DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

DIGITAL NOTES

ON

DEEP LEARNING

(R20A6610)

Prepared by

K.Chandusha

MALLA REDDY COLLEGE OF ENGINEERING&TECHNOLOGY

(AutonomousInstitution–UGC,Govt.ofIndia)

Recognizedunder2(f)and12(B)ofUGCACT1956

(AffiliatedtoJNTUH,Hyderabad,Approved byAICTE-AccreditedbyNBA&NAAC–'A'Grade-ISO9001:2015 Certified)

Maisammaguda,Dhulapally(PostVia. Hakimpet),Secunderabad–500100,TelanganaState,India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong which involves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students into competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practicesforpromotingacademicachievementtoproduceinternationallyacceptedcompetitiveand world class professionals.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1–ANALYTICALSKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2–TECHNICALSKILLS

- Tofacilitatethegraduateswiththetechnicalskillsthatpreparethemforimmediateemploymentandpursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging pursuing higher education and research based on their interest.

PEO3–SOFTSKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4–PROFESSIONALETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting them to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B.Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1.FundamentalsandcriticalknowledgeoftheComputerSystem:-AbletoUnderstandtheworkingprinciples of the computer System and its components, Apply the knowledge to build, asses, and analyze the software and hardware aspects of it.

2.The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3.Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify their search gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals,andanengineeringspecializationtothesolutionofcomplexengineeringproblems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design / development of solutions: Design solutions for complex engineeringproblemsanddesignsystemcomponentsorprocessesthatmeetthespecifiedneedswithappropriateconsideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SYLLABUS

IV YearB. TechCSE **L/T/P/C3/-/-/-3**

(R20A6610)DEEPLEARNING

COURSEOBJECTIVES:

1. To understand the basic concepts and techniques of Deep Learning and the need of Deep Learningtechniques in real-world problems
2. TounderstandCNNalgorithmsandthewaytoevaluateperformanceofthe CNN architectures.
3. ToapplyRNNandLSTMtolearn,predictandclassifythereal-worldproblems in theparadigmsofDeepLearning.
4. Tounderstand,learnanddesignGANsfortheselectedproblems.
5. TounderstandtheconceptofAuto-encodersandenhancingGANsusingauto-encoders.

UNIT-I:

INTRODUCTIONTODEEPLEARNING:HistoricalTrendsinDeepLearning,Why DL is Growing, Artificial Neural Network, Non-linear classification example using Neural Networks: XOR/XNOR, Single/Multiple Layer Perceptron, Feed Forward Network, Deep Feed- forward networks, Stochastic Gradient –Based learning, Hidden Units, Architecture Design, Back- Propagation.

UNIT-II:

CONVOLUTION NEURAL NETWORK (CNN): Introduction to CNNs and their applications in computer vision, CNN basic architecture, Activation functions-sigmoid, tanh, ReLU, Softmax layer, Types of pooling layers, Training of CNN in TensorFlow, various popular CNN architectures: VGG, Google Net, ResNet etc, Dropout, Normalization, Data augmentation

UNIT-III

RECURRENT NEURAL NETWORK (RNN): Introduction to RNNs and their applications in sequential data analysis, Back propagation through time (BPTT), Vanishing Gradient Problem, gradient clipping Long Short Term Memory (LSTM) Networks, Gated Recurrent Units, Bidirectional LSTMs, Bidirectional RNNs.

UNIT-IV

GENERATIVE ADVERSARIAL NETWORKS (GANS): Generative models, Concept and principles of GANs, Architecture of GANs (generator and discriminator networks), Comparison between discriminative and generative models, Generative Adversarial Networks (GANs), Applications of GANs.

UNIT-V

AUTO-ENCODERS: Auto-encoders, Architecture and components of auto-encoders (encoder and decoder), Training an auto-encoder for data compression and reconstruction, Relationship between Autoencoders and GANs, Hybrid Models: Encoder-Decoder GANs.

TEXTBOOKS:

1. DeepLearning:AnMITPressBookbyIanGoodfellowandYoshuaBengioAaron Courville.
2. MichaelNielson,NeuralNetworksandDeepLearning,DeterminationPress,2015.
3. SatishKumar,Neuralnetworks:AclassroomApproach,TataMcGraw-HillEducation, 2004.

REFERENCES:

1. DeepLearningwithPython,FrancoisChollet,Manningpublications,2018
2. Advanced Deep Learning with Keras, Rowel Atienza, PACKT Publications, 2018

COURSEOUTCOMES:

CO1:Understandthebasicconceptsandtechniquesof DeepLearningandthe needofDeepLearningtechniquesinreal-worldproblems.

CO2:UnderstandCNNalgorithmsandthewaytoevaluateperformanceof theCNNarchitectures.

CO3:ApplyRNNandLSTMtolearn,predictandclassifythereal-world problemsintheparadigmsofDeepLearning.

CO4:Understand,learnanddesignGANsfortheselectedproblems.

CO5:UnderstandtheconceptofAuto-encodersandenhancingGANsusingauto-encoders.

UNIT-I:

INTRODUCTIONTODEEPLEARNING: HistoricalTrendsin Deep Learning, Why DL is Growing, Artificial Neural Network,Non-linear classification example using Neural Networks: XOR/XNOR, Single/Multiple Layer Perceptron, Feed Forward Network, Deep Feed- forward networks, Stochastic Gradient –Based learning, Hidden Units, Architecture Design, Back- Propagation, Deep learning frameworks and libraries (e.g., TensorFlow/Keras, PyTorch).

INTRODUCTIONTODEEPLEARNING:

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasinglypopular in recent years due to the advances in processing power and the availability oflarge datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and functionofthehumanbrain's biologicalneurons,andtheyaredesignedtolearnfromlarge amounts of data.

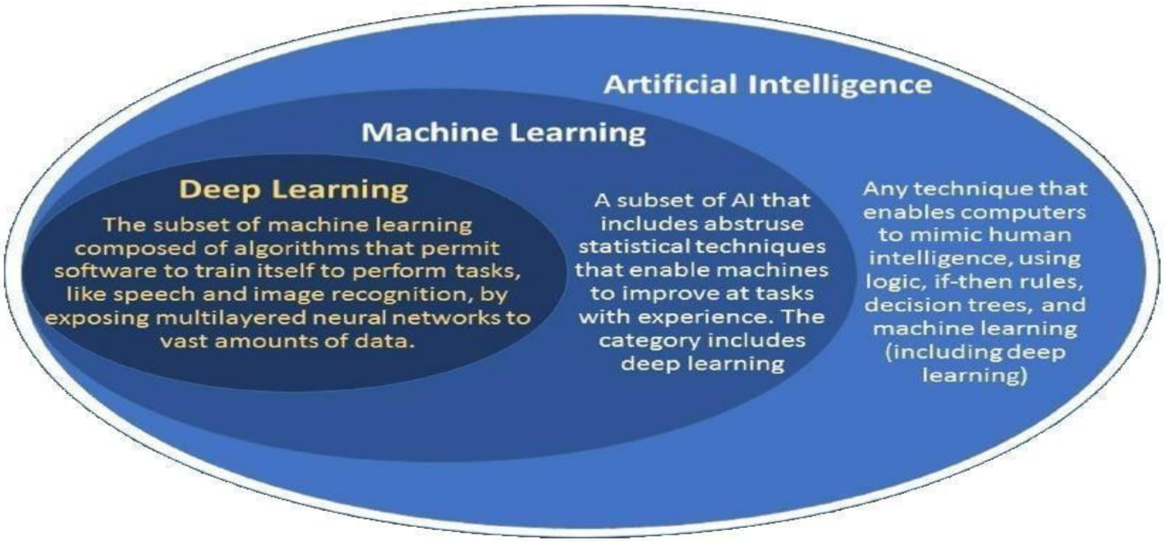
1. Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data.
2. The key characteristic of Deep Learning is the use of deep neural networks,which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns andfeatures in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering.
3. Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs).
4. Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the developmentofspecializedhardware,suchasGraphicsProcessingUnits (GPUs), has made it easier to train deep neural networks.

In summary, Deep Learning is a subfield of Machine Learning that involves the useof deep neural networks to model and solve complex problems. Deep Learning hasachievedsignificantsuccessinvariousfields,anditsuseisexpectedtocontinuetogrow as more data becomes available, and more powerful computing resources becomeavailable.

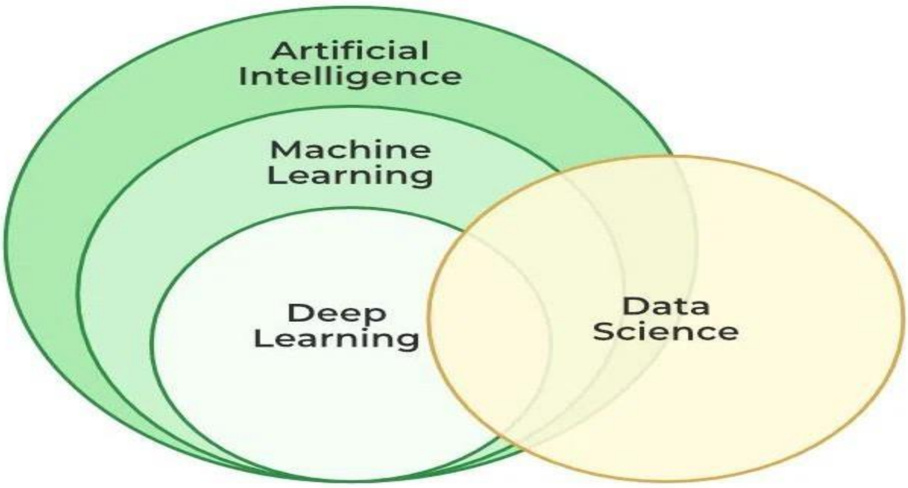
WhatisDeepLearning?

Deep learning is the branch of " Machine Learning "which is based on artificial neural network architecture. An artificial neural network or ANN uses layers of interconnected nodes called neurons that work together to process and learn from theinput data.

In a fully connected Deep neural network, there is an input layer and one or more hiddenlayersconnectedoneaftertheother.Eachneuronreceivesinputfromthe previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. The layers of the neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.



Few Insights about AI,ML and DL



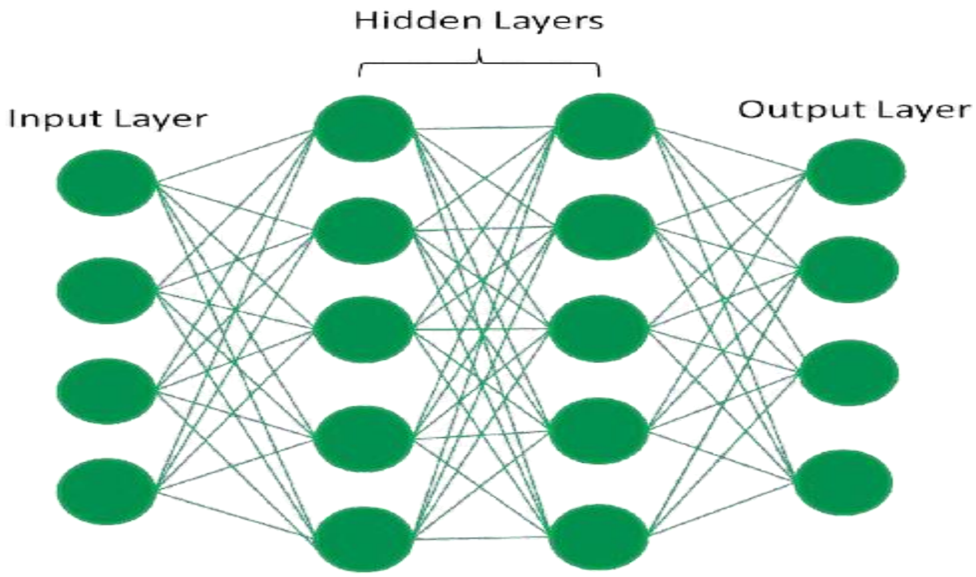
Today, Deep learning has become one of the most popular and visible areas of machine learning, due to its success in a variety of applications, such as computer vision, natural language processing, and Reinforcement learning.

Deep learning can be used for supervised, unsupervised as well as reinforcement machine learning. it uses a variety of ways to process these.

- **Supervised Machine Learning:** Supervised machine learning is the machinelearning technique in which the neural network learns to make predictions or classify data based on the labeled datasets. Here we input both input features along with the target variables. the neural network learns to make predictions based on the cost or error that comes from the difference between thepredictedandtheactualtarget,thisprocessisknownasbackpropagation. Deep learning algorithms like Convolutional neural networks, Recurrent neural networks are used for many supervised tasks like image classifications and recognition, sentiment analysis, language translations, etc.
- **UnsupervisedMachineLearning:** Unsupervisedmachinelearning is the machine learning technique in which the neural network learns to discover the patterns or to cluster the dataset based on unlabeled datasets. Here thereare no target variables. while the machine has to self-determined the hidden patterns or relationships within the datasets. Deep learning algorithms like autoencoders and generative models are used for unsupervised tasks like clustering, dimensionality reduction, and anomaly detection.
- **ReinforcementMachineLearning**: ReinforcementMachineLearning is the machinelearning techniqueinwhichanagentlearnstomakedecisionsin an environment to maximize a reward signal. The agent interacts with the environment by taking action and observing the resulting rewards. Deeplearning can be used to learn policies, or a set of actions, that maximizes the cumulativerewardovertime.Deepreinforcementlearningalgorithmslike Deep Q networks and Deep Deterministic Policy Gradient (DDPG) are used to reinforce tasks like robotics and game playing etc.

Artificialneuralnetworks:

"Artificialneuralnetworks" arebuiltontheprinciplesofthestructureand operationofhumanneurons. Itis alsoknown as neural networks or neural nets. An artificial neural network's input layer, which is the first layer, receives input from external sources and passes it on to the hidden layer, which is the second layer. Each neuron in the hidden layer gets information from the neurons in the previous layer, computes the weightedtotal,andthentransfersit tothe neuronsinthe nextlayer.Theseconnections are weighted, which means that the impacts of the inputs from the preceding layer aremore or less optimized by giving each input a distinct weight. These weights are then adjusted during the training process to enhance the performance of the model.



FullyConnectedArtificialNeuralNetwork

Artificial neurons, also known as units, are found in artificial neural networks. The wholeArtificialNeuralNetwork iscomposed oftheseartificialneurons,whichare arranged in a series of layers. The complexities of neural networks will depend on the complexities of the underlying patterns in the dataset whether a layer has a dozen units or millions of units.Commonly, Artificial Neural Network has an inputlayer,anoutputlayer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about.

Ina fullyconnectedartificialneural network,thereis aninputlayerandone or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. Then, after passing through one or more hidden layers, this data is transformed into valuable data for the output layer. Finally, the output layer provides an output in the form of an artificial neural network's response to the data that comes in.

Units are linked to one another from one layer to another in the bulk of neural networks. Each of these links has weights that control how much one-unit influences another. The neural network learns more and more about the data as it moves from oneunit to another, ultimately producing an output from the output layer.

DifferencebetweenMachineLearningandDeepLearning:

Machine learningand deep learning both are subsets of artificial intelligence but there are many similarities and differences between them.

Machine Learning	Deep Learning
Apply statistical algorithms to learn the hidden patterns and relationships in the dataset.	Uses artificial neural networkarchitecture to learn the hidden patterns and relationships in the dataset.
Canworkonthesmalleramountof dataset	Requiresthelargervolumeofdataset compared to machine learning
Betterforthelow-label task.	Better for complex task like image processing, natural language processing, etc.
Takeslesstimetotrainthemodel.	Takesmoretimetotrainthe model.
A model is created by relevant features which are manually extracted from images to detect an object in the image.	Relevant features are automatically extracted from images. It is an end-to-end learning process.
Lesscomplexandeasytointerprettheresult.	Morecomplex,itworksliketheblackbox interpretationsoftheresultarenoteasy.
It can work on the CPU or requires less computingpowerascomparedtodeep learning.	Itrequiresahigh-performancecomputer with GPU.

Typesofneuralnetworks:

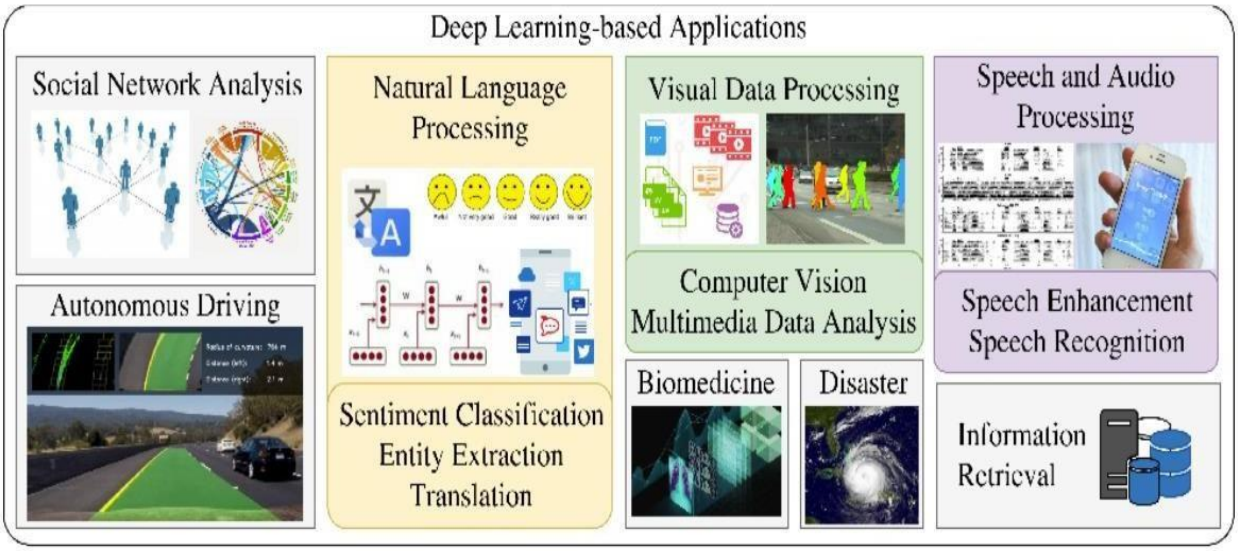
DeepLearningmodelsareabletoautomaticallylearnfeaturesfromthedata, whichmakesthemwell-suitedfortaskssuchasimagerecognition,speechrecognition, andnaturallanguageprocessing.Themostwidelyusedarchitecturesindeeplearningare feedforward neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

Feedforward neural networks (FNNs)are the simplest type of ANN, with a linear flow of information through the network. FNNs have been widely used for tasks such as image classification, speech recognition, and natural language processing.

Convolutional Neural Networks (CNNs)are specifically for image and video recognitiontasks. CNNs are able to automatically learn features from the images, which makes them well-suitedfortaskssuchasimageclassification,objectdetection,andimage segmentation.
Recurrent Neural Networks (RNNs)are a type of neural network that is able to process sequential data, such as time series and natural language. RNNs are able to maintain an internalstatethatcapturesinformationaboutthepreviousinputs,whichmakesthem well-suitedfortaskssuchasspeechrecognition,naturallanguageprocessing,and language translation.

ApplicationsofDeepLearning:

The main applications of deep learning can be divided into computer vision, natural language processing (NLP), and reinforcement learning.



Computervision

In computer vision, Deep learning models can enable machines to identify and understand visual data. Some of the main applications of deep learning in computer vision include:

- **Object detection and recognition:** Deep learning model can be used to identify and locate objects within images and videos, making it possible for machines to perform tasks such as self-driving cars, surveillance, and robotics.
- **Image classification:** Deep learning models can be used to classify images into categories such as animals, plants, and buildings. This is used in applicationssuch as medical imaging, quality control, and image retrieval.

- **Imagesegmentation:** Deeplearningmodelscanbeusedforimage segmentation into different regions, making it possible to identify specific features within images.

Naturallanguageprocessing(NLP):

In NLP, the Deep learning model can enable machines to understand and generate human language. Some of the main applications of deep learning inNLPinclude:

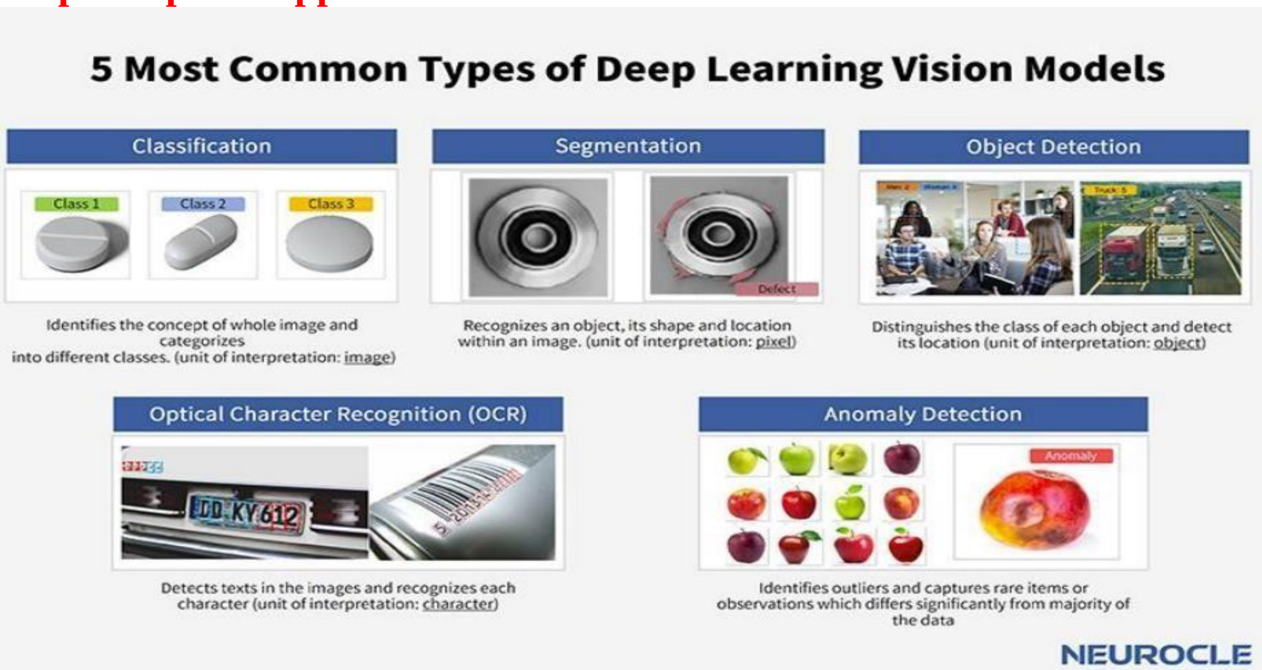
- **Automatic Text Generation** – Deep learning model can learn the corpus of text andnewtextlikesummaries,essayscanbeautomaticallygeneratedusing these trained models.
- **Language translation:** Deep learning models can translate text from one language to another, making it possible to communicate with people from different linguistic backgrounds.
- **Sentiment analysis:** Deep learning models can analyze the sentiment of a piece oftext,makingitpossibletodeterminewhetherthetextispositive,negative, or neutral. This is used in applications such as customer service, social media monitoring, and political analysis.
- **Speech recognition:** Deep learning models can recognize and transcribe spoken words, making it possible to perform tasks such as speech-to-text conversion, voice search, and voice-controlled devices.

Reinforcementlearning:

In reinforcement learning, deep learning works as training agents to take action inan environment to maximize a reward. Some of the main applications of deep learning in reinforcement learning include:

- **Game playing:** Deep reinforcement learning models have been able to beat human experts at games such as Go, Chess, and Atari.
- **Robotics:** Deep reinforcement learning models can be used to train robots to perform complex tasks such as grasping objects, navigation, and manipulation.
- **Control systems:** Deep reinforcement learning models can be used to control complex systems such as power grids, traffic management, and supply chain optimization.

PopularspecificapplicationsofDL:



2. Computational Resources: For training the deep learning model, it is computationally expensive because it requires specialized hardware like GPUs and TPUs.
3. Time-consuming: While working on sequential data depending on the computational resource it can take very large even in days or months.
4. Interpretability:Deeplearningmodelsarecomplex,itworkslikeablack box.it is very difficult to interpret the result.
5. Overfitting: when the model is trained again and again, it becomes too specializedforthetrainingdata,leadingtooverfittingandpoorperformance on new data.

AdvantagesofDeepLearning:

1. High accuracy: Deep Learning algorithms can achieve state-of-the-art performance in various tasks, such as image recognition and natural language processing.
2. Automated feature engineering: Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.
3. Scalability: Deep Learning models can scale to handle large and complexdatasets, and can learn from massive amounts of data.
4. Flexibility: DeepLearning models can be appliedto a wide range of tasks andcan handle various types of data, such as images, text, and speech.
5. Continual improvement: Deep Learning models can continually improve their performance as more data becomes available.

DisadvantagesofDeepLearning:

1. Highcomputationalrequirements:DeepLearningmodelsrequirelarge amounts of data and computational resources to train and optimize.
2. Requires large amounts of labeled data: Deep Learning models often require a large amount of labeled data for training, which can be expensive and time-consuming to acquire.
3. Interpretability:DeepLearningmodelscanbechallengingtointerpret,making itdifficulttounderstandhowtheymakedecisions. Overfitting: Deep Learning models can sometimes overfit to the training data, resulting in poor performance on new and unseen data.
4. Black-box nature: Deep Learning models are often treated as black boxes,making it difficult to understand how they work and how they arrived at their predictions.

   In summary, while Deep Learning offers many advantages,including high accuracy and scalability, it also has some disadvantages, such as high computational requirements, the need for large amounts of labeled data, andinterpretabilitychallenges.Theselimitationsneedtobecarefully considered when deciding whether to use Deep Learning for a specific task.

HistoricalTrendsinDeepLearning:

Deep learning has experienced significant historical trends since its inception. Here are some key milestones and trends that have shaped the field:

1. **Early Developments:** Deep learning traces its roots back to the 1960s with the development of Artificial Neural Networks (ANNs).
   - The idea of using interconnected nodes inspired by the human brain's structure laid the foundation for later deep learning advancements.
2. **WinterofAI:**Inthe1970sand1980s,deeplearningfacedaperiodofstagnation known as the "AI winter."
   - Limited computational power, insufficient data, and theoretical challenges hindered progress in the field, leading to decreased interest and funding.
3. **Backpropagation:** In the 1980s, the backpropagation algorithm, which efficiently trains deep neural networks, was rediscovered and popularized.
   - This breakthrough allowed for more efficient training of multi-layer neural networks, addressing some of the limitations faced during the AI winter.
4. **Rise of Convolutional Neural Networks (CNNs):** In the late 1990s and early 2000s, CNNs gained prominence in the fieldof computer vision.
   - TheLeNet-5architecturedevelopedbyYannLeCunrevolutionized image recognition tasks and demonstrated the potential of deep learning in visual perception.
5. **BigDataandGPUs:**Theearly2010smarkedaturningpointfor deeplearningwiththeadventofbigdataandtheavailabilityofpowerful Graphics Processing Units (GPUs).
   - Theabundanceoflabeleddata,combinedwithGPUacceleration, enabled the training of large-scale deep neuralnetworks and significantly improved performance.
6. ***ImageNetandDeepLearningRenaissance:***TheImageNetLargeScale VisualRecognitionChallengein2012,wonbyadeepneuralnetworkknown as AlexNet, brought deep learning into the spotlight.
   - This event sparked a renaissance in the field, encouraging researcherstoexploredeeplearningarchitecturesandtechniques across various domains.
7. **DeepLearninginNaturalLanguageProcessing(NLP):**Deeplearning

techniques, particularly recurrent neural networks(RNNs) and later transformer models, have made substantial advancements in NLP tasks.

- Models like LSTM (Long Short-Term Memory) and BERT (Bidirectional Encoder Representations from Transformers) have achieved state-of-the-art results in tasks like machine translation, sentiment analysis, and question answering.

8. **Generative Models:** The introduction of generative models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) opened up possibilities for generating realistic images, videos, and audio.

- GANs,inparticular,havedemonstratedimpressivecapabilitiesin generating synthetic data.

9. **TransferLearning andPretraining:** Transferlearninghas become a prevalent technique in deep learning, enabling models to leverage knowledge from pretraining on large datasets and then fine-tune on specific tasks.

- Thisapproachhasledtosignificantperformanceimprovementsand reduced training time, especially in scenarioswith limited labeled data.

10. **ExplainabilityandInterpretability:**Asdeeplearningmodels have become increasingly complex, researchers havefocused on improving their explainability and interpretability.

- Techniques like attention mechanisms, saliency maps, and model-agnosticinterpretabilitymethodsaimtoshedlightonthe decision-making processes of deep learning models.

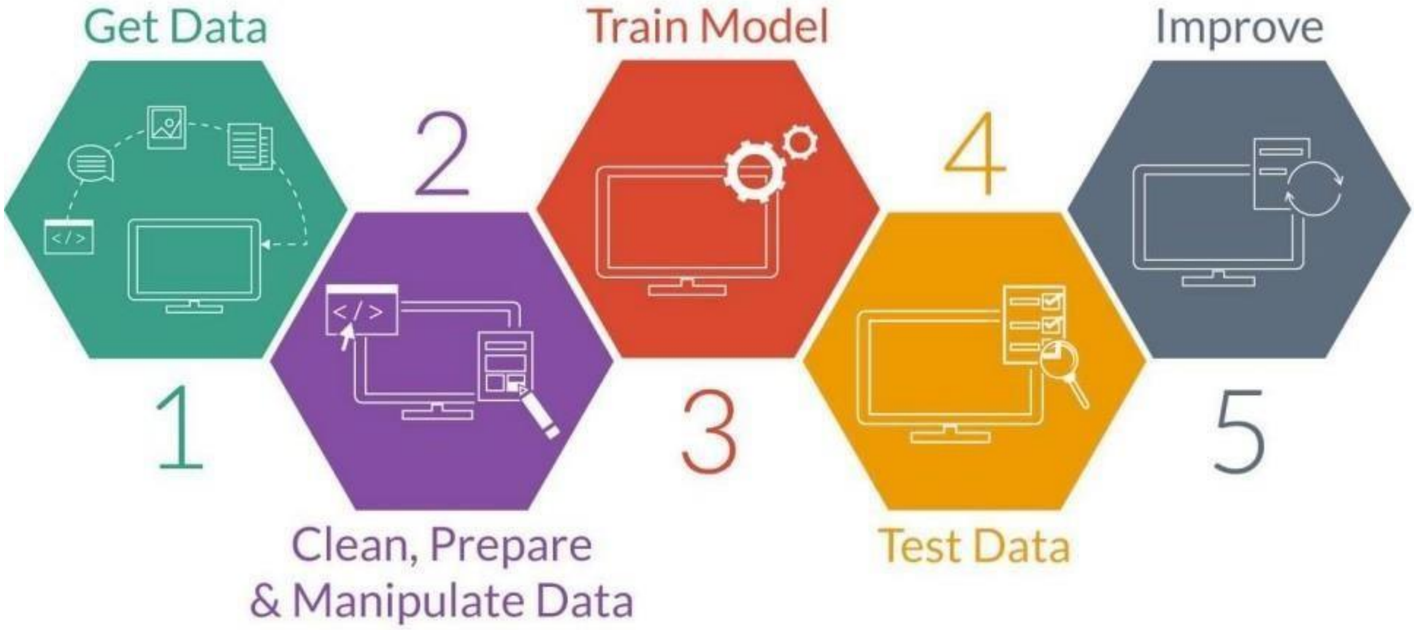
Why DLisGrowing:

- ProcessingpowerneededforDeeplearningisreadilybecomingavailable using GPUs, Distributed Computing and powerful CPUs.

- Moreover,asthedataamountgrows,DeepLearningmodelsseemto outperform Machine Learning models.

- Focusoncustomizationandrealtime decision.

- Uncover patterns that is hard to detect using traditional techniques. Find latent features (super variables) without significant manual feature engineering.



Processin ML/DL:

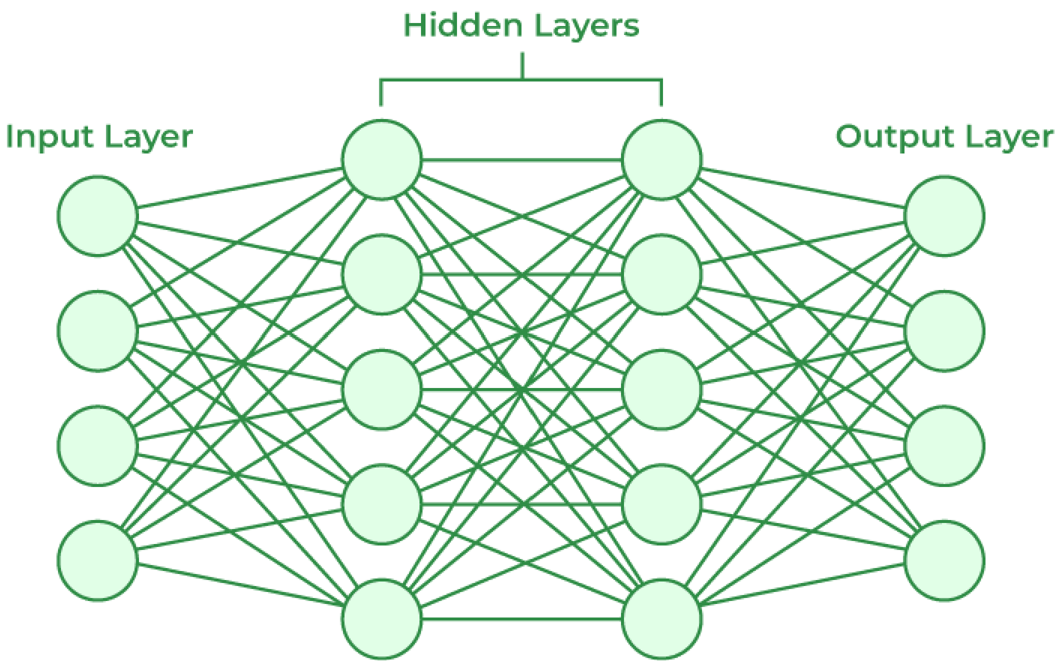
ArtificialNeuralNetworks:

Artificial Neural Networks contain artificial neurons which are called **units**. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system.

A layer can have only a dozen units or millions of units as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset. Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers.

The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

In the majority of neural networks, units are interconnected from one layer toanother. Each of these connections has weights that determine the influence of one unit on another unit. As the data transfers from one unit to another, the neural network learns more and more about the data which eventually results in an output from the output layer.



Thestructuresandoperationsofhumanneuronsserveasthebasisforartificial neural networks. It is also known as neural networks or neural nets. The input layer of an artificial neural network is the first layer, and it receives input from external sources and releases it to the hidden layer, which is the second layer. In the hidden layer, each neuron receives input from the previous layer neurons, computes the weighted sum, and sends it to the neurons in the next layer.

These connections are weighted means effects of the inputs from the previous layer are optimized more or less by assigning different-different weights to each input and it is adjusted during the training process by optimizing these weights for improved model performance.

ArtificialneuronsvsBiologicalneurons

The concept of artificial neural networks comes from biological neurons found in animal brains So they share a lot of similarities in structure and function wise.

