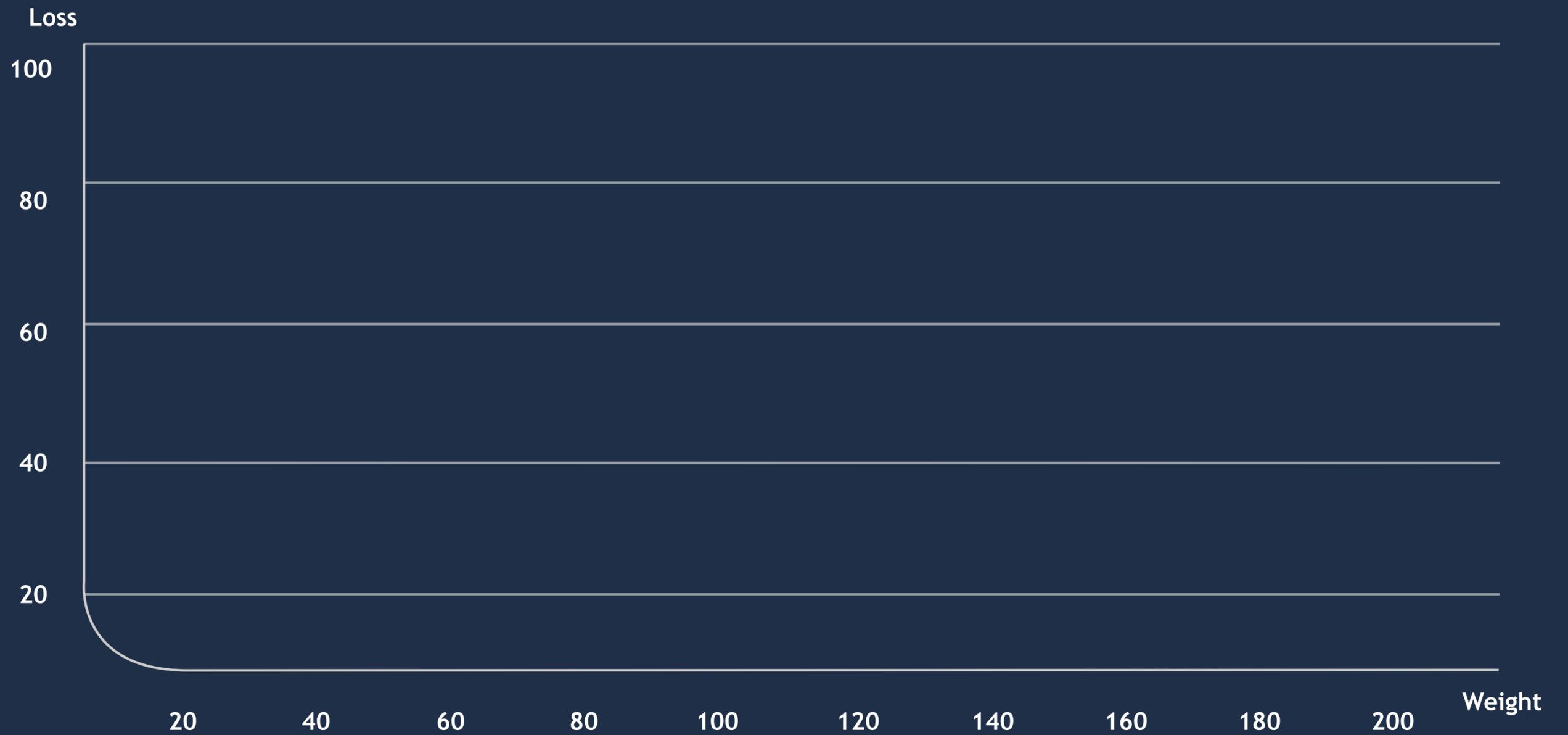
# COST/LOSS



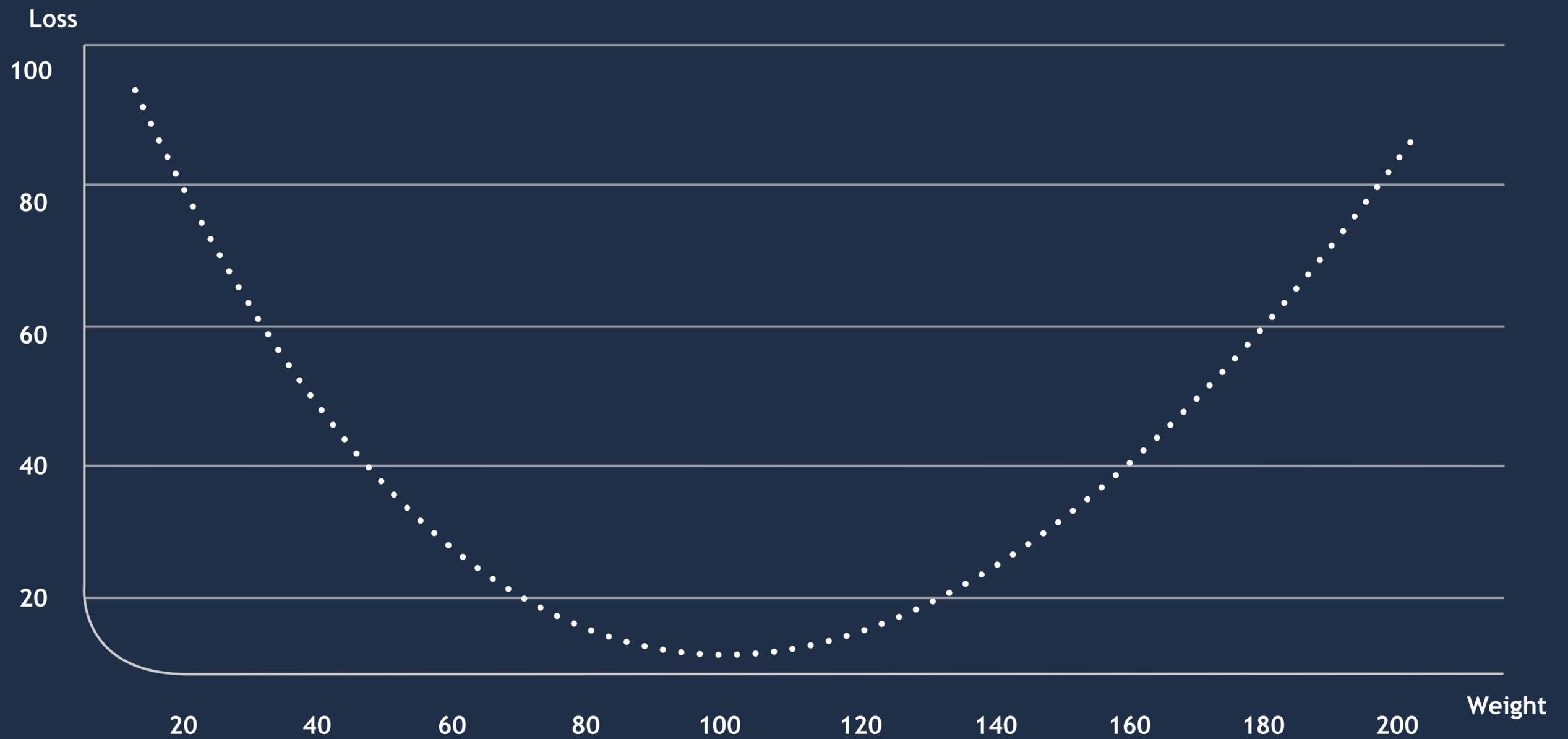
# COST/LOSS



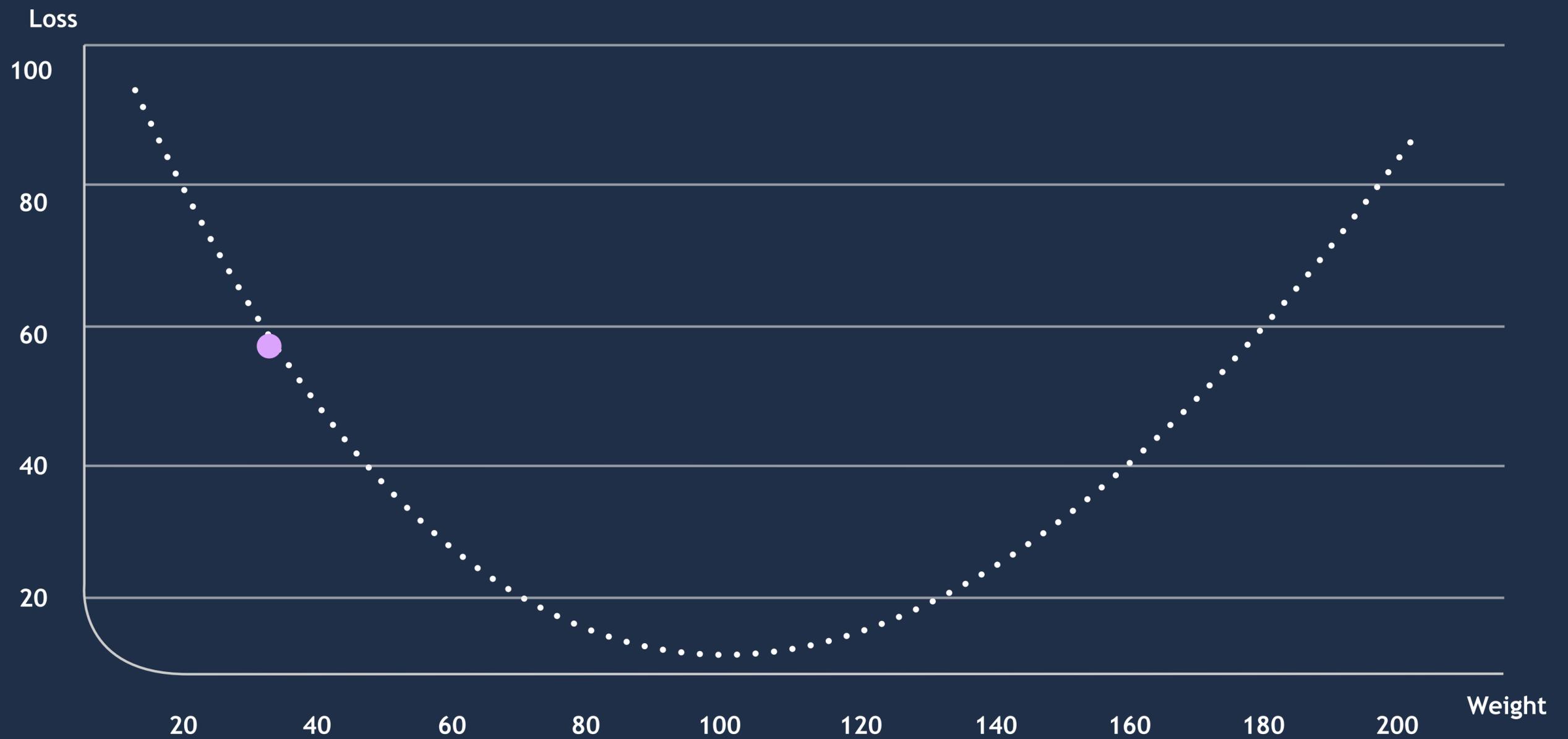
# OPTIMIZER



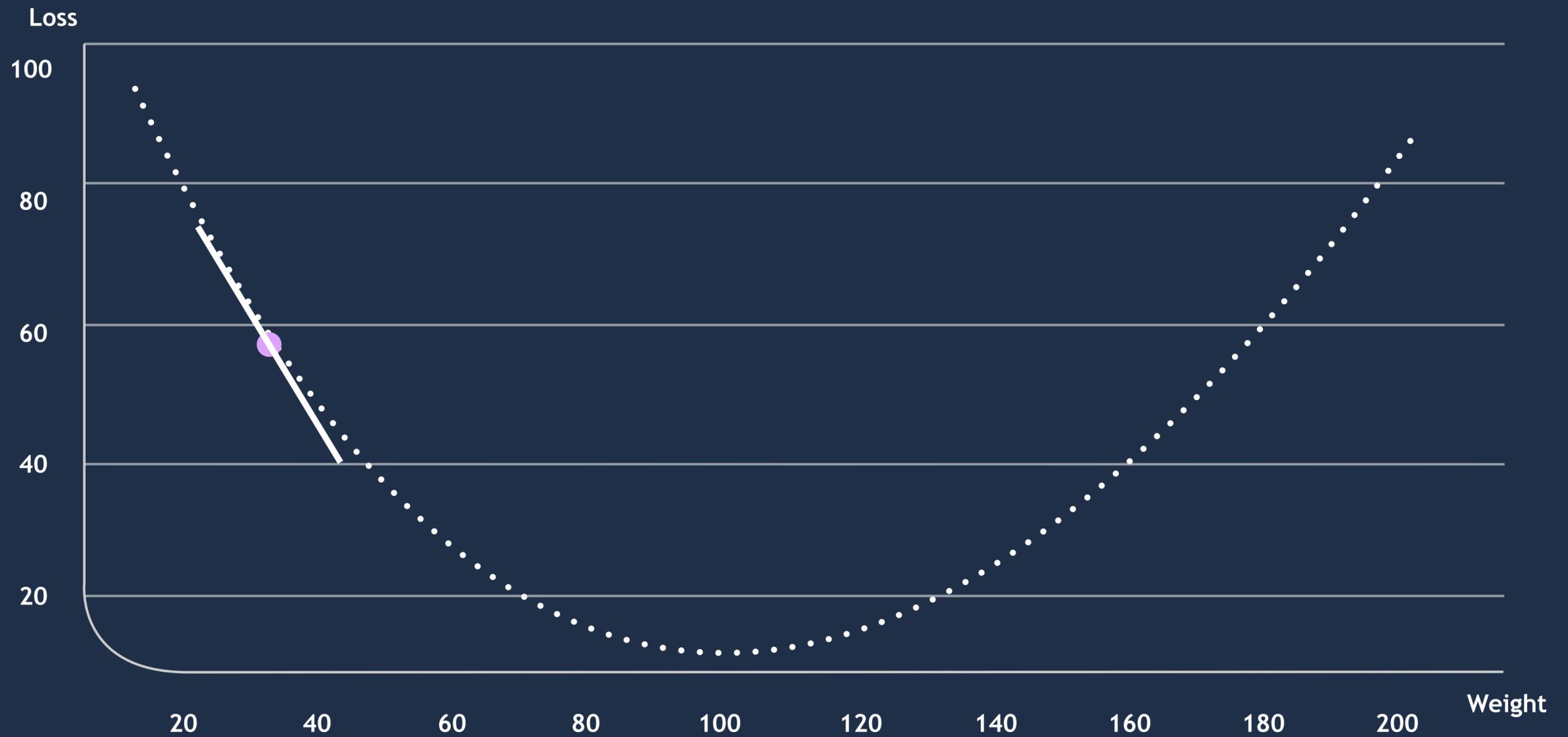
