

Neural Network Essentials

DR. TUHIN CHATTOPADHYAY
FOUNDER & CEO, TUHIN AI ADVISORY

A massive interconnection of computing cells, also known as neurons, comprises a neural network, which is the heart and soul of all artificial intelligence (AI) algorithms. Almost 80 years ago, in 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts from the University of Illinois at Chicago first proposed the neural network. After a series of evolutions, the deep neural networks (DNNs) — which are usually those with more than two hidden layers — are now used for image recognition, image classification, object detection, speech recognition, language translation, natural language processing (NLP), and natural language generation (NLG).

The present Refcard first introduces the concept of a neural network by drawing an analogy with the biological neural network and subsequently walks readers through the essential components of neural networks, which are considered the building blocks. Common neural architectures are discussed thereafter, defining the underlying arrangement of the neurons and the specific purposes they serve.

Then we'll discuss the different AI accelerators specifically designed for the efficient processing of DNN workloads, along with the neural network optimizers that work on the learning rate to reduce overall loss and improve accuracy. Finally, we will cover various applications of DNNs across industries and explore the power of leveraging high-performance computing (HPC) with AI.

WHAT ARE NEURAL NETWORKS?

Our biological neural network receives sensations from the external environment through our five senses. Then the neurons transmit these sensations to our other nerve cells. Artificial neural networks (ANNs), in the same manner, collect the data as inputs, and the neurons consolidate all the information. Like the synapses of the biological neural network, the interconnections in an ANN transform the input data within the hidden layers. The output in the ANN gets generated like the mechanisms of the axon in a biological neural network that channelizes the output.

CONTENTS

- What Are Neural Networks?
- Essentials of Neural Networks
 - Neural Network Components
 - Common Neural Architectures
 - Neural Network Model Optimization
 - Neural Network Chips
- Conclusion
 - Additional Resources

ESSENTIALS OF NEURAL NETWORKS

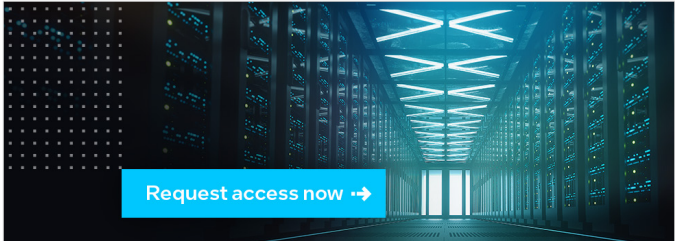
This section will first illuminate the constituent components of the neural network followed by the various network architectures that illustrate how neurons are arranged. The subsequent deliberations on neural network chips and optimizers will demonstrate a seamless implementation of the network for improved accuracy and speed.

NEURAL NETWORK COMPONENTS

It's critical to first appreciate the nuts and bolts of a neural network, which is composed of the three layers — input, hidden, and output — as well as the perceptron, activation functions, weights, and biases.


INPUT, HIDDEN, AND OUTPUT LAYERS

The single *input layer* accepts external independent variables that help in predicting the desired outcome. All the independent variables of the model are a part of the input layer. The one-to-many interconnected *hidden layers* are configured based on the purpose that the neural network is going to serve, like object detection and classification through visual recognition and NLP.



[Request access now →](#)

Test and Run AI Workloads
on the Latest Intel® Hardware
With Intel® DevCloud



Get the Most Out of Your AI Deployments From Start to Finish.

OpenVINO™ toolkit: AI Inference, optimized for amazing!



Streamlined Development

Build your AI application with common frameworks such as TensorFlow and PyTorch and then leverage the Neural Network Compression Framework.



High Performance, Deep Learning

Help your model run faster by using the OpenVINO™ toolkit Post-Training Optimization Tool.



Write Once, Deploy Anywhere

Take a single application across multiple accelerators and environments. CPUs, GPUs, VPUs, on-premises, on-device, in the browser, or in the cloud—it all works!



