SEMINAR PRESENTATION ON DEEP LEARNING

By:
Ishaya Jeremiah Ayock

ECOLE POLYTECHNIQUE DE THIES,SENEGAL

September 17, 2020

Outline

Differentiate Between a dog and a Cat

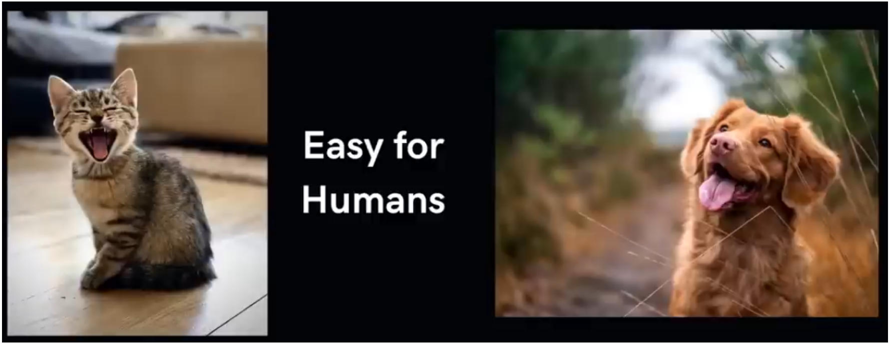


Figure: Cat and Dog

What is Deep Learning?

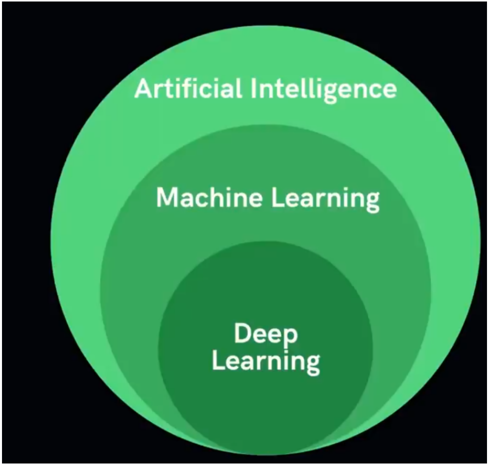


Figure: AI vs Deep learning vs Machine Learning.

Introduction

Machine Learning

Introduction

Machine Learning

Machine Learning Involves teaching computers to recognize patterns in data.

Introduction

Machine Learning

Machine Learning Involves teaching computers to recognize patterns in data.

What is Deep Learning(DL)?

Introduction

Machine Learning

Machine Learning Involves teaching computers to recognize patterns in data.

What is Deep Learning(DL)?

DL is a sub-field of machine learning (ML) in artificial intelligence(AI) that deals with algorithms inspired from the biological structure and functioning of a brain to aid machines with intelligence.

Note

Introduction

Machine Learning

Machine Learning Involves teaching computers to recognize patterns in data.

What is Deep Learning(DL)?

DL is a sub-field of machine learning (ML) in artificial intelligence(AI) that deals with algorithms inspired from the biological structure and functioning of a brain to aid machines with intelligence.

Note

"Deep Learning does't do different things, **it does things differently**"

When do Machine Learning Fails?

While ML works very well for a variety of problems, it fails to excel in some specific cases that seem to be very easy for humans such as;

When do Machine Learning Fails?

While ML works very well for a variety of problems, it fails to excel in some specific cases that seem to be very easy for humans such as;

- Classifying an image as a cat or dog.

When do Machine Learning Fails?

While ML works very well for a variety of problems, it fails to excel in some specific cases that seem to be very easy for humans such as;

- Classifying an image as a cat or dog.
- Distinguishing an audio clip as of a male or female voice.

When do Machine Learning Fails?

While ML works very well for a variety of problems, it fails to excel in some specific cases that seem to be very easy for humans such as;

- Classifying an image as a cat or dog.
- Distinguishing an audio clip as of a male or female voice.
- Performance on image and other unstructured data types.

When do Machine Learning Fails?

While ML works very well for a variety of problems, it fails to excel in some specific cases that seem to be very easy for humans such as;

- Classifying an image as a cat or dog.
- Distinguishing an audio clip as of a male or female voice.
- Performance on image and other unstructured data types.

Africa Digital Native

Sagesse Devoir
Ecole Polytechnique Thiès

Why Deep Learning Now?

- Data Prevalent

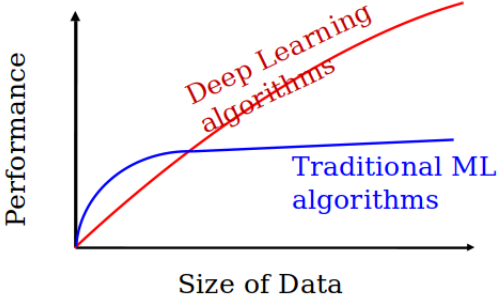
Why Deep Learning Now?

- Data Prevalent
- Improved Hardware Architecture

Why Deep Learning Now?

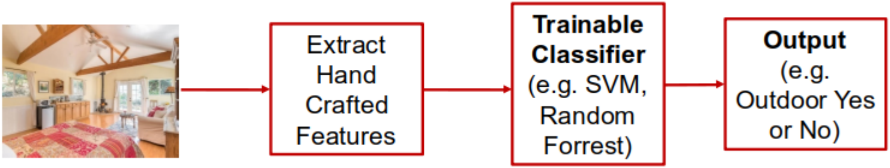
- Data Prevalent
- Improved Hardware Architecture
- New Software Architectures(tensor flow,PyTorch)

Deep Learning vs Traditional Algorithm



Traditional Pattern Recognition

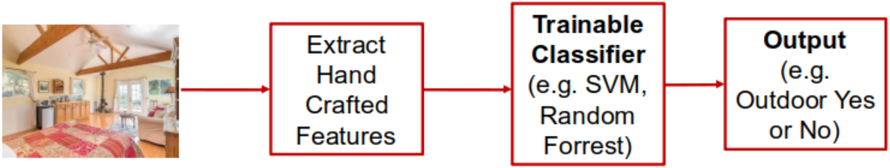
Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.