- **Structure**: The structure of artificial neural networks is inspired by biological neurons. A biological neuron has a cell body or soma to process the impulses, dendrites to receive them, and an axon that transfers them to other neurons.The input nodes of artificial neural networks receive input signals, the hidden layer nodes compute these input signals, and the output layer nodes compute the final output by processing the hidden layer's results using activation functions.

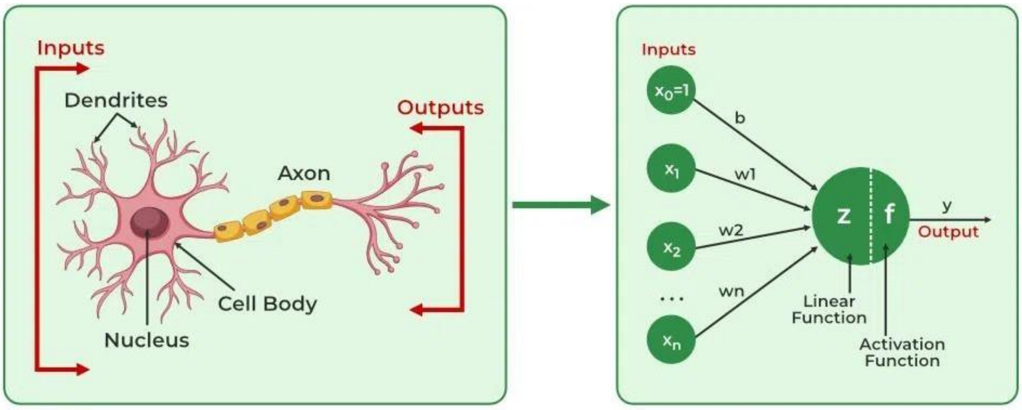
BiologicalNeuron	ArtificialNeuron
Dendrite	Inputs

BiologicalNeuron	ArtificialNeuron
CellnucleusorSoma	Nodes
Synapses	Weights
Axon	Output

- **Synapses**: Synapses are the links between biological neurons that enable the transmission of impulses from dendrites to the cell body. Synapses are theweightsthatjointheone-layernodestothenext-layernodesinartificial neurons. The strength of the links is determined by the weight value.
- **Learning**: In biological neurons, learning happens in the cell body nucleus or soma,whichhasanucleusthathelpstoprocesstheimpulses.Anaction potential is produced and travels through the axons if the impulses are powerful enough to reach the threshold. This becomes possible by synaptic plasticity,which represents the ability of synapses to become stronger or weaker over timein reaction to changes in their activity. In artificial neural networks, backpropagation is a technique used for learning, which adjusts the weights betweennodesaccordingtotheerror ordifferences betweenpredictedand actual outcomes.

BiologicalNeuron	ArtificialNeuron
Synapticplasticity	Backpropagations

- **Activation**:Inbiologicalneurons,activationisthe firing rate oftheneuron which happens when the impulses are strong enough to reach the threshold. In artificial neural networks, A mathematical function known as an activation function maps the input to the output, and executes activations.



HowdoArtificialNeuralNetworkslearn?

Artificial neural networks are trained using a training set. For example, suppose you want to teach an ANN to recognize a cat. Then it is shown thousands of different images of catssothatthenetworkcanlearntoidentifyacat.Oncetheneuralnetworkhasbeen trained enough using images of cats, then you need to check if it can identify cat images correctly. This is done by making the ANN classify the images it is provided by deciding whether they are cat images or not.The output obtained by the ANN is corroborated by a human-provided description of whether the image is a cat image or not.

If the ANN identifies incorrectly then back-propagation is used to adjust whatever it has learned during training. Backpropagationis done by fine-tuning the weights of the connections in ANN units based on the error rate obtained. This process continues until the artificial neural network can correctly recognize a cat in an image with minimal possibleerror rates.

WhatarethetypesofArtificialNeuralNetworks?

- **Feedforward Neural Network:** The feedforward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided travels in a single direction. It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist. So, the feedforward neural network has a front-propagated wave only and usually doesnot have backpropagation.
- **Convolutional Neural Network:** A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layer that uses a convolution operationontheinputandthenpassestheresultobtainedintheformofoutput to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.
- **Modular Neural Network:**A Modular Neural Network contains a collection of different neural networks that work independently towards obtaining the output withnointeractionbetweenthem.Eachofthedifferentneuralnetworks

performs a different sub-task by obtaining unique inputs compared to other networks. The advantage of this modular neural network is that it breaks down a large and complex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.

- **Radial basis function Neural Network:**Radial basis functions are those functions that consider the distance of a point concerning the center. RBF functions have two layers.In the first layer, the input is mapped into all theRadial basis functions in the hidden layer and then the output layer computes the output in the next step. Radial basis function nets are normally used to model the data that represents any underlying trend or function.
- **Recurrent Neural Network:**The Recurrent Neural Network saves the output ofalayerandfeedsthisoutputbacktotheinputtobetterpredicttheoutcomeof the layer. The first layer in the RNN is quite similar to the feed-forward neural network and the recurrentneuralnetwork starts once the outputof the firstlayeriscomputed.Afterthislayer,eachunitwillremembersomeinformationfrom thepreviousstepsothatitcanactasamemorycellinperforming computations.

ApplicationsofArtificialNeuralNetworks

1. **Social Media:** Artificial Neural Networks are used heavily in Social Media. For example,let'stakethe **'Peopleyoumayknow'** featureonFacebookthat suggestspeoplethatyoumightknowinreallifesothatyoucansendthem friend requests. Well, this magical effect is achieved by using Artificial Neural Networks that analyze your profile, your interests, your current friends, and also their friends and various other factors to calculate the people you mightpotentially know. Another common application of Machine Learningin social media is **facial recognition**. This is done by finding around 100 reference points on the person's face and then matching them with those already available in the database using convolutional neural networks.
2. **Marketing and Sales:** When you log onto E-commerce sites like Amazon and Flipkart, they will recommend your products to buy based on your previous browsing history. Similarly, suppose you love Pasta, then Zomato, Swiggy, etc. will show you restaurant recommendations based on your tastes and previous orderhistory.Thisistrueacrossallnew-agemarketingsegmentslikeBook sites,Movieservices,Hospitalitysites,etc.anditisdonebyimplementing **personalized marketing**. This uses Artificial Neural Networks to identify the customer likes, dislikes, previous shopping history, etc., and thentailor the marketing campaigns accordingly.
3. **Healthcare**: Artificial Neural Networks are used in Oncology to train algorithms thatcanidentify canceroustissueatthemicroscopiclevelatthesameaccuracy as trained physicians. Various rare diseases may manifest in physical characteristics and can be identified in their premature stages by using **Facial Analysis** on the patient photos. So the full-scale implementation of Artificial Neural Networks in the healthcare environment can only enhance the diagnostic abilities of medical experts and ultimately lead to the overall improvement in the quality of medical care all over the world.
4. **Personal Assistants:** Applications like Siri, Alexa, Cortana, etc., and also heard thembasedonthephonesyouhave!!!Thesearepersonalassistantsandan

exampleofspeechrecognitionthatuses **NaturalLanguageProcessing** to interact with the users and formulate a response accordingly. Natural Language Processing uses artificial neural networks that are made to handle many tasks of these personal assistants such as managing the language syntax, semantics,correct speech, the conversation that is going on, etc.

NeuralNetwork,Non-linearclassificationexampleusingNeural Networks: XOR/XNOR:

XORproblemwithneuralnetworks:

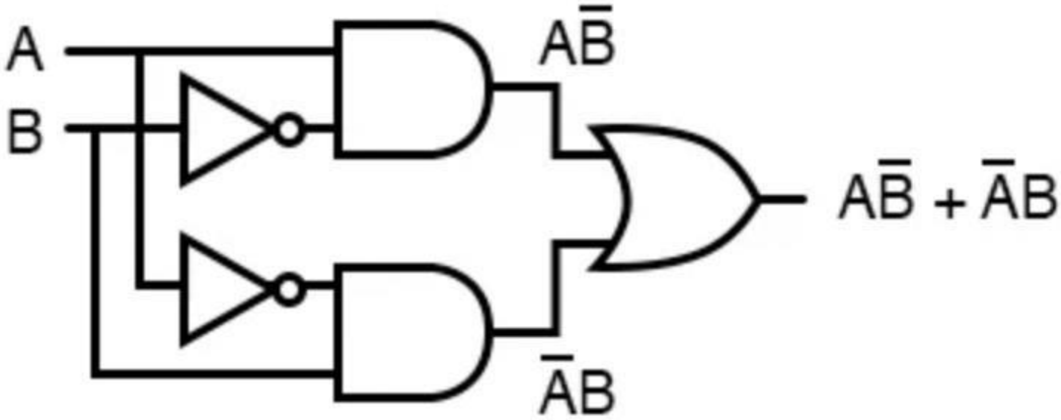
Among various logical gates, the XOR or also known as the "exclusive or" problem is one of the logical operations when performed onbinaryinputs that yield output for different combinations of input, and for the same combination of input no output is produced. The outputs generated by the XOR logic are notlinearlyseparable in the hyperplane. So, in this article let us see what is the XOR logic and how to integrate the XOR logic using neural networks.

From the below truth table, it can be inferred that XOR produces an output for different states of inputs and for thesame inputs the XOR logic does not produce any output. The Output of XOR logic is yielded by the equation as shown below.

X	Y	Output
0	0	0
0	1	1
1	0	1
1	1	0

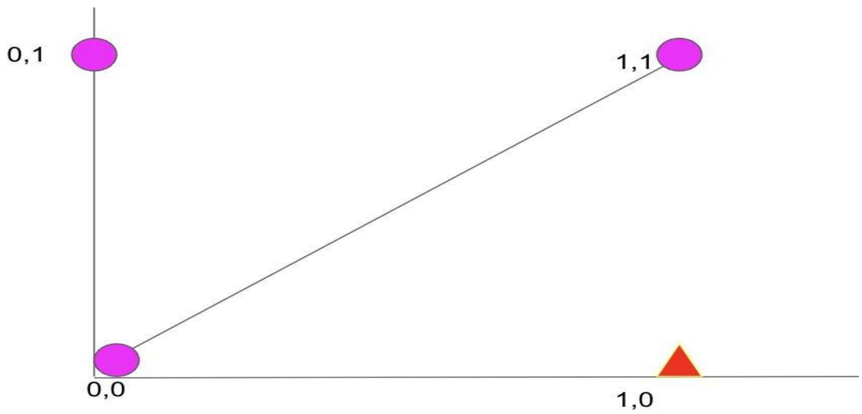
Output=X.Y'+X'.Y

The XOR gate can be usually termed as a combination of NOT and AND gates and this type of logic finds its vast application in cryptography and fault tolerance. The logical diagram of an XOR gate is shown below.



Thelinearseparabilityofpoints

Linearseparabilityofpoints is the ability to classify the datapoints in thehyperplane by avoiding the overlapping of the classes in the planes. Each of the classes should fall above or below the separating line and then they are termed as linearly separable data points. With respect to logical gates operations like AND or OR the outputs generated by this logic are linearly separable in the hyperplane. The linear separable data points appear to be as shown below.



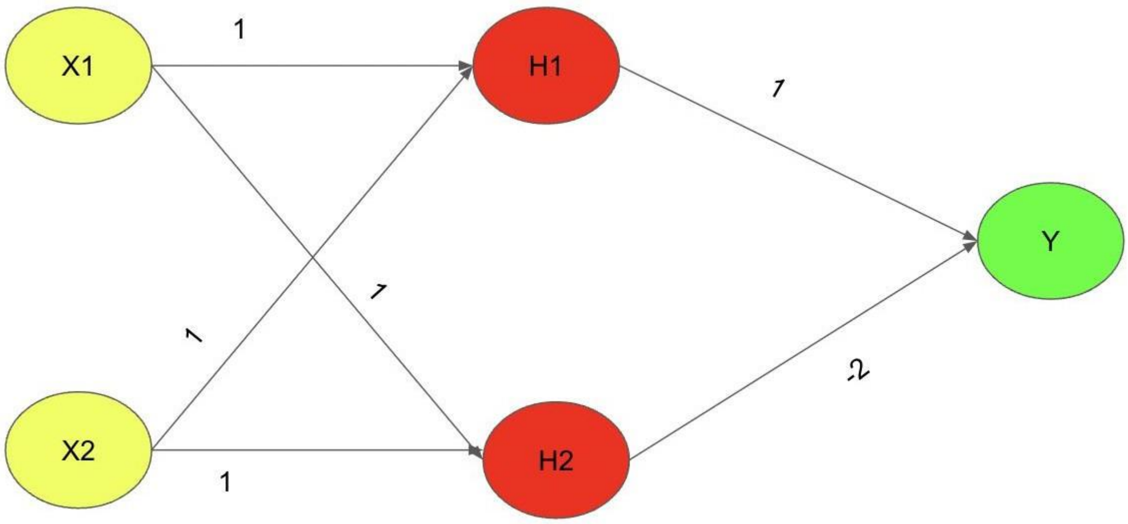
So here we can see that the pink dots and red triangle points in the plot do not overlap each other and the linear line is easily separating the two classes where the upper boundary of the plot can be considered as one classification and the below region can be considered as the other region of classification.

Needforlinearseparabilityinneuralnetworks

Linear separability is required in neural networks is required asbasic operations of neural networks would be in N-dimensional space and the data points of the neural networks have to be linearly separable to eradicate the issues with wrong weight updation and wrong classifications Linear separability of data is also considered as one of the prerequisites which help in the easy interpretation of input spaces into points whether the network is positive and negative and linearly separate the data pointsin the hyperplane.

HowtosolvetheXORproblemwithneuralnetworks:

The XORproblemwith neural networkscan be solved byusing Multi-Layer Perceptrons or a neural network architecture with an input layer, hidden layer, and output layer. So during the forward propagationthrough the neural networks, the weights get updated to the corresponding layers and the XOR logic gets executed. The Neural network architecture to solve the XOR problem will be as shown below.



So with this overall architecture and certain weight parameters between each layer, the XOR logic output can be yielded through forward propagation. The overall neural network architecture uses the ReLu activation function to ensure the weights updated in each of the processes to be 1 or 0 accordingly where for the positive set of weights the output at the particular neuron will be 1 and for a negative weight updation at the particular neuron will be 0 respectively. So let us understand one outputfor the first input state

Example:ForX1=0andX2=0weshouldgetaninputof0.Letussolveit.

Solution:

ConsideringX1=0andX2=0

H1=RELU(0.1+0.1+0)=0

H2=RELU(0.1+0.1+0)=0

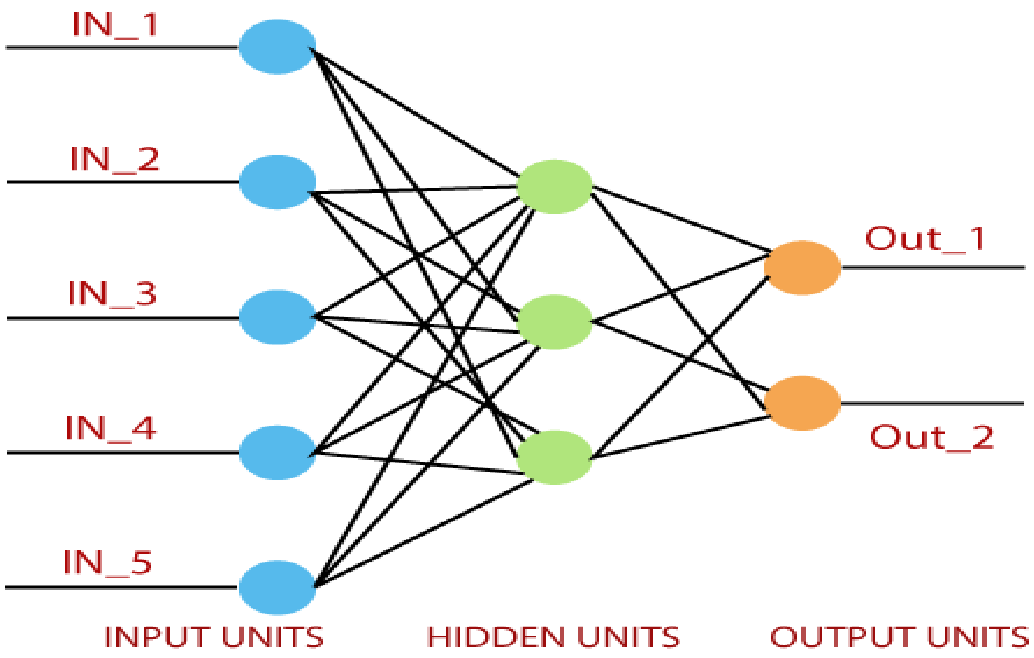
So now we have obtained the weights that were propagated from the input layertothehidden layer. Now,letus propagate fromthehiddenlayer to the output layer.

Y=RELU(0.1+0.(-2))=0

This is how multi-layer neural networks or also known as Multi-Layer perceptrons (MLP) are used to solve the XOR problem and for all other input sets the architecture provided above can be verified and the right outcome for XOR logic can be yielded.

So, amongthevariouslogicaloperations,XORlogical operationisone such problem wherein linear separability of data points is not possible using single neurons or perceptrons. So, for solving the XOR problem for neural networks it is necessary to use multiple neurons in the neural network architecture with certain weights and appropriate activation functions to solve the XOR problem with neural networks.

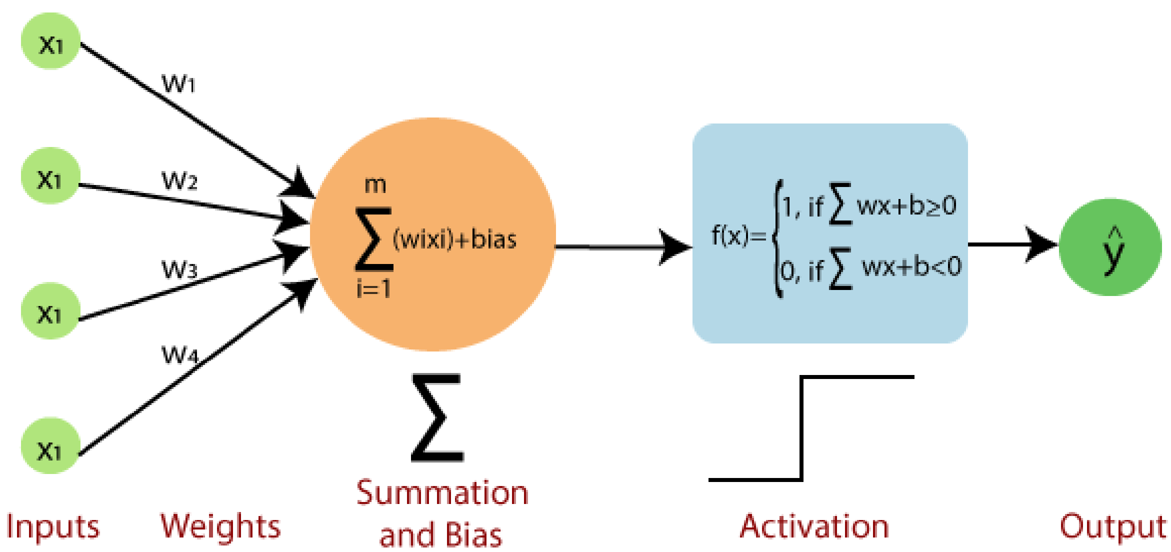
A perceptron is a neural network unit that does a precise computationtodetectfeaturesintheinputdata.Perceptronismainlyused toclassifythedataintotwoparts.Therefore,itisalsoknownas**Linear BinaryClassifier**.



Perceptron uses the step function that returns +1 if the weightedsum of itsinput 0 and -1.

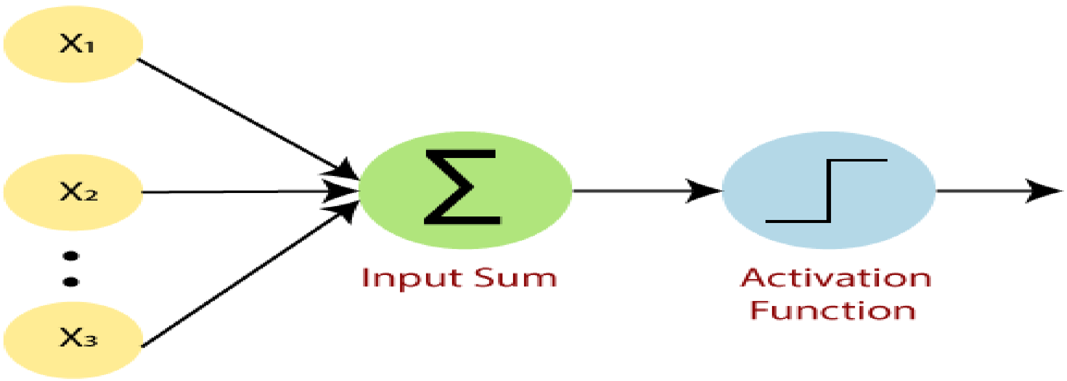
The activation function is used to map the input between the required valuelike (0, 1) or (-1, 1).

Aregularneuralnetworklookslikethis:

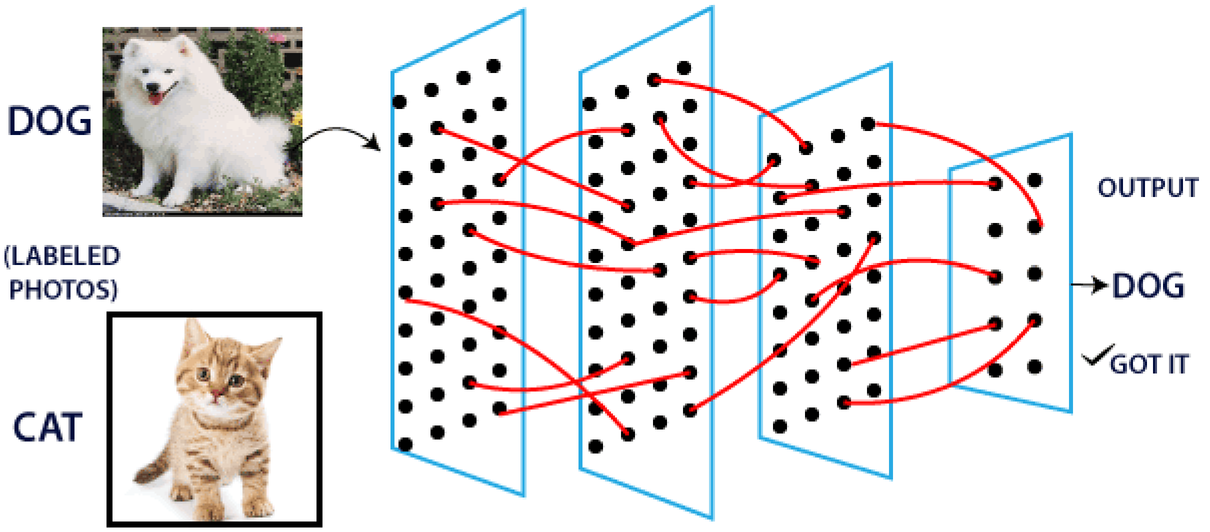


Theperceptronconsistsof4parts.

- **InputvalueorOneinputlayer:**Theinputlayeroftheperceptronismadeof artificial input neurons and takes the initial data into the system for further processing.
- **WeightsandBias:**
  Weight: It represents the dimension or strength of the connection between units.Iftheweighttonode1tonode2hasahigherquantity,thenneuron1 has a more considerable influence on the neuron.
  Bias: It is the same as the intercept added in a linear equation. It is an additionalparameterwhichtaskistomodifytheoutputalongwiththe weighted sum of the input to the other neuron.
- **Netsum:**Itcalculatesthetotalsum.
- **ActivationFunction:** Aneuroncanbeactivatedornot,isdeterminedbyan activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



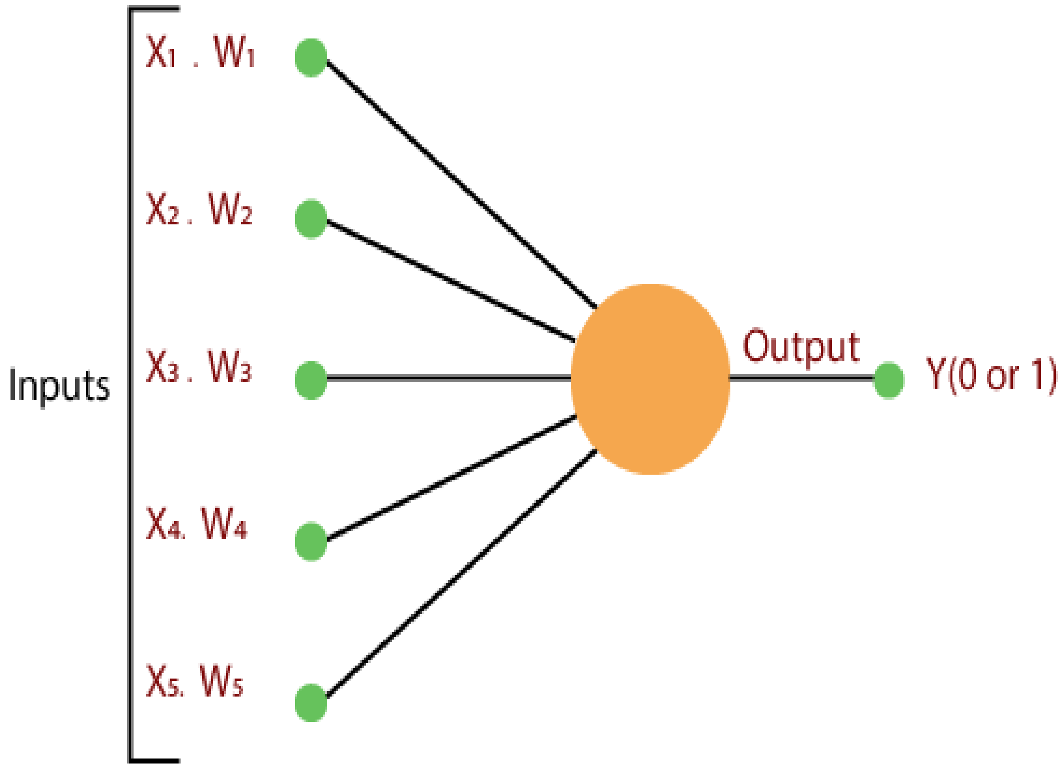
Astandardneuralnetworklookslikethebelowdiagram.



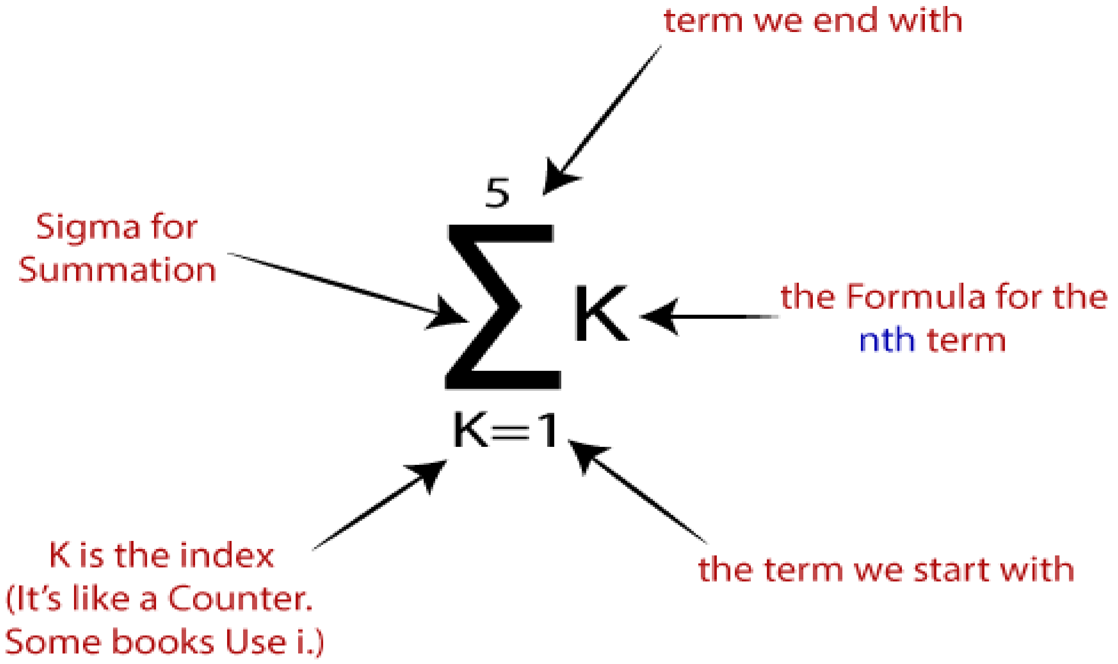
How doesitwork?

Theperceptronworksonthesesimplestepswhicharegiven below:

a. Inthefirststep,alltheinputsxaremultipliedwiththeirweights**w**.



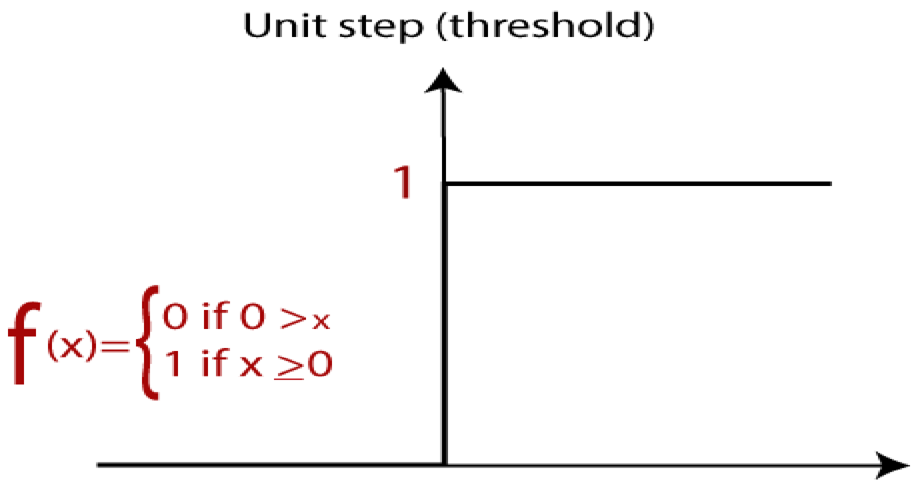
b. Inthisstep,addalltheincreasedvaluesandcallthemthe**Weightedsum**.



c. Inthelaststep,applytheweightedsumtoacorrect**ActivationFunction**. **For**

Example:

AUnitStepActivationFunction,



Therearetwotypesofarchitecture.Thesetypesfocusonthefunctionalityof artificial neural networks as follows-

- SingleLayer Perceptron

- Multi-LayerPerceptron

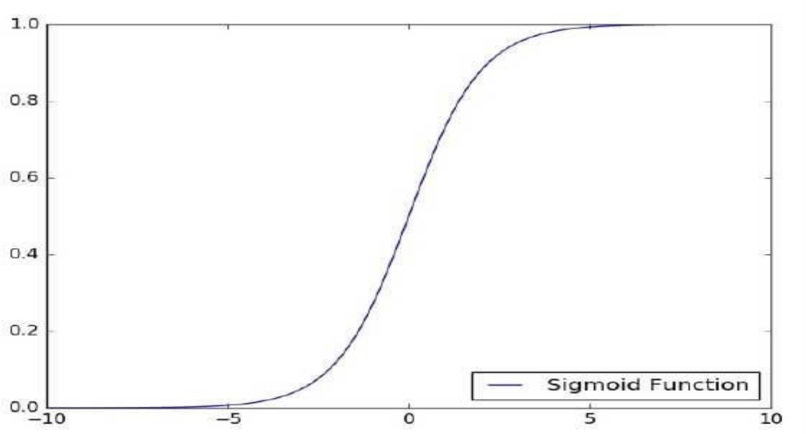
SingleLayerPerceptron

The single-layer perceptron was the first neural network model, proposed in 1958 by Frank Rosenbluth. It is one of the earliest models for learning. Our goal is to find a linear decision function measured by the weight vector w and the bias parameter b.

To understand the perceptron layer, it is necessary to comprehend artificial neural networks (ANNs). The artificial neural network (ANN) is an information processing system, whose mechanism is inspired by the functionality of biological neural circuits. An artificial neural network consists of several processing units thatare interconnected.

This is the first proposal when the neural model is built. The content of the neuron's local memory contains a vector of weight. The single vector perceptron is calculatedbycalculatingthesumoftheinputvectormultipliedbythecorresponding element of the vector, with each increasing the amount of the corresponding component of the vector by weight. The value that is displayed in the output is the input of an activation function.

Let us focus on the implementation of a single-layer perceptron for an image classification problem using TensorFlow. The best example of drawing a single-layer perceptron is through the representation of "**logistic regression**."



Now,wehavetodothefollowingnecessary stepsoftraininglogisticregression-

- The weights are initialized with the random values at the origination of eachtraining.

- For each element of the training set, the error is calculated with the difference between the desired output and the actual output. The calculated error isused to adjust the weight.
- The process is repeated until the fault made on the entire training set is less than the specified limit until the maximum number of iterations has been reached.

we will understand the concept of a multi-layer perceptron and its implementation in Python using the TensorFlow library.

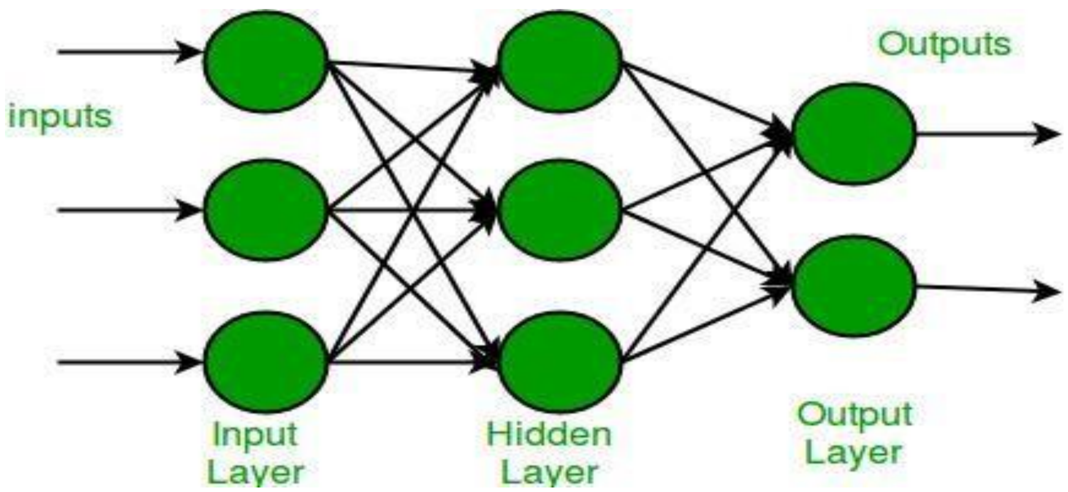
Multi-layerPerceptron:

Multi-layerperceptionis alsoknownas MLP.Itis fully connecteddense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network, we combine neurons together so that the outputs of some neurons are inputs of other neurons.