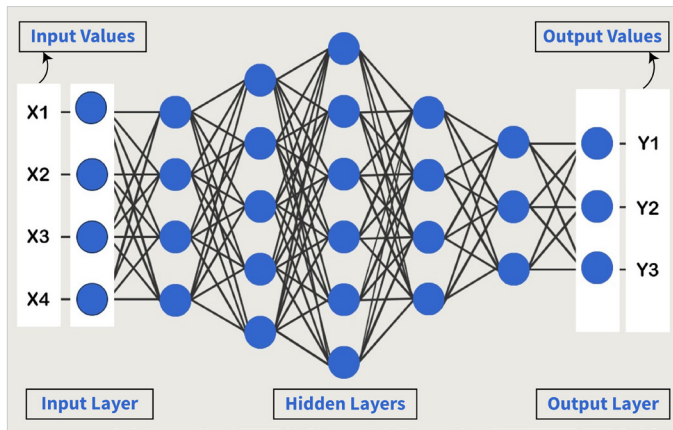
Download the OpenVINO™ Toolkit



Hidden layers are a function of the weighted sum of the inputs/predictors. When the network contains multiple hidden layers, each hidden unit is a function of the weighted sum of the units of the previous hidden layer.

The *output layer*, as a function of the hidden layers, contains the target (dependent) variables. For any image classification, the output layer segregates the input data into multiple nodes as per the desired objective of the model.

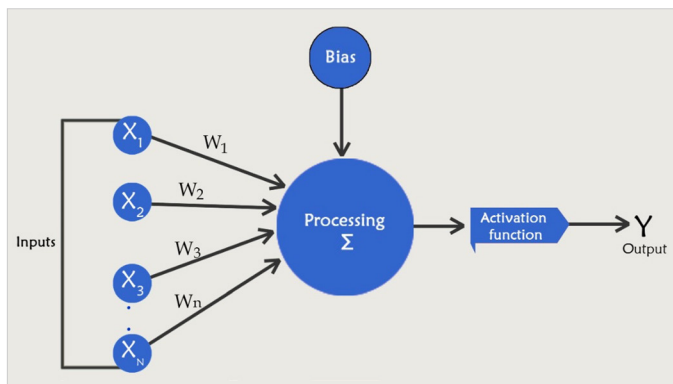
Figure 1: Input, hidden, and output layers



PERCEPTRON

Frank Rosenblatt, an experimental psychologist from Cornell, was intrigued by the ability of neurons to learn and created a simple perceptron with multiple inputs, a single processor, and a singular output. Thus, the perceptron is a building block of a neural network that comprises a single layer.

Figure 2: Perceptron



ACTIVATION FUNCTIONS

An activation function, also known as a transfer function, controls the amplitude of a neuron's output. In a deep neural network with multiple hidden layers, the activation function links the weighted sums of units in a layer to the values of units in the succeeding layer. The same activation function is used for all the hidden layers.

Activation functions can be either linear or non-linear, and the most commonly used ones are summarized in Table 1 in the next column.

Table 1

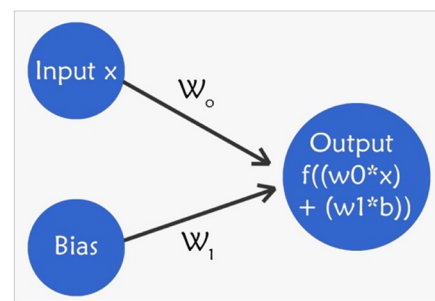
TYPE	DESCRIPTION	FORMULA
Rectified Linear Activation (ReLU)	A linear function that will output the input directly if it is positive, but if the input is negative, the output is 0	$\max(0, x)$
Logistic (Sigmoid)	An "S" curve that generates an output between 0 and 1 and is expressed as probability	$\frac{1}{1 + e^{-x}}$
Hyperbolic Tangent (TanH)	Like a Sigmoid function but symmetrical in nature and therefore produces better results; takes real-valued arguments and transforms them to the range (-1, 1)	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
Linear	Takes real-valued arguments and returns them unchanged	$f(x) = x$
Softmax	Commonly used for a classification problem with multiple classes and returns the probability of each class	$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

In case there is any confusion about which activation function to use that best suits your use case, it is advisable to use ReLU since it helps to overcome the vanishing gradient problem and allows the models to be better trained.

WEIGHTS AND BIASES

Weights signify the importance of the corresponding feature (input variable) in predicting the output. They also explain the relationship between that feature and the target output. The figure below illustrates that the output is a summation of the x (input) times the connection weight w_0 and the b (bias) times the connection weight w_1 .

Figure 3: Weights and biases



Biases are like constants in a linear function $y = mx + c$ where m = weights and c = bias. Without a bias, the model will pass through the origin only, and such scenarios are far from the reality. Thus, the bias helps in transposing the line and makes the model more flexible.

