Google Cloud

The Science of Neural Networks

Ryan Gillard

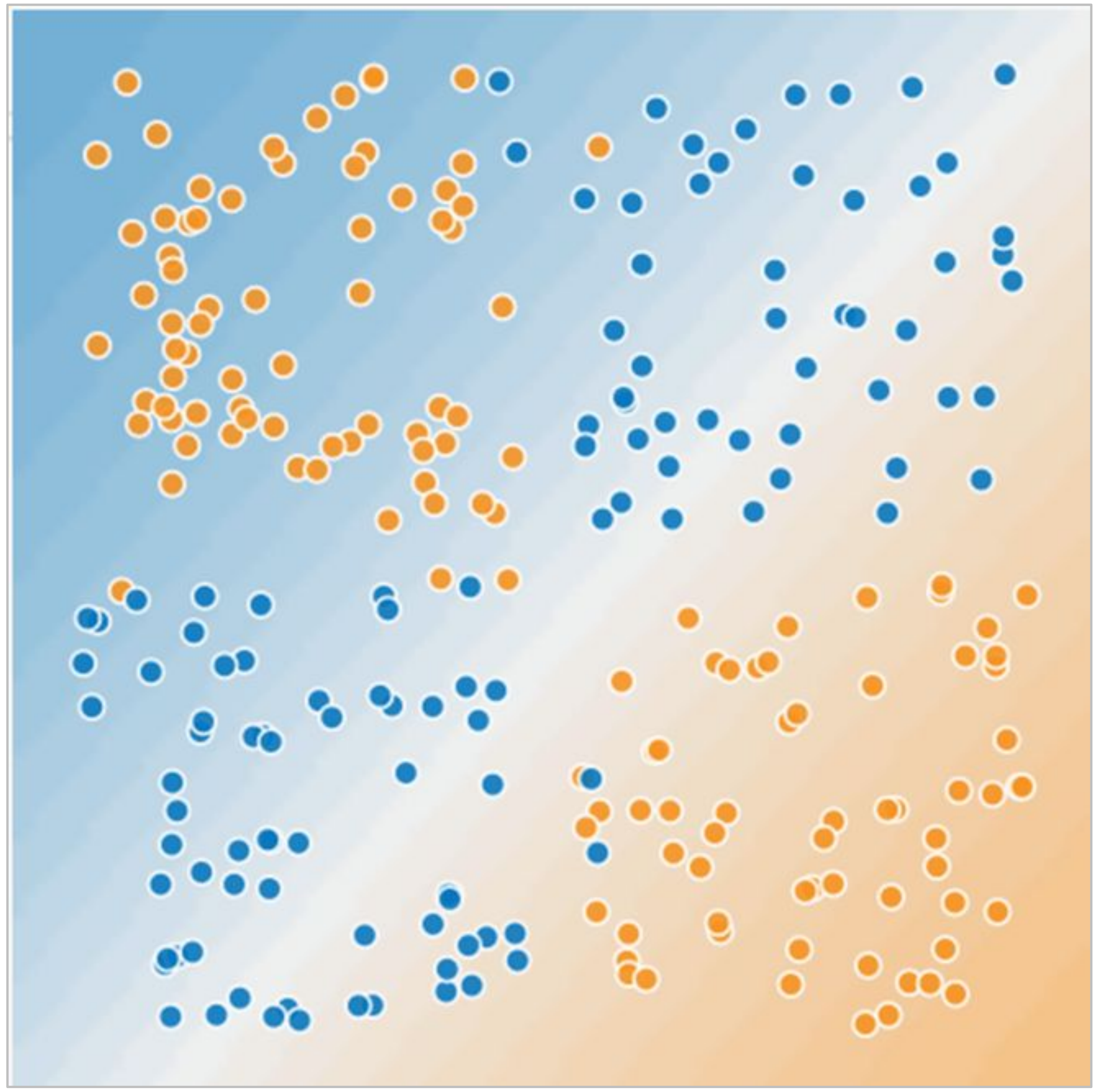
Machine Learning on
Google Cloud Platform

Google Cloud

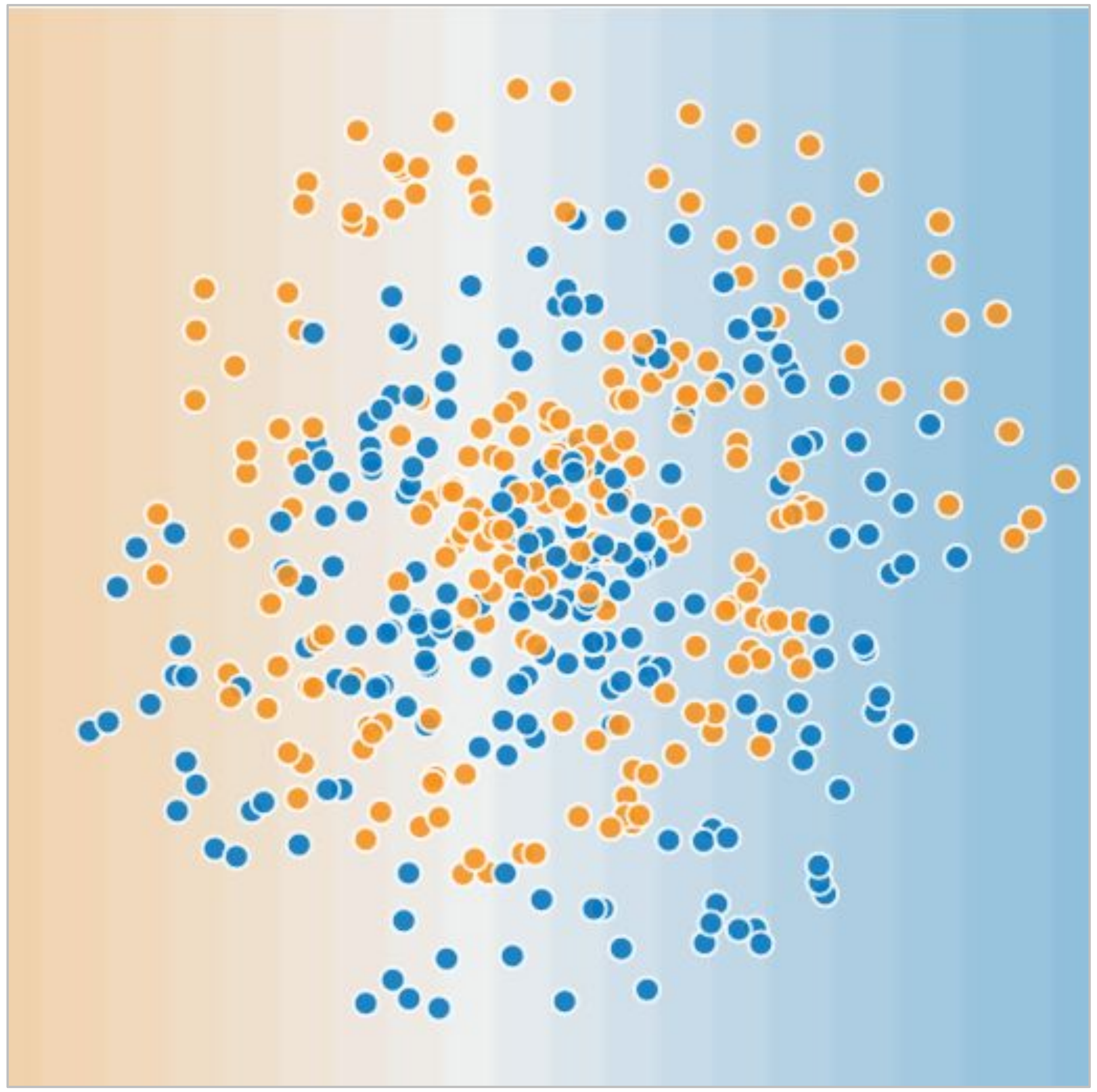
Introduction to Neural Networks

Ryan Gillard

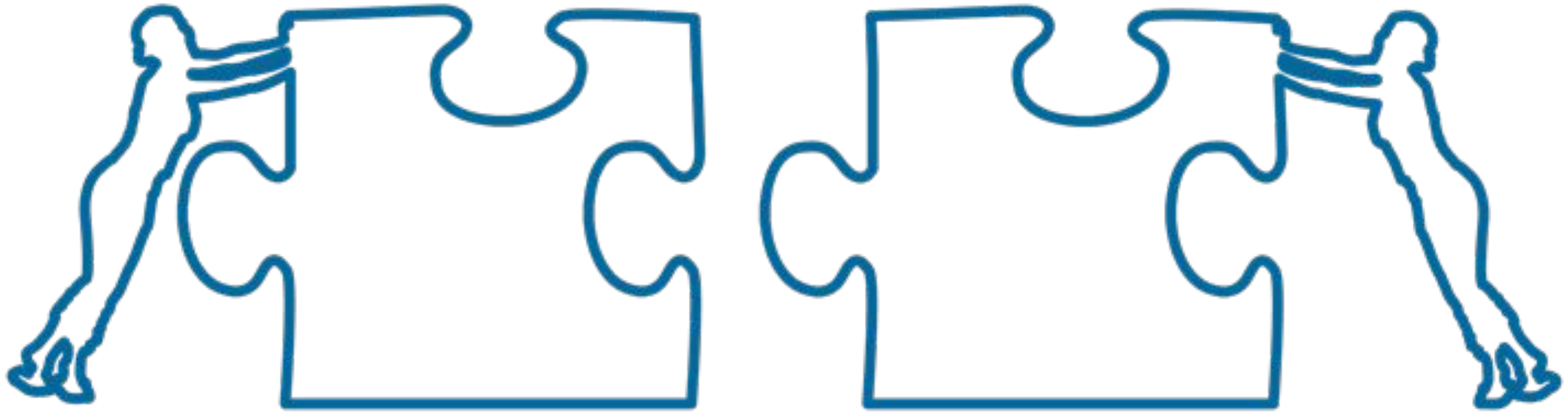
Feature crosses help linear models work in nonlinear problems



But there tends to be a limit ...



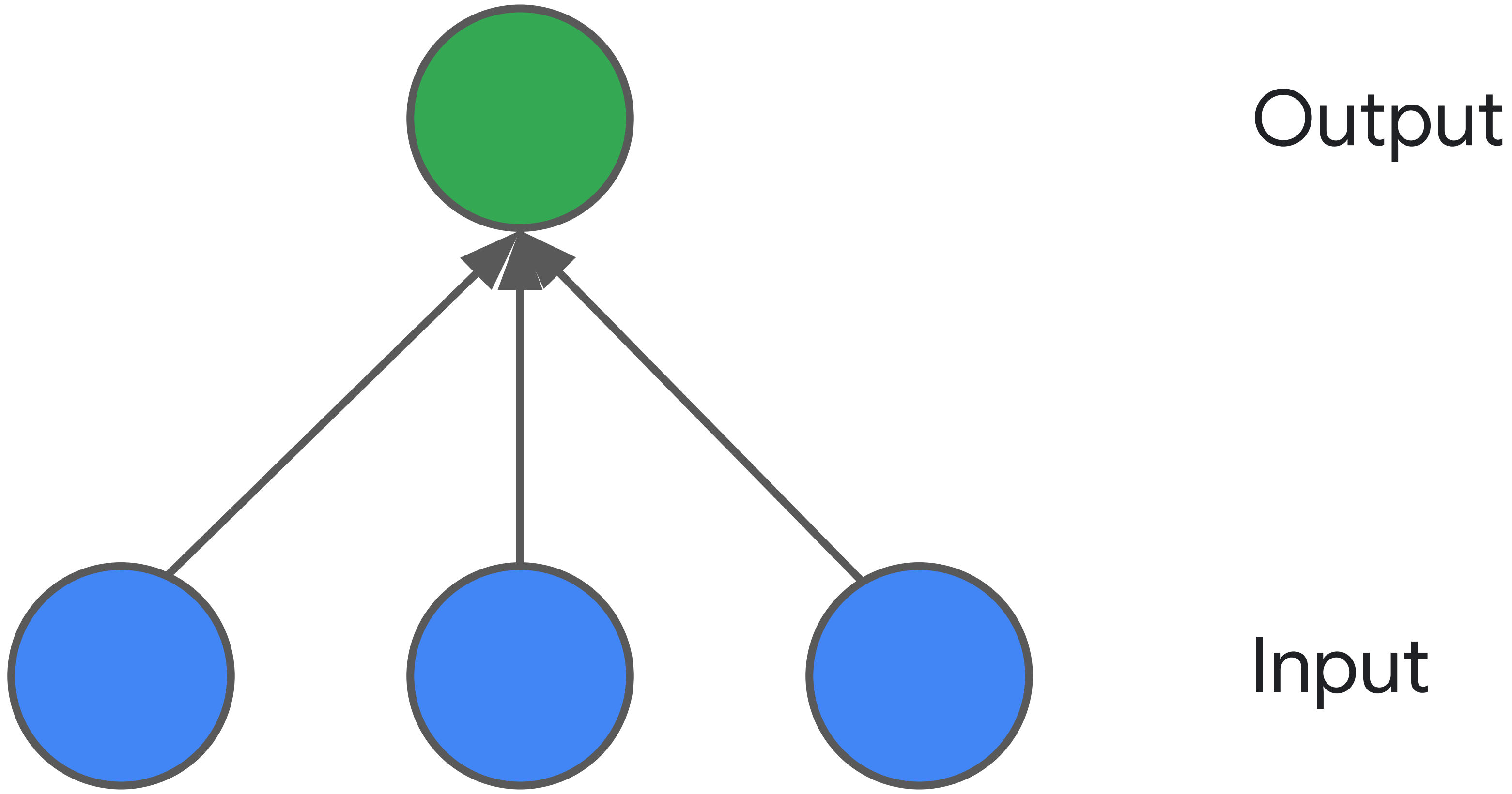
Combine features as an alternative to feature crossing



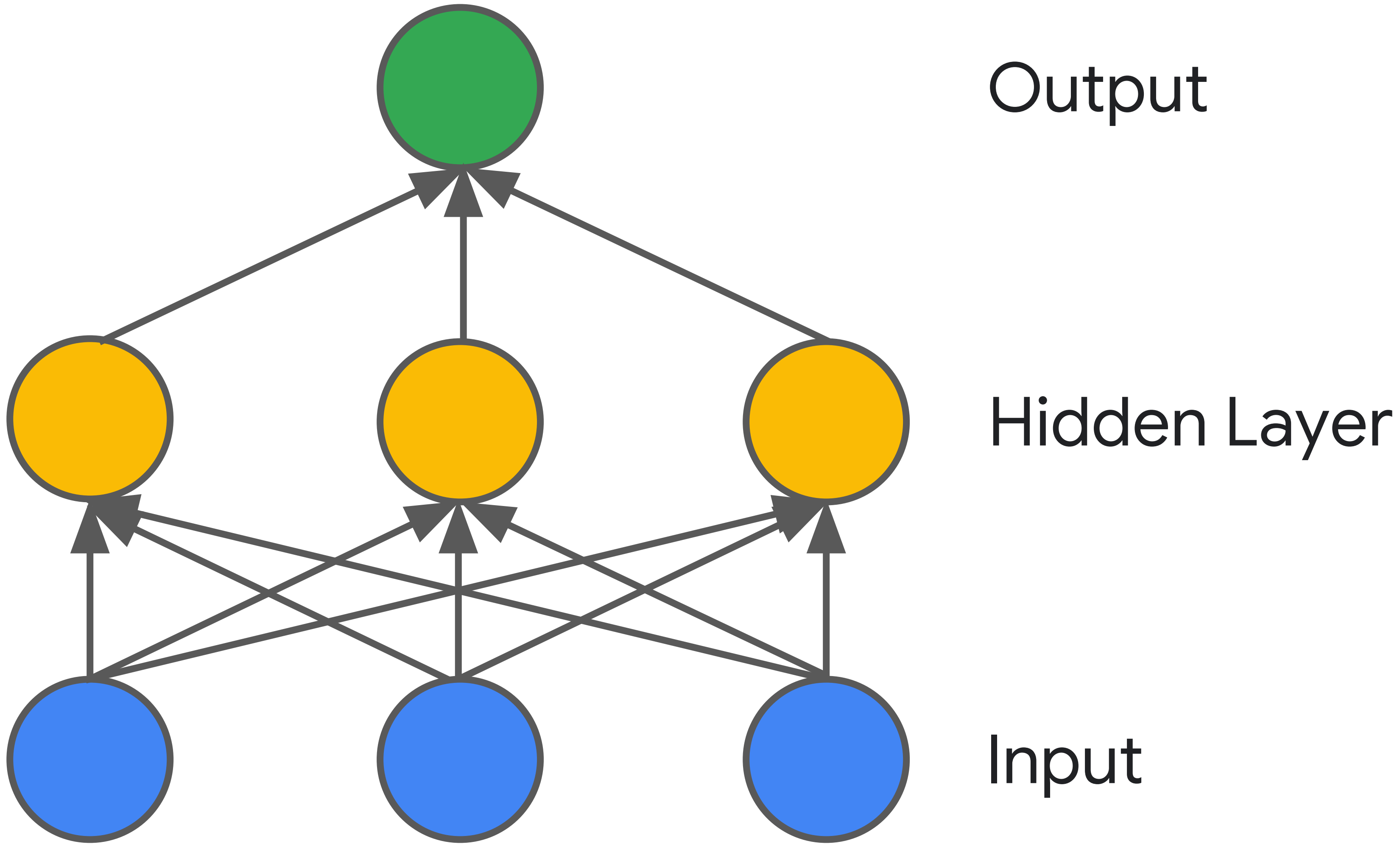
Structure the model so that features are combined
Then the combinations may be combined

How to choose the combinations?
Get the model to learn them

A Linear Model can be represented as nodes and edges

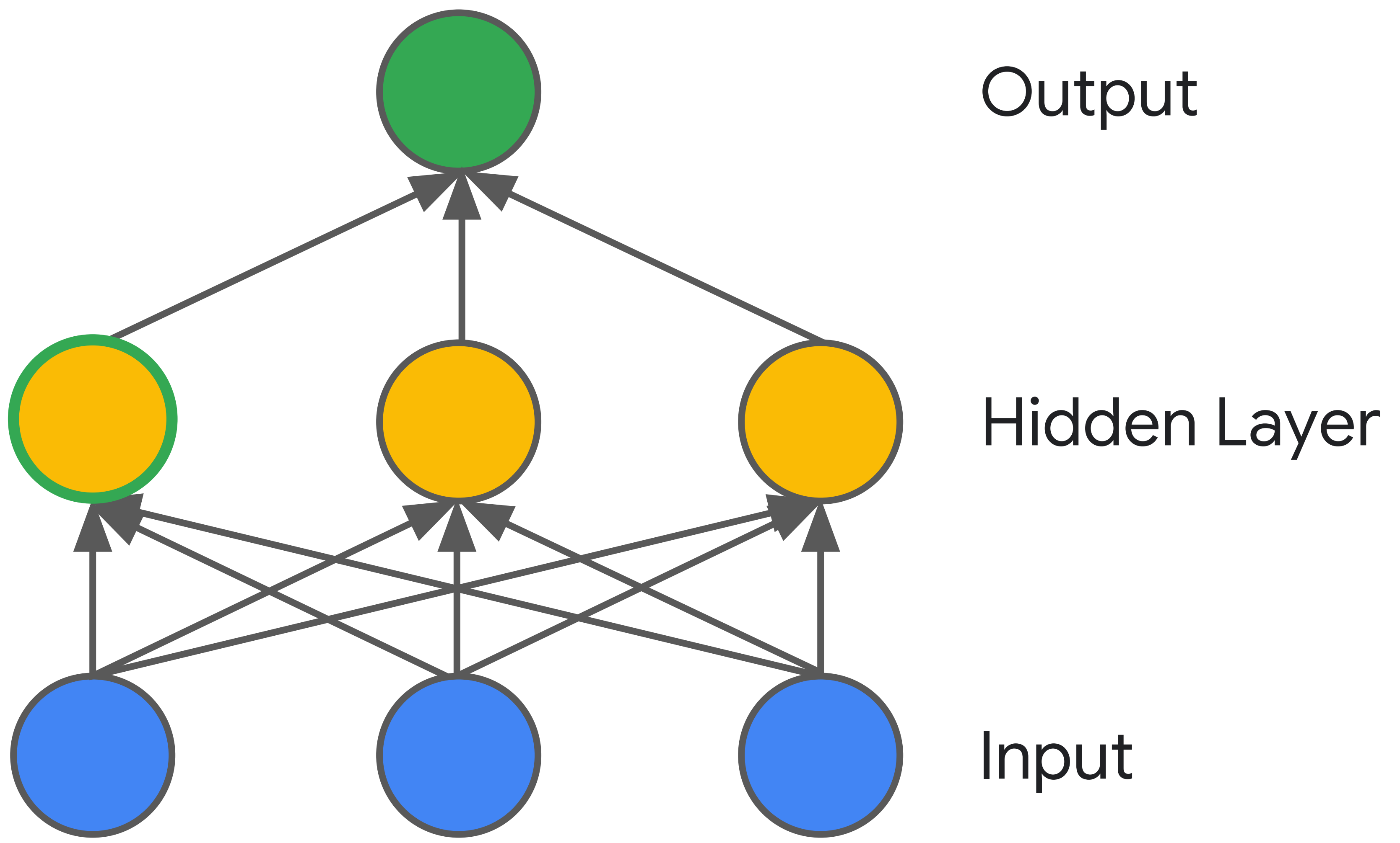


Add Complexity: Non-Linear?



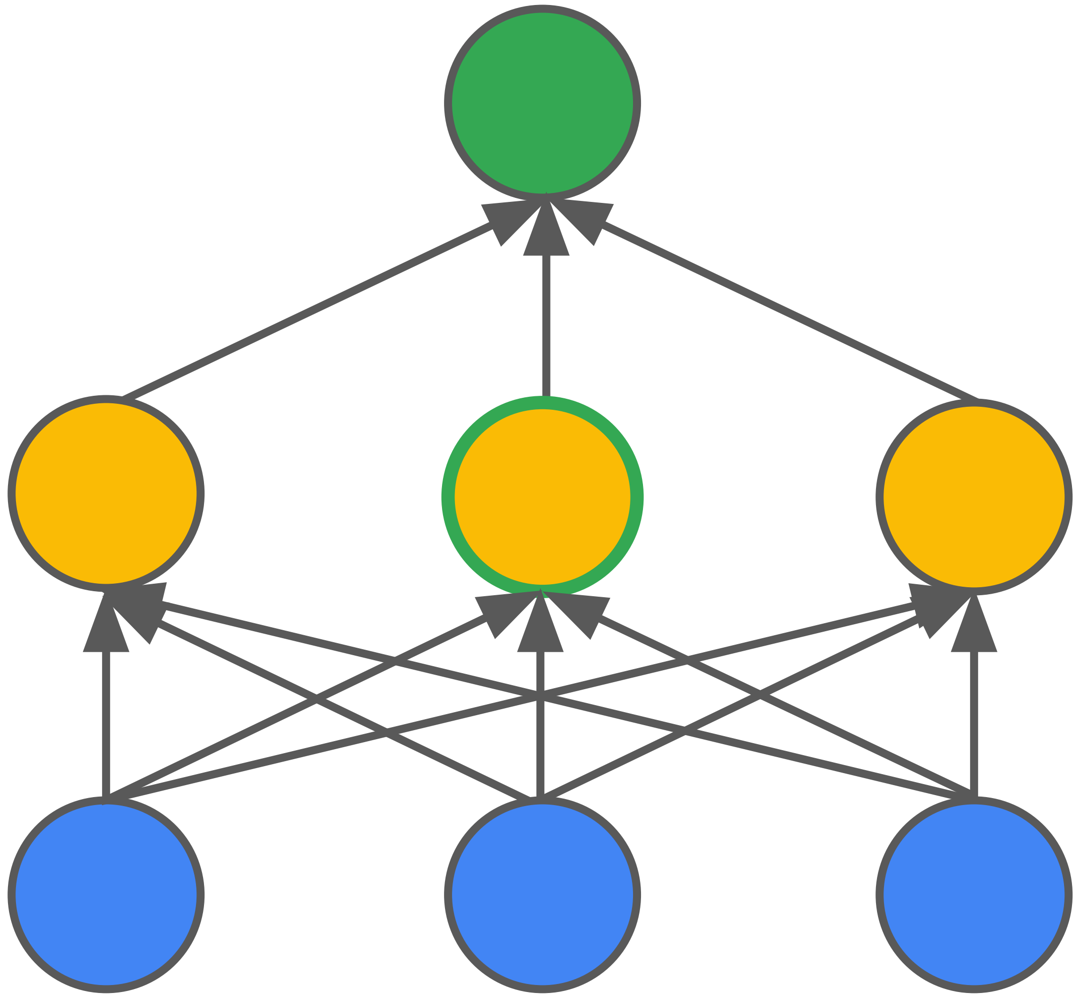
Add Complexity: Non-Linear?

$$h_1 = w_1 * x_1 + w_4 * x_2 + w_7 * x_3$$



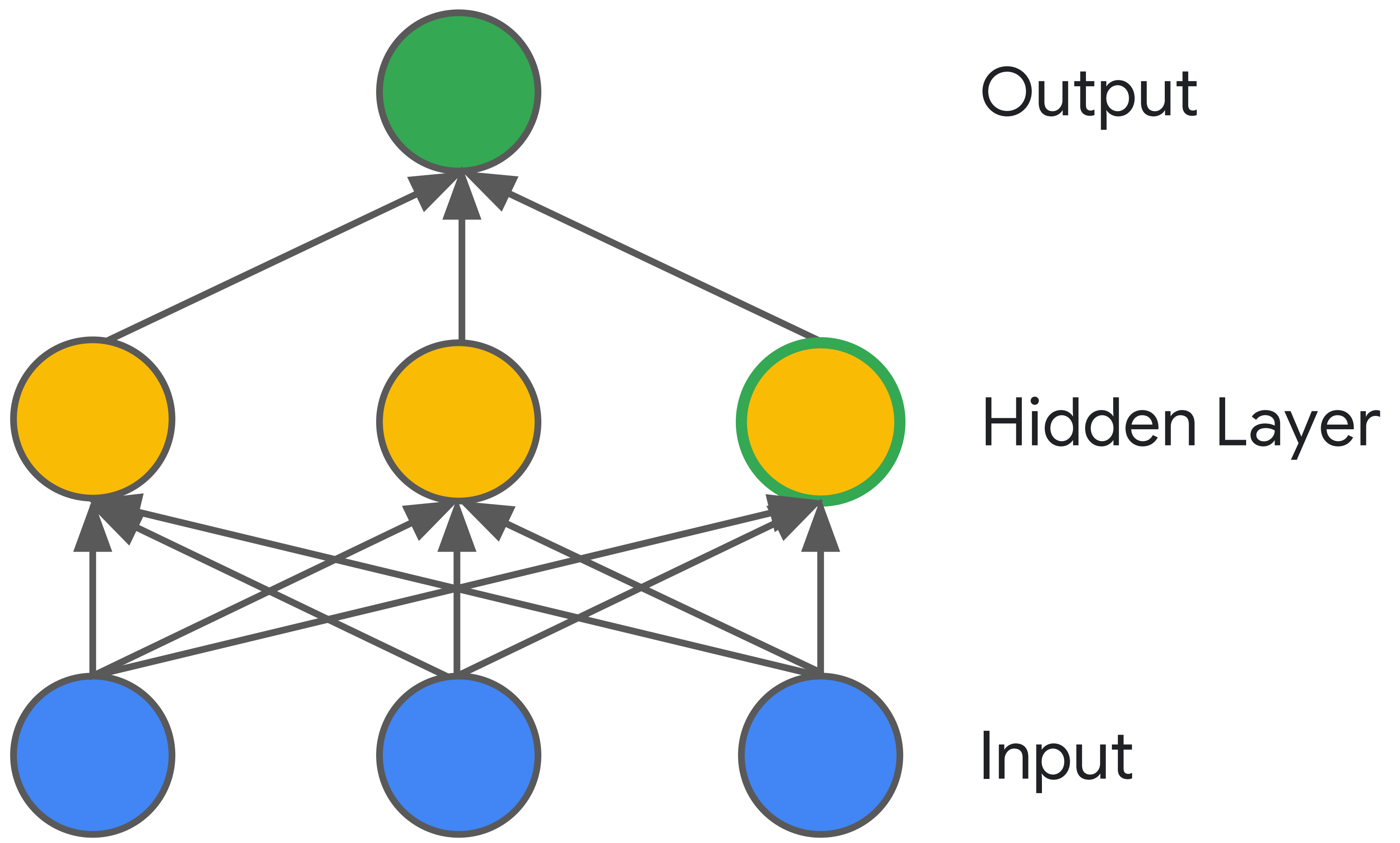
Add Complexity: Non-Linear?

$$h_2 = w_2 * x_1 + w_5 * x_2 + w_8 * x_3$$



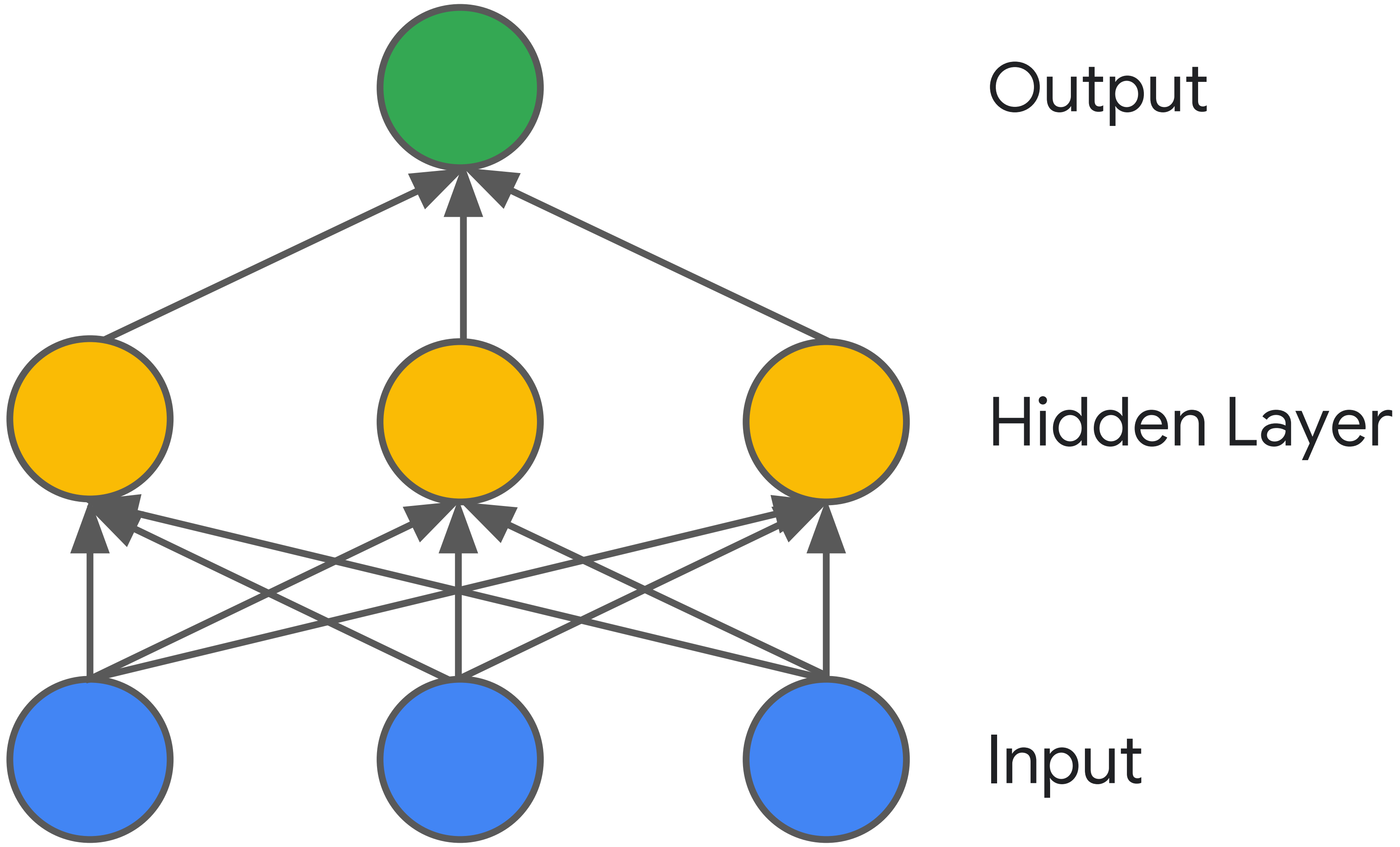
Add Complexity: Non-Linear?

$$h_3 = w_3 * x_1 + w_6 * x_2 + w_9 * x_3$$



Add Complexity: Non-Linear?

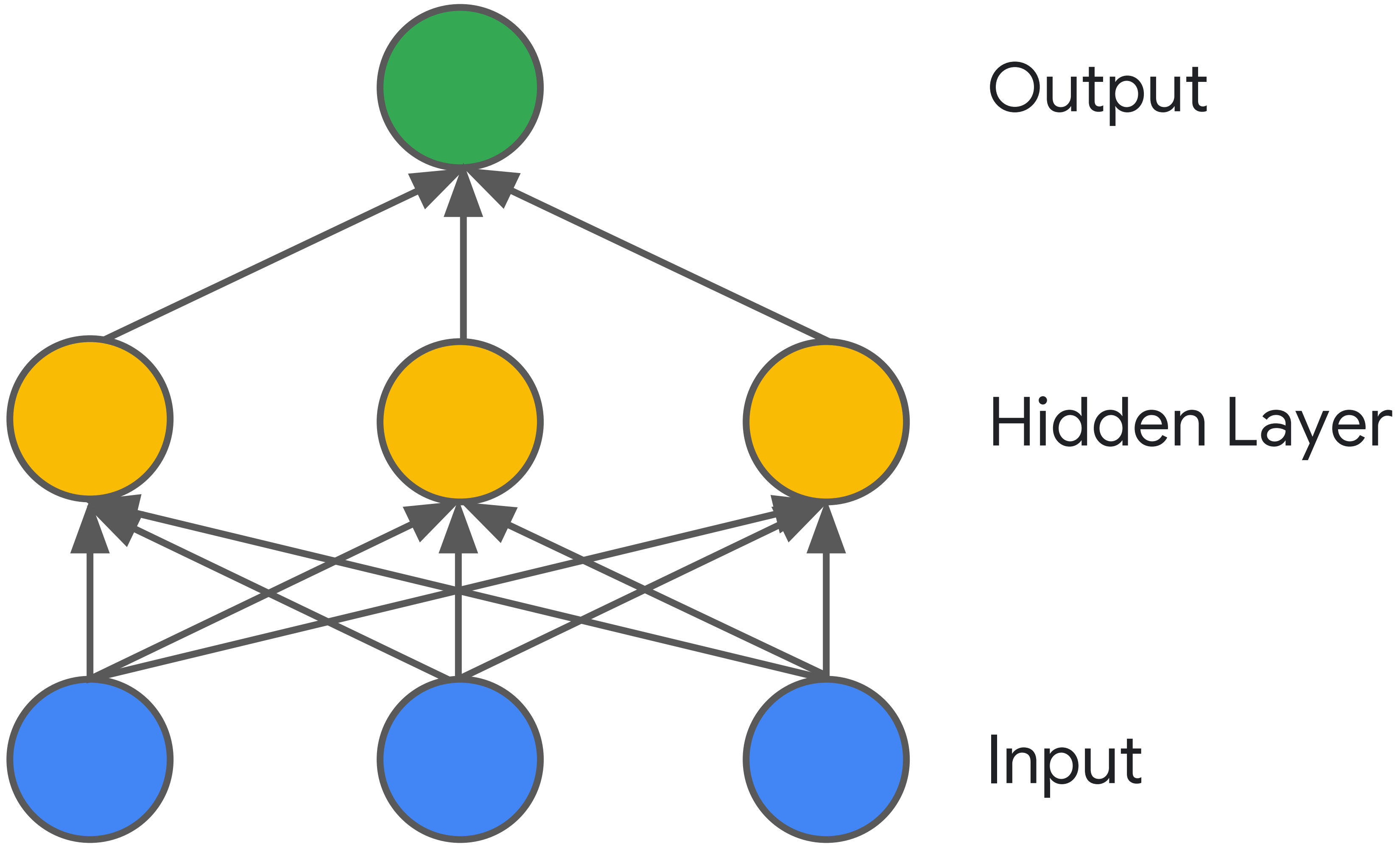
$$output = w_{10} * h_1 + w_{11} * h_2 + w_{12} * h_3$$



Add Complexity: Non-Linear?

$$output = w_{10} * h_1 + w_{11} * h_2 + w_{12} * h_3$$

$$\begin{aligned} &= (w_{10} * w_1 + w_{11} * w_2 + w_{12} * w_3) * x_1 \\ &+ (w_{10} * w_4 + w_{11} * w_5 + w_{12} * w_6) * x_2 \\ &+ (w_{10} * w_7 + w_{11} * w_8 + w_{12} * w_9) * x_3 \end{aligned}$$

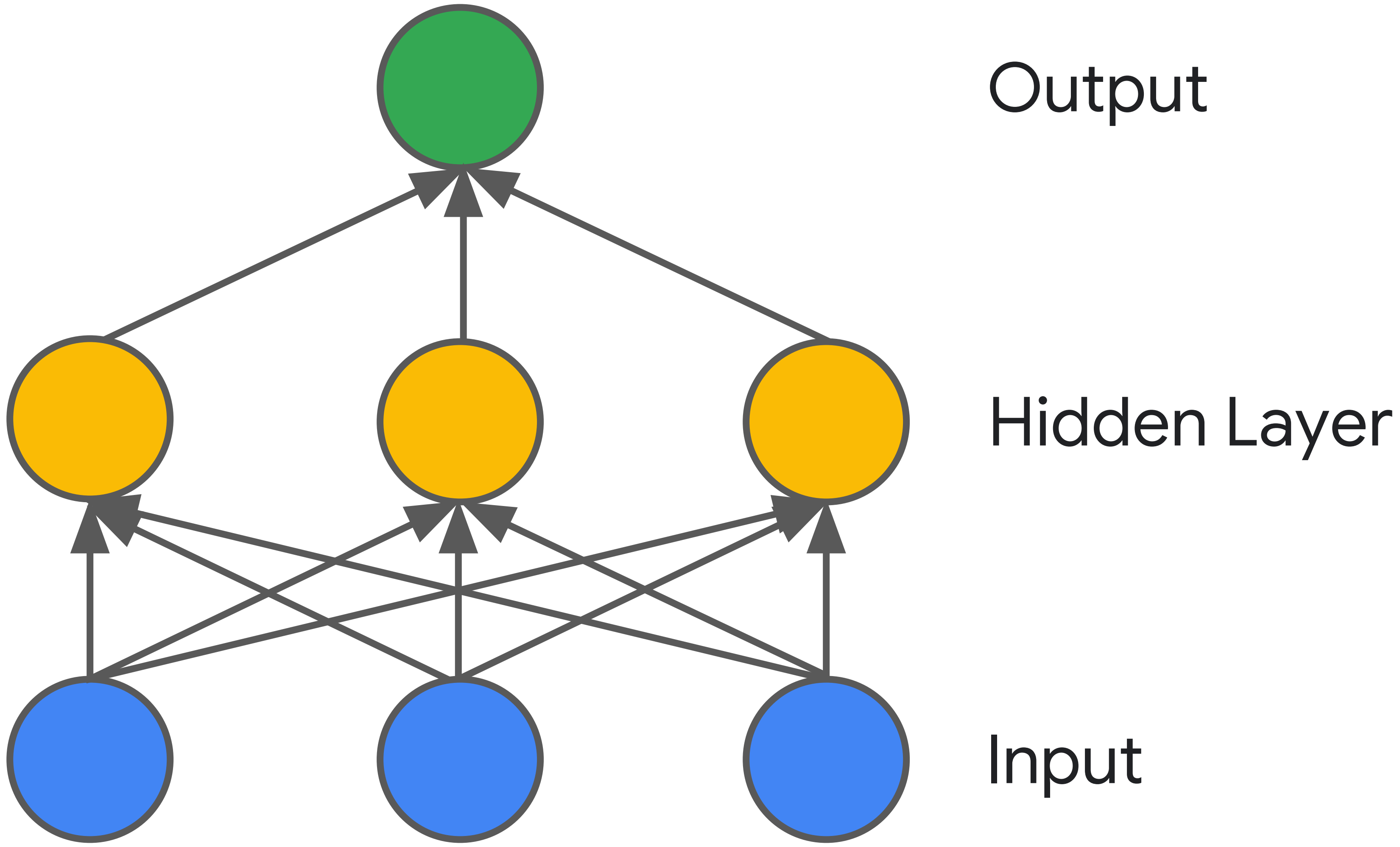


Add Complexity: Non-Linear?

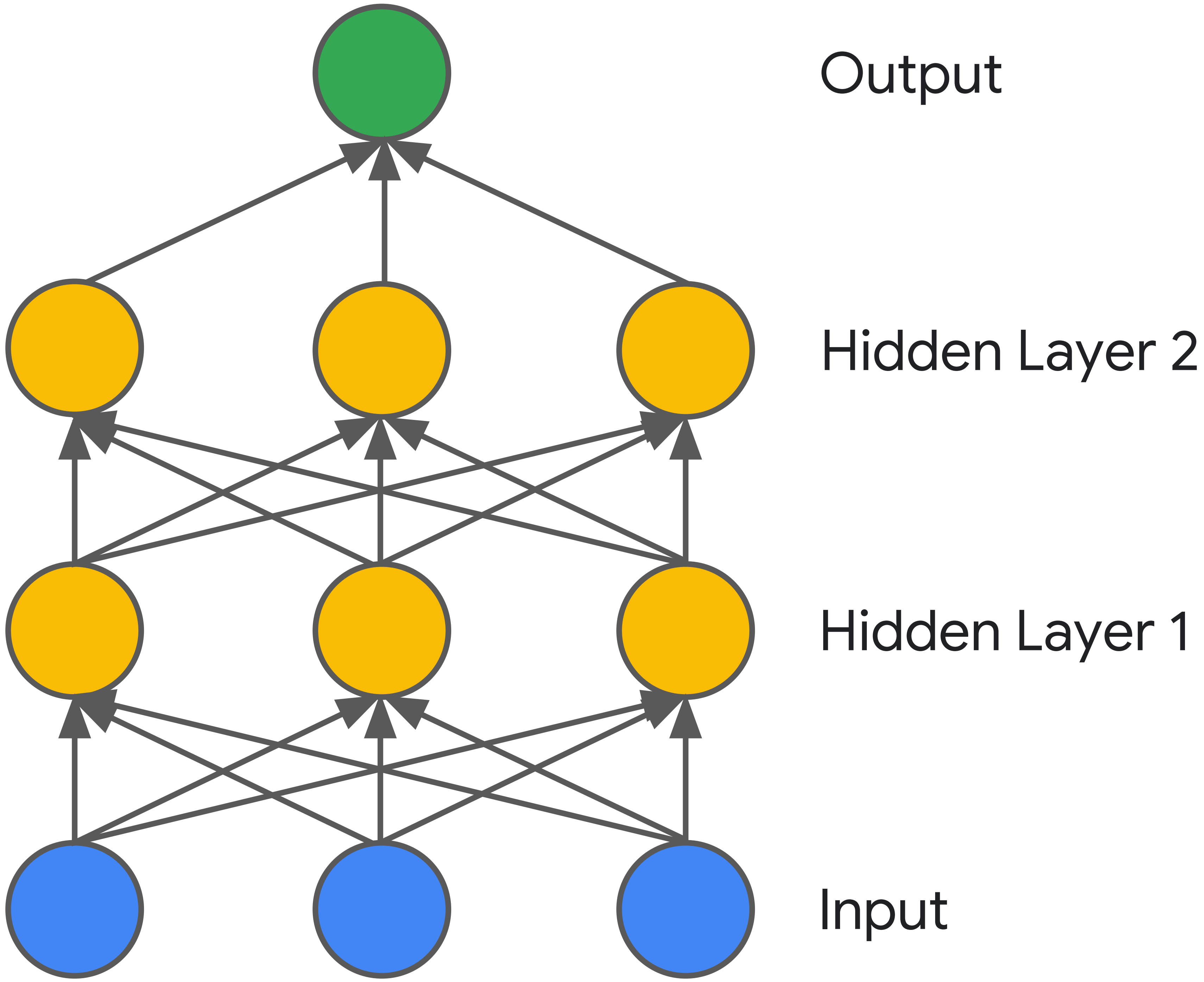
$$output = w_{10} * h_1 + w_{11} * h_2 + w_{12} * h_3$$

$$= (w_{10} * w_1 + w_{11} * w_2 + w_{12} * w_3) * x_1$$
$$+ (w_{10} * w_4 + w_{11} * w_5 + w_{12} * w_6) * x_2$$
$$+ (w_{10} * w_7 + w_{11} * w_8 + w_{12} * w_9) * x_3$$

$$= W_1 * x_1 + W_2 * x_2 + W_3 * x_3$$



More Complex: Non-Linear?



More Complex: Non-Linear?

$$H_1 = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

3x1 3x3 3x1

More Complex: Non-Linear?

$$H_2 = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix} \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

3x1 3x3 3x3 3x1

$$= \begin{bmatrix} W_1 & W_2 & W_3 \\ W_4 & W_5 & W_6 \\ W_7 & W_8 & W_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

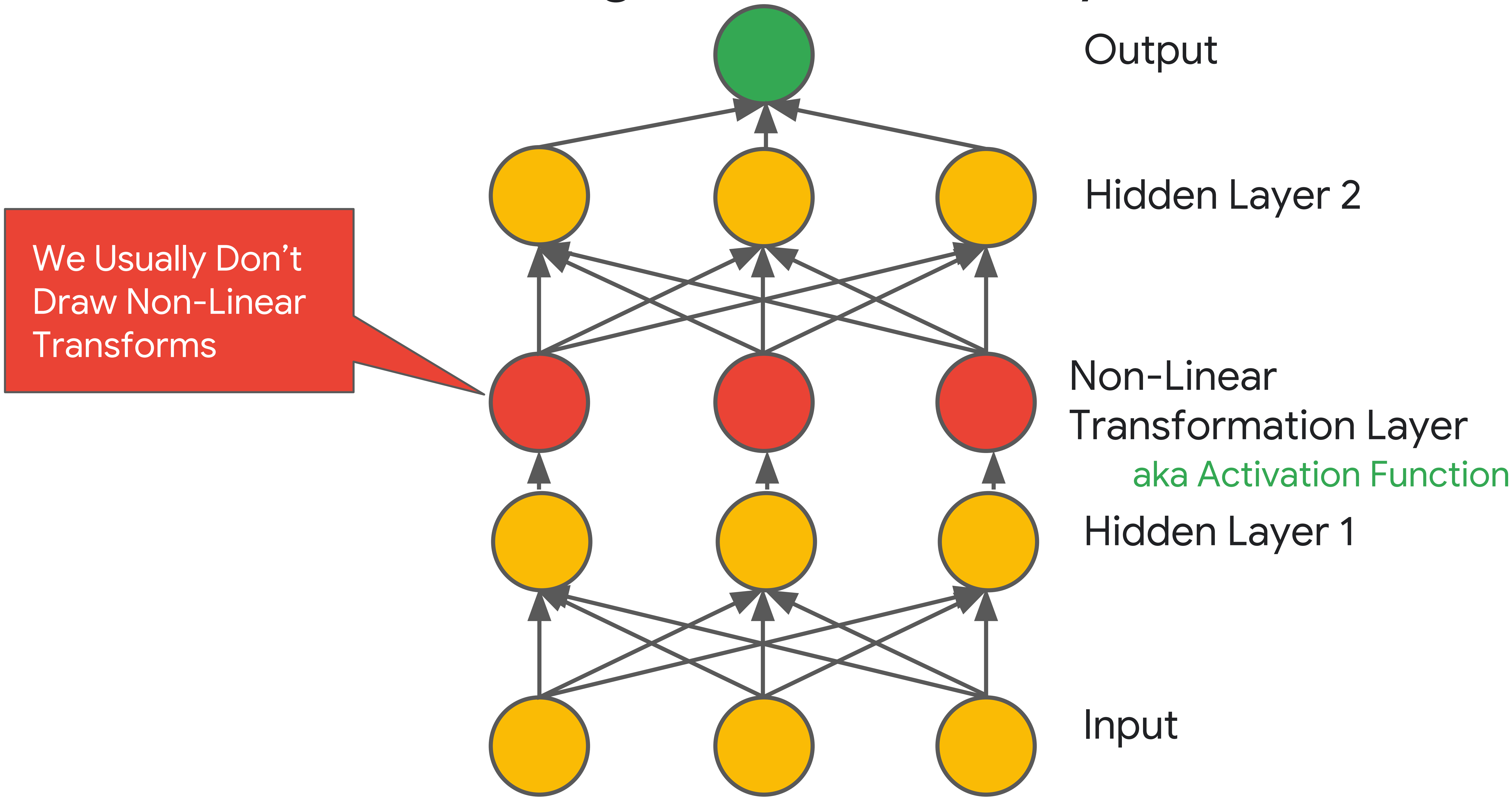
3x3 3x1

More Complex: Non-Linear?

$$\underset{\text{1x1}}{\hat{y}} = \underset{\text{1x3}}{\begin{bmatrix} w_{19} & w_{20} & w_{21} \end{bmatrix}} \underset{\text{3x3}}{\begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix}} \underset{\text{3x3}}{\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}} \underset{\text{3x1}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}$$

$$= \underset{\text{1x3}}{\begin{bmatrix} W'_1 & W'_2 & W'_3 \end{bmatrix}} \underset{\text{3x1}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}$$

Adding a Non-Linearity



Adding a Non-Linearity

$$\hat{y} = \begin{bmatrix} w_{19} & w_{20} & w_{21} \end{bmatrix} \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix} f\left(\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right)$$

1x1 1x3 3x3 3x3 3x1

$$= \begin{bmatrix} w_{19} & w_{20} & w_{21} \end{bmatrix} \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix} \begin{bmatrix} max(0, w_1x_1 + w_2x_2 + w_3x_3) \\ max(0, w_4x_1 + w_5x_2 + w_6x_3) \\ max(0, w_7x_1 + w_8x_2 + w_9x_3) \end{bmatrix}$$

1x3 3x3 3x1

Non-linearity Quiz

Why is it important adding non-linear activation functions to neural networks?

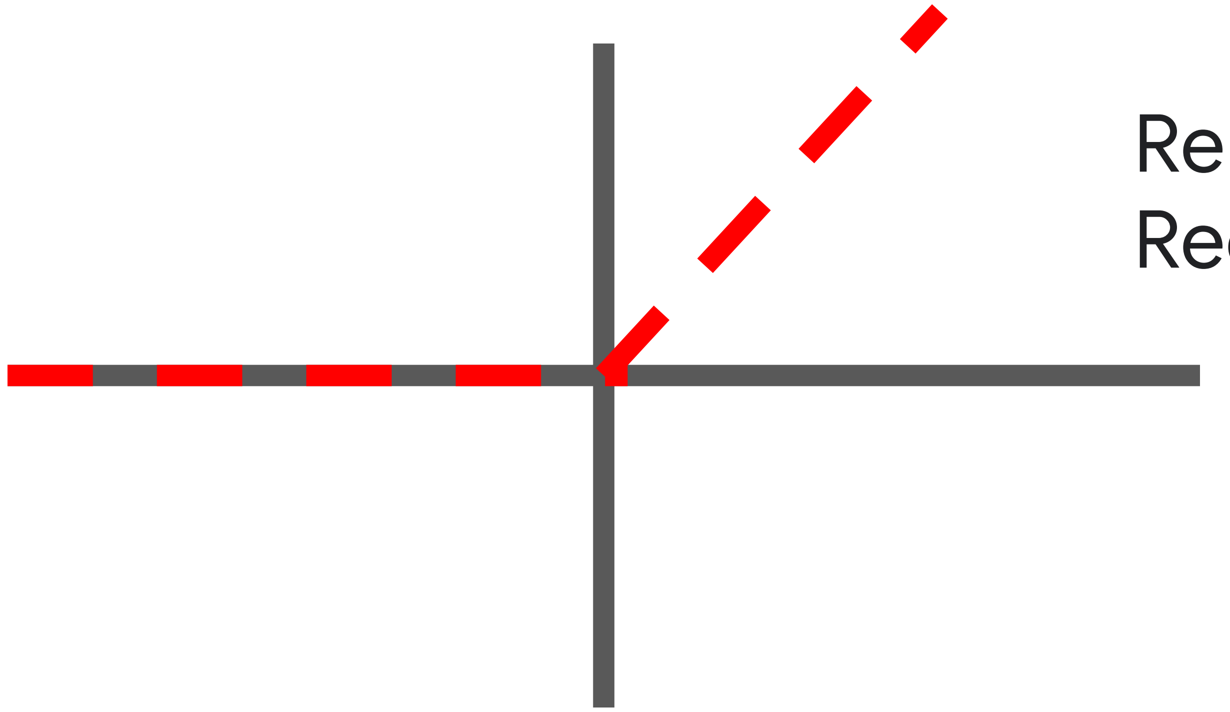
A. Adds regularization
B. Increases the number of dimensions
C. Invokes early stopping
D. Stops the layers from collapsing back into just a linear model

Non-linearity Quiz

Why is it important adding non-linear activation functions to neural networks?

A. Adds regularization
B. Increases the number of dimensions
C. Invokes early stopping
D. Stops the layers from collapsing back into just a linear model

Our favorite non-linearity is the Rectified Linear Unit

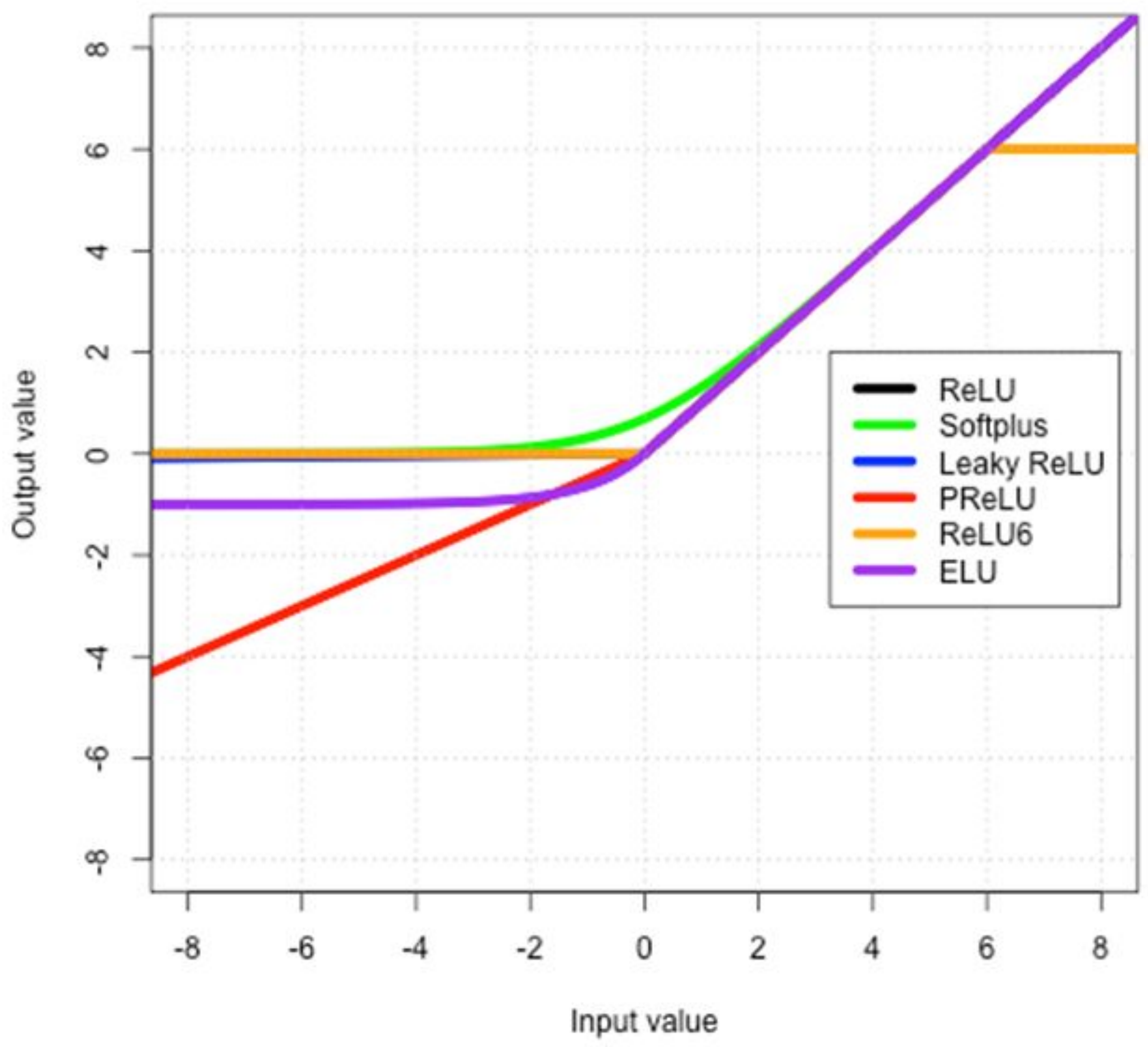


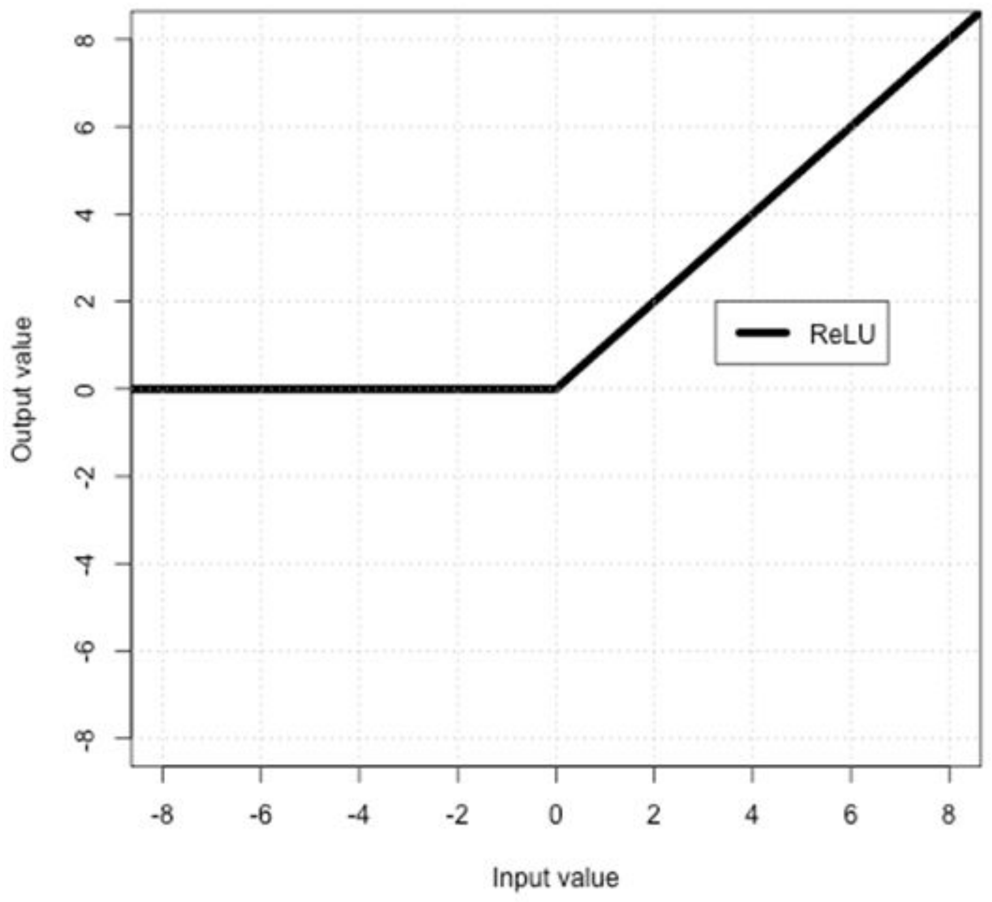
ReLU
Rectified Linear Unit

$$f(x) = max(0, x)$$

There are many different ReLU variants

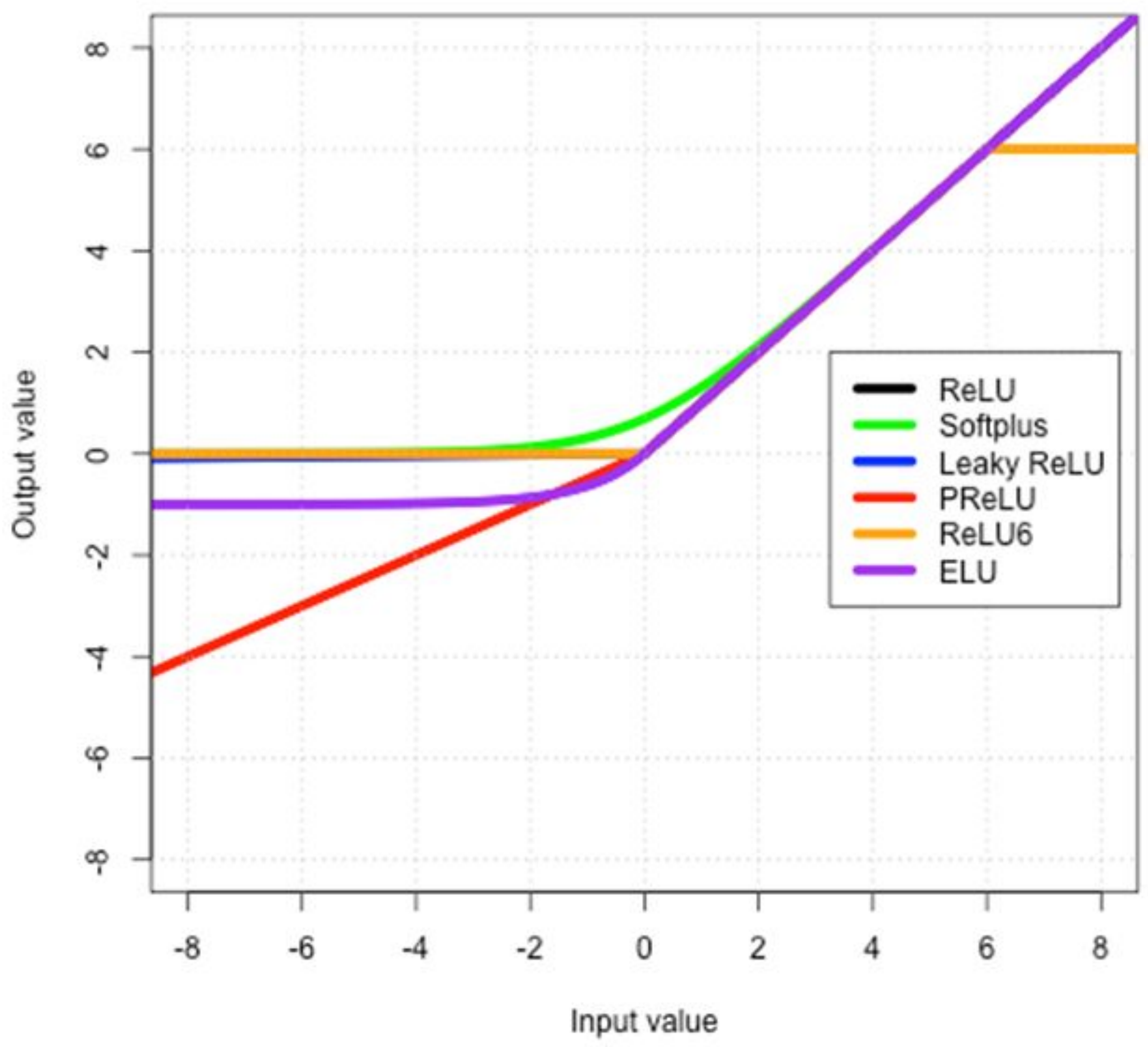
$$ReLU = f(x) = \begin{Bmatrix} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{Bmatrix}$$

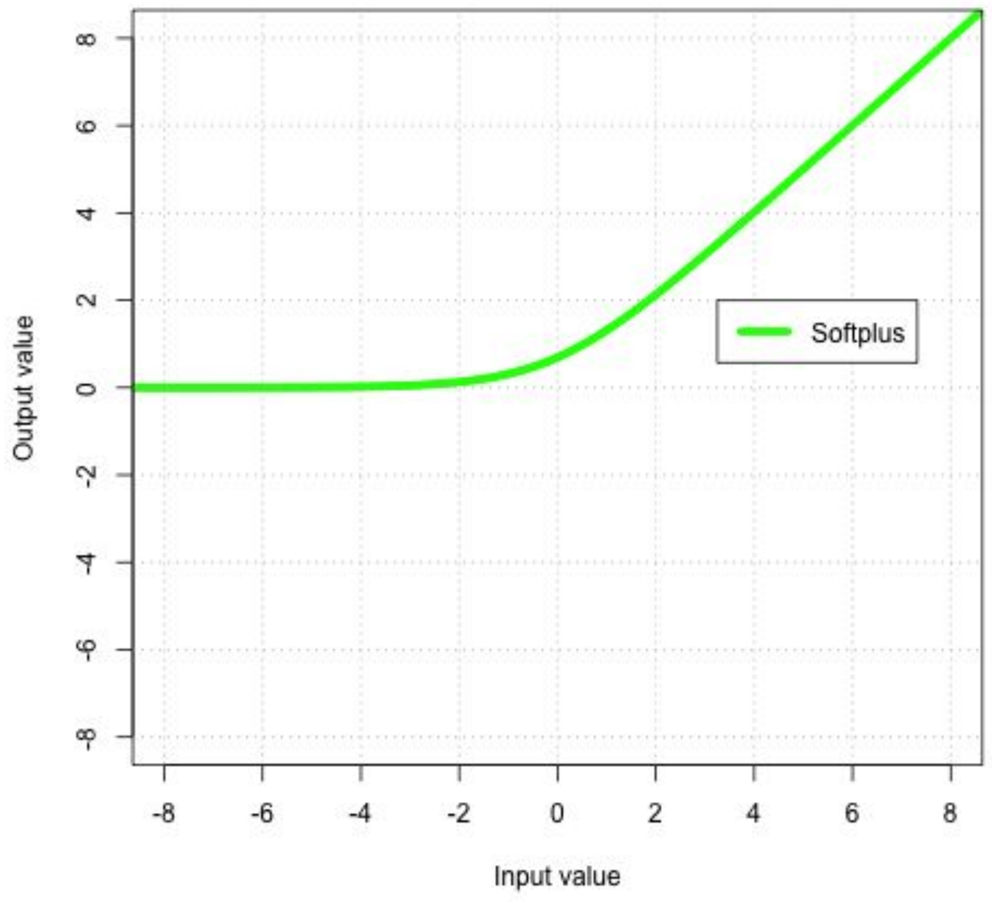




There are many different ReLU variants

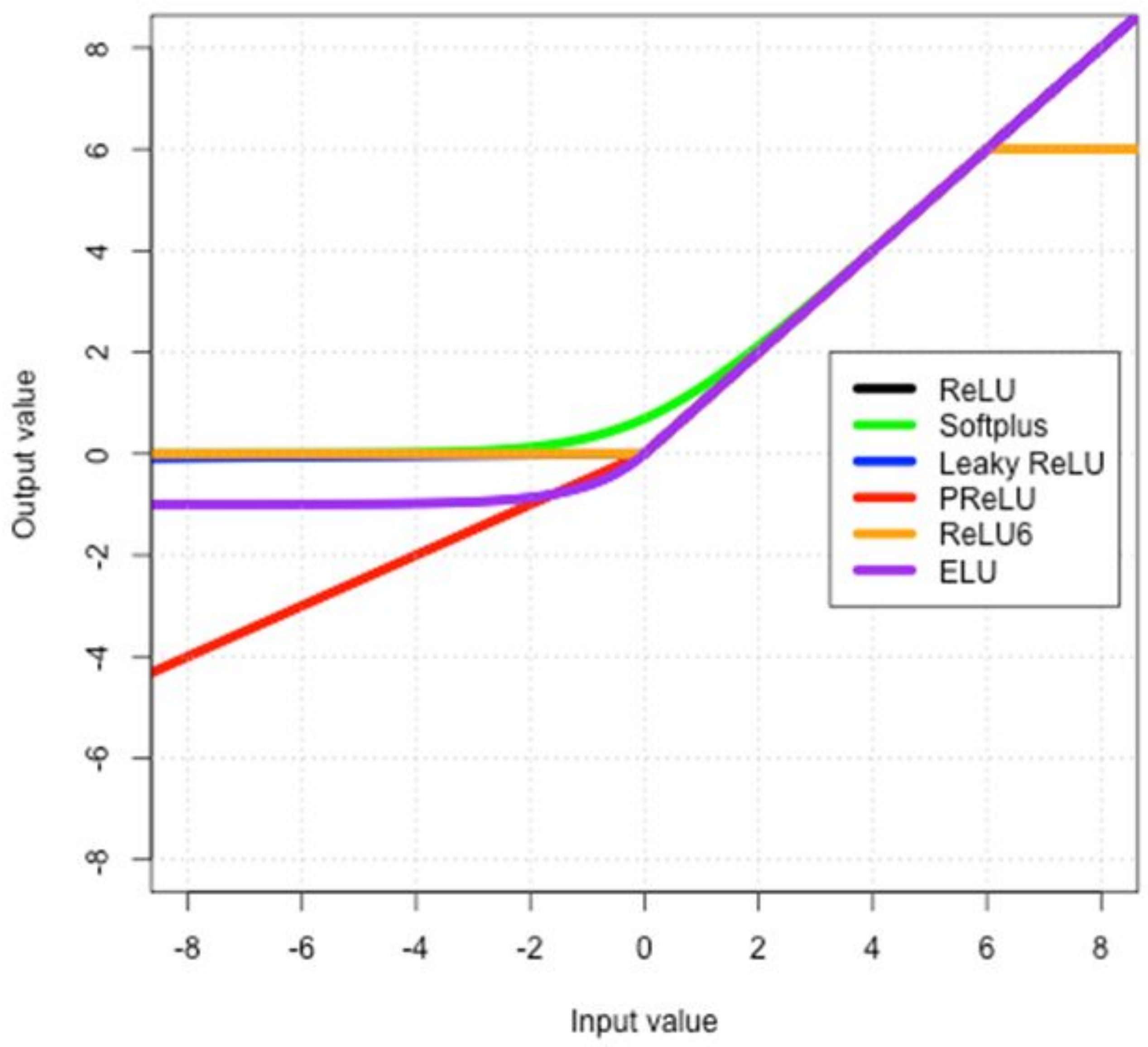
$$Softplus = ln(1 + e^x)$$

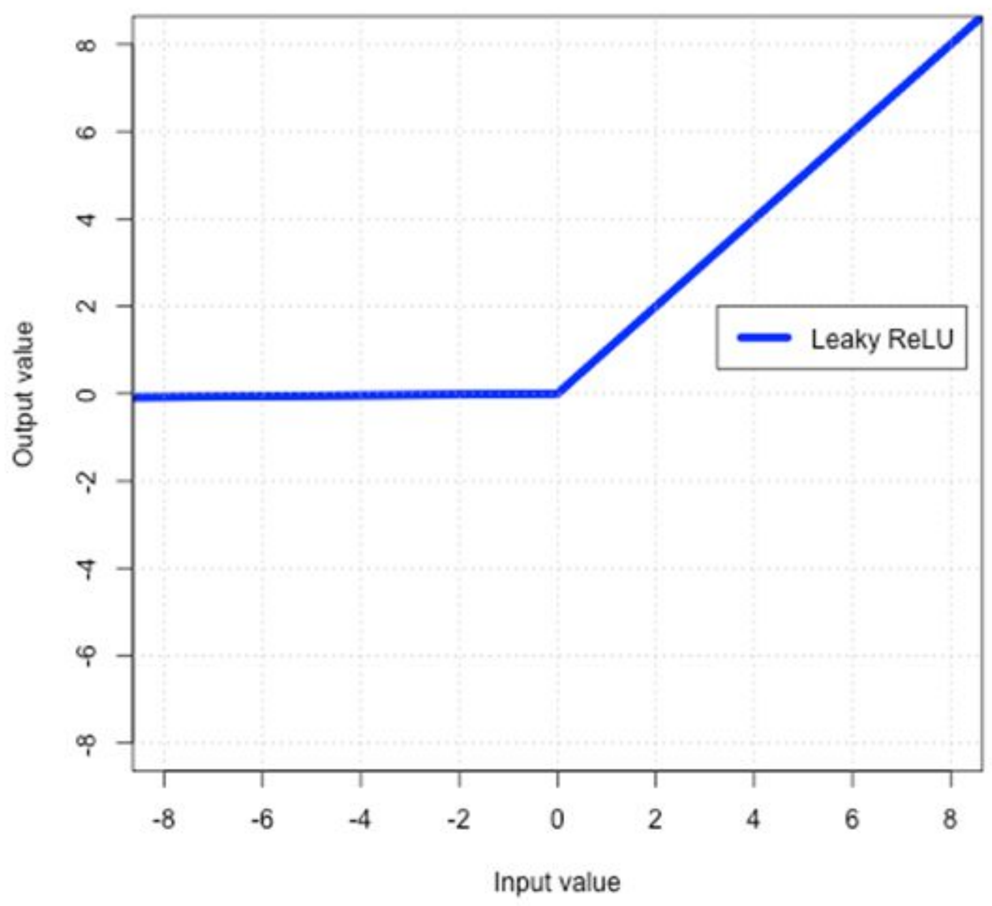




There are many different ReLU variants

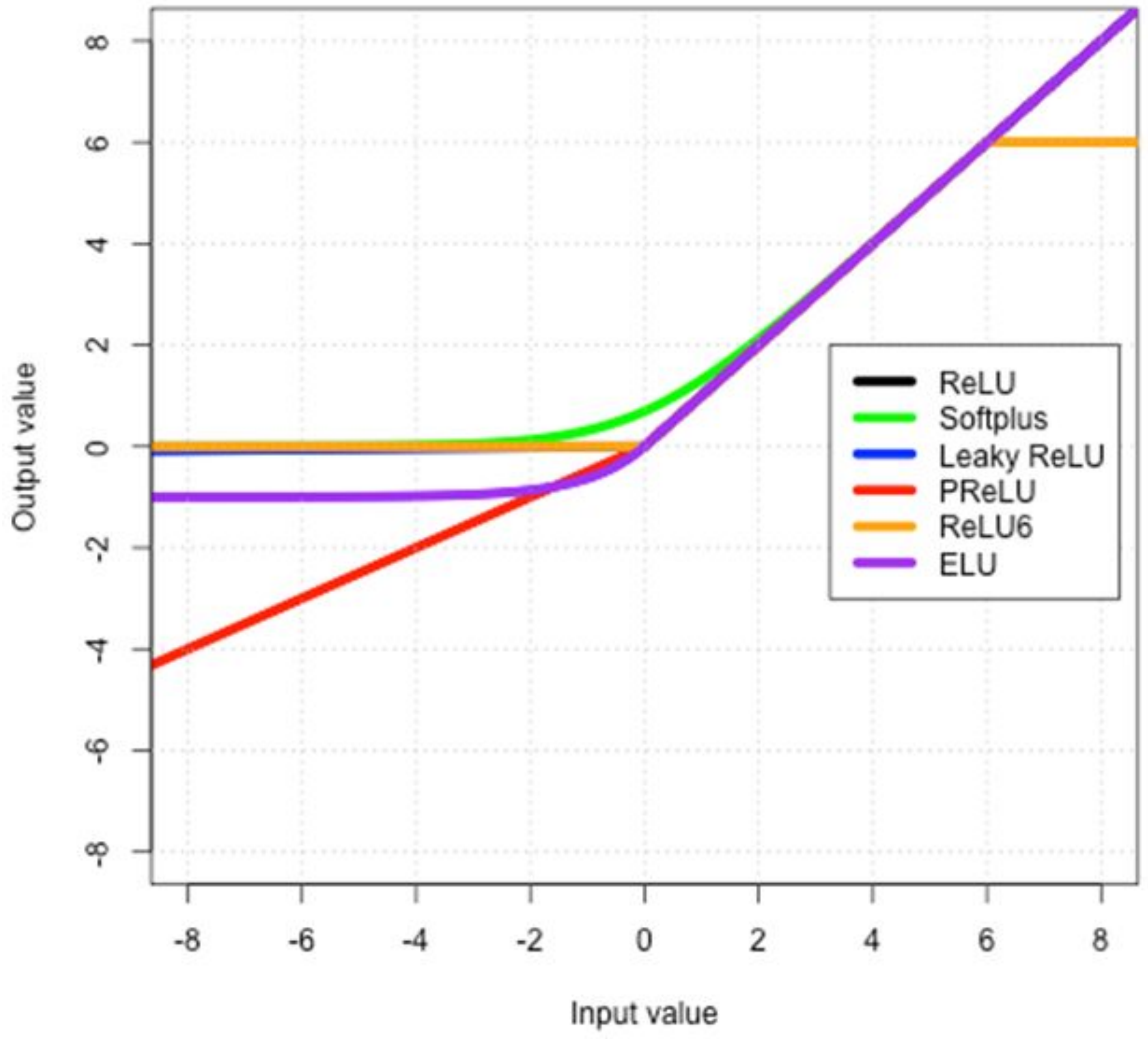
$$Leaky\ ReLU = f(x) = \begin{Bmatrix} 0.01x & for\ x < 0 \\ x & for\ x \geq 0 \end{Bmatrix}$$

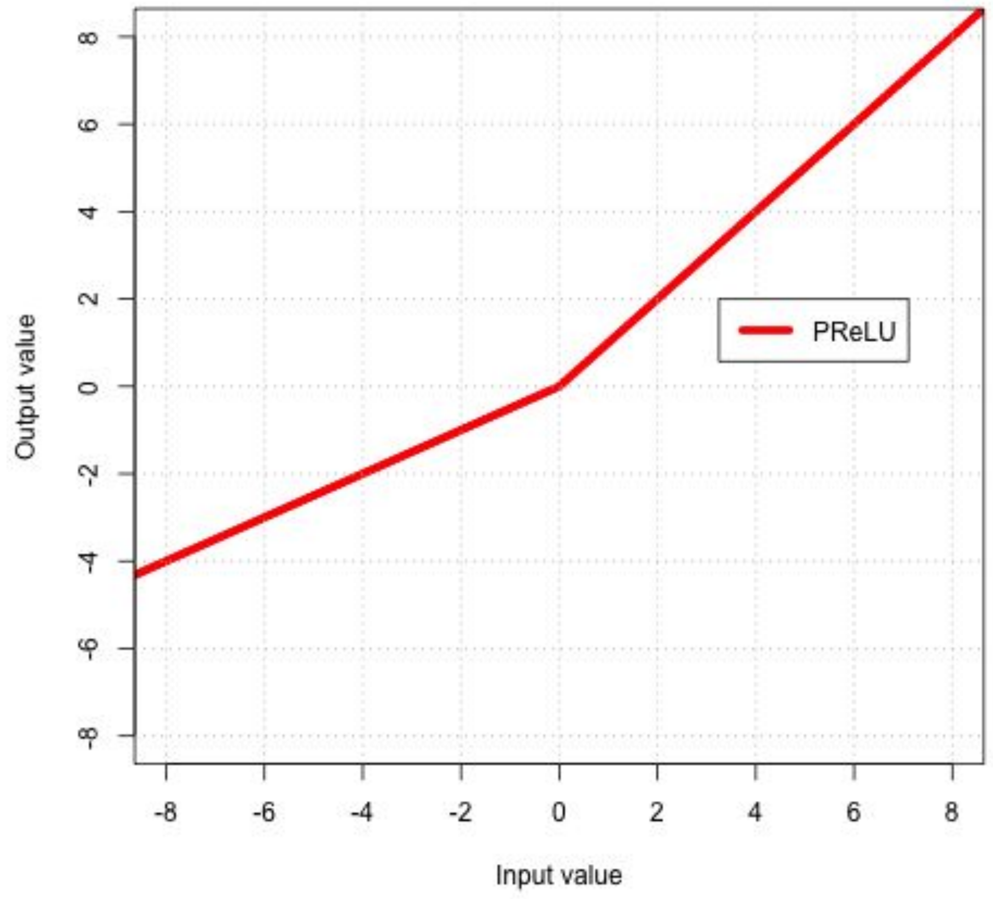




There are many different ReLU variants

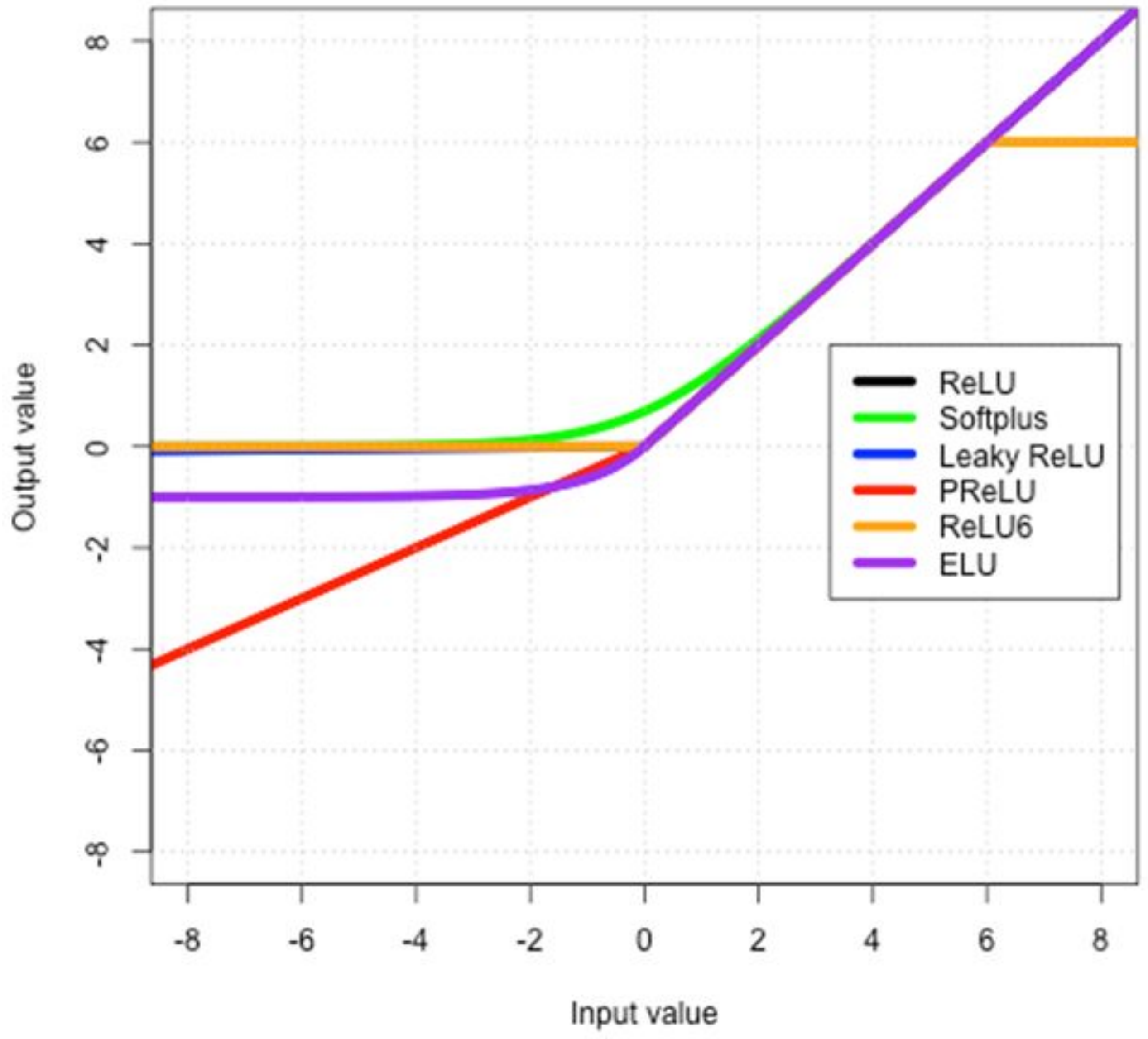
$$PReLU = f(x) = \begin{Bmatrix} \alpha x & for\ x < 0 \\ x & for\ x \geq 0 \end{Bmatrix}$$

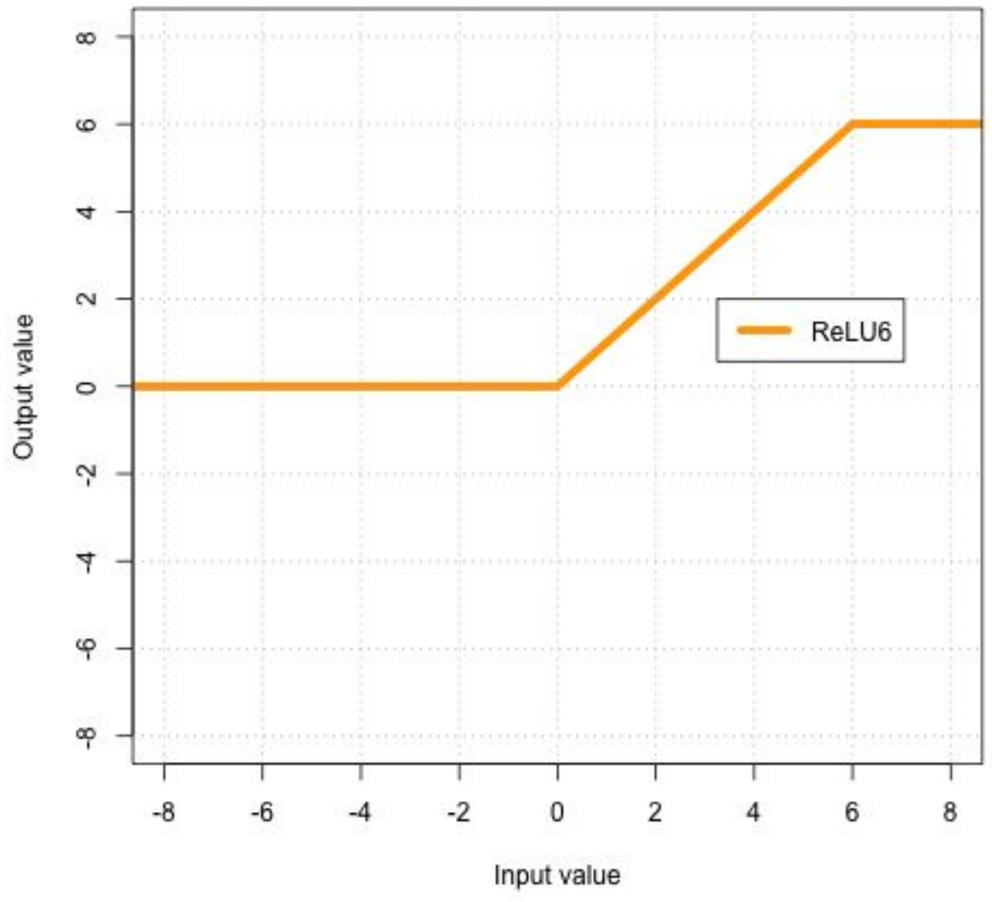




There are many different ReLU variants

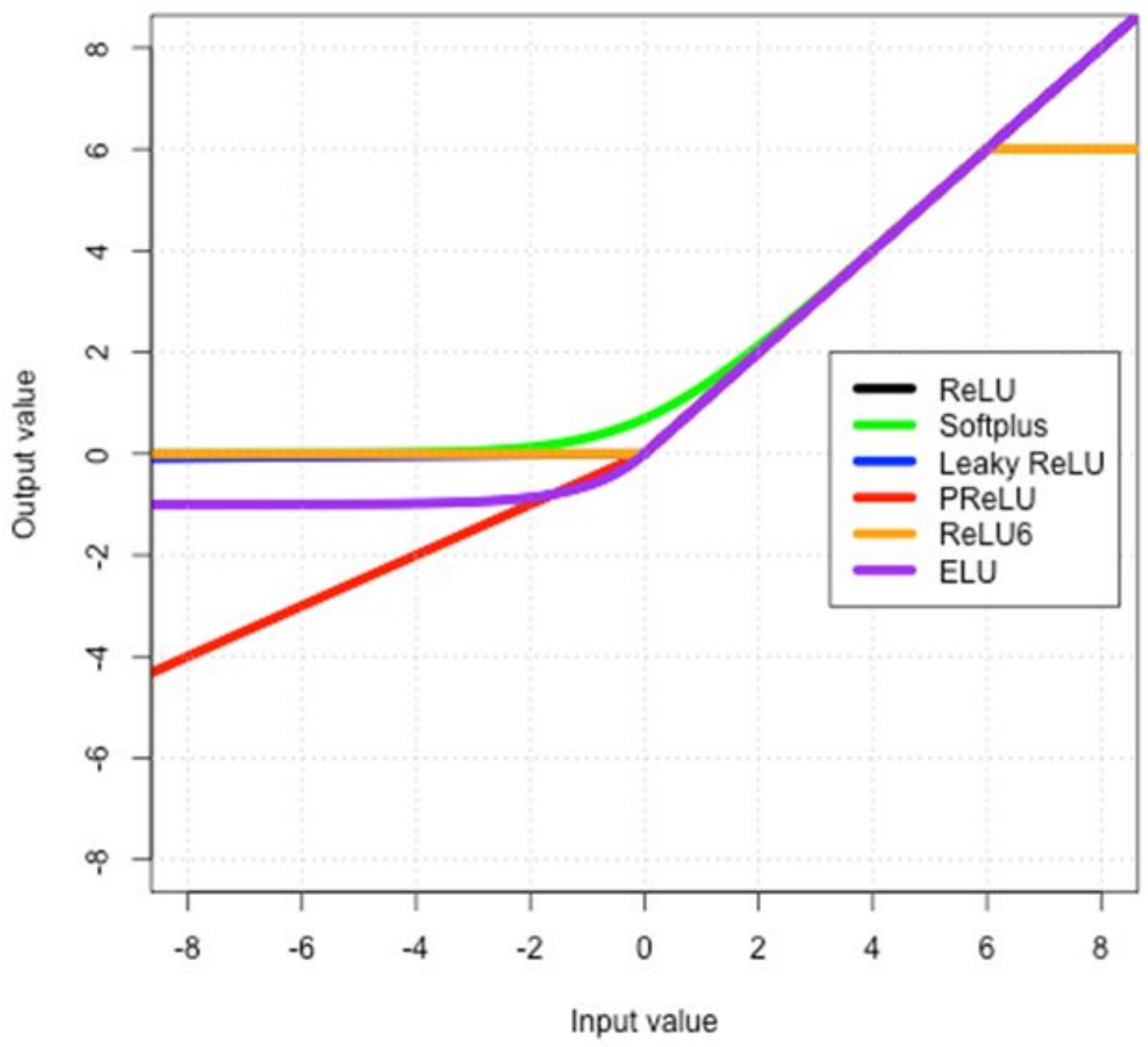
$$ReLU6 = min(max(0, x), 6)$$

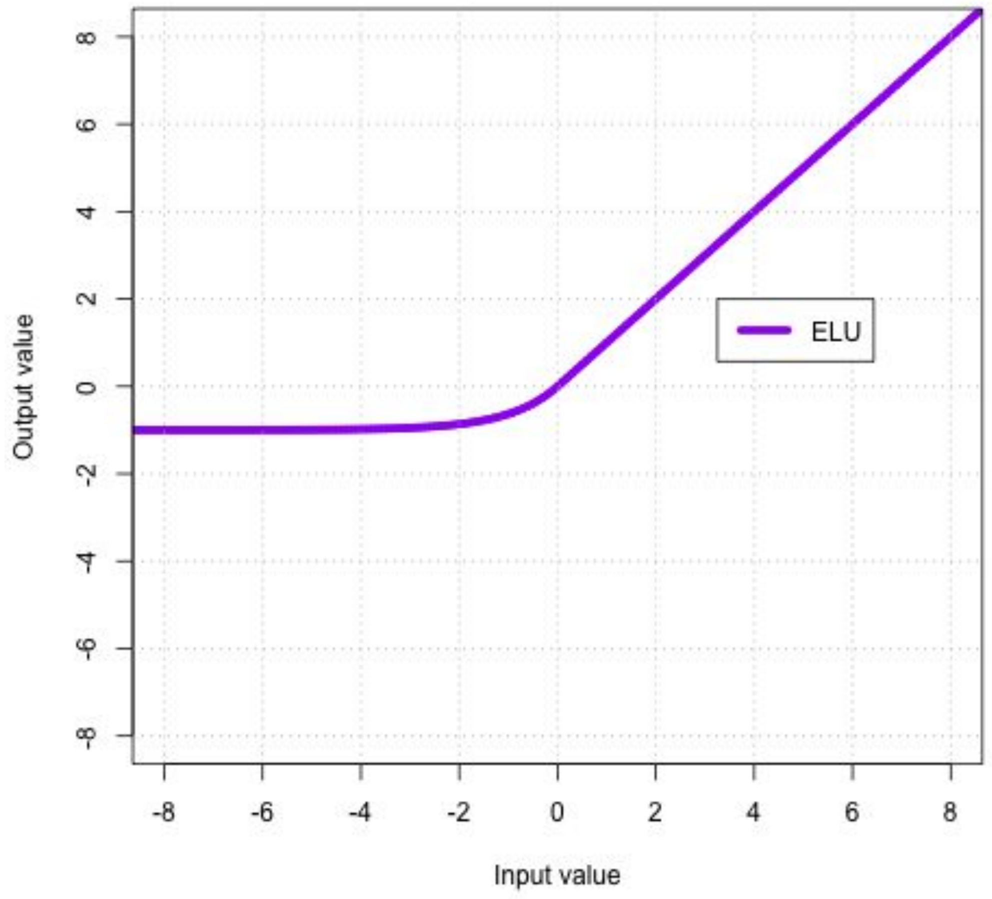




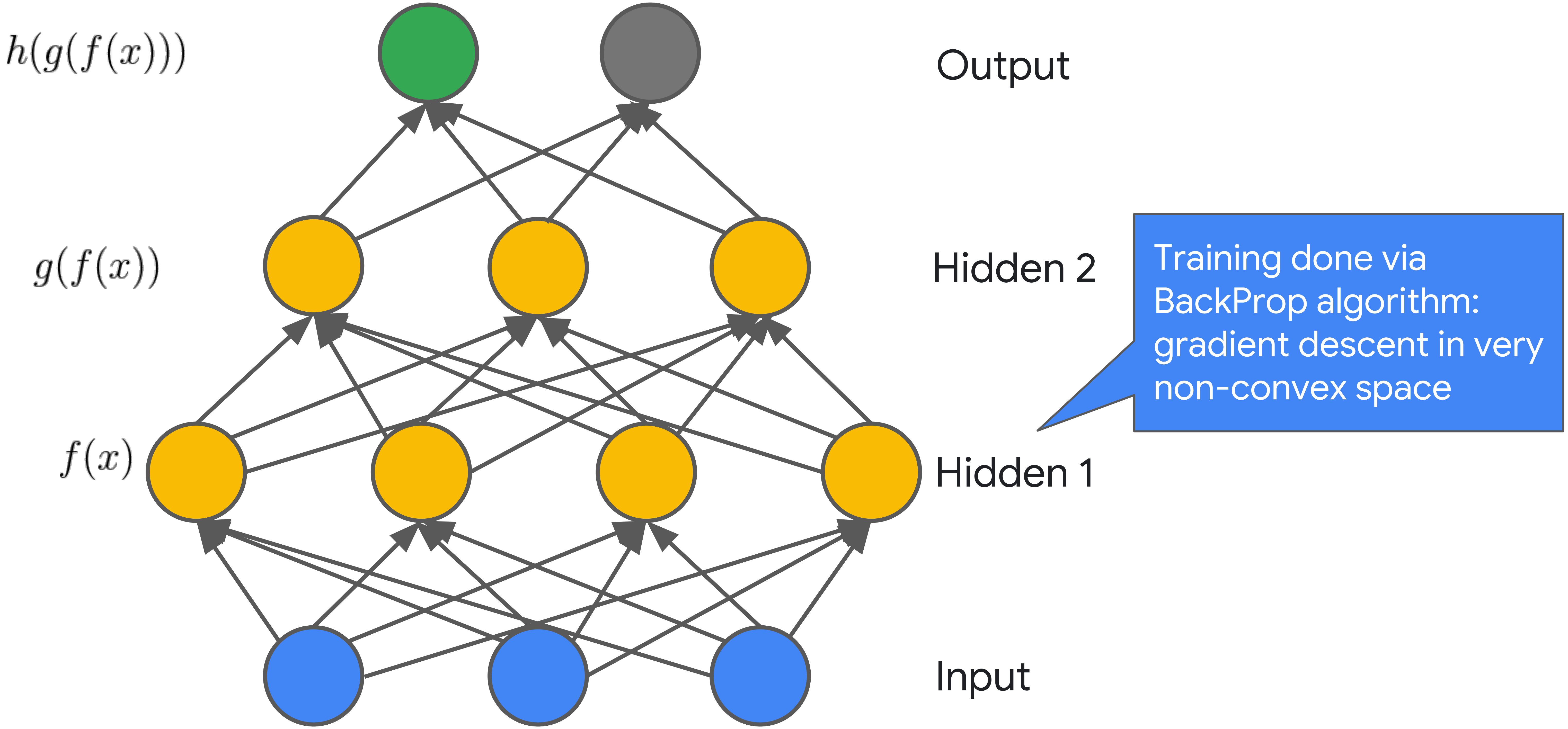
There are many different ReLU variants

$$ELU = f(x) = \begin{cases} \alpha(e^x - 1) & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

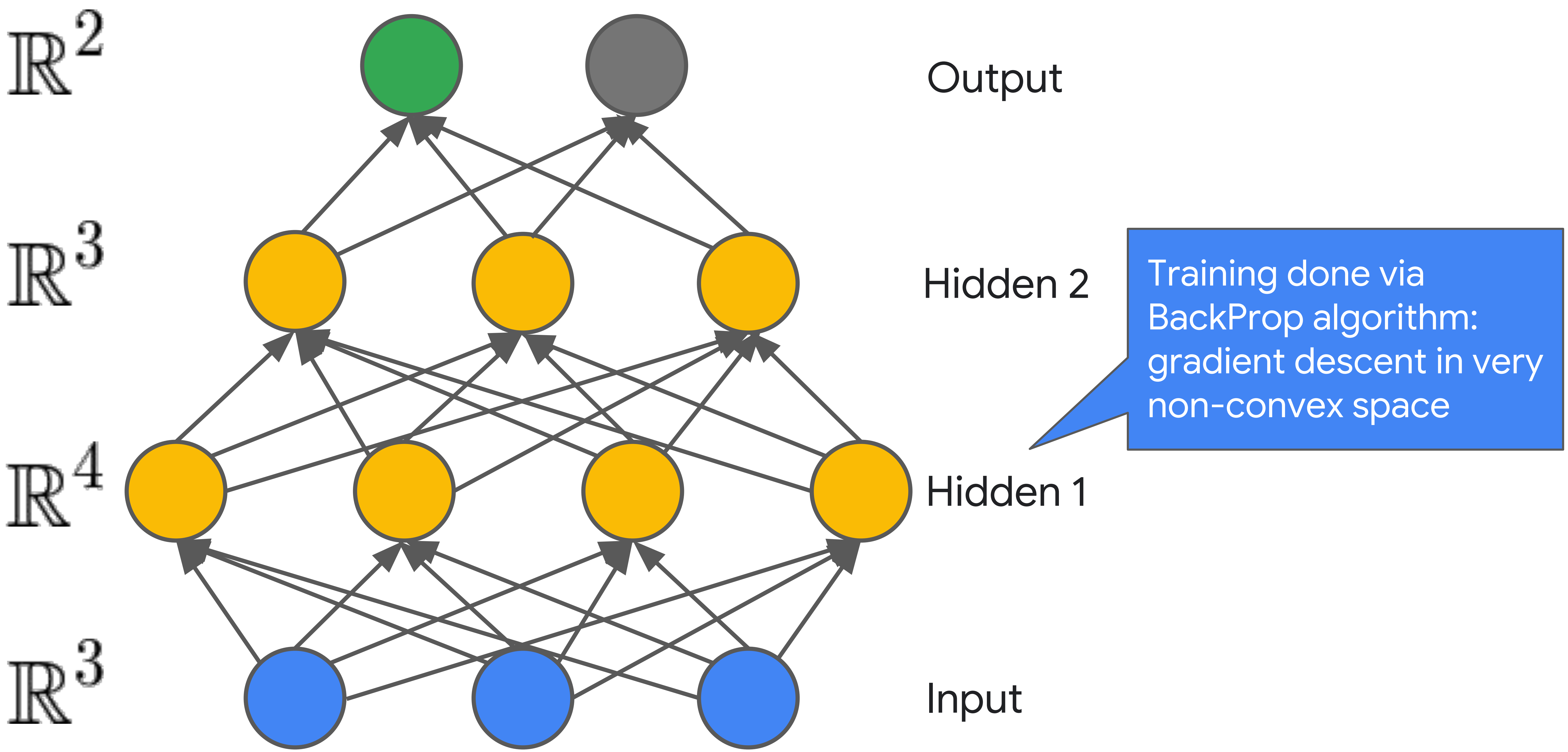




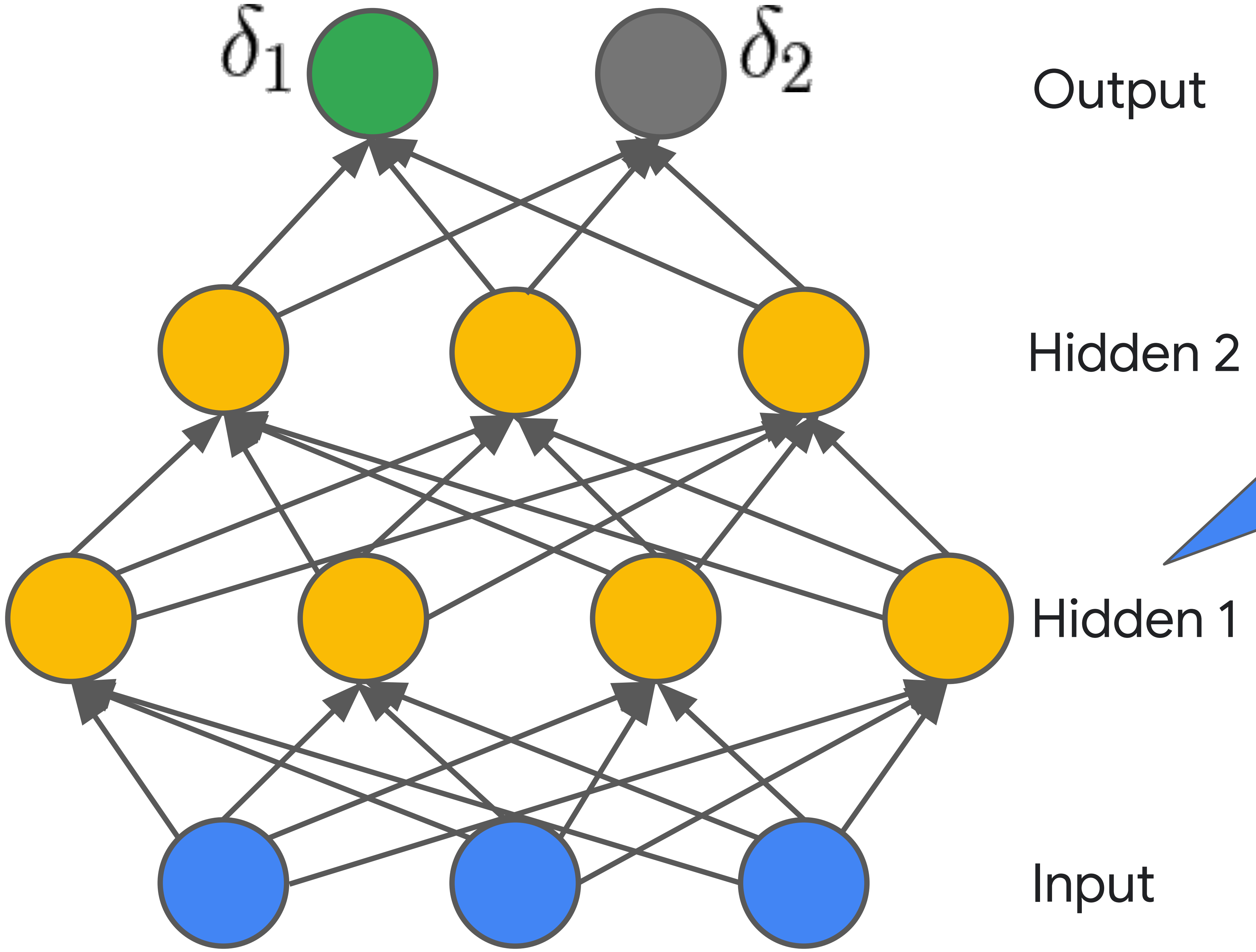
Neural Nets Can Be Arbitrarily Complex



Neural Nets Can Be Arbitrarily Complex



Neural Nets Can Be Arbitrarily Complex



Training done via BackProp algorithm: gradient descent in very non-convex space

Neural Network Complexity Quiz

Neural networks can be arbitrarily complex. To increase hidden dimensions, I can add _______. To increase function composition, I can add _______. If I have multiple labels per example, I can add _______.

A. Layers, neurons, outputs
B. Neurons, layers, outputs
C. Layers, outputs, neurons
D. Neurons, outputs, layers

Neural Network Complexity Quiz

Neural networks can be arbitrarily complex. To increase hidden dimensions, I can add _______. To increase function composition, I can add _______. If I have multiple labels per example, I can add _______.

A. Layers, neurons, outputs
B. Neurons, layers, outputs
C. Layers, outputs, neurons
D. Neurons, outputs, layers

Lab

Neural Networks playground

Lab: Neural Networks playground

Solve these problems in two ways: one by feature engineering, the other by adding layers:

https://goo.gl/2eig4q

https://goo.gl/wXbGDW

https://goo.gl/i9r55D

Lab: Neural Networks playground

Camtasia

Training neural networks

DNNRegressor usage is similar to LinearRegressor

```
myopt = tf.train.AdamOptimizer(learning_rate=0.01)

model = tf.estimator.DNNRegressor(model_dir=outdir,
                    hidden_units=[100, 50, 20],
                                 feature_columns=INPUT_COLS,
                                 optimizer=myopt,
                                 dropout=0.1)

NSTEPS = (100 * len(traindf)) / BATCH_SIZE
model.train(input_fn=train_input_fn, steps=NSTEPS)
```

Use momentum-based optimizers e.g. Adagrad (the default) or Adam.

Specify number of hidden nodes.

Optionally, can also regularize using dropout

Three common failure modes for gradient descent

Gradients can vanish			Problem
Each additional layer can successively reduce signal vs. noise			Insight
Using ReLu instead of sigmoid/tanh can help			Solution

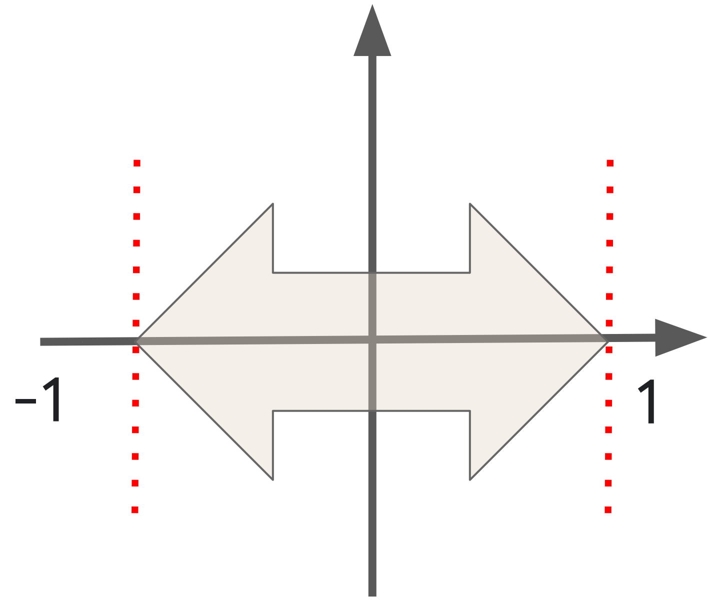
Three common failure modes for gradient descent

Gradients can vanish	Gradients can explode		
			Problem
Each additional layer can successively reduce signal vs. noise	Learning rates are important here		Insight
Using ReLu instead of sigmoid/tanh can help	Batch normalization (useful knob) can help		Solution

Three common failure modes for gradient descent

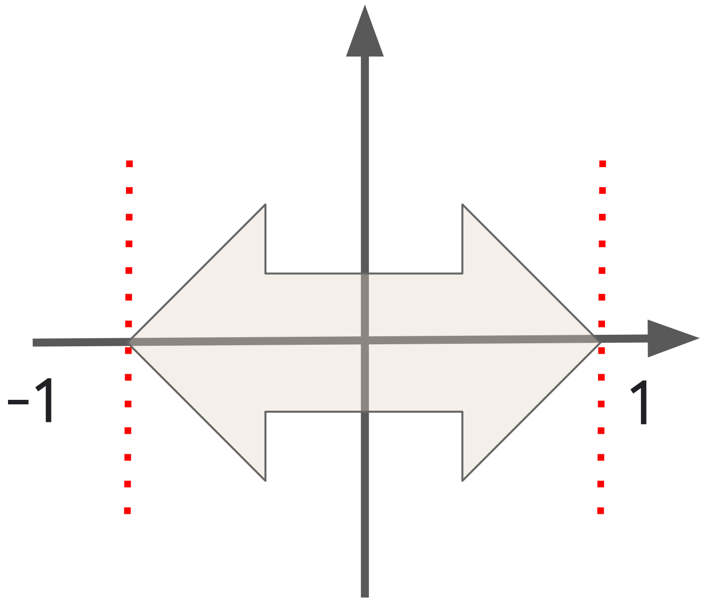
Gradients can vanish	**Gradients can explode**	**ReLu layers can die**	Problem
Each additional layer can successively reduce signal vs. noise	Learning rates are important here	Monitor fraction of zero weights in TensorBoard	Insight
Using ReLu instead of sigmoid/tanh can help	Batch normalization (useful knob) can help	Lower your learning rates	Solution

There are benefits if feature values are small numbers

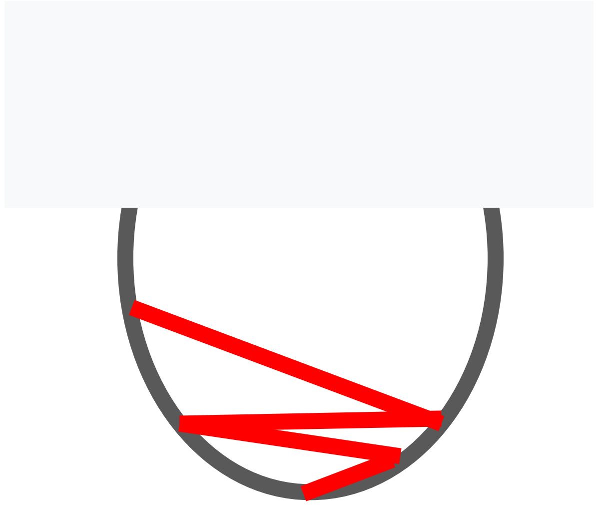


Roughly zero-centered, [-1, 1] range often works well

There are benefits if feature values are small numbers

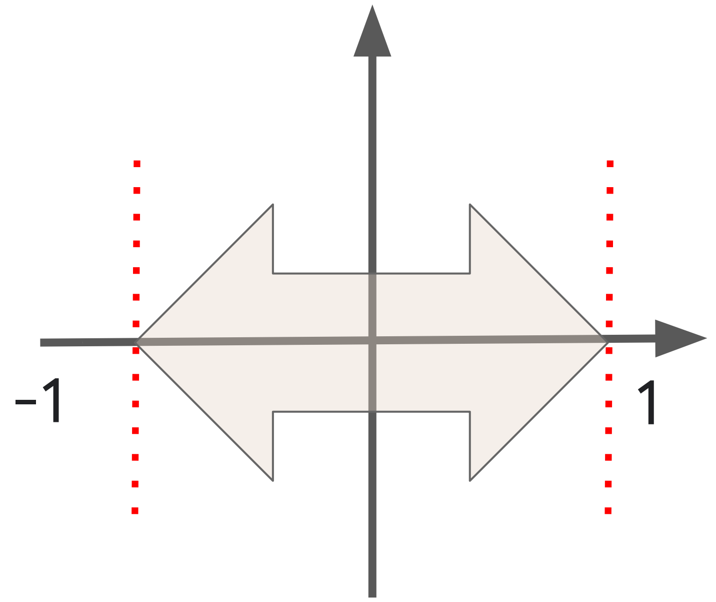


Roughly zero-centered, [-1, 1] range often works well

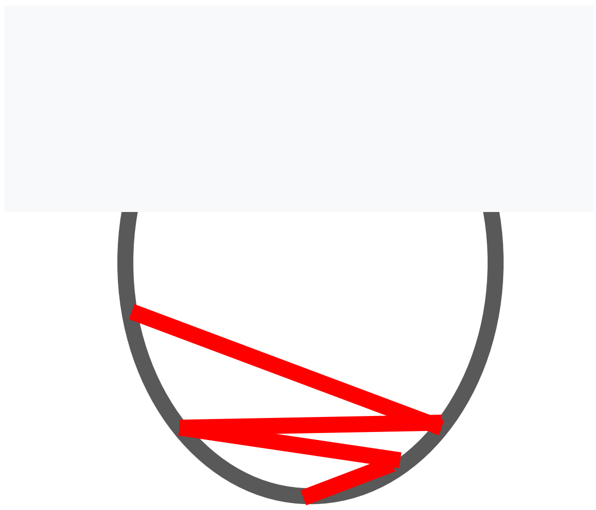


Small magnitudes help gradient descent converge and avoid NaN trap

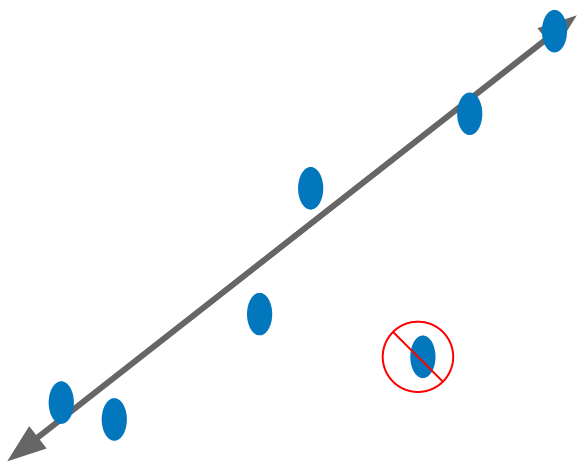
There are benefits if feature values are small numbers



Roughly zero-centered, [-1, 1] range often works well

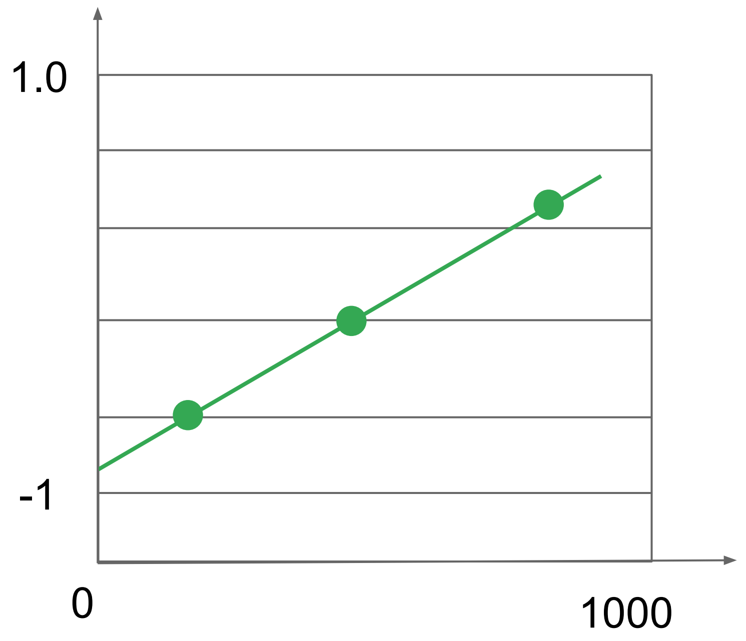


Small magnitudes help gradient descent converge and avoid NaN trap



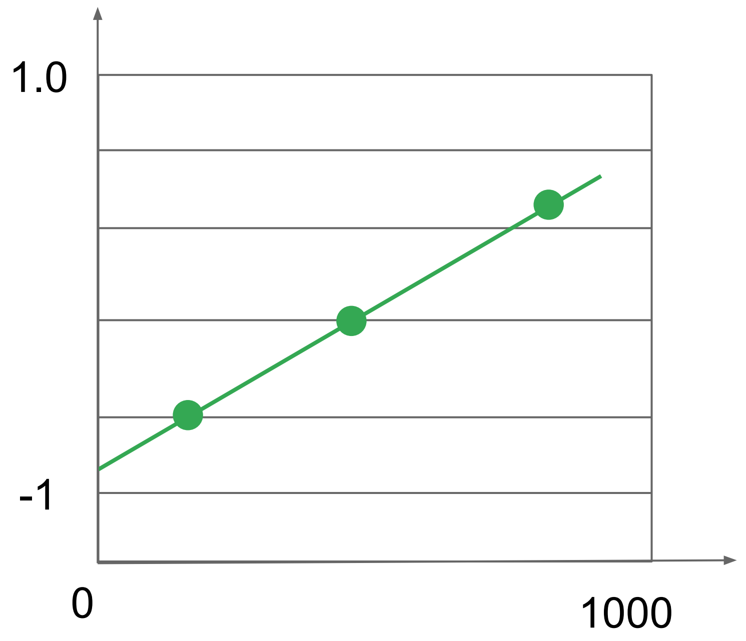
Avoiding outlier values helps with generalization

We can use standard methods to make feature values scale to small numbers

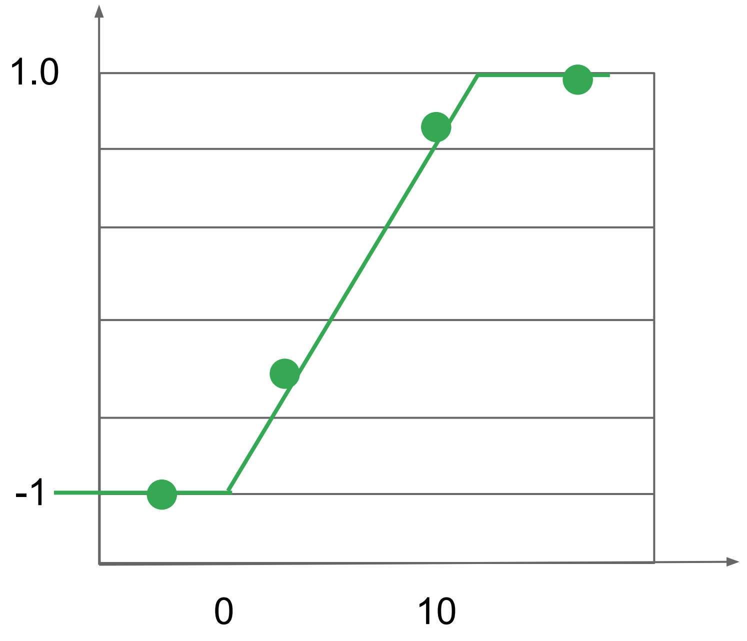


Linear scaling

We can use standard methods to make feature values scale to small numbers

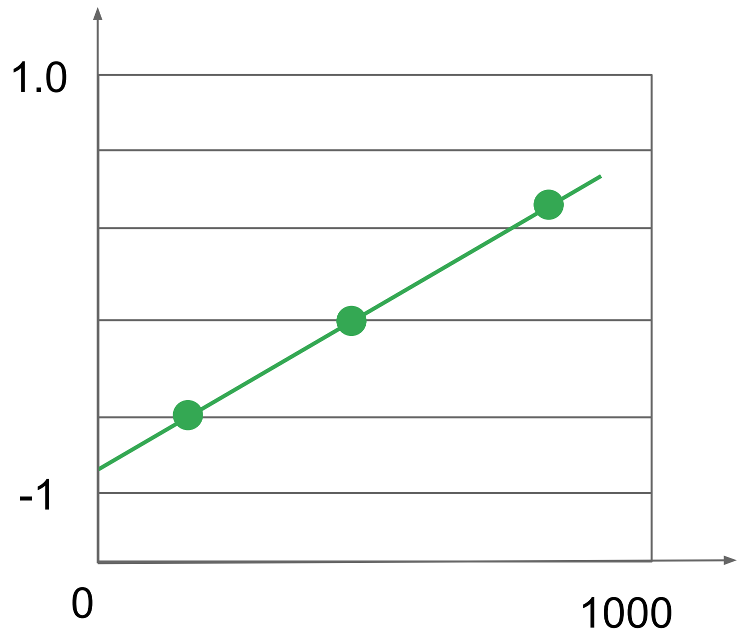


Linear scaling

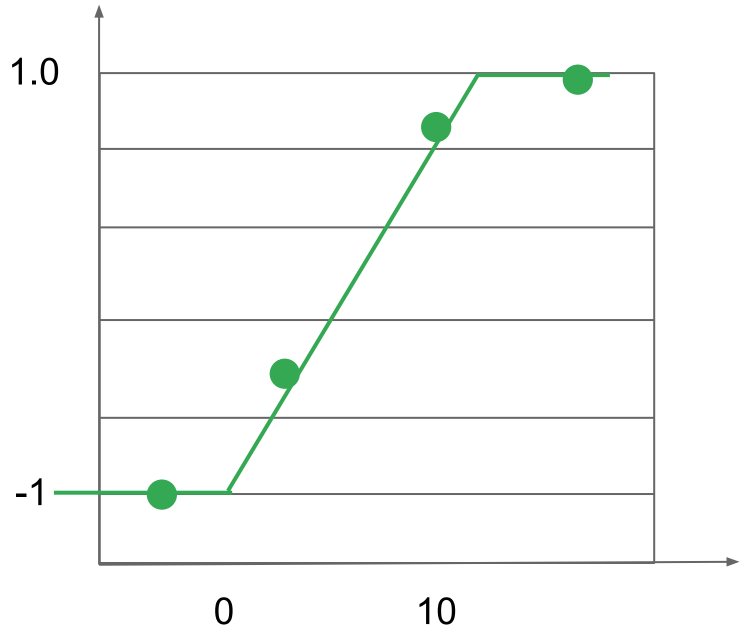


Hard cap (clipping) to max, min

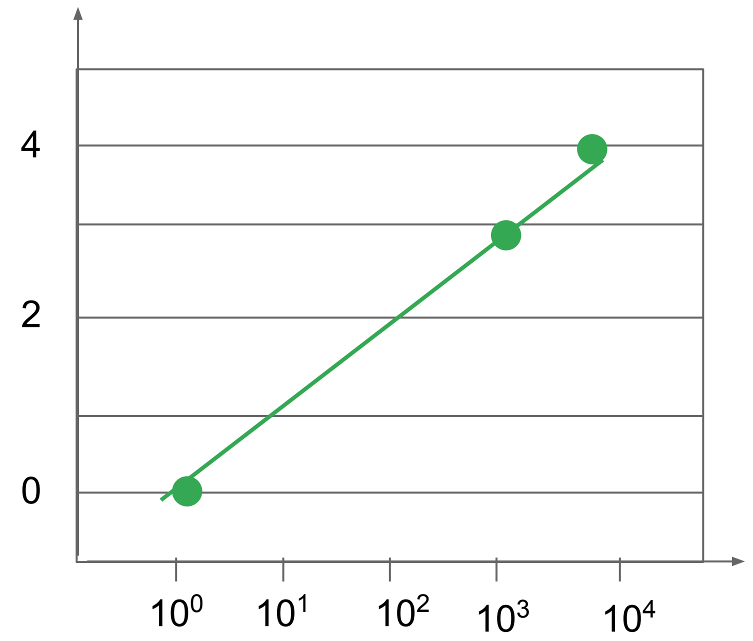
We can use standard methods to make feature values scale to small numbers



Linear scaling



Hard cap (clipping) to max, min



Log scaling

Gradient Descent Debugging Quiz

Which of these is good advice if my model is experiencing exploding gradients?

A. Lower the learning rate
B. Add weight regularization
C. Add gradient clipping
D. Add batch normalization
E. See a doctor
F. C,D
G. A,C,D
H. A,B,C,D

Gradient Descent Debugging Quiz

Which of these is good advice if my model is experiencing exploding gradients?

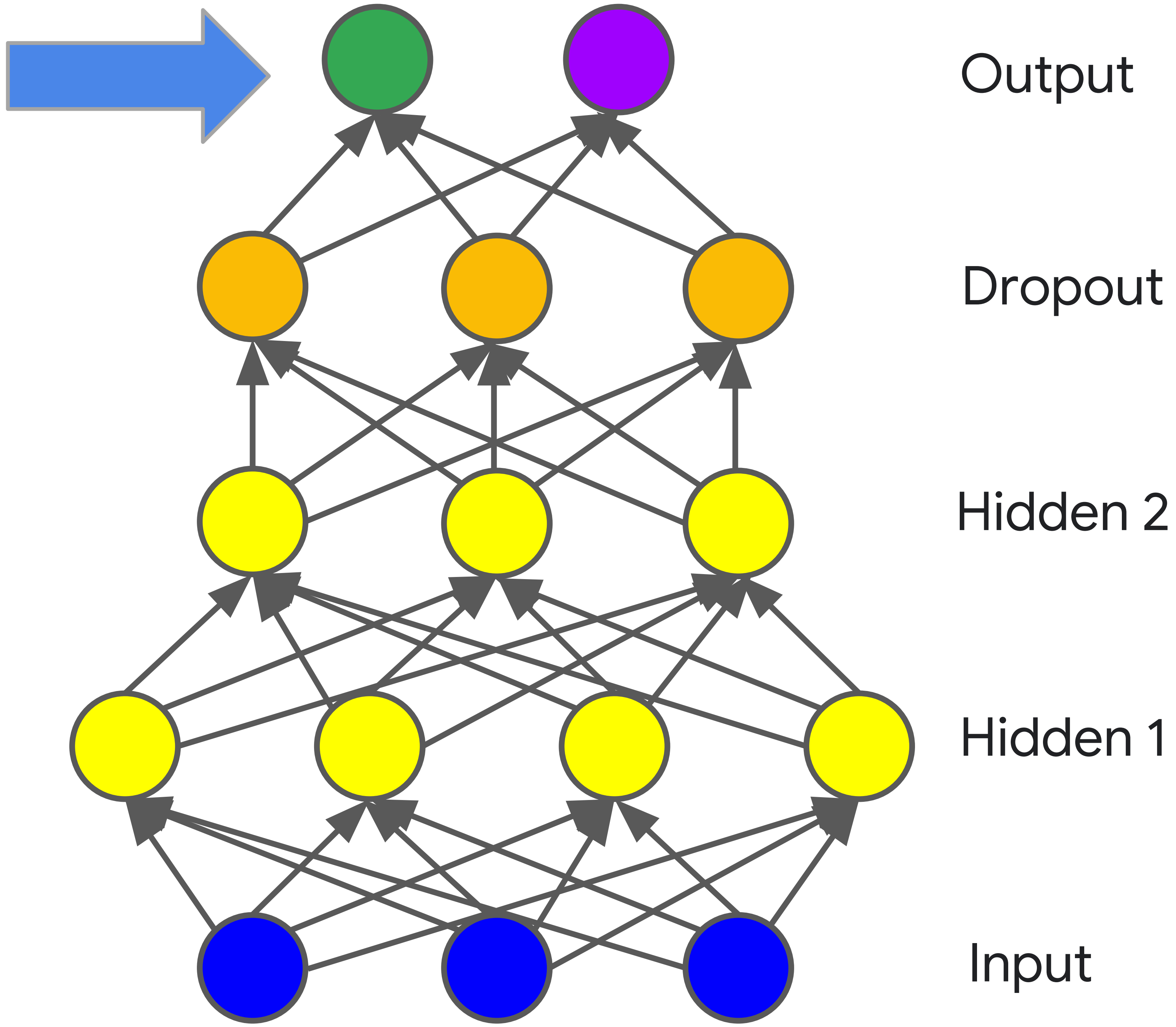
A. Lower the learning rate
B. Add weight regularization
C. Add gradient clipping
D. Add batch normalization
E. See a doctor
F. C,D
G. A,C,D
H. $\boxed{\text{A,B,C,D}}$

Dropout layers are a form of regularization

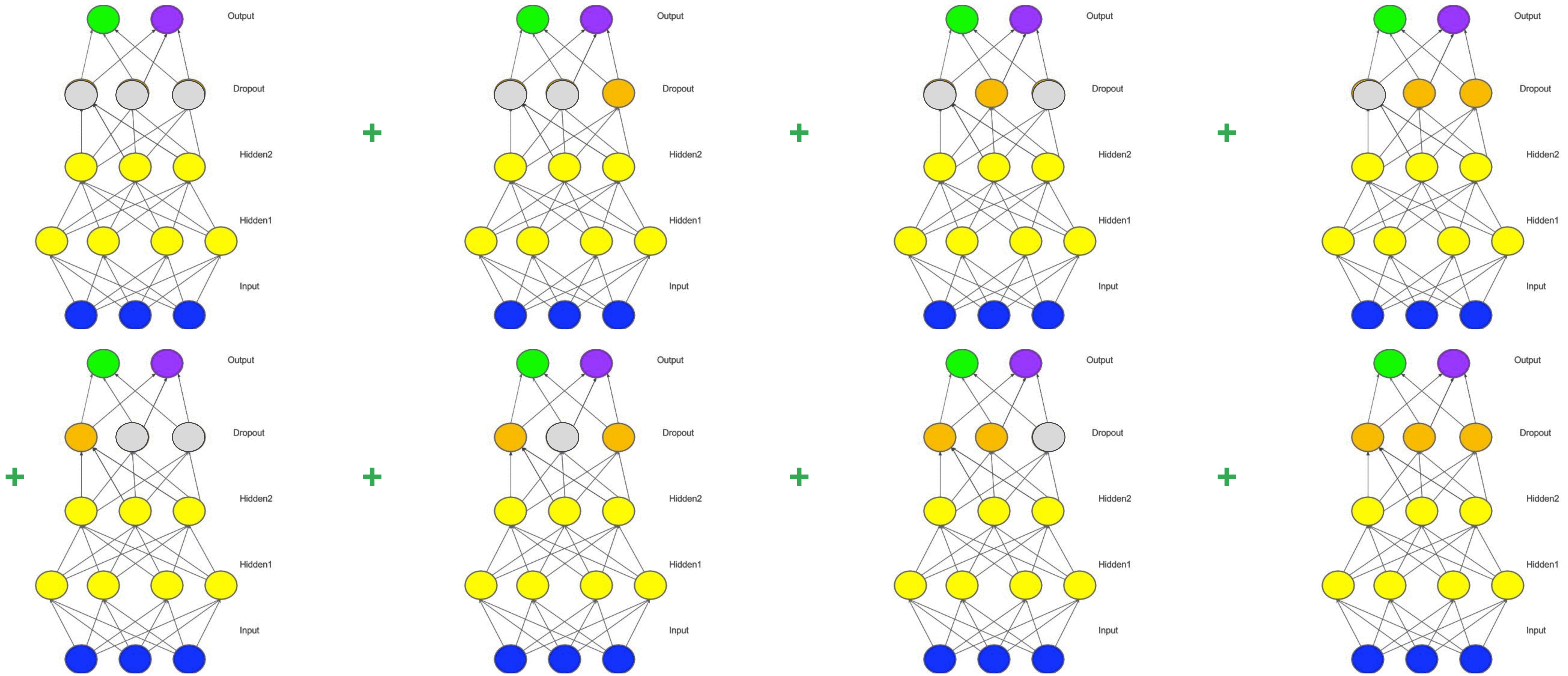
Dropout works by randomly "dropping out" unit activations in a network for a single gradient step

During training only!
In prediction all nodes are kept

Helps learn "multiple paths" --
think: ensemble models,
random forests



Dropout simulates ensemble learning



The more you drop out, the stronger the regularization

0.0 = no dropout regularization

Intermediate values more useful, a value of dropout=0.2 is typical

1.0 = drop everything out! learns nothing

0.0

1.0

Dropout Quiz

Dropout acts as another form of ______. It forces data to flow down ______ paths so that there is a more even spread. It also simulates ______ learning. Don't forget to scale the dropout activations by the inverse of the ______. We remove dropout during ______.

A. Hyperparameter tuning, similar, deep, drop probability, training
B. Hyperparameter tuning, multiple, deep, drop probability, inference
C. Regularization, multiple, ensemble, keep probability, training
D. Regularization, multiple, ensemble, drop probability, inference
E. Regularization, multiple, ensemble, keep probability, inference
F. Hyperparameter tuning, multiple, deep, keep probability, inference
G. Regularization, similar, ensemble, keep probability, inference

Dropout Quiz

Dropout acts as another form of ______. It forces data to flow down ______ paths so that there is a more even spread. It also simulates ______ learning. Don't forget to scale the dropout activations by the inverse of the ______. We remove dropout during ______.

A. Hyperparameter tuning, similar, deep, drop probability, training
B. Hyperparameter tuning, multiple, deep, drop probability, inference
C. Regularization, multiple, ensemble, keep probability, training
D. Regularization, multiple, ensemble, drop probability, inference
E. Regularization, multiple, ensemble, keep probability, inference
F. Hyperparameter tuning, multiple, deep, keep probability, inference
G. Regularization, similar, ensemble, keep probability, inference

Lab

Using Neural Networks to build a ML model

Ryan Gillard

Lab: Using Neural Networks to build ML model

In this lab, you will use the DNNRegressor class in TensorFlow to predict median housing price

1. The data is based on 1990 census data from California.

2. This data is at the city block level, so these features reflect the total number of rooms in that block, or the total number of people who live on that block, respectively

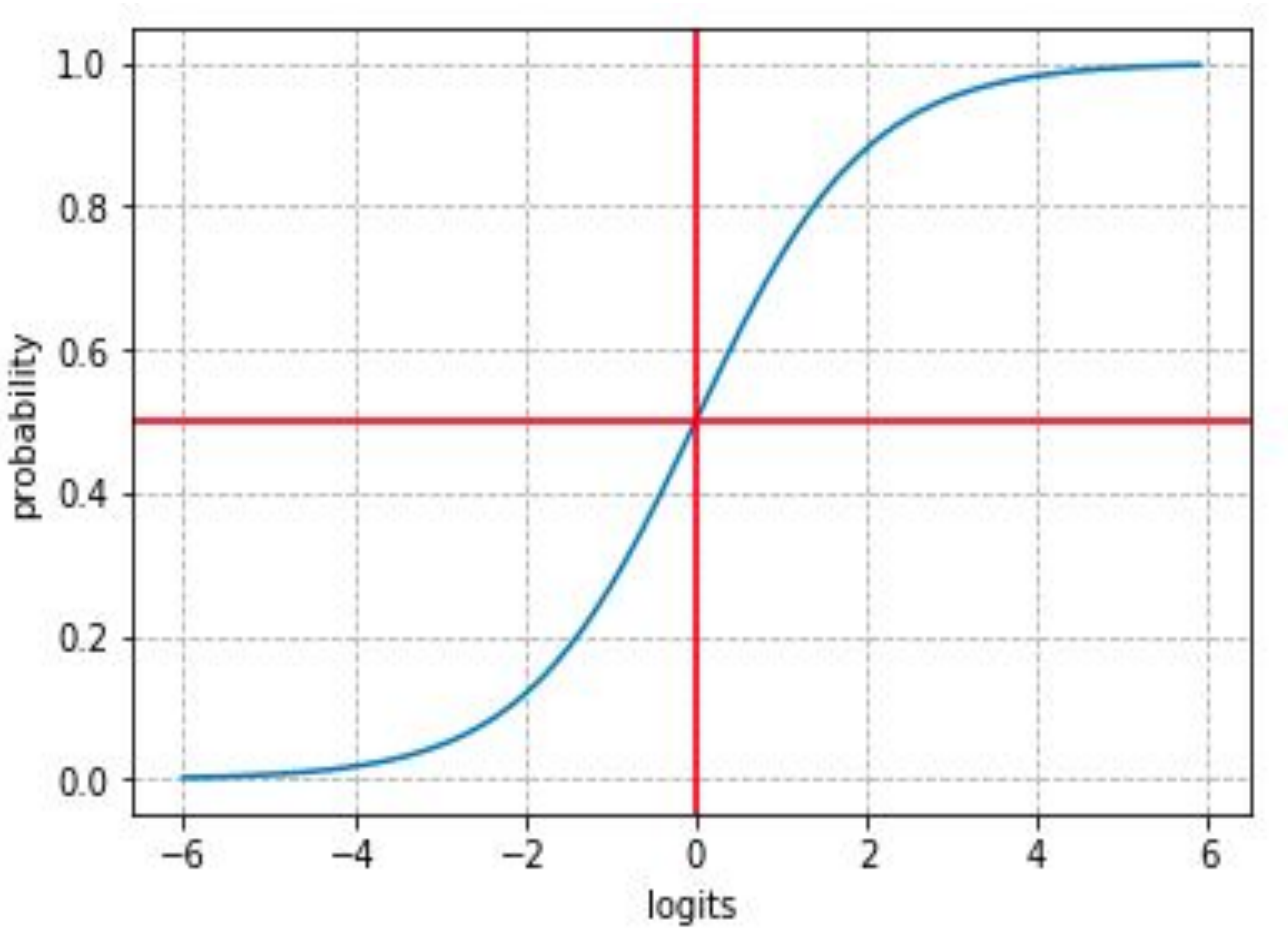
Lab: Using Neural Networks to build ML model

Camtasia

Multi-class neural nets

Ryan Gillard

Logistic regression provides useful probabilities for binary-class problems



There are lots of multi-class problems

Pit

Stalls

Circle

Suite

How do we extend the logits idea to multi-class classifiers?

There are lots of multi-class problems

Model 1	
Positive Class	Negative Class
Pit	Other

Model 2	
Positive Class	Negative Class
Stalls	Other

Model 3	
Positive Class	Negative Class
Circle	Other

Model 4	
Positive Class	Negative Class
Suite	Other

There are lots of multi-class problems

Model 1	
Positive Class	Negative Class
Pit	Stalls

Model 2	
Positive Class	Negative Class
Pit	Circle

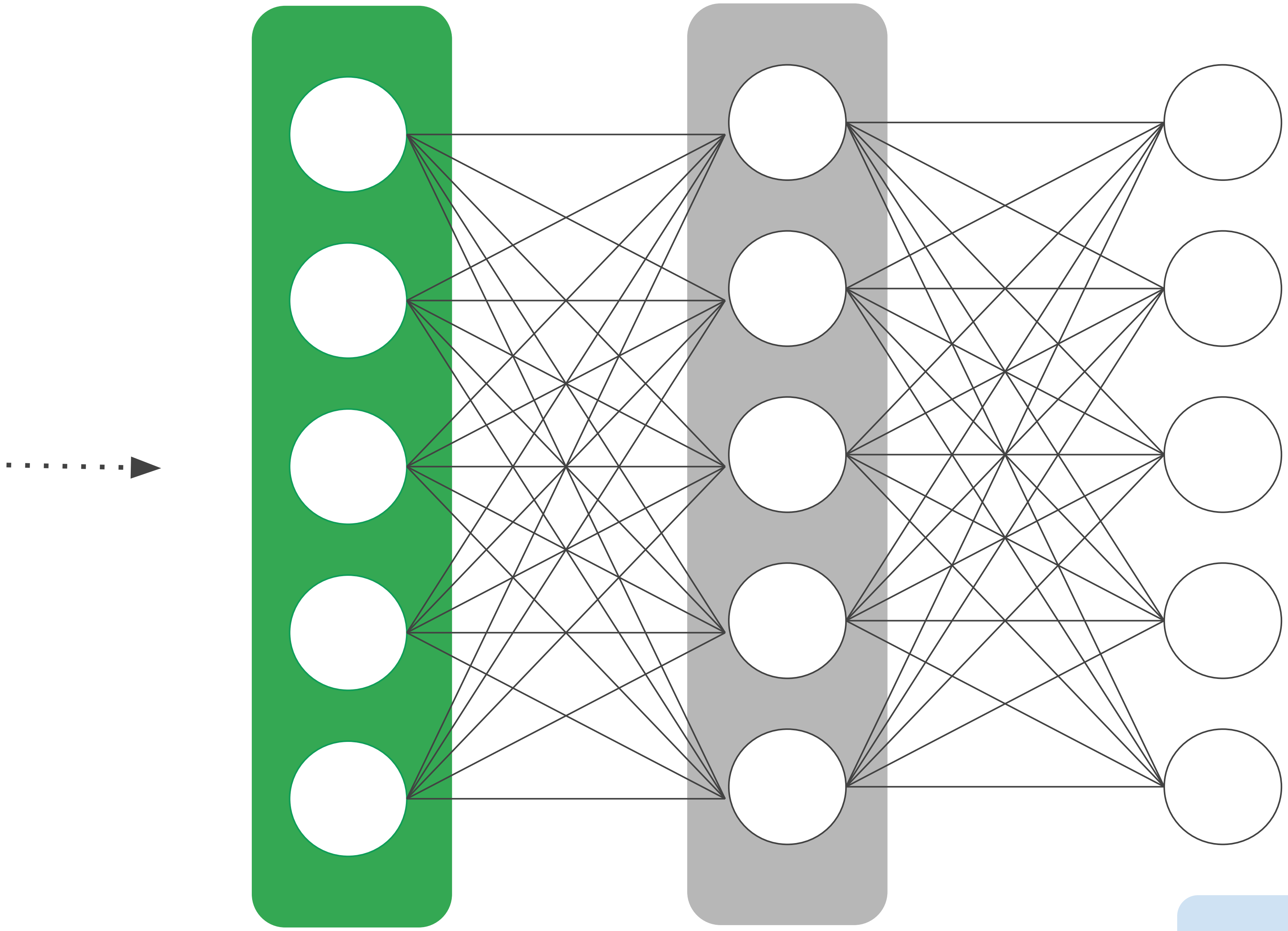
Model 3	
Positive Class	Negative Class
Pit	Suite

Model 4	
Positive Class	Negative Class
Stalls	Circle

Model 5	
Positive Class	Negative Class
Stalls	Suite

Model 6	
Positive Class	Negative Class
Circle	Suite

Idea: Use separate output nodes for each possible class



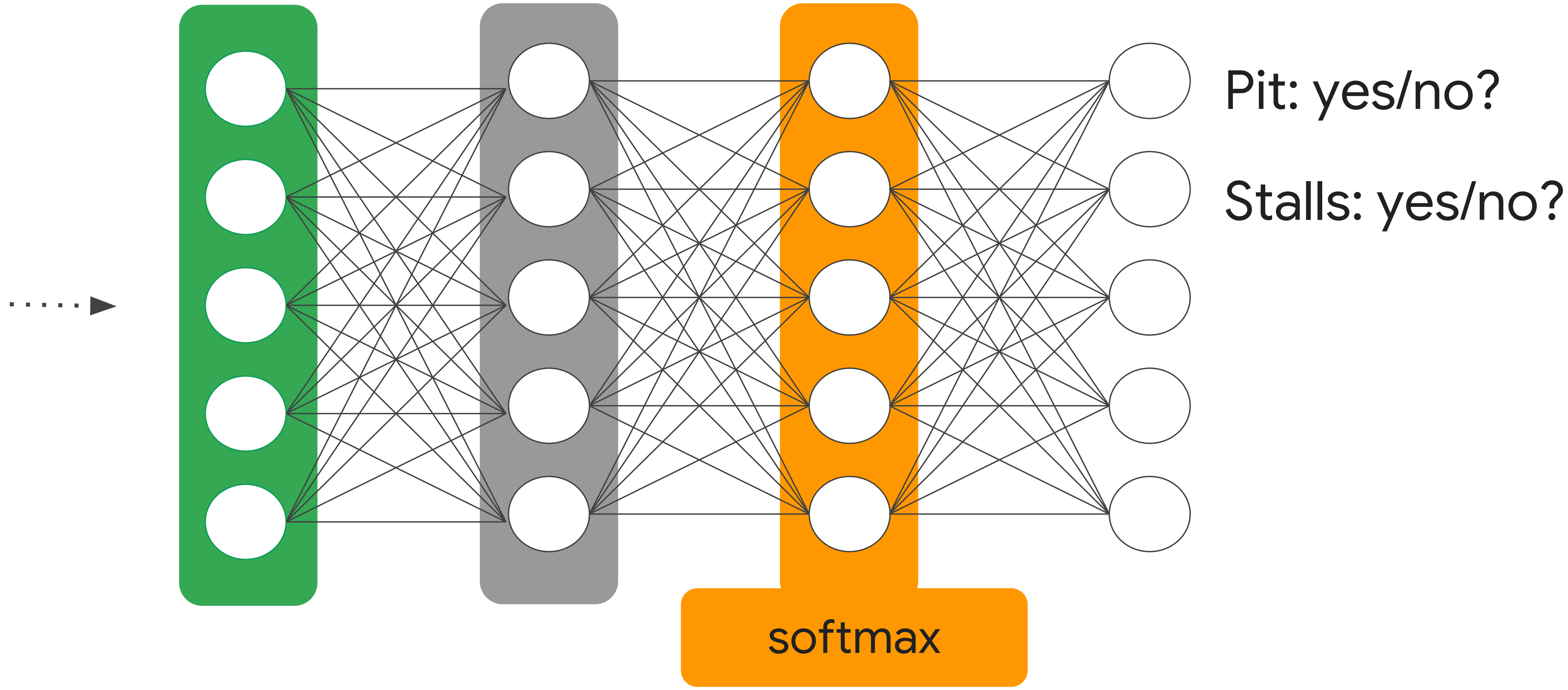
Pit: yes/no?

Stalls: yes/no?

one-vs-all

Add additional constraint, that total of outputs = 1.0

$$p(y=j|\mathbf{x}) = \frac{exp(\mathbf{w}_j^T\mathbf{x}+b_j)}{\sum_{k\in K} exp(\mathbf{w}_k^T\mathbf{x}+b_k)}$$



Use one softmax loss for all possible classes

```
logits = tf.matmul(...) # logits for each output node
                          -> shape = [batch_size, num_classes]
labels = ...             # one-hot encoding in [0, num_classes)
                          -> shape = [batch_size, num_classes]
loss = tf.reduce_mean(
   tf.nn.softmax_cross_entropy_with_logits_v2(
                                    logits, labels) -> shape = [batch_size]
)
```

Use one softmax loss for all possible classes

```
logits = tf.matmul(...) # logits for each output node
                          -> shape = [batch_size, num_classes]
labels = ...             # index in [0, num_classes)
                          -> shape = [batch_size]
loss = tf.reduce_mean(
   tf.nn.sparse_softmax_cross_entropy_with_logits(
                                      logits, labels) -> shape = [batch_size]
)
```

Use softmax only when classes are mutually exclusive

“Multi-Class, Single-Label Classification”

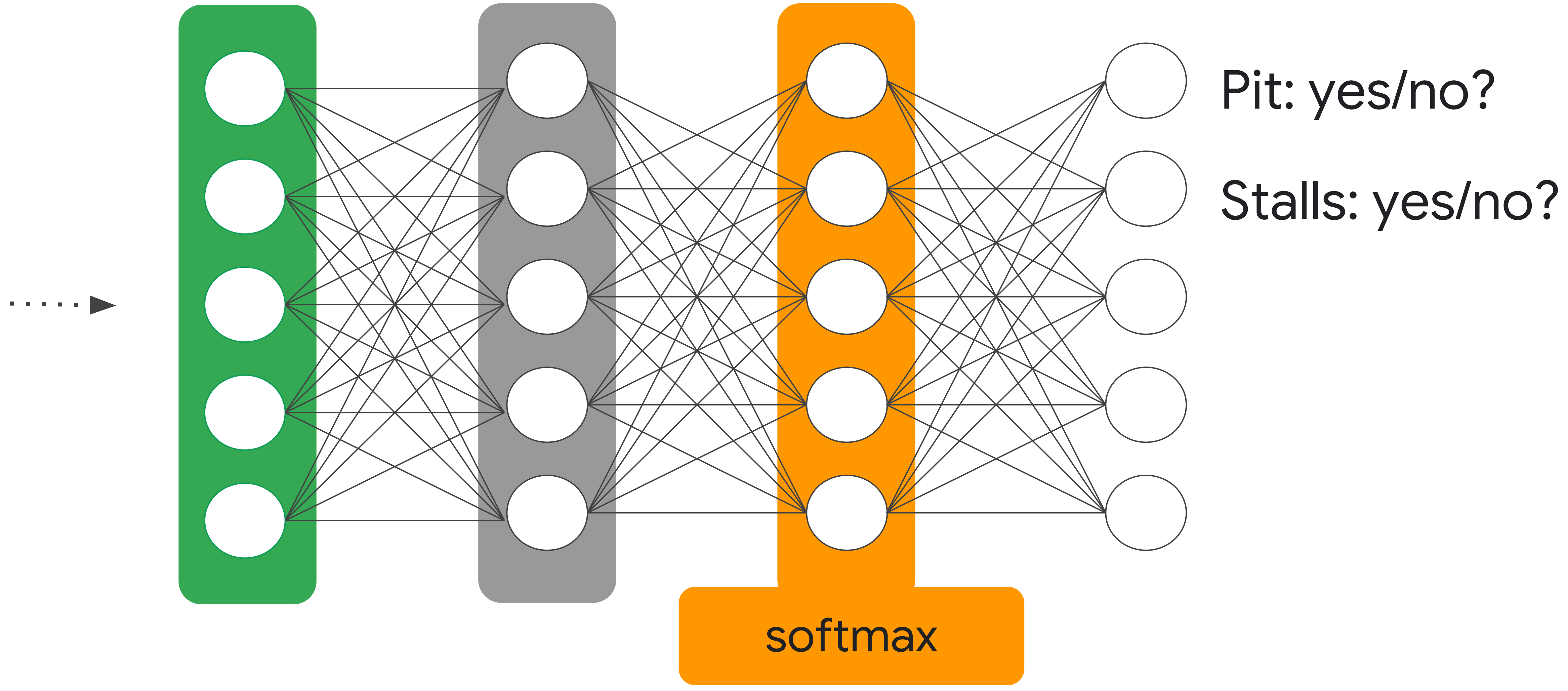
An example may be a member of only one class.

Are there multi-class settings where examples may belong to more than one class?

```
tf.nn.sigmoid_cross_entropy_with_logits(
                                  logits, labels) -> shape = [batch_size, num_classes]
```

If you have hundreds or thousands of classes, loss computation can become a significant bottleneck

Need to evaluate every output node for every example



Approximate versions of softmax exist

Candidate Sampling calculates for all the positive labels, but only for a random sample of negatives: tf.nn.sampled_softmax_loss

Noise-contrastive approximates the denominator of softmax by modeling the distribution of outputs: tf.nn.nce_loss

Softmax Quiz

For our classification output, if we have both mutually exclusive labels and probabilities, we should use ______. If the labels are mutually exclusive, but the probabilities aren't, we should use ______. If our labels aren't mutually exclusive, we should use ______.

I. tf.nn.sigmoid_cross_entropy_with_logits
II. tf.nn.sparse_softmax_cross_entropy_with_logits
III. tf.nn.softmax_cross_entropy_with_logits_v2

A. III, II, I
B. I, II, III
C. III, I, II
D. II, III, I

Softmax Quiz

For our classification output, if we have both mutually exclusive labels and probabilities, we should use ______. If the labels are mutually exclusive, but the probabilities aren't, we should use ______. If our labels aren't mutually exclusive, we should use ______.

I. tf.nn.sigmoid_cross_entropy_with_logits

II. tf.nn.sparse_softmax_cross_entropy_with_logits

III. tf.nn.softmax_cross_entropy_with_logits_v2

A. III, II, I
B. I, II, III
C. III, I, II
D. II, III, I

cloud.google.com