Agentleintroductionto**neuralnetworks&TensorFlow**canbefoundhere:

- NeuralNetworks
- IntroductiontoTensorFlow

A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output andit can have any number of hidden layers and each hidden layer can have any numberofnodes.AschematicdiagramofaMulti-LayerPerceptron(MLP)isdepicted below.



In the multi-layer perceptron diagram above, we can see that there are three inputsand thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwardstheir output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoidactivationfunctiontakesrealvaluesasinputandconvertsthemtonumbers between 0 and 1 using the sigmoid formula.

FeedForwardNetwork:

Whyareneuralnetworks used?

Neuronal networks can theoretically estimate any function, regardless of its complexity. Supervised learning is a method of determining the correct Y for a fresh X by learning a function that translates a given X into a specified Y. But what are the differences between neural networks and other methods of machine learning? The answer is based on the Inductive Bias phenomenon, a psychological phenomenon.

Machine learning models are built on assumptions such as the one where X and Y are related. An Inductive Bias of linear regression is the linear relationship between X and Y. In this way, a line or hyperplane gets fitted to the data.

When X and Y have a complex relationship, it can get difficult for a LinearRegression method to predict Y. For this situation, the curve must be multi-dimensional or approximate to the relationship.

A manual adjustment is needed sometimes based on the complexity of the function and thenumberoflayers within thenetwork. In most cases, trialand error methods combined with experience get used to accomplishingthis. Hence, this is the reason these parameters are called hyperparameters.

Whatisa feedforwardneural network?

Feed forward neural networks are artificial neural networksin which nodes do not form loops. This type of neural network is also known as a multi-layer neural network as all information is only passed forward.

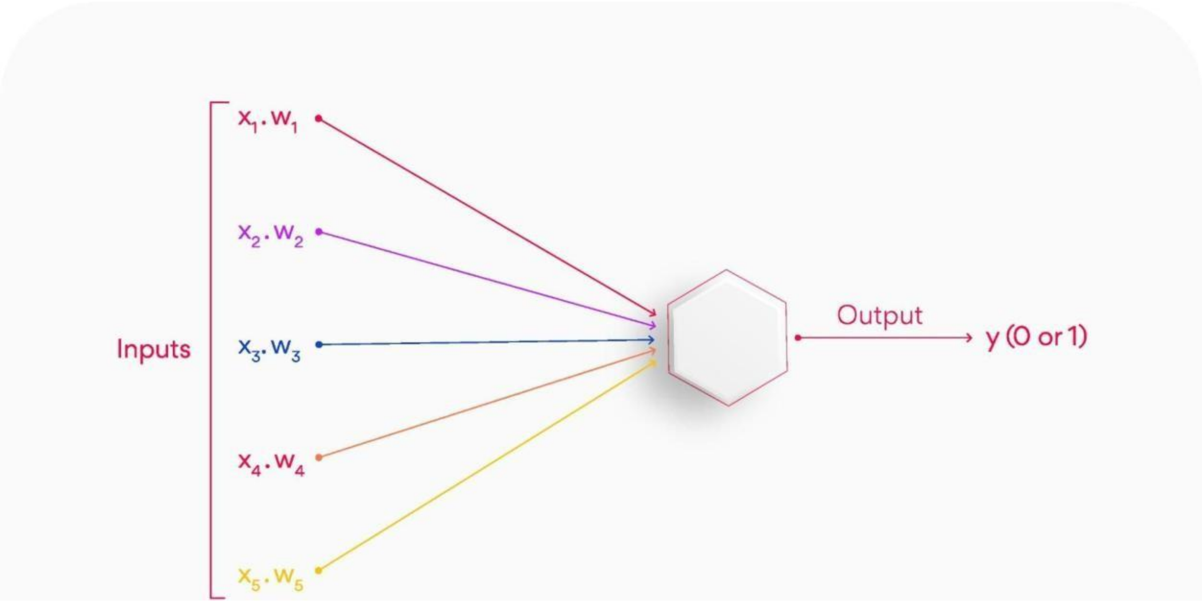
During data flow, input nodes receive data, which travel through hidden layers, and exit output nodes. Nolinks exist in the network that could get used to bysending information back from the output node.

Afeed forwardneuralnetworkapproximatesfunctionsinthefollowingway:

- Analgorithm calculatesclassifiers byusingthe formulay=f* (x).
- Inputxisthereforeassignedtocategoryy.
- According to the feed forward model, y = f (x; θ). This value determines the closestapproximation of the function.

Feed forward neural networks serve as the basis for object detection in photos, as shown in the Google Photos app.

Whatistheworkingprincipleofafeedforwardneuralnetwork?



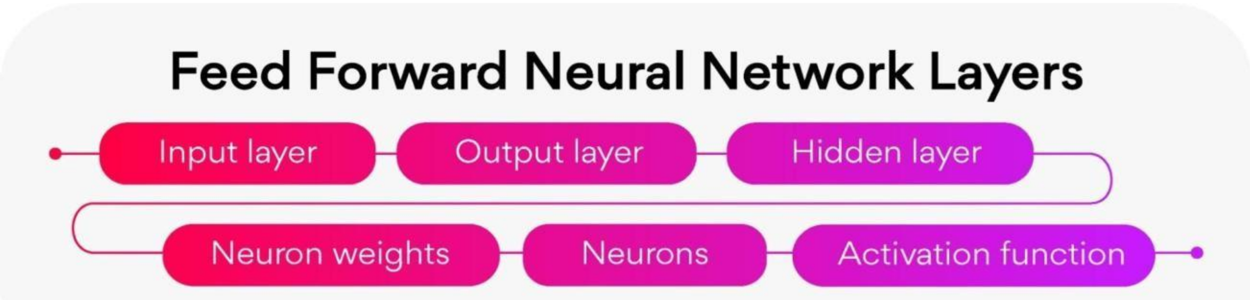
When the feed forward neural network gets simplified, it can appear as a single layer perceptron.

This model multiplies inputs with weights as they enter the layer. Afterward, the weighted input values get added together to get the sum. As long as the sum of the values rises aboveacertain threshold, set at zero,theoutput valueis usually1, whileifit falls below the threshold, it is usually -1.

As a feed forward neural network model, the single-layer perceptron often gets used for classification. Machine learning can also get integrated into single-layer perceptrons. Through training, neural networks can adjust their weights based on a property called the delta rule, which helps them compare their outputs with the intended values.

As a result of training and learning, gradient descent occurs. Similarly, multi-layered perceptrons update their weights. But, this process gets known as back-propagation. If this is the case, the network's hidden layers will get adjusted according to the output valuesproduced by the final layer.

Layersof feedforwardneuralnetwork



- Inputlayer:

The neurons of this layer receive input and pass it on to the other layers of the network. Feature or attribute numbers in the dataset must match the number of neurons in the input layer.

- Outputlayer:

According to the type of model getting built, this layer represents the forecasted feature.

- Hiddenlayer:

Input and output layers get separated by hidden layers. Depending on the type of model, there may be several hidden layers.

There are several neurons in hidden layers that transform the input beforeactually transferring it to the next layer. This network gets constantly updated with weights in order to make it easier to predict.

- Neuronweights:

Neurons get connected by a weight, which measures their strength or magnitude. Similartolinearregression coefficients,inputweightscanalsogetcompared. Weight is normally between 0 and 1, with a value between 0 and 1.

- Neurons:

Artificial neurons get used in feed forward networks, which later get adapted from biological neurons. A neural network consists of artificial neurons. Neurons functionin two ways: first, they create weighted input sums, and second, they activate the sums to make them normal.

Activation functions can either be linear or nonlinear. Neurons have weights based on their inputs. During the learning phase, the network studies these weights.

- ActivationFunction:

Neurons are responsible for making decisions in this area. According to the activation function, the neurons determine whether to make a linear or nonlinear decision. Since it passes through so many layers, it prevents the cascading effect from increasing neuron outputs.

An activation function can be classified into three major categories: sigmoid, Tanh, and Rectified Linear Unit (ReLu).

a) Sigmoid:

Input values between0and1getmappedtotheoutputvalues.

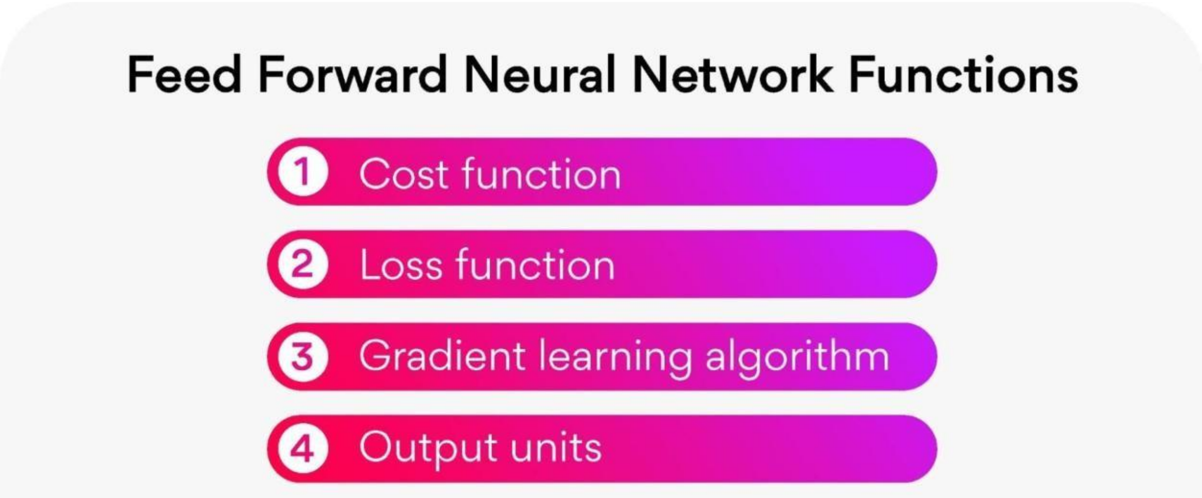
b) Tanh:

A valuebetween-1and 1getsmappedto theinputvalues.

c) RectifiedLinearUnit:

Onlypositivevaluesareallowedtoflowthroughthisfunction.Negative values get mapped to 0.

Functioninfeedforwardneuralnetwork:



Cost function

In a feed forward neural network, the cost function plays an important role.The categorized data points are little affected by minor adjustments to weights and biases. Thus, a smooth cost function can get used to determine a method ofadjusting weights and biases to improve performance.

Followingisa definitionofthemeansquareerrorcostfunction:

$$C(w, b) \equiv \frac{1}{2n} \sum_{x} \|y(x) - a\|^2.$$

Where,

w=theweightsgatheredinthenetwork b =

biases

n= numberofinputsfortraining

a=outputvectors x

= input

‖v‖=vectorv'snormallength

Lossfunction

The loss function of a neural network gets used to determine if an adjustment needs to be made in the learning process.

Neurons in the output layer are equal to the number of classes. Showing the differences between predicted and actual probability distributions. Following is the cross-entropy loss for binary classification.

Cross Entropy Loss:

$$L(\Theta) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

Asa resultofmulticlasscategorization,across-entropyloss occurs:

Cross Entropy Loss:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log\left(\hat{y}_i\right)$$

Gradientlearning algorithm

In the gradientdescentalgorithm, the next point gets calculatedbyscaling the gradient at the current position by a learning rate. Then subtracted from the current position by the achieved value.

To decrease the function, it subtracts the value (to increase, it would add). As an example, here is how to write this procedure:

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

The gradient gets adjusted by the parameter η, which also determines thestep size. Performance is significantly affected by the learning rate in machine learning.

Output units

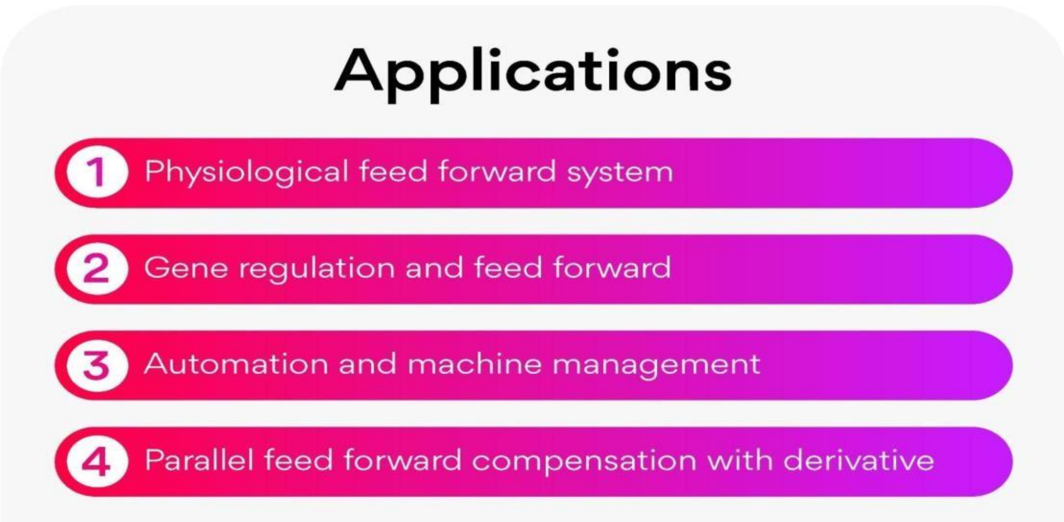
In the output layer, output units are those units that provide the desired output or prediction, thereby fulfilling the task that the neural network needs to complete.

There is a close relationship between the choice of output units and the cost function. Any unit that can serve as a hidden unit can also serve as an output unit ina neural network.

AdvantagesoffeedforwardNeuralNetworks

- Machinelearningcanbeboostedwithfeedforwardneuralnetworks'simplified architecture.
- Multi-networkinthefeedforwardnetworksoperateindependently,witha moderated intermediary.
- Complextasksneedseveralneuronsinthenetwork.
- Neural networks can handle and process nonlinear data easily comparedto perceptrons and sigmoid neurons, which are otherwise complex.
- A neural network deals with the complicated problem of decision boundaries.
- Depending on the data, the neural network architecture can vary. For example, convolutional neural networks (CNNs) perform exceptionally well in image processing, whereas Recurrent Neural Networks(RNNs) perform well in text and voice processing.
- Neural networks need Graphics Processing Units (GPUs) to handle large datasets for massive computational and hardware performance. Several GPUs get used widely in the market, including Kaggle Notebooks and Google Collab Notebooks.

Applicationsoffeedforwardneuralnetworks:



Therearemanyapplicationsfortheseneuralnetworks.Thefollowingareafewof them.

A) Physiologicalfeedforwardsystem

Itispossibletoidentifyfeedforwardmanagementinthissituationbecausethecentral involuntary regulates the heartbeat before exercise.

B) Generegulationandfeedforward

Detectingnon-temporarychangestotheatmosphereisafunctionofthismotifasafeed forward system. You can find the majority of this pattern in the illustrious networks.

C) Automationandmachinemanagement

Automationcontrolusingfeedforwardisoneofthedisciplinesinautomation.

D) Parallelfeedforwardcompensationwithderivative

An open-loop transfer converts non-minimum part systems into minimum part systems using this technique.

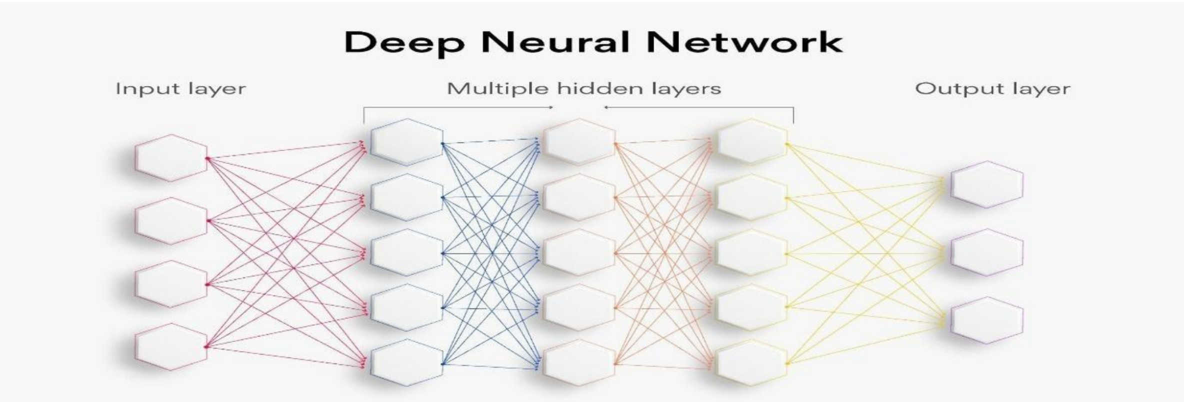
Understandingthemathbehindneuralnetworks

Typical deep learning algorithms are neural networks (NNs). As a result of their unique structure, their popularity results from their 'deep' understanding of data.

Furthermore, NNs are flexible in terms of complexity and structure. Despite all the advanced stuff, they can't work without the basic elements: they may work better with the advanced stuff, but the underlying structure remains the same.

DeepFeed-forwardnetworks:

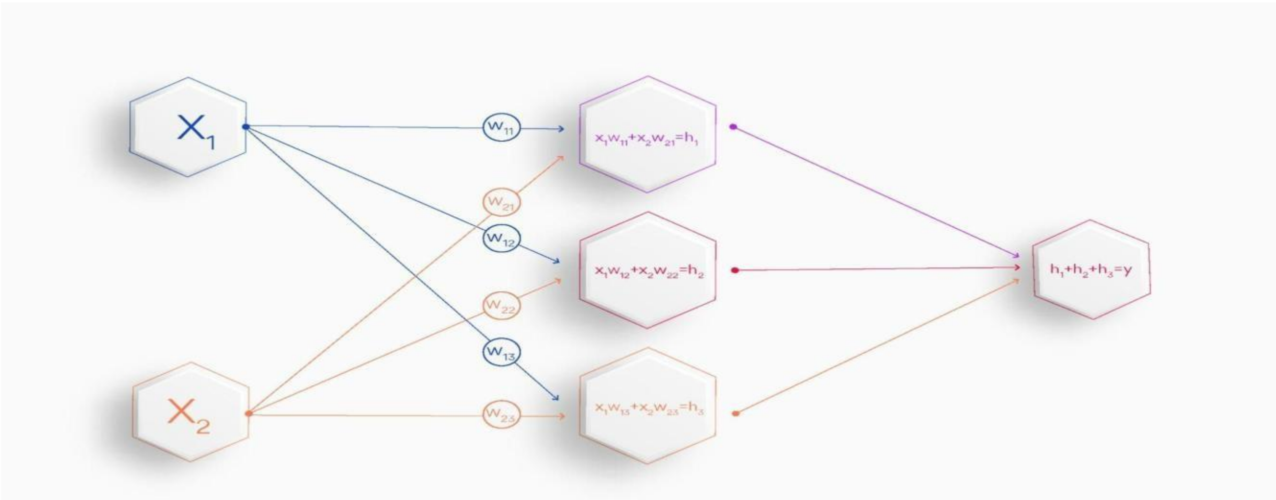
NNsget constructed similarlyto ourbiologicalneurons, and theyresemble the following:



Neurons are hexagons in this image. In neural networks, neurons getarranged into layers: input is the first layer, and output is the last with the hiddenlayer in the middle.

NN consists of two main elements that compute mathematical operations. Neurons calculate weighted sumsusinginput dataandsynaptic weights sinceneural networks are just mathematical computations based on synaptic links.

Thefollowingisasimplifiedvisualization:



Ina matrixformat,itlooks as follows:

$$[x_1 \quad x_2]\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} x_1w_{11}+x_2w_{21} \\ x_1w_{12}+x_2w_{22} \\ x_1w_{13}+x_2w_{23} \end{bmatrix}' = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}'$$

Inthe third step,avectorofonesgetsmultipliedbytheoutput of ourhidden layer:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}' \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = h_1 + h_2 + h_3 = y$$

Using the output value, we can calculate the result. Understanding these fundamental concepts will make building NN much easier, and you will be amazed at how quickly you can do it. Every layer's output becomes the following layer's input.

Thearchitectureofthenetwork:

In a network, the architecture refers to the number of hidden layers and unitsin each layer that make up the network.A feed forward network based on the Universal Approximation Theorem must have a "squashing" activation function at least on one hidden layer.

The network can approximate any Borel measurable function within a finite-dimensional space with at least some amount of non-zero error when there are enough hidden units. It simply states that we can always represent any functionusing the multi-layer perceptron (MLP), regardless of what function we try to learn.

Thus, we now know there will always be an MLP to solve our problem, but there is no specific method for finding it. It is impossible to say whether it will be possible to solve the given problem if we use N layers with M hidden units.

Research is still ongoing, and for now, the only way to determine this configuration is by experimenting with it. While it is challenging to find theappropriate architecture, we need to try many configurations before finding the one that can represent the target function.

There are two possible explanationsfor this.Firstly, the optimization algorithm may not find the correct parameters, and secondly, the training algorithms may use the wrong function because of overfitting.

Whatisbackpropagationinfeedforwardneuralnetwork?

Backpropagation is a technique based on gradient descent. Each stage of a gradient descent process involves iteratively moving a function in the opposite direction of its gradient (the slope).

The goal is to reduce the cost function given the training data while learning a neural network. Network weights and biases of all neurons in each layer determine the cost function. Backpropagation gets used to calculate the gradient of the cost function iteratively. And then update weights and biases in the opposite direction to reduce the gradient.

We must define the error of the backpropagation formula to specify i^{th} neuron in the i^{th} layer of a network for the j-th training. Example as follows (in which $z_i^{[l](j)}$ represents the weighted input to the neuron, and L represents the loss.)

$$\delta_i^{[l](j)} = \frac{\partial \mathcal{L}(\hat{y}^{(j)}, y^{(j)})}{\partial z_i^{[l](j)}}$$

In backpropagationformulas,theerrorisdefinedasabove:

Below is the full derivation of the formulas. For each formula below, L stands for the output layer, g for the activation function, ∇ the gradient, W[l]T layer l weights transposed.

A proportional activation of neuron i at layer l based on b_{li} bias from layer i to layer i, w[k] weight from layer l to layer l-1, and a^{k-1}activation of neuron k at layer l-1 for training example j.

$$\delta^{[L](j)} = \nabla_{\hat{y}^{(j)}} \mathcal{L} \odot (g^{[L]})' (z^{[L](j)}) = \hat{y}^{(j)} - y^{(j)}$$

$$\delta^{[l](j)} = W^{[l+1]^T} \delta^{[l+1](j)} \odot (g^{[l]})' (z^{[l](j)})$$

$$\frac{\partial L}{\partial b_i^{[l]}} = \delta_i^{[l](j)}$$

$$\frac{\partial L}{\partial w_{ik}^{[l]}} = \delta_i^{[l](j)} a_k^{[l-1](j)}$$

The first equation shows how to calculate the error at the output layer for sample j. Following that, we can use the second equation to calculate the error in the layer just before the output layer.

Based on the error values for the next layer, the second equation cancalculate the error in any layer. Because this algorithm calculates errors backward, it is known as backpropagation. For sample j, we calculate the gradient of the loss function by taking the third and fourth equations and dividing them by the biases and weights.

Wecan update biasesand weights by averaging gradients of the lossfunction relative to biases and weights for all samples using the average gradients. The process is known as batch gradient descent. We will have to wait a long time if we have too many samples.

If each sample has a gradient, it is possible to update the biases/weights accordingly. The process is known as stochastic gradient descent. Even though this algorithm is faster than batch gradient descent, it does not yield a good estimate of the gradient calculated using a single sample.

It is possible to update biases and weights based on the average gradients of batches. It gets referred to as mini-batch gradient descent and gets preferred overthe other two.

StochasticGradientDescent(SGD):

Gradient Descent is an iterative optimization process that searches for an objective function'soptimumvalue(Minimum/Maximum).Itisoneofthemostusedmethodsfor changing a model's parameters in order to reduce a cost function in machine learningprojects.

Theprimarygoalofgradientdescentistoidentifythemodelparametersthat provide the maximum accuracy on both training and test datasets. In gradient descent, the gradient is a vector pointing in the general direction of the function's steepest rise at a particularpoint.Thealgorithmmightgraduallydroptowardslowervaluesofthefunction by moving in the opposite direction of the gradient, until reaching the minimum of the function.

TypesofGradientDescent:

Typically,therearethreetypesofGradientDescent:

1. BatchGradientDescent
2. StochasticGradientDescent
3. Mini-batchGradientDescent

1. StochasticGradientDescent(SGD):

Stochastic Gradient Descent(SGD) isa variant of the GradientDescentalgorithm that is used for optimizing machine learning models. It addresses the computational inefficiency of traditional Gradient Descent methods when dealing with large datasets in machine learning projects.

In SGD, instead of using the entire dataset for each iteration, only a single random trainingexample(orasmallbatch)isselectedtocalculatethegradientandupdatethe model parameters. This random selection introduces randomness into the optimization process, hence the term "stochastic" in stochastic Gradient Descent.

TheadvantageofusingSGDisitscomputationalefficiency,especiallywhen dealing with large datasets. By using a single example or a small batch, the computational cost per iteration is significantly reduced compared to traditional Gradient Descent methods that require processing the entire dataset.

StochasticGradientDescentAlgorithm:

- **Initialization**:Randomlyinitializetheparametersofthemodel.
- **SetParameters**:Determinethenumberofiterationsandthelearningrate (alpha) for updating the parameters.
- **Stochastic Gradient Descent Loop**: Repeat the following steps until the model converges or reaches the maximum number of iterations:

a. Shufflethetrainingdatasettointroducerandomness.

b. Iterateovereachtrainingexample(orasmallbatch)intheshuffledorder.

c. Computethegradientofthecostfunctionwithrespecttothemodel parameters using the current training example (or batch).

d. Update the model parameters by taking a step in the direction of the negativegradient, scaled by the learning rate.

e. Evaluate the convergence criteria, such as the difference in the cost function between iterations of the gradient.

- **ReturnOptimizedParameters**:Oncetheconvergencecriteriaaremet orthemaximumnumberofiterationsisreached,returntheoptimizedmodel parameters.

In SGD, since only one sample from the dataset is chosen at random for eachiteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minimum and with a significantly shorter training time.

HiddenUnits:

Inneural networks, a hidden layer is located between the input and output of the algorithm,inwhichthefunctionappliesweightstotheinputsanddirectsthemthrough anactivation function as the output. In short, the hidden layers perform nonlinear transformations of the inputs entered into the network. Hidden layers vary depending on the function of the neural network, and similarly, the layers may vary depending on their associated weights.

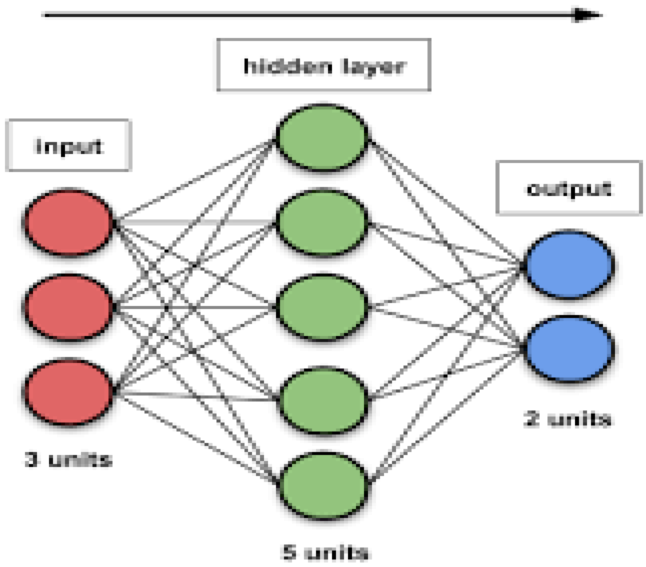
HowdoesaHiddenLayerwork?

Hidden layers, simply put, are layers of mathematical functions each designed to produce an output specific to an intended result. For example, some forms of hidden layers are known as squashing functions. These functions are particularly useful when the intended output of the algorithm is aprobabilitybecause they take an input and produce an output value between 0 and 1, the range for defining probability.

Hidden layers allow for the function of a neural network to be broken down into specific transformations of the data. Each hidden layer function is specialized to produce a defined output. For example, a hidden layer functions that are used to identify human eyes and ears may be used in conjunction by subsequent layers to identify faces in images. While the functions to identify eyes alone are not enough to independently recognize objects, they can function jointly within a neural network.

HiddenLayersandMachine Learning:

Hidden layers are very common in neural networks, however their use andarchitecture often vary from case to case. As referenced above, hidden layers can beseparated by their functional characteristics. For example, in a CNN used for object recognition, a hidden layer that is used to identify wheels cannot solely identify a car, however when placed in conjunction with additional layers used to identify windows, a large metallic body, and headlights, the neural network can then make predictions and identify possible cars within visual data.



ChoosingHidden Layers

1. Wellifthedataislinearlyseparablethen youdon'tneedanyhidden layers at all.

2. If data is less complex and is having fewer dimensions or featuresthen neural networks with 1 to 2 hidden layers would work.

3. Ifdataishavinglargedimensionsorfeaturesthentogetan optimum solution, 3 to 5 hidden layers can be used.

It should be kept in mind that increasing hidden layers would also increase the complexity of the model and choosing hidden layers such as 8, 9, or in two digits may sometimes lead to overfitting.

ChoosingNodesinHidden Layers

Once hidden layers have been decided the next task is to choose the number of nodes in each hidden layer.

1. The number of hidden neurons should be between the size of theinput layer and the output layer.

2. Themostappropriatenumberofhiddenneuronsis

Sqrt(inputlayernodes*outputlayernodes)

3. The number of hidden neurons should keep on decreasing in subsequent layers to get more and more close to pattern and feature extraction and to identify the target class.

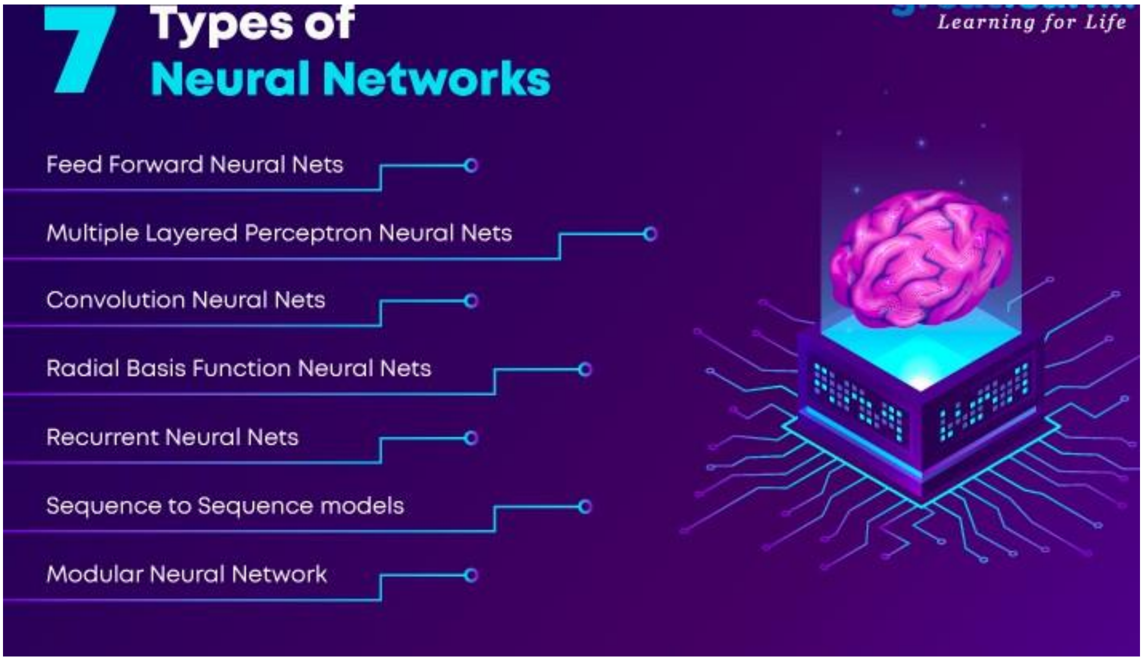
The above algorithms are only a general use case and they can be moulded according to use case.Sometimes the number of nodes in hidden layers can increase also in subsequent layers and the number of hidden layers can also be more than the ideal case.

This whole depends upon the use case and problem statement that we are dealing with.

ArchitectureDesign:

Typesofneuralnetworksmodelsarelistedbelow:

- Perceptron
- FeedForwardNeural Network
- MultilayerPerceptron
- ConvolutionalNeuralNetwork
- RadialBasisFunctionalNeuralNetwork
- RecurrentNeuralNetwork
- LSTM– LongShort-Term Memory
- SequencetoSequenceModels
- ModularNeural Network

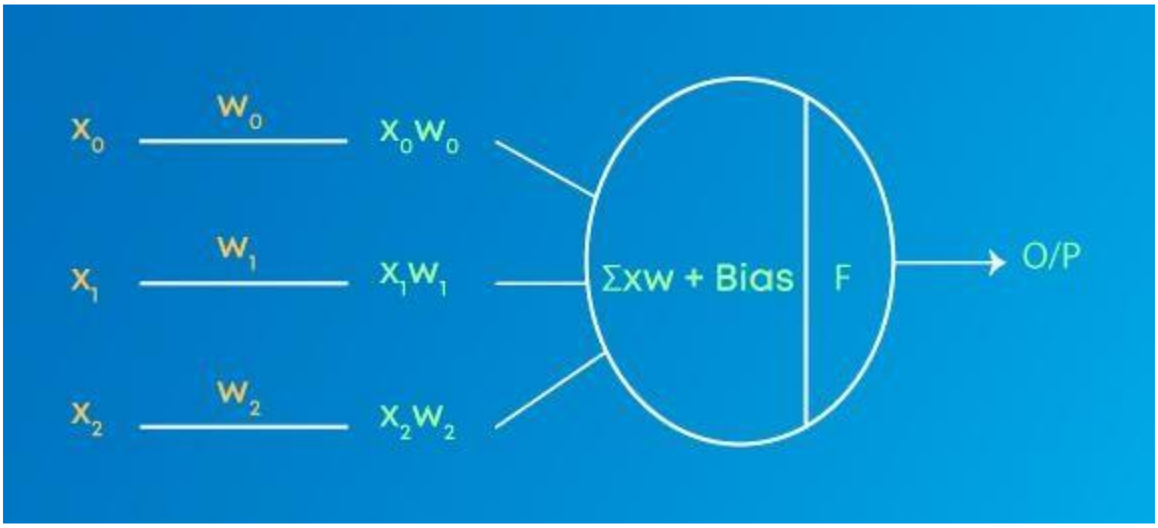


AnIntroductiontoArtificialNeuralNetwork

Neuralnetworksrepresent deeplearningusingartificialintelligence.Certain application scenarios are too heavy or out of scope for traditional machine learningalgorithms to handle. As they are commonly known, Neural Network pitches in such scenarios and fills the gap. Also, enroll in theneural networks and deep learningcourse and enhance your skills today.

