Fundamentals of Artificial Neural Networks

Adnane Ait Nasser | AI Research Consultant at ACENET

A regional partner of the
Digital Research Alliance of Canada

Learning Objectives

- Describe the structure and function of Artificial Neural Networks (ANNs).
- Understand when to use ANNs over classical machine learning models.
- Build and train a basic ANN using Python (TensorFlow or PyTorch).
- Explain key training concepts, including activation functions, loss, optimization, and overfitting.
- Differentiate between specialized neural network architectures, including:
  - CNNs for image and spatial data,
  - RNNs for sequential data,
  - Autoencoders for compression and anomaly detection,
  - GANs for synthetic data generation.
- Understand the limitations of RNNs and how they motivated the development of **Transformer-based models**, which will be explored in the next session on LLMs.

Recap

1. ML vs AI
2. Supervised Learning – Labeled Data
   a. **Classification** – Classifying objects into classes (e.g., cats and dogs, healthy and unhealthy, flower species, etc.).
   b. **Regression** – Predicting continuous values (e.g., house price).
3. Unsupervised Learning – Working with unlabeled Data
   a. **Clustering** – (e.g., discovering patterns or groupings without predefined categories).
4. Evaluation Metrics – Accuracy, Recall, Precision, etc.
5. Workflow and Best practices.
6. Hands-on examples of ML models.

Why Do We Need Artificial Neural Networks (ANNs)?

Traditional ML models work well when the relationships between input features and target labels are relatively simple or can be captured using linear or manually engineered features. **But** these models hit their limits when faced with:

- **High-dimensional data**
- **Non-linear and hierarchical relationships**
- **Unstructured data**
- **Scalability**

“In many real-world problems, crafting the right features becomes too hard or even impossible.”

Overview

- Introduction to Artificial Neural Networks (ANNs):
  - Structure: input, hidden, and output layers.
  - **Key concepts:** neurons, weights, activation functions, loss, epochs, overfitting, Backpropagation.
  - Building and training a simple ANN using Python (PyTorch).
- Specialized Neural Network Architectures (FFNs, CNNs, RNNs, Autoencoders, and GANs).
- Conceptual bridge to Transformers.

What Is an ANN?

- A method of AI that teaches computers to process data in a way that is inspired by the biological neural networks that constitute human brain.

- A Type of Machine Learning process, called Deep Learning based on a collection of connected units or nodes called "artificial neurons," which loosely model the neurons in a biological brain.

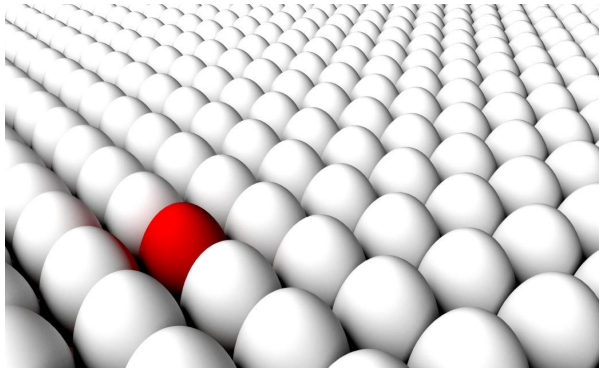
- Each connection, like the synapses, can transmit a signal from one artificial neuron to another.

Computer Vision

Image Recognition

Medical Diagnosis

Anomaly detection

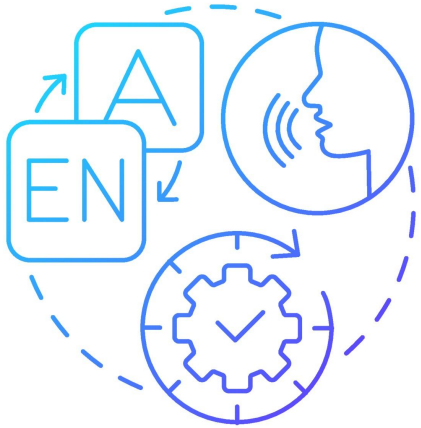


ACENET
accelerate discovery

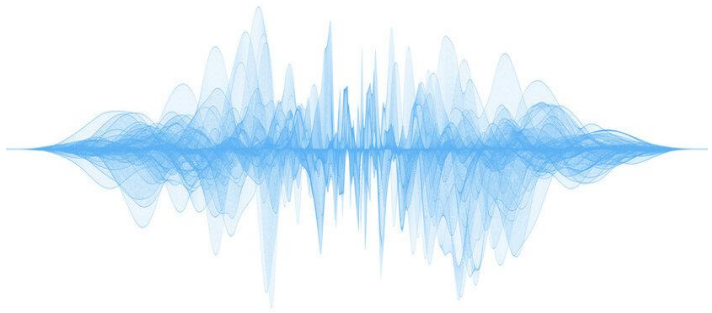
A regional partner of the
Digital Research Alliance of Canada

Natural Language Processing

Real Time Translation



Voice Recognition



Virtual Assistants

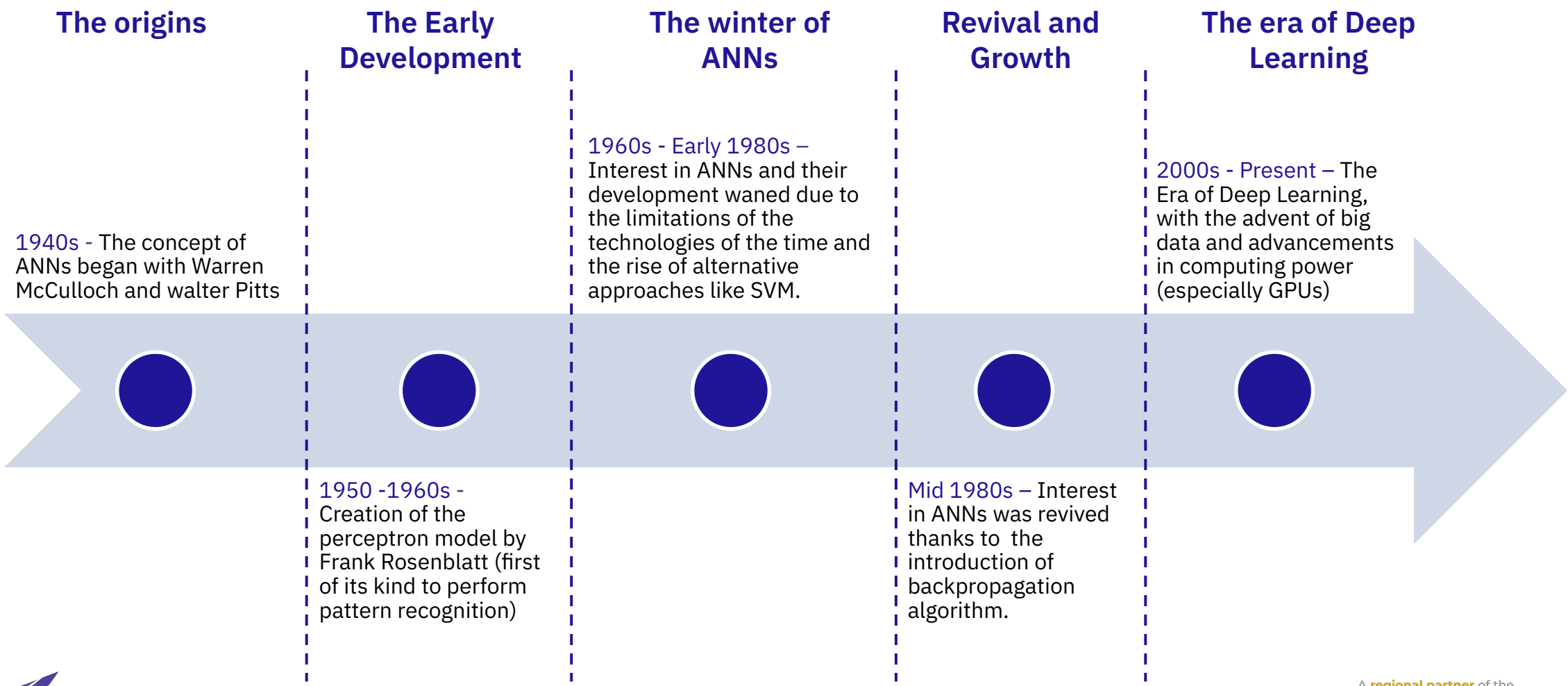
Recommender Systems

Targeted Advertising

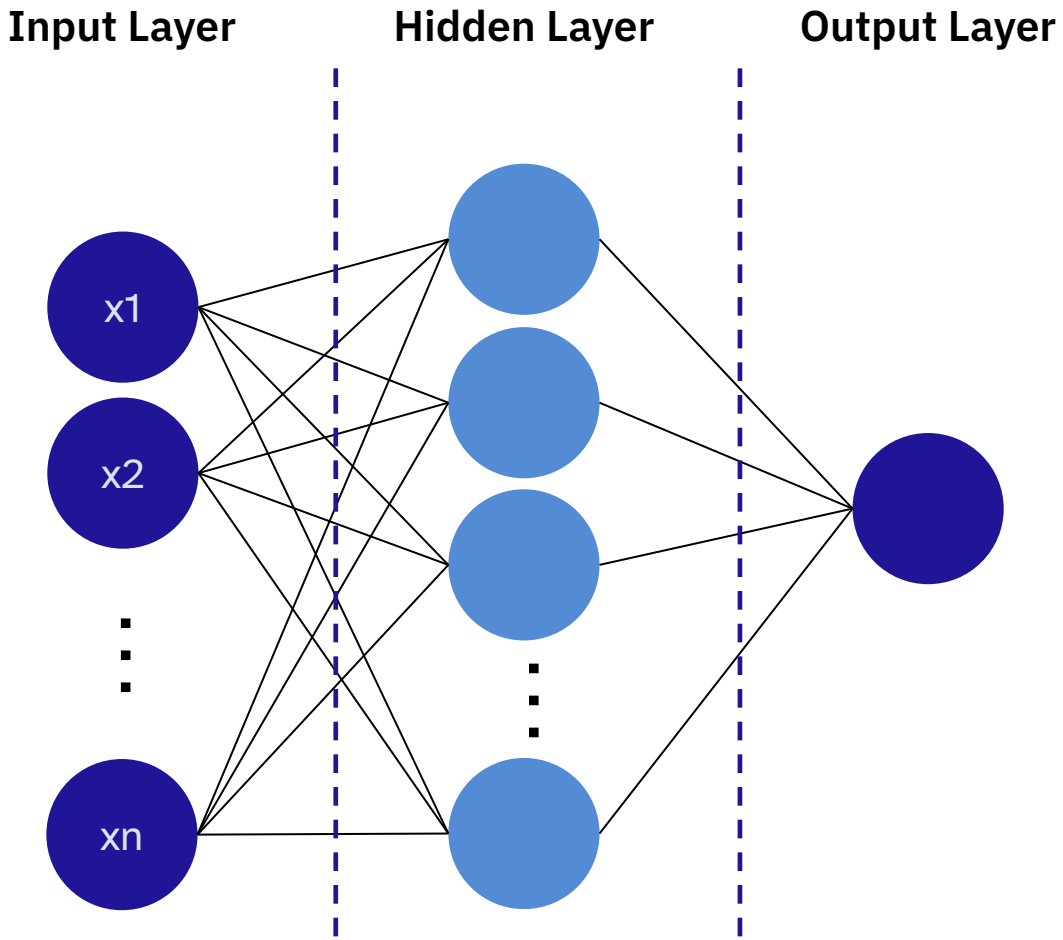
Content Curation

Recommender Systems

History of ANNs



Anatomy of Artificial Neural Network (ANNs)



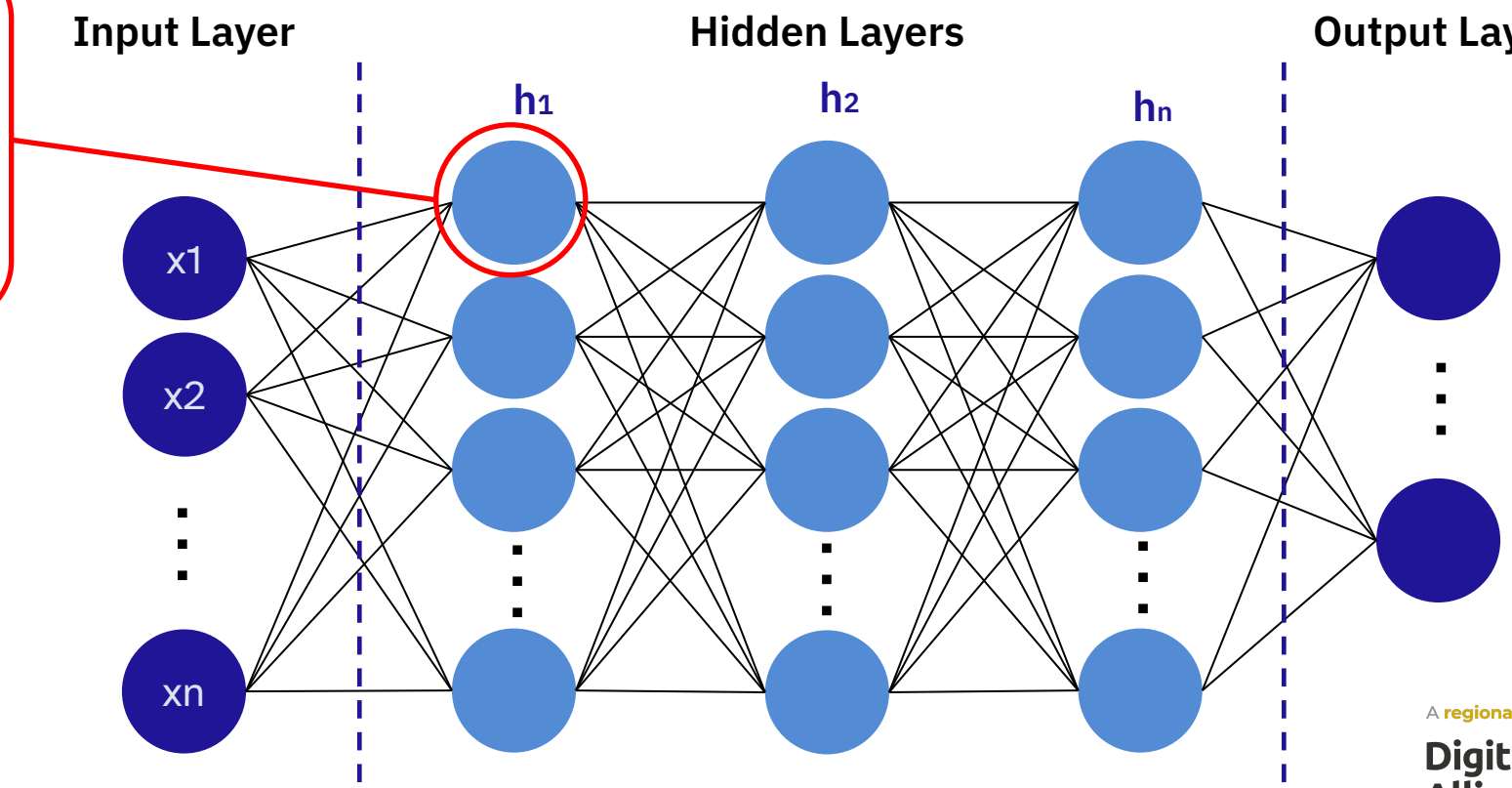
ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Anatomy of Artificial Neural Networks

A class of models that are built with layers. Commonly used types of neural networks including Convolutional Neural Networks (CNNs) for images and Recurrent Neural Networks (RNNs) for sequences.