COMMON NEURAL ARCHITECTURES

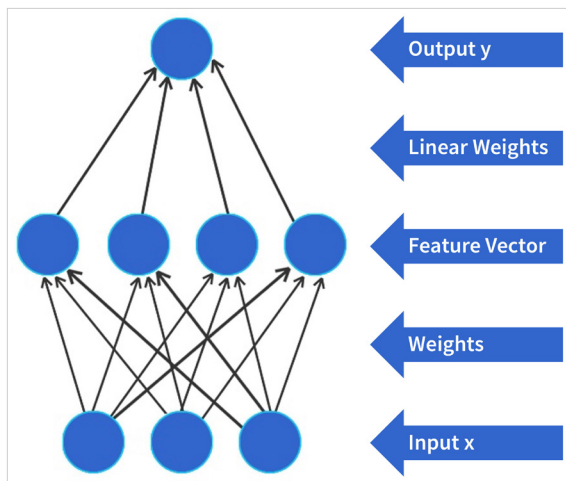
The neural network architecture is composed of neurons, and the way these neurons are arranged creates the structure that defines how the algorithm is going to learn. The arrangements have the input and the output layers with hidden layers in between that increase the model's

computational and processing power. The key architectures are discussed below.

RADIAL BASIS FUNCTION

The radial basis function (RBF) has a single non-linear hidden layer called a "feature vector," where the number of neurons in the hidden layer should be more than the number of neurons in the input layer to cast the data into a higher dimensional space. Thus, RBF increases the dimension of the feature vector to make the classification highly separable in high-dimensional space. The figure below illustrates how the inputs (x) are transformed to output (y) with through a single hidden layer (i.e., feature vector), which connects to x and y through the weights.

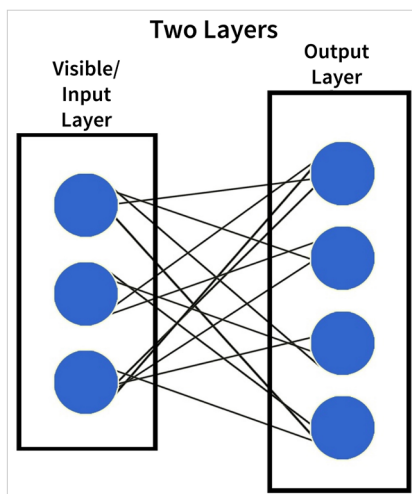
Figure 4: Radial basis function



RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann machines (RBMs) are unsupervised learning algorithms with two-layer neural networks comprising a visible/input layer and the hidden layer without any intra-layer connections — i.e., no two nodes in the layers are connected, which creates restriction. RBMs are used for recommendation engines of movies, pattern recognition (e.g., understanding handwritten text), and radar target recognition for real-time intra-pulse recognition.

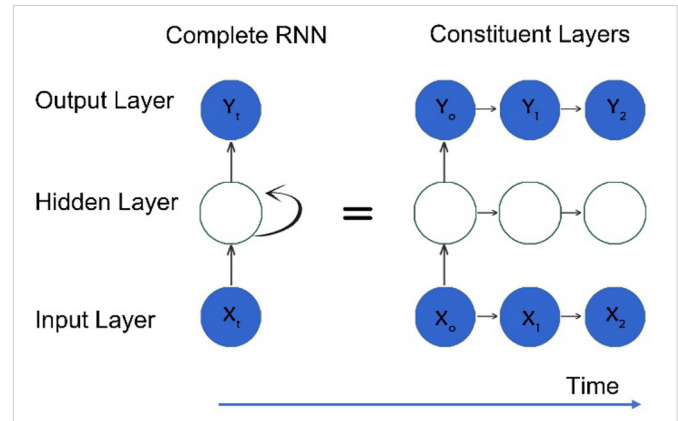
Figure 5: Restricted Boltzmann machines



RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) consider input as time series to generate output as time series with at least one connection cycle. RNNs are universal approximators: They can approximate virtually any dynamical system. RNNs are used for time series analyses like stock predictions, sales forecasting, natural language processing and translation, chatbots, image captioning, and music synthesis.