# OPTIMIZER



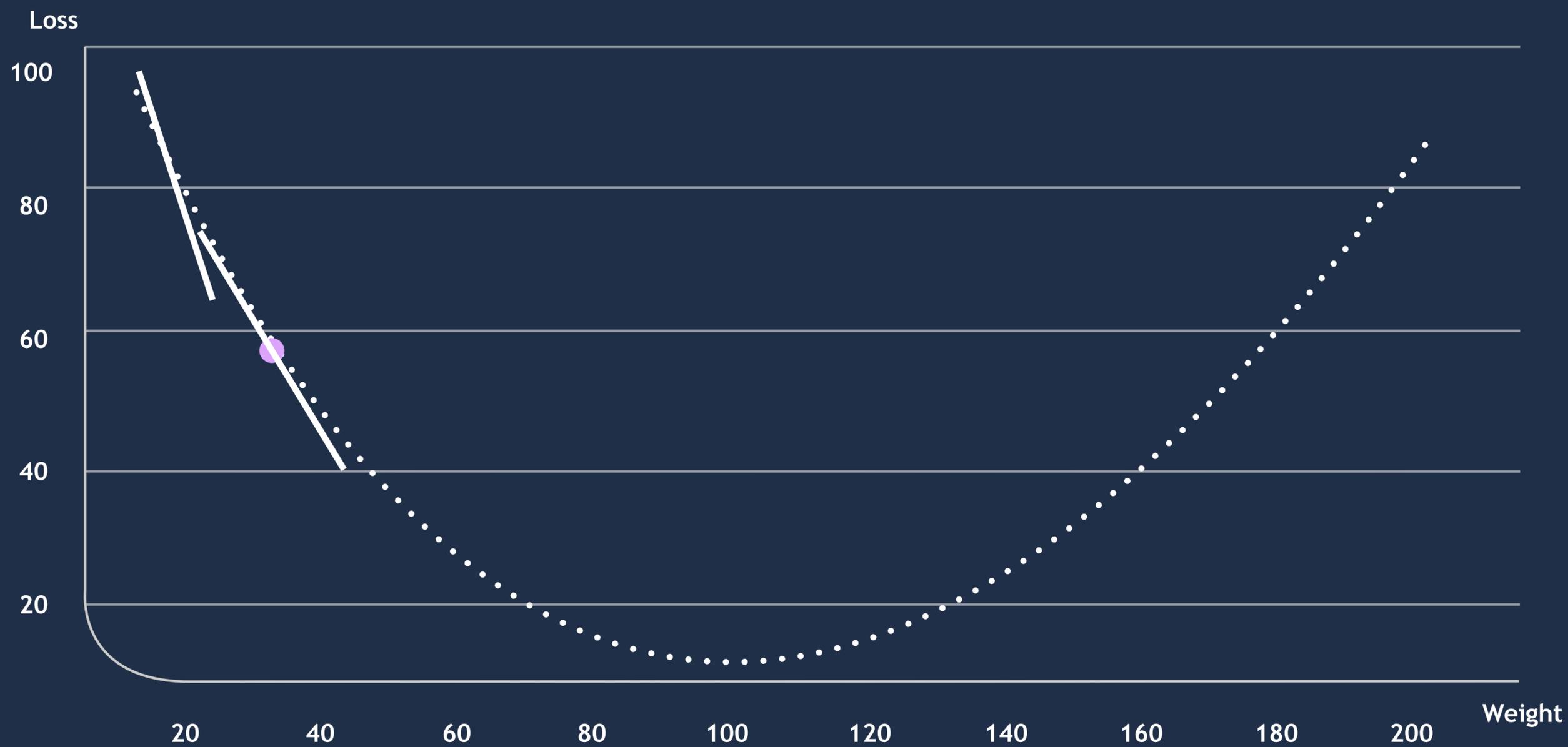
# OPTIMIZER



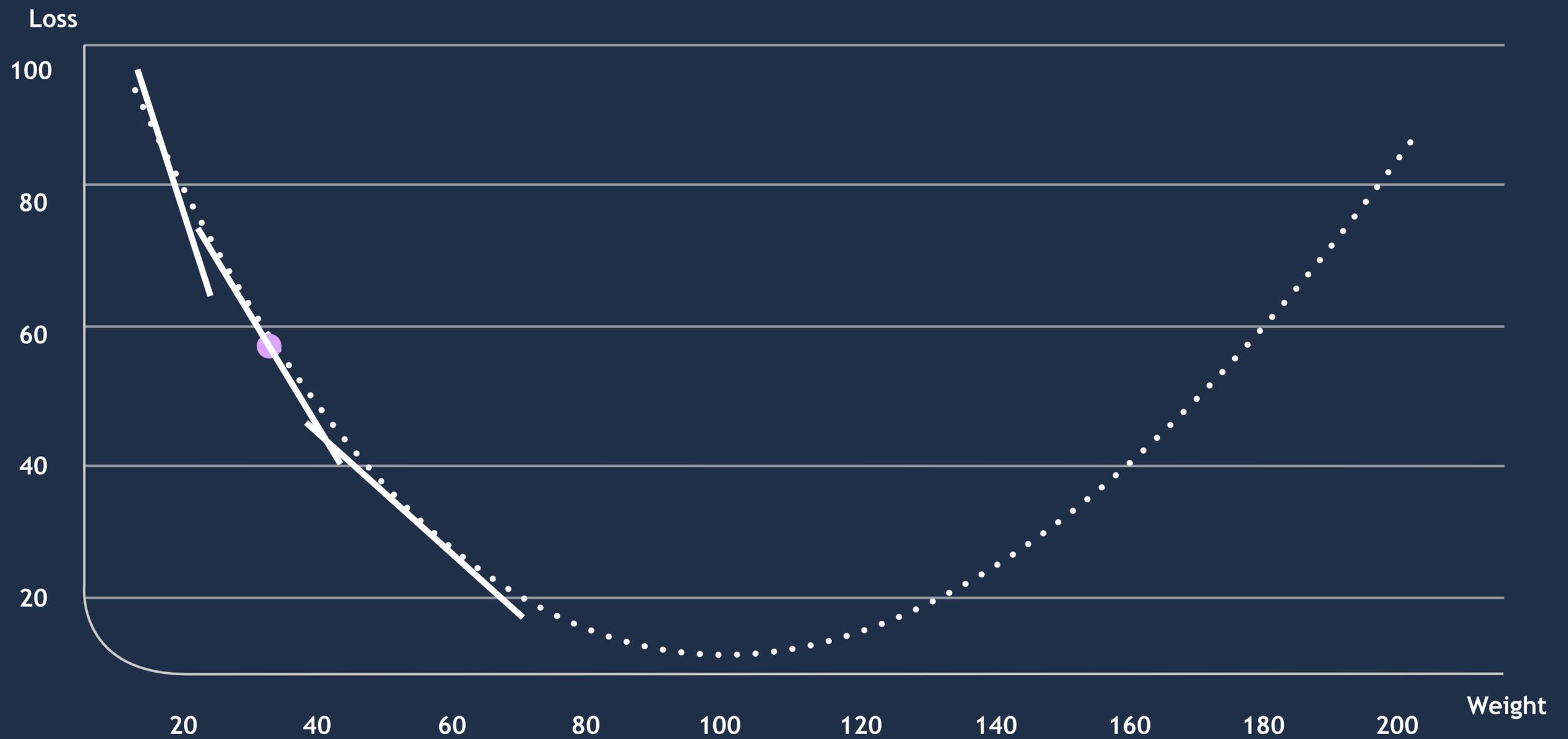
# OPTIMIZER



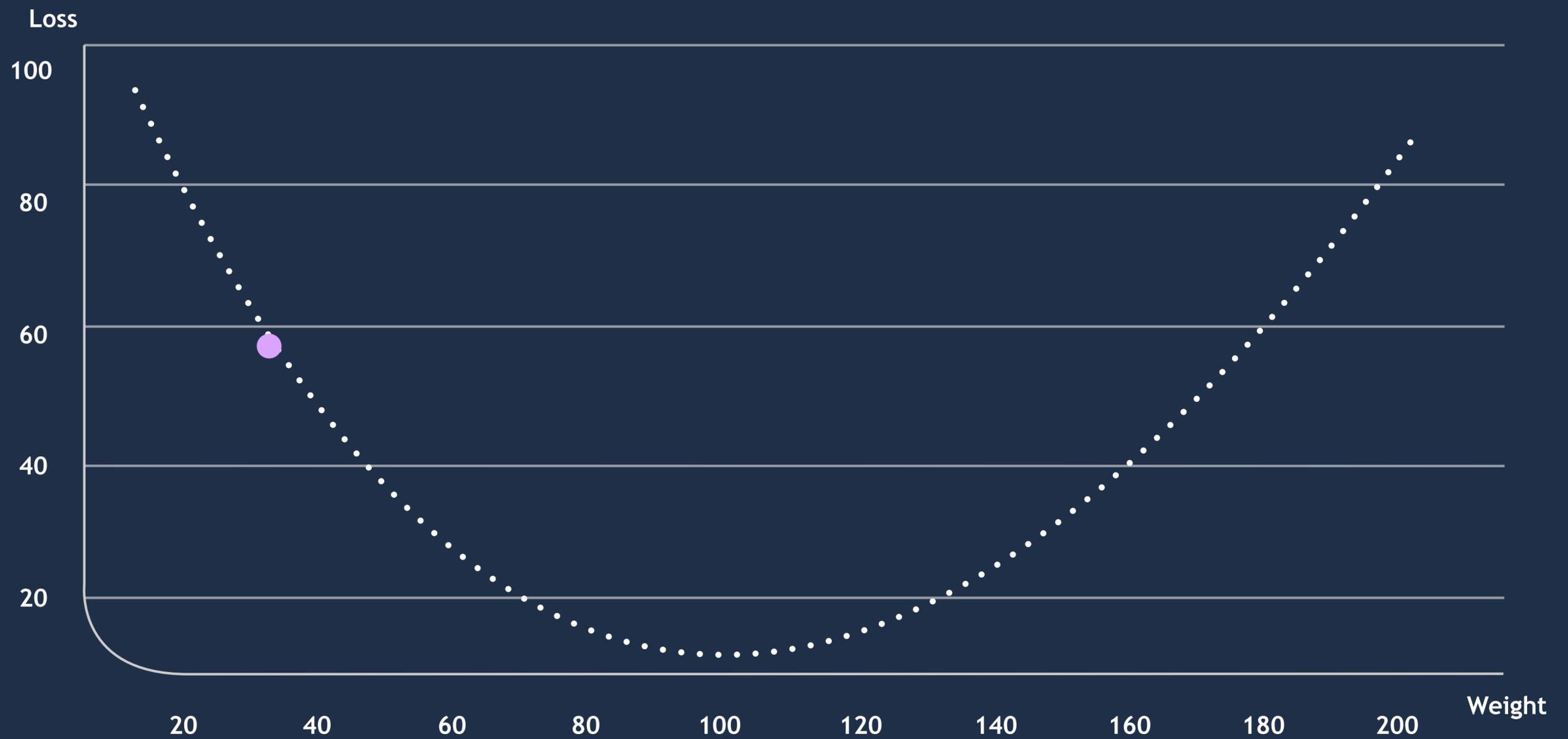