Neurons are the core components of most neural networks. They are simple functions that take input values and produce a single output value. Each neuron also has a bias, which determines its "default" behavior.



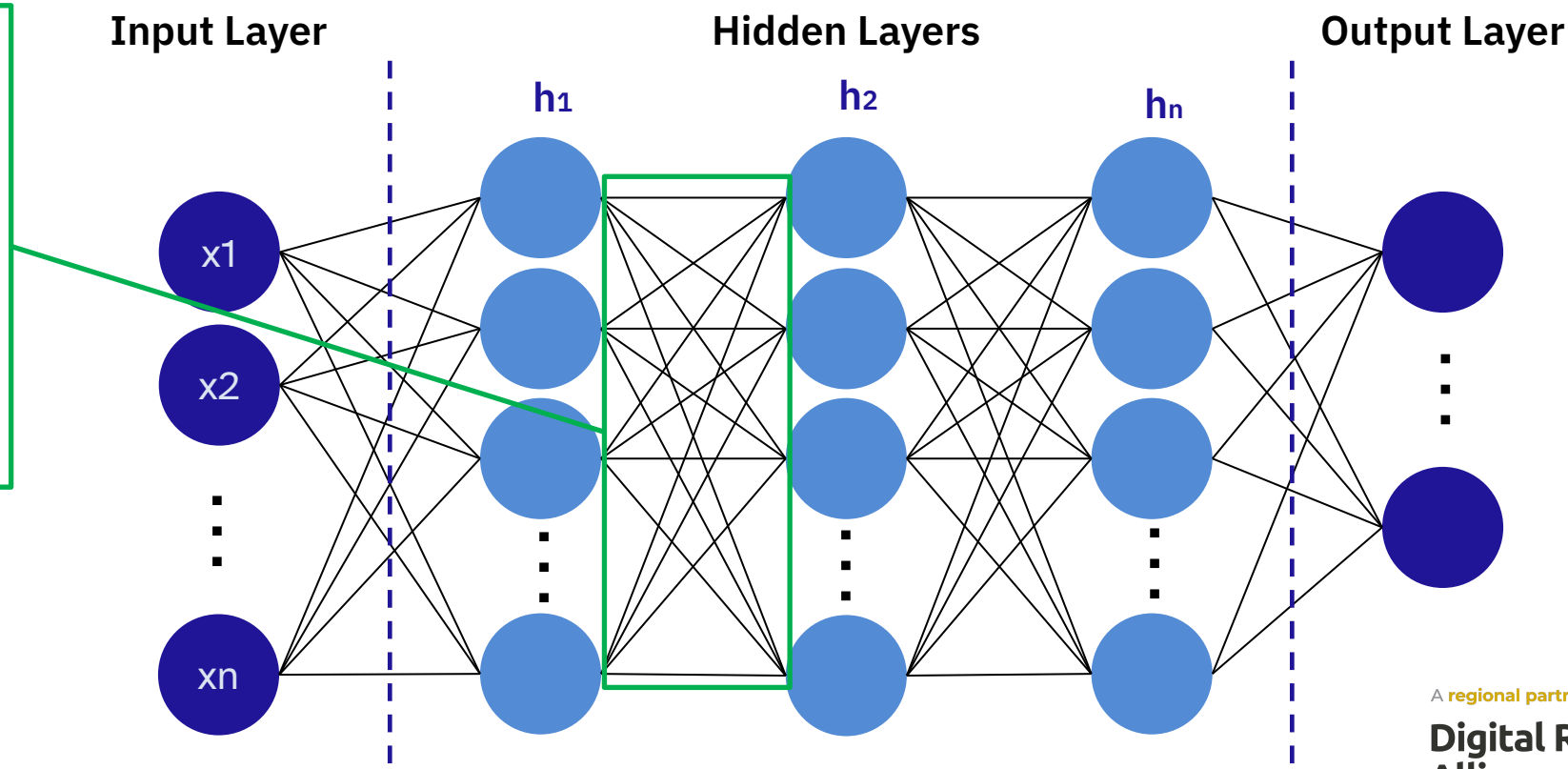
ACENET accelerate discovery

A regional partner of the Digital Research Alliance of Canada

Anatomy of Artificial Neural Networks

Composed of simple operations and weights applied to data both simultaneously and successively to extract patterns that depend on the data.

The connections between neurons indicate which neurons pass what information to the next layer in the network.
The weight (Strength) of connections determines the path taken through the network, and thus, the output of the network.

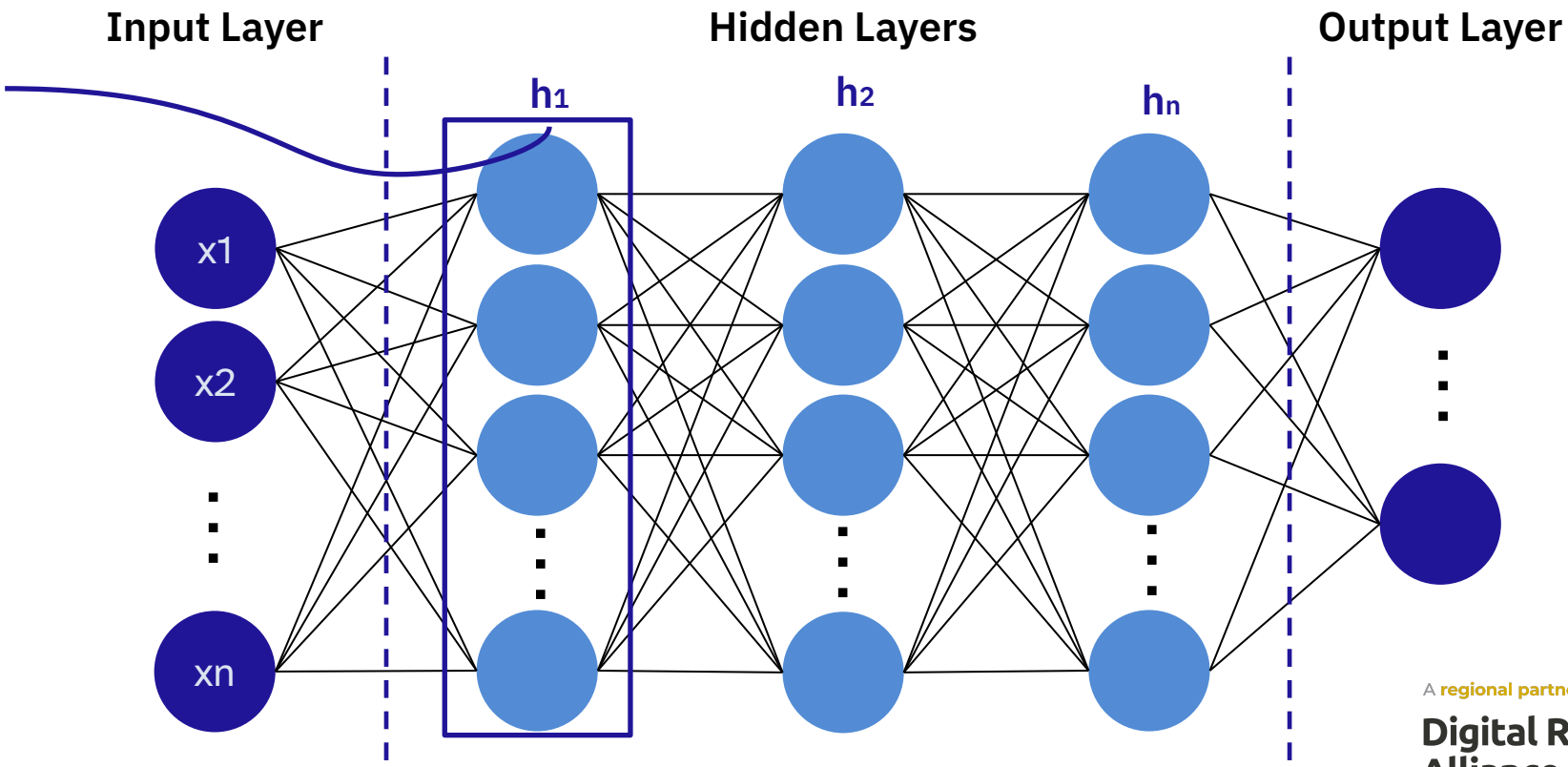


ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Anatomy of Artificial Neural Networks

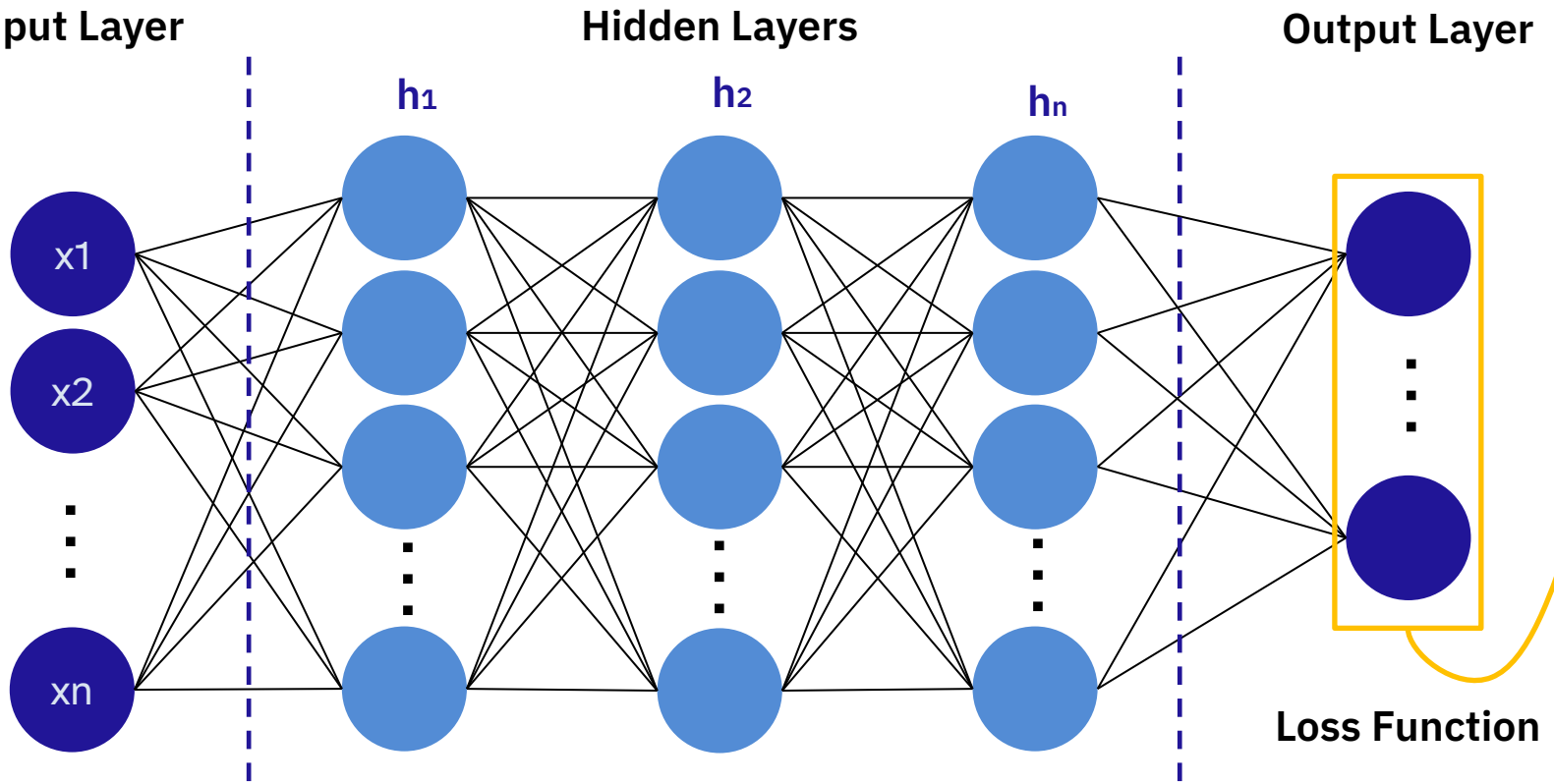
Layers are groups of neurons that perform operations simultaneously. In general, the more layers a network has, the more flexible it becomes — allowing it to learn more complex patterns. However, this increased flexibility also raises the risk of overfitting. Networks with many layers are referred to as “deep” neural networks.



ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Anatomy of Artificial Neural Networks

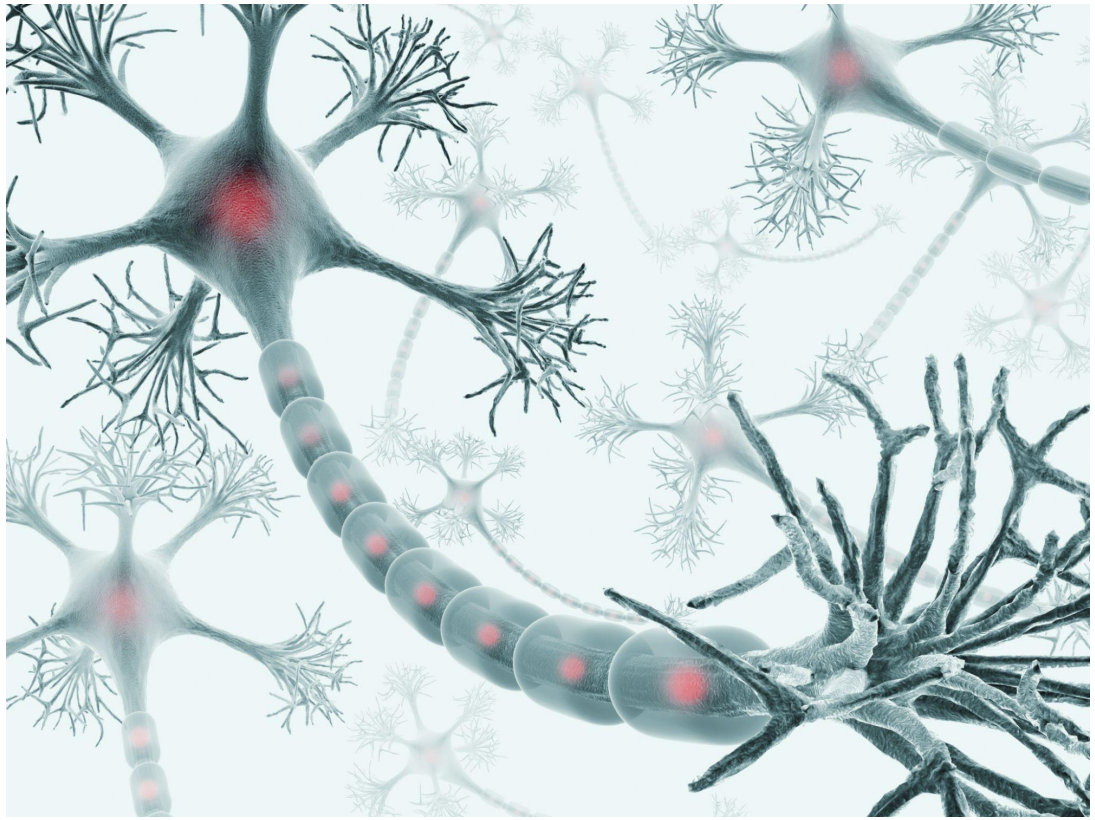


The Loss function is the target we set for the network to aim for. It is set by the user, and is the error we attempt to minimize.

We use this as our metric to judge if the neural network is getting better or worse as it trains.

Neurons

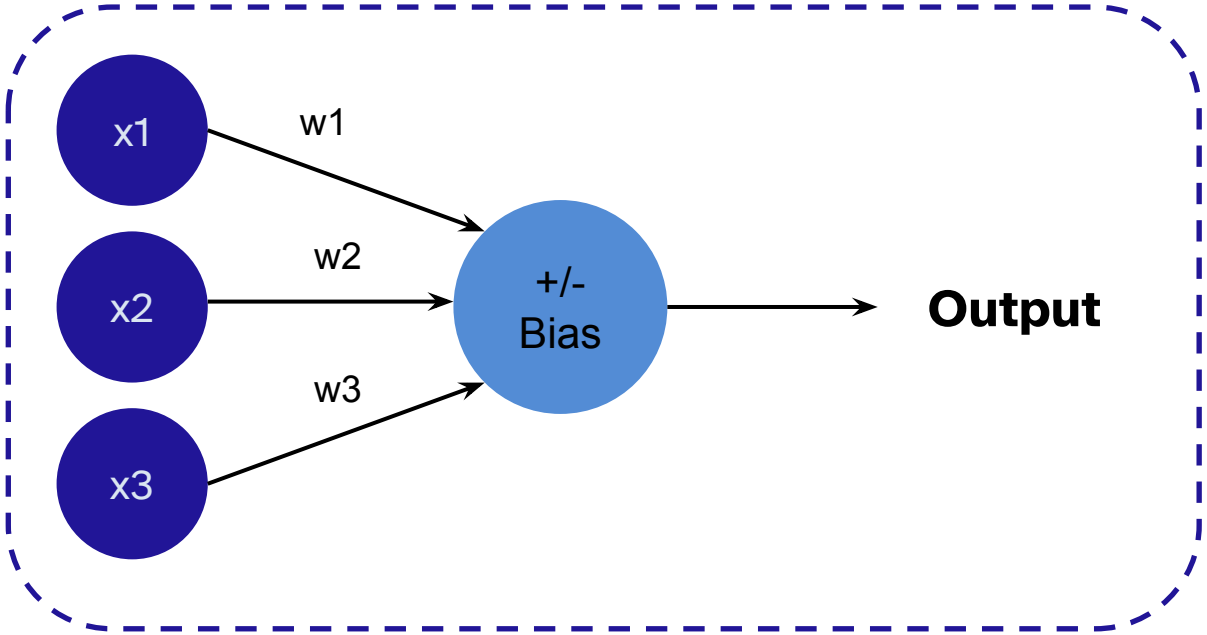
- Neurons are similar to logic gates that take in some number, and output a bool (1/0), or another number.
- Given enough neurons, and proper input, a neural net can assemble itself using backpropagation (stay tuned).
- Individually, neurons aren't much use, but when there are many, properly trained, they become very powerful.



Anatomy of Neurons

Neuron Output = $f(\sum(weight * input) + bias)$

- ***f*** is the activation function, one of several simple mathematical functions.
- **Weight** indicates how influential an input is to a neuron.
- **Biases** indicate the default output from a neuron.

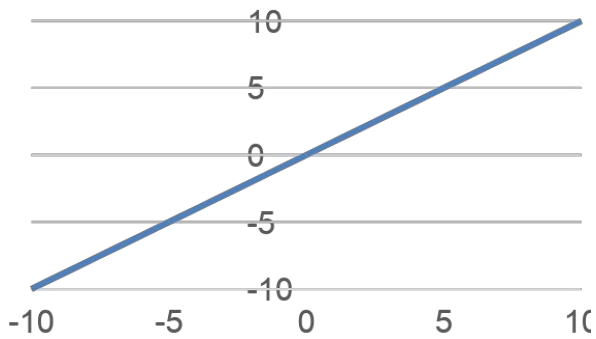


Activation Functions

Linear

$$\hat{y} = wx + b$$

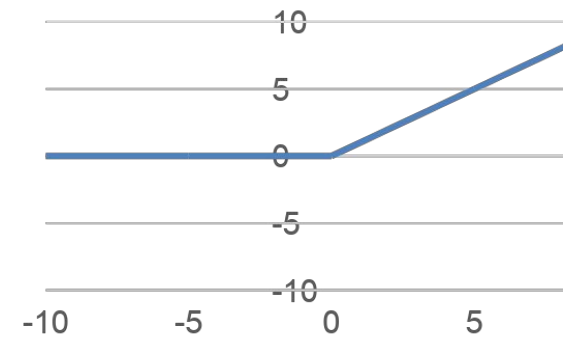
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$

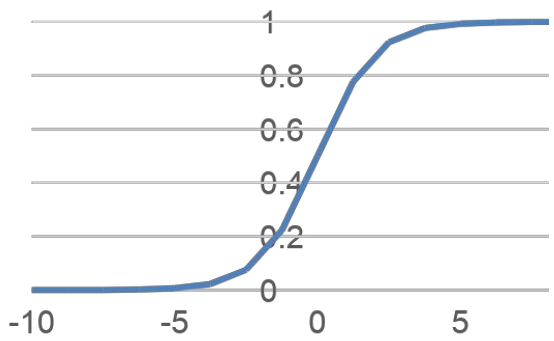
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid

$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```



SoftMax Activation Function

- SoftMax Neurons are a special kind of layer, where they take in all the output values from the network and convert them into weighted probabilities for prediction.
- The largest values are considered the predicted category and can be output alongside a confidence score.

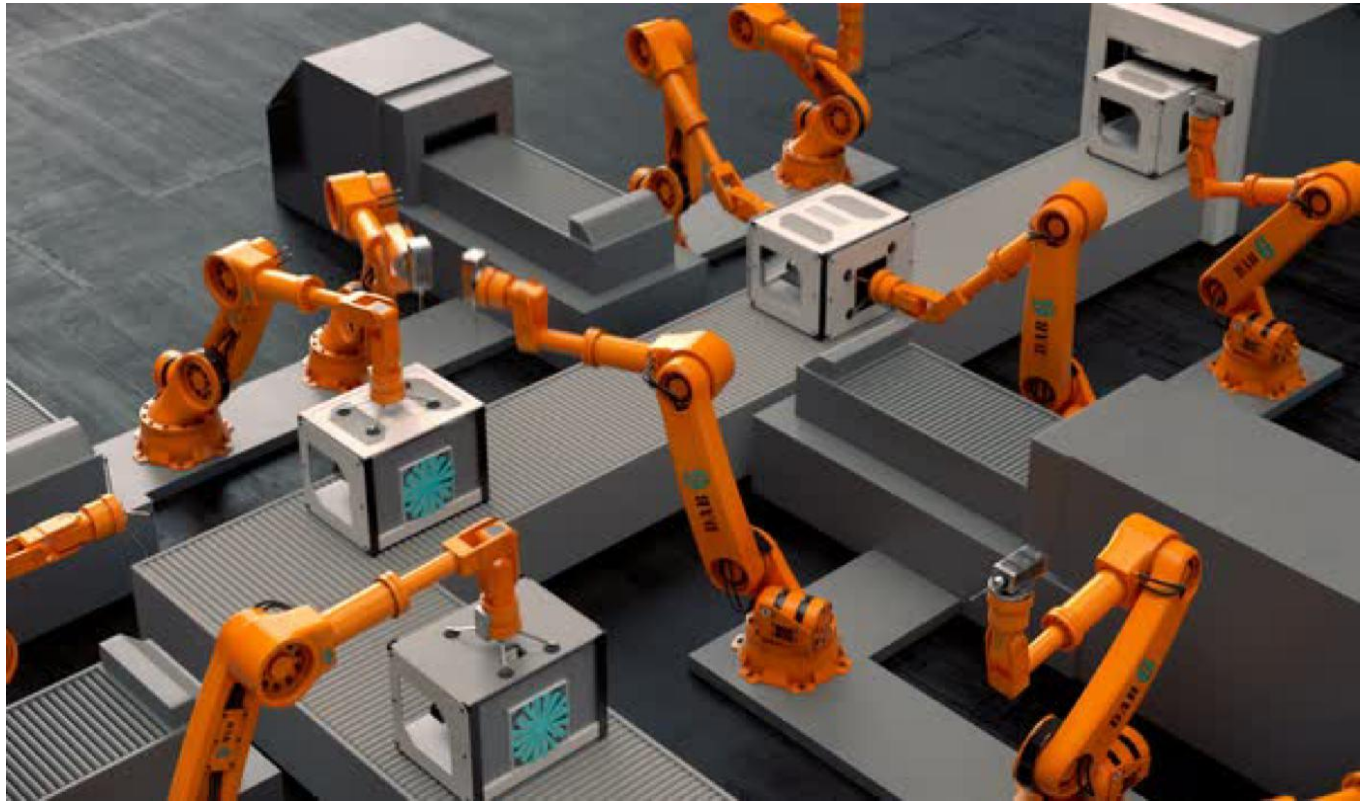
$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

- Note: SoftMax is differentiable, and therefore can be used with backpropagation for training purposes. Some algorithms convert SoftMax into binary categories or just go with the largest output.

Feed Forward Neural Networks (FNNs)

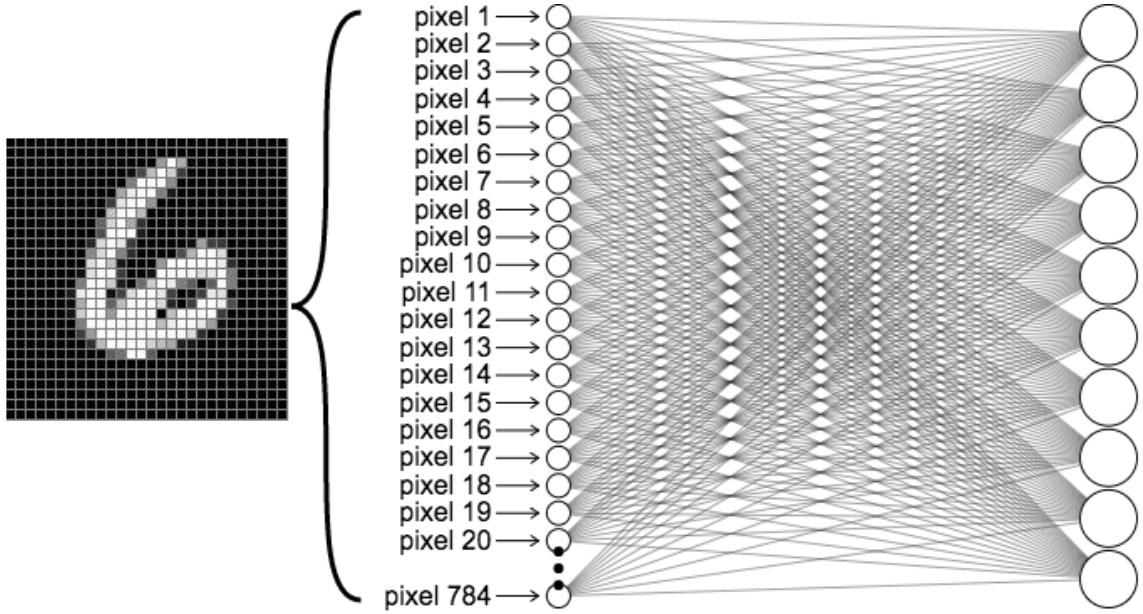
Feed Forward Neural Networks (FNNs)

- Feed forward networks are the simplest neural networks.
- A signal is input from the data that is being processed.
- The neural network performs rapid calculations to gain some insight from the data or classify it.
- They operate in one direction.