Figure 6: Recurrent neural networks



Long short-term memory (LSTM) — which is composed of forget, input, and output gates — has several applications including time series predictions and natural language understanding and generation. LSTM is primarily used to capture long-term dependencies. The forget gate decides whether to retain the information from the previous timestamp or "forget" it. A less complex variant with a smaller number of gates form the **gated recurrent unit (GRU)**.

The GRU is a simplified variant of LSTM where forget and input gates are combined into a single update gate, and the cell state and hidden state are also combined. Thus, a GRU uses less memory and is therefore faster than LSTM.

CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) are widely popular for image classification. A CNN assigns weights and biases to objects in the image for classification purposes. An image comprising a matrix of pixel values is processed through the convolutional layer, pooling layer, and fully connected (FC) layer. The pooling layer reduces the spatial size of the convolved feature.

The final output layer generates a confidence score to determine how likely it is that an image belongs to a defined class. CNNs are widely used in Facebook and other social media platforms to monitor content.

DEEP REINFORCEMENT LEARNING

deep RL, short for deep reinforcement learning, creates a perfect synergy by amalgamating the power of AI and reinforcement learning. Learning through reinforcement refers to the algorithm of rewarding for the right decision and punishing for the wrong one. Applications of deep RL include load balancing, robotics, industrial operations, traffic control, and recommendation systems.

GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) use two neural networks, a generator, and a discriminator. While the generator helps in generating image, voice, and video content, the discriminator classifies them as either from the domain or generated. The two models are trained for a zero-sum game until it's proven that the generator model is producing reasonable results.

NEURAL NETWORK CHIPS

Neural network chips provide the power of computing infrastructure through processing speed, storage, and networking that make the chips capable of quickly running neural network algorithms on vast amounts of data. Network chips break the tasks into multiple sub-tasks, which can run into multiple cores concurrently to increase the processing speed.

TYPES OF AI ACCELERATORS

Specialized AI accelerators have been designed that vary significantly depending on the model size, supported framework, programmability, learning curve, target throughput, latency, and cost. Such hardware includes the graphical processing unit (GPU), vision processing unit (VPU), field-programmable gate array (FPGA), central processing unit (CPU), and Tensor Processing Unit (TPU).

While some accelerators like GPUs are more capable of handling computer graphics and image processing, FPGAs demand field programming using hardware description languages (HDLs) like VHDL and Verilog, and TPUs by Google are more specialized for neural network machine learning. Let's look at each of them separately below to understand their capabilities.

GPUs were originally developed for graphics processing and are now widely used for deep learning (DL). Their benefit is parallel processing through the following five architectures:

- Single instruction, single data (SISD)
- Single instruction, multiple data (SIMD)
- Multiple instructions, single data (MISD)
- Multiple instructions, multiple data (MIMD)
- Single instruction, multiple threads (SIMT)

A GPU computes faster than a CPU as it devotes more transistors to data processing, which help to maximize the memory bandwidth for large datasets with medium-to-large models and larger effective batch sizes.

VPUs are optimized DL processors aimed at enabling computer vision tasks with ultra-low power requirements without compromising performance. Thus, VPUs are optimized for deep learning inference by leveraging the power of pre-trained CNN models.

An **FPGA** has thousands of memory units that run parallel architectures at low power consumption. It consists of reprogrammable logic gates to create custom circuits. FPGAs are used for autonomous driving and automated spoken language recognition and search.

CPUs with MIMD architecture are brilliant in task optimization and are more suitable for applications with limited parallelism, such as sparse DNNs, RNNs that have dependency on the steps, and small models with small effective batch sizes.

A **TPU** is Google's custom-developed application-specific integrated circuit (ASIC) that is used to accelerate DL workloads. TPUs provide high throughput for large batch sizes and are suitable for models that train for weeks, dominated by matrix computations.

AI ACCELERATORS FOR DEEP LEARNING INFERENCE

AI accelerators are required for DL inference for faster computation through parallel computational capabilities. They have high bandwidth memory that can allocate four to five times more bandwidth between processors than traditional chips. A couple of leading AI accelerators for DL inference are AWS Inferentia, a custom-designed ASIC, and Open Visual Inference and Neural Network Optimization (OpenVINO), an open-source toolkit for optimizing and deploying AI inference.

They both boost deep learning performance for performing tasks like computer vision, speech recognition, NLP, and NLG. OpenVINO uses models trained in frameworks including TensorFlow, PyTorch, Caffe, and Keras, and optimizes model performance with acceleration from CPU, GPU, VPU, and iGPU.