# OPTIMIZER



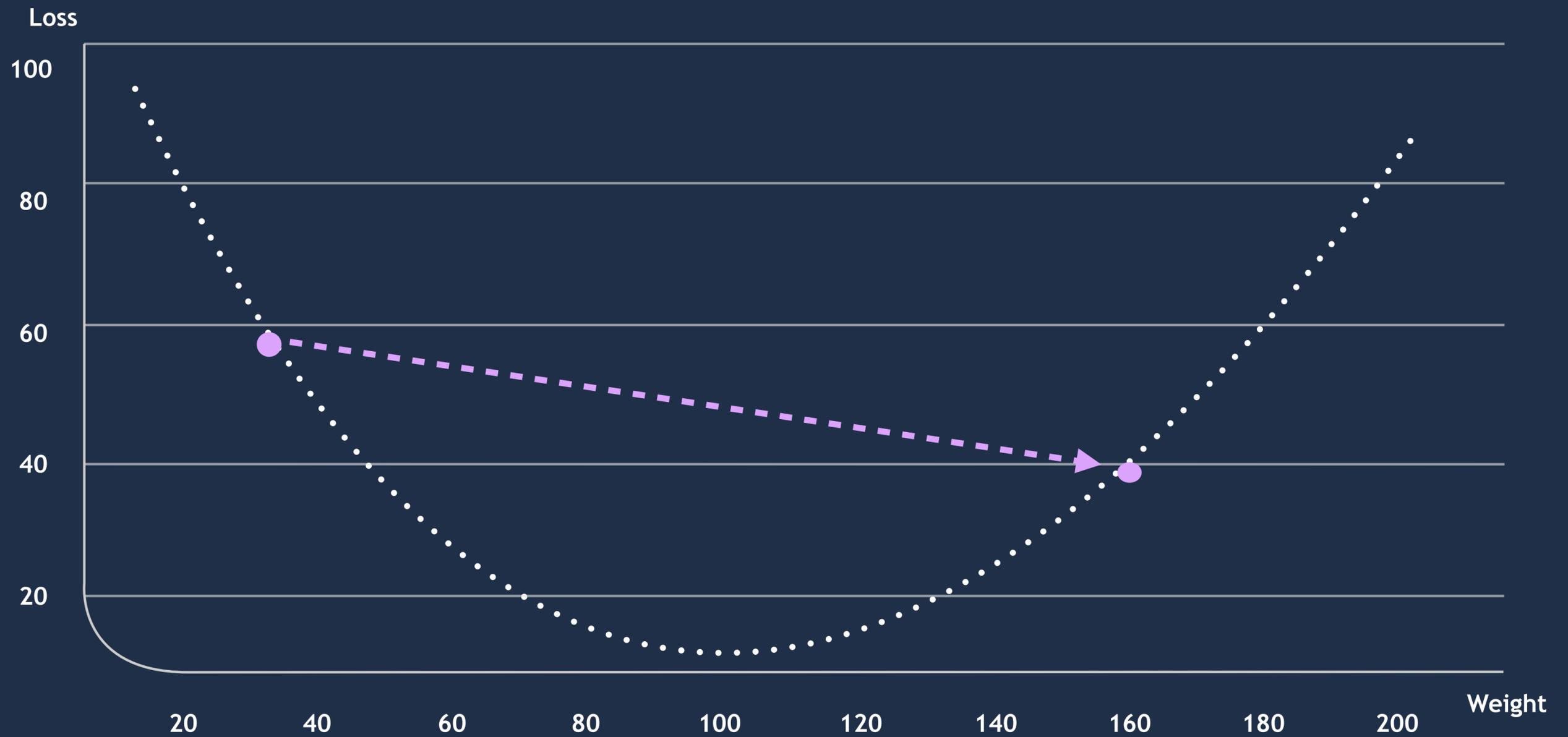
# OPTIMIZER



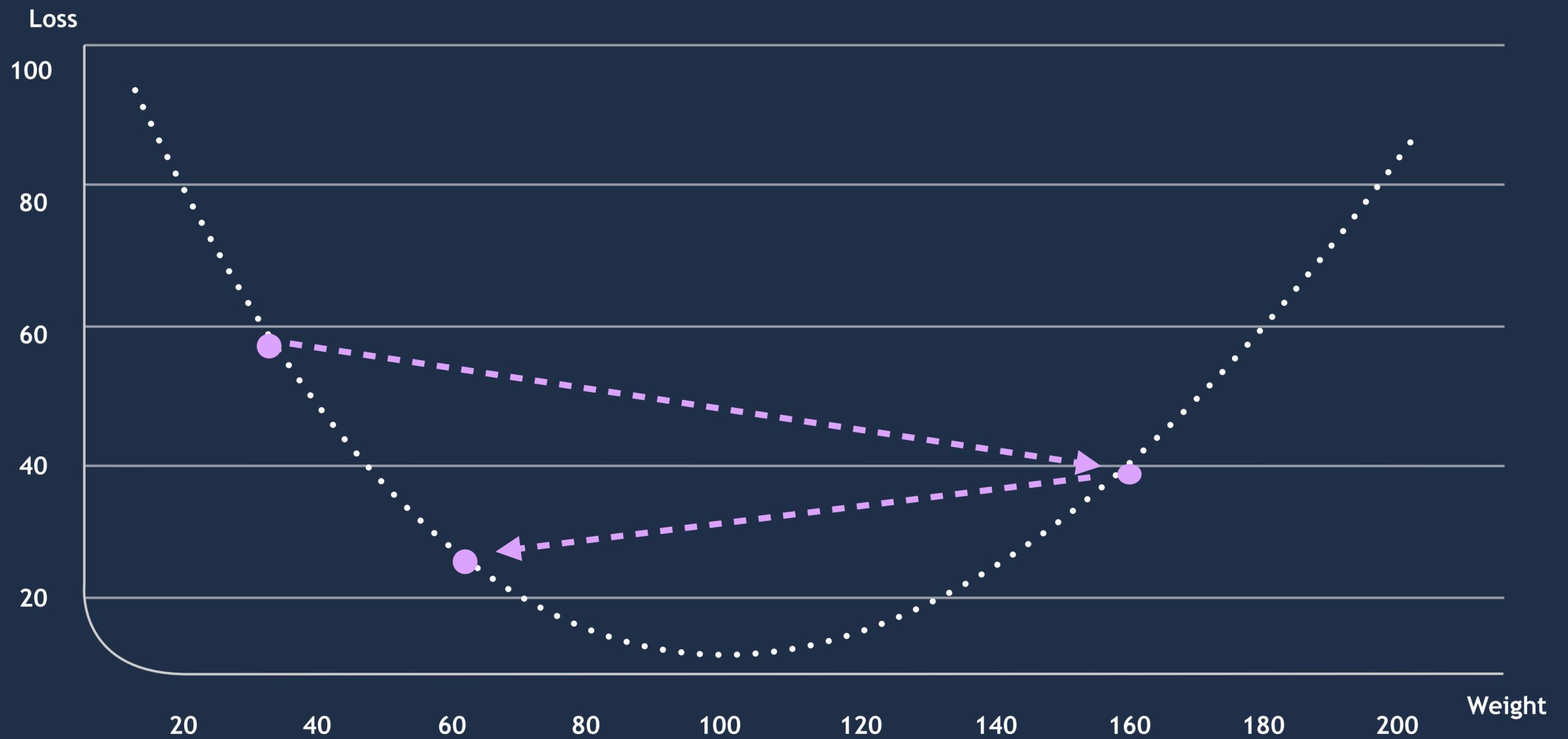
# OPTIMIZER



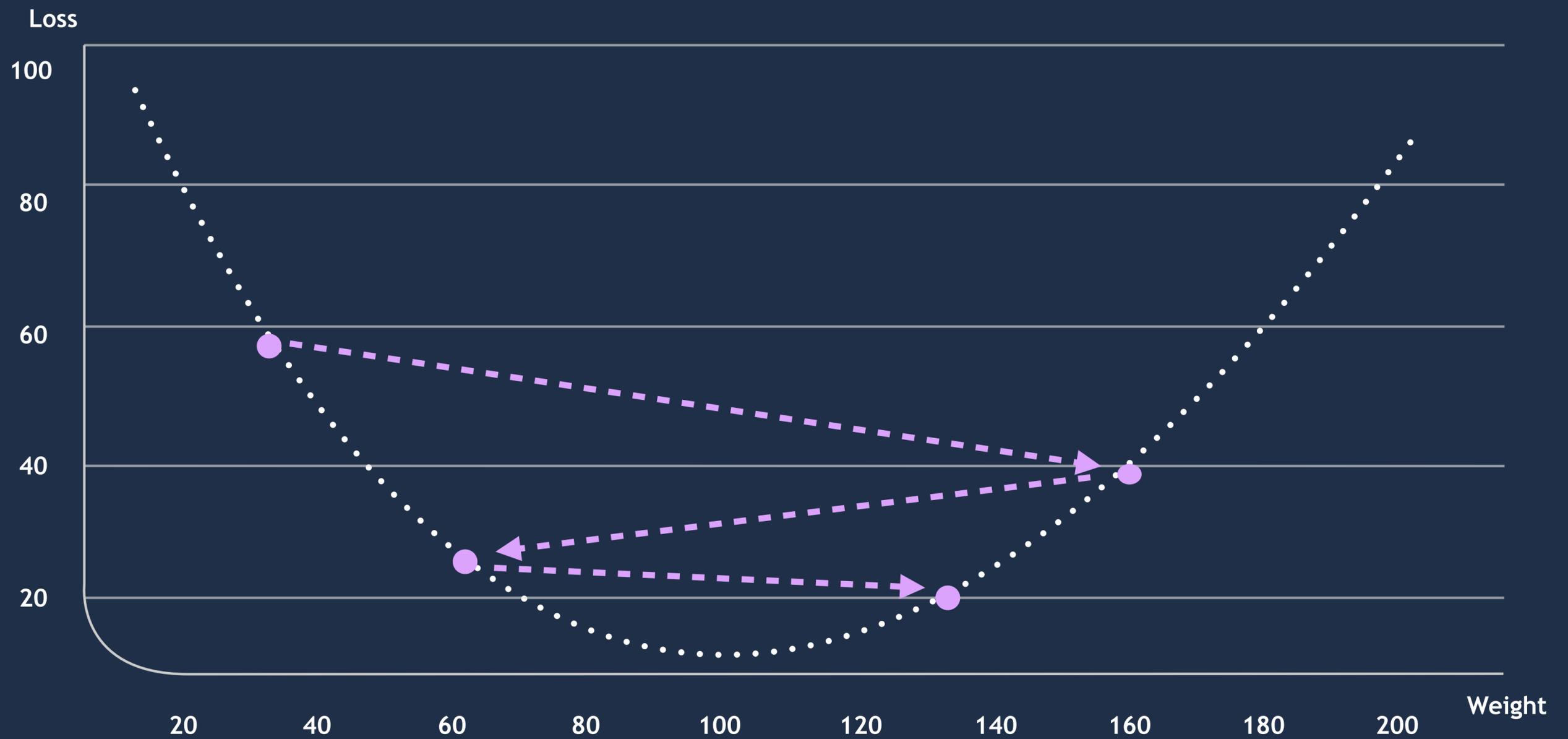
# OPTIMIZER



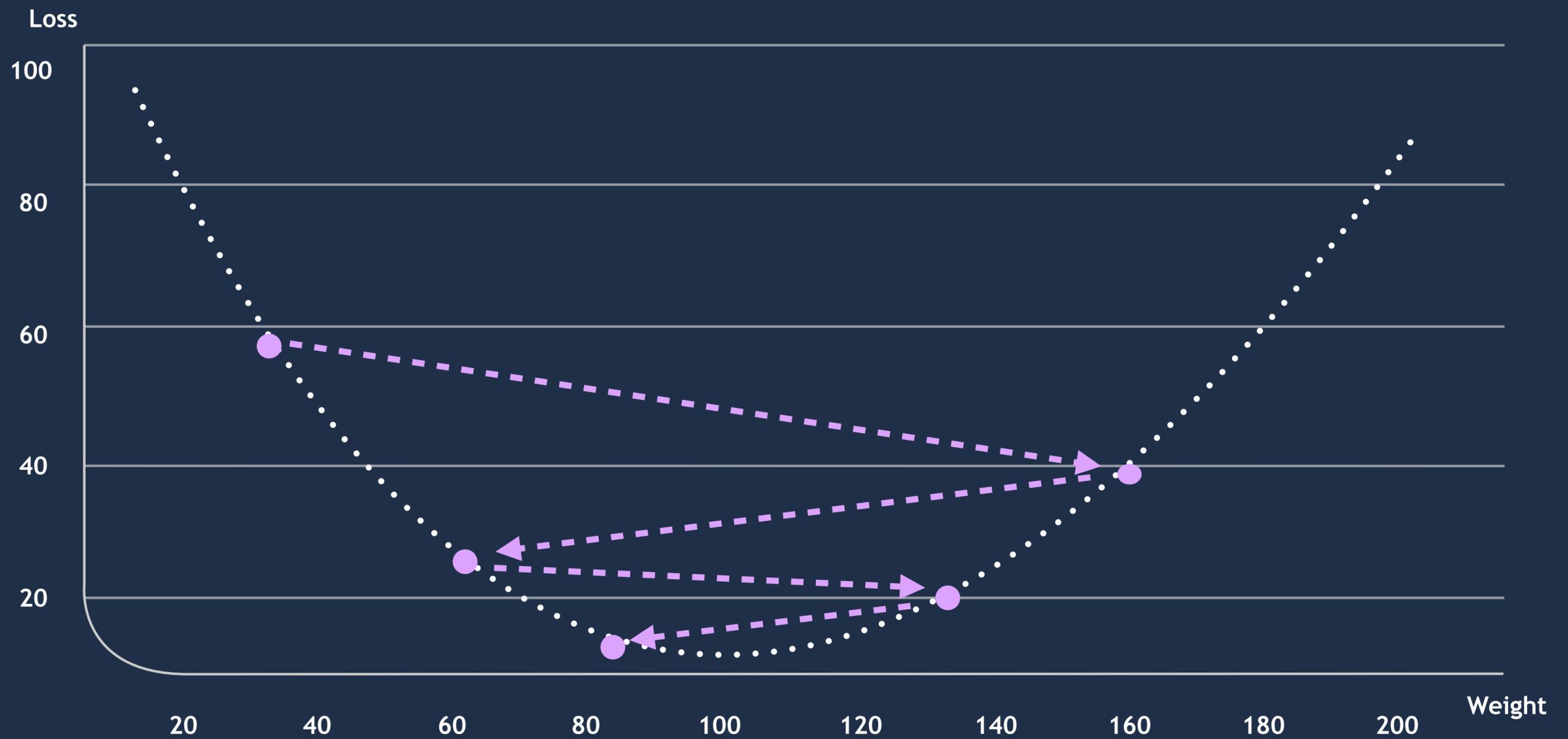