Limitations

Traditional Pattern Recognition

Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.

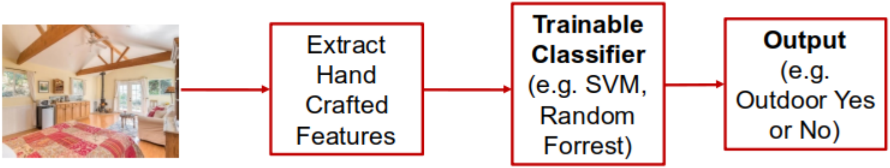


Limitations

- Very tedious and costly to develop hand crafted features.

Traditional Pattern Recognition

Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.

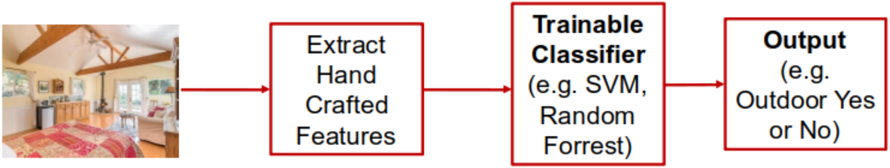


Limitations

- Very tedious and costly to develop hand crafted features.
- The hand-crafted features are usually highly dependents on one application.

Traditional Pattern Recognition

Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.



Limitations

- Very tedious and costly to develop hand crafted features.
- The hand-crafted features are usually highly dependents on one application.

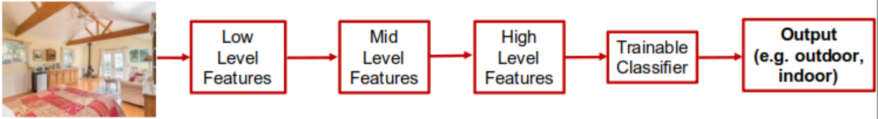
Deep Learning

Deep Learning

Deep learning has an inbuilt automatic multi stage feature learning process that learns rich hierarchical representations(i.e.features).

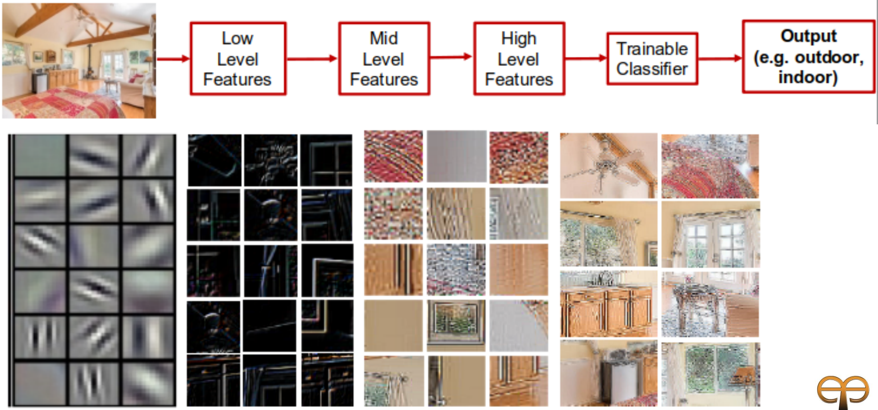
Deep Learning

Deep learning has an inbuilt automatic multi stage feature learning process that learns rich hierarchical representations(i.e.features).



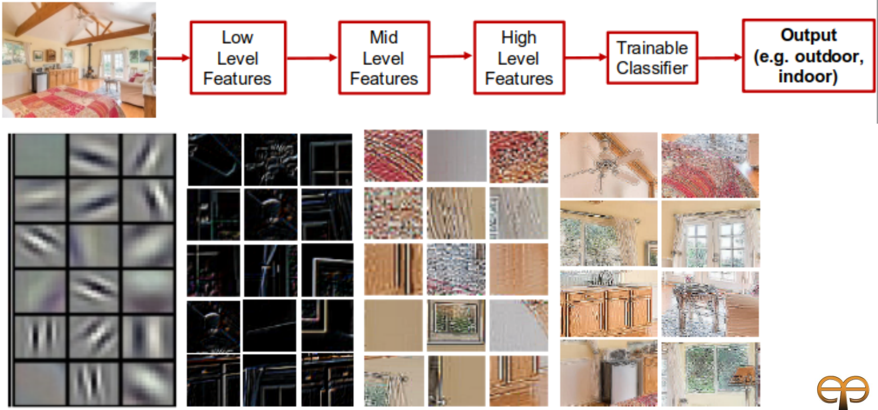
Deep Learning

Deep learning has an inbuilt automatic multi stage feature learning process that learns rich hierarchical representations(i.e.features).



Deep Learning

Deep learning has an inbuilt automatic multi stage feature learning process that learns rich hierarchical representations(i.e.features).



Neural Network

Neural Network

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

What they really do.

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

What they really do.

- Neural networks takes in data as input.

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

What they really do.

- Neural networks takes in data as input.
- Train themselves to understand patterns in the data .

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

What they really do.

- Neural networks takes in data as input.
- Train themselves to understand patterns in the data .
- Output useful predictions.

Neural Network

Neural Network

Its an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
The fundamental building block of the Neural Network is the **Neuron.**

What they really do.

- Neural networks takes in data as input.
- Train themselves to understand patterns in the data .
- Output useful predictions.

Neural Networks(ANN)

Neural Networks(ANN)

The building blocks of neural networks including **neurons**, **weights** and **activation functions**.

Neural Networks(ANN)

The building blocks of neural networks including **neurons**, **weights** and **activation functions**.
The building blocks are used in layers to create networks and its trained from the example data.

Neural Networks(ANN)

The building blocks of neural networks including **neurons**, **weights** and **activation functions**.
The building blocks are used in layers to create networks and its trained from the example data.
Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

Neural Networks(ANN)

The building blocks of neural networks including **neurons**, **weights** and **activation functions**.
The building blocks are used in layers to create networks and its trained from the example data.
Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

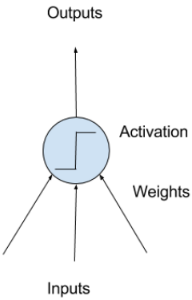
Neurons

Neurons

The building block for neural networks are artificial neurons. These are simple computational units that have weighted input signals and produce an output signal using an activation function.