NEURAL NETWORK MODEL OPTIMIZATION

Deep learning model optimizations are used for various scenarios, including video analytics as well as computer vision. Since most of these computation-intensive analyses are done in real time, the following objectives are critical:

- Faster performance
- Reduced computational requirements
- Optimized space usage

For example, OpenVINO provides seamless optimization of neural networks with the help of the following tools:

- **Model Optimizer** – Converts models from multiple frameworks to Intermediate Representation (IR); these can then be concluded with OpenVINO Runtime. OpenVINO Runtime plugins are software components that comprise full implementation for inference on hardware such as CPUs, GPUs, and VPUs.
- **Post-Training Optimization Toolkit (POT)** – Accelerates the inference speed of IR models by applying post-training automated model quantization through the DefaultQuantization and AccuracyAwareQuantization algorithms.
- **Neural Network Compression Framework (NNCF)** – Integrates with PyTorch and TensorFlow to quantize and compress the model through pruning. The commonly used compression algorithms are 8-bit quantization, filter pruning, sparsity, mixed-precision quantization, and binarization.

CONCLUSION

Industries, and across specific sectors, have wide applications of AI in the current era. Keeping that in view, in this Refcard, we delved into the roots of AI algorithms, as well as explored the fundamentals of neural networks, the architectural intricacies of DNNs, and hardware requirements for the best performance. The table below presents some of the major applications of deep neural networks across the industry:

Table 2

INDUSTRY	EXAMPLES
Retail	<ul style="list-style-type: none"> Self-checkouts Automated measurement of product dimensions for space optimization in planograms Automated stock replenishment
Healthcare	<ul style="list-style-type: none"> Medical image classification CANcer Distributed Learning Environment (CANDLE)
Government	Crime prevention
Logistics and warehouse	Robots for material handling and delivery
Manufacturing	Quality control and defect classification
Automotive	Autonomous driving
Financial services	<ul style="list-style-type: none"> Fraud detection Anti-money laundering and risk analysis Loan processing Clearing and settlement of trades Options pricing

With advances in the sophistication of AI algorithms, it's become increasingly important to look beyond AI chips to reduce time to model/time to insight and increase accuracy. Since high-performance computing is a shared resource, containerization solutions like Kubernetes pave the way to give users more control and to process data at scale.

The National Center for Supercomputing Applications (NCSA) at Illinois is spearheading applications of the confluence between AI and HPC across industries ranging from genome mapping to autonomous transportation.

ADDITIONAL RESOURCES

TensorFlow is an end-to-end open-source deep learning framework developed by Google:

- TensorFlow website – <https://www.tensorflow.org>
- TensorFlow GitHub – <https://github.com/tensorflow/tensorflow>

TensorFlow Graph Neural Networks (GNNs) is a library for graph-structured data using TensorFlow:

- "Introducing TensorFlow Graph Neural Networks" – <https://blog.tensorflow.org/2021/11/introducing-tensorflow-gnn.html>
- TensorFlow GNN GitHub – <https://github.com/tensorflow/gnn>

PyTorch is a Python package to use the power of GPUs and other accelerators built on a tape-based autograd system:

- PyTorch website – <https://pytorch.org>
- PyTorch GitHub – <https://github.com/pytorch/pytorch>

Keras is a high-level neural network application programming interface (API) written in Python for DNNs that can run on top of CNTK, TensorFlow, and Theano:

- Keras website – <https://keras.io>
- Keras GitHub – <https://github.com/keras-team/keras>

OpenVINO is an open-source toolkit for optimizing and deploying AI inference:

- OpenVINO overview – <https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html>
- OpenVINO Documentation – <https://docs.openvino.ai/latest/index.html>

WRITTEN BY DR. TUHIN CHATTOPADHYAY,
FOUNDER & CEO, TUHIN AI ADVISORY



Tuhin spent the first 10 years of his career in academia and research, teaching business statistics, analytics, and technology at several reputed B-Schools of India. As a corporate practitioner, Tuhin has a proven record of accomplishment as a transformational leader in organizations like The Nielsen Company. Currently, he runs his own consultancy for providing a full suite of AI, analytics, CTO/CAO-as-a-Service, and digital transformation services to clients.



600 Park Offices Drive, Suite 300
Research Triangle Park, NC 27709
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully – and with intention – challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.

Copyright © 2022 DZone, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means of electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.