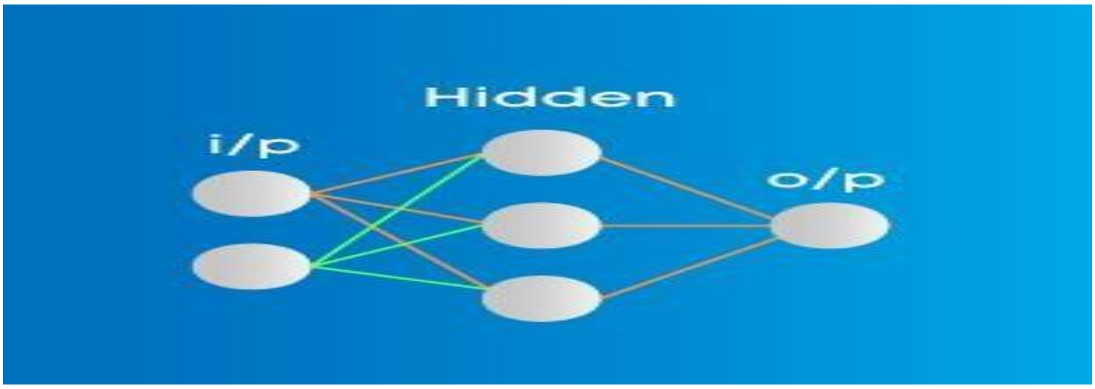
Artificial neural networks are inspired by the biological neurons within the human body which activate under certain circumstances resulting in a related action performed bythe body in response. Artificial neural nets consist of various layers of interconnected artificial neurons powered by activation functions that help in switching them ON/OFF. Like traditionalmachine algorithms, here too, there are certain values that neural nets learn in the training phase.

Briefly, each neuron receives a multiplied version of inputs and random weights, which is then added with a static bias value (unique to each neuron layer); this is then passed to an appropriate activation function which decides the final value to be given out of the neuron. There are various activation functions available as per the nature of input values. Once the output is generated from the final neural net layer, loss function (input vs output) is calculated,andbackpropagationisperformedwheretheweightsareadjustedtomaketheloss minimum. Finding optimal values of weights is what the overall operation focuses around. Please refer to the following for better understanding.



Weightsare numeric values that are multiplied by inputs. In backpropagation, they are modified to reduce the loss. In simple words, weights are machine learned values fromNeural Networks. They self-adjust depending on the difference between predicted outputs vs training inputs.

ActivationFunctionisamathematicalformulathathelpstheneurontoswitchON/OFF.



- **Inputlayer** representsdimensionsoftheinputvector.
- **Hidden layer** represents the intermediary nodes that divide the input space into regions with (soft) boundaries. It takes in a set of weighted input and produces output through an activation function.
- **Outputlayer** representstheoutputoftheneural network.

Backpropagation:

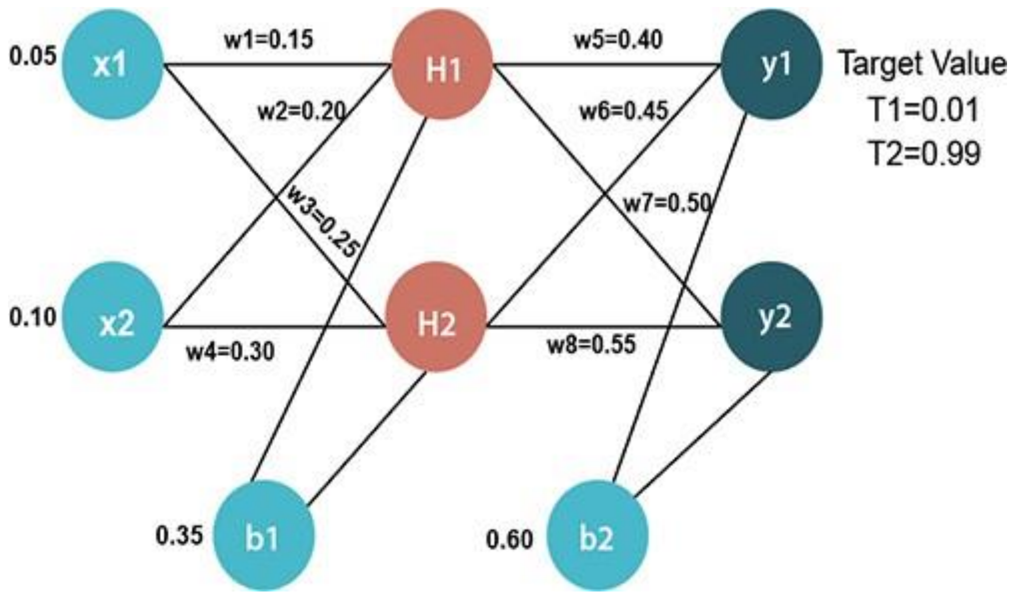
BackpropagationProcessinDeepNeural Network:

Backpropagationis one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model,weusegradientdescenttooptimizetheparameter.Similarlyherewealsouse gradient descent algorithm using Backpropagation.

For a single training example, **Backpropagation**algorithm calculates the gradient of the**error function**. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates theupdated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation.

Now, how error function is used in Backpropagation and howBackpropagation works? Let start with an example and do it mathematically to understand how exactly updates the weight using Backpropagation.



Inputvalues

X1=0.05
X2=0.10

Initialweight

W1=0.1 W5=0.40
W2=0.20 W6=0.45
W3=0.25 W7=0.50
W4=0.30 W8=0.55

BiasValues

b1=0.35 b2=0.60

TargetValues

T1=0.01
T2=0.99

Now,wefirstcalculatethevaluesofH1andH2byaforward pass.

ForwardPass

TofindthevalueofH1wefirstmultiplytheinputvaluefromtheweightsas

$H1 = x1 \times w_1 + x2 \times w_2 + b1$

$H1 = 0.05 \times 0.15 + 0.10 \times 0.20 + 0.3$

H1=0.3775

TocalculatethefinalresultofH1,weperformedthesigmoid functionas

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{H1}}}$$

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{0.3775}}}$$

$$\mathbf{H1_{final} = 0.593269992}$$

WewillcalculatethevalueofH2in thesamewayas H1

$H2 = x1 \times w_3 + x2 \times w_4 + b1$

$H2 = 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35$

H2=0.3925

TocalculatethefinalresultofH1,weperformedthesigmoid functionas

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{H2}}}$$

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{0.3925}}}$$

$$\mathbf{H2_{final} = 0.596884378}$$

Now, we calculate thevalues of y1 and y2 inthe same way as we calculate the H1 and H2. To find the value of y1, we first multiply the input value i.e., the outcome of H1 and H2 from the weights as

$y1 = H1 \times w_5 + H2 \times w_6 + b2$

$y1 = 0.593269992 \times 0.40 + 0.596884378 \times 0.45 + 0.60$

$\mathbf{y1 = 1.10590597}$

Tocalculatethefinalresultofy1weperformedthesigmoidfunctionas

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{y1}}}$$

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{1.10590597}}}$$

$$\mathbf{y1_{final} = 0.75136507}$$

Wewill calculatethevalueofy2 in thesame wayas y1

$y2 = H1 \times w_7 + H2 \times w_8 + b2$

$y2 = 0.593269992 \times 0.50 + 0.596884378 \times 0.55 + 0.60$

$\mathbf{y2 = 1.2249214}$

TocalculatethefinalresultofH1,weperformedthesigmoid functionas

$$y2_{final} = \frac{1}{1 + \frac{1}{e^{y2}}}$$

$$y2_{final} = \frac{1}{1 + \frac{1}{e^{1.2249214}}}$$

$$\mathbf{y2_{final} = 0.772928465}$$

Our target values are 0.01 and 0.99. Our y1 and y2 value is not matched with our target values T1 and T2. Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs. The total error is calculated as

$$E_{total} = \sum \frac{1}{2}(\text{target} - \text{output})^2$$

So,thetotalerror is

$$= \frac{1}{2}(t1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2$$

$$= 0.274811084 + 0.0235600257$$

$$E_{total} = \mathbf{0.29837111}$$

Now,wewillbackpropagatethiserror toupdatetheweightsusingabackward pass.

Backwardpassattheoutputlayer

To update the weight, we calculate the error correspond to each weight with the help of a total error. The error on weight w is calculated by differentiating total error with respect to w.

$$\text{Error}_w = \frac{\partial E_{total}}{\partial w}$$

Weperformbackwardprocesssofirstconsiderthelastweightw5as

$$\text{Error}_{w5} = \frac{\partial E_{total}}{\partial w5} \dots\dots\dots..(1)$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2 \dots\dots\dots..(2)$$

From equation two, it is clear that we cannot partially differentiate it with respect to w5 because there is no any w_5. We split equation one into multiple terms so that we can easily differentiate it with respect to w5 as

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1\_final}{\partial y1} \times \frac{\partial y1}{\partial w5} \dots\dots\dots\dots (3)$$

Now,wecalculateeachtermonebyonetodifferentiateE_{total}withrespectto $w_5$as

$$\frac{\partial E_{total}}{\partial y1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial y1_{final}}$$

$$= 2 \times \frac{1}{2} \times (T1 - y1_{final})^{2-1} \times (-1) + 0$$

$$= -(T1 - y1_{final})$$

$$= -(0.01 - 0.75136507)$$

$$\mathbf{\frac{\partial E_{total}}{\partial y1_{final}} = 0.74136507 \ldots \ldots . (4)}$$

$$y1_{final} = \frac{1}{1 + e^{-y1}} \ldots \ldots \ldots \ldots \ldots \ldots \ (5)$$

$$\frac{\partial y1_{final}}{\partial y1} = \frac{\partial(\frac{1}{1 + e^{-y1}})}{\partial y1}$$

$$= \frac{e^{-y1}}{(1 + e^{-y1})^2}$$

$$= e^{-y1} \times (y1_{final})^2 \ldots \ldots \ldots \ldots (6)$$

$$y1_{final} = \frac{1}{1 + e^{-y1}}$$

$$e^{-y1} = \frac{1 - y1_{final}}{y1_{final}} \ldots \ldots \ldots \ldots (7)$$

Puttingthevalueofe^{-y}in equation(5)

$$= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2$$

$$= y1_{final} \times (1 - y1_{final})$$

$$= 0.75136507 \times (1 - 0.75136507)$$

$$\mathbf{\frac{\partial y1_{final}}{\partial y1} = 0.186815602 \ldots \ldots . (8)}$$

$$y1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \ldots \ldots \ldots \ldots \ldots \ldots \ (9)$$

$$\frac{\partial y1}{\partial w5} = \frac{\partial(H1_{final} \times w5 + H2_{final} \times w6 + b2)}{\partial w5}$$

$$= H1_{final}$$

$$\mathbf{\frac{\partial y1}{\partial w5} = 0.596884378 \ldots \ldots (10)}$$

So, we put the values of $\frac{\partial E_{total}}{\partial y1_{final}}, \frac{\partial y1_{final}}{\partial y1}$, and $\frac{\partial y1}{\partial w5}$ in equation no (3) to find the final result.

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1} \times \frac{\partial y1}{\partial w5}$$

$$= 0.74136507 \times 0.186815602 \times 0.593269992$$

$$\mathbf{Error_{w5} = \frac{\partial E_{total}}{\partial w5} = 0.0821670407 \ldots\ldots\ldots (11)}$$

Now,wewillcalculatetheupdatedweightw5$_{new}$withthehelpofthefollowing formula

$$w5_{new} = w5 - \eta \times \frac{\partial E_{total}}{\partial w5} \quad \text{Here}, \eta = \text{learning rate} = 0.5$$

$$= 0.4 - 0.5 \times 0.0821670407$$

$$\mathbf{w5_{new} = 0.35891648 \ldots\ldots\ldots (12)}$$

In the same way, we calculate w6$_{new}$, w7$_{new}$, and w8$_{new}$and this will give us the following values

w5$_{new}$=0.35891648
w6$_{new}$=408666186
w7$_{new}$=0.511301270
w8$_{new}$=0.561370121

BackwardpassatHiddenlayer

Now, we will backpropagate to our hidden layer and update the weight w1, w2, w3, and w4 as we have done with w5, w6, w7, and w8 weights. We will calculate the error at w1 as

$$\text{Error}_{w1} = \frac{\partial E_{total}}{\partial w1}$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

From equation (2), it is clear that we cannot partially differentiate it with respect to w1 because there is no any w1. We split equation (1) into multiple termsso that we can easily differentiate it with respect to w1 as

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1\_final}{\partial H1} \times \frac{\partial H1}{\partial w1} \ldots \ldots \ldots \ldots \ (13)$$

Now,wecalculateeachtermonebyonetodifferentiateE_{total}withrespectto w1as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial H1} \ldots \ldots \ldots \ldots (14)$$

Weagain splitthisbecausethereisnoanyH1^{final}termin E^{toatal}as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}} \ldots \ldots \ldots (15)$$

$\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ willagainsplitbecauseinE1andE2thereisnoH1term. Splittingisdoneas

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}} \ldots \ldots \ldots (16)$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final} \ldots \ldots \ldots (17)$$

Weagain Splitboth$\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$becausethereisno anyy1andy2termin E1andE2. We split it as

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1} \ldots \ldots \ldots (18)$$

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2} \ldots \ldots \ldots (19)$$

Now,wefindthevalueof$\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$byputtingvaluesinequation(18)and(19)as From equation (18)

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{\text{fianl}}} \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

$$= \frac{\partial(\frac{1}{2}(T1 - y1_{\text{final}})^2)}{\partial y1_{\text{final}}} \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

$$= 2 \times \frac{1}{2}(T1 - y1_{\text{final}}) \times (-1) \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

Fromequation(8)

$$= 2 \times \frac{1}{2}(0.01 - 0.75136507) \times (-1) \times \ 0.186815602$$

$$\mathbf{\frac{\partial E_1}{\partial y1} = 0.138498562} \ldots\ldots\ldots (20)$$

Fromequation(19)

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{\text{fianl}}} \times \frac{\partial y2_{\text{final}}}{\partial y2}$$

$$= \frac{\partial(\frac{1}{2}(T2 - y2_{\text{final}})^2)}{\partial y2_{\text{final}}} \times \frac{\partial y2_{\text{final}}}{\partial y2}$$

$$= 2 \times \frac{1}{2}(T2 - y2_{\text{final}}) \times (-1) \times \frac{\partial y2_{\text{final}}}{\partial y2} \ldots\ldots\ldots (21)$$

$$y2_{\text{final}} = \frac{1}{1 + e^{-y2}} \ldots\ldots\ldots\ldots\ldots\ldots \ (22)$$

$$\frac{\partial y2_{\text{fianl}}}{\partial y2} = \frac{\partial(\frac{1}{1 + e^{-y2}})}{\partial y2}$$

$$= \frac{e^{-y2}}{(1 + e^{-y2})^2}$$

$$= e^{-y2} \times (y2_{\text{final}})^2 \ldots\ldots\ldots\ldots\ldots (23)$$

$$y2_{\text{final}} = \frac{1}{1 + e^{-y2}}$$

$$e^{-y2} = \frac{1 - y2_{\text{final}}}{y2_{\text{final}}} \ldots\ldots\ldots\ldots\ldots (24)$$

Puttingthevalueofe^{-y2}in equation(23)

$$= \frac{1 - y2_{final}}{y2_{final}} \times (y2_{final})^2$$

$$= y2_{final} \times (1 - y2_{final})$$

$$= 0.772928465 \times (1 - 0.772928465)$$

$$\frac{\partial \mathbf{y2_{fianl}}}{\partial \mathbf{y2}} = \mathbf{0.175510053} \ldots\ldots\ldots (\mathbf{25})$$

Fromequation(21)

$$= 2 \times \frac{1}{2}(0.99 - 0.772928465) \times (-1) \times \ 0.175510053$$

$$\frac{\partial \mathbf{E_1}}{\partial \mathbf{y1}} = -\mathbf{0.0380982366126414} \ldots\ldots\ldots (\mathbf{26})$$

Nowfromequation(16)and(17)

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial (H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial (H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times w5$$

$$= 0.138498562 \times 0.40$$

$$\frac{\partial \mathbf{E_1}}{\partial \mathbf{H1_{final}}} = \mathbf{0.0553994248} \ldots\ldots\ldots (\mathbf{27})$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final}$$

$$= -0.0380982366126414 \times \frac{\partial (H1_{final} \times w_7 + H2_{final} \times w_8 + b2)}{\partial H1_{final}}$$

$$= -0.0380982366126414 \times w7$$

$$= -0.0380982366126414 \times 0.50$$

$$\frac{\partial \mathbf{E_2}}{\partial \mathbf{H1_{final}}} = -\mathbf{0.0190491183063207} \ldots\ldots\ldots (\mathbf{28})$$

Put thevalueof $\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ inequation(15)as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}}$$

$$= 0.0553994248 + (-0.0190491183063207)$$

$$\mathbf{\frac{\partial E_{total}}{\partial H1_{final}} = 0.0364908241736793 \ldots \ldots \ldots (29)}$$

Wehave$\frac{\partial E_{total}}{\partial H1_{final}}$,weneedto figureout $\frac{\partial H1\_final}{\partial H1}$, $\frac{\partial H1}{\partial w1}$as

$$\frac{\partial H1_{final}}{\partial H1} = \frac{\partial(\frac{1}{1+e^{-H1}})}{\partial H1}$$

$$= \frac{e^{-H1}}{(1+e^{-H1})^2}$$

$$e^{-H1} \times (H1_{final})^2 \ldots \ldots \ldots (30)$$

$$H1_{final} = \frac{1}{1+e^{-H1}}$$

$$e^{-H1} = \frac{1 - H1_{final}}{H1_{final}} \ldots \ldots \ldots \ldots \ldots (31)$$

Puttingthevalueofe^{-H1}in equation(30)

$$= \frac{1 - H1_{final}}{H1_{final}} \times (H1_{final})^2$$

$$= H1_{final} \times (1 - H1_{final})$$

$$= 0.593269992 \times (1 - 0.593269992)$$

$$\mathbf{\frac{\partial H1_{final}}{\partial H1} = 0.2413007085923199}$$

We calculate the partial derivative of the total net input to H1 with respect to w1 the same as we did for the output neuron:

$$H1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \ldots \ldots \ldots \ldots \ldots \ldots \ (32)$$

$$\frac{\partial y1}{\partial w1} = \frac{\partial(x1 \times w1 + x2 \times w3 + b1 \times 1)}{\partial w1}$$

$$= x1$$

$$\frac{\partial \mathbf{H1}}{\partial \mathbf{w1}} = \mathbf{0.05} \ldots \ldots \ldots (\mathbf{33})$$

So, we put the values of $\frac{\partial E_{total}}{\partial H1_{final}}, \frac{\partial H1_{final}}{\partial H1}$, and $\frac{\partial H1}{\partial w1}$ in equation (13) to find the final result.

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1_{final}}{\partial H1} \times \frac{\partial H1}{\partial w1}$$

$$= 0.0364908241736793 \times 0.2413007085923199 \times 0.05$$

$$\mathbf{Error_{w1}} = \frac{\partial \mathbf{E_{total}}}{\partial \mathbf{w1}} = \mathbf{0.000438568} \ldots \ldots \ldots (\mathbf{34})$$

Now,wewillcalculatetheupdatedweightw1_{new}withthehelpofthefollowing formula

$$w1_{new} = w1 - \eta \times \frac{\partial E_{total}}{\partial w1} \text{ Here } \eta = \text{learning rate} = 0.5$$

$$= 0.15 - 0.5 \times 0.000438568$$

$$\mathbf{w1_{new}} = \mathbf{0.149780716} \ldots \ldots \ldots (\mathbf{35})$$

Inthesameway, wecalculatew2_{new},w3_{new},andw4andthiswillgiveusthe following values

$w1_{new}$=0.149780716
$w2_{new}$=0.19956143
$w3_{new}$=0.24975114
$w4_{new}$=0.29950229

We have updated all the weights. We found the error 0.298371109 on the network when we fed forward the 0.05 and 0.1 inputs. In the first round of Backpropagation,thetotalerrorisdownto 0.291027924.Afterrepeatingthisprocess 10,000,thetotalerrorisdownto0.0000351085.Atthispoint,theoutputsneurons generate 0.159121960 and 0.984065734 i.e., nearby our target value when we feedforward the 0.05 and 0.1.

Deeplearningframeworksandlibraries:

DeepLearningFrameworks:

Keras, TensorFlow and PyTorch are among the top three frameworks that are preferred by Data Scientists as well as beginners in the field of Deep Learning. This comparison on Keras vs TensorFlow vs PyTorch will provide you with acrisp knowledge about the top Deep Learning Frameworks and help you find out which one is suitable for you. In this blog you will get a complete insight into the above three frameworks in the following sequence:

- IntroductiontoKeras,TensorFlow&PyTorch
- ComparisonFactors
- FinalVerdict

Introduction

Keras

Keras is an open source neural networklibrary written in Python. It is capable ofrunningontopofTensorFlow.Itisdesignedtoenablefastexperimentation with **deep neural networks**.

TensorFlow

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library that is used for **machine learning** applications like neural networks.

PyTorch

PyTorchis an open-source **machine learning**library for Python, based on Torch. It is used for applications such as **natural language processing** and was developed by Facebook's AI research group.

ComparisonFactors

All the three frameworks are related to each other and also have certain basic differences that distinguishes them from one another.

The**parameters**thatdistinguishthem:

- LevelofAPI
- Speed
- Architecture
- Debugging
- Dataset
- Popularity

LevelofAPI

Keras is a **high-level API**capable ofrunning ontop of TensorFlow,CNTK and Theano. It has gained favor for its ease of use and syntactic simplicity,facilitating fast development.

TensorFlow is a framework that provides both **high and low level** APIs. Pytorch, on the other hand, is a **lower-level API** focused on direct work with array expressions.Ithasgainedimmenseinterestinthelastyear,becomingapreferred solutionforacademicresearch,andapplicationsofdeeplearningrequiring optimizing custom expressions.

Speed

The performance is comparatively **slower**in**Keras**whereas TensorFlow and PyTorch provide a similar pace which is fast and suitable for **high performance**.

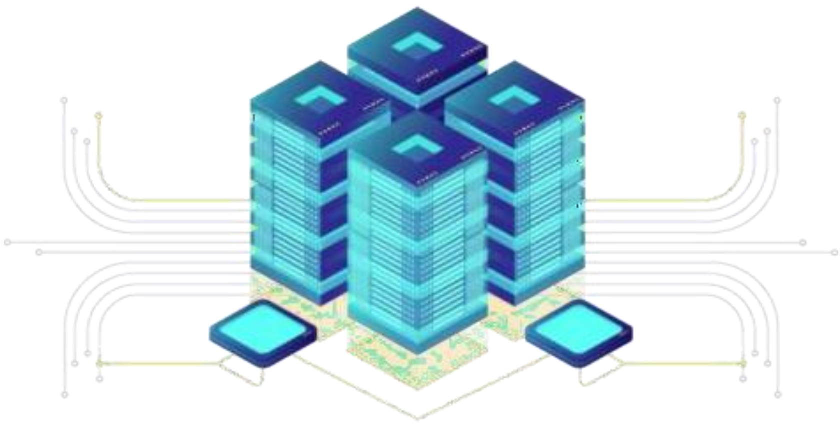
Architecture

Kerashas a **simple**architecture. It is more readable and concise. Tensorflow on the other hand is not very easy to use even though it provides Keras as a framework that makes work easier. PyTorch has a **complex** architecture and the readability is less when compared to Keras.

Debugging

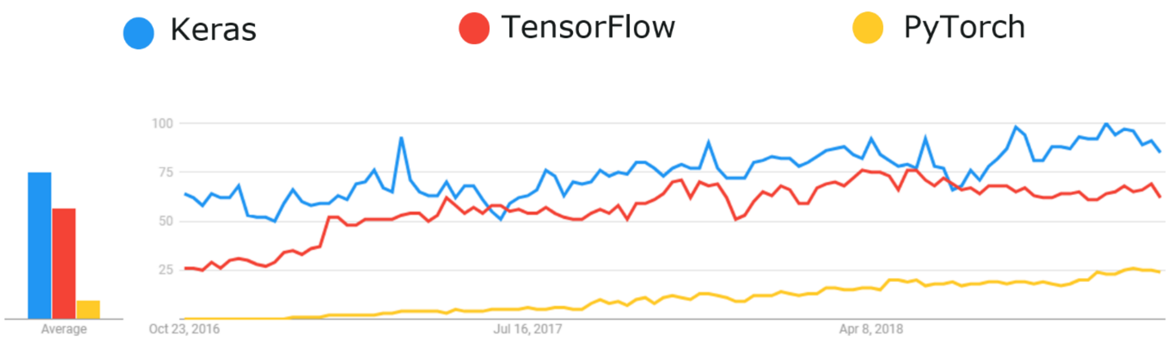
In keras, there is usually very **less frequent**need to debug simple networks. But in case of Tensorflow, it is quite **difficult**to perform debugging. **Pytorch**on the other hand has **better debugging capabilities** as compared to the other two.

Dataset



Keras is usually used for **small datasets**as it is comparatively slower. On the otherhand,TensorFlowandPyTorchareusedfor **highperformance**models and **large datasets** that require fast execution.

Popularity



With the increasing demand in the field of **Data Science,**there has been an enormous growth of **Deep learning technology**in the industry. With this, all the three frameworks havegained quite a lot of popularity. **Keras**tops the list followedbyTensorFlowandPyTorch.Ithasgainedimmensepopularitydueto its **simplicity** when compared to the other two.

These were the parameters that distinguish all the three frameworks but there is no absolute answer to which one is better. The choice ultimately comes down to

- Technicalbackground
- Requirementsand
- Ease ofUse

FinalVerdict

Now coming to the final verdict of Keras vs TensorFlow vs PyTorch let’s have a look at the situations that are most **preferable**for each one of these three deep learning frameworks

Kerasismost suitablefor:

- RapidPrototyping
- SmallDataset
- Multipleback-endsupport

TensorFlowismostsuitable for:

- LargeDataset
- HighPerformance
- Functionality
- ObjectDetection

PyTorchismostsuitable for:

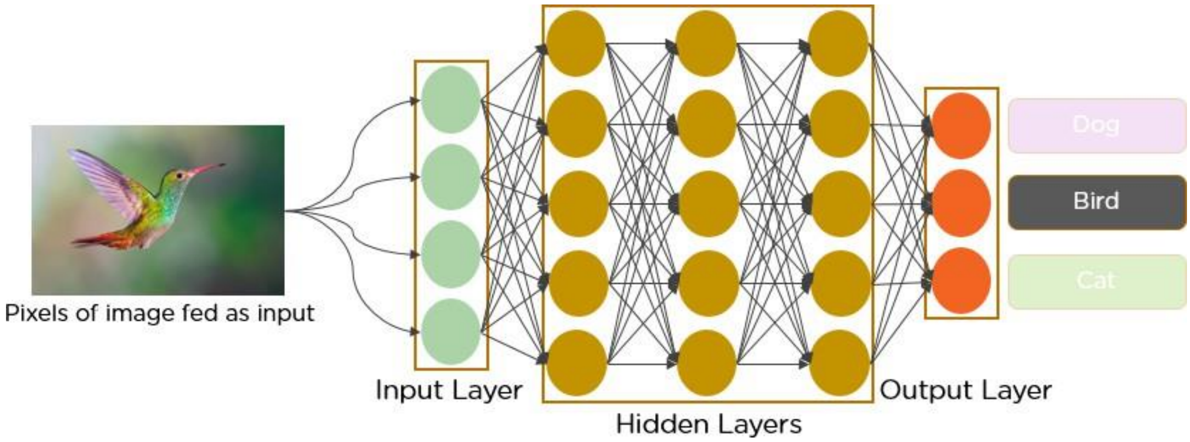
- Flexibility
- ShortTrainingDuration
- Debuggingcapabilities

UNIT-II:

CONVOLUTIONNEURALNETWORK(CNN): IntroductiontoCNNs and their applications in computer vision, CNN basic architecture, Activation functions-sigmoid, tanh, ReLU, Softmax layer, Types of pooling layers, Training of CNN in TensorFlow, various popular CNN architectures:VGG, GoogleNet,ResNetetc, Dropout,Normalization, Data augmentation

IntroductiontoCNNsandtheirapplicationsincomputervision:

Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks (also known as CNN or ConvNet) in deep learning, especially when it comes to Computer Vision applications.



Sincethe1950s,the earlydaysofAI,researchershavestruggledtomake asystemthatcanunderstandvisualdata.Inthefollowingyears,thisfieldcame to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms, and that tooby a large margin.

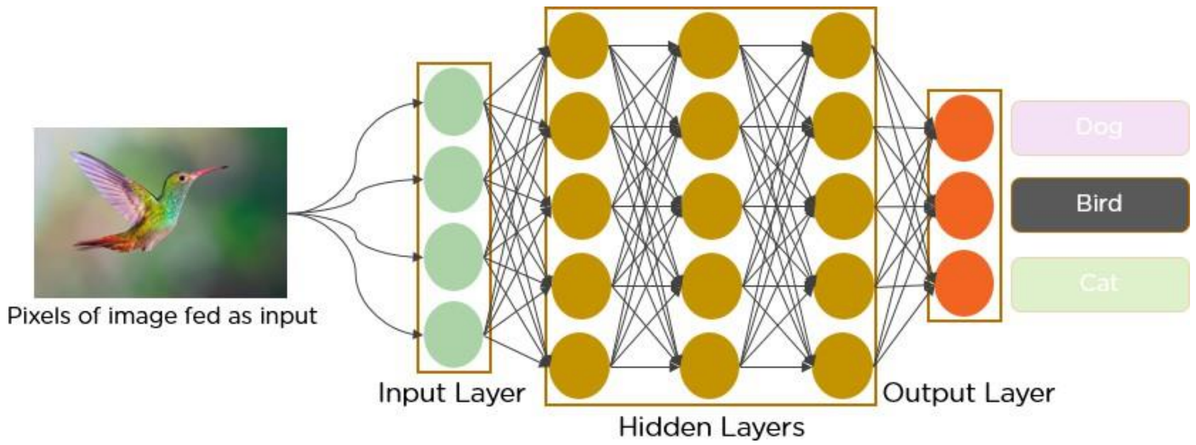
The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contestwith an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision.

BackgroundofCNNs

CNN's were first developed and used around the 1980s. The most that a CNNcoulddoatthattimewasrecognizehandwrittendigits.Itwasmostlyused inthepostalsectorstoreadzipcodes,pincodes,etc.Theimportantthingto remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks (also known as CNN or ConvNet) in deep learning, especially when it comes to Computer Vision applications.



Since the 1950s, the early days of AI, researchers have struggled to make a systemthatcan understand visualdata.In the following years, thisfield came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms, and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest withan amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications and hence a part of any computer vision course online. So let's take a look at the workings of CNNs or CNN algorithm in deep learning.

- BackgroundofCNNs
- WhatIsaCNN?
- Howdoesitwork?
- WhatIsaPoolingLayer?
- Limitationsof CNNs

BackgroundofCNNs

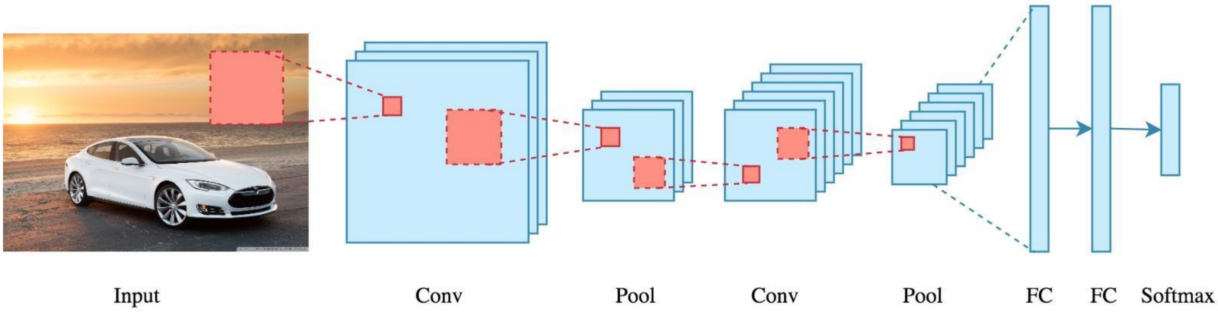
CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In 2012, Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

WhatIsa CNN?

In deep learning, a **Convolutional Neural Network**(**CNN/ConvNet**) is a class of deep neural networks, most commonly applied toanalyze visual imagery.

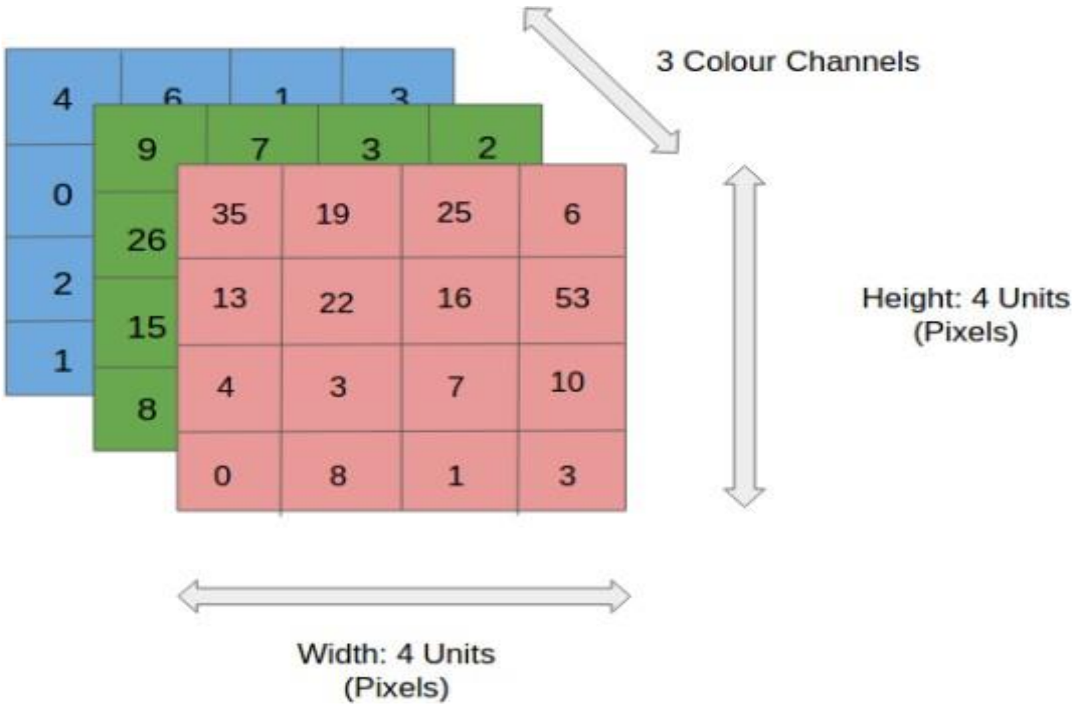
Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functionsthat produces a third function that expresses how the shape of one is modified by the other.