# OPTIMIZER



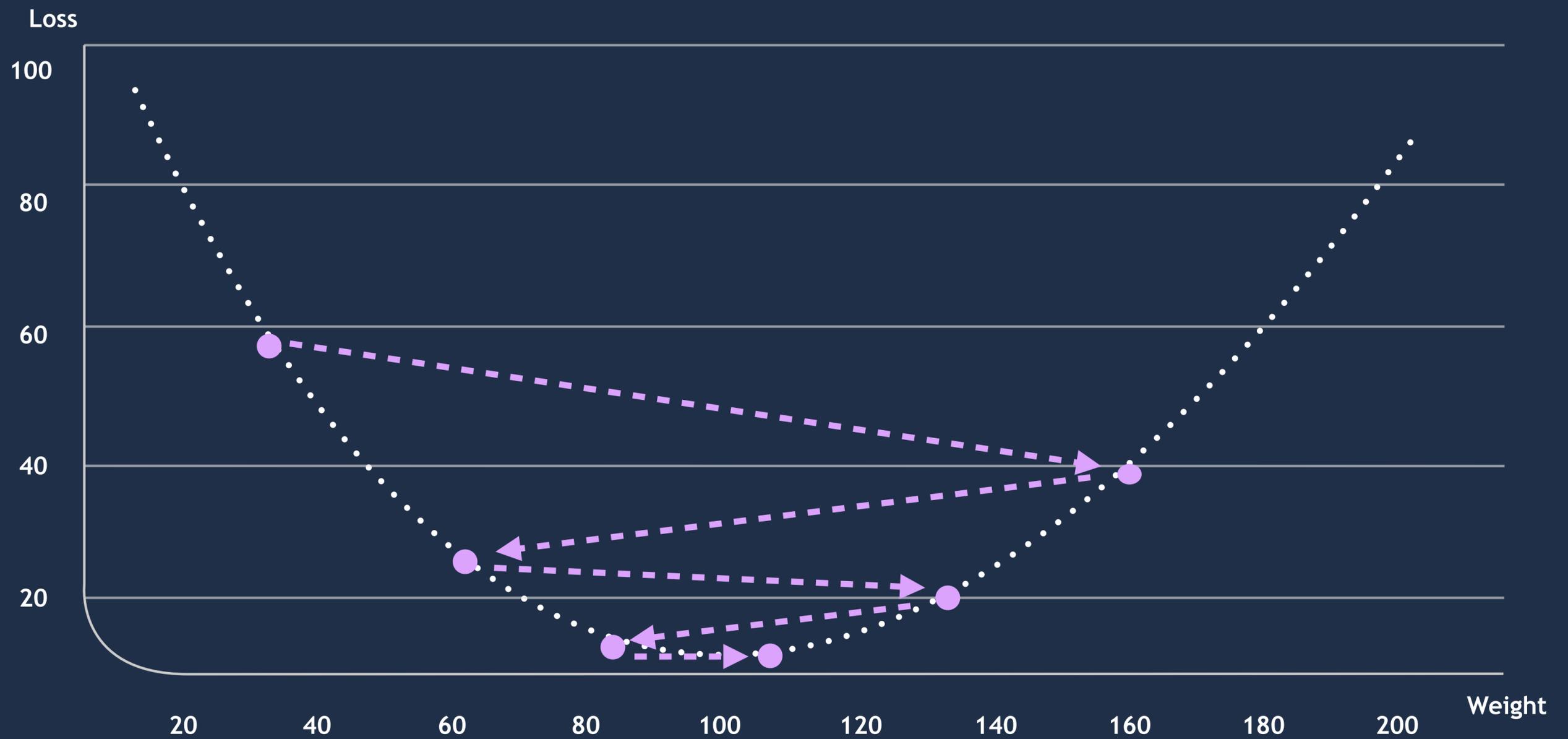
# OPTIMIZER



# OPTIMIZER



# OPTIMIZER



```
1 import * as tf from '@tensorflow/tfjs';  
2  
3 const setupTraining = (model) => {
```

```
11 }
```