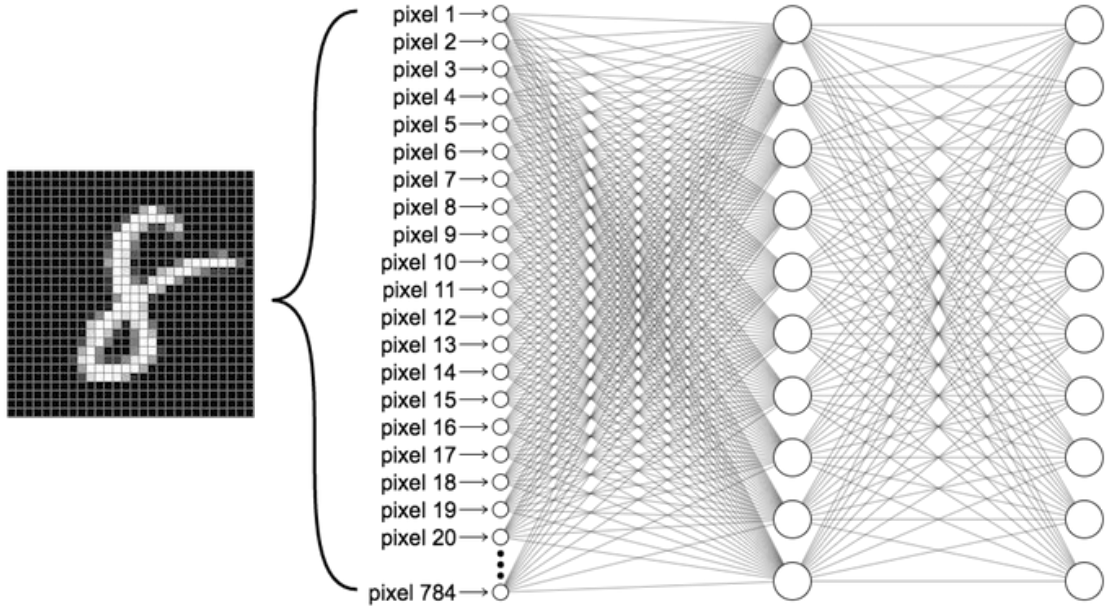
Feed Forward Neural Networks (FNNs)

- MNIST is a dataset of digitized handwritten numbers from 0-9.
- Pixels have values from 0-255.
- MNIST Net ends in 10 sigmoid neurons, [0 or 1], for each digit.
  - Takes 784 (28^2) input integers.
  - Gives 10 Boolean values. (nine 0's and a 1)



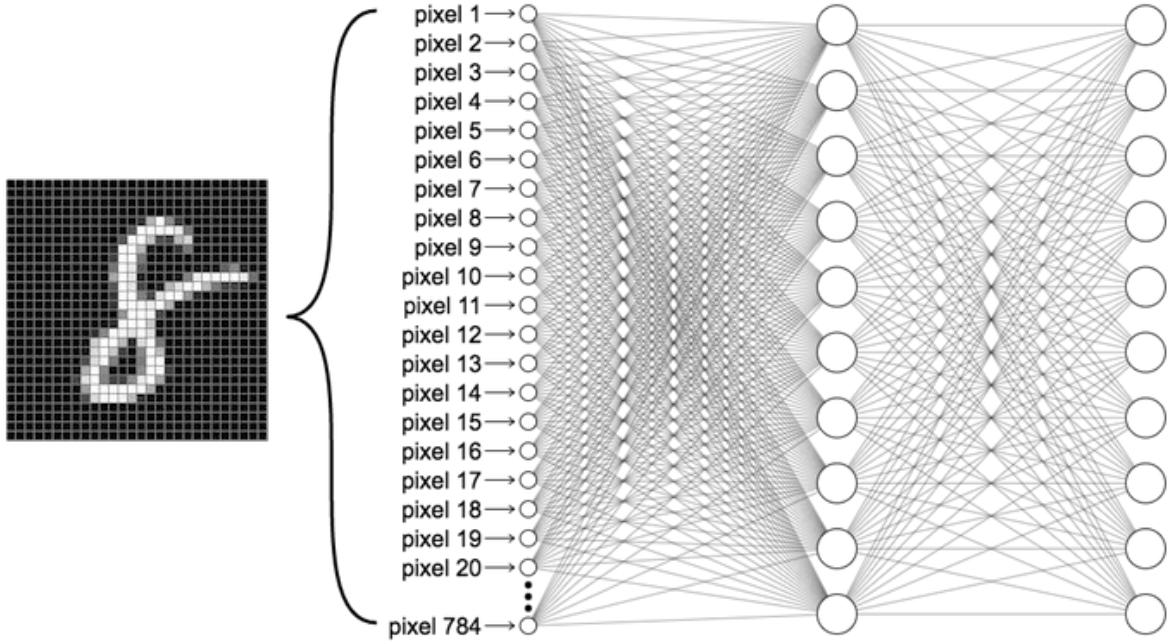
What Does Training Look Like

- The network **WILL** guess wrong during the first iteration and adjust the weights and biases, getting closer to correct answers.
- Depending which input nodes are given a value, different pathways through the network are taken.



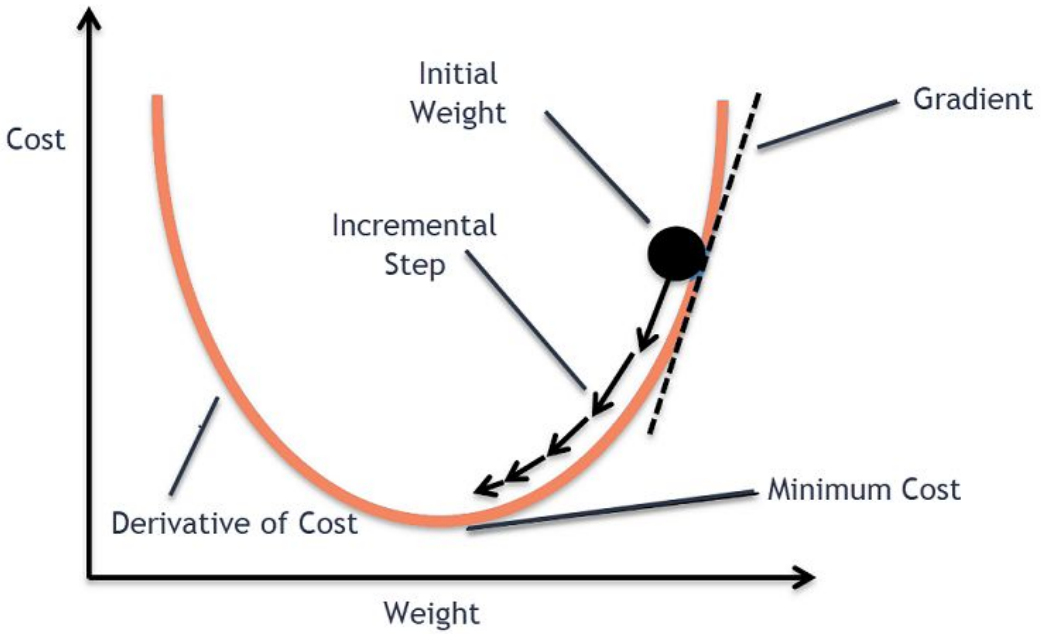
What's Going on Inside?

- Short answer: It's difficult to tell.
- Inside the neural network, is the latent space, an intermediate step, which is meaningless to us.
- The latent space is a compressed and partially processed signal from the data.
- Similar to our nervous system.
  - Observable, but not readily interpretable.



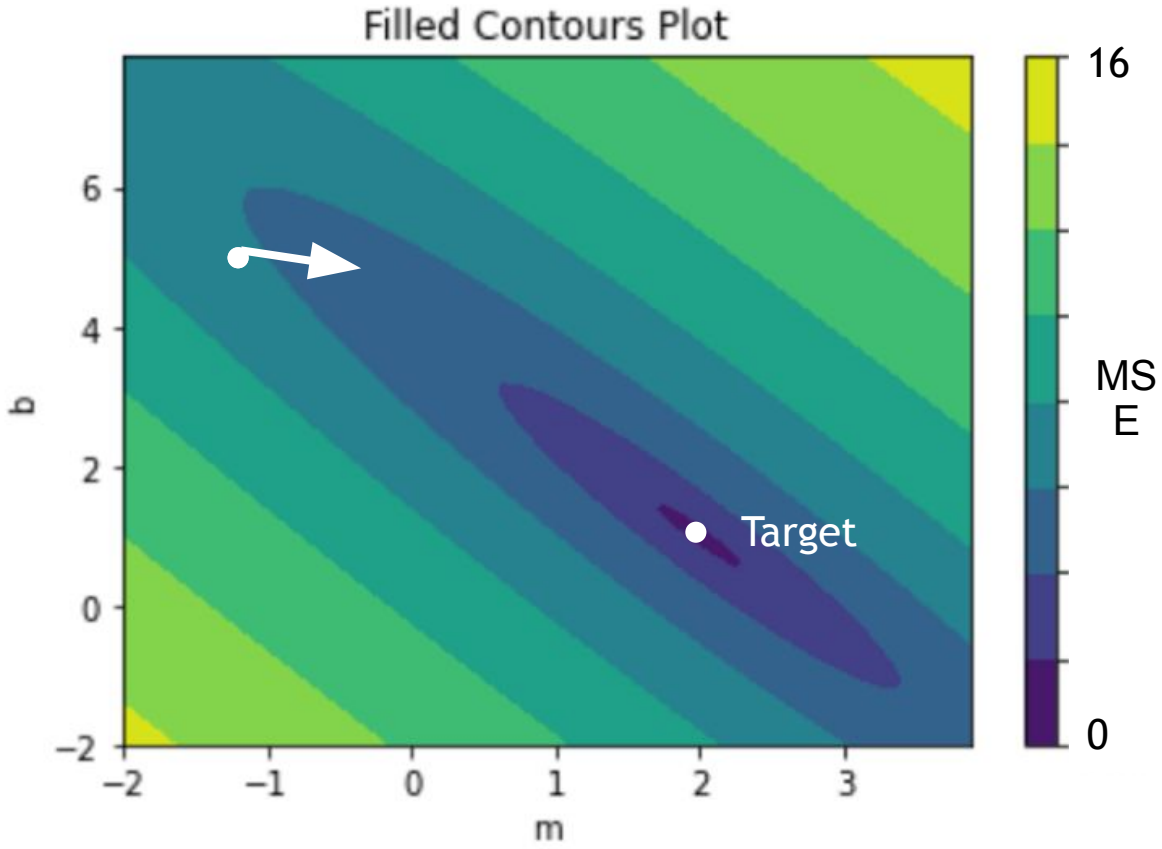
Gradient Descent

- Gradient descent is an optimization algorithm.
- It minimizes a function (typically a loss function) iteratively.
- It calculates the gradient of the function.
- Parameters are adjusted in the direction of steepest descent.
- The process continues until a stopping criterion is met.