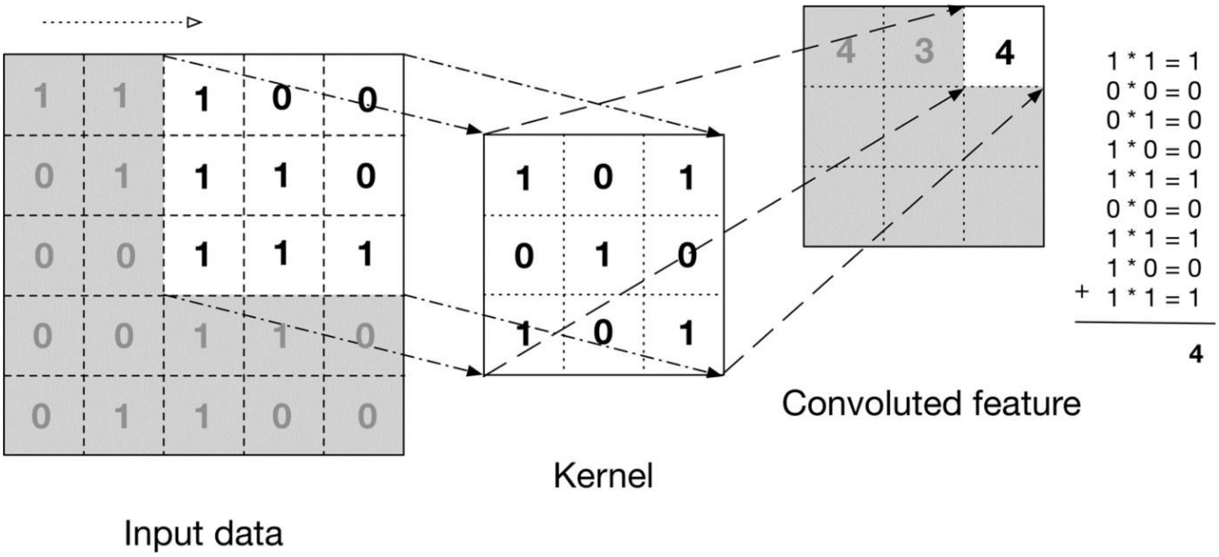
Bottom line is that the ConvNet role to reduce the images into a form thatiseasiertoprocess,withoutlosingfeatures crucialforgoodprediction.

Howdoesitwork?

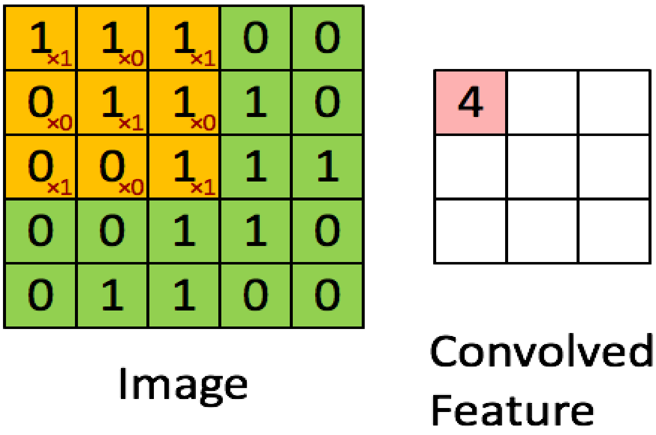
Before we go to the working of CNN's let's cover the basics such as what is an image and how is it represented. An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image isthe same but it has a single plane. Take a look at this image to understand more.



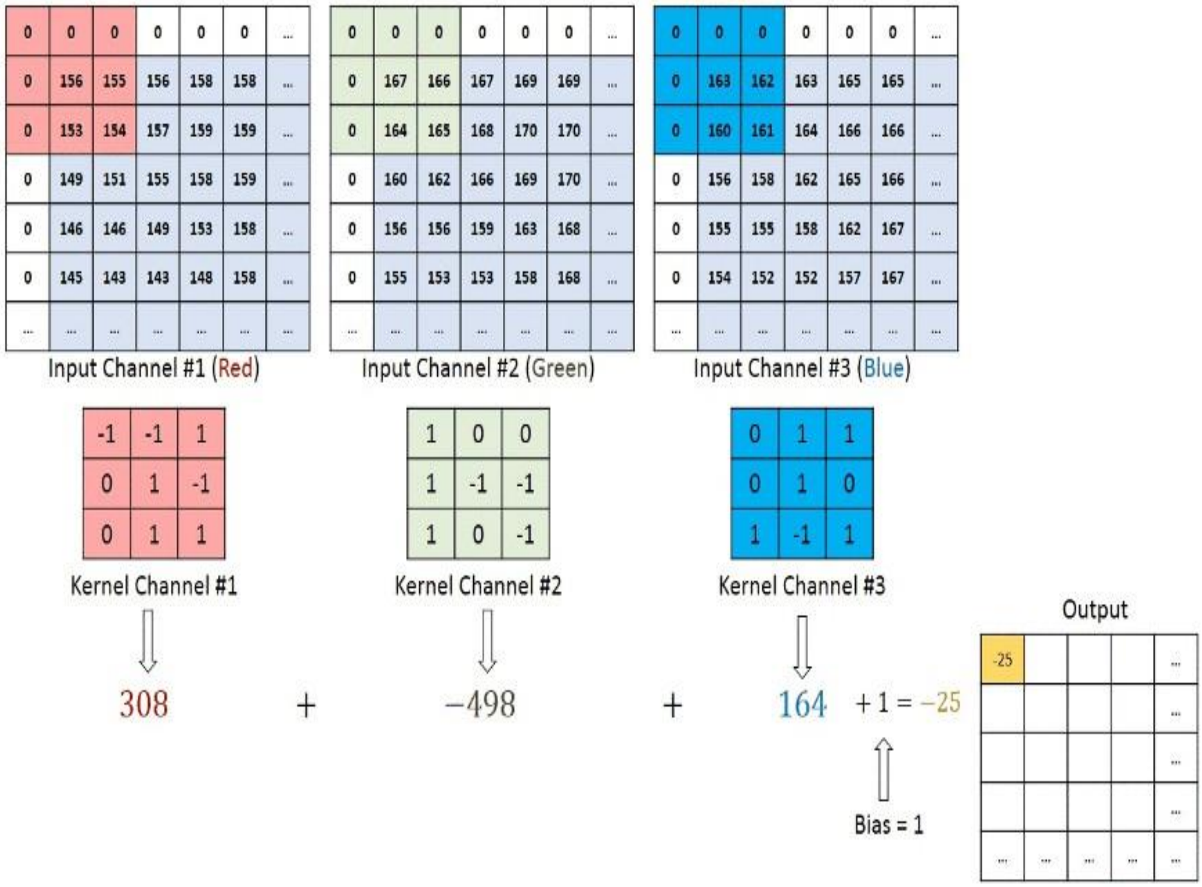
Forsimplicity,considergrayscaleimagestounderstandhowCNNs work.



The above image shows what a convolution is.We take a filter/kernel (3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

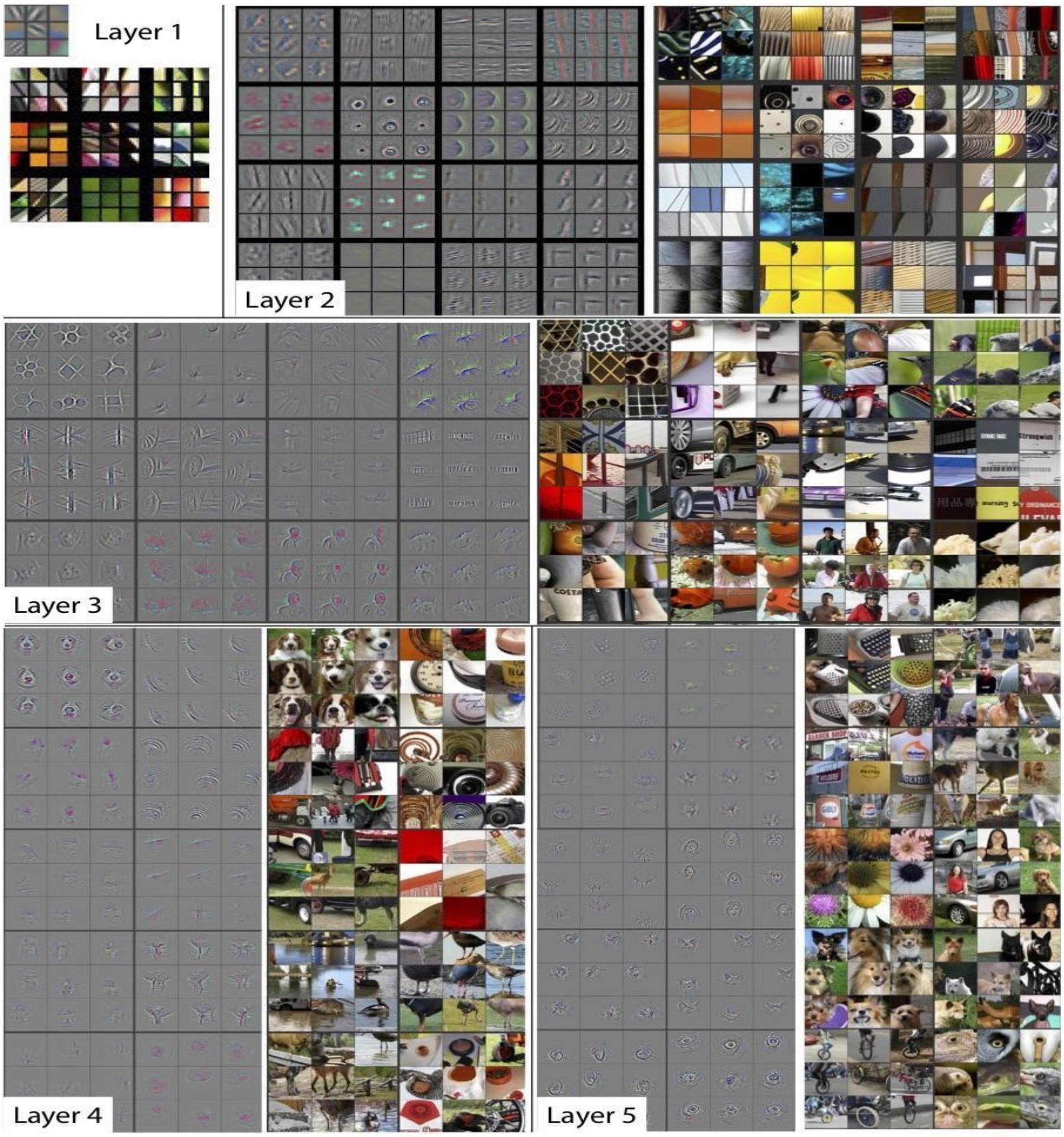


In the case of RGB color, channel take a look at this animation to understand its working.

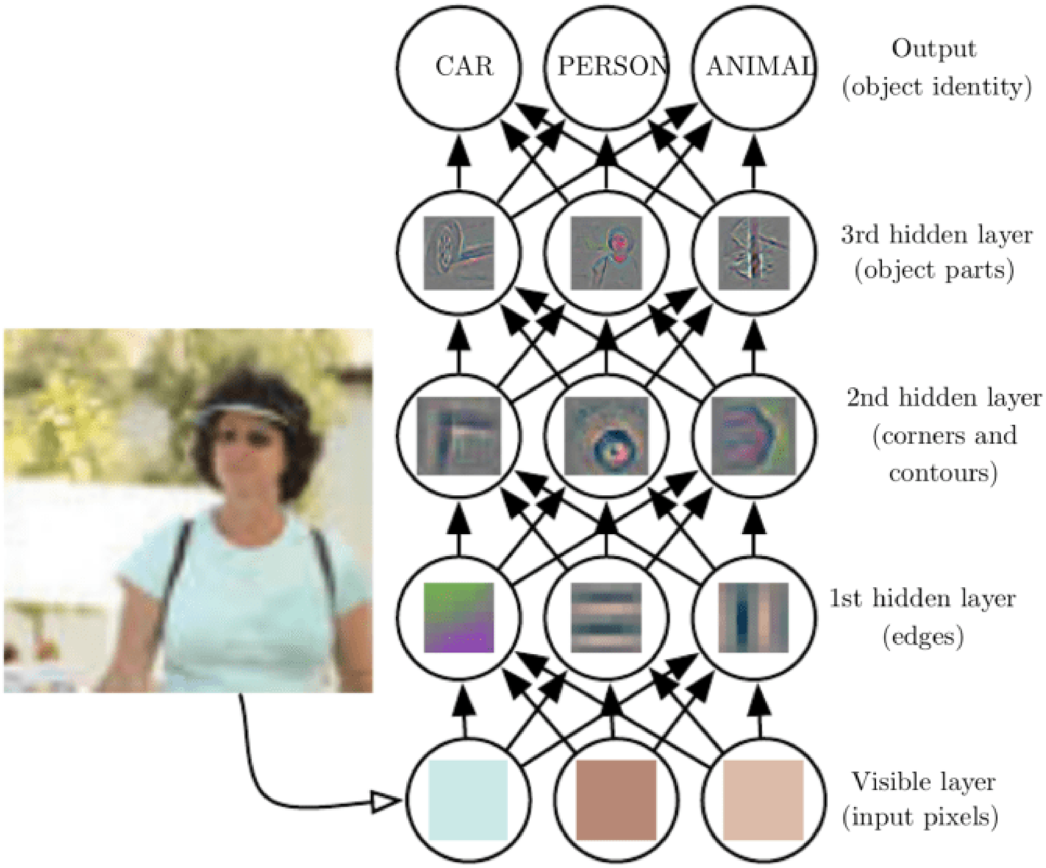


Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sumofmultipleinputsandoutputsanactivationvalue.Whenyouinputan image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

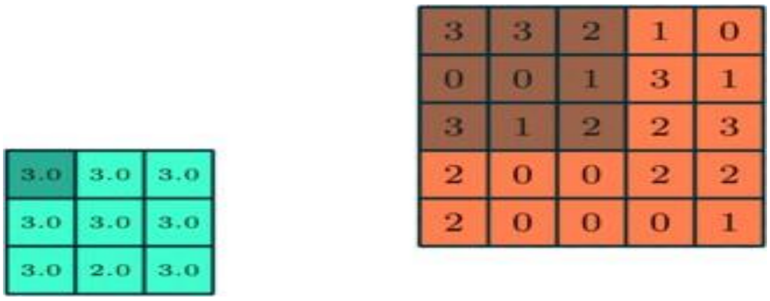


Based on the activation map of the final convolution layer, the classificationlayeroutputsasetofconfidencescores(valuesbetween0 and 1) that specify how likely the image is to belong to a "class." For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains anyof those animals.



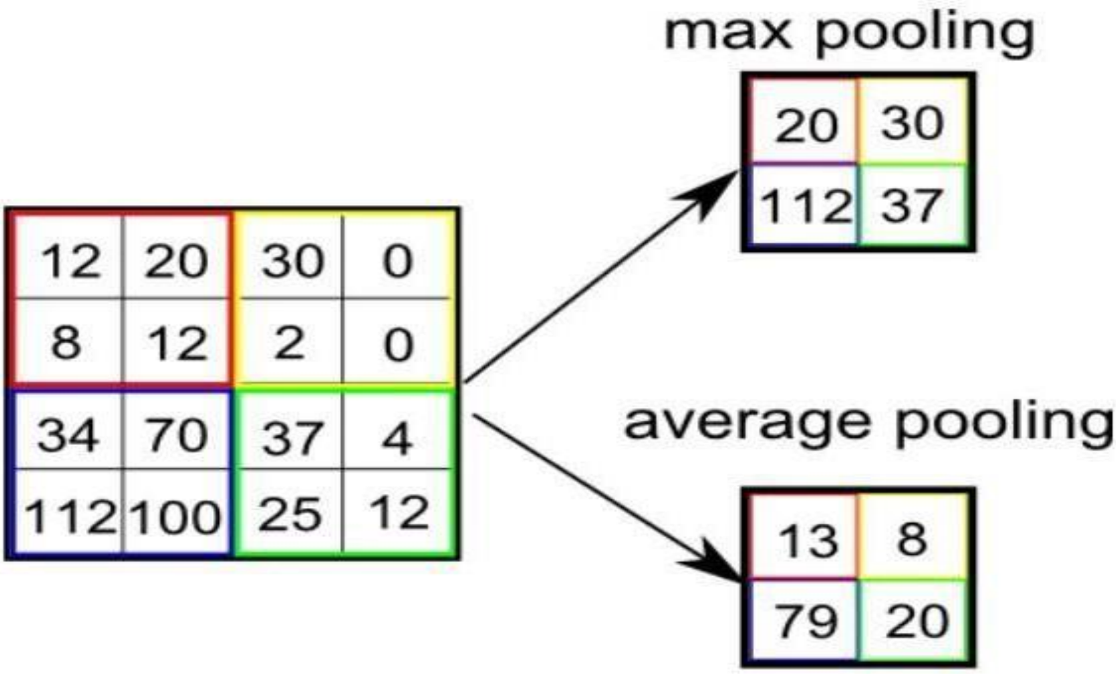
WhatIsaPoolingLayer?

Similar to the Convolutional Layer, the Pooling layer is responsiblefor reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** by reducing the dimensions. There are two types of pooling average pooling and max pooling.



In Max Pooling, the maximum value of a pixel from a portion of the imagecoveredbythekernelisfoundout.MaxPoolingalsoperformsas a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

On the other hand,**Average Pooling**returns the **average of all the values**from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.



BenefitsofUsingCNNsforMachineandDeepLearning

Deep learning is a form of machine learning that requires a neural network with a minimum of three layers. Networks with multiple layers are more accurate than single-layer networks. Deep learning applications often use CNNs or RNNs (recurrent neural networks).

The CNN architecture is especially useful for image recognition and image classification, as well as other computer vision tasks becausetheycan processlarge amounts of data andproducehighlyaccurate predictions. CNNs can learn the features of an object through multiple iterations, eliminating the need for manual feature engineering tasks like feature extraction.

It is possible to retrain a CNN for a new recognition task or build a new model based on an existing network with trained weights. This is known as transfer learning. This enables ML model developers to apply CNNs to different use cases without starting from scratch.

WhatAreConvolutionalNeuralNetworks(CNNs)?

A Convolutional Neural Network (CNN) is a type of deep learning algorithm specificallydesigned for image processing and recognition tasks. Compared to alternative classification models, CNNs require less preprocessing as they can automatically learn hierarchical feature representations from raw inputimages.Theyexcelat assigningimportanceto variousobjectsandfeatureswithintheimagesthrough convolutional layers, which apply filters to detect local patterns.

The connectivity pattern in CNNs is inspired by the visual cortex in the human brain, where neurons respond to specific regions or receptive fields in the visual space. This architecture enables CNNs to effectively capture spatial relationships and patterns in images. By stacking multiple convolutional and pooling layers,CNNscanlearn increasinglycomplex features, leading tohigh accuracyin taskslike image classification, object detection, and segmentation.

ConvolutionalNeuralNetworkArchitectureModel

Convolutional neural networks are known for their superiority over other artificial neural networks, given their ability to process visual, textual, and audio data. The CNN architecture comprises three main layers: convolutional layers, pooling layers, and a fully connected (FC) layer.

There can be multiple convolutional and pooling layers. The more layers in the network, the greaterthecomplexityand(theoretically)theaccuracyofthemachinelearningmodel.Eachadditional layerthatprocessestheinputdataincreasesthemodel'sabilitytorecognizeobjectsandpatternsinthe data.

TheConvolutional Layer

Convolutional layers are the key building block of the network, where most of the computations are carried out. It works by applying a filter to the input data to identify features. This filter, known as a feature detector, checks the image input's receptive fields for a given feature. This operation is referred to as convolution.

The filter is a two-dimensional array of weights that represents part of a 2-dimensional image. A filter is typically a 3×3 matrix, although there are other possible sizes. The filter is applied to a region withintheinput imageandcalculatesadotproductbetweenthe pixels,whichisfedto anoutputarray.The filter then shifts and repeats the process until it has covered the whole image. The final output of all the filter processes is called the feature map.

The CNN typically applies the ReLU (Rectified Linear Unit) transformation to each feature map after every convolution to introduce nonlinearity to the ML model. A convolutional layer is typically followed by a pooling layer. Together, the convolutional and pooling layers make up a convolutionalblock.

Additional convolution blocks will follow the first block, creating a hierarchical structure with later layers learning from the earlier layers. For example, a CNN model might train to detect cars inimages. Cars can be viewed as the sum of their parts, including the wheels, boot, and windscreen. Each feature of a car equates to a low-level pattern identified by the neural network, which then combines these parts to create a high-level pattern.

ThePoolingLayers

A pooling or down sampling layer reduces the dimensionality of the input. Like a convolutional operation, pooling operations use a filter to sweep the whole input image, but it doesn't use weights. The filter instead uses an aggregation function to populate the output array based on the receptive field'svalues.

Therearetwokeytypesof pooling:

- **Averagepooling:**Thefiltercalculatesthereceptivefield'saveragevaluewhenitscanstheinput.

- **Max pooling:**The filter sends the pixel with the maximum value to populate the output array.This approach is more common than average pooling.

Pooling layers are important despite causing some information to be lost, because they help reduce the complexity and increase the efficiency of the CNN. It also reduces the risk of overfitting.

TheFullyConnected(FC)Layer

ThefinallayerofaCNNisafullyconnectedlayer.

The FC layer performs classification tasks using the features that the previous layers and filters extracted. Instead of ReLu functions, the FC layer typically uses a softmax function that classifies inputs more appropriately and produces a probability score between 0 and 1.

BasicArchitectureof CNN:

BasicArchitecture

TherearetwomainpartstoaCNNarchitecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction.
- The network of feature extraction consists of many pairs of convolutional or pooling layers.
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- This CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarizes the existing features contained in an original set of features. There are many CNN layersas shown in the CNN architecture diagram.

ConvolutionLayers

There are three types of layers that make up the CNN which are the convolutionallayers,poolinglayers,andfully-connected(FC)layers.When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropoutlayerandtheactivationfunctionwhicharedefinedbelow.

1. ConvolutionalLayer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation ofconvolutionisperformedbetweentheinputimageandafilterofa particularsizeMxM.Byslidingthefilterovertheinputimage,thedot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fedto other layers to learn several other features of the input image.

TheconvolutionlayerinCNNpassestheresulttothenextlayer onceapplyingtheconvolutionoperationintheinput.Convolutional layersinCNNbenefitalotastheyensurethespatialrelationship between the pixels is intact.

2. Pooling Layer

In most cases, a ConvolutionalLayerisfollowedbya PoolingLayer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connectionsbetweenlayersandindependentlyoperatesoneachfeature map. Depending upon method used, there are several types of Pooling operations. It basically summarises the features generated by a convolution layer.

InMaxPooling,thelargestelementistakenfromfeaturemap. Average Pooling calculates the average of the elements in a predefined sized Imagesection.Thetotalsumoftheelementsinthepredefinedsectionis computedinSumPooling.ThePoolingLayerusuallyservesasabridge between the Convolutional Layer and the FC Layer.

This CNN model generalises the features extracted by the convolution layer, and helps the networks to recognise the features independently. With the help of this, the computations are also reduced in a network.

3. FullyConnectedLayer

TheFullyConnected(FC)layerconsistsoftheweightsandbiases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fedto the FC layer. The flattened vector then undergoes few more FC layerswherethemathematicalfunctionsoperationsusuallytakeplace.Inthis stage, the classification process begins to take place. The reason two layersare connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce the human supervision

4. Dropout

Usually, when all the features are connected to the FC layer, it cancauseoverfittinginthetrainingdataset.Overfittingoccurswhena particularmodelworkssowellonthetrainingdatacausinganegative impact in the model’s performance when used on a newdata.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting inreduced size of the model. On passing a dropout of0.3, 30% ofthe nodes are dropped out randomly from the neural network.

Dropout results in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

5. ActivationFunctions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

Itaddsnon-linearitytothenetwork.Thereareseveralcommonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classificationCNNmodel,sigmoidandsoftmaxfunctionsarepreferredafor a multi-class classification, generally softmax us used. In simple terms, activation functions in a CNN model determine whether a neuron should be activatedornot.Itdecideswhethertheinputtotheworkisimportantor not to predict using mathematical operations.

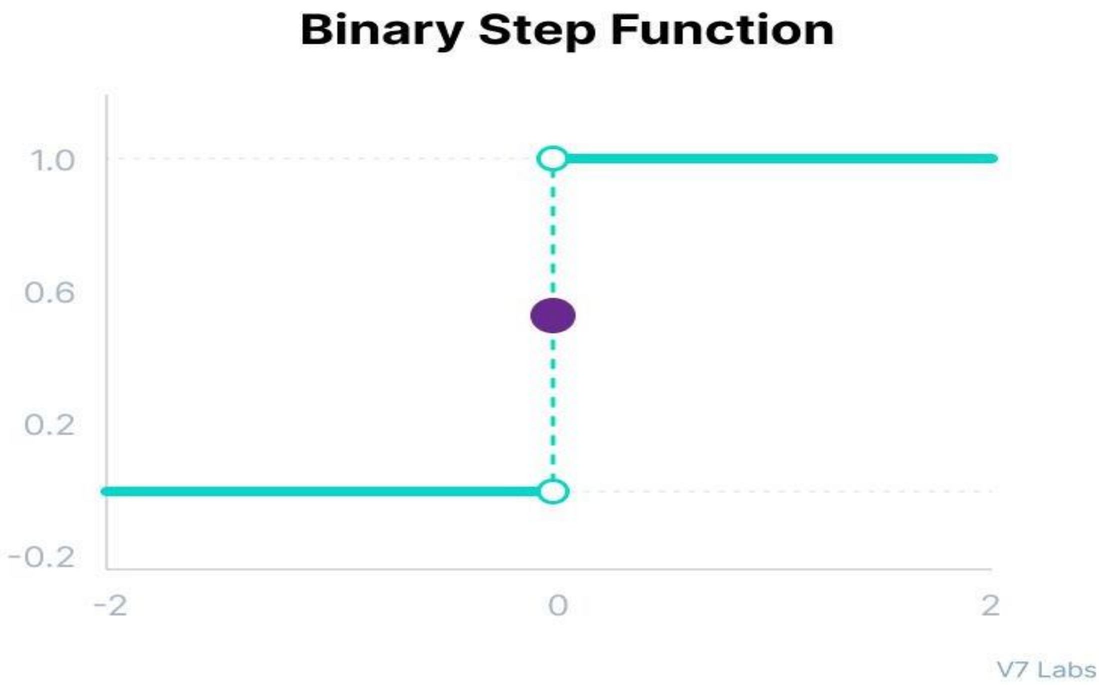
TypesofNeuralNetworks

Activation Functions

Thepopularactivationfunctionsare

a) BinaryStepFunction

Binarystepfunctiondependsonathresholdvaluethatdecideswhether aneuronshouldbeactivatedornot.Theinputfedtotheactivationfunctionis comparedtoacertainthreshold;iftheinputisgreaterthanit,thentheneuronis activated,elseitisdeactivated,meaningthatitsoutputisnotpassedontothe next hidden layer.



Mathematically,itcanberepresentedas:

$$\textit{Binary step}$$

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

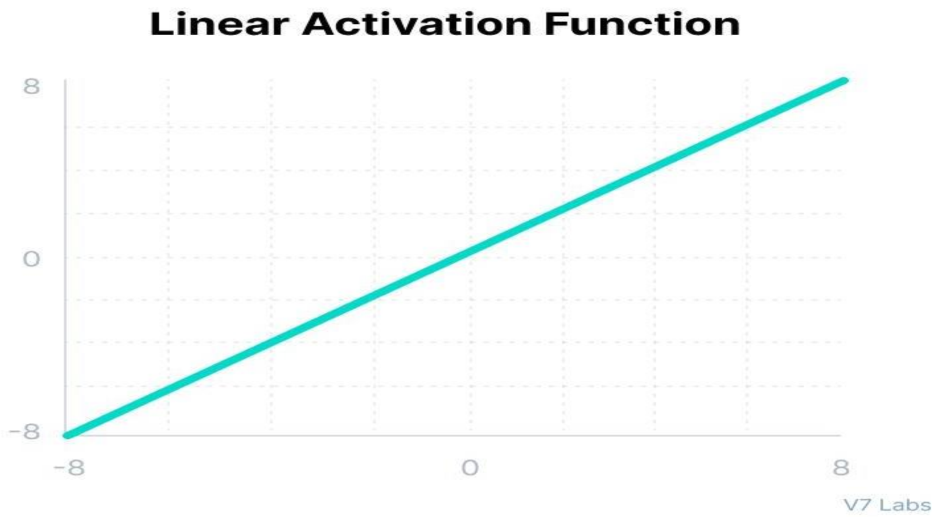
Thelimitationsofbinarystep functionare asfollows:

- Itcannotprovidemulti-valueoutputs–forexample,itcannotbeusedfor multi-class classificationproblems.
- Thegradientofthestepfunctioniszero,whichcausesahindranceinthe backpropagation process.

b) LinearActivationFunction:

Thelinearactivationfunction,alsoknownas"noactivation,"or"identity function"(multipliedx1.0),iswheretheactivationisproportionaltotheinput.

The function doesn't do anything to the weighted sum of the input, it simply spitsoutthevalueitwasgiven.



Mathematically,itcanberepresentedas:

$$\textit{Linear}$$

$$f(x) = x$$

However,alinearactivationfunctionhas twomajorproblems:

- It'snotpossibletousebackpropagationasthederivativeofthefunction isaconstantandhasnorelationtotheinputx.
- Alllayersoftheneuralnetworkwillcollapseintooneifalinearactivation functionisused.Nomatterthenumberoflayersintheneuralnetwork,

thelastlayerwillstillbealinearfunctionofthefirstlayer.So,essentially, alinearactivationfunctionturnstheneuralnetworkintojustonelayer.

Non-LinearActivationFunctions

Thelinearactivationfunctionshownaboveissimplyalinearregression model.Becauseof its limited power, this does not allow the model to create complexmappingsbetweenthenetwork'sinputsandoutputs.

Non-linear activation functions solve the following limitations of linear activation functions:

- Theyallowbackpropagationbecausenowthederivativefunctionwould berelatedtotheinput,andit'spossibletogobackandunderstandwhich weightsintheinputneuronscanprovideabetterprediction.
- Theyallowthestackingofmultiplelayersofneuronsastheoutputwould nowbeanon-linearcombinationofinputpassedthroughmultiplelayers. Anyoutputcanberepresentedasafunctionalcomputationinaneural network.

Belowaretendifferentnon-linearneuralnetworksactivationfunctionsand their characteristics.

a) Sigmoid/LogisticActivationFunction

This function takes any real value as input and outputs values in the rangeof0to1.Thelargertheinput(morepositive),thecloser theoutputvalue will be to 1.0, whereas the smaller the input (more negative), the closer the outputwillbeto0.0,asshownbelow.

Mathematicallyitcanberepresentedas:

$$\textit{Sigmoid / Logistic}$$

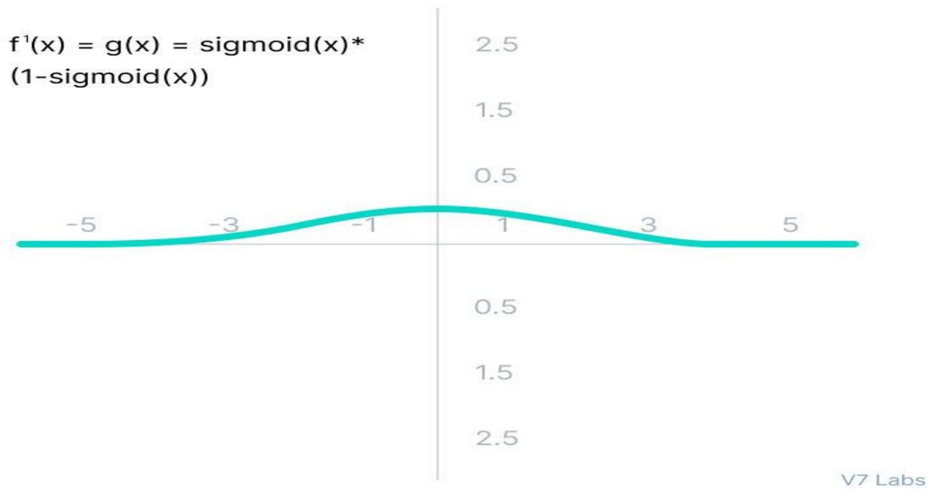
$$f(x) = \frac{1}{1 + e^{-x}}$$

Here’s why sigmoid/logistic activation function is one of the most widely used functions:

- Itiscommonlyusedformodelswherewehavetopredicttheprobability asanoutput.Sinceprobabilityofanythingexistsonlybetweentherange of0and1,sigmoidistherightchoicebecauseofitsrange.
- The function is differentiable and provides a smooth gradient, i.e., preventingjumpsinoutputvalues.ThisisrepresentedbyanS-shapeof thesigmoidactivationfunction.

Thelimitationsofsigmoidfunctionarediscussedbelow:

- Thederivativeofthefunctionisf'(x)=sigmoid(x)*(1-sigmoid(x)).

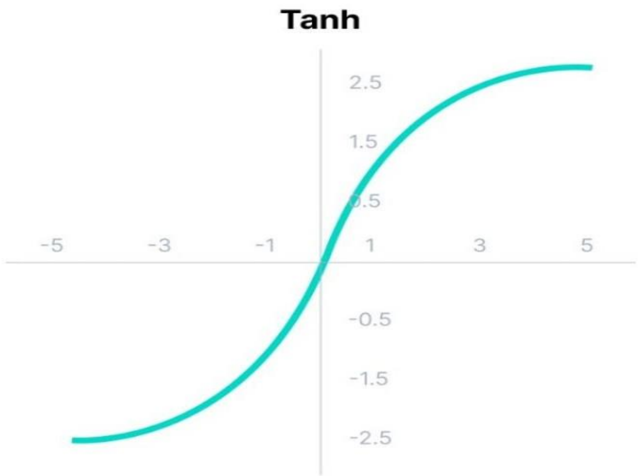


FromtheaboveFigure,thegradientvaluesareonlysignificantforrange -3 to 3, and the graph gets much flatter in other regions.It implies that for values greater than 3 or less than -3, the function will have very small gradients.Asthegradientvalueapproacheszero,thenetworkceasestolearn andsuffersfromthe*Vanishinggradient*problem.

- Theoutputofthelogisticfunctionisnotsymmetricaroundzero.Sothe outputofalltheneuronswillbeofthesamesign.Thismakesthetrainingofthen euralnetworkmoredifficultandunstable.

b) TanhFunction(HyperbolicTangent)

Tanhfunctionisverysimilartothesigmoid/logisticactivationfunction, andevenhasthesameS-shapewiththedifferenceinoutputrangeof-1to1. InTanh,thelargertheinput(morepositive),theclosertheoutputvaluewillbe to1.0,whereasthesmallertheinput(morenegative),thecloserthe outputwill be to -1.0.



