REGULARIZATION

REGULARIZATION

USAGE

Used in the development for more complex and deep models.

WHY?

A method to penalize models when it starts to memorize patterns (or over-learning from training datasets). Over-learning occurs when the model does not learn from the feature but memorizes some samples of the training data.

IMPACT

Forces the model to generalize towards unseen data. Useful only for complicate models with multiple hidden layers, and for large complex data like images.

CAVEAT

It will slightly decrease the training performance metrics but will increase testing performance metrics.

COMMONLY USED

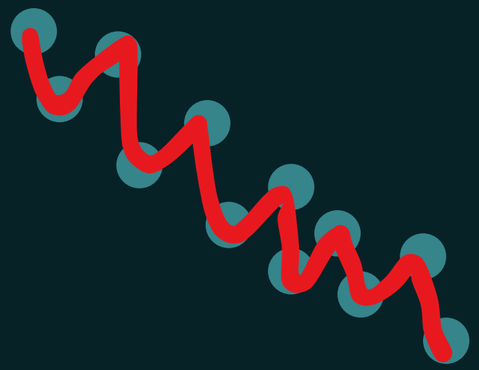
DROPOUT — Modifying the model by shutting down some neurons.

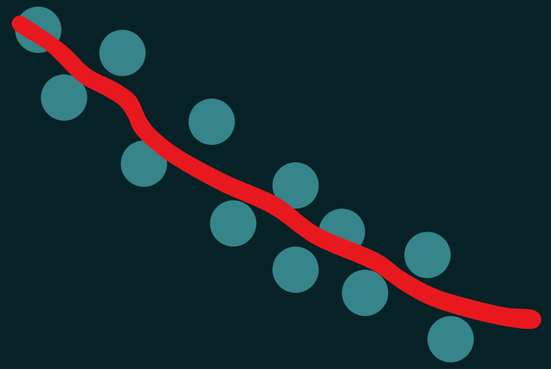
COST FUNCTIONS — Adding a penalty term in the form of a loss function, L1 or L2.

DATA AUGMENTATION — Modification of data, or addition of new data.

WHAT DOES IT DO?

Adds a level of complexity into th model, forcing the model to smoothen the solution.





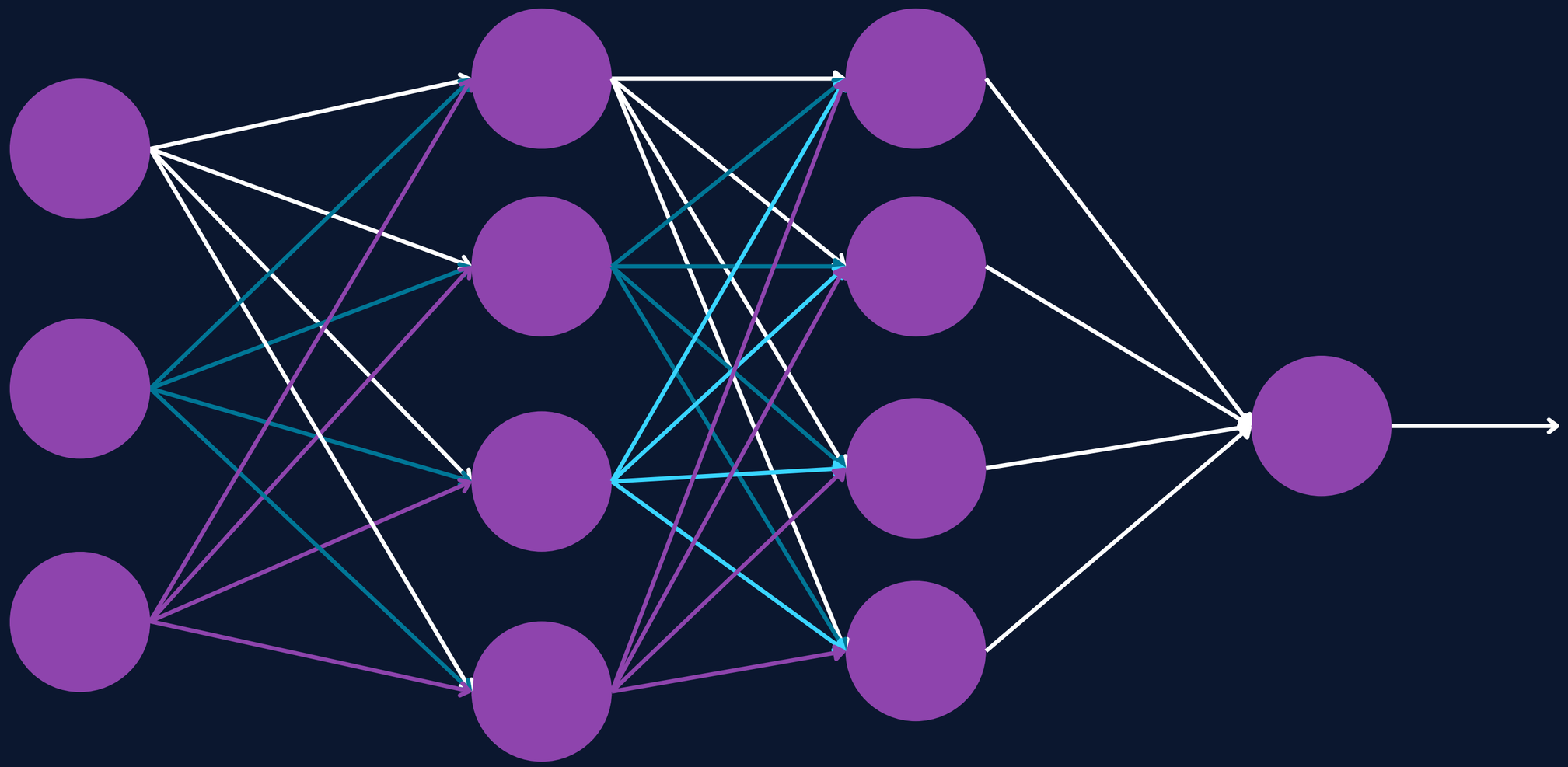
SOME PYTORCH NOTES

CAUTIONARY TALES

- Models built in Pytorch can switch between training mode and evaluation mode.
- We need to be aware when we need to use the .train() or the .eval(), and when to switch between the two.
- Gradients are only computed during training. (Why? What happens during training?)
- Regularization (drop-our and batch normalization) are applied only in the training phase, and not in the evaluation phase. (Why?)
- model.train(), training mode, regularization is active.
- model.eval(), evaluation mode, regularization is off. Only use if you are using regularization in place.
- model.no_grad(), only used in testing mode and switches off the gradient computation, the gradient is not computed and useful only for large models.