Neurons

The building block for neural networks are artificial neurons. These are simple computational units that have weighted input signals and produce an output signal using an activation function.



Neurons

The building block for neural networks are artificial neurons. These are simple computational units that have weighted input signals and produce an output signal using an activation function.

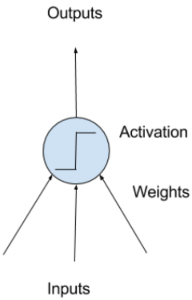


Figure: Model of a simple Neuron.

Networks of Neurons

Networks of Neurons

Neurons are arranged into networks of neurons. A row of neurons is called a layer and one network can have multiple layers. The architecture of the neurons in the network is often called the network topology.

Networks of Neurons

Neurons are arranged into networks of neurons. A row of neurons is called a layer and one network can have multiple layers. The architecture of the neurons in the network is often called the network topology.

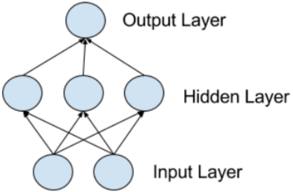
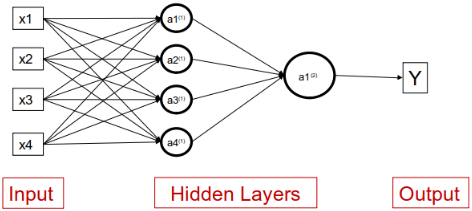
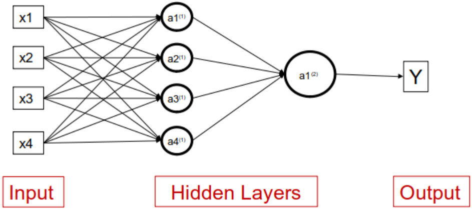
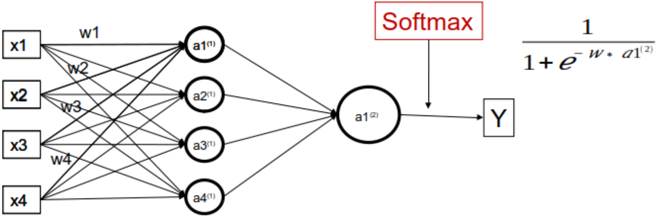


Figure: Model of a simple Network.


x1
x2
x3
x4
a1(1)
a2(1)
a3(1)
a4(1)
a1(2)
Y
Input
Hidden Layers
Output

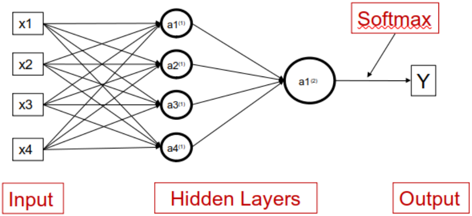

x1
x2
x3
x4
a1(1)
a2(1)
a3(1)
a4(1)
a1(2)
Y
Input
Hidden Layers
Output



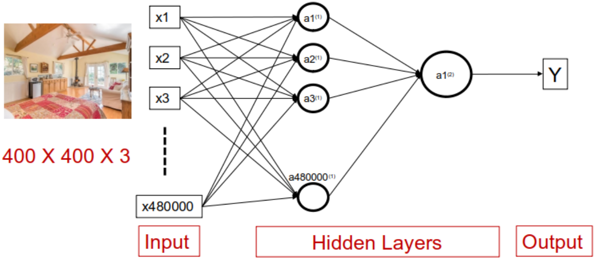
$$a_1^{(1)} = f(W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + W_4 * X_4)$$

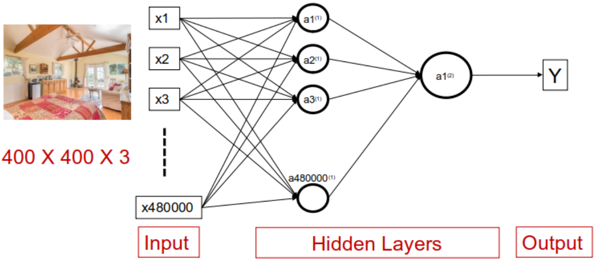
$$f()is\ activation\ function:\ Relu\ or\ Sigmoid\ Reu:\ max(0,x)$$

$$a_1^{(1)} = max(0, W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + W_4 * X_4)$$


x1
x2
x3
x4
a1(1)
a2(1)
a3(1)
a4(1)
a1(2)
Softmax
Y
Input
Hidden Layers
Output

$$4*4 + 4 + 1$$


400 X 400 X 3
x1
x2
x3
x480000
a1(1)
a2(1)
a3(1)
a480000(1)
a1(2)
Y
Input
Hidden Layers
Output



No of Param $480000 * 480000 + 480000 + 1 =$ approximately 230 Billion !!!

Advantages

Advantages

- It does feature extraction, no engineering of features.

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features
- Better optimization

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features
- Better optimization
- A new level of Noise robustness

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features
- Better optimization
- A new level of Noise robustness
- Multi-task and transferring learning

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features
- Better optimization
- A new level of Noise robustness
- Multi-task and transferring learning
- Better Architectures.

Advantages

- It does feature extraction, no engineering of features.
- Moving toward raw features
- Better optimization
- A new level of Noise robustness
- Multi-task and transferring learning
- Better Architectures.

Disadvantages

Disadvantages

- Need a large amount of data set.

Disadvantages

- Need a large amount of data set.
- Because of the large data set, training time is usually significant.

Disadvantages

- Need a large amount of data set.
- Because of the large data set, training time is usually significant.
- Parameters are hard to interpret–although there is progress being made.

Disadvantages

- Need a large amount of data set.
- Because of the large data set, training time is usually significant.
- Parameters are hard to interpret–although there is progress being made.
- Hyper parameter Tuning is non-trivial.

Learning process of a Neural Network

Neural Network

Learning process of a Neural Network

Neural Network

The learning process of a neural network can be broken down into two;

Learning process of a Neural Network

Neural Network

The learning process of a neural network can be broken down into two;

- Forward Propagation.