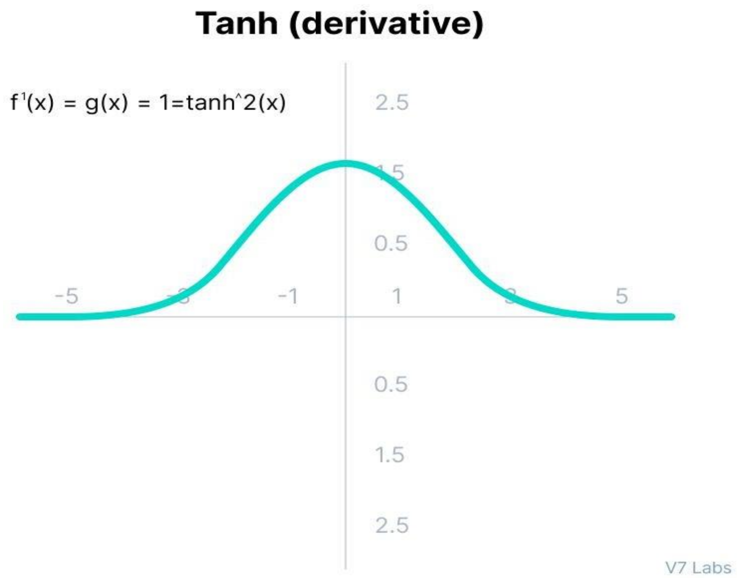
Mathematically,itcanberepresentedas:

$$Tanh$$

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Advantagesofusingthisactivationfunctionare:

- TheoutputofthetanhactivationfunctionisZerocentered;hencewecan easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between-1to;therefore,themeanforthehiddenlayercomesouttobe 0orveryclosetoit.Ithelpsincenteringthedataandmakeslearningfor the next layer much easier.



It also faces the problem of vanishing gradients similar to the sigmoid activationfunction.Plusthegradientofthetanhfunctionismuchsteeperas comparedtothesigmoidfunction.

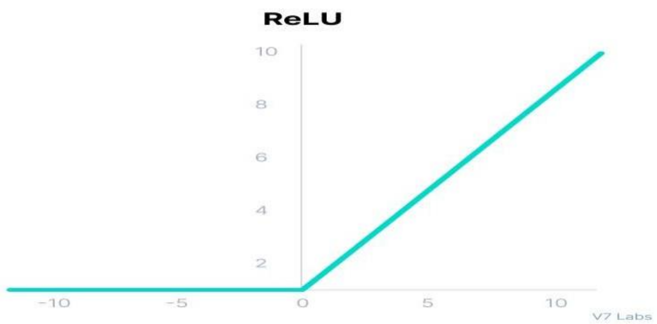
Note: **Althoughbothsigmoidandtanhfacevanishinggradientissue, tanhiszerocentered,andthegradientsarenotrestrictedtomoveina certain direction. Therefore, in practice, tanh nonlinearity is always preferred to sigmoid nonlinearity.**

c) ReLUFunction

ReLUstandsforRectifiedLinearUnit.Althoughitgivesanimpressionof a linear function, ReLU has a derivative function and allows for backpropagation whilesimultaneouslymaking itcomputationallyefficient.

ThemaincatchhereisthattheReLUfunctiondoesnotactivateallthe neurons at the same time.

Theneuronswillonlybedeactivatediftheoutputofthelinear transformationislessthan0.



Mathematically,itcanberepresentedas:

$$ReLU$$

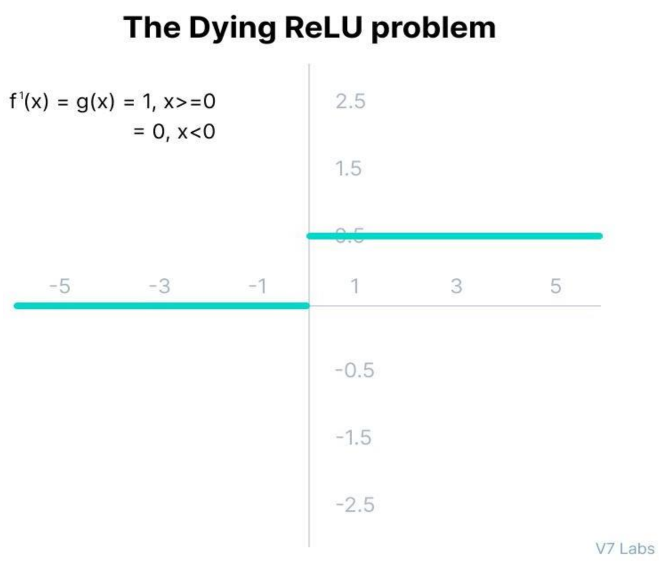
$$f(x) = max\ (0, x)$$

Theadvantages ofusingReLUasanactivation functionareas follows:

- Sinceonlyacertainnumberofneuronsareactivated,theReLUfunction isfarmorecomputationallyefficientwhencomparedtothesigmoidand tanh functions.
- ReLU accelerates the convergence of gradient descent towards the global minimum of theloss functiondue to its linear, non-saturating property.

Thelimitationsfacedbythisfunctionare:

- TheDyingReLUproblem.



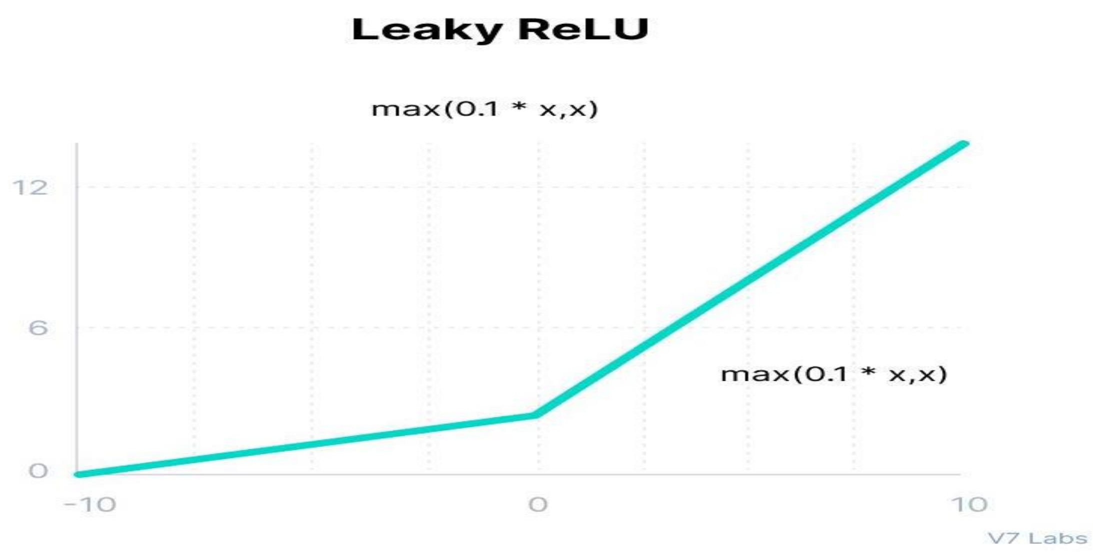
The negative side of the graph makes the gradient value zero. Due to thisreason,duringthebackpropagationprocess,theweightsandbiasesfor someneuronsarenotupdated.Thiscancreatedeadneuronswhichneverget activated.

- Allthenegativeinputvaluesbecomezeroimmediately,whichdecreases themodel'sabilitytofitortrainfromthedataproperly.

Note: ForbuildingthemostreliableMLmodels,splityourdataintotrain,validation, and test sets.

d) LeakyReLU Function

LeakyReLUisanimprovedversionofReLUfunctiontosolvetheDying ReLUproblemasithasasmallpositiveslopeinthenegativearea.



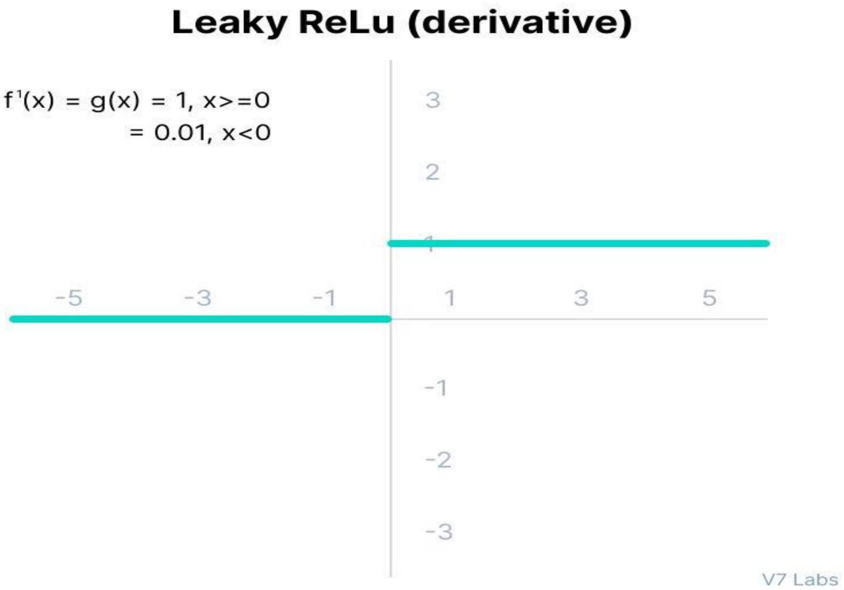
Mathematically,itcanberepresentedas:

$$\textit{Leaky ReLU}$$

$$f(x) = max\ (0.1x, x)$$

TheadvantagesofLeakyReLUaresameasthatofReLU,inadditionto thefactthatitdoesenablebackpropagation,evenfornegativeinputvalues.By makingthisminormodificationfornegativeinputvalues,thegradientoftheleft sideofthegraphcomesouttobeanon-zerovalue.Therefore,we wouldno longerencounterdeadneuronsinthatregion.

`HereisthederivativeoftheLeakyReLUfunction.

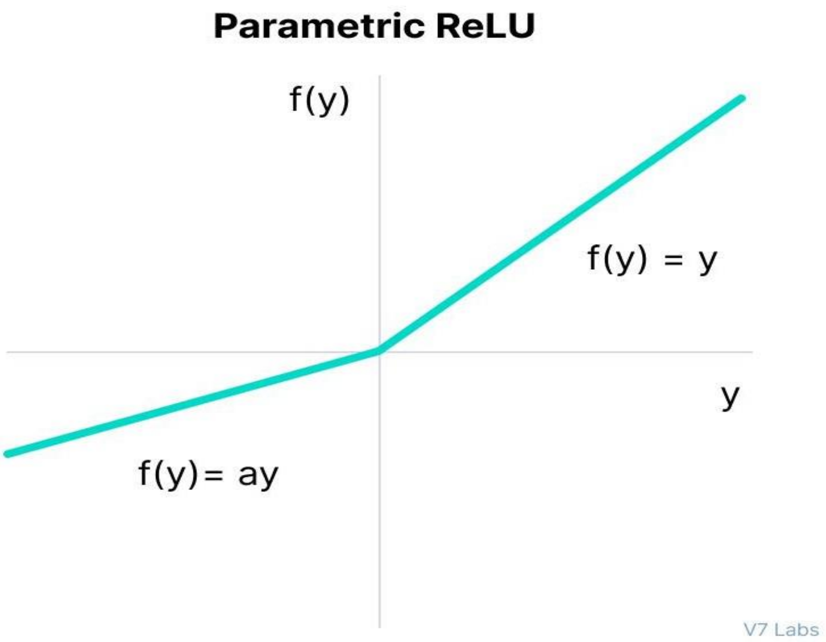


Thelimitationsthatthisfunctionfacesinclude:

- Thepredictionsmaynotbeconsistentfornegativeinputvalues.
- Thegradientfornegativevaluesisasmallvaluethatmakesthelearning ofmodelparameterstime-consuming.

d) *ParametricReLUFunction*

Parametric ReLU is another variant of ReLU that aims to solve the problemofgradient'sbecomingzerofortheleft halfoftheaxis.Thisfunction provides the slope of the negative part of the function as an argument*a*. By performingbackpropagation,themostappropriatevalueof*a*islearnt.



Mathematically,itcanberepresentedas:

$$\text{Parametric ReLU}$$

$$f(x) = max\ (ax, x)$$

Where*"a"*is theslopeparameterfornegativevalues.

TheparameterizedReLUfunctionisusedwhentheleakyReLUfunction stillfailsatsolvingtheproblemofdeadneurons,andtherelevantinformationis notsuccessfullypassedtothenextlayer.

This function’s limitation is that it may perform differently for different problemsdependinguponthevalueofslopeparameter***a.***

TypesofpoolingLayers:

AConvolutionalneuralnetwork(CNN)isaspecialtypeofArtificialNeuralNetworkthat is usually used for image recognition and processing due to its ability to recognize patterns in images. It eliminates the need to extract features from visual data manually. It learns images by sliding a filter of some size on them and learning not just the features from the data but also keeps Translation invariance.

Thetypicalstructureofa CNNconsistsof threebasiclayers

1. **Convolutional layer:**These layers**generate a feature map**by sliding a filter over the input image and recognizing patterns in images.
2. **Poolinglayers:**Theselayers **downsamplethefeaturemap**tointroduceTranslation invariance, which reduces the overfitting of the CNN model.
3. **FullyConnectedDenseLayer:**Thislayercontainsthe**samenumberofunitsasthenumber of classes** and the output activation function such as “softmax” or “sigmoid”

Whatare Pooling layers?

Pooling layers are one of the building blocks of Convolutional Neural Networks. Where Convolutional layers **extract features**from images, Pooling layers **consolidate the features**learned by CNNs. Its purpose is to gradually shrink the representation’s spatial dimension to minimize the number of parameters and computations in the network.

WhyarePoolinglayersneeded?

Thefeaturemapproducedbythefiltersof Convolutionallayersislocation-dependent. For example, If an object in an image has shifted a bit it might not be recognizable by the Convolutional layer. So, it means that the feature map records the precise positions offeatures in the input. What pooling layers provide is “Translational Invariance” which makes the CNN invariant to translations, i.e., even if the input of the CNN is translated, the CNN will still be able to recognize the features in the input.

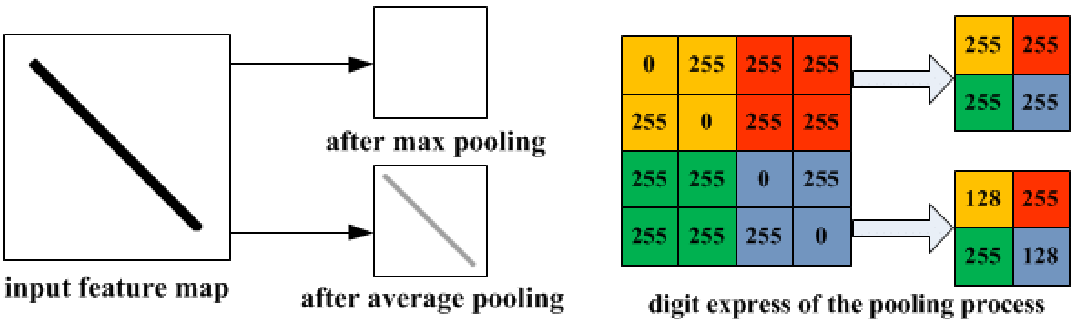
In all cases, poolinghelps to make the representation become approximatelyinvariant to smalltranslations of the input. Invariance to translation means that ifwe translate the input by a small amount, the values of most of the pooled outputs do not change.

HowdoPoolinglayersachieve that?

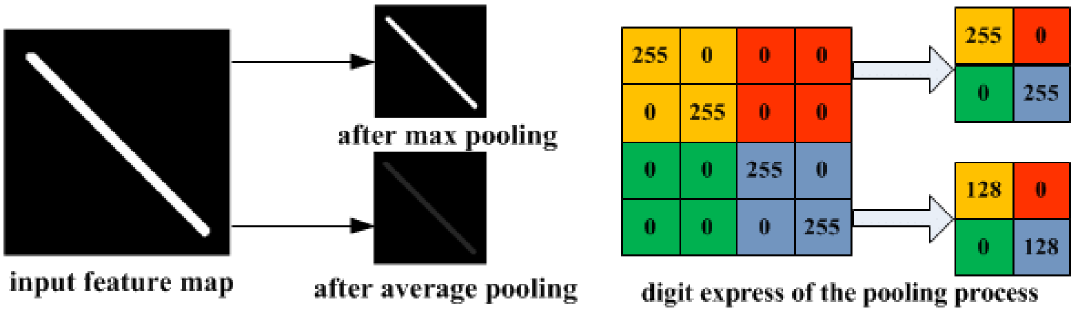
A Pooling layer is added after the Convolutional layer(s), as seen in the structure of a CNN above. It down samples the output of the Convolutional layers by sliding the filter of some size with some stride size and calculating the maximum or average of the input.

Thereare **twotypesofpooling**thatare used:

1. **Max pooling**: This works by selecting the maximum value from every pool. Max Pooling retains the**most prominent**features of the feature map, and the returned image is sharper than the original image.
2. **Average pooling**: This pooling layer works by getting the average of the pool. Average pooling retains the**average values**of features of the feature map. It smoothes the image while keeping the essence of the feature in an image.



(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

TheworkingofPoolingLayersusingTensorFlow.CreateaNumPyarray and reshape it.

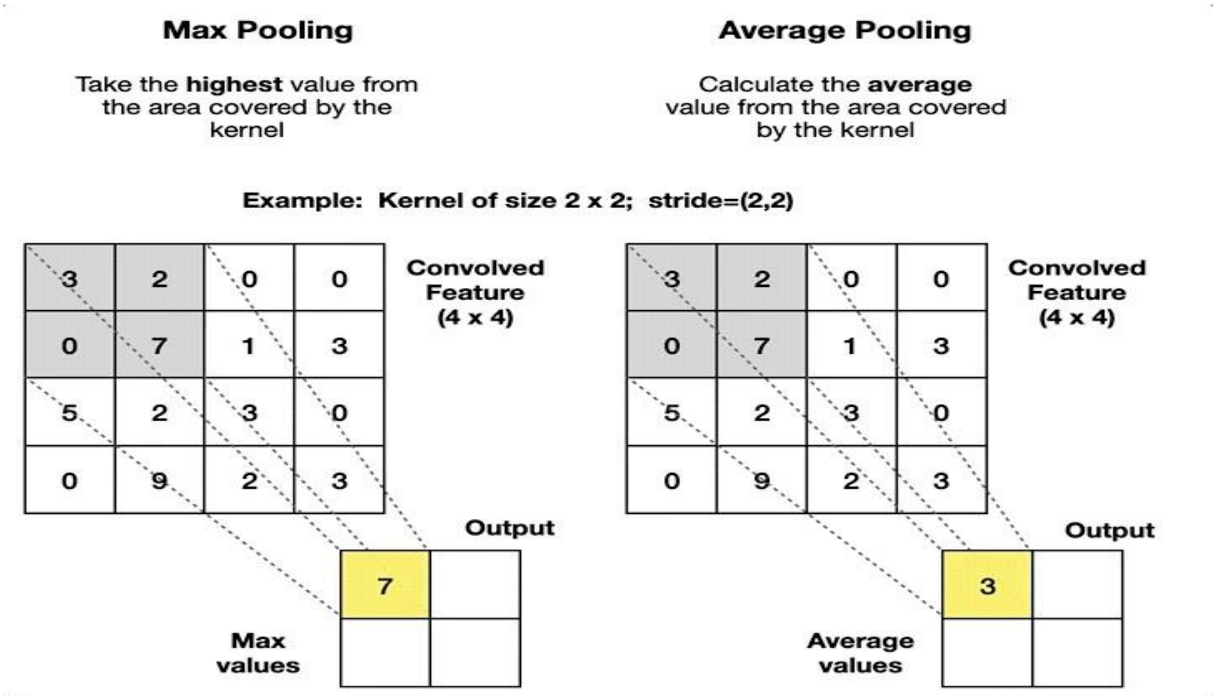
MaxPooling

Create a MaxPool2D layer with pool_size=2 and strides=2. Apply the MaxPool2D layer to the matrix, and you will get the MaxPooled output in the tensor form. By applying it tothematrix,theMax poolinglayerwillgothroughthematrix bycomputingthemax ofeach 2×2poolwithajumpof2.Printtheshapeofthetensor.Usetf.squeezetoremovedimensions of size 1 from the shape of a tensor.

Average Pooling

Create an AveragePooling2D layer with the same 2 pool_size and strides. Apply the AveragePooling2Dlayer tothematrix. Byapplyingit tothematrix,theaveragepoolinglayer will go through the matrix by computing the average of 2×2 for each pool with a jump of 2. Print the shape of the matrix and Use tf.squeeze to convert the output into a readable form by removing all 1 size dimensions.

The GIF here explains how these pooling layers go through the input matrix and computes the maximum or average for max pooling and average pooling, respectively.



GlobalPooling Layers

Global Pooling Layers often replace the classifier's fully connected or Flatten layer. The model instead ends with a convolutional layer that produces as many feature maps as there are target classes and performs global average pooling on each of the feature maps to combine each feature map into a single value.

Create the same NumPy array but with a different shape. By keeping the same shape as above, the Global Pooling layers will reduce them to one value.

GlobalAverage Pooling

Considering a tensor of shape**h*w*n**, the output of the Global Average Pooling layer is a single value across **h*w** that summarizes the presence of the feature. Instead of downsizingthepatchesoftheinputfeaturemap,theGlobalAveragePoolinglayerdownsizes the whole **h*w** into 1 value by taking the average.

GlobalMaxPooling

With the tensor of shape **h*w*n**, the output of the Global Max Pooling layer is a single value across**h*w**that summarizes the presence of a feature. Instead of downsizing the patchesoftheinputfeaturemap,theGlobalMaxPoolinglayerdownsizesthe whole **h*w** into 1 value by taking the maximum.

TrainingofCNNinTensorFlow

The MNIST database (**Modified National Institute of Standard Technology database**) is an extensive database of handwritten digits, which is used for training various image processing systems. It was created by "**reintegrating**" samples from the original dataset of the **MNIST**.

If we get familiarized with the building blocks of Connects, we can build one with TensorFlow. We can use the MNIST dataset for image classification.

Preparing the data is the same as in the previous tutorial. We can run codeand jump directly into the architecture of CNN.

Here, the code isexecuted in **Google Colab**(an online editor of machine learning).WecangotoTensorFloweditorthroughthebelowlink:
https://colab.research.google.com

Theseare thestepsusedtotrainingtheCNN.

Steps:

Step 1: Upload Dataset

Step 2: The Input layer

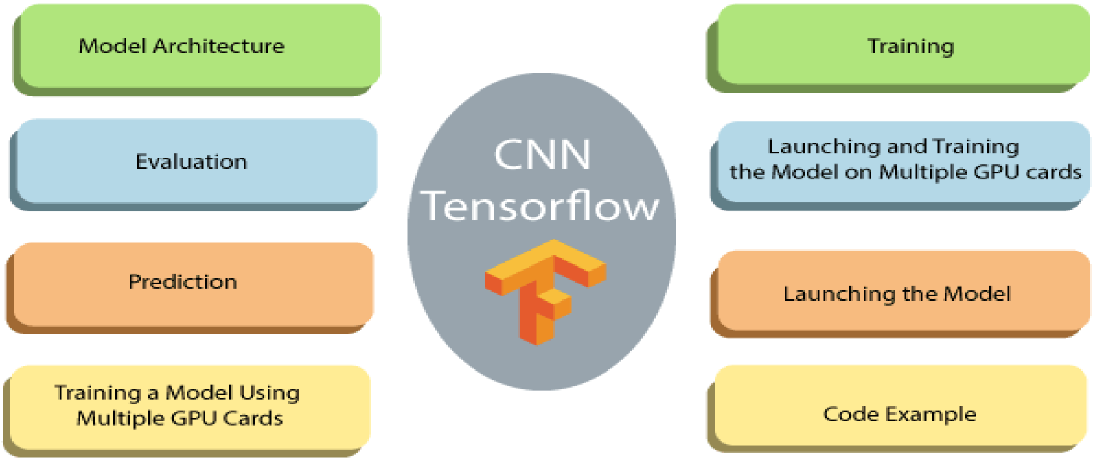
Step3: Convolutionallayer

Step 4: Pooling layer

Step5: ConvolutionallayerandPoolingLayer

Step6: Denselayer

Step7: Logit Layer



Step1:UploadDataset

The MNIST dataset is available with scikit for learning in this URL (Unified ResourceLocator).Wecandownloaditandstoreitinourdownloads.We canupload it with fetch_mldata ('MNIST Original').

Createatest/trainset

Weneed tosplitthedatasetinto**train_test_split**.

Scalethefeatures

Finally,wescalethefunctionwiththehelpof **MinMaxScaler**.

1. **import**numpyasnp
2. **import**tensorflowastf
3. fromsklearn.datasets**import**fetch_mldata
4. #ChangeUSERNAMEbytheusernameofthe machine
5. ##WindowsUSER
6. mnist=fetch_mldata('C:\\Users\\USERNAME\\Downloads\\MNISToriginal')
7. ##MacUser
8. mnist=fetch_mldata('/Users/USERNAME/Downloads/MNISToriginal')
9. print(mnist.data.shape)
10. print(mnist.target.shape)
11. fromsklearn.model_selection**import**train_test_split

12. A_train,A_test,B_train,B_test=train_test_split(mnist.data,mnist.target,test_size=0.2, random_state=45)
13. B_train= B_train.astype(int)
14. B_test=B_test.astype(int)
15. batch_size=len(X_train)
16. print(A_train.shape,B_train.shape,B_test.shape)
17. ##rescale
18. fromsklearn.preprocessingimportMinMaxScaler
19. scaler=MinMaxScaler()
20. #Trainthe Dataset
21. X_train_scaled=scaler.fit_transform(A_train.astype(np.float65))

1. #testthedataset
2. X_test_scaled=scaler.fit_transform(A_test.astype(np.float65))
3. feature_columns=[tf.feature_column.numeric_column('x',shape=A_train_scaled.shape[1:])]
4. X_train_scaled.shape[1:]

DefiningtheCNN(ConvolutionalNeuralNetwork)

CNN uses filters on the pixels of any image to learn detailed patterns comparedto global patterns with a traditional neural network. To create CNN, we have to define:

1. **A convolutional Layer:** Apply the number of filters to the feature map. After convolution, we need to use a relay activation function to add non-linearity to the network.
2. **Pooling Layer:**The next step after the Convention is to downsampling the maximum facility. The objective is to reduce the mobility of the feature map to prevent overfitting and improve the computation speed. Max pooling is a traditional technique, which splits feature maps into subfields and only holds maximum values.
3. **Fully connected Layers:**All neurons from the past layers are associated with the other next layers. The CNN has classified the label according to the features from convolutional layers and reduced with any pooling layer.

CNNArchitecture

- **ConvolutionalLayer:**Itapplies145x5filters(extracting5x5-pixelsub-regions),

- **Pooling Layer:**This will perform max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap).
- **ConvolutionalLayer:**Itapplies365x5filters,withReLUactivationfunction
- **PoolingLayer:**Again,performsmaxPoolingwitha2x2filterandstrideof 2.
- **1,764 neurons,**with the dropout regularization rate of 0.4 (where the probability of 0.4 that any given element will be dropped in training)
- **Dense Layer (LogitsLayer):**Thereare tenneurons, oneforeachdigittargetclass(0- 9).

ImportantmodulestouseincreatingaCNN:

1. Conv2d().Constructatwo-dimensionalconvolutionallayerwiththenumberoffilters, filter kernel size, padding, and activation function like arguments.
2. max_pooling2d (). Construct a two-dimensional pooling layer using the max-pooling algorithm.
3. Dense().Constructadenselayerwiththehiddenlayersand units

Wecandefinea functiontobuildCNN.

The following represents steps to construct every building block before wrapping everything in the function.

Step2:Inputlayer

```
1. #Inputlayer
2. defcnn_model_fn(mode,features, labels):
3. input_layer=tf.reshape(tensor=features["x"],shape=[-1,26,26,1])
```

Weneedtodefineatensorwiththeshapeofthedata.Forthat,wecanuse the**module tf.reshape**. In this module, we need to declare the tensor to reshapeand to shape the tensor. The first argument is the feature of the data, that is defined in the argument of a function.

A picture has a width, a height, and a channel. The**MNIST**dataset is a monochromic picture with the**28x28**size. We set the batch size into -1 in the shape argument so that it takestheshapeofthefeatures["x"]. Theadvantageisto tunethe batch size to hyperparameters. If the batch sizeis 7, the tensor feeds**5,488**values (**28 * 28 * 7**).

Step3:ConvolutionalLayer

```
1. #firstCNNLayer
```

```
2. conv1=tf.layers.conv2d(
3. inputs=input_layer,
4. filters=18,
5. kernel_size=[7,7],
6. padding="same",
7. activation=tf.nn.relu)
```

The first convolutional layer has 18 filters with the kernel size of 7x7 with equal padding. The same padding has both the output tensor and input tensor have the same width and height. TensorFlow will add zeros in the rowsand columns to ensure the same size. We use the ReLu activation function. The output size will be [28, 28, and 14].

Step4:Pooling layer

The next step after the convolutional is pooling computation. The pooling computation will reduce the extension of the data. We can use the module max_pooling2d with a size of 3x3 and stride of 2. We use the previous layer as input. The output size can be [batch_size, 14, 14, and 15].

```
1. ##firstPoolingLayer
2. pool1=tf.layers.max_pooling2d(inputs=conv1,pool_size=[3,3],strides=2)
```

Step5:PoolingLayerand SecondConvolutionalLayer

The second CNN has exactly 32 filters, with the output size of [batch_size, 14, 14, 32]. The size of the pooling layer has the same as ahead, and output shape is [batch_size, 14, 14, and18].

```
1. conv2= tf.layers.conv2d(
2. inputs=pool1,
3. filters=36,
4. kernel_size=[5,5],
5. padding="same",
6. activation=tf.nn.relu)
7. pool2=tf.layers.max_pooling2d(inputs=conv2,pool_size=[2,2],strides=2).
```

Step 6:Fullyconnected (Dense)Layer

We have to define the fully-connected layer. The feature map has to be compressed before to be combined with the dense layer. We can use the module reshape with a size of **7*7*36**.

The dense layer will connect**1764**neurons. We add a ReLu activation function and can add a ReLu activation function. We add a dropout regularization term with a rateof0.3,meaning30percentoftheweightswillbe0.Thedropouttakesplaceonly along the training phase. The**cnn_model_fn()**has an argument mode to declare if the model needs to trained or to be evaluate.

```
1. pool2_flat=tf.reshape(pool2, [-1,7*7*36])
2. dense=tf.layers.dense(inputs=pool2_flat,units=7*7*36,activation=tf.nn.relu)
3. dropout=tf.layers.dropout(inputs=dense,rate=0.3,training=mode==tf.esti
   mator.ModeKeys.TRAIN)
```

Step7:Logits Layer

Finally,wedefinethelastlayerwiththepredictionofmodel.Theoutputshape is equal to the batch size 12, equal to the total number of images in the layer.

```
1. #LogitLayer
2. logits=tf.layers.dense(inputs=dropout,units=12)
```

We can create a dictionary that contains classes and the possibility of each class. The module returns the highest value with tf.argmax() if the logit layers. The softmax function returns the probability of every class.

PopularCNNarchitectures-VGG,GoogleNet,ResNet:

TypesofConvolutionalNeuralNetworkAlgorithms

LeNet

LeNet is a pioneering CNN designed for recognizing handwritten characters. It was proposed by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in the late 1990s. LeNet consists of a series of convolutional and pooling layers, as well as a fully connected layer and softmax classifier. It was among the first successful applications of deep learning for computer vision. It has been used by banks to identify numbers written on cheques in grayscale input images.

VGG

VGG (Visual GeometryGroup) is a research group within the Department of Engineering Science at the Universityof Oxford. The VGG group is well-known for its work in computer vision, particularlyin the area of convolutional neural networks (CNNs).

One of the most famous contributions from the VGG group is the VGG model, also known as VGGNet. The VGG model is a deep neural network that achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge in 2014, and has been widely used as a benchmarkfor image classification and object detection tasks.

The VGG model is characterized by its use of small convolutional filters (3×3) and deep architecture (up to 19 layers), which enables it to learn increasingly complex features from input images. The VGG model also uses max pooling layers to reduce the spatial resolution of the feature maps and increase the receptive field, which can improve its ability to recognize objects of varying scales and orientations.

The VGG model has inspired many subsequent research efforts in deep learning, including the development of even deeper neural networks and the use of residual connections to improve gradient flow and training stability.

ResNet

ResNet (short for “Residual Neural Network”) is a family of deep convolutional neural networks designed to overcome the problem of vanishing gradients that are common in very deep networks. Theidea behind ResNet is to use “residual blocks” that allow for the direct propagation of gradients throughthe network, enabling the training of very deep networks.

A residual block consists of two or more convolutional layers followed by an activation function, combined with a shortcut connection that bypasses the convolutional layers and adds the original input directly to the output of the convolutional layers after the activation function.

This allows the network to learn residual functions that represent the difference between the convolutional layers' input and output, rather than trying to learn the entire mapping directly. The use of residual blocks enables the training of very deep networks, with hundreds or thousands of layers, significantly alleviating the issue of vanishing gradients.

GoogLeNet

GoogLeNet is a deep convolutional neural network developed by researchers at Google. It was introduced in 2014 and won the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)that year, with a top-five error rate of 6.67%.

GoogLeNet is notable for its use of the Inception module, which consists of multiple parallel convolutional layers with different filter sizes, followed by a pooling layer, and concatenation of the outputs. This design allows the network to learn features at multiple scales and resolutions, while keeping the computational cost manageable. The network also includes auxiliary classifiers at intermediate layers, which encourage the network to learn more discriminative features and prevent overfitting.

GoogLeNet builds upon the ideas of previous convolutional neural networks, including LeNet, which was one of the first successful applications of deep learning in computer vision. However, GoogLeNet is much deeper and more complex than LeNet.