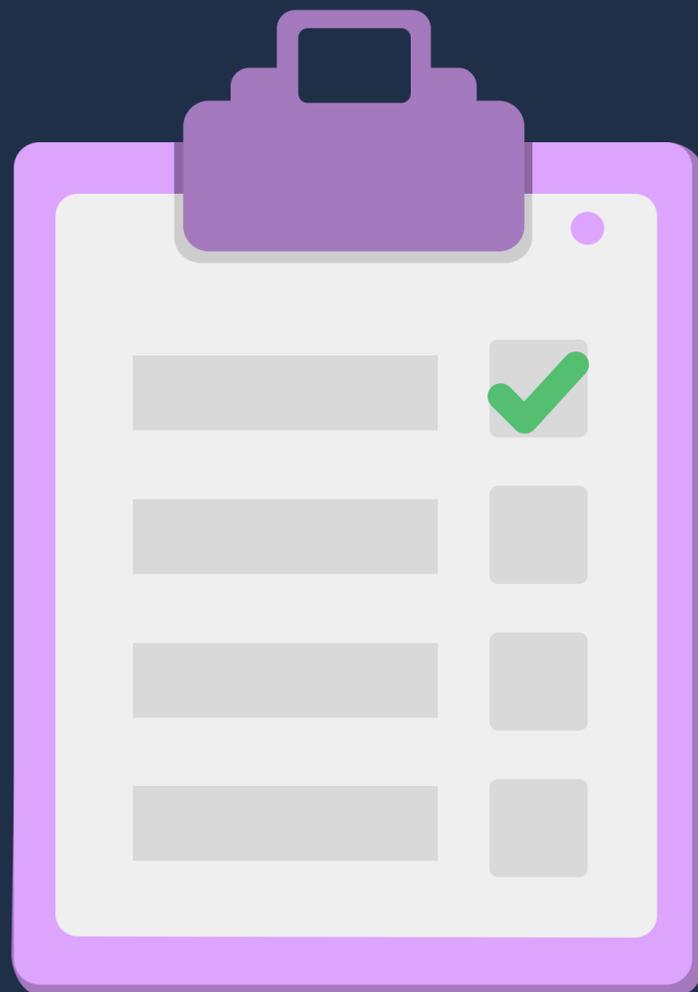
```
1 import * as tf from '@tensorflow/tfjs';  
2  
3 const setupTraining = (model) => {  
4     const learningRate = 0.042;
```

```
11 }
```

```
1 import * as tf from '@tensorflow/tfjs';
2
3 const setupTraining = (model) => {
4   const learningRate = 0.042;
5   const optimizer = tf.train.sgd(learningRate);
6
7   // ...
8
9   // ...
10
11 }
```

```
1 import * as tf from '@tensorflow/tfjs';
2
3 const setupTraining = (model) => {
4   const learningRate = 0.042;
5   const optimizer = tf.train.sgd(learningRate);
6
7   model.compile({
8     optimizer: optimizer,
9     loss: "meanSquaredError"
10  });
11 }
```