THE LOSS CURVE

The Gradient	Which direction loss decreases the most
The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters

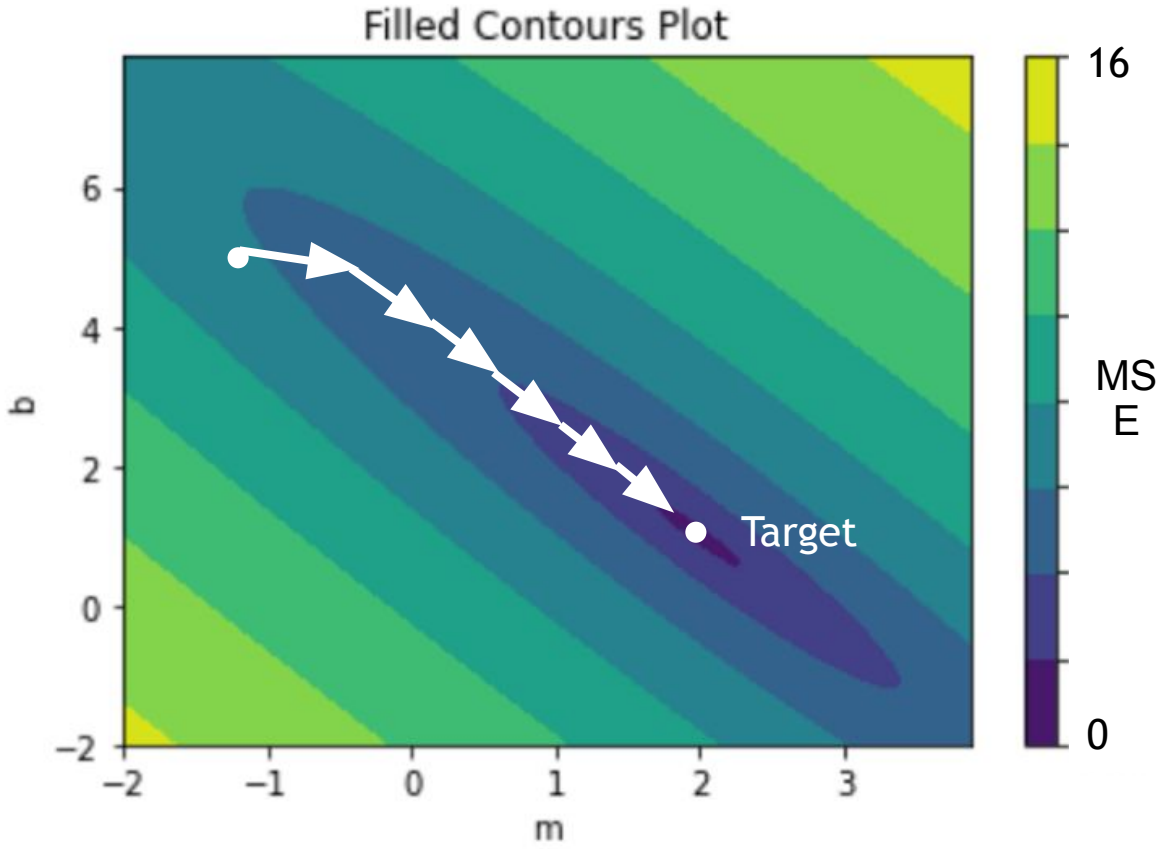


ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

THE LOSS CURVE

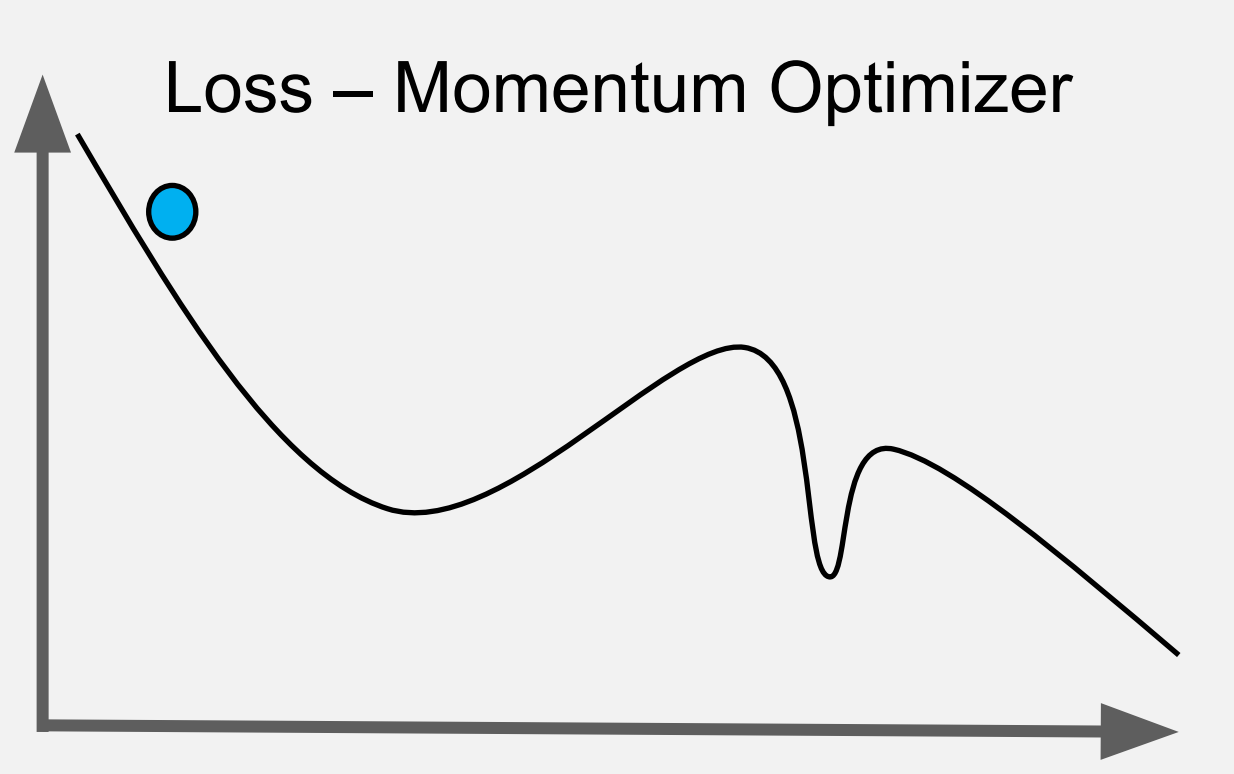
The Gradient	Which direction loss decreases the most
The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters



ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

OPTIMIZERS



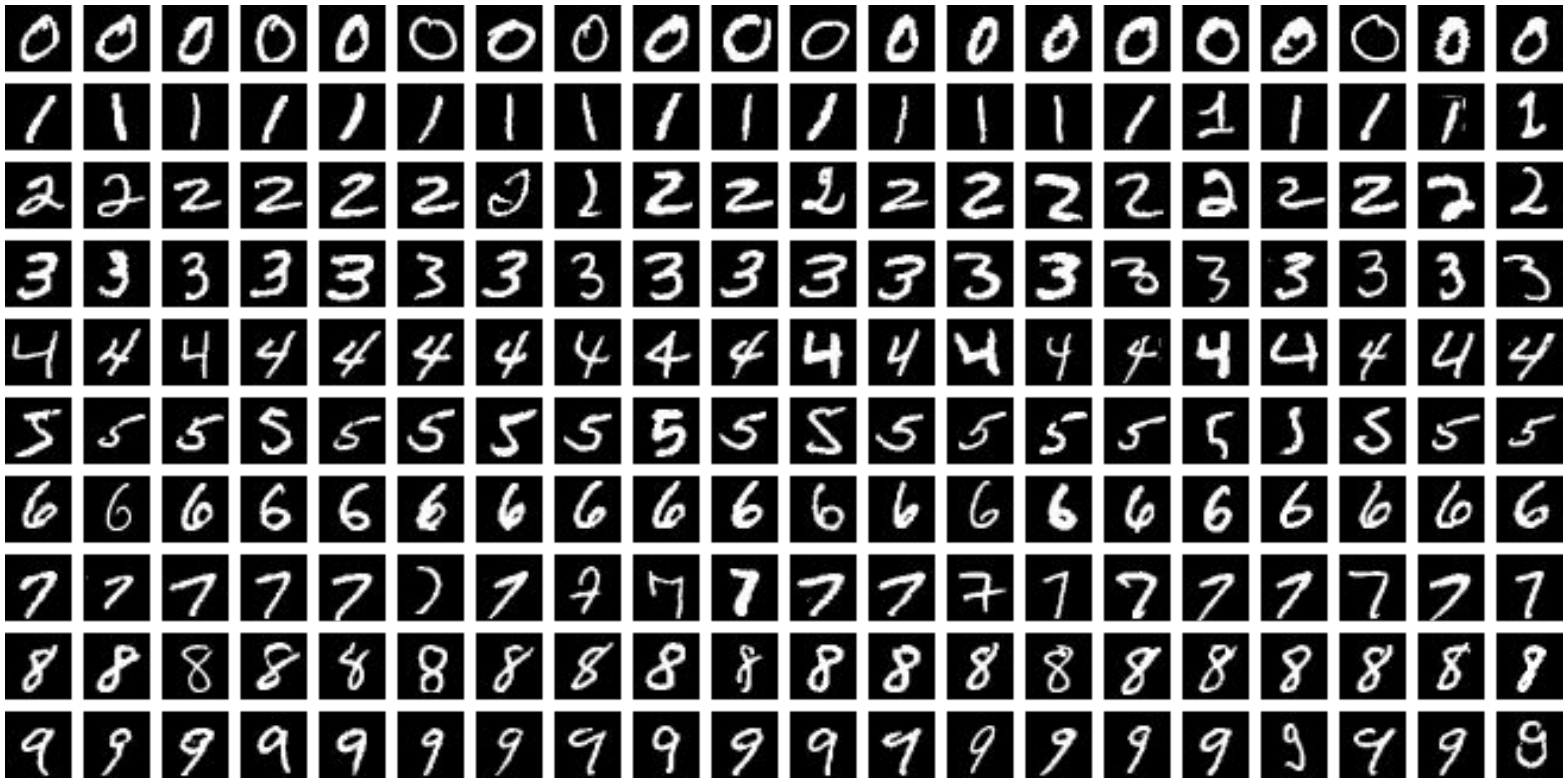
- Adam
- Adagrad
- RMSprop
- SGD

Training/Back Propagation

- Back Propagation is gradient descent* on all of the parameters of the neural network, where the variable being “descended” is a measure of error.
  - Similar to tuning a radio. You can tell you’re close but can’t tell where the exact frequency is.
- We define what measure of error we care about, and this is the loss function.
- Done using partial derivatives to determine how each weight should be adjusted.
- For more details about Backpropagation http://neuralnetworksanddeeplearning.com/

Coding Example – Image Classification with the MNIST Dataset

Notebook



ACENET
accelerate discovery

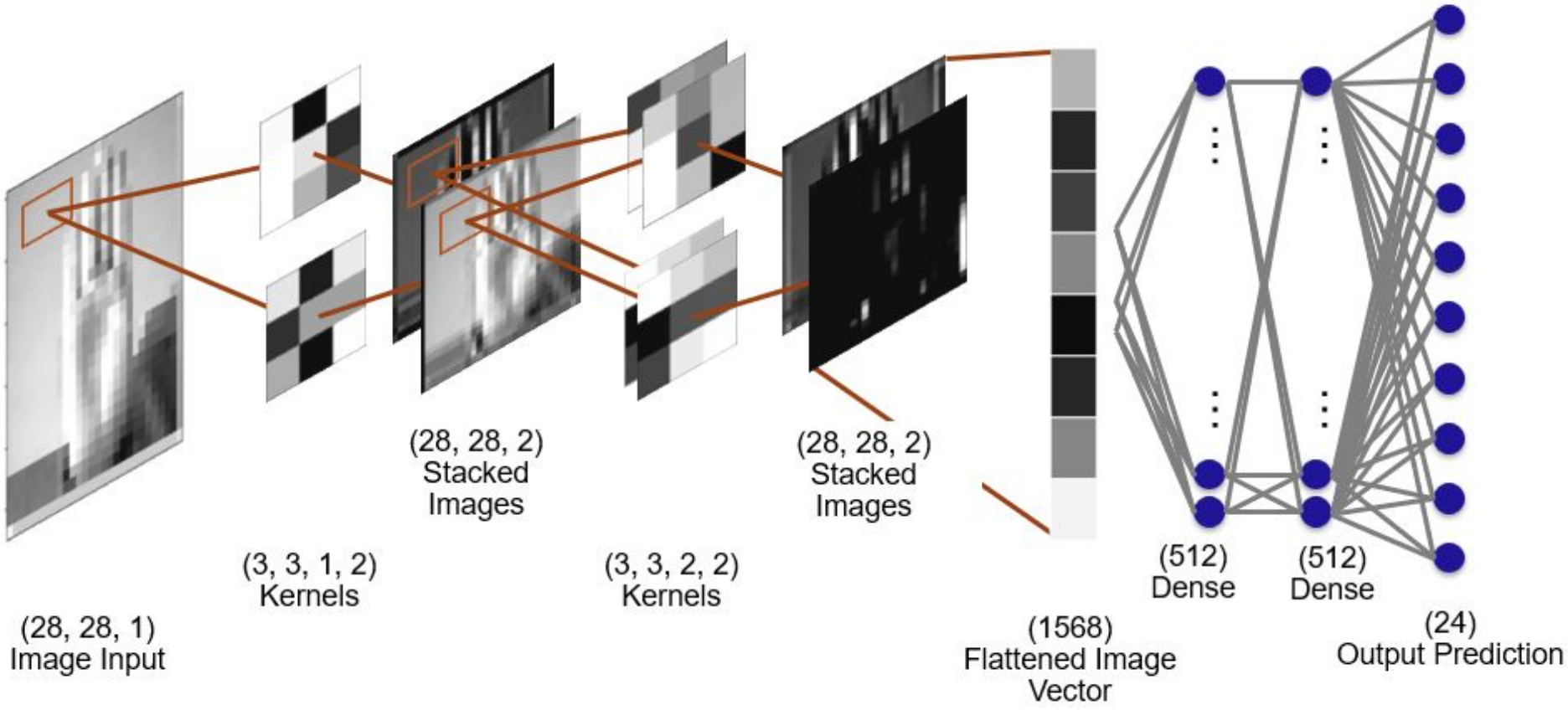
A regional partner of the
Digital Research Alliance of Canada

Specialized Neural Network Architectures

Convolutional Neural Networks (CNNs)

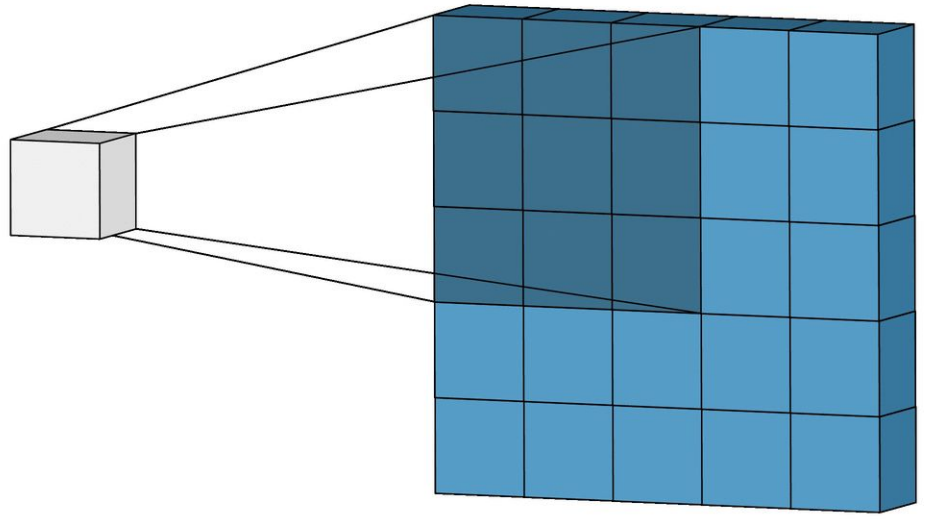
- Class of deep neural networks primarily used for image recognition and computer vision tasks.
- Consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers.
- **Convolutional layers** apply convolution operations to input images, using learnable filters to extract features such as edges, textures, and shapes.
- **Pooling layers** reduce the spatial dimensions of the feature maps generated by convolutional layers, helping to decrease computational complexity and improve translation invariance.
- **Fully connected layers** aggregate the extracted features and perform classification or regression tasks based on the learned representations.