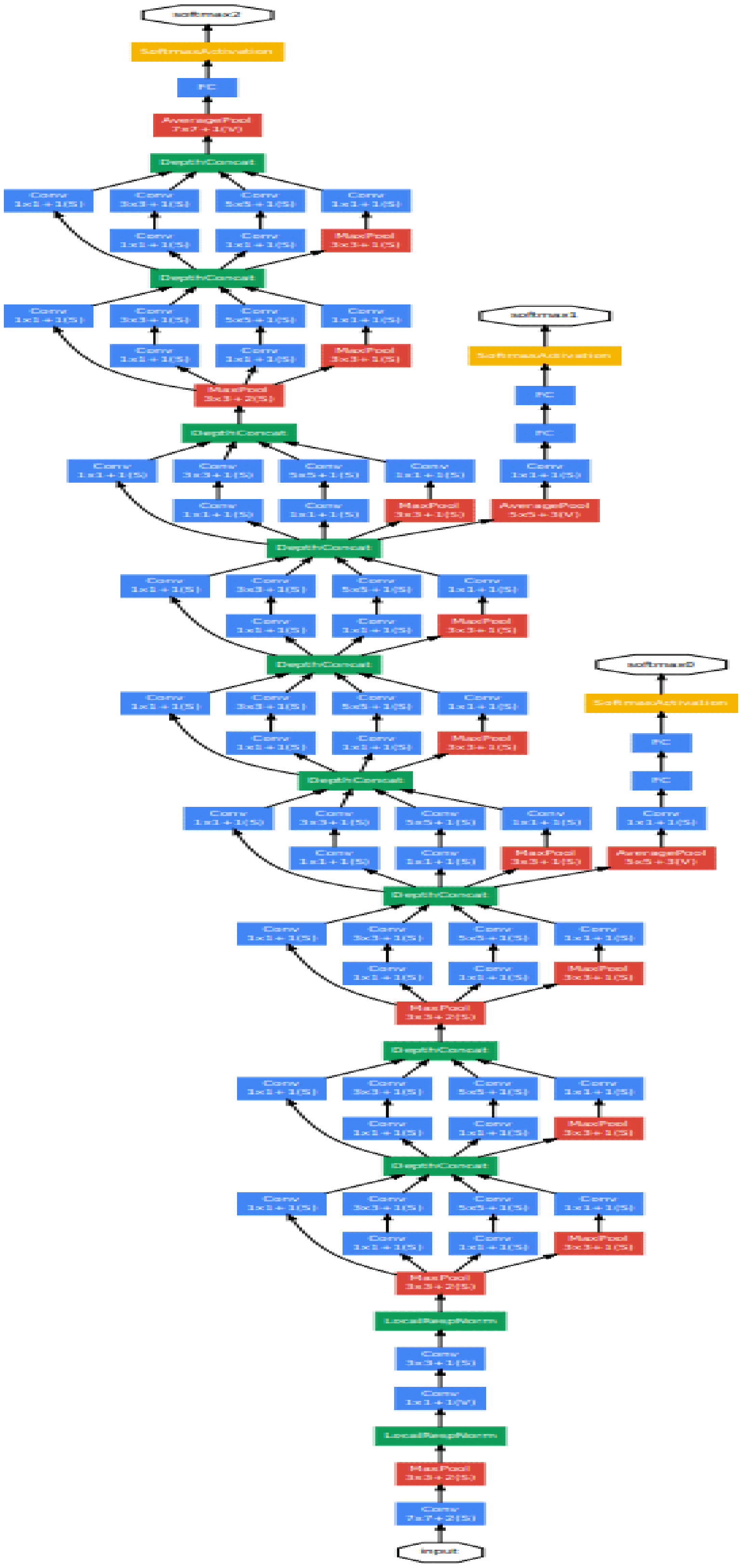


Figure 3: GoogLeNet network with all the bells and whistles

Dropout:

The term “dropout” refers to dropping out the nodes (input and hidden layer) in a neural network (as seen in Figure 1). All the forward and backwards connections with a dropped node are temporarily removed, thus creating a newnetworkarchitectureoutoftheparentnetwork.Thenodesaredroppedby a dropout probability of p.

Considergiveninputx:{1,2,3,4,5}tothefullyconnectedlayer.Wehave a dropout layer with probability p = 0.2 (or keep probability = 0.8). During the forward propagation (training) from the input x, 20% of the nodes would be dropped, i.e. the x could become {1, 0, 3, 4, 5} or {1, 2, 0, 4, 5} and so on. Similarly, it applied to the hidden layers.

For instance, if the hidden layers have 1000 neurons (nodes) and a dropout is applied with drop probability = 0.5, then 500 neurons would be randomly dropped in every iteration (batch).

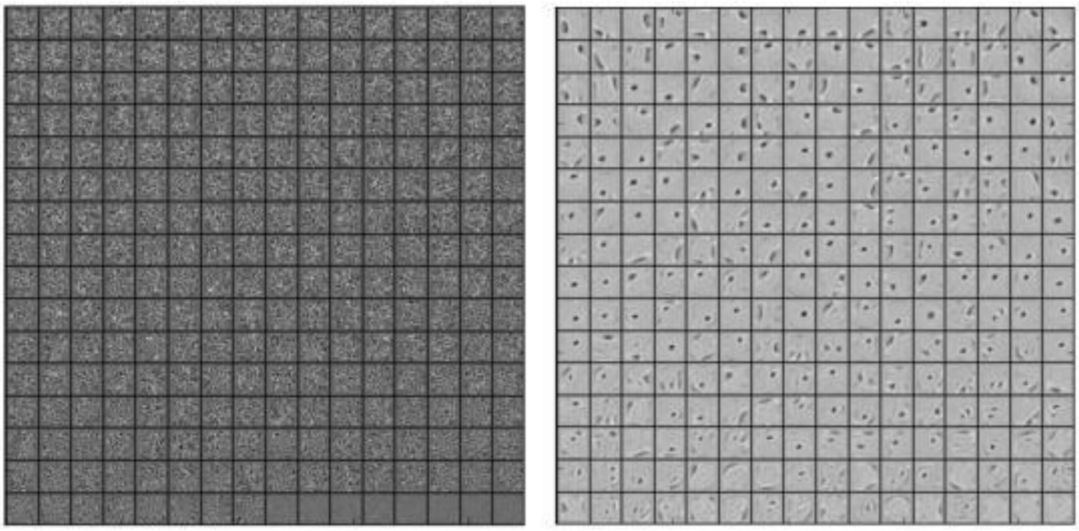
Generally, for the input layers, the keep probability, i.e. 1- drop probability, is closer to 1, 0.8 being the best as suggested by the authors. For the hidden layers, the greater the drop probability more sparse the model, where 0.5 is the most optimised keep probability, that states dropping 50% of the nodes.

HowdoesDropoutsolvetheOverfittingproblem?

In the overfitting problem, the model learns the statistical noise. To be precise, the main motive of training is to decrease the loss function, given all the units (neurons). So in overfitting, a unit may change in a way that fixes up themistakesoftheother units.Thisleadstocomplexco-adaptations,whichin turn leads to the overfitting problem because this complex co-adaptation fails to generalise on the unseen dataset.

Now, if we use dropout, it prevents these units to fix up the mistake of otherunits,thuspreventingco-adaptation,asineveryiterationthepresenceof a unit is highly unreliable. So, by randomly dropping a few units (nodes), it forces the layers to take more or less responsibility for the input by taking a probabilistic approach.

This ensures that the model is getting generalised and hence reducing the overfitting problem.



(a) Without dropout (b) Dropout with $p = 0.5$.

Figure2:(a)Hiddenlayerfeatureswithoutdropout;

(b)Hiddenlayerfeatureswithdropout

Fromfigure2,wecaneasilymakeoutthatthehiddenlayerwithdropout is learning more of the generalised features than the co-adaptations in the layer without dropout. It is quite apparent, that dropout breaks such inter-unit relations and focuses more on generalisation.

DropoutImplementation

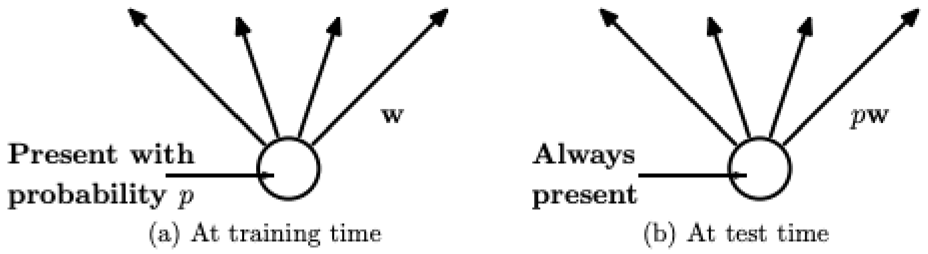


Figure3:(a)Aunit(neuron)duringtrainingispresentwitha probability p and is connected to the next layer with weights ‘w’**;**

(b) A unitduring inference/prediction is always present and is connected to the next layer with weights, ‘pw’

In the original implementation of the dropout layer, during training, a unit (node/neuron) in a layer is selected with a keep probability (1-drop probability). This creates a thinner architecture in the given training batch, and every time this architecture is different.

Inthestandardneuralnetwork,duringtheforwardpropagationwehave the following equations:

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

Figure4:Forwardpropagationofastandardneuralnetwork

where:
z:denotethevectorofoutputfromlayer(l+1)beforeactivation y:
denote the vector of outputs from layer l
w:weightofthelayerl b:
bias of the layer l

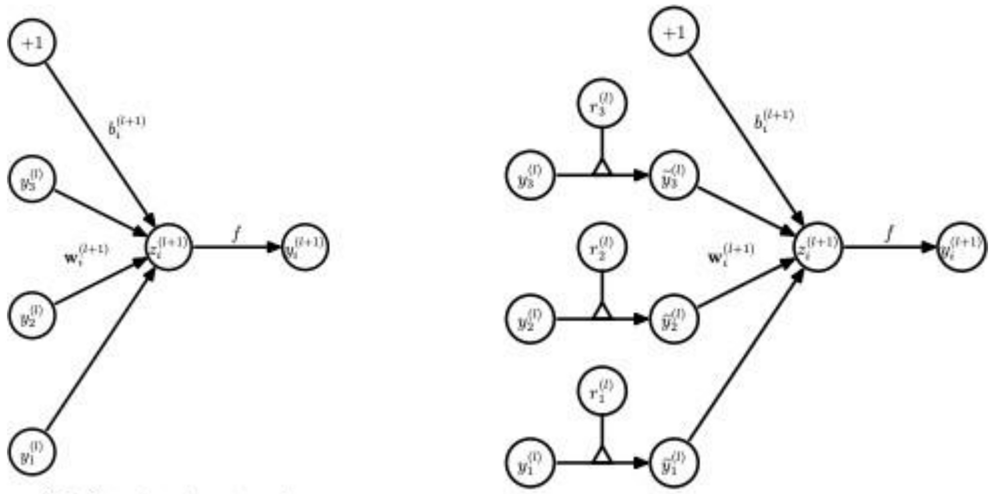
Further, with the activation function, z is transformed into the output for layer (l+1). Now, if we have a dropout, the forward propagation equations change in the following way:

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p),\\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)},\\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\widetilde{\mathbf{y}}^{l} + b_i^{(l+1)},\\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

Figure5:Forwardpropagationofalayerwithdropout

So, before we calculate**z,**the input to the layer is sampled and multiplied element-wise with the independent Bernoulli variables.**r**denotes the Bernoulli random variables each of which has a probability p of being 1.

Basically,**r**acts as a mask to the input variable, which ensures only a few unitsarekeptaccordingtothekeepprobabilityofadropout.Thisensuresthat we have thinned outputs "**y(bar)**", which is given as an input to the layer during feed-forward propagation.



(a) Standard network (b) Dropout network

Training Deep Neural Networks is a difficult task that involves several problems to tackle. Despite their huge potential, they can be slow and be prone to overfitting. Thus, studies on methods to solve these problems are constant in Deep Learning research.

Batch Normalization – commonly abbreviated as Batch Norm – is one of thesemethods.Currently,itisawidelyusedtechniqueinthefieldofDeep

Learning.ItimprovesthelearningspeedofNeuralNetworksandprovides regularization, avoiding overfitting.

Normalization:

Normalization is a pre-processing technique used to standardize data.In other words, having different sources of data inside the same range. Not normalizing the data before training can cause problems in our network, making it drastically harder to train and decrease its learning speed.

For example, imagine we have a car rental service. Firstly, we want to predict a fair price for each car based on competitors' data. We have two features per car: the age in years and the total amount of kilometers it has been driven for. These can have very different ranges, ranging from 0 to 30 years, while distance couldgo from0up tohundredsofthousandsofkilometers.Wedon'twantfeatures to have these differences in ranges, as the value with the higher range might bias our models into giving them inflated importance.

There are two main methods to normalize our data. The moststraightforward method is to scale it to a range from 0 to 1. The data point to normalize,the mean of the data set,the highest value, andthe lowest value. This technique is generally used in the inputs of the data. The non- normalized data points with wide ranges can cause instability in Neural Networks. The relatively large inputs can cascade down to the layers, causing problems such as exploding gradients.

Theother techniqueused to normalize datais forcing thedatapoints to have a mean of 0 and a standard deviation of 1, using the following formula:

$$x_{normalized} = \frac{x - m}{s}$$

beingthe data point to normalize,the mean of the data set, andthe standard deviation of the data set. Now, each data point mimics a standard normal distribution.Havingallthefeaturesonthisscale,noneofthemwillhaveabias, and therefore, our models will learn better.

InBatchNorm,weusethislasttechniquetonormalizebatchesofdata inside the network itself.

BatchNormalization

Batch Norm is a normalization technique done between the layers of a NeuralNetwork instead of in the raw data. It isdone along mini-batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier.

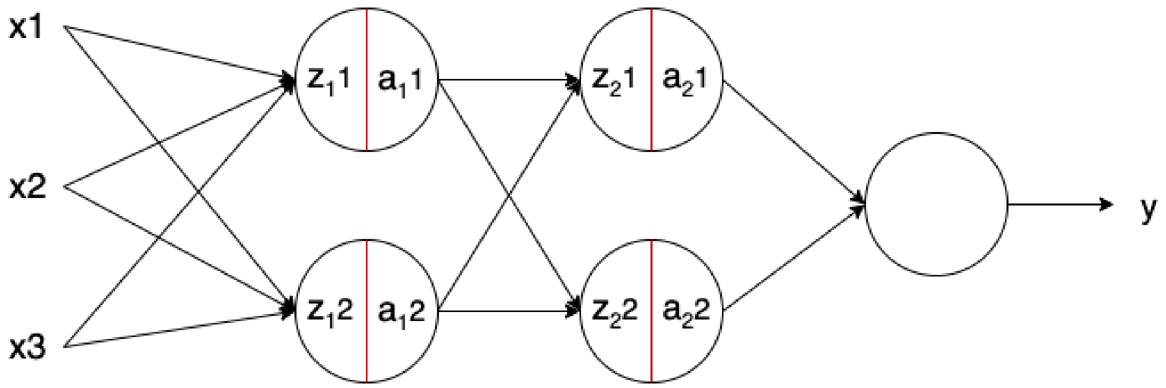
Following thetechniqueexplained in thepreviоussection,wecandefinethe normalization formula of Batch Norm as:

$$z^N = \left(\frac{z - m_z}{s_z}\right)$$

being$m_z$the mean of the neurons' output and $s_z$the standard deviation of the neurons' output.

HowIs ItApplied?

Thefollowingimagerepresentsaregularfeed-forwardNeural Network:are the inputs,the output of the neurons,the output of the activation functions, andthe output of the network:



Batch Norm–in the image represented with a red line–is applied to the neurons'outputjustbeforeapplyingtheactivationfunction.Usually,aneuronwithout BatchNormwouldbecomputedasfollows:

$$z = g(w, x) + b; \qquad a = f(z)$$

beingthelineartransformationofthe neuron, theweightsoftheneuron, thebiasoftheneurons,and theactivationfunction.Themodellearnsthe parameters and. Adding Batch Norm, it looks as:

$$z = g(w, x); \qquad z^N = \left(\frac{z - m_z}{s_z}\right) \cdot \gamma + \beta; \qquad a = f(z^N)$$

being the output of Batch Norm, the mean of the neurons' output,thestandarddeviationoftheoutputoftheneurons,and learningparametersof Batch Norm. Note that the bias of the neurons () is removed. This is because as we subtractthemean m_z,anyconstantoverthevaluesof *z*–suchas *b*–canbe ignored as it will be subtracted by itself.

The parameters βandγshift the mean and standard deviation, respectively. Thus, the outputs of Batch Norm over a layer result in a distribution withameanandastandarddeviationof γ.Thesevaluesarelearnedover epochs and the other learning parameters, such as the weights of the neurons, aiming to decrease the loss of the model.

DataAugmentation:

Algorithms can use machine learning to identify different objects and classify them for image recognition. This evolving technology includes using Data Augmentation to produce better-performing models. Machine learning models need to identify an object in any condition, even if it is rotated, zoomed in, or a grainy image. Researchers needed an artificial way of adding training data with realistic modifications.

Data augmentation is the addition of new data artificially derived from existing trainingdata.Techniquesinclude resizing, flipping, rotating, cropping, padding,etc. It helps to address issues like overfitting and data scarcity, and it makes the model robust with better performance. Data Augmentation provides many possibilities to alter the original image and can be useful to add enough data for larger models.

DataAugmentationinaCNN:

Convolutional Neural Networks (CNNs) can do amazing things if there is sufficient data. However, selecting the correct amount of training data for allof the features that need to be trained is a difficult question. If the user does not have enough, the networkcanoverfiton the trainingdata.Realisticimages contain a variety of sizes, poses, zoom, lighting, noise, etc.

To make the network robust to these commonly encountered factors, the method of Data Augmentation is used. By rotating input images todifferent angles, flipping images along different axes, or translating/cropping the images the network will encounter these phenomena during training.

As more parameters are added to a CNN, it requires more examples to show to the machine learning model. Deeper networks can have higher performance, but the trade-off is increased training data needs and increased training time.

DataAugmentationTechniques	DataAugmentationFactor
Flipping	2-4x(ineachdirection)
Rotation	Arbitrary
Translation	Arbitrary
Scaling	Arbitrary
SaltandPepperNoise Addition	Atleast2x(dependsontheimplementation)

Atableoutliningthefactorbywhichdifferentmethodsmultiplytheexistingtraining data.

DataAugmentationTechniques:

Some libraries use Data Augmentation by actually copying the training images and saving these copies as part of the total. This produces new training examples to feed to the machine learning model. Other libraries simply define a set of transformsto perform on the input training data. These transforms are appliedrandomly.Asa result,the space the optimizer issearchingis increased. Thishastheadvantagethatitdoesnotrequireextra diskspacetoaugmentthe training.

ImageDataAugmentationinvolvesthetechniquessuchas

a) Flips:

By Flipping images, the optimizer will not become biased that particular features of an image are only on one side. To do this augmentation, theoriginal training image is flipped vertically or horizontally over one axis of the image. As a result, the features continually change directions.

StellathePuppysittingonacarseat StellathePuppyFlippedovertheverticalaxis.

Flipping is a similar augmentation as rotation, however, it produces mirrorimages.Aparticularfeaturesuchastheheadof apersoneitherstayson top, on the left, on the right, or at the bottom of the image.

b) Rotation:

Rotation is an augmentation that is commonly performed at 90-degree anglesbutcanevenhappenatsmallerorminuteanglesiftheneedformore data is great. For rotation, the background color is commonly fixed so that it can blend when the image is rotated. Otherwise, the model can assume the background change is a distinct feature. This works best when the background is the same in all rotated images.

StellathePuppysittingonacarseatStella thePuppyrotated90 degrees.

Specific features move in rotations. For example, the head of a person will be rotated 10, 22.7, or -8 degrees. However, rotation does not change the orientation of the feature and will not produce mirror images like flips. This helps models not consider the angle to be a distinct feature of the human.

c) Translation:

Translation of an image means shifting the main object in the image in various directions. For example, consider a person in the canter with all their parts visible in the frame and take it as a base image. Next, shift the person to one corner with the legs cut from the bottom as one translated image.

StellathePuppysittingonacarseat Stella thePuppytranslatedandcroppedsoshe's onlypartlyvisible.

d) Scaling:

Scaling provides more diversity in the training data of a machine learning model. Scaling the image will ensure that the object is recognized by the network regardlessof howzoomedin oroutthe image is. Sometimes the object istinyin the center. Sometimes, the object is zoomed in the image and even cropped at some parts.

Stella thePuppy sitting onacarseat Stella thePuppyscaleduptobeeven largerthan sheis in**reallife.**

e) Salt andPepperNoiseAddition

Salt and pepper noise addition is the addition of black and white dots (looking like salt and pepper) to the image. This simulates dust and imperfections in real photos. Even if the cameraof thephotographeris blurryorhasspots on it, the image would be better recognized by the model. The training data set is augmented to train the model with more realistic images.

StellathePuppysittingonacarseat StellathePuppywithSaltandPeppernoiseadded totheimage

BenefitsofDataAugmentationinaCNN

- Prediction improvement in a model becomes more accurate because DataAugmentationhelpsinrecognizingsamplesthemodelhasnever seen before.
- There is enough data for the model to understand and train all the availableparameters.Thiscanbeessentialinapplicationswheredata collection is difficult.
- HelpspreventthemodelfromoverfittingduetoDataAugmentation creating more variety in the data.
- Can save timeinareaswherecollectingmoredata istime-consuming.
- Can reducethecostrequiredforcollectingavarietyofdataifdata collection is costly.

DrawbacksofDataAugmentation:

Data Augmentation is not useful when the variety required by the application cannot be artificially generated. For example, if one were training a bird recognition model and the training data contained only red birds. The training data could be augmented by generating pictures with the color of the bird varied.

However, the artificial augmentation method may not capture the realisticcolor details of birds when there is not enough variety of data to start with. For example, if the augmentation method simply varied red for blue or green, etc. Realistic non-red birds may have more complex color variations and the model may fail to recognize the color. Having sufficient data is still important if one wants Data Augmentation to work properly.

UNIT-III

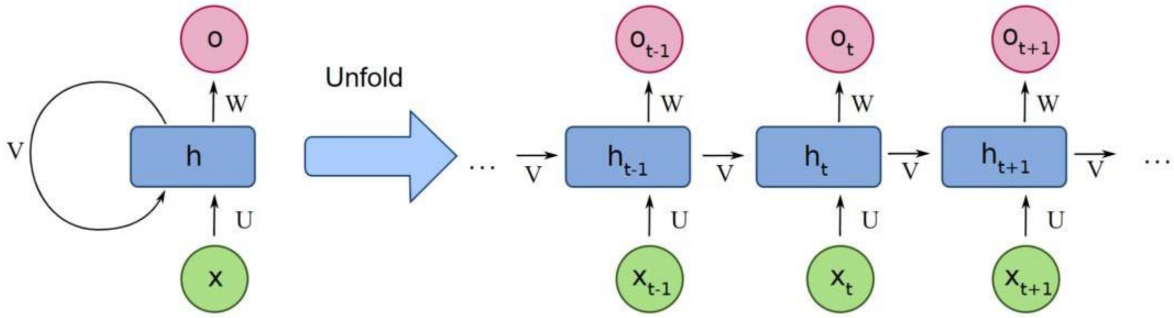
RECURRENT NEURAL NETWORK (RNN): Introduction to RNNs and their applications in sequential data analysis, Back propagation through time (BPTT), Vanishing Gradient Problem, gradient clipping Long Short-Term Memory (LSTM) Networks, Gated Recurrent Units, Bidirectional LSTMs, Bidirectional RNNs.

IntroductiontoRNNsandtheirapplicationsin sequentialdataanalysis:

RecurrentNeuralNetwork (RNN) worksbetterthanasimpleneural network when data is sequential like Time-Series data and text data.

ADeepLearningapproachformodellingsequentialdataisRNN:

RNNs were the standard suggestion for working with sequential data beforetheadventofattentionmodels.Specificparametersforeach element of the sequence may be required by a deep feedforward model. It may also be unable to generalize to variable-length sequences.



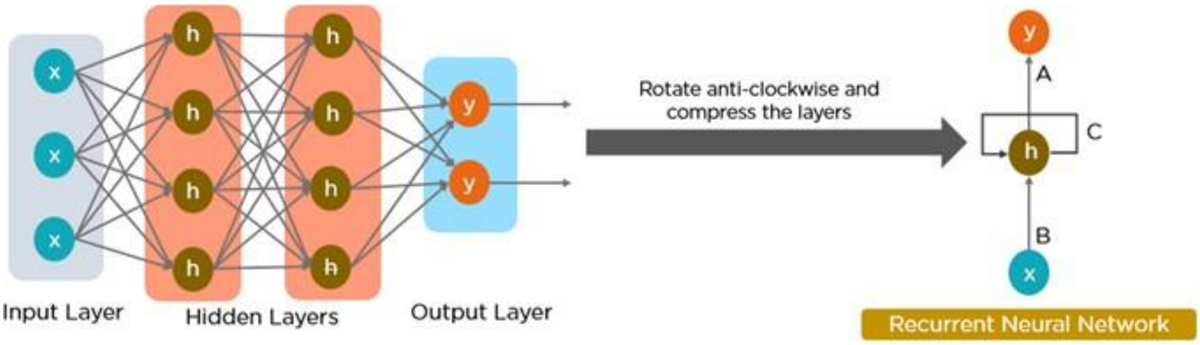
Recurrent Neural Networks use the same weights for each elementof the sequence, decreasing the number of parameters and allowing the model to generalize to sequences of varying lengths. RNNs generalize to structured data other than sequential data, such as geographical or graphical data, because of its design.

Recurrent neural networks, like many other deep learningtechniques, are relatively old. They were first developed in the 1980s, but we didn't appreciate their full potential until lately. The advent of long short- term memory (LSTM) in the 1990s, combined with an increase in computational power and the vast amounts of data that we now have todeal with, has really pushed RNNs to the forefront.

WhatisaRecurrentNeuralNetwork(RNN)?

Neural networks imitate the function of the human brain in the fieldsof AI, machine learning, and deep learning, allowing computer programs to recognize patterns and solve common issues.

RNNs are a type of neural network that can be used to model sequence data. RNNs, which are formed from feedforward networks, are similartohumanbrainsintheirbehaviour.Simplysaid,recurrentneural networkscananticipatesequentialdatainawaythatotheralgorithmscan't.

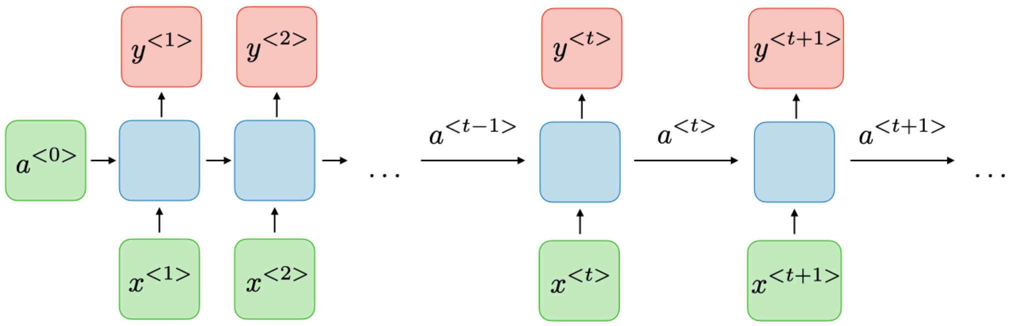


All of the inputs and outputs in standard neural networks are independent of one another, however in some circumstances, such aswhen predicting the next word of a phrase, the prior words are necessary, and so the previous words must be remembered. As a result, RNN was created, which used a Hidden Layer to overcome the problem. The most important component of RNN is the Hidden state, which remembersspecific information about a sequence.

RNNs have a Memory that stores all information about the calculations. It employs the same settings for each input since it produces the same outcome by performing the same task on all inputs or hidden layers.

TheArchitectureofaTraditionalRNN

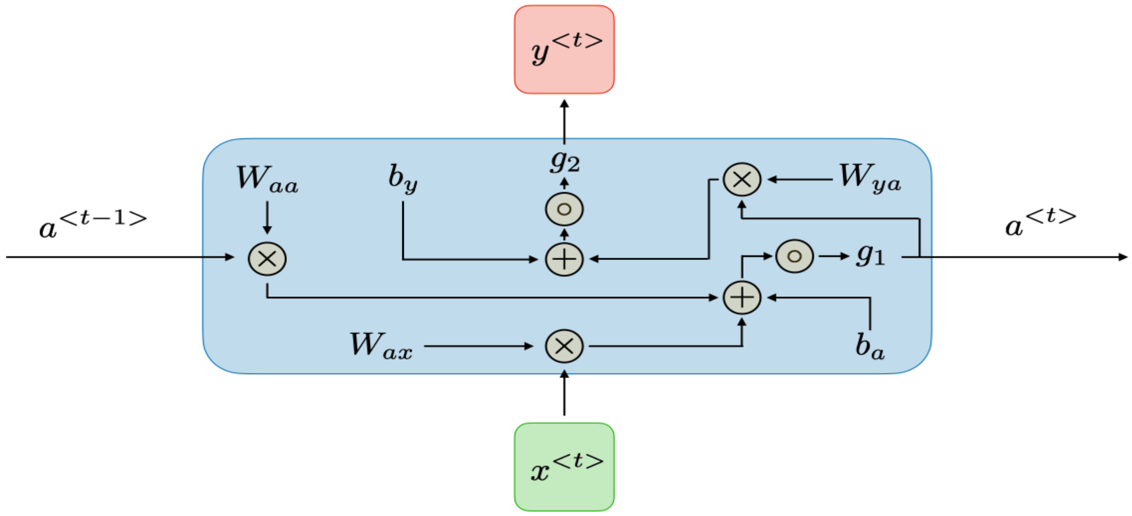
RNNs are a type of neural network that has hidden states and allows past outputs to be used as inputs. They usually go like this:



For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.



RNN architecture can vary depending on the problem you're trying to solve. From those with a single input and output to those with many (with variations between).

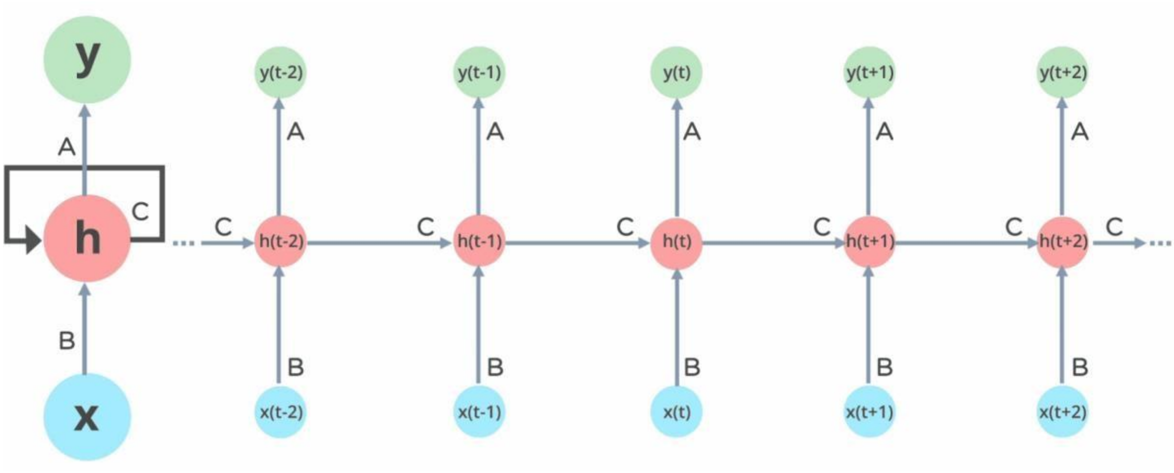
BelowaresomeexamplesofRNNarchitectures.

- **One To One:**There is only one pair here. A one-to-one architectureis used in traditional neural networks.
- **One To Many:**A single input in a one-to-many network might resultin numerous outputs. One too many networks are used in the production of music, for example.

- **Many To One:** In this scenario, a single output is produced by combining many inputs from distinct time steps. Sentiment analysis andemotion identification usesuchnetworks,in which theclass label is determined by a sequence of words.
- **Many To Many:**Formany tomany,therearenumerousoptions. Two inputs yield three outputs. Machine translation systems, such as English to French or vice versa translation systems, use many to many networks.

HowdoesRecurrentNeuralNetworkswork?

The information in recurrent neural networks cycles through a loop to the middle-hidden layer.



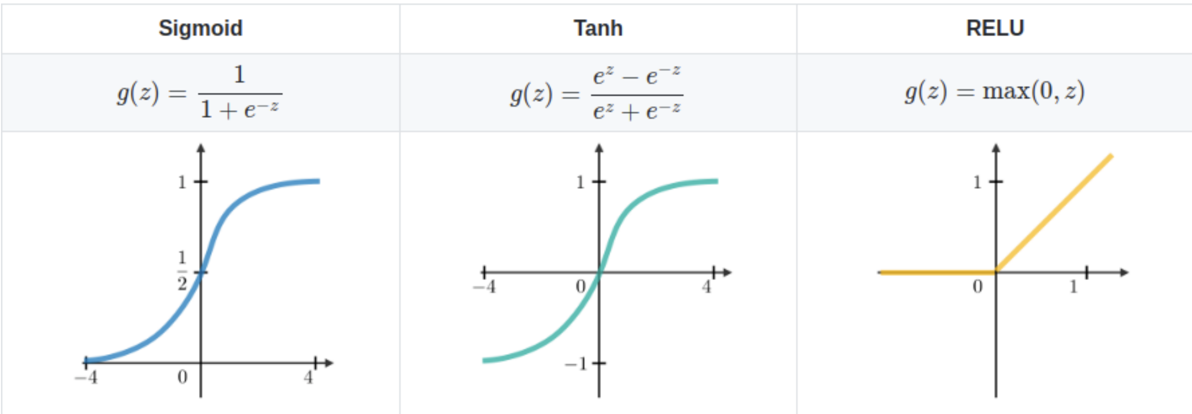
The input layer **x**receives and processes the neural network's input before passing it on to the middle layer.

Multiple hidden layers can be found in the middle layer **h**, each with its own activation functions,weights,and biases.Youcanutilizearecurrent neural network if the various parameters of different hidden layers are not impacted by the preceding layer, i.e. There is no memory in the neural network.

The different activation functions, weights, and biases will be standardized by the Recurrent Neural Network, ensuring that each hidden layer has the same characteristics. Rather than constructing numerous hidden layers, it will create only one and loop over it as many times as necessary.

CommonActivationFunctions:

A neuron's activation function dictates whether it should be turned on or off. Nonlinear functions usually transform a neuron's output to a number between 0 and 1 or -1 and 1.



Thefollowingaresomeofthemostcommonlyutilizedfunctions:

- **Sigmoid:**Theformula **g(z)=1/(1+e^-z)**is usedtoexpress this.
- **Tanh:**Theformula **g(z)=(e^-z–e^-z)/(e^-z+e^-z)**isusedto express this.
- **ReLu:**The formula **g(z)=max(0,z)**is usedto express this.

ApplicationsofRNNNetworks:

1. MachineTranslation:

RNN can be used to build a deep learning model that can translatetext from one language to another without the need for human intervention. You can, for example, translate a text from your native language to English.

2. Text Creation:

RNNs can also be used to build a deep learning model for text generation. Based on the previous sequence of words/characters used in the text, a trained modellearns the likelihoodofoccurrenceofa word/character. A model can be trained at the character, n-gram, sentence, or paragraph level.

3. Captioningofimages:

The process of creating text that describes the content of an image is known as image captioning. The image's content can depict the object as well as the action of the object on the image. In the image below, for example,thetraineddeep learning modelusingRNNcandescribetheimage as "A lady in a green coat is reading a book under a tree.”

4. RecognitionofSpeech:

This is also known asAutomatic Speech Recognition (ASR), and it is capable of converting human speech into written or text format. Don't mix up speech recognition and voice recognition; speech recognition primarily focuses on converting voice data into text, whereas voice recognition identifies the user's voice.

Speech recognition technologies that are used on a daily basis by various users include Alexa, Cortana, Google Assistant, and Siri.