Learning process of a Neural Network

Neural Network

The learning process of a neural network can be broken down into two;

- Forward Propagation.
- Back Propagation.

Learning process of a Neural Network

Neural Network

The learning process of a neural network can be broken down into two;

- Forward Propagation.
- Back Propagation.

Forward Propagation

Forward Propagation

Forward Propagation is the propagation of information from the input layer down to the output layer of the network. The input layers are several neurons x_1 down to x_n

Forward Propagation

Forward Propagation is the propagation of information from the input layer down to the output layer of the network. The input layers are several neurons x_1 down to x_n

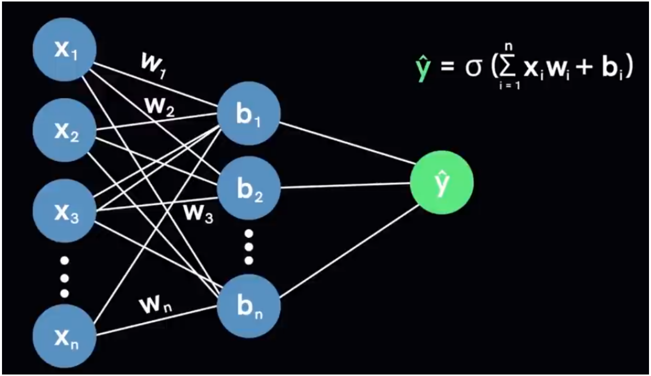


Figure: Forward propagation

Forward Propagation

Forward Propagation is the propagation of information from the input layer down to the output layer of the network. The input layers are several neurons x_1 down to x_n

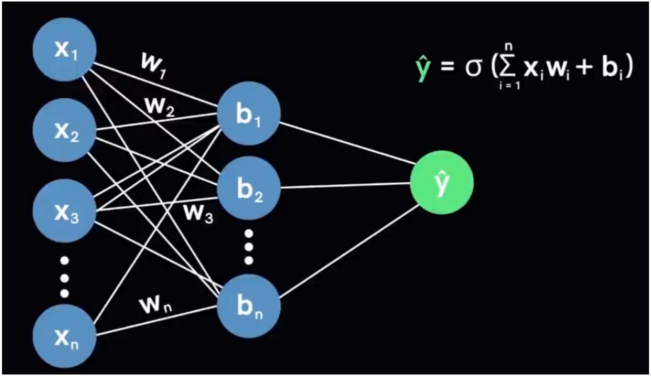


Figure: Forward propagation

Terms

Terms

- **Weight**:
  The weight of a neuron determines how important that neuron is in that in the layer.

Terms

- **Weight**:
  The weight of a neuron determines how important that neuron is in that in the layer.
- **Bias**:
  This allows the for the shifting of the σ(activation function) to the left or right.

Terms

- **Weight**:
  The weight of a neuron determines how important that neuron is in that in the layer.
- **Bias**:
  This allows the for the shifting of the σ(activation function) to the left or right.

Back Propagation

Back Propagation

Back Propagation is like the forward propagation,except in the opposite direction. Information is pass from the output layer down to the **hidden layers** not the **input layer**.

Back Propagation

Back Propagation is like the forward propagation,except in the opposite direction. Information is pass from the output layer down to the hidden layers not the input layer.

- Its the reason why neural networks are powerful.

Back Propagation

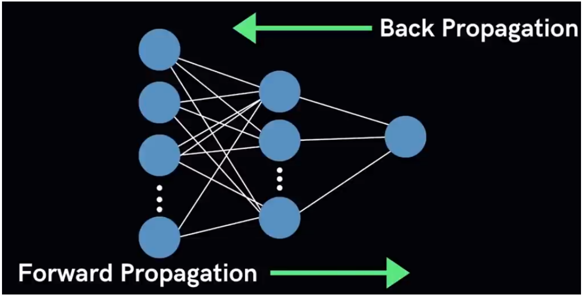
Back Propagation is like the forward propagation,except in the opposite direction. Information is pass from the output layer down to the hidden layers not the input layer.

- Its the reason why neural networks are powerful.
- Its the reason why neural networks can learn by there-selves.

Back Propagation

Back Propagation is like the forward propagation,except in the opposite direction. Information is pass from the output layer down to the **hidden layers** not the **input layer**.

- Its the reason why neural networks are powerful.
- Its the reason why neural networks can learn by there-selves.



Back Propagation

Back Propagation is like the forward propagation,except in the opposite direction. Information is pass from the output layer down to the hidden layers not the input layer.

- Its the reason why neural networks are powerful.
- Its the reason why neural networks can learn by there-selves.

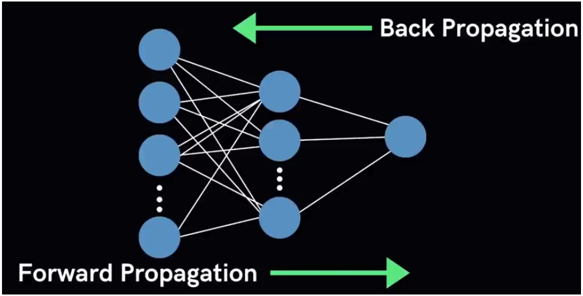


Figure: Back propagation

Note

Forward propagation

In the last step of forward propagation, a neural network split out the prediction into Right or Wrong.

Note

Forward propagation

In the last step of forward propagation, a neural network split out the prediction into Right or Wrong.

Backward propagation

Note

Forward propagation

In the last step of forward propagation, a neural network split out the prediction into Right or Wrong.

Backward propagation

In the backward propagation, the neural network evaluate it performance and check if its right or wrong. If wrong, the network uses what is called the loss function to quantify the deviation from the expected output. And this information is now sent back to the hidden layers for the weight and biases to be adjusted so that the network accuracy increases.

Note

Forward propagation

In the last step of forward propagation, a neural network split out the prediction into Right or Wrong.

Backward propagation

In the backward propagation, the neural network evaluate it performance and check if its right or wrong. If wrong, the network uses what is called the loss function to quantify the deviation from the expected output. And this information is now sent back to the hidden layers for the weight and biases to be adjusted so that the network accuracy increases.