Convolutional Neural networks (CNNs)

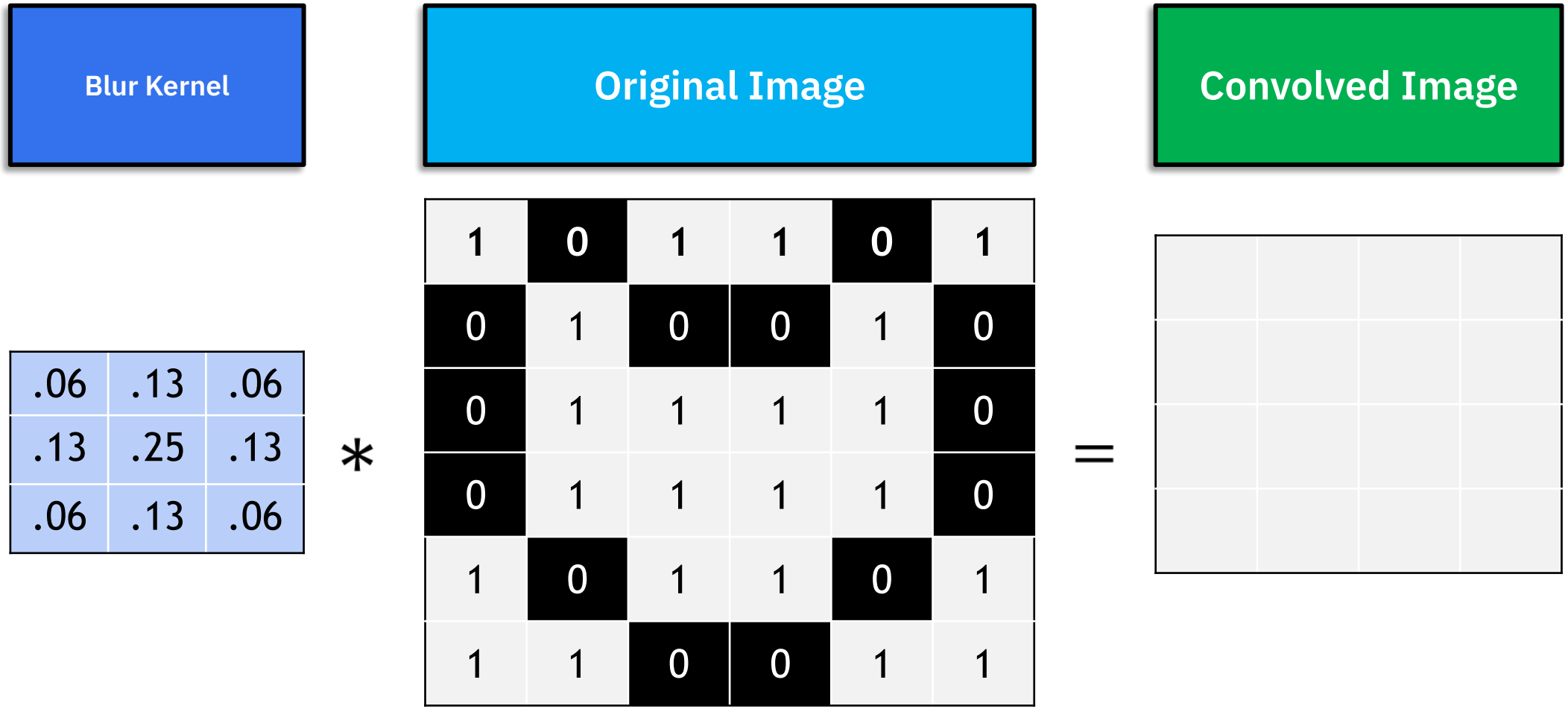


Kernels and Convolution

- The convolutional layer is the core building block of the CNN.
- It carries the main portion of the network's computational load.
- Performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.



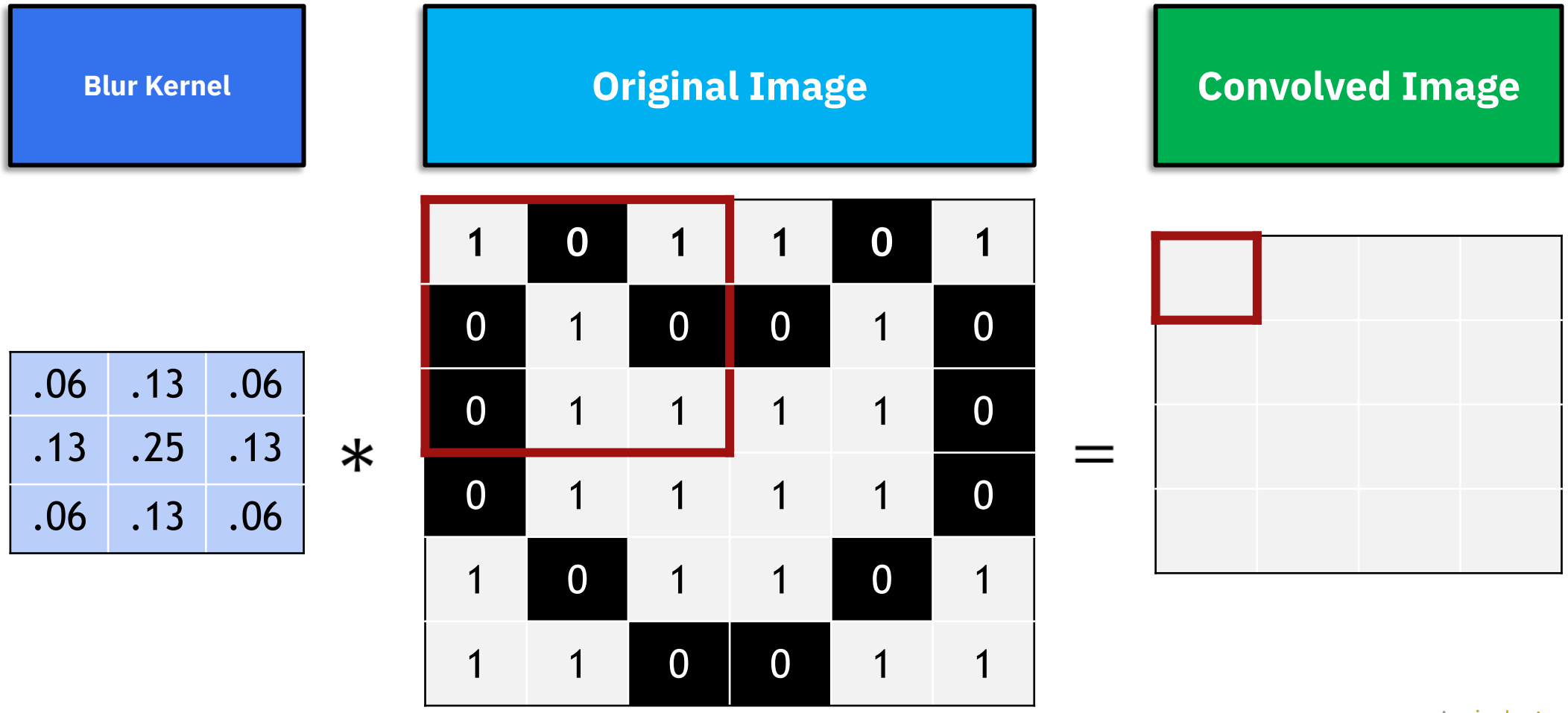
Kernels and Convolution



ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Kernels and Convolution



ACENET
accelerate discovery

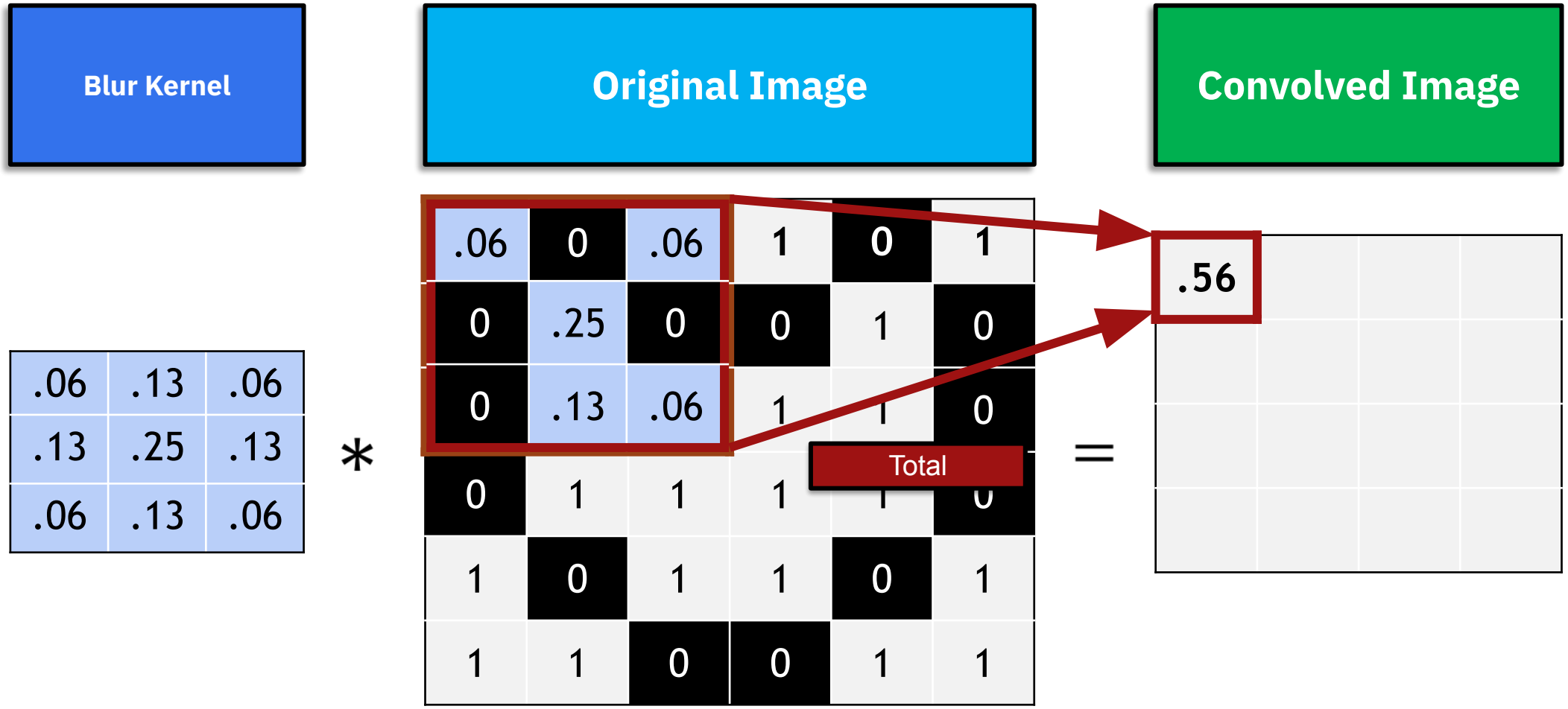
A regional partner of the
Digital Research Alliance of Canada

Kernels and Convolution

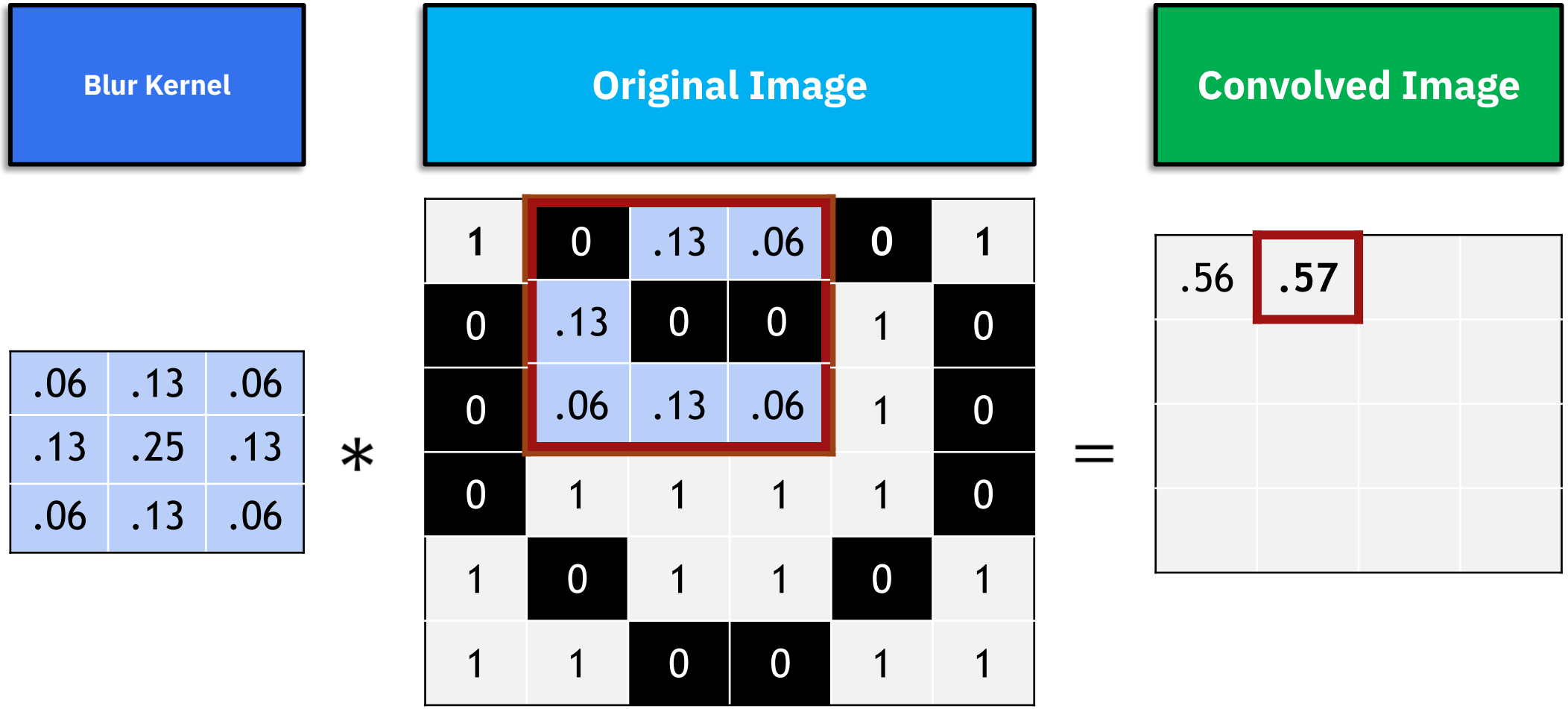
ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Kernels and Convolution