5. ForecastingofTimeSeries:

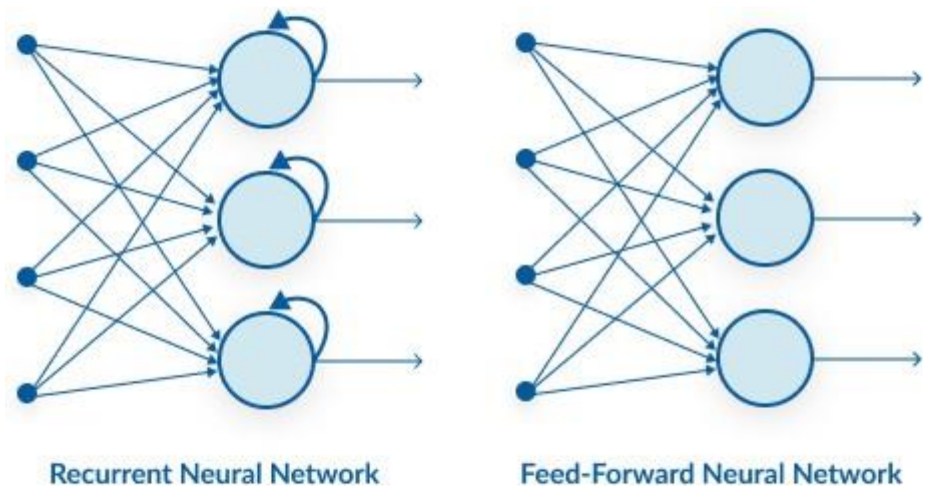
After being trained on historical time-stamped data, an RNN can be used to create a time series prediction model that predicts the future outcome. The stock market is a good example.

For example, Stock market data can be used to build a machine learning model that can forecast future stock prices based on what the model learns from historical data. This can assist investors in making data-driven investment decisions.

RecurrentNeuralNetworkVsFeedforwardNeuralNetwork:

A feed-forward neural network has only one route ofinformation flow: from the input layer to the output layer, passing through the hidden layers. The data flows across the network in a straight route, never going through the same node twice.

The information flow between an RNN and a feed-forward neural network is depicted in the two figures below.



Feed-forward neural networks are poor predictions of what will happen next because theyhave no memoryof the information theyreceive. Because it simply analyses the current input, a feed-forward network hasno idea of temporal order. Apart from its training, it has no memory of what transpired in the past.

The information is in an RNN cycle via a loop. Before making a judgment, it evaluates the current input as well as what it has learned from past inputs. A recurrent neural network, on the other hand, may recall due to internal memory. It produces output, copies it, and then returns it to the network.

BackpropagationThroughTime-RNN:

Backpropagation is a training algorithm that we use for training neural networks. When preparing a neural network, we are tuning the network's weights to minimize the error concerning the available actual values with the help of the Backpropagation algorithm. Backpropagation is a supervised learning algorithm as we find errors concerning already given values.

The backpropagation training algorithm aims to modify the weights of a neural network to minimize the error of the network results compared to some expected output in response to corresponding inputs.

ThegeneralalgorithmofBackpropagationisasfollows:

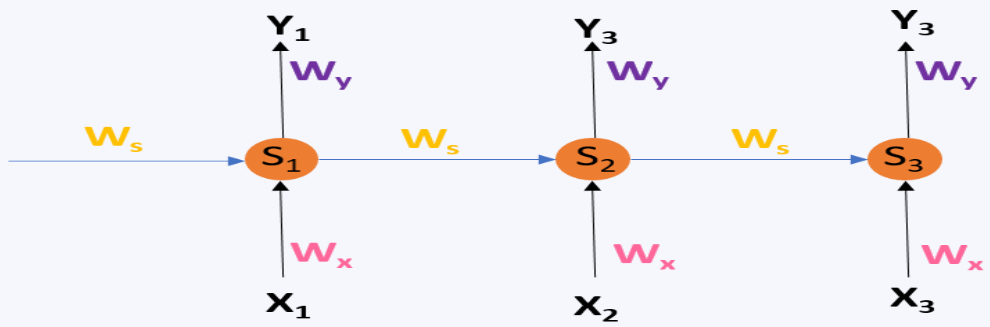
1. We first train input data and propagate it through the network to get an output.
2. Compare the predicted outcomes to the expected results and calculate the error.
3. Then,wecalculatethederivativesoftheerrorconcerningthenetwork weights.
4. We use these calculated derivatives to adjust the weights to minimize the error.
5. Repeattheprocessuntiltheerrorisminimized.

In simple words, Backpropagation is an algorithm where the informationof cost function is passed on through the neural network in the backward direction. The Backpropagation training algorithm is ideal for training feed- forward neural networks on fixed-sized input-output pairs.

UnrollingTheRecurrentNeuralNetwork

Recurrent Neural Network deals with sequential data. RNN predicts outputs using not only the current inputs but also by considering those that occurred before it. In other words, the current outcome depends on the current production and a memory element (which evaluates the past inputs).

ThebelowfiguredepictsthearchitectureofRNN.



We use Backpropagation for training such networks with a slight change. We don't independently train the network at a specific time "t." We train it at aparticulartime"t"aswellasallthathashappenedbeforetime"t"liket-1,t-2,t-3.

S_1, S_2, $S_3$are the hidden states at time t_1, t_2, t_3, respectively, andWs is theassociated weight matrix.

x_1, x_2, $x_3$are the inputs at time t_1, t_2, t_3, respectively, and $W_x$is the associated weight matrix.

Y_1, Y_2, $Y_3$are the outcomes at time t_1, t_2, t_3, respectively, and $W_y$is the associated weight matrix.

At time t_0, we feed input x0 to the network and output y0. At time t1, we provideinputx$_1$tothenetworkandreceiveanoutputy1.Fromthefigure,wecan see that to calculate the outcome. The network uses input x and the cell state from the previous timestamp. To calculate specific Hidden state and output at each step, here is the formula:

$$S_t = g_1(W_x x_t + W_s S_{t-1})$$

$$Y_t = g_2(W_Y S_t)$$

To calculate the error, we take the output and calculate its errorconcerning the actual result, but we have multiple outputs at each timestamp.Thus, the regular Backpropagation won't work here. Therefore, we modify thisalgorithm and call the new algorithm as Backpropagation through time.

BackpropagationThroughTime

Ws, Wx, and Wy do not change across the timestamps, which means thatfor all inputs in a sequence, the values of these weights are the same.

Theerrorfunctionisdefinedas:

$$E_t = (d_t - Y_t)^2$$

Thepointstoconsiderare:

Whatisthetotallossforthisnetwork?

Howdoweupdatetheweights,Ws,Wx,andWy?

The total loss we have to calculate is the sum in overall timestamps, i.e., E0+E1+E2+E3+...Now tocalculatetheerrorgradientconcerning Ws,Wx,andWy. It is relatively easy to calculate the loss derivative concerning Wy as the derivative only depends on the current timestamp values.

Formula:

$$\frac{\delta E_N}{\delta W_Y} = \frac{\delta E_N}{\delta Y_N} . \frac{\delta Y_N}{\delta W_Y}$$

ThencalculatingthederivativeoflossconcerningWsandWx,becomescomplex.

Formula:

$$\frac{\delta E}{\delta W_S} = \frac{\delta E}{\delta s_3} . \frac{\delta s_3}{\delta W_S}$$

The value of $s_3$depends on s_2, which is a function of W_s. Therefore, we cannot calculate the derivative of s_3, taking $s_2$as constant. In RNN networks, the derivative has two parts, implicit and explicit. We assume all other inputs as constant in the explicit part, whereas we sum over all the indirect paths in the implicit part.

Therefore we calculate the derivative as :

$$\frac{\delta E_3}{\delta W_s} = \frac{\delta E_3}{\delta Y_3} . \frac{\delta Y_3}{\delta S_3} . \frac{\delta S_3}{\delta W_s} + \frac{\delta E_3}{\delta Y_3} . \frac{\delta Y_3}{\delta S_3} . \frac{\delta S_3}{\delta S_2} . \frac{\delta S_2}{\delta W_S} + \frac{\delta E_3}{\delta Y_3} . \frac{\delta Y_3}{\delta S_3} . \frac{\delta S_3}{\delta S_2} . \frac{\delta S_2}{\delta S_1} . \frac{\delta S_1}{\delta W_s}$$

Thegeneralexpressioncanbewrittenas:

$$\frac{\delta E_N}{\delta W_S} = \sum_{i=1}^{N} \frac{\delta E_N}{\delta Y_N} . \frac{\delta Y_N}{\delta S_i} . \frac{\delta S_i}{\delta W_S}$$

Similarly,forWx,itcanbewrittenas:

$$\frac{\delta E_N}{\delta W_X} = \sum_{i=1}^{N} \frac{\delta E_N}{\delta Y_N} . \frac{\delta Y_N}{\delta S_i} . \frac{\delta S_i}{\delta W_X}$$

Now that we have calculated allthree derivatives,we can easilyupdate the weights. This algorithm is known as Backpropagation through time (BPTT), aswe used values across all the timestamps to calculate the gradients.

Thealgorithmataglance:

- Wefeedasequenceoftimestampsofinputandoutputpairstothe network.
- Then,weunrollthenetworkthencalculateandaccumulateerrors across each timestamp.

- Finally,werollupthenetworkandupdateweights.
- Repeattheprocess.

Limitationsof BPTT:

BPTT has difficulty with local optima. Local optima are a more significant issue with recurrent neural networks than feed-forward neural networks. The recurrent feedback in such networks creates chaotic responses in the error surface,which causeslocal optima to occur frequentlyandinthe wrong locations on the error surface.

When using BPTT in RNN, we face problems such as exploding gradient and vanishing gradient. To avoid issues such as exploding gradient, we use a gradient clipping method to check if the gradient value is greater than the threshold or not at each timestamp. If it is, we normalize it. This helps to tackle exploding gradient.

We can use BPTT up to a limited number of steps like 8 or 10. If we backpropagate further, the gradient becomes too negligible and is a Vanishing gradient problem. To avoid the vanishing gradient problem, some of the possible solutions are:

- Using ReLU activation function in place of tanh or sigmoid activation function.
- Properinitializingthe weightmatrixcanreducetheeffectofvanishing gradients. For example, using an identity matrix helps us tackle this problem.
- UsinggatedcellssuchasLSTMorGRUs.

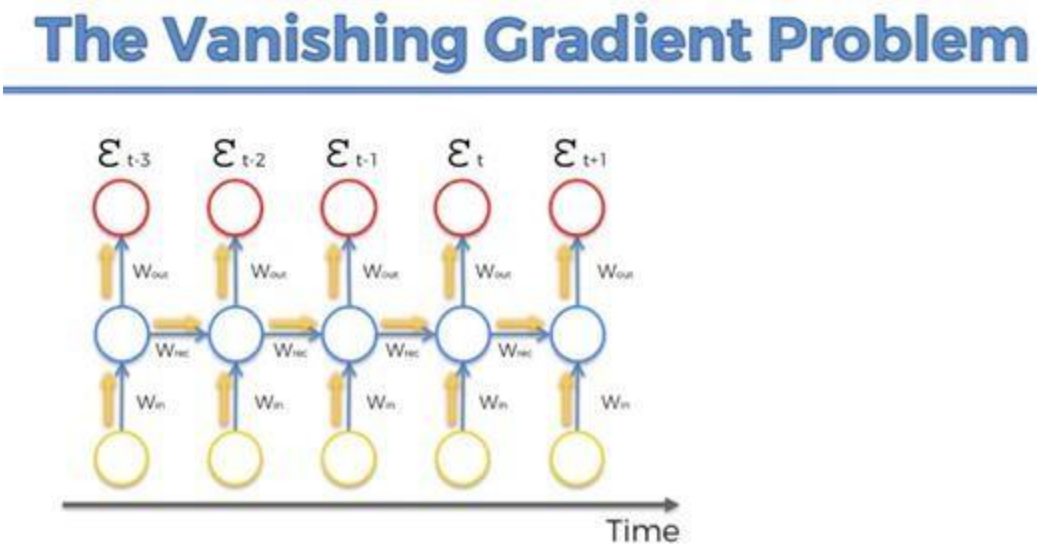
VanishingGradientProblem:

Thegradient descentalgorithmfindstheglobal minimumof thecostfunctionthat is going to be an optimal setup for the network. Information travels through the neural network from input neurons to the output neurons, while the error is calculated and propagated back through the network to update the weights.

ItworksquitesimilarlyforRNNs,butadditionaltasksinclude:

- Firstly, information travels through time in RNNs, which means that informationfromprevioustimepointsisusedasinputforthenexttime points.
- Secondly,wecancalculatethecostfunction,ortheerror,ateachtimepoint.

Basically, during the training, your cost function compares your outcomes (red circles on the image below) to your desired output. As a result, you have these values throughout the time series, for every single one of these red circles.



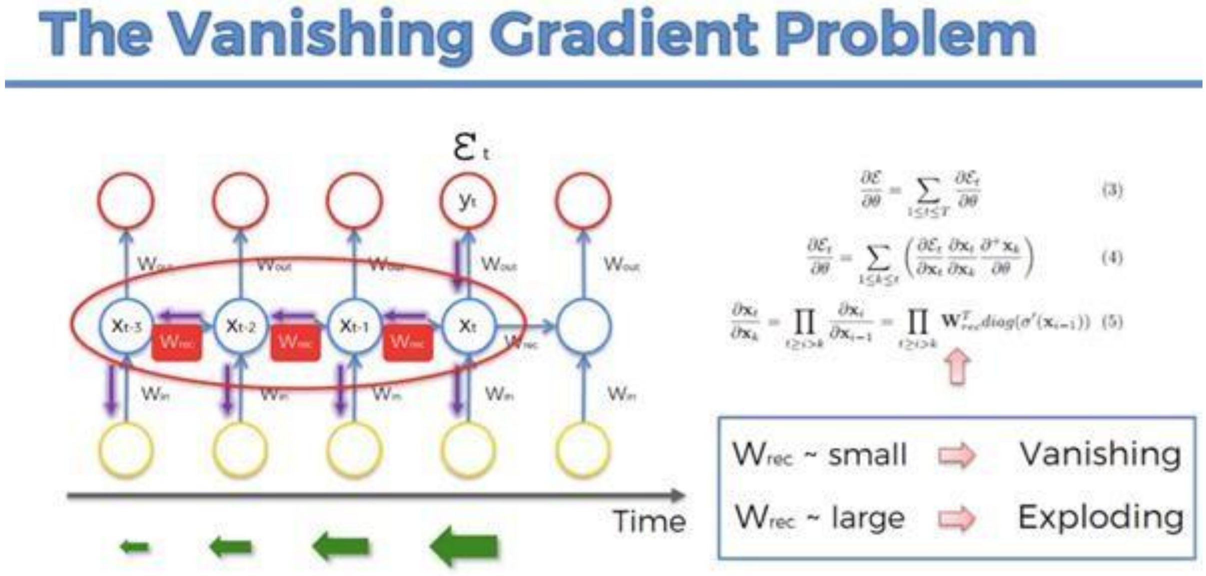
The focus is on one error term e_t. We calculate the cost function e_t and then propagate the cost function back through the network because of the need to updatethe weights.

Essentially, every single neuron that participated in the calculation of the output, associated with this cost function, should have its weight updated in order to minimize that error. And the thing with RNNs is that it's not just the neurons directly belowthisoutputlayer thatcontributedbutall of theneuronsfarbackintime.So, you have to propagate all the way back through time to these neurons.

The problem relates to updating w_{rec} (weight recurring) – the weight that isused to connect the hidden layers to themselves in the unrolled temporal loop.

For instance, to get from x_{t-3} to x_{t-2} we multiply x_{t-3} by wrec. Then, to get from x_{t-2} to x_{t-1} we again multiply x_{t-2} by w_{rec}. So, we multiply with the same exact weight multipletimes, andthisiswherethe problemarises:when we multiplysomethingbya small number, the value decreases very quickly.

As we know, weights are assigned at the start of the neural network with the random values, which are close to zero, and from there the network trains them up. But, when you start with wrec close to zero and multiply x_t, x_{t-1}, x_{t-2}, x_{t-3}, … by this value, your gradient becomes less and less with each multiplication.



Whatdoesthismeanforthenetwork?

The lower the gradient is, the harder it is for the network to update the weights and the longer it takes to get to the final result.

For instance, 1000 epochs might be enough to get the final weight for the time point t, but insufficient for training the weights for the time point t-3 due to a verylow gradient at this point. However, the problem is not only that half of the network is not trained properly.

The output of the earlier layers is used as the input for the further layers. Thus, the training for the time point t is happening all along based on inputs that are coming fromuntrained layers. So, because of the vanishing gradient, the whole network is not being trained properly.

To sum up, if wrec is small, you have vanishing gradient problem, and if wrec is large, you have exploding gradient problem. For the vanishing gradient problem,the further you go through the network, the lower your gradient is and the harder it is to train the weights, which has a domino effect on all of the further weightsthroughout the network.

That was the main roadblock to using Recurrent Neural Networks. However, the possible solutions to this problem are as follows:

Solutionstothevanishinggradientproblem

Incaseofexplodinggradient,youcan:

- Stopbackpropagatingafteracertainpoint,whichisusuallynotoptimal because not all of the weights get updated.
- Penalizeorartificiallyreducegradient.
- Putamaximumlimitonagradient.

Incaseofvanishinggradient,youcan:

- Initializeweightssothatthepotentialforvanishinggradientisminimized.
- HaveEchoStateNetworksthataredesignedtosolvethevanishinggradient problem.
- HaveLongShort-TermMemoryNetworks(LSTMs).

GradientclippingLongShort-TermMemory(LSTM)Networks:

Training a neural network can become unstable given the choice of error function, learning rate, or even the scale of the target variable. Large updates to weightsduringtrainingcancausea numericaloverfloworunderflowoften referred to as “Exploding Gradients.”

The problem of exploding gradients is more common with recurrent neural networks, such as LSTMs given the accumulation ofgradients unrolled overhundreds of input time steps.

A common and relatively easy solution to the exploding gradients problem isto change the derivative of the error before propagating it backward through the network and using it to update the weights. Two approaches include rescaling the gradients given a chosen vector norm and clipping gradient values that exceed a preferred range. Together, these methods are referred to as “Gradient Clipping.”

- Trainingneuralnetworkscanbecomeunstable,leadingtoanumerical overflow or underflow referred to as exploding gradients.

- The training process can be made stable by changing the error gradients either by scaling the vector norm or clipping gradient values to a range.
- How to update anMLP model for aregression predictive modeling problem with exploding gradients to have a stable training process using gradient clipping methods?

ExplodingGradientsandClipping

Neural networks are trained using the stochastic gradient descentoptimization algorithm. This requires first the estimation of the loss on one or more training examples, then the calculation of the derivative of the loss, which is propagated backward through the network in order to update the weights. Weights are updated using a fraction of the back propagated error controlled by the "LearningRate".

It is possible for the updates to the weights to be so large that the weights either overflow or underflow their numerical precision. In practice, the weights can take on the value of an "*NaN*" or "*Inf*" when they overflow or underflow and for practical purposes the network will be useless from that point forward, forever predicting NaN values as signals flow through the invalid weights.

The difficulty that arises is that when the parameter gradient is very large, a gradient descent parameter update could throw the parameters very far, into aregion where the objective function is larger, undoing much of the work that hadbeen done to reach the current solution.

The underflow or overflowof weights generally refers to asan instability of the network training process and is known by the name "*exploding gradients*" as the unstable training process causes the network to fail to train in such a way that the model is essentially useless.

In a given neural network, such as a Convolutional Neural Network or Multilayer Perceptron, this can happen due to a poor choice of configuration. Some examples include:

- Poorchoiceoflearningratethatresultsinlargeweight updates.

- Poor choice of data preparation, allowing large differences in the target variable.
- Poorchoiceoflossfunction,allowingthecalculationof largeerrorvalues.

Exploding gradients is also a problem in recurrent neural networks such as the LongShort-TermMemorynetworkgiventheaccumulationoferrorgradientsin the unrolled recurrent structure.

Exploding gradients can be avoided in general by careful configuration of the networkmodel,suchaschoiceofsmalllearningrate,scaledtargetvariables,and astandard loss function. Nevertheless, exploding gradients may still be an issue with recurrent networks with a large number of input time steps.

One difficulty when training LSTM with the full gradient is that the derivatives sometimes become excessively large, leading to numerical problems. To prevent this, [we] clipped the derivative of the loss with respect to the network inputs to the LSTM layers (before the sigmoid and tanh functions are applied) to lie within a predefined range.

A common solution to exploding gradients is to change the error derivative before propagating it backward through the network and using it to update the weights. By rescaling the error derivative, the updates to the weights will also be rescaled, dramatically decreasing the likelihood of an overflow or underflow.

Thereare two mainmethodsforupdatingtheerrorderivativeasfollows:

- GradientScaling.
- GradientClipping.

Gradient scaling involves normalizing the error gradient vector such thatvector norm (magnitude) equals a defined value, such as 1.0. One simplemechanism to deal with a sudden increase in the norm of the gradients is to rescale them whenever they go over a threshold

Gradient clipping involves forcing the gradient values (element-wise) to a specific minimum or maximum value if the gradient exceeded an expected range.Together, these methods are often simply referred to as “*gradient clipping*.”

When the traditional gradient descent algorithm proposes to make a verylarge step, the gradient clipping heuristic intervenes to reduce the step size to be small enough that it is less likely to go outside the region where the gradientindicates the direction of approximately steepest descent. It is a method that only addresses the numerical stability of training deep neural network models and does not offer any general improvement in performance.

The value for the gradient vector norm or preferred range can be configuredby trial and error, by using common values used in the literature or by first observing common vector norms or ranges via experimentation and then choosing a sensible value.

Experimental analysis reveals that for a given task and model size, training is not very sensitive to this [gradient norm] hyperparameter and the algorithm behaves well even for rather small thresholds.

It is common to use the same gradient clipping configuration for all layers in the network. Nevertheless, there are examples where a larger range of error gradients are permitted in the output layer compared to hidden layers.

The output derivatives […]were clipped in the range [−100, 100], and the LSTM derivatives were clipped in the range [−10, 10]. Clipping the output gradients proved vital for numerical stability; even so, the networks sometimes had numerical problems late on in training, after they had started overfitting on the training data.

GatedRecurrentUnit(GRU):

A Gated Recurrent Unit (GRU) is a Recurrent Neural Network (RNN) architecture type. Like other RNNs, a GRU can process sequential data such as time series, natural language, and speech. The main difference between a GRU and other RNN architectures, such as the Long Short-Term Memory (LSTM) network, is how the network handles information flow through time.

Example:

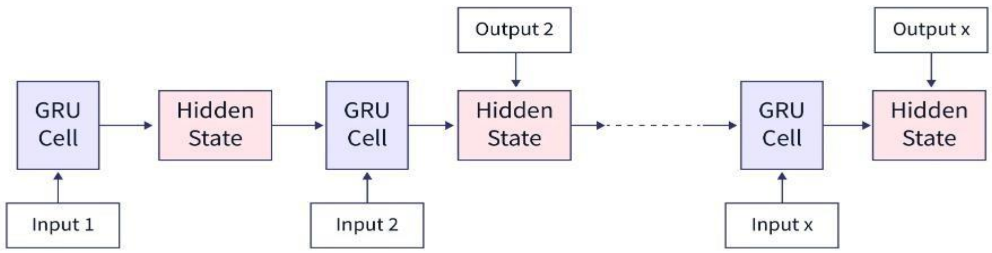
"Mymom gavemea bicycleonmy birthdaybecauseshe knew thatI wanted to go biking with my friends."

As it can be observed from the above sentence, words that affect each other can be further apart. For example, "bicycle" and "go biking" are closely related but are placed further apart in the sentence. An RNN network finds tracking the state with such a long context difficult. It needs to find out what information is important. However, a GRU cell greatly alleviates this problem.

GRUnetworkwasinventedin2014.Itsolvesproblemsinvolvinglongsequenceswith contextsplacedfurtherapart,liketheabovebikingexample.Thisispossiblebecauseofhow the GRU cell in the GRU architecture is built.

UnderstandingtheGRUCell:

The GRU cell is the basic building block of a GRU network. It comprises three main components: an update gate, a reset gate, and a candidate hidden state.



One of the key advantages of the GRU cell is its simplicity. Since it has fewer parameters than a long short-term memory (LSTM) cell, it is faster to train and run and less prone to overfitting.

Additionally, one thing to remember is that the GRU cell architecture is simple, the cell itself is a black box, and the final decision on how much we should consider the past state and how much should be forgotten is taken by this GRU cell.

GRUvsLSTM

	GRU	LSTM
Structure	Simplerstructurewithtwogates (update and reset gate)	More complexstructurewiththree gates (input, forget, and output gate)
Parameters	Fewer parameters (3 weight matrices)	Moreparameters (4weight matrices)

	GRU	LSTM
Training	Fastertotrain	Slow to train
Space Complexity	Inmostcases,GRUtendtouse fewermemoryresourcesduetoits simpler structure and fewer parameters,thusbettersuitedfor largedatasetsorsequences.	LSTMhasamorecomplexstructureand alargernumberofparameters,thusmight require more memory resources and couldbelesseffectiveforlargedatasets orsequences.
Performance	Generallyperformedsimilarlyto LSTMonmanytasks,butinsome cases,GRUhasbeenshownto outperformLSTMandviceversa. It'sbettertotrybothandseewhich worksbetterforyourdatasetand task.	LSTM generally performs well on many tasks but is more computationally expensive and requires more memory resources. LSTM has advantages over GRU in natural language understanding and machine translation tasks.

TheArchitectureofGRU

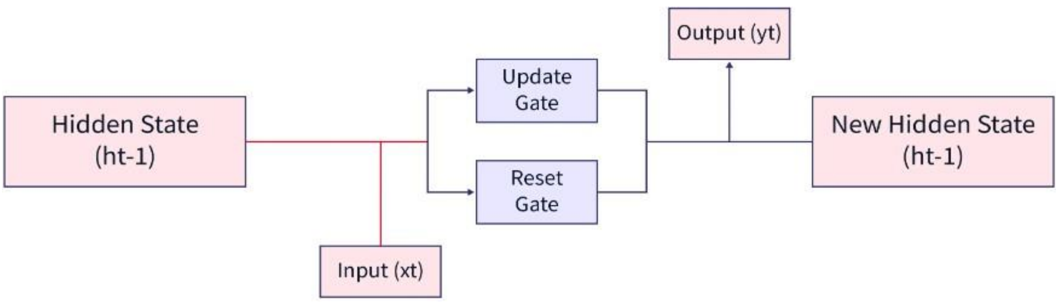
AGRUcellkeepstrackoftheimportantinformationmaintainedthroughoutthe network. A GRU network achieves this with the following two gates:

- ResetGate
- UpdateGate.

GivenbelowisthesimplestarchitecturalformofaGRUcell.AGRUcelltakestwo inputs:

1. Theprevioushiddenstate
2. Theinputinthecurrenttimestamp.

The cell combinestheseandpasses them through the update and reset gates. To get the output in the current timestep, we must pass this hidden state through a dense layer with softmax activation to predict the output. Doing so, a new hidden state is obtained and then passed on to the next time step.



Updategate

An update gate determines what current GRU cell will pass information to the next GRU cell. It helps in keeping **track of the most important information**.

Obtainingtheoutput oftheUpdateGateinaGRUcell:

The input to the update gate is the hidden layer at the previous timestep, $h_{(t-1)}$ and the current input (x_t). Both have their weights associated with them which are learned during the training process. Let us say that the weights associated with$h_{(t-1)}$ is$U_{(z)}$, and that of $x_t$is W_z. The output of the update gate $Z_t$is given by,

$$z_t=\sigma(W(z)x_t+U(z)h_{(t-1)}$$

Resetgate

A reset gate**identifies the unnecessary information** and decides what informationto be laid off from the GRU network. Simply put, it decides what information to delete atthe specific timestamp.

Obtainingtheoutput oftheResetGatein aGRUcell:

The input to the reset gate is the hidden layer at the previous timestep $h(t-1)$andthe current input xt. Both have their weights associated withthem whichare learned during thetrainingprocess.Letussaythattheweights associatedwith $h(t-1)$isUr,andthatof xt is W_r. The output of the update gate rt is given by,

$$r_t=\sigma(W_{(r)}x_t+U_{(r)}h_{(t-1)})$$

It is important to note that the weights associated with the hidden layer at the previous timestep and the current input are different for both gates. The values for these weights are learned during the training process.

HowDoesGRU Work?

Gated Recurrent Unit (GRU)networks process sequential data, such as time series or natural language, bypassing the hidden state from one time step to the next. The hidden state is a vector that captures the information from the past time steps relevant to the currenttimestep.ThemainideabehindaGRUistoallowthenetworktodecidewhat information from the last time step is relevant to the current time step and what information can be discarded.

CandidateHiddenState

A candidate's hidden state is calculated from the reset gate. This is used todetermine the information stored from the past. This is generally called the memory component in a GRU cell. It is calculated by,

$$h_{t'} = \tanh(W_x t + rt \odot U h_{t-1})$$

Here,W-weightassociatedwiththecurrentinput

r_t-Outputoftheresetgate

U-Weightassociatedwiththehiddenlayeroftheprevious timestep

h_t-Candidatehiddenstate.

Hidden state

The following formula gives the new hidden state and depends on the update gate and candidate hidden state.

$$h_t = z_t \odot h_{t-1} + (1-z_t) \odot h_{t'}$$

Here,z_t-OutputofupdategateKaTeXparseerror Expected'EOF'got''''atposition2: h'$_t$ -

Candidate hidden state

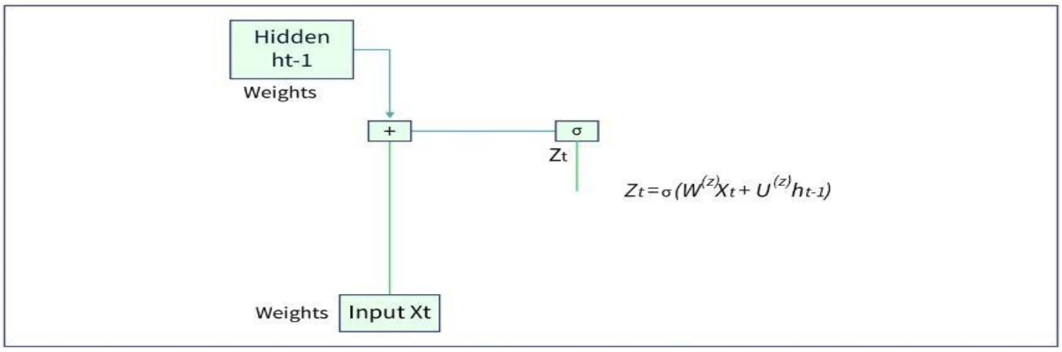
h_{t-1}-Hiddenstateattheprevious timestep

It can be observed that whenever $z_t$is 0, the information at the previously hidden layer gets forgotten. It is updated with the value of the new candidate hidden layer (as1−$z_t$willbe1).If ztis1,thentheinformationfromthepreviously hidden layerismaintained.This is how the most relevant information is passed from one state to the next.

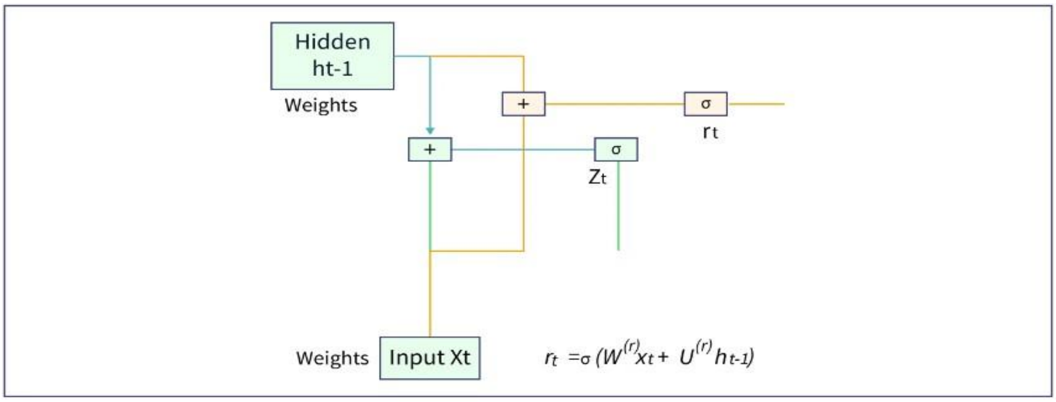
ForwardPropagationinaGRUCell

InaGatedRecurrentUnit(GRU)cell,theforwardpropagationprocessincludes several steps:

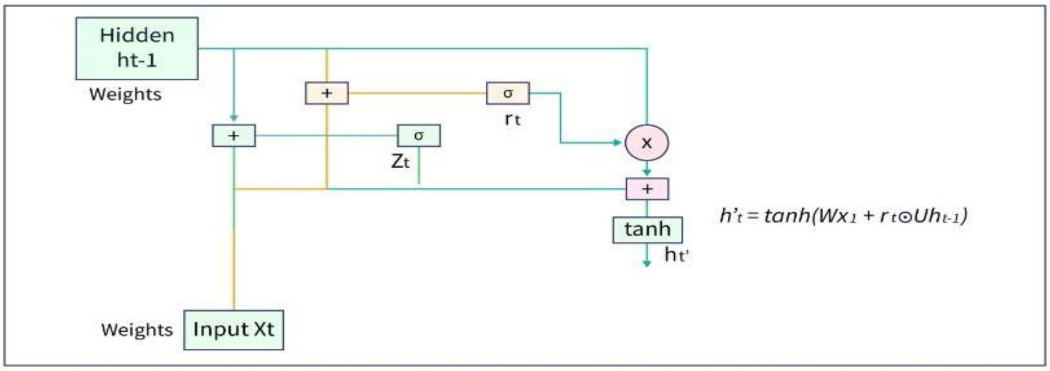
- Calculatetheoutput oftheupdategate(*zt*)usingtheupdategateformula:



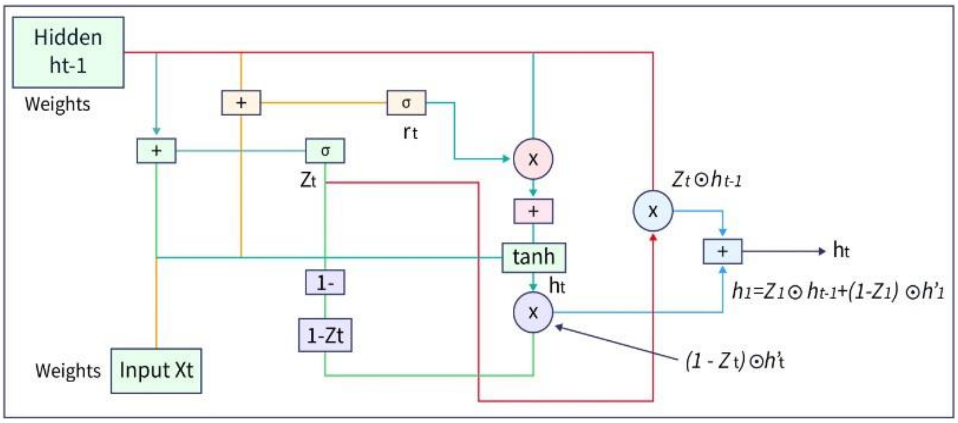
- Calculatetheoutputoftheresetgate(*rt*)usingtheresetgateformula:



- Calculatethecandidate'shiddenstate.