Example on how it works.

Example on how it works.

#	Weight	Goods	Car/ Truck
1	15	02	Car
2	34	67	Truck
3	42	54	Truck

Figure: Car and Truck

Example on how it works.

#	Weight	Goods	Car/ Truck
1	15	02	Car
2	34	67	Truck
3	42	54	Truck

Figure: Car and Truck

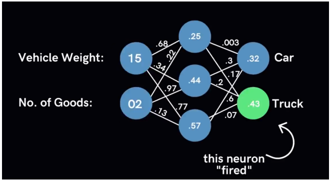


Figure: implementation

Example on how it works.CONT...

- Use a Loss function to quantify the deviation from the original output.

Example on how it works.CONT...

- Use a Loss function to quantify the deviation from the original output.
- Go backwards and adjust initial weights and biases.

Example on how it works.CONT...

- Use a Loss function to quantify the deviation from the original output.
- Go backwards and adjust initial weights and biases.
- Values adjusted to better fit prediction model.

Example on how it works.CONT...

- Use a Loss function to quantify the deviation from the original output.
- Go backwards and adjust initial weights and biases.
- Values adjusted to better fit prediction model.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.
- Compare the predicted value with the expect values and calculate the loss.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.
- Compare the predicted value with the expect values and calculate the loss.
- Perform back propagation to propagate this loss back through the network.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.
- Compare the predicted value with the expect values and calculate the loss.
- Perform back propagation to propagate this loss back through the network.
- Update the parameters based on the loss.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.
- Compare the predicted value with the expect values and calculate the loss.
- Perform back propagation to propagate this loss back through the network.
- Update the parameters based on the loss.
- Iterate previous steps till loss is minimized.

Summary of the process.

The above process can be summarized as follows;

Learning Algorithm

- Initialize the parameters with random values(weights and biases).
- Feed input data to network.
- Compare the predicted value with the expect values and calculate the loss.
- Perform back propagation to propagate this loss back through the network.
- Update the parameters based on the loss.
- Iterate previous steps till loss is minimized.

The most common terminologies used in Neural Network is as follows;

The most common terminologies used in Neural Network is as follows;

Terms Used

The most common terminologies used in Neural Network is as follows;

Terms Used

- **1. Activation Functions**.

The most common terminologies used in Neural Network is as follows;

Terms Used

- **1. Activation Functions**.
  - They introduce non-linearity into the network.

The most common terminologies used in Neural Network is as follows;

Terms Used

- **1. Activation Functions**.
  - They introduce non-linearity into the network.
  - They decide whether a neuron can contribute to the next layer.

The most common terminologies used in Neural Network is as follows;

Terms Used

- **1. Activation Functions**.
  - They introduce non-linearity into the network.
  - They decide whether a neuron can contribute to the next layer.

Choice Activation Function

- Binary classification problems; **Sigmoid**

Choice Activation Function

- Binary classification problems; **Sigmoid**
- If unsure : **ReLU(or Modified ReLU)**

Choice Activation Function

- Binary classification problems; **Sigmoid**
- If unsure : **ReLU(or Modified ReLU)**

Why Non-Linear Activation Functions?

Choice Activation Function

- Binary classification problems; **Sigmoid**
- If unsure : **ReLU(or Modified ReLU)**

Why Non-Linear Activation Functions?

Because they introduce non-linearity into the network. And simply means your activation function must be none linear.

Choice Activation Function

- Binary classification problems; **Sigmoid**
- If unsure : **ReLU(or Modified ReLU)**

Why Non-Linear Activation Functions?

Because they introduce non-linearity into the network. And simply means your activation function must be none linear.

Choice Activation Function

- Binary classification problems; **Sigmoid**
- If unsure : **ReLU(or Modified ReLU)**

Why Non-Linear Activation Functions?

Because they introduce non-linearity into the network. And simply means your activation function must be none linear.

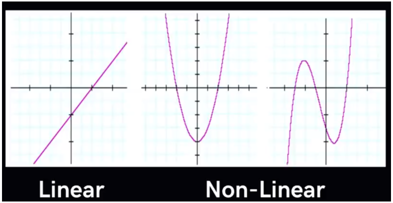


Figure: Linear and Non-linear function.

2. Loss Function

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output.

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output. Its a metric that helps a network understand whether it is learning in the right direction There many different kinds of loss functions e.g;

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output. Its a metric that helps a network understand whether it is learning in the right direction There many different kinds of loss functions e.g;

- **Regression:** Squared Error, Absolute error loss, Huber loss.

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output. Its a metric that helps a network understand whether it is learning in the right direction There many different kinds of loss functions e.g;

- **Regression:** Squared Error, Absolute error loss, Huber loss.
- **Binary Classification problems**: Binary Cross-Entropy, Hinge loss.

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output. Its a metric that helps a network understand whether it is learning in the right direction There many different kinds of loss functions e.g;

- **Regression:** Squared Error, Absolute error loss, Huber loss.
- **Binary Classification problems**: Binary Cross-Entropy, Hinge loss.
- **Multi-Class Classification:** Multi-Class Cross-Entropy, Kullback Divergence.

2. Loss Function

A loss function is a way of quantifying the deviation of the predicted output by the neural network to the expected output. Its a metric that helps a network understand whether it is learning in the right direction There many different kinds of loss functions e.g;

- **Regression:** Squared Error, Absolute error loss, Huber loss.
- **Binary Classification problems**: Binary Cross-Entropy, Hinge loss.
- **Multi-Class Classification:** Multi-Class Cross-Entropy, Kullback Divergence.

3. Optimizers

Gradient Descent.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.
- Fast,Robust, Flexible.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.
- Fast,Robust, Flexible.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.
- Fast,Robust, Flexible.

The Gradient of a function is the vector of partial derivative with respect to all independent variables.

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.
- Fast,Robust, Flexible.

The Gradient of a function is the vector of partial derivative with respect to all independent variables.

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)\right)$$

3. Optimizers

Gradient Descent.