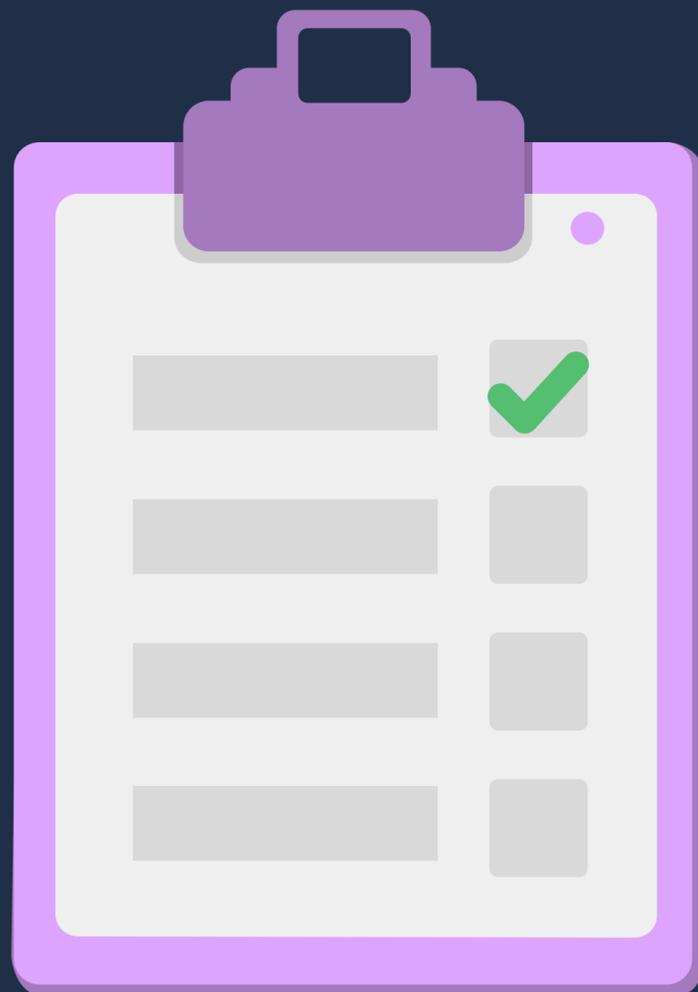
# CHECKLIST



✓ Data

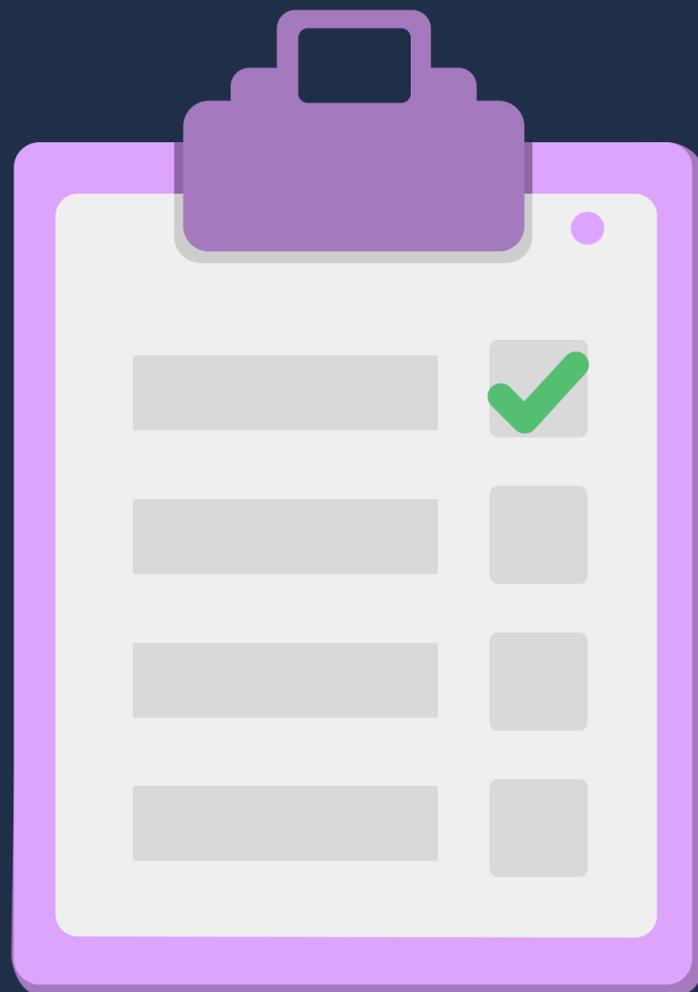
✓ Network

# CHECKLIST



- ✓ Data
- ✓ Network
- ✓ Setup training

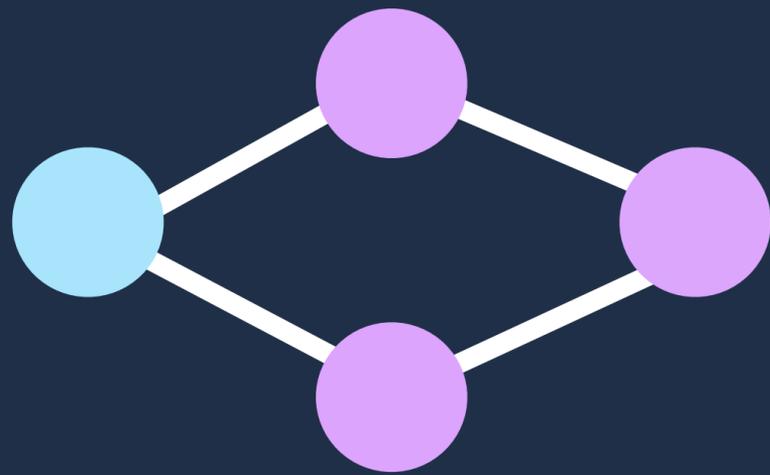
# CHECKLIST



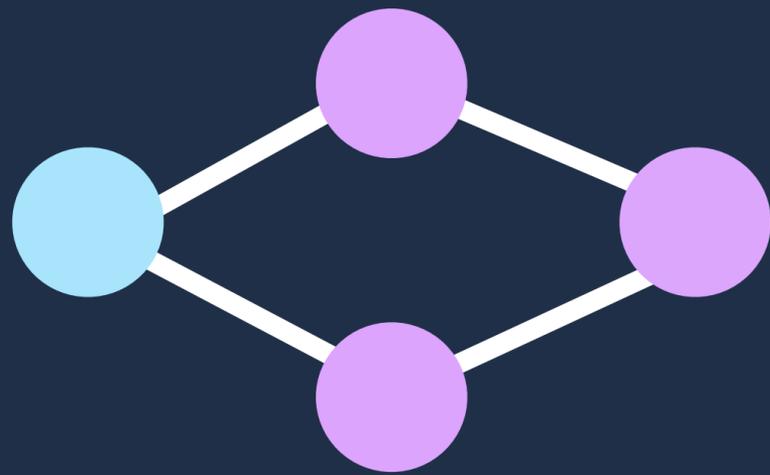
- ✓ Data
- ✓ Network
- ✓ Setup training

Run

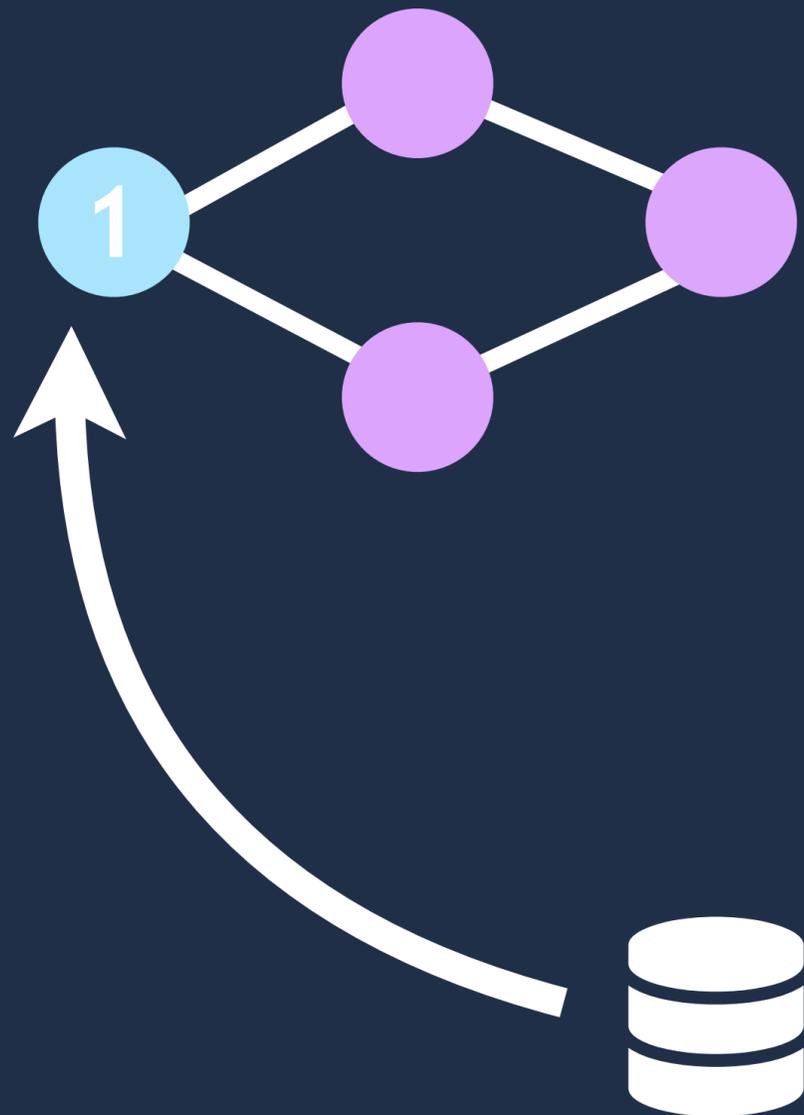
# RUN



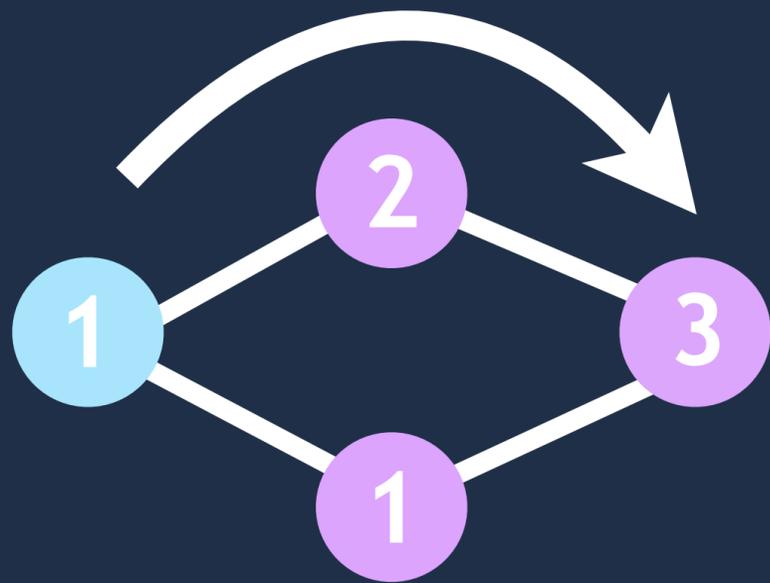