Kernels and Convolution



A regional partner of the Digital Research Alliance of Canada

Kernels and Convolution

Blur Kernel

.06	.13	.06
.13	.25	.13
.06	.13	.06

*

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

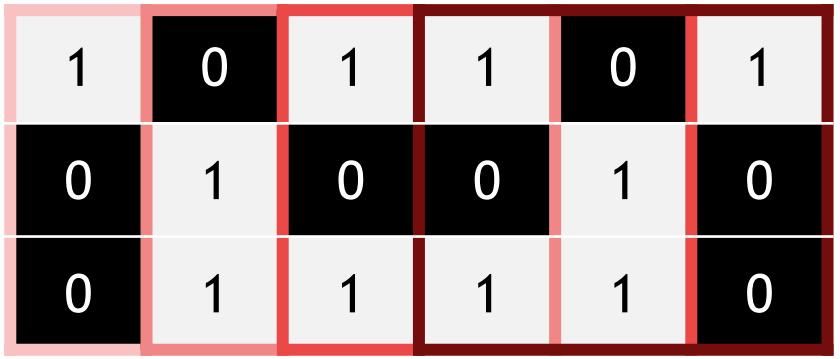
=

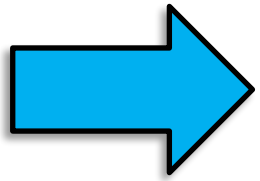
Convolved Image

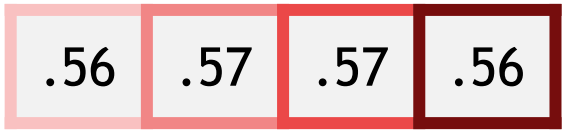
.56	.57	.57	.56
.7	.82	.82	.7
.69	.95	.95	.69
.64	.69	.69	.64

Stride

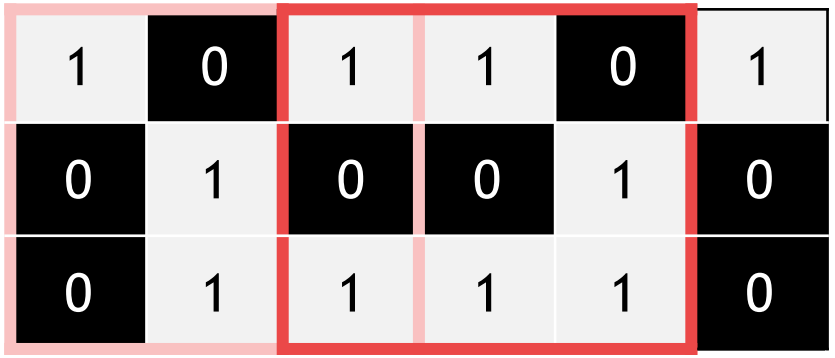
Stride 1

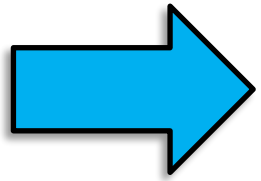




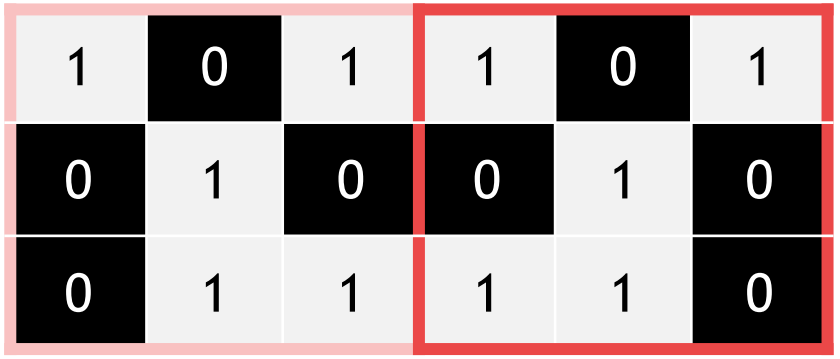


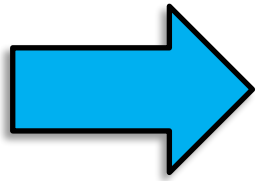
Stride 2





Stride 3





A regional partner of the
Digital Research Alliance of Canada

Padding

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

Zero Padding

0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Padding

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

Mirror Padding

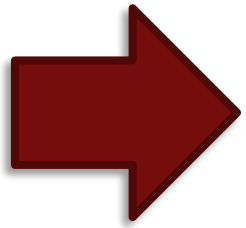
1	1	0	1	1	0	1	1
1	1	0	1	1	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
1	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1
1	1	1	0	0	1	1	1

ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Pooling

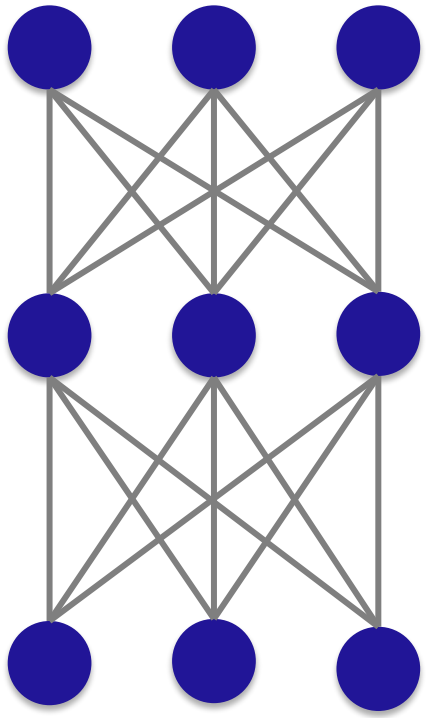
110	256	153	67
12	89	88	43
10	15	50	55
23	9	49	23



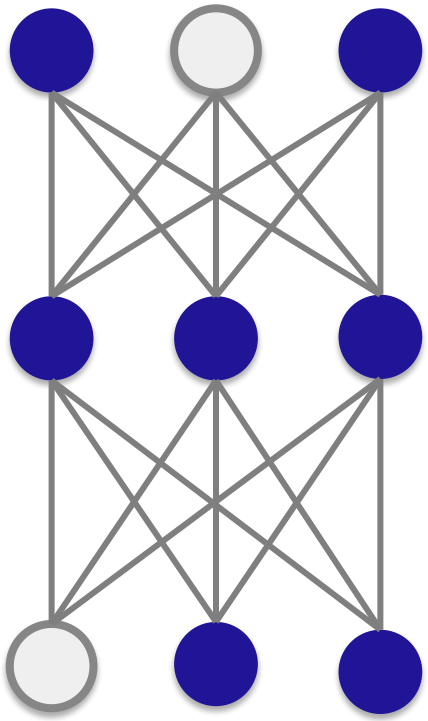
256	153
23	55

A regional partner of the
Digital Research Alliance of Canada

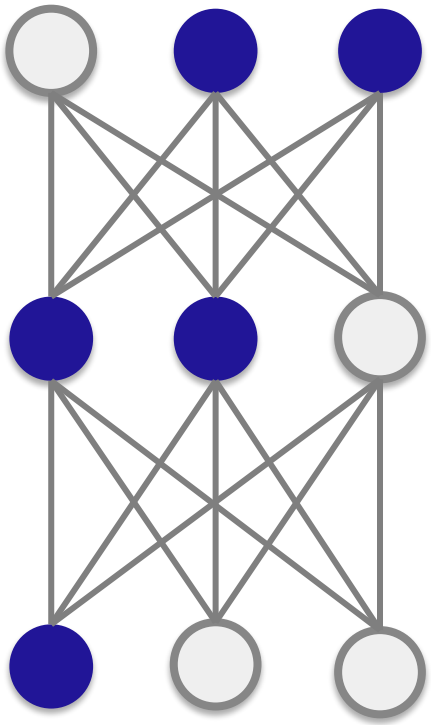
Dropout



rate = 0



rate = .2



rate = .4

Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN)

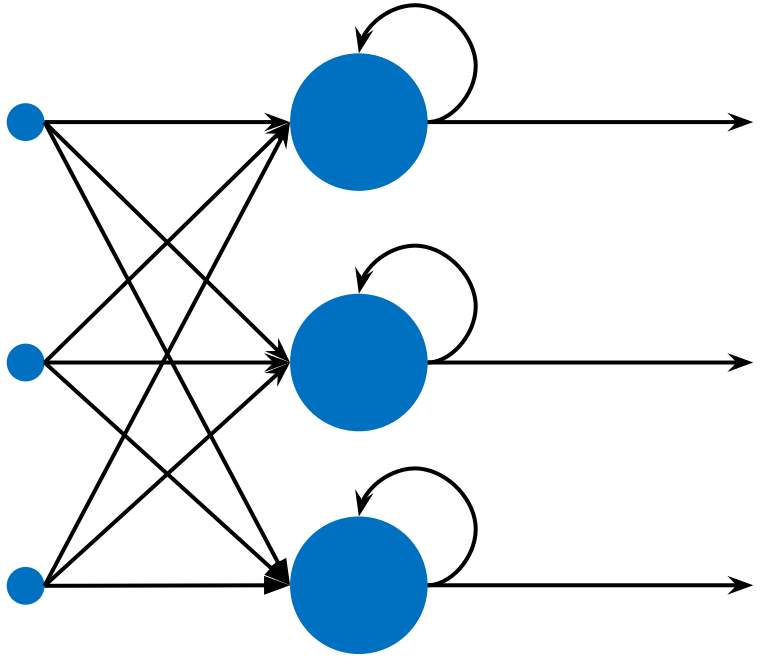
- Class of class of neural network that are helpful in modeling sequence data.
- The first (and only) type of neural network that remembers its input, due to an internal memory.
- Preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more.
- Can remember important things about the input they received, which allows them to be very precise in predicting what's coming next.

When do you need to use an RNN?

"RNNs are helpful when the temporal order in a sequence matters — but today, models like Transformers are often preferred for their ability to capture long-range dependencies more effectively"

How do Recurrent Neural Networks Work?

- The information cycles through a loop.
- RNNs add the immediate past to the present.
- Has two inputs: the present and the recent past.
- Has several variants of architecture including, Bidirectional-RNN, Long short-term Memory (LSTM), and Gated Recurrent Units (GRU).