Its an iterative algorithm that starts off at a random point on the loss function and travels down its slope in steps until it reaches the lowest point(minimum) of the function.

- Most popular optimizer.
- Fast,Robust, Flexible.

The Gradient of a function is the vector of partial derivative with respect to all independent variables.

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)\right)$$

Stochastic Gradient Descent.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.
- Its uses *Momentum* to accumulate gradients.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.
- Its uses *Momentum* to accumulate gradients.
- Less intensive computational.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.
- Its uses *Momentum* to accumulate gradients.
- Less intensive computational.

Backpropagation

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.
- Its uses *Momentum* to accumulate gradients.
- Less intensive computational.

Backpropagation

Its simply gradient decent implemented on a network.

Stochastic Gradient Descent.

- Its like the gradient decent, except it uses a *subset* of the training examples rather than the entire data set.
- SGD is an implementation of gradient decent that uses *batches* on each pass.
- Its uses *Momentum* to accumulate gradients.
- Less intensive computational.

Backpropagation

Its simply gradient decent implemented on a network.

Adagrad

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.
- Learning rate tends to get small with time.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.
- Learning rate tends to get small with time.

RMSprop

- A specialized version of Adagrad.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.
- Learning rate tends to get small with time.

RMSprop

- A specialized version of Adagrad.
- Accumulates Gradients in a fixed window.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.
- Learning rate tends to get small with time.

RMSprop

- A specialized version of Adagrad.
- Accumulates Gradients in a fixed window.
- Similar to Adaprop.

Adagrad

Adam: Stands for *Adaptive Moment Estimation*. Its uses the concept of momentum.

- Adapts Learning rate to individual features
- Some weights will have different learning rates.
- Ideal for sparse data sets with many input examples missing.
- Learning rate tends to get small with time.

RMSprop

- A specialized version of Adagrad.
- Accumulates Gradients in a fixed window.
- Similar to Adaprop.

Parameters and Hyper-parameters

Model Parameters

This are variables that are internal to the neural network.

Parameters and Hyper-parameters

Model Parameters

This are variables that are internal to the neural network.

- The values can be estimated right from the data.

Parameters and Hyper-parameters

Model Parameters

This are variables that are internal to the neural network.

- The values can be estimated right from the data.
- Not set manually.

Parameters and Hyper-parameters

Model Parameters

This are variables that are internal to the neural network.

- The values can be estimated right from the data.
- Not set manually.
- They are required by the model to make predictions and are saved as part of the learned model(e.g Weights,Biases)

Parameters and Hyper-parameters

Model Parameters

This are variables that are internal to the neural network.

- The values can be estimated right from the data.
- Not set manually.
- They are required by the model to make predictions and are saved as part of the learned model(e.g Weights,Biases)

Hyper-Parameters

Hyper-Parameters

This are configurations that are external to the neural network. Their values cannot be estimated right from the data.

Hyper-Parameters

This are configurations that are external to the neural network. Their values cannot be estimated right from the data.

- If the parameter is manually specify, its a hyper-parameter.

Hyper-Parameters

This are configurations that are external to the neural network. Their values cannot be estimated right from the data.

- If the parameter is manually specify, its a hyper-parameter.
- There is no best way of finding the best value.

Hyper-Parameters

This are configurations that are external to the neural network. Their values cannot be estimated right from the data.

- If the parameter is manually specify, its a hyper-parameter.
- There is no best way of finding the best value.
- When a DL algorithm is tuned, you are really tuning the hyper-parameter(e.g Learning rate)

Hyper-Parameters

This are configurations that are external to the neural network. Their values cannot be estimated right from the data.

- If the parameter is manually specify, its a hyper-parameter.
- There is no best way of finding the best value.
- When a DL algorithm is tuned, you are really tuning the hyper-parameter(e.g Learning rate)

These are needed when the dataset size is **large**

These are needed when the dataset size is **large**

- **Epochs**:
  This is when an entire dataset is passed forward and backward through the neural network only once.

These are needed when the dataset size is **large**

- **Epochs**:
  This is when an entire dataset is passed forward and backward through the neural network only once.
- **Batch**
  Dividing a large dataset into small batches and feed those batches into the neural network.

These are needed when the dataset size is **large**

- **Epochs**:
  This is when an entire dataset is passed forward and backward through the neural network only once.
- **Batch**
  Dividing a large dataset into small batches and feed those batches into the neural network.
- **Batch Size**
  The total number of training examples in a batch.

These are needed when the dataset size is **large**

- **Epochs**:
  This is when an entire dataset is passed forward and backward through the neural network only once.
- **Batch**
  Dividing a large dataset into small batches and feed those batches into the neural network.
- **Batch Size**
  The total number of training examples in a batch.
- **Iterations**
  Number of batches needed to complete one epoch.

These are needed when the dataset size is **large**

- **Epochs**:
  This is when an entire dataset is passed forward and backward through the neural network only once.
- **Batch**
  Dividing a large dataset into small batches and feed those batches into the neural network.
- **Batch Size**
  The total number of training examples in a batch.
- **Iterations**
  Number of batches needed to complete one epoch.

Regularization

Overfitting

Regularization

Overfitting

A model should perform well on training data and new test data. But this is a problem in deep learning.

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Handling Overfitting in Deep Learning

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Handling Overfitting in Deep Learning

- Dropout

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Handling Overfitting in Deep Learning

- Dropout
- Augmentation(**More Data - Better Model**)

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Handling Overfitting in Deep Learning

- Dropout
- Augmentation(**More Data - Better Model**)
- Early Stopping

Regularization

Overfitting

A model should perform well on training data and new test data.
But this is a problem in deep learning.
The most common problem faced in deep learning is over fitting.