# RUN



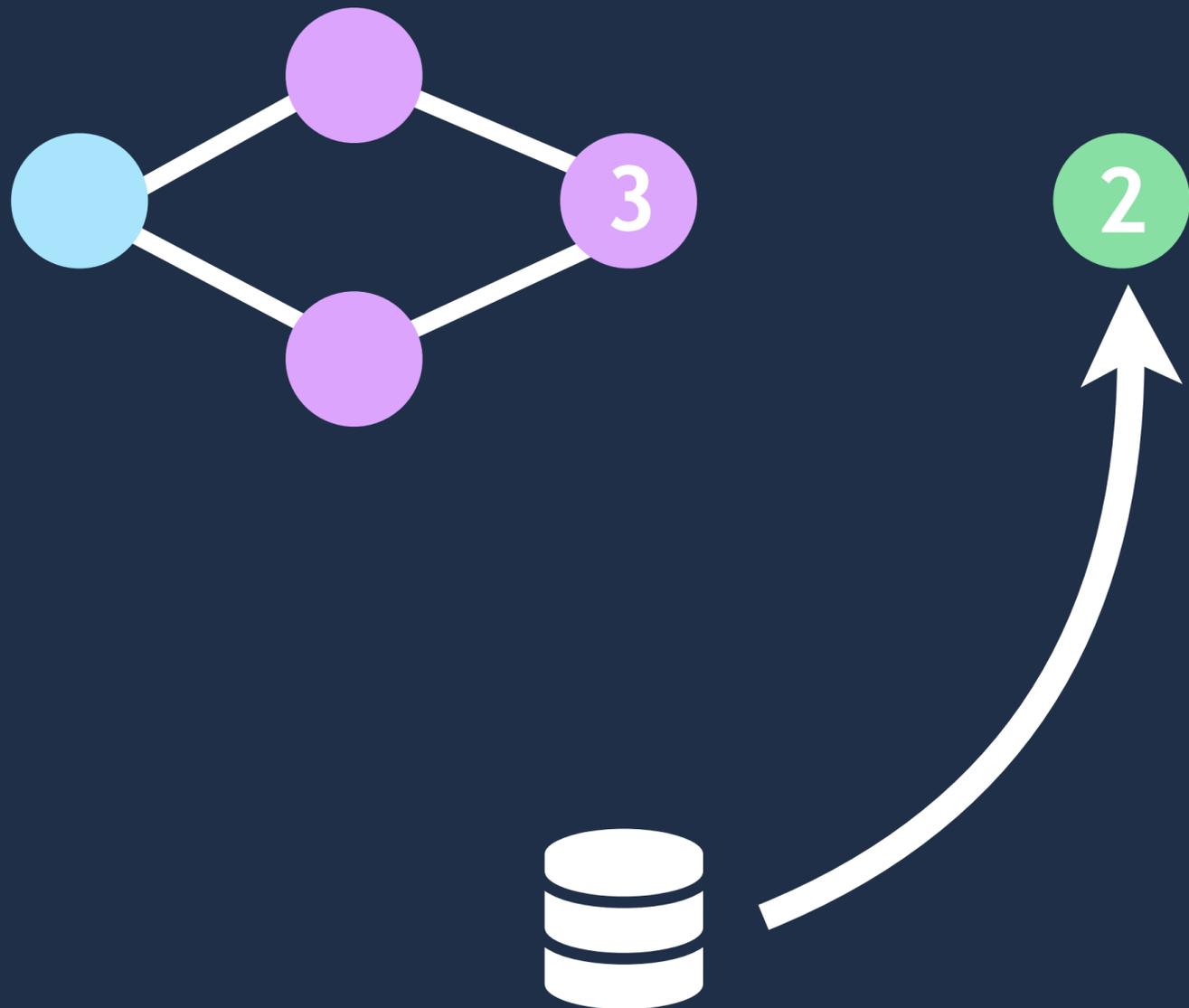
# RUN



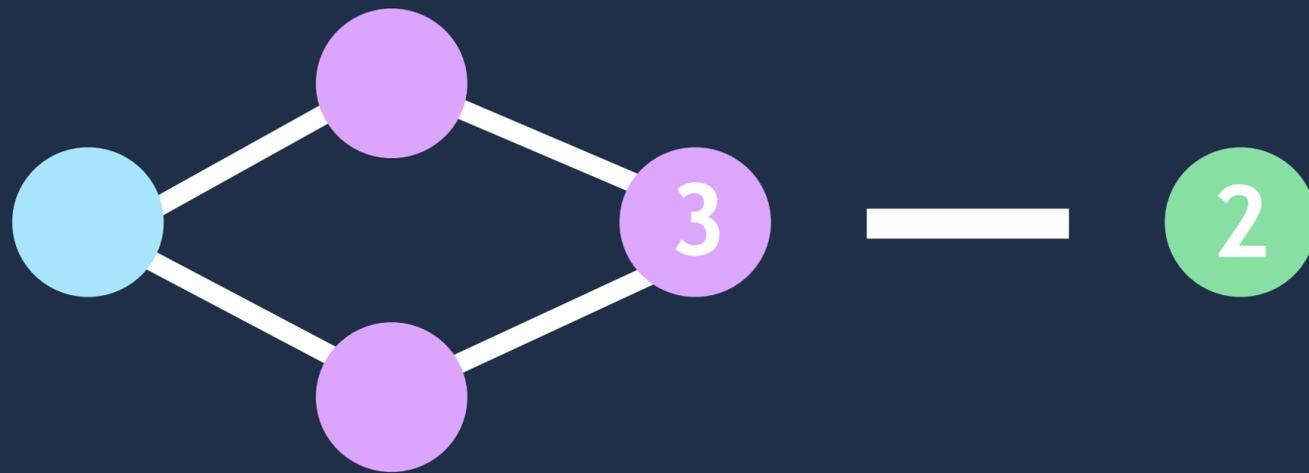
# RUN



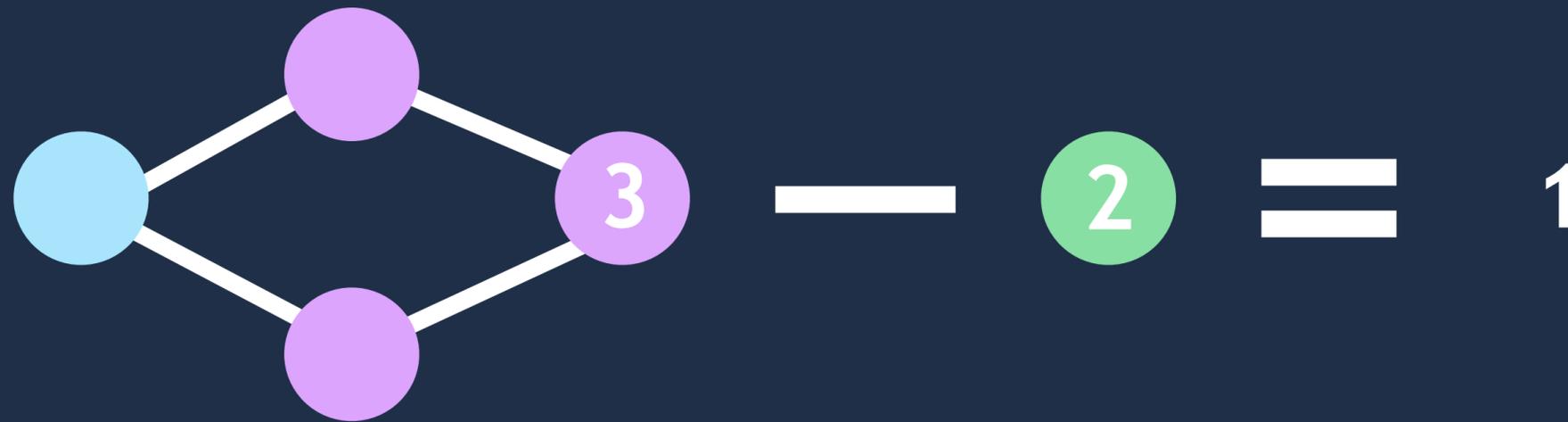
# RUN



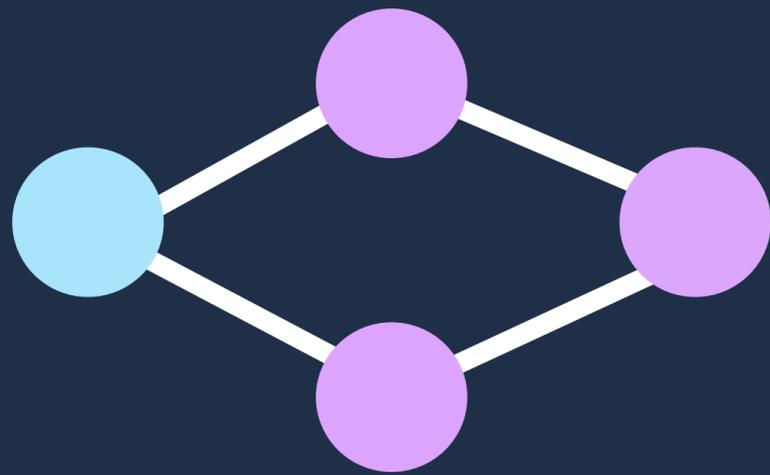
# RUN



# RUN



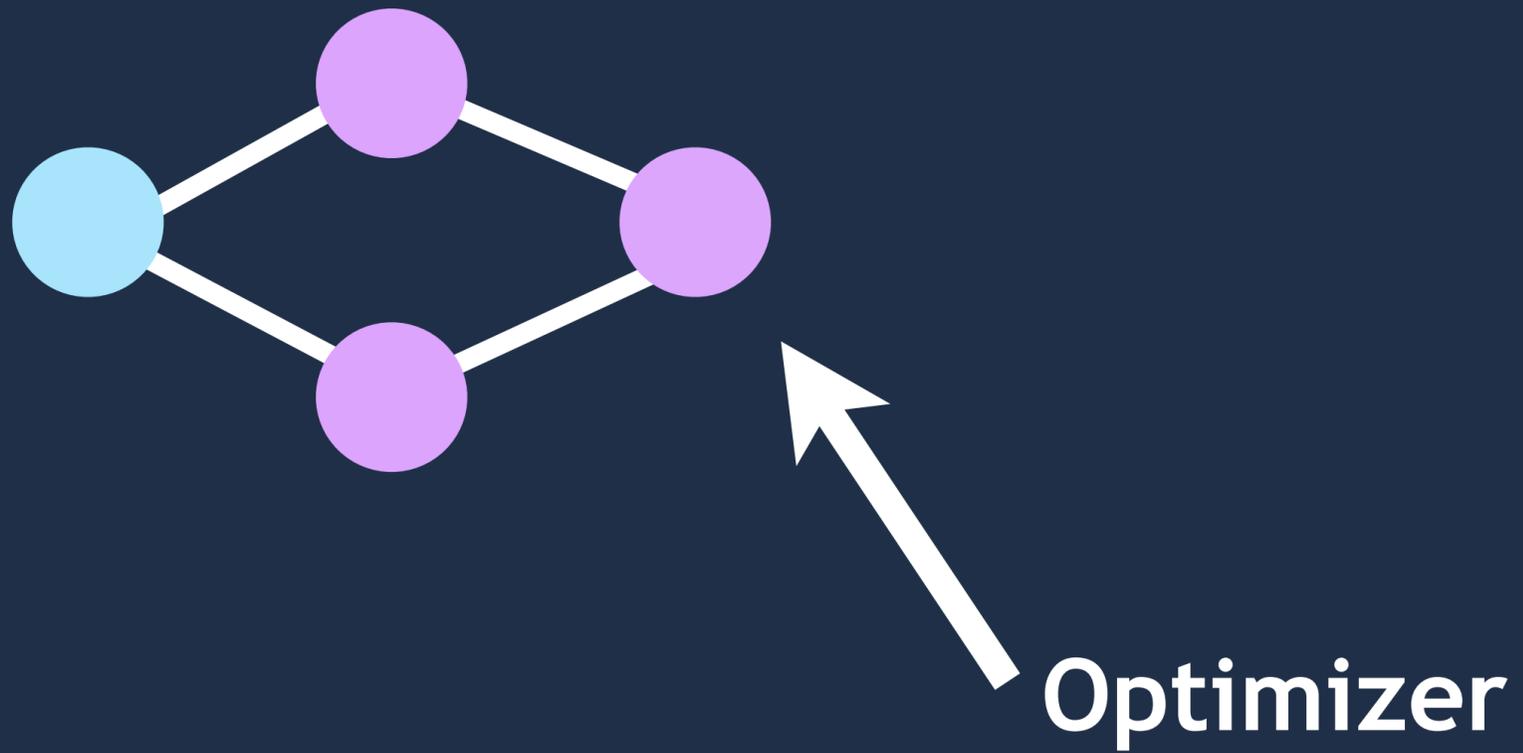
# RUN



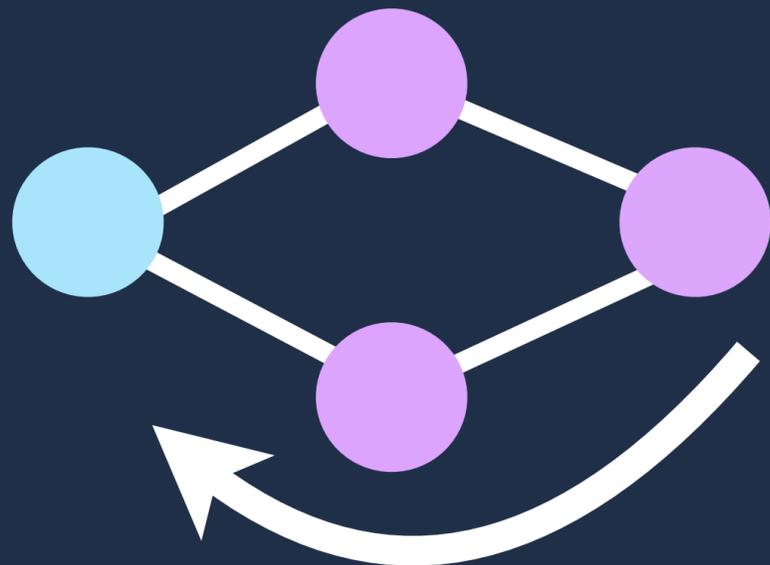
Optimizer



# RUN

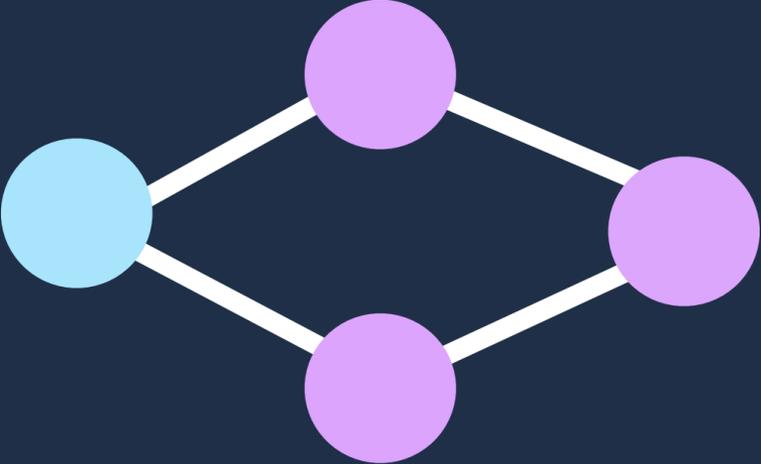
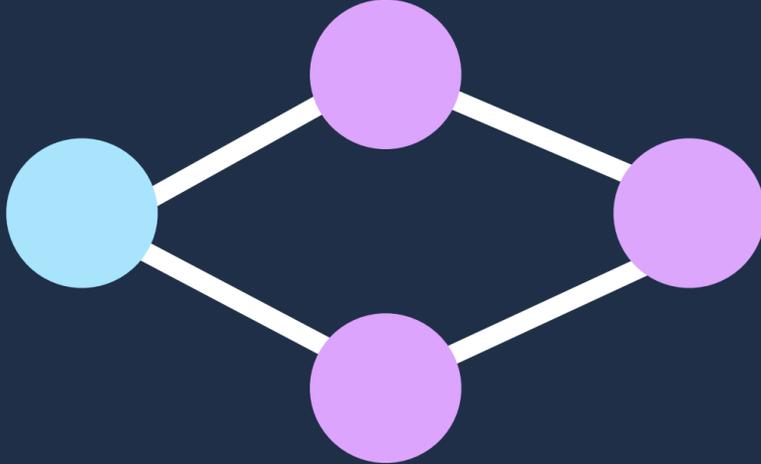
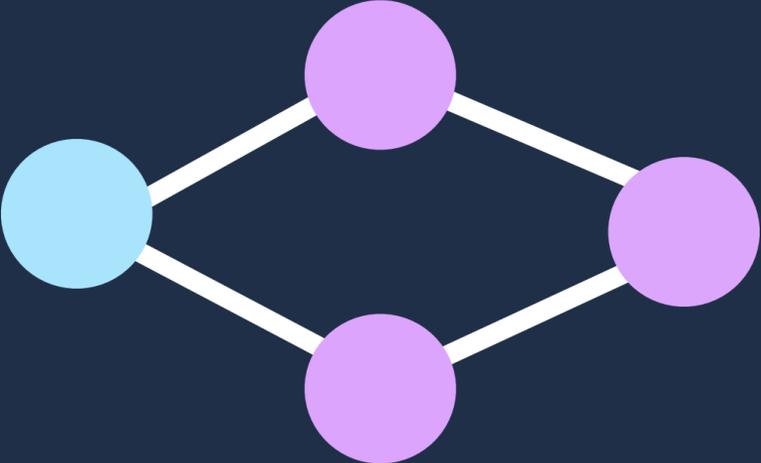


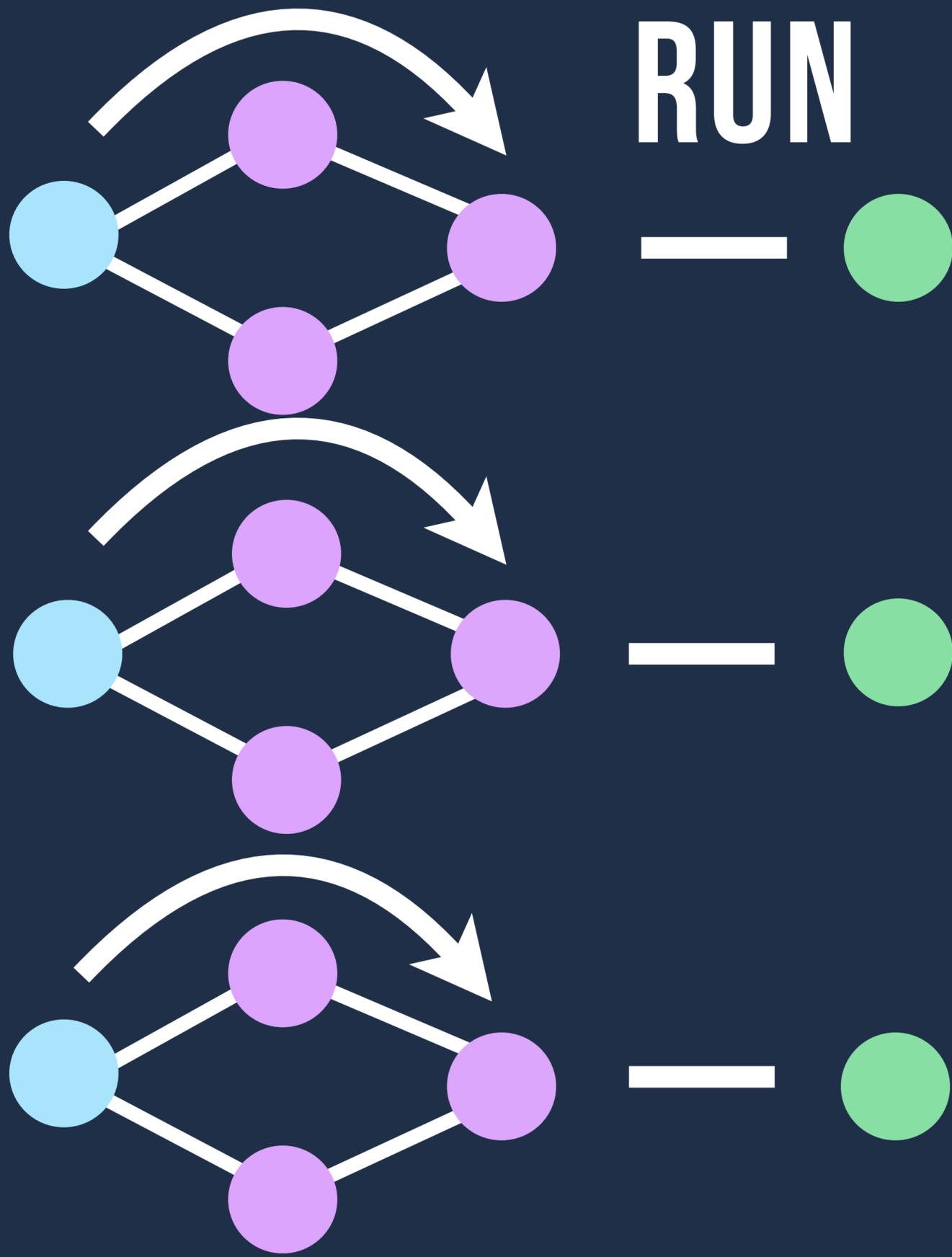
# RUN

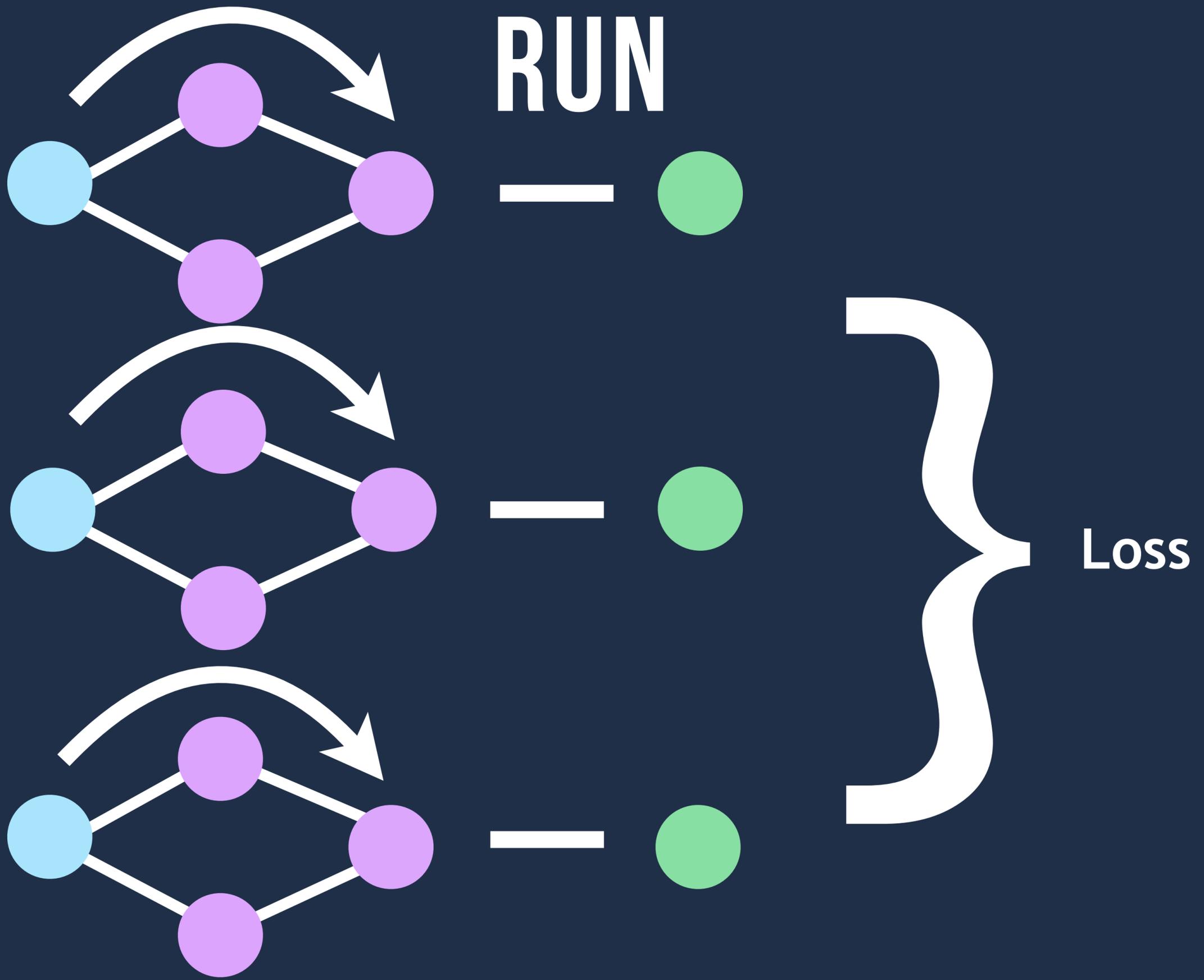


Optimizer

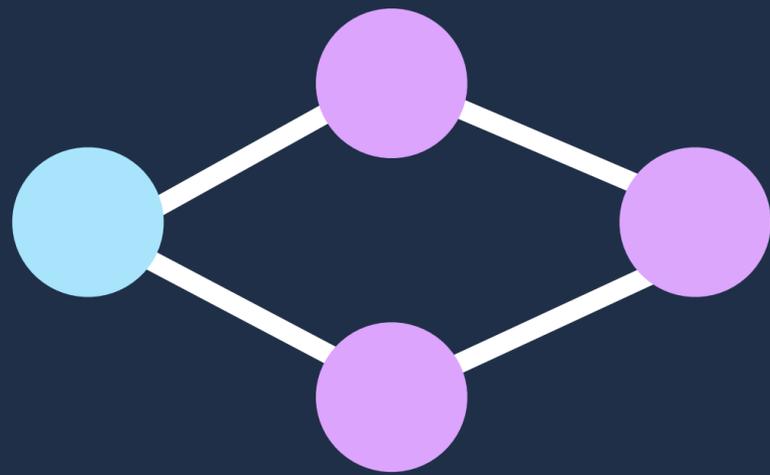
# RUN







# RUN



Loss

# RUN



```
1 ✓ const train = (model, trainingData) => {  
2     const batchSize = 32;  
3     const epochs = 5;
```

```
17 }
```

```
1  ✓ const train = (model, trainingData) => {  
2    const batchSize = 32;  
3    const epochs = 5;  
4  
5  ✓  return model.fit(  
6    trainingData.features,  
7    trainingData.targets,
```

```
16  });  
17  }
```

```
1  ✓ const train = (model, trainingData) => {  
2    const batchSize = 32;  
3    const epochs = 5;  
4  
5  ✓  return model.fit(  
6    trainingData.features,  
7    trainingData.targets,  
8  ✓  {  
9    batchSize: batchSize,  
10   epochs: epochs  
11   }
```

```
16   });
```

```
17 }
```

```
1  ✓ const train = (model, trainingData) => {
2    const batchSize = 32;
3    const epochs = 5;
4
5  ✓   return model.fit(
6      trainingData.features,
7      trainingData.targets,
8  ✓   {
9      batchSize: batchSize,
10     epochs: epochs
11   }
12  ✓ ).then(history => {
13     const losses = history.history.loss;
14     const avgLoss = losses.reduce((acc, val) => acc += val, 0) / losses.length;
15     return avgLoss
16   });
17 }
```

**DEMO**

Start

Batch #	input	target	predicted	cost
0	rgb(222,165,255)	rgb(198,255,165)		0

Stop

Start

Batch #	input	target	predicted	cost
0	rgb(222,165,255)	rgb(198,255,165)		0

Stop

# LINKS

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>
- <https://github.com/Kulbear/deep-learning-nano-foundation/wiki>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>
- <https://github.com/Kulbear/deep-learning-nano-foundation/wiki>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>
- <https://github.com/Kulbear/deep-learning-nano-foundation/wiki>
- <https://www.coursera.org/learn/machine-learning>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>
- <https://github.com/Kulbear/deep-learning-nano-foundation/wiki>
- <https://www.coursera.org/learn/machine-learning>

# LINKS

- <https://github.com/Chimney42/deeplearn-complementary-colors>
- <https://js.tensorflow.org/>
- <https://github.com/Kulbear/deep-learning-nano-foundation/wiki>
- <https://www.coursera.org/learn/machine-learning>
- <https://fontawesome.com/>



# WE ARE HIRING!



Amsterdam



London



Zurich



Berlin

[www.container-solutions.com/careers](http://www.container-solutions.com/careers)  
[jobs@container-solutions.com](mailto:jobs@container-solutions.com)

@CHIMNEY42