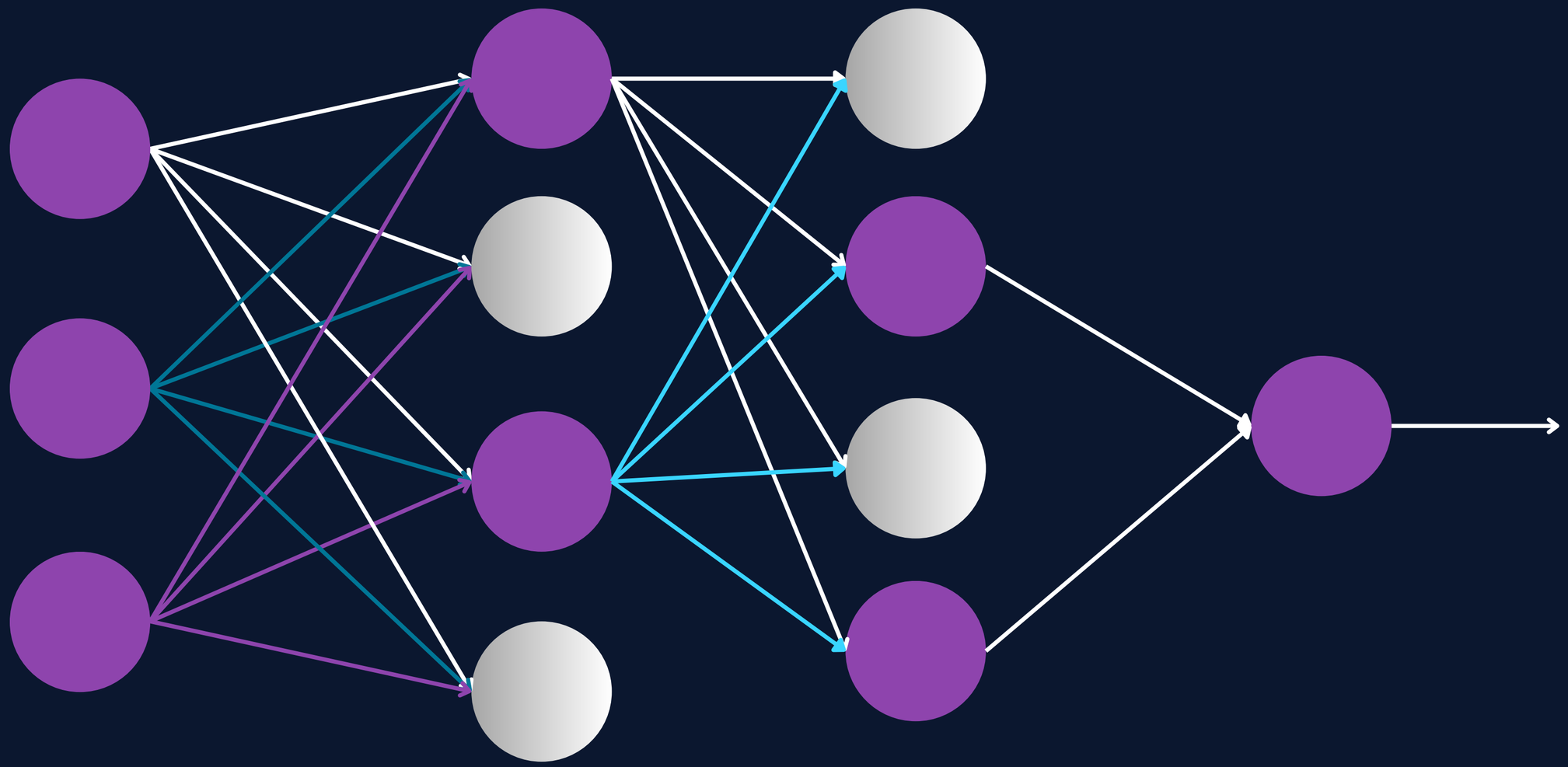
REGULARIZATION

DROP-OUT



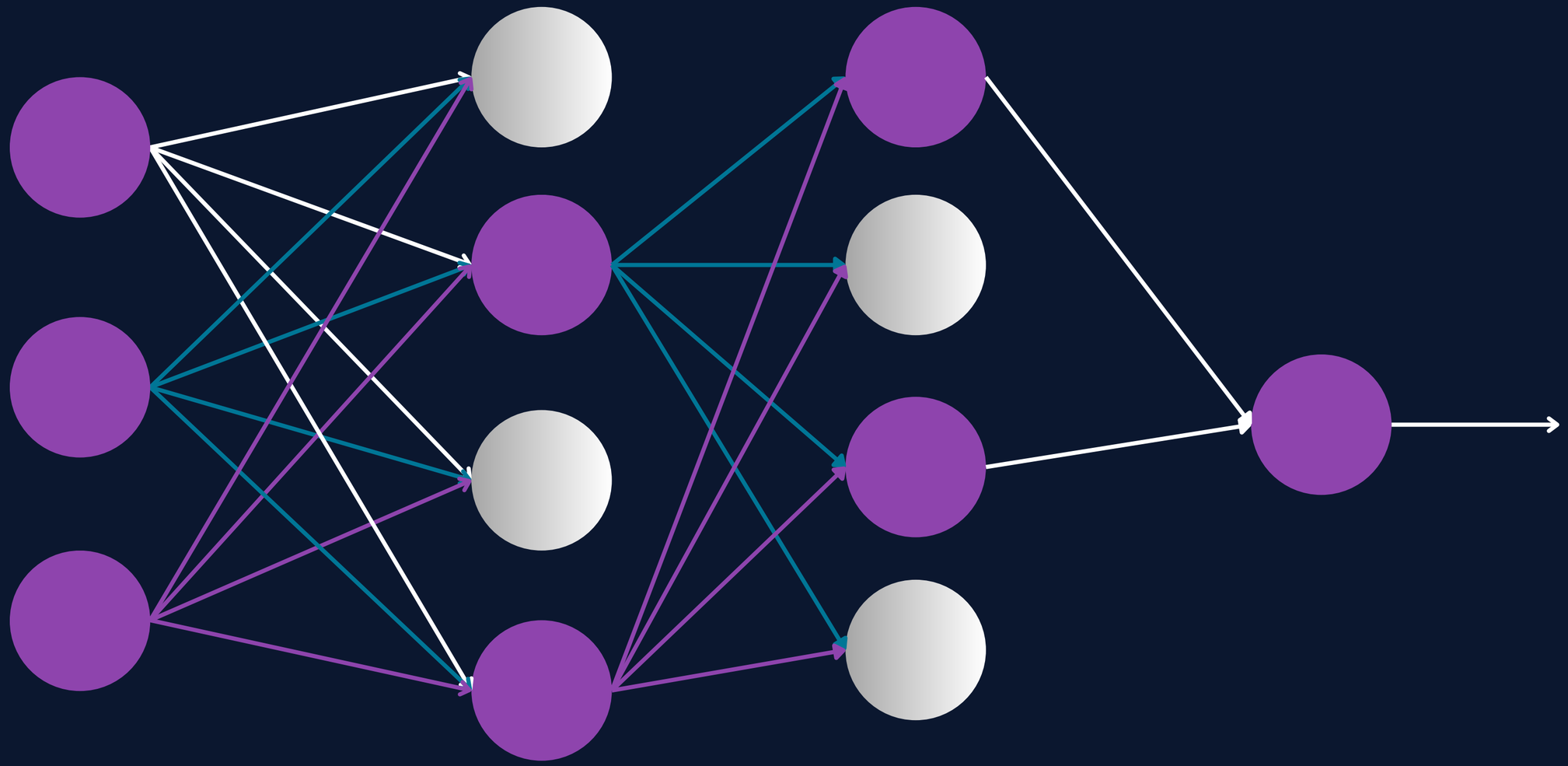
Prob. of Dropout, P = 0.5

Epoch 1:



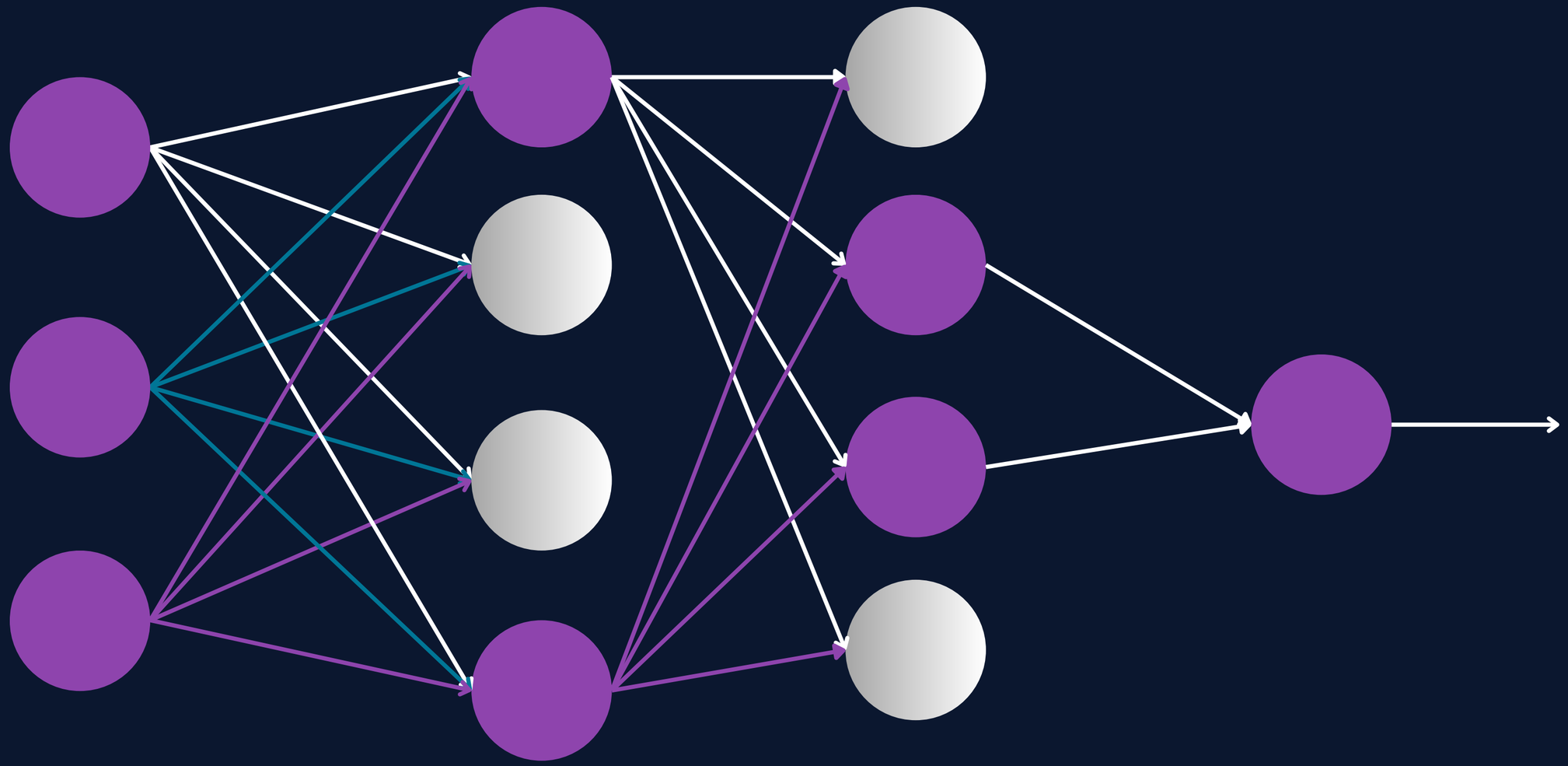
Prob. of Dropout, P = 0.5

Epoch 2:



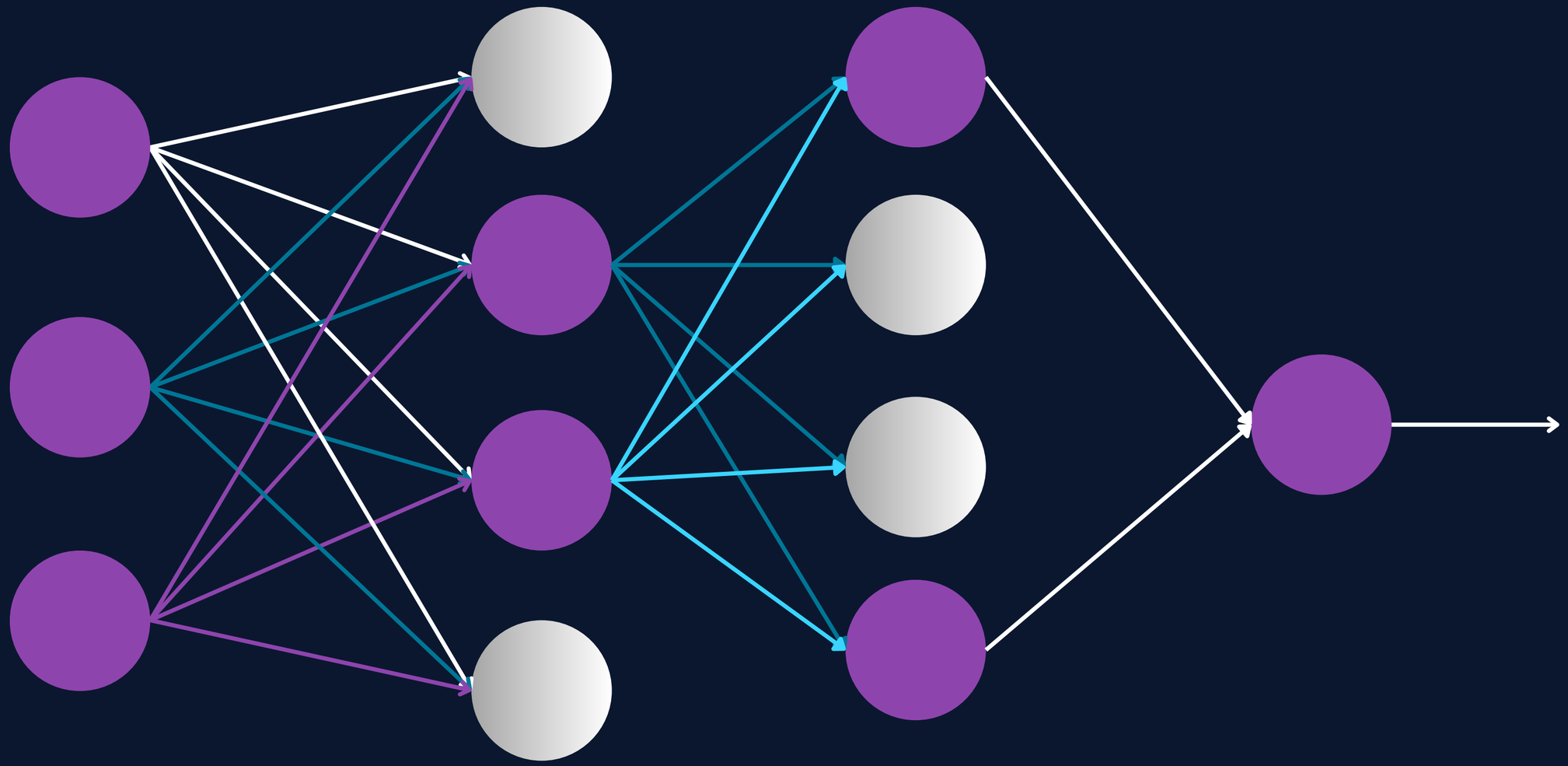
Prob. of Dropout, P = 0.5

Epoch 3:



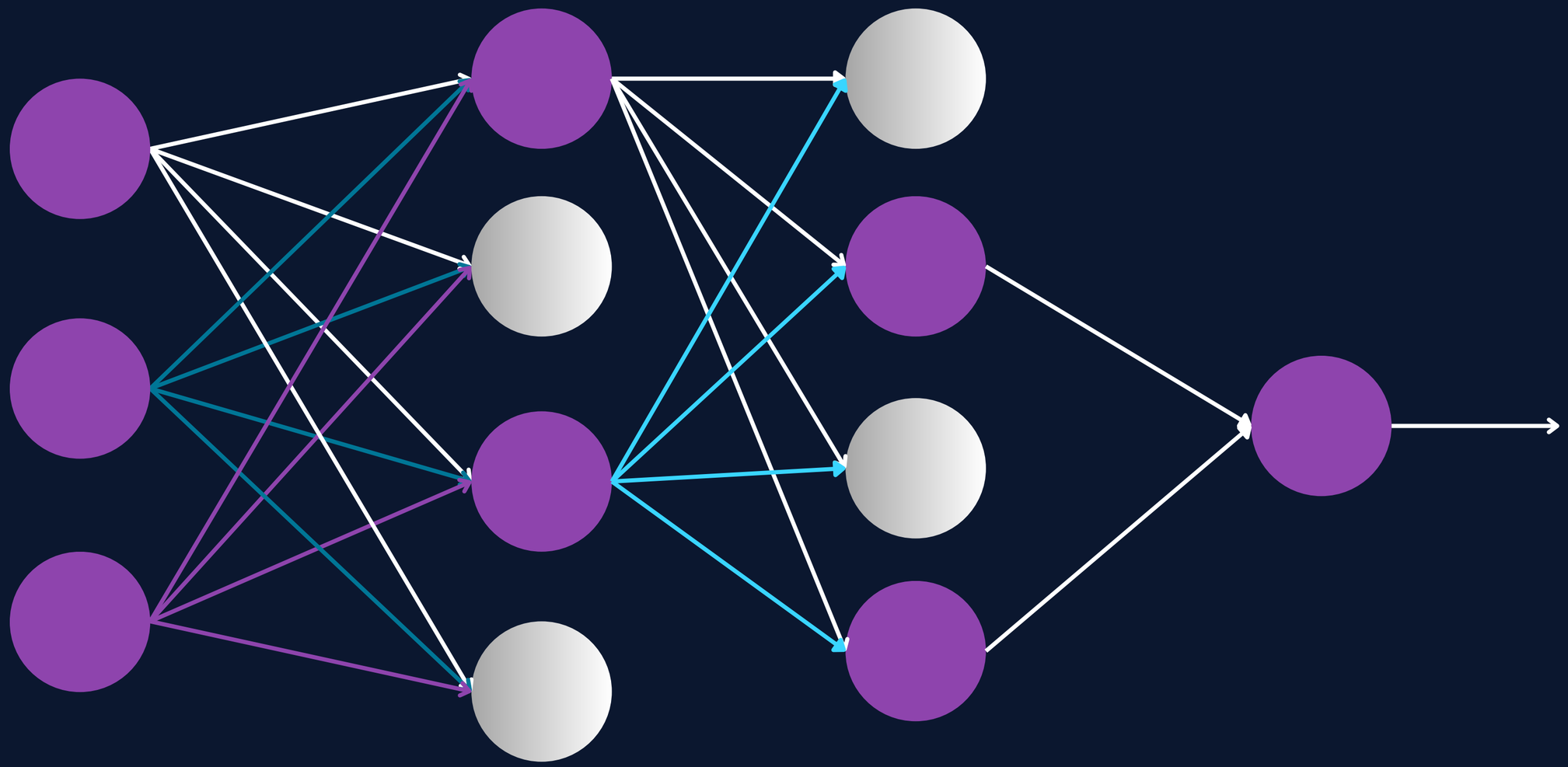
Prob. of Dropout, P = 0.5

Epoch 4:



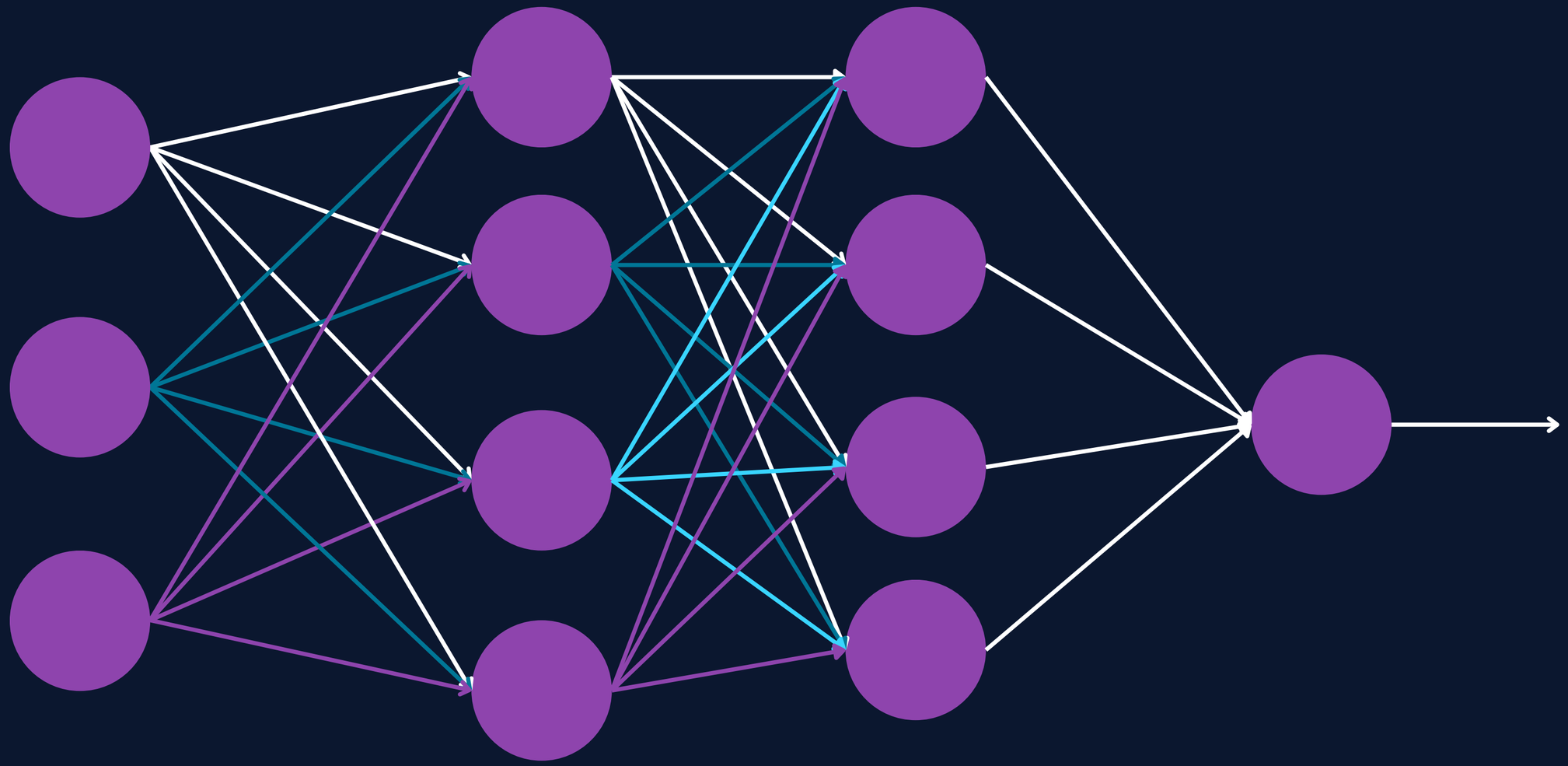
Prob. of Dropout, P = 0.5

Epoch 1:



Prob. of Dropout, P = 0.5

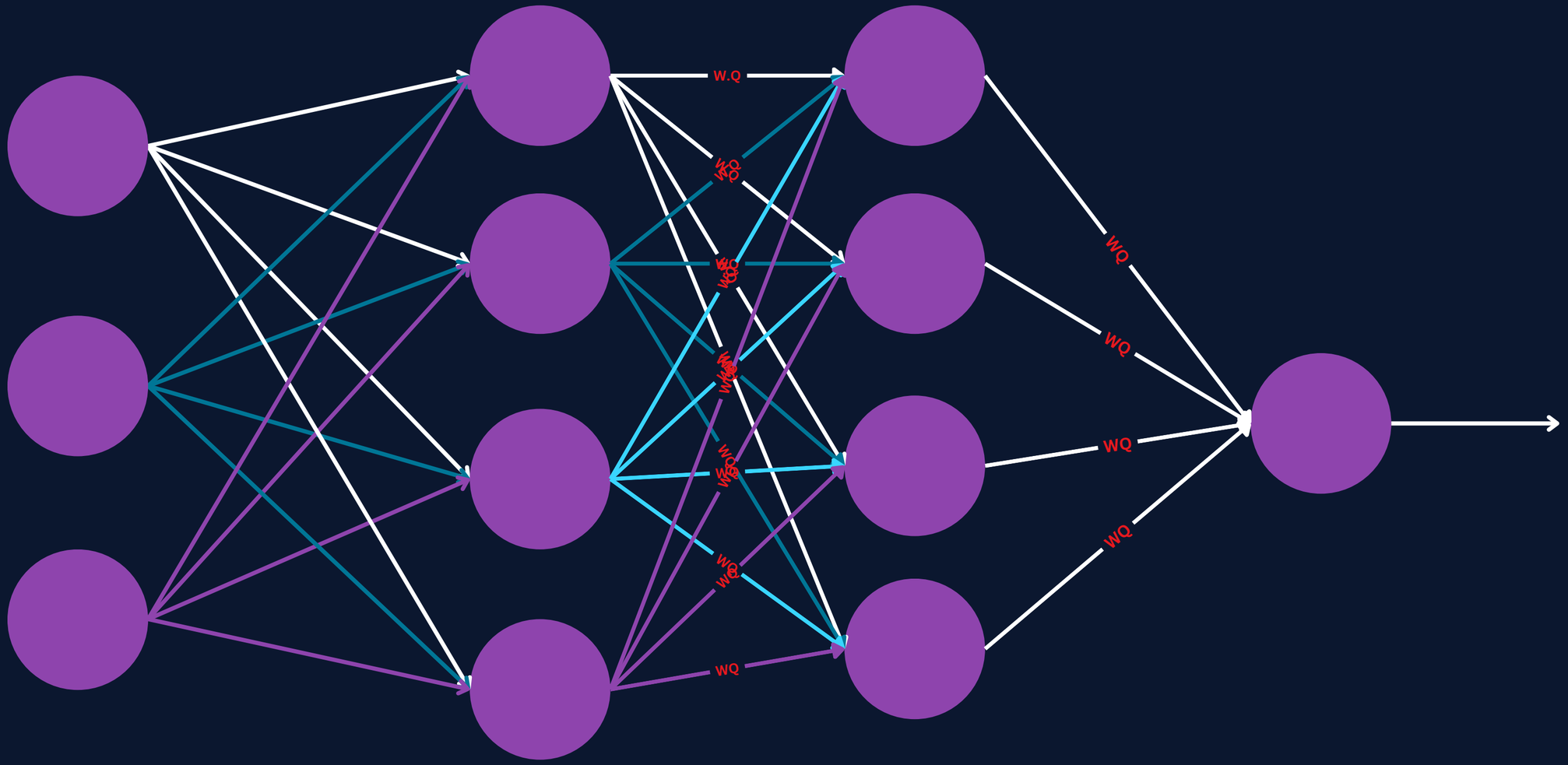
Evaluation Phase



Questions:

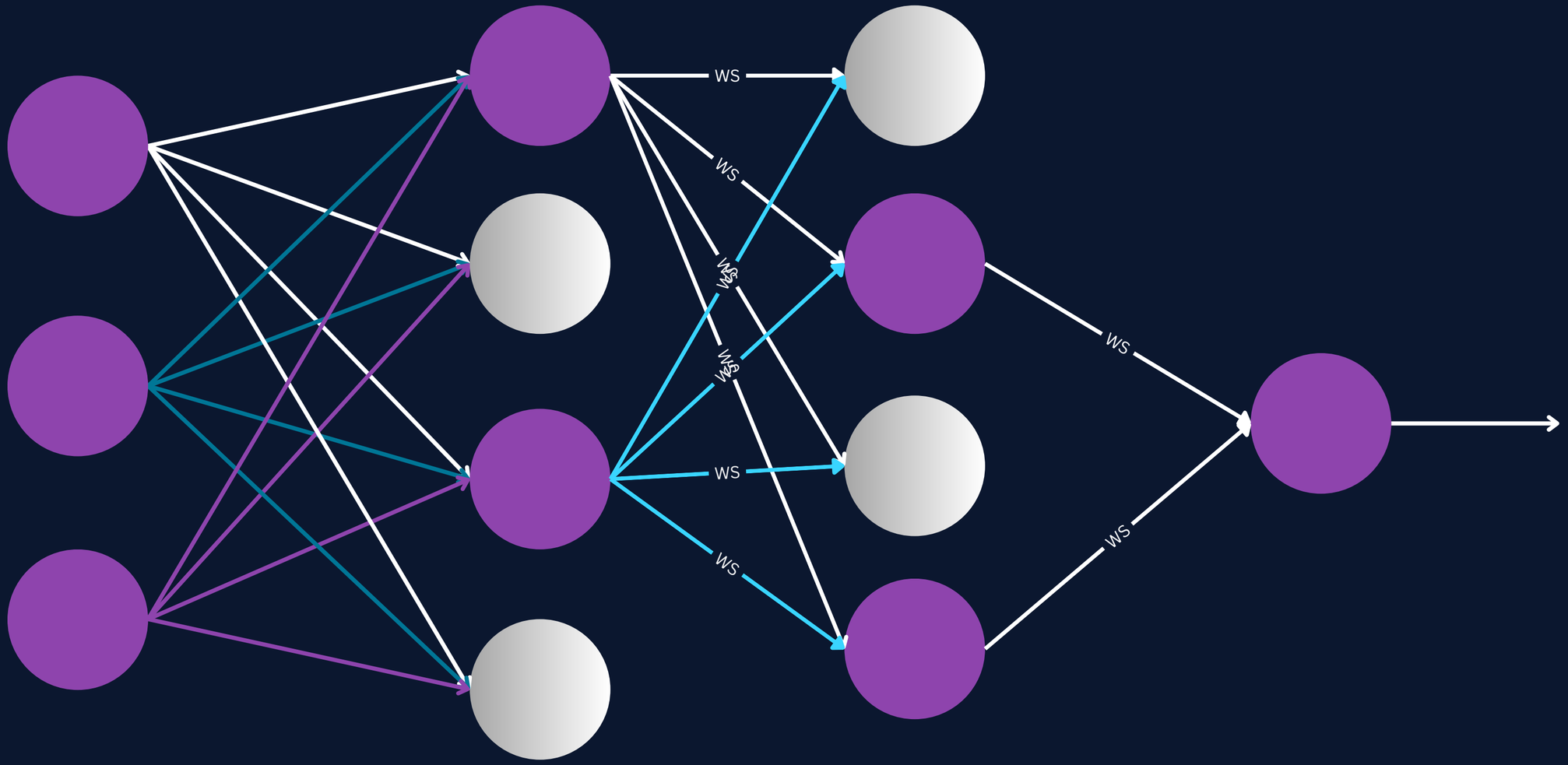
1. **Between the training and the evaluation phase, which of the two has a fewer numbers of active unit?**
2. **How does fewer activation unit affect the overall value of the activation function?**
3. **Think of an (engineering) way to fix with the issue of the difference between the active units between the training and the evaluation phase?**

Evaluation Phase: Solution 1



$$Q=1-P$$

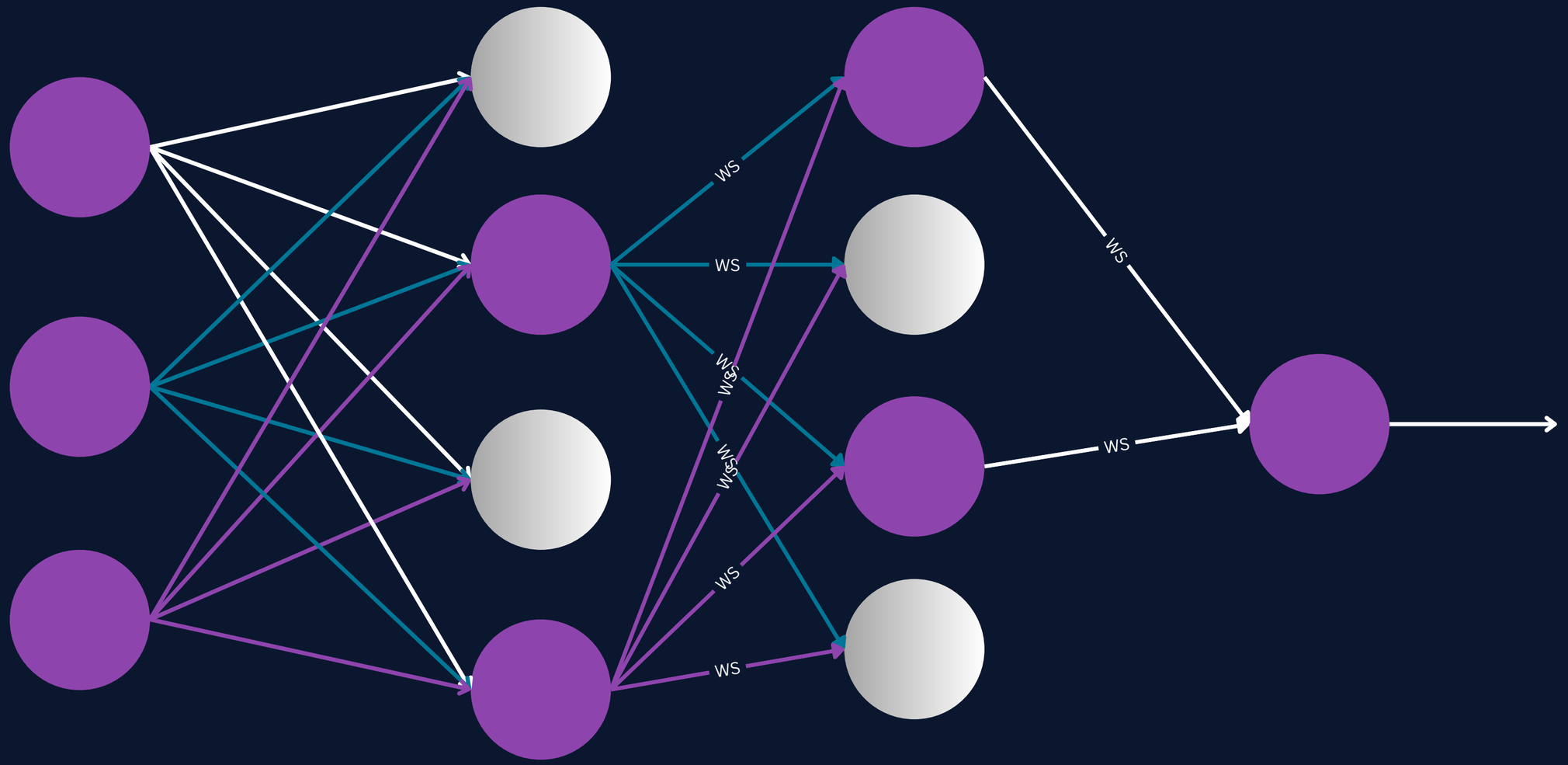
Solution 2: Epoch 1



Prob. of Dropout, P = 0.5

$$S = P^{-1}$$

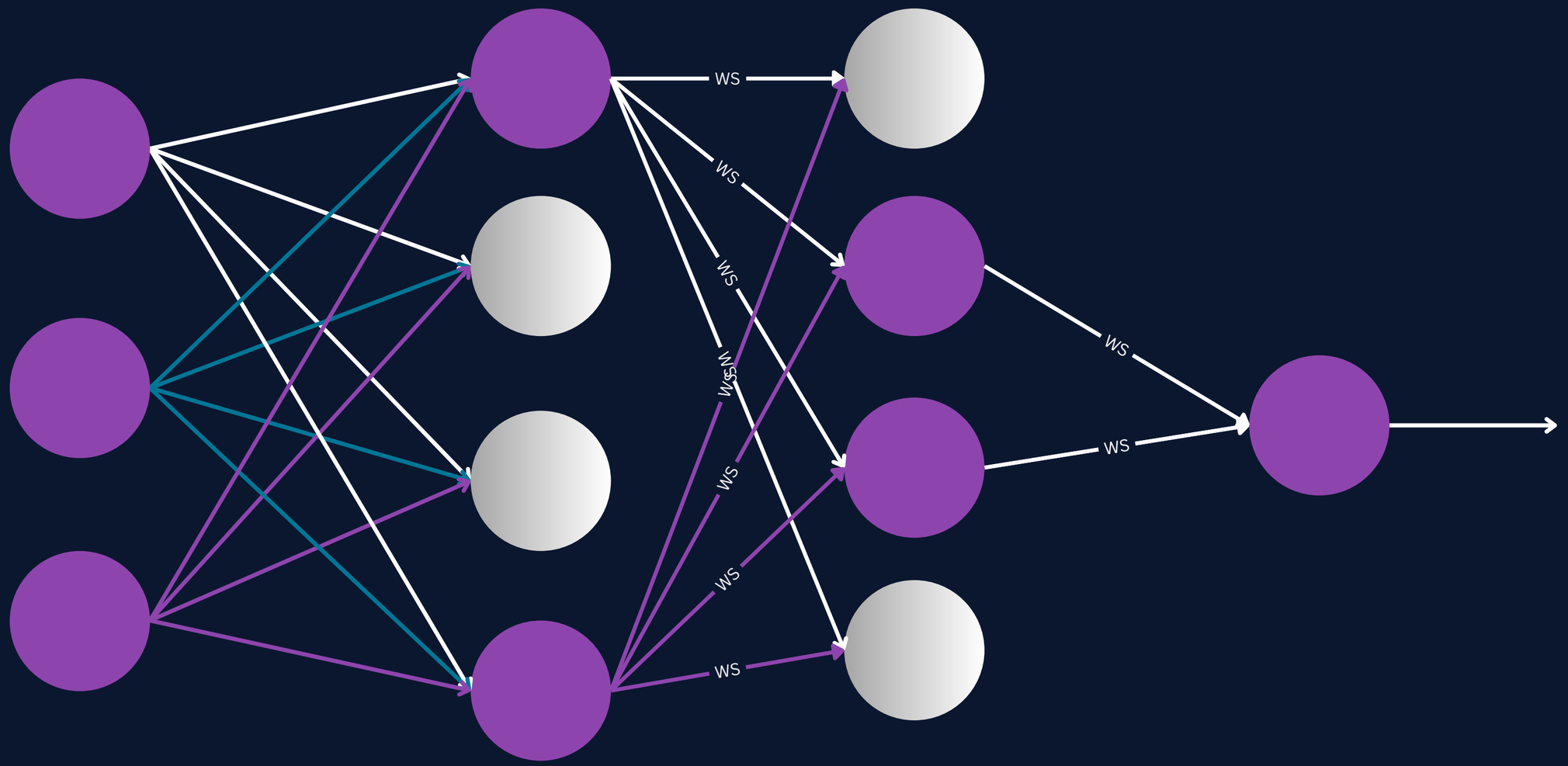
Epoch 2:



Prob. of Dropout, P = 0.5

$$S = P^{-1}$$

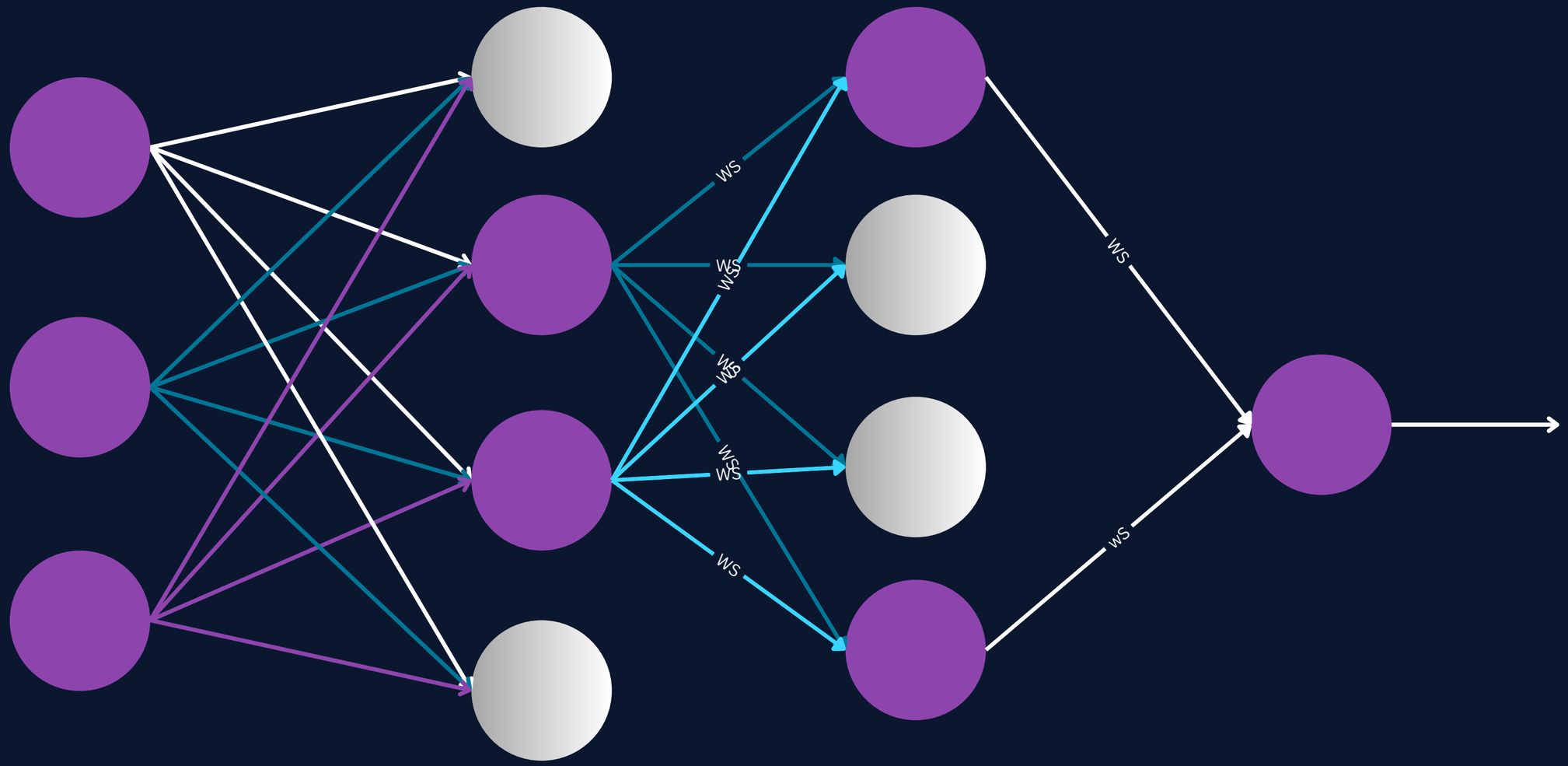
Epoch 3:



Prob. of Dropout, P = 0.5

$$S = P^{-1}$$

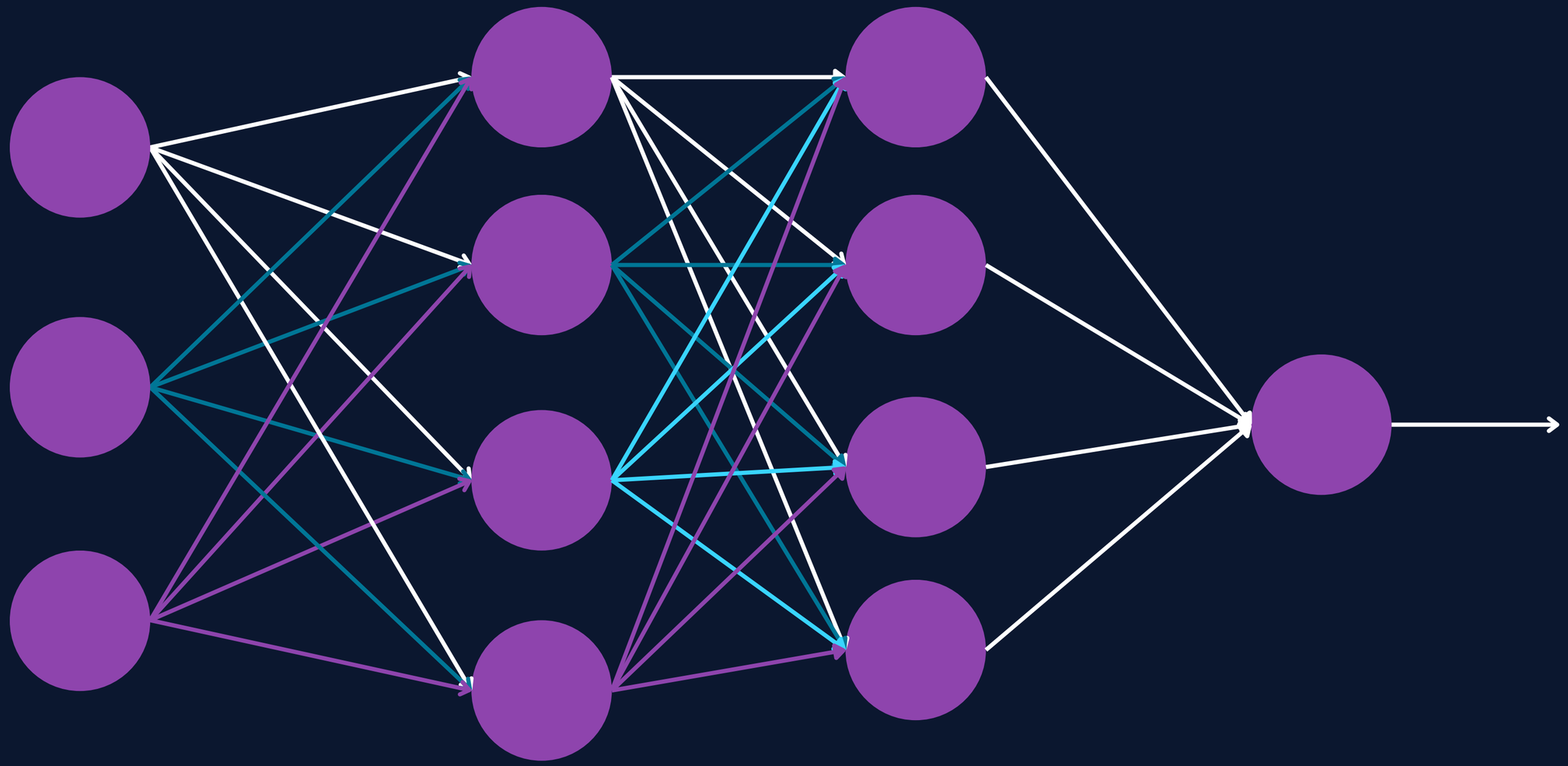
Epoch 4:



Prob. of Dropout, P = 0.5

$$S = P^{-1}$$

If Solution 2 is applied then no need to recalibrate Evaluation Phase



WHY DOES IT WORK?

TRAINNING

Prevents a single node from learning too much information (representation).

LEARNING

Forces the distribution of different information to different units in different layers.

STABILITY

Since information is distributed, the model does not rely on a few sets of nodes, and is more stable.

JUPYTER NOTEBOOK DISCUSSION

REGULARIZATION

L1/L2 REGULARIZATION

L1 = Lasso Regularization
L2 = Ridge Regression
W = Matrix of all trainable parameters
f = Cost function
L = Loss Function

$$W = \underset{w}{argmin}\,(f)$$

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right)$$

Weight Regularization

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right) + Penalty$$

L1 (Lasso) Regularization

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right) + \theta\|\omega\|$$

L2 (Ridge) Regularization

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right) + \theta\|\omega\|_2^2$$

Question:

1. **Which is ideal: Loss > Penalty, or, Loss < Penalty? Why?**
2. **How do you force the Penalty to be smaller than the Loss?**

L1 (Lasso) Regularization

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right) + \theta\|\omega\|$$

What is this term in Linear Algebra?

L1 (Lasso) Regularization

$$\|\omega\| = \sum_{i=1}^{n} |w_i| \quad \longleftrightarrow \quad \theta\|\omega\| = \theta \sum_{i=1}^{n} |w_i|$$

Question:

What does $\theta\|\omega\|$ do to the entire penalty term?

L1 (Lasso) Regularization

$$\|\omega\| = \sum_{i=1}^{n} |w_i| \quad \longleftrightarrow \quad \theta\|\omega\| = \theta \sum_{i=1}^{n} |w_i|$$

Question:

What does $\theta\|\omega\|$ do to the entire penalty term?

Ans: The lowest possible value for $\theta\|\omega\|$
is zero, hence it creates a sparsity value in W

L1 (Lasso) Regularization: Sparsity

$$\|\omega\| = \sum_{i=1}^{n} |w_i| \quad \longleftrightarrow \quad \theta\|\omega\| = \theta \sum_{i=1}^{n} |w_i|$$

Question:

What does $\theta\|\omega\|$ do to the entire penalty term?

Ans: The lowest possible value for $\theta\|\omega\|$
is zero, hence it creates a sparsity value in W

L2 (Ridge) Regularization

$$f = \frac{1}{n}\sum_{i=1}^{n} L\left(y', y\right) + \theta\|\omega\|_2^2$$

What is this term in Linear Algebra?

L2 (Ridge) Regularization

$$\|\omega\|_p^n = \sum_{i=1}^{n} \left(|w_i|^p\right)^{\frac{n}{p}}$$

$$\|\omega\|_2^2 = \sum_{i=1}^{n} \left(|w_i|^2\right)^{\frac{2}{2}}$$

Question: What does the term $\theta\|\omega\|_2^2$ do?

L2 (Ridge) Regularization

$$\|\omega\|_p^n = \sum_{i=1}^{n} \left(|w_i|^p\right)^{\frac{n}{p}}$$

$$\|\omega\|_2^2 = \sum_{i=1}^{n} \left(|w_i|^2\right)^{\frac{2}{2}}$$

Question: What does the term $\theta\|\omega\|_2^2$ do?

Answer:

If ω_i is small then $\theta\|\omega\|_2^2$ is also small hence $W = \underset{w}{argmin}\,(f)$ will only try to focus on minimizing $\frac{1}{n}\sum_{i=1}^{n} L\,(y', y)$

L2 (Ridge) Regularization

$$\|\omega\|_p^n = \sum_{i=1}^{n} \left(|w_i|^p\right)^{\frac{n}{p}}$$

$$\|\omega\|_2^2 = \sum_{i=1}^{n} \left(|w_i|^2\right)^{\frac{2}{2}}$$

Question: What does the term $\theta\|\omega\|_2^2$ do?

Answer:

If ω_i is large then $\theta\|\omega\|_2^2$ is also large hence $W = \underset{w}{argmin}\,(f)$ will focus on minimizing the entire: $\frac{1}{n}\sum_{i=1}^{n} L\,(y', y) + \theta\|\omega\|_2^2$

L2 (Ridge) Regularization: Weight Decay

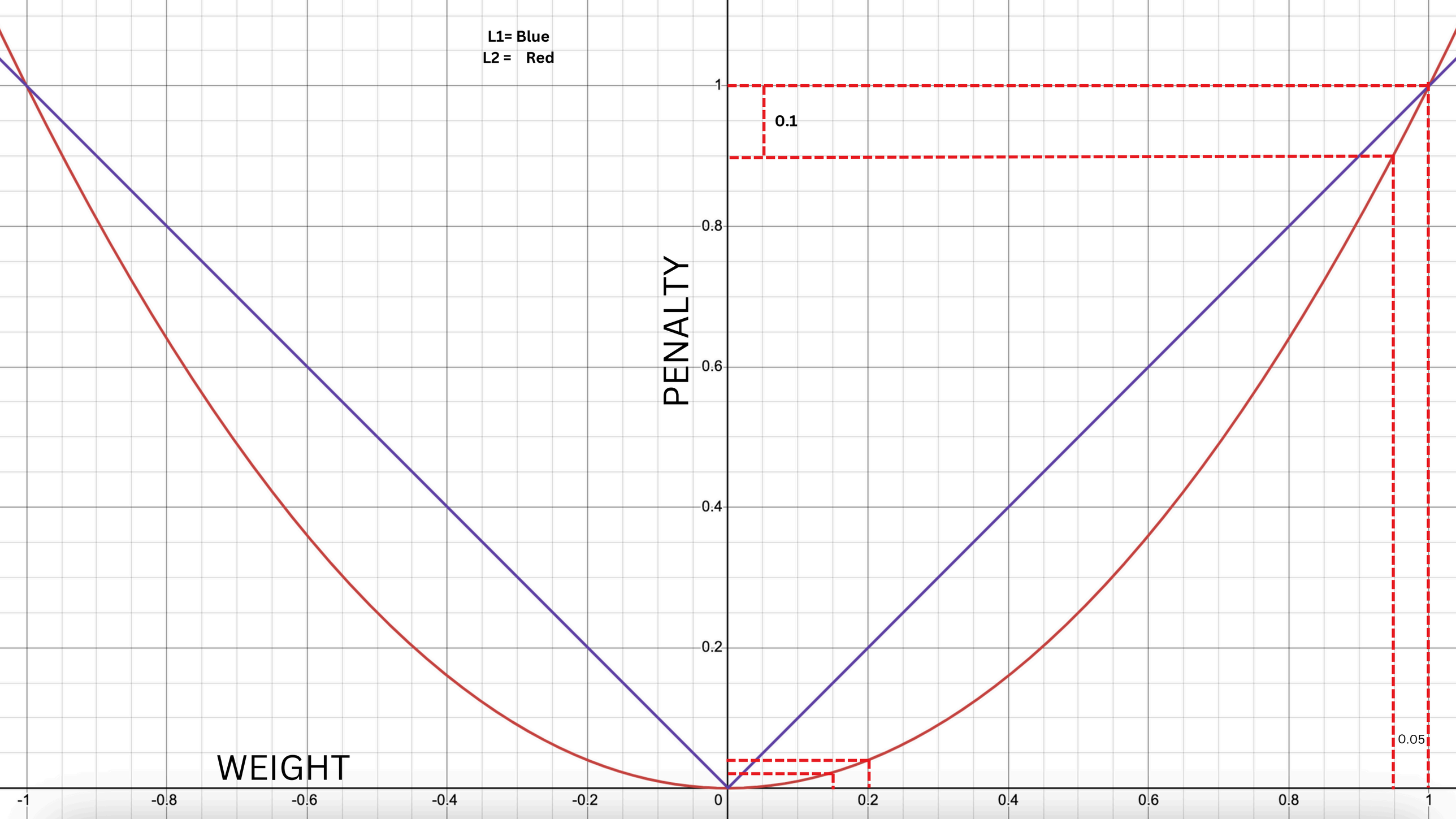
$$\|\omega\|_p^n = \sum_{i=1}^{n} \left(|w_i|^p\right)^{\frac{n}{p}}$$

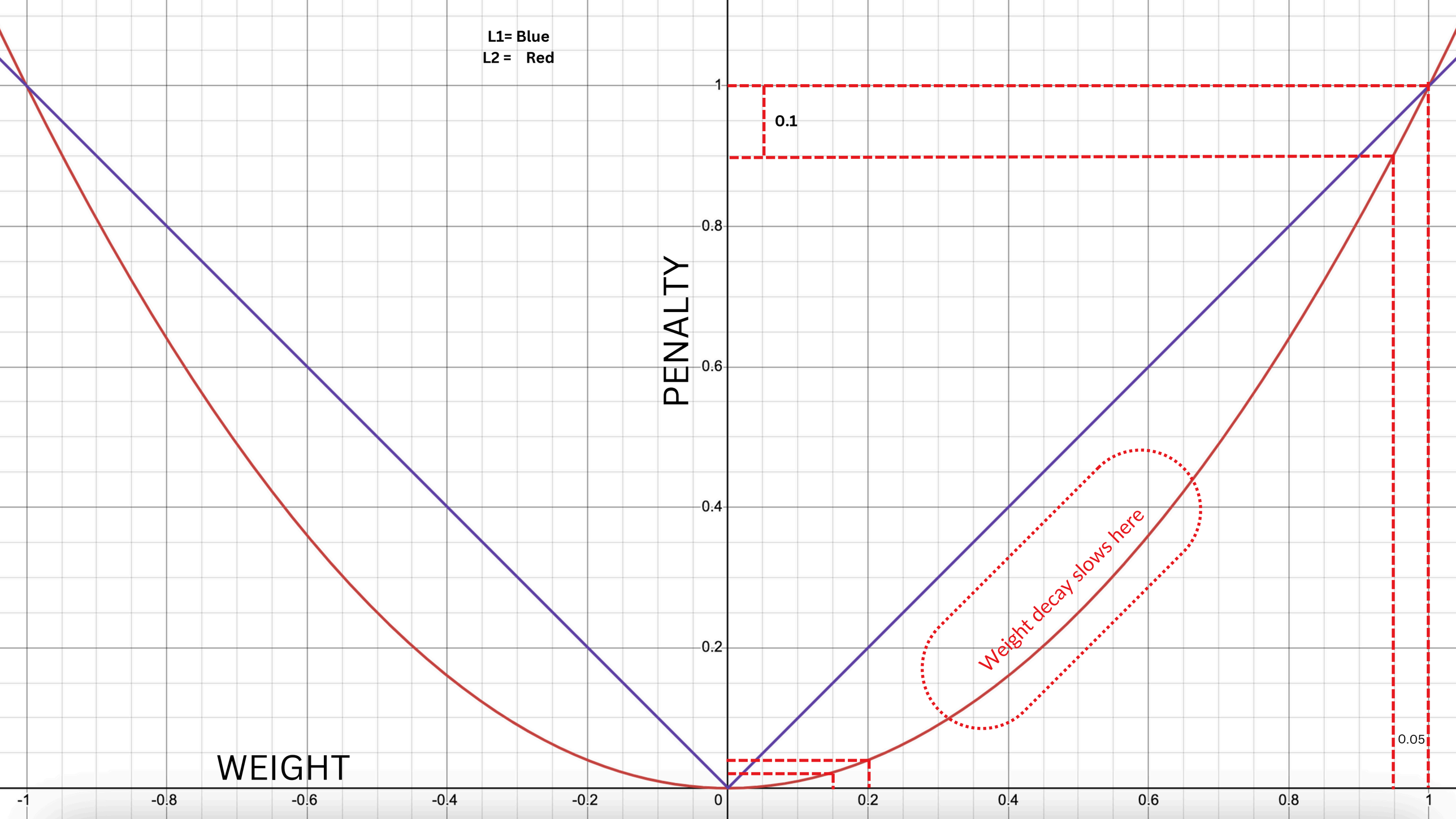
$$\|\omega\|_2^2 = \sum_{i=1}^{n} \left(|w_i|^2\right)^{\frac{2}{2}}$$

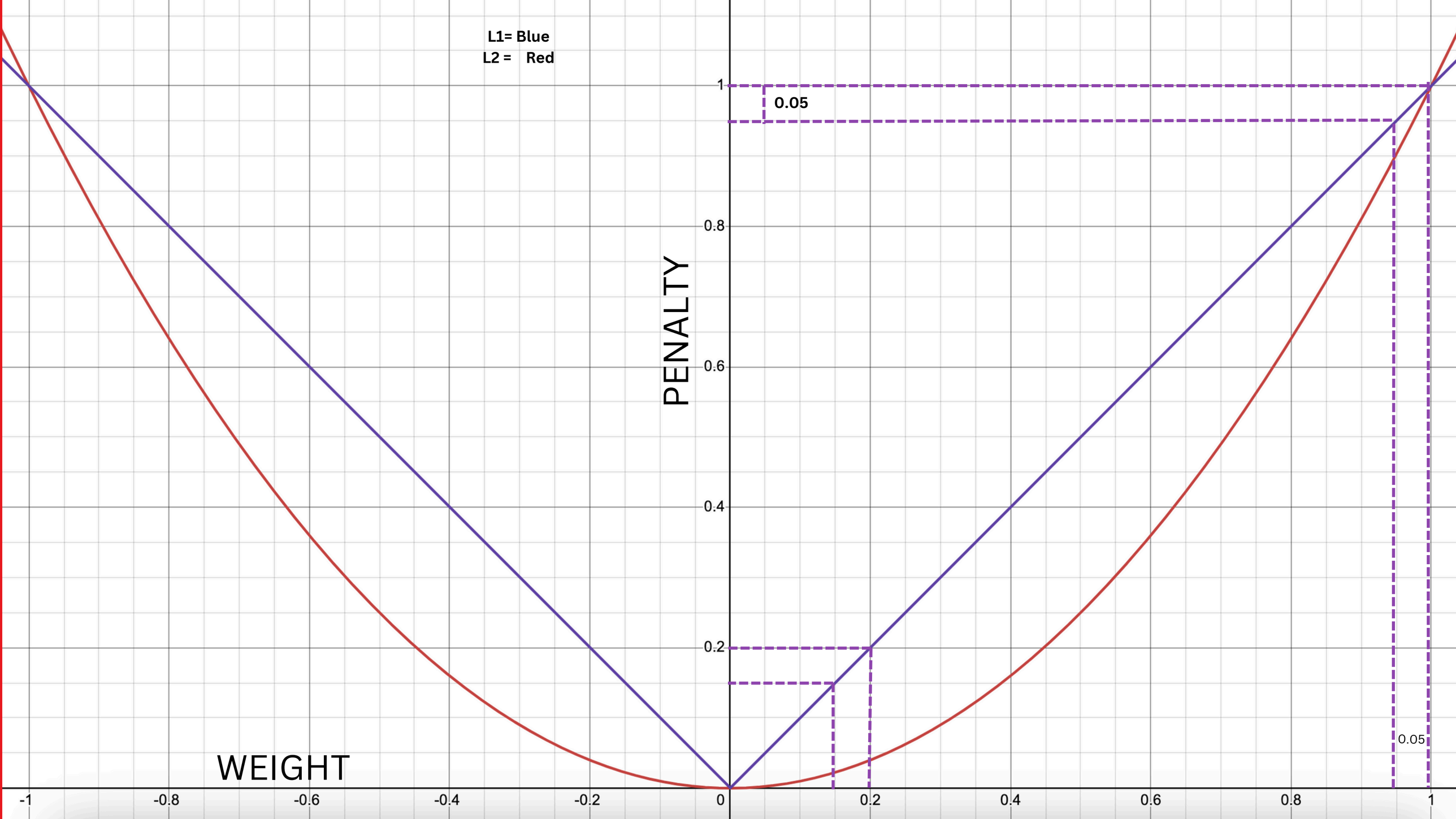
Question: What does the term $\theta\|\omega\|_2^2$ do?

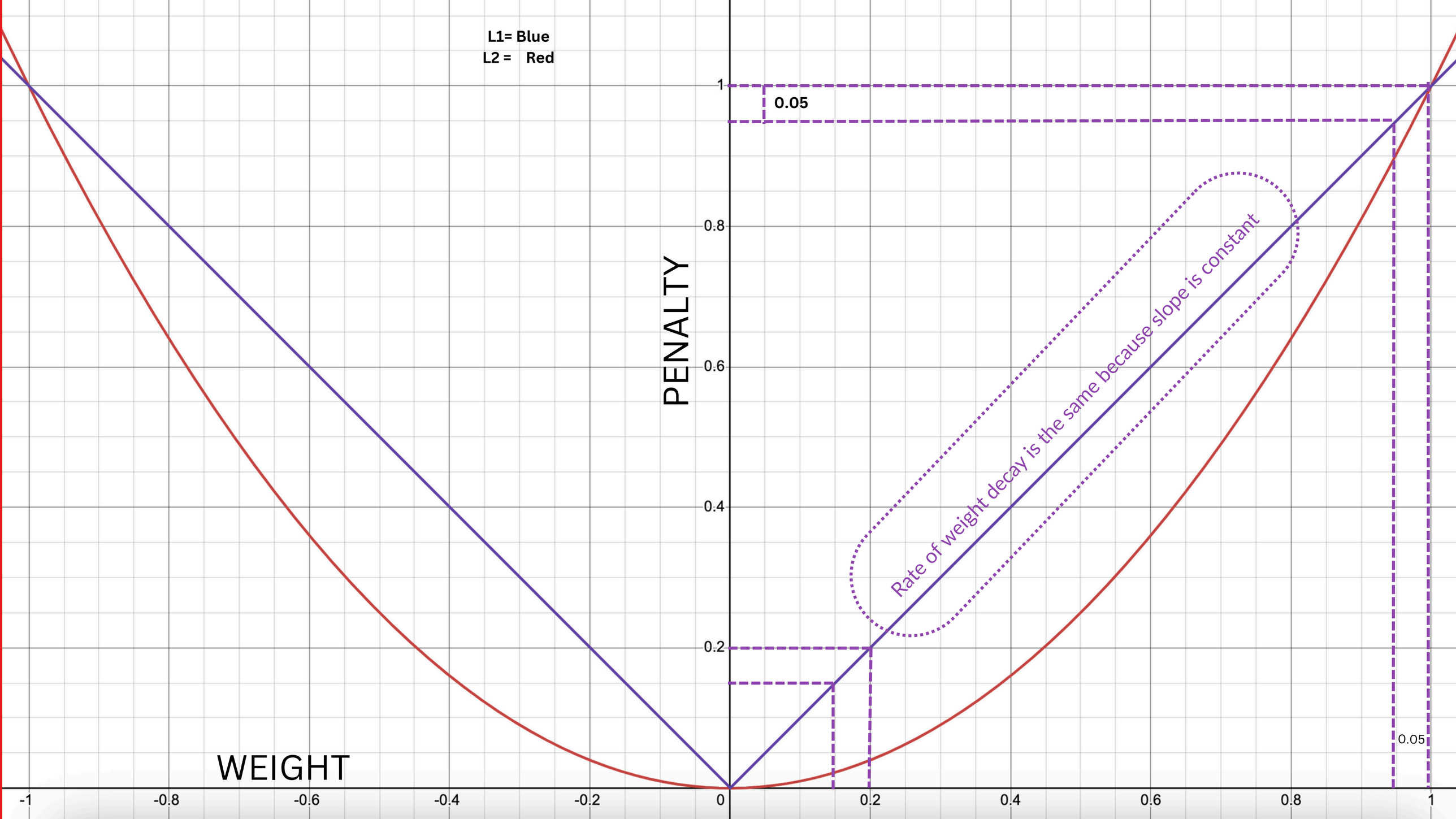
Answer:

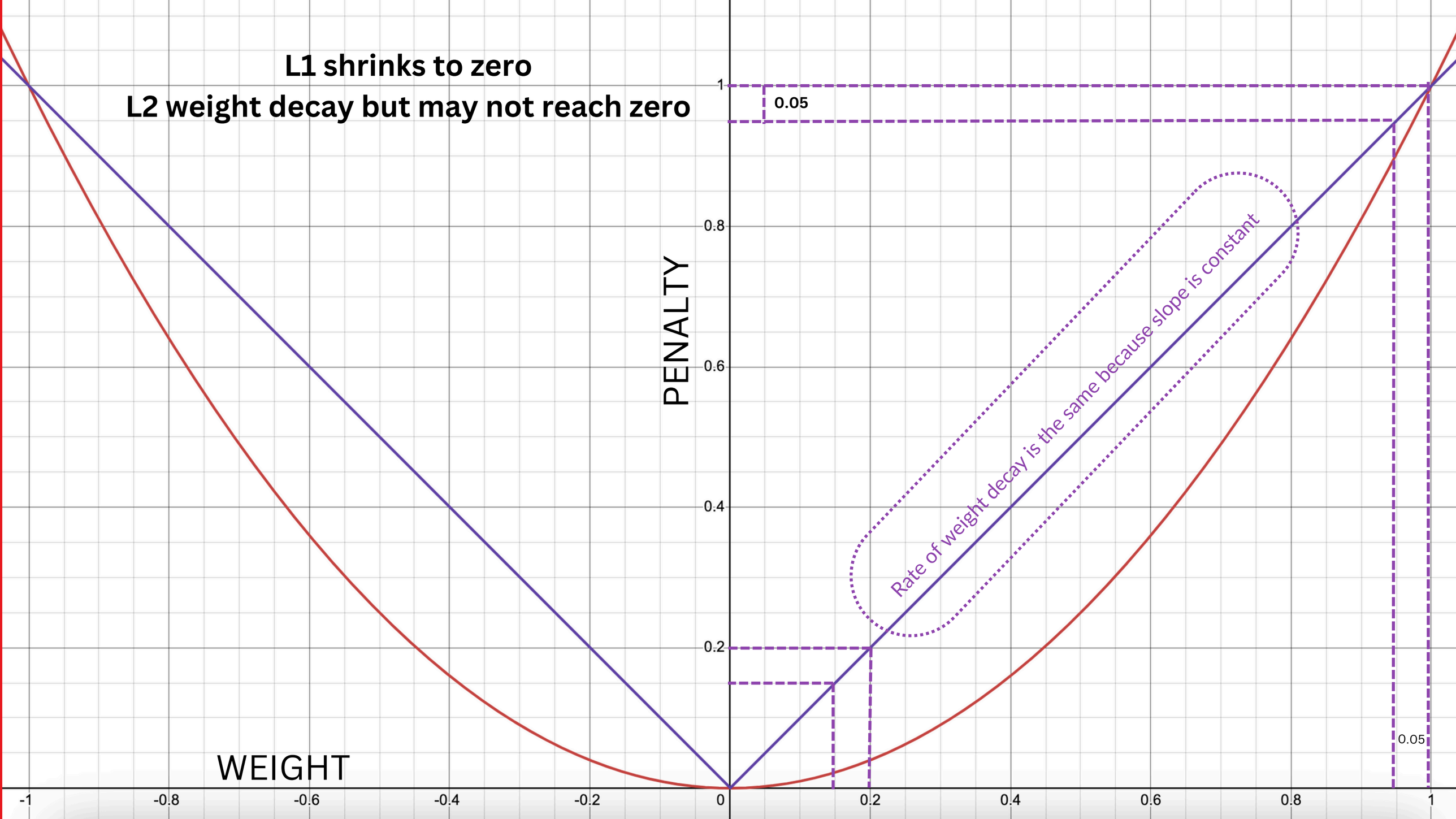
If ω_i is large then $\theta\|\omega\|_2^2$ is also large hence $W = \underset{w}{argmin}\,(f)$ will focus on minimizing the entire: $\frac{1}{n}\sum_{i=1}^{n} L\,(y', y) + \theta\|\omega\|_2^2$


L1= Blue
L2 = Red
PENALTY
WEIGHT
0.1
0.05
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1


L1= Blue
L2 = Red
PENALTY
WEIGHT
0.1
0.05
Weight decay slows here
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1


L1= Blue
L2 = Red
0.05
PENALTY
WEIGHT
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1
0.05


L1= Blue
L2 = Red
PENALTY
WEIGHT
0.05
Rate of weight decay is the same because slope is constant
0.05
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1


L1 shrinks to zero
L2 weight decay but may not reach zero
PENALTY
WEIGHT
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1
0.05
0.05
Rate of weight decay is the same because slope is constant

WHY DOES IT WORK?

TRAINNING

Prevents nodes to learn from sample specific information.

LEARNING

Prevents overfitting.

STABILITY

Large weights leads to instability, penalizing large weights will solve the issue.

When to use L1/L2 Regularization:

1. **For large complex architecture with many weights.**
2. **When training accuracy is very high compared to testing accuracy.**
3. **L1 for high dimensional data.**
4. **L2 for data with high collinearity.**
5. **If $|w|$ is the number of weights then,**

$$\lambda = \frac{\alpha}{2 \times |w|}$$

REGULARIZATION

BATCH REGULARIZATION

FULL BATCH

What happens if you use the entire dataset?

- Forward Propagation
- Loss Calculation
- Back Propagation
- Updating of weights and biases

Mini-Batch Normalization

Use each individual k’s to train and test the neural networks

Rule of Thumbs:

1. **Use k between 2 and 512.**
2. **Lower k decreases computation time.**
3. **Batching is a form of regularization by averaging the loss over many samples preventing over fitting. (Question: Does it smoothen the loss curve or not?).**

THANK YOU!

A presentation is a method of sharing information, ideas, or arguments with an audience using spoken words and visual aids. It typically involves a speaker conveying content to inform, persuade, or entertain the listeners.

www.linkedin.com/in/gerard-ompad

gerard.ompad@gmail.com

https://github.com/gerard-ompad