Handling Overfitting in Deep Learning

- Dropout
- Augmentation(**More Data - Better Model**)
- Early Stopping

Types of Neural Network Architectures

Types of Neural Network Architectures

- Fully Connected Feed forward Neural Network

Types of Neural Network Architectures

- Fully Connected Feed forward Neural Network
- Convolutional Neural Network(CNN)

Types of Neural Network Architectures

- Fully Connected Feed forward Neural Network
- Convolutional Neural Network(CNN)
- Recurrent Neural Network(RNN)

Types of Neural Network Architectures

- Fully Connected Feed forward Neural Network
- Convolutional Neural Network(CNN)
- Recurrent Neural Network(RNN)

Fully Connected Feed forward Neural Network

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Each neuron has a non-linear activation function that allows deep learning to model complex problems.e.g;

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Each neuron has a non-linear activation function that allows deep learning to model complex problems.e.g;

- Sigmoid

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Each neuron has a non-linear activation function that allows deep learning to model complex problems.e.g;

- Sigmoid
- Tanh

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Each neuron has a non-linear activation function that allows deep learning to model complex problems.e.g;

- Sigmoid
- Tanh
- ReLU

Fully Connected Feed forward Neural Network

Each Neuron is connected to every subsequent layer with no backward connections.

Each neuron has a non-linear activation function that allows deep learning to model complex problems.e.g;

- Sigmoid
- Tanh
- ReLU

Networks

Networks

We create Networks with various

Networks

We create Networks with various

- inputs.

Networks

We create Networks with various

- inputs.
- outputs.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.
- Neurons per hidden layer.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.
- Neurons per hidden layer.
- Activation functions.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.
- Neurons per hidden layer.
- Activation functions.

And this allows us to create a powerful deep neural network.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.
- Neurons per hidden layer.
- Activation functions.

And this allows us to create a powerful deep neural network. Also the more neurons, the wider the network and more hidden layers, the deep the network becomes.

Networks

We create Networks with various

- inputs.
- outputs.
- Hidden Layers.
- Neurons per hidden layer.
- Activation functions.

And this allows us to create a powerful deep neural network. Also the more neurons, the wider the network and more hidden layers, the deep the network becomes.

Model Building

Definition

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

1. **Define your model**. Create a Sequential model and add configured layers.

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

1. **Define your model**. Create a Sequential model and add configured layers.
2. **Compile your model**. Specify loss function and optimizers and call the *compile()* function on the model.

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

1. **Define your model**. Create a Sequential model and add configured layers.
2. **Compile your model**. Specify loss function and optimizers and call the *compile()* function on the model.
3. **Fit your model**. Train the model on a sample of data by calling the *fit()* function on the model.

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

1. **Define your model**. Create a Sequential model and add configured layers.
2. **Compile your model**. Specify loss function and optimizers and call the *compile()* function on the model.
3. **Fit your model**. Train the model on a sample of data by calling the *fit()* function on the model.
4. **Make predictions**. Use the model to generate predictions on new data by calling functions such as *evaluate()* or *predict()* on the model.(**KERAS**)

Model Building

Definition

The main type of model is a sequence of layers called a Sequential which is a linear stack of layers. You create a Sequential and add layers to it in the order that you wish for the computation to be performed.

1. **Define your model**. Create a Sequential model and add configured layers.
2. **Compile your model**. Specify loss function and optimizers and call the *compile()* function on the model.
3. **Fit your model**. Train the model on a sample of data by calling the *fit()* function on the model.
4. **Make predictions**. Use the model to generate predictions on new data by calling functions such as *evaluate()* or *predict()* on the model.(**KERAS**)

Note that

Note that

- Neural networks are not models of the brain but are instead computational models for solving complex machine learning problems.

Note that

- Neural networks are not models of the brain but are instead computational models for solving complex machine learning problems.
- Neural networks are made up of neurons that have weights and activation functions.

Note that

- Neural networks are not models of the brain but are instead computational models for solving complex machine learning problems.
- Neural networks are made up of neurons that have weights and activation functions.
- Networks are organized into layers of neurons and are trained using stochastic gradient descent.

Note that

- Neural networks are not models of the brain but are instead computational models for solving complex machine learning problems.
- Neural networks are made up of neurons that have weights and activation functions.
- Networks are organized into layers of neurons and are trained using stochastic gradient descent.
- It is a good idea to prepare your data before training a neural network model.

Note that

- Neural networks are not models of the brain but are instead computational models for solving complex machine learning problems.
- Neural networks are made up of neurons that have weights and activation functions.
- Networks are organized into layers of neurons and are trained using stochastic gradient descent.
- It is a good idea to prepare your data before training a neural network model.

CNN

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

There are three types of layers in a Convolutional Neural Network:

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

There are three types of layers in a Convolutional Neural Network:

- Convolutional Layers.

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

There are three types of layers in a Convolutional Neural Network:

- Convolutional Layers.
- Pooling Layers.

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

There are three types of layers in a Convolutional Neural Network:

- Convolutional Layers.
- Pooling Layers.
- Fully-Connected Layers.

CNN

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

Building Blocks of Convolutional Neural Networks

There are three types of layers in a Convolutional Neural Network:

- Convolutional Layers.
- Pooling Layers.
- Fully-Connected Layers.

Layers

Input/Visible Layers

Layers

Input/Visible Layers

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset.

Layers

Input/Visible Layers

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset.

Hidden Layers

Layers

Input/Visible Layers

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset.

Hidden Layers

Layers after the input layer are called hidden layers because they are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value. Deep learning can refer to having many hidden layers in your neural network.

Layers

Input/Visible Layers

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset.

Hidden Layers

Layers after the input layer are called hidden layers because they are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value. Deep learning can refer to having many hidden layers in your neural network.

Output Layer

Output Layer

Output Layer

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling.

Output Layer

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling.

Example

Output Layer

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling.

Example

- A regression problem may have a single output neuron and the neuron may have no activation function.

Output Layer

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling.

Example

- A regression problem may have a single output neuron and the neuron may have no activation function.
- A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the primary class.

Output Layer

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling.

Example

- A regression problem may have a single output neuron and the neuron may have no activation function.
- A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the primary class.

Convolutional Layers

Convolutional Layers

Convolutional layers are comprised of Filters and Feature Maps.

Convolutional Layers

Convolutional layers are comprised of Filters and Feature Maps.

Convolutional Layers

Convolutional layers are comprised of Filters and Feature Maps.

The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron.

Convolutional Layers

Convolutional layers are comprised of Filters and Feature Maps.

The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

Convolutional Layers

Convolutional layers are comprised of Filters and Feature Maps.

The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

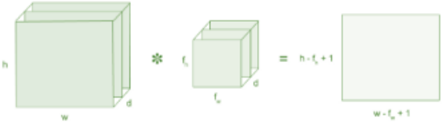


Figure: Image matrix multiplies

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

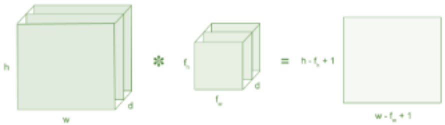


Figure: Image matrix multiplies

Figure: Filter Matrix

Strides

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Pooling Layers

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large.

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Types of Pooling

- Max Pooling.

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Types of Pooling

- Max Pooling.
- Min pooling.

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Types of Pooling

- Max Pooling.
- Min pooling.
- Average Pooling.

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Types of Pooling

- Max Pooling.
- Min pooling.
- Average Pooling.
- Sum Pooling.

Pooling Layers

In this layer, the number of parameters are reduce when the images are too large. Spatial pooling also called sub-sampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

Types of Pooling

- Max Pooling.
- Min pooling.
- Average Pooling.
- Sum Pooling.

Max Pooling

Max Pooling

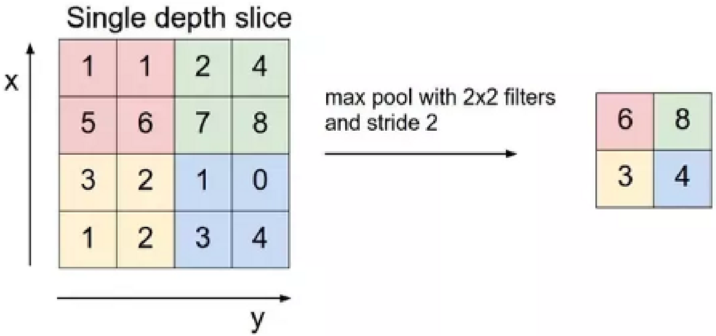


Figure: Max Pooling

Fully Connected Layers

Fully Connected Layers

Fully connected layers are the normal flat feed forward neural network layer.

Fully Connected Layers

Fully connected layers are the normal flat feed forward neural network layer. The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

Fully Connected Layers

Fully connected layers are the normal flat feed forward neural network layer. The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

Fully Connected Layers

Fully connected layers are the normal flat feed forward neural network layer. The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

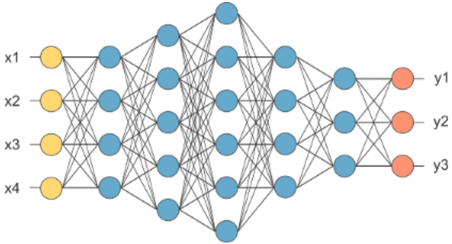


Figure: Fully connected layer

From the above figure, the feature map matrix will be converted as vector $(x1, x2, x3, \ldots)$. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or Sigmoid to classify the outputs as cat, dog, car, truck e.t.c.

CNN Architecture

From the above figure, the feature map matrix will be converted as vector $(x1, x2, x3, \ldots)$. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or Sigmoid to classify the outputs as cat, dog, car, truck e.t.c.

CNN Architecture

From the above figure, the feature map matrix will be converted as vector $(x1, x2, x3, \dots)$. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or Sigmoid to classify the outputs as cat, dog, car, truck e.t.c.

CNN Architecture

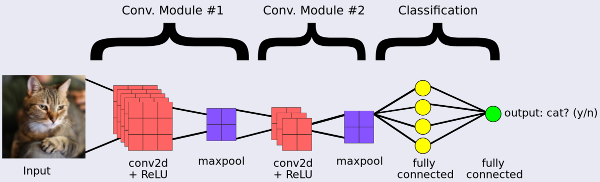


Figure: CNN Architecture

Summary

Summary

- Provide input image into convolution layer.

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.
- Add as many Convolutional layers until satisfied

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.
- Add as many Convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer).

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.
- Add as many Convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer).
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.
- Add as many Convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer).
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

Applications of CNN

CNN

Applications of CNN

CNN

- Computer Vision

Applications of CNN

CNN

- Computer Vision
- Image Recognition

Applications of CNN

CNN

- Computer Vision
- Image Recognition
- Image Processing

Applications of CNN

CNN

- Computer Vision
- Image Recognition
- Image Processing
- Image Segmentation

Applications of CNN

CNN

- Computer Vision
- Image Recognition
- Image Processing
- Image Segmentation
- Video Analysis

Applications of CNN

CNN

- Computer Vision
- Image Recognition
- Image Processing
- Image Segmentation
- Video Analysis
- Natural Language Processing

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)
- Preprocessing the data.

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)
- Preprocessing the data.
- Training the Model.

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)
- Preprocessing the data.
- Training the Model.
- Evaluation.

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)
- Preprocessing the data.
- Training the Model.
- Evaluation.
- Optimization

Creating Deep Learning Models

Models Creations

There are fundamentally five steps in creating deep learning models.

- Gathering Data(UCL, Kaggle,google dataset search and reddit.)
- Preprocessing the data.
- Training the Model.
- Evaluation.
- Optimization

Keras

About Keras

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

- Modularity:

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

- Modularity:
- Minimalism:

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

- Modularity:
- Minimalism:
- Extensibility:

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

- Modularity:
- Minimalism:
- Extensibility:
- Python:

Keras

About Keras

Keras is a Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development.

- Modularity:
- Minimalism:
- Extensibility:
- Python:

Recommendations

- **Deep Learning** Ian Goodfellow, Yoshua Bengio, and Aaron Courville
- Grokking Deep Learning by Andrew W. Trask
- Deep Learning with Python by Francois Chollet

1. https://www.edureka.co/blog/introduction-to-machine-learning/
2. https://magoosh.com/data-science/convolutional-neural-networks-explained/
3. https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks.
4. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Conclusion


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
SUKSAMA
GRAZIE
MEHRBANI
BİYAN
SHUKRIA
BOLZİN
MERCI
JUSPAXAR
EFCHARISTO