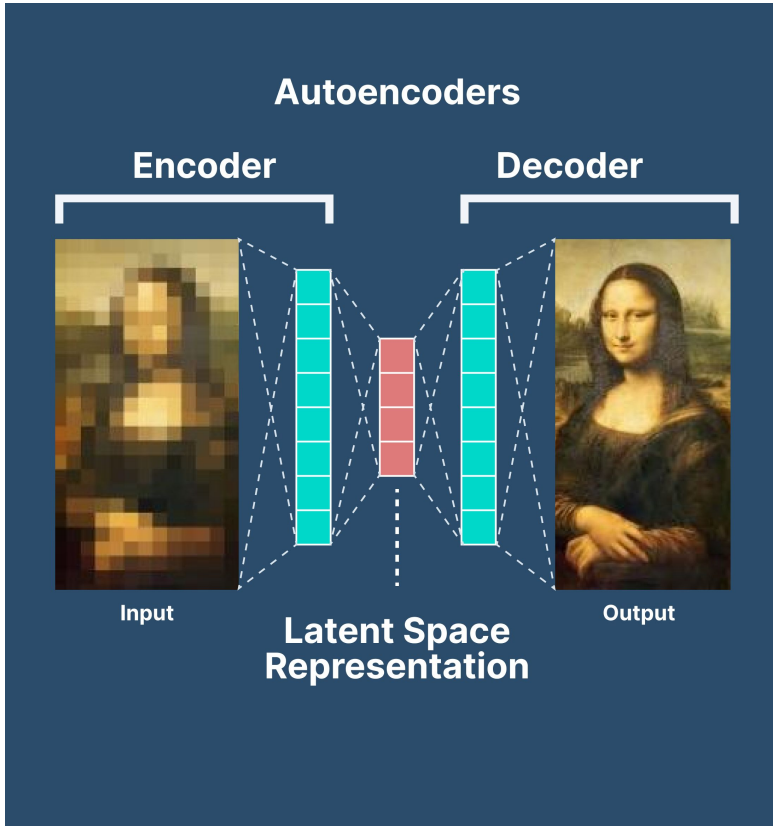


Recurrent Neural Network

Auto-Encoders

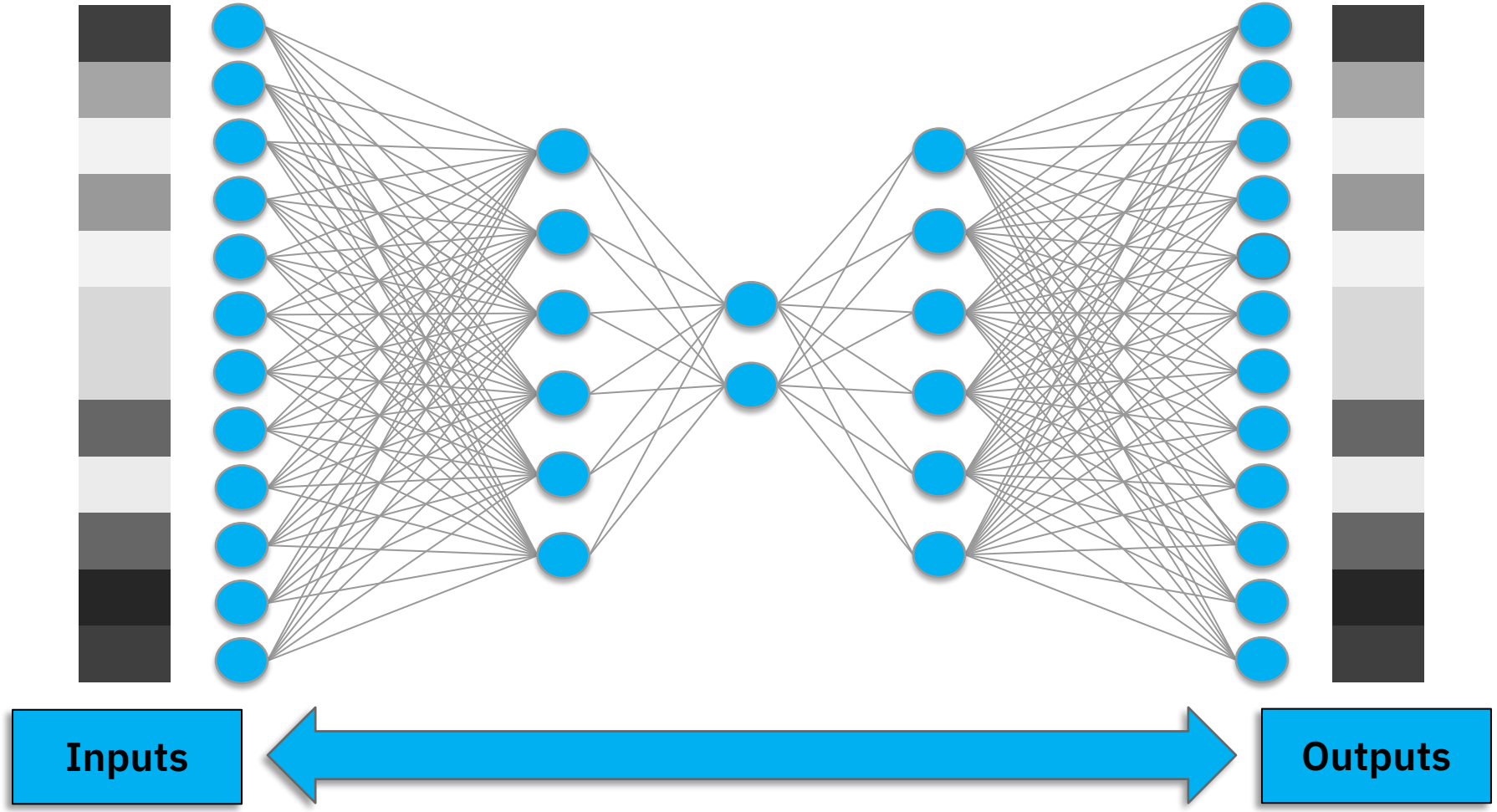
Auto Encoders

- They are primarily used for dimensionality reduction, feature learning, and data compression tasks.
- Able to find a compressed representation of the input data such as image, video, text, speech, etc.
- Can be used to:
  - Transform noisy images into clean data using denoising auto encoders.
  - Add colors to grayscale images (image colorization).
  - Compress images to save memory.
  - Reduce dimensionality of data.
  - Extract the most important features of the input data (features extraction).

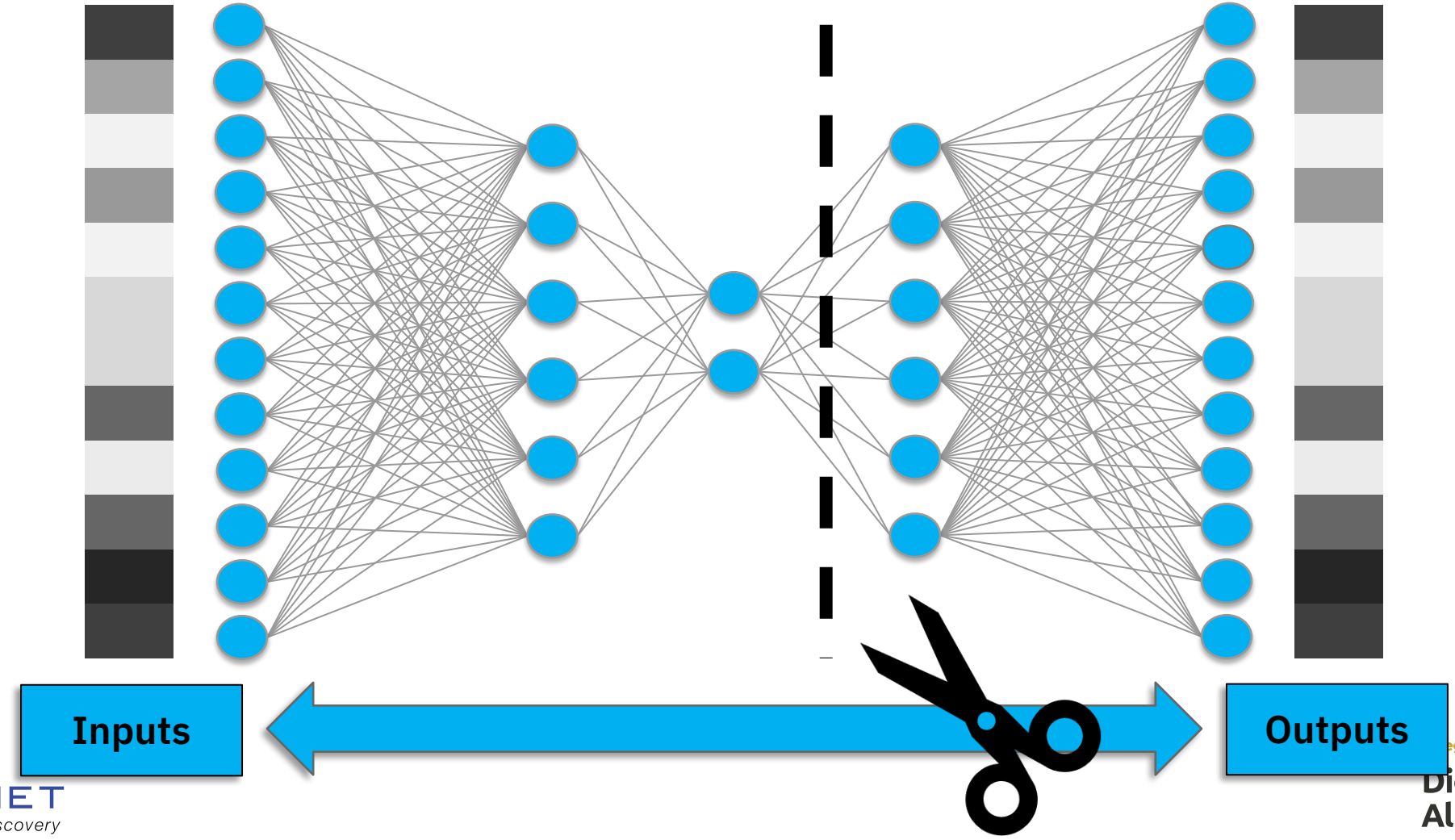


Source

AUTOENCODERS



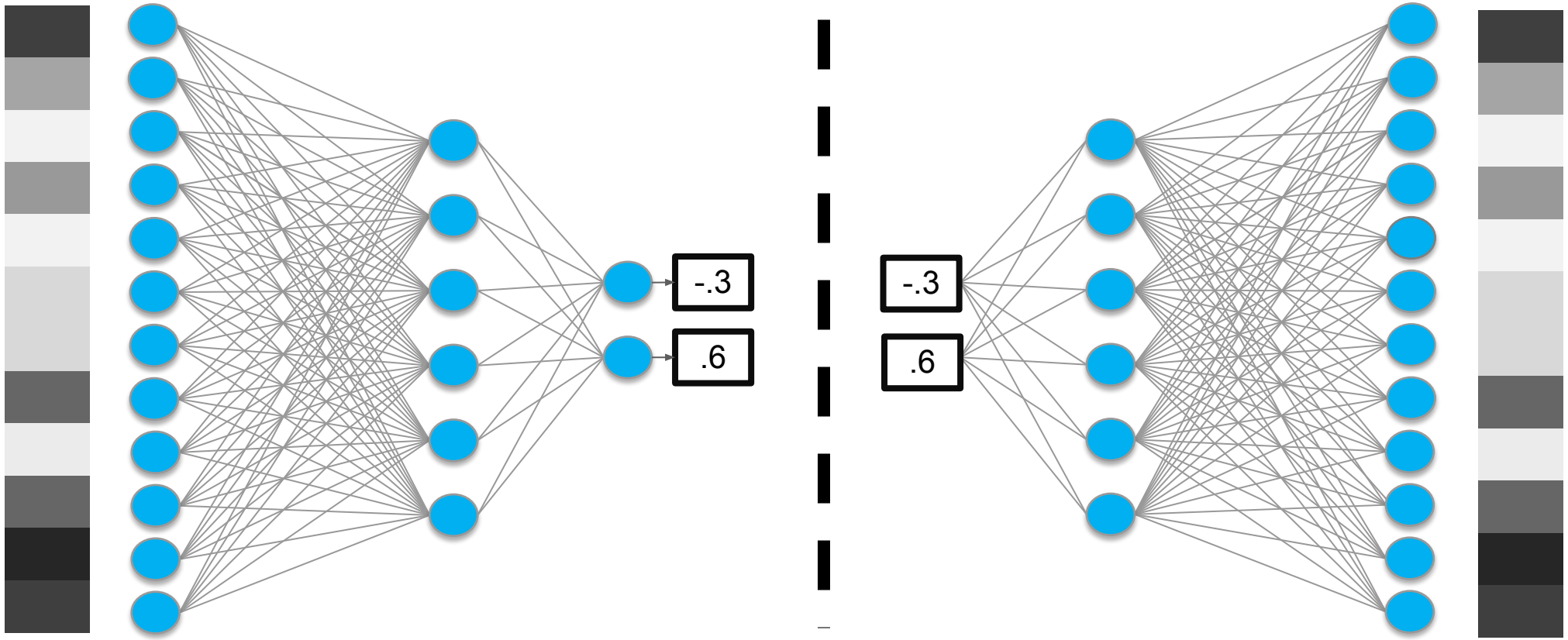
AUTOENCODERS



ACENET
accelerate discovery

regional partner of the
Digital Research Alliance of Canada

AUTOENCODERS



Encoder

Decoder

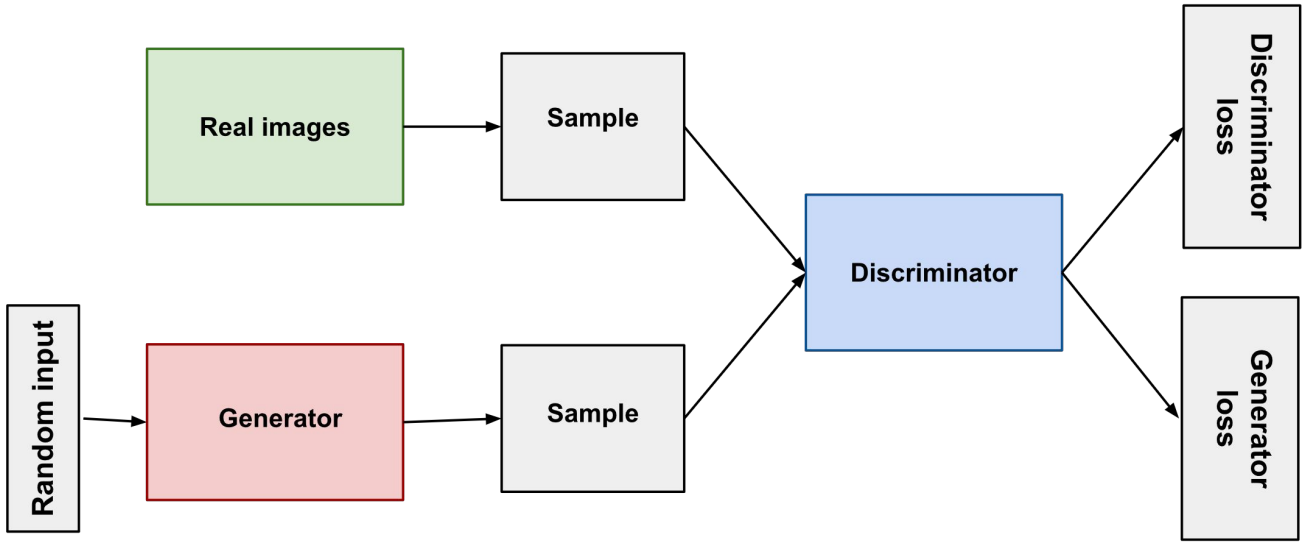
ACENET
accelerate discovery

A regional partner of the
Digital Research Alliance of Canada

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)

- A deep learning architecture that has two parts (the generator and the discriminator).
- The **generator**, learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The **discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.



Source

Next Session: Large Language Models (LLMs)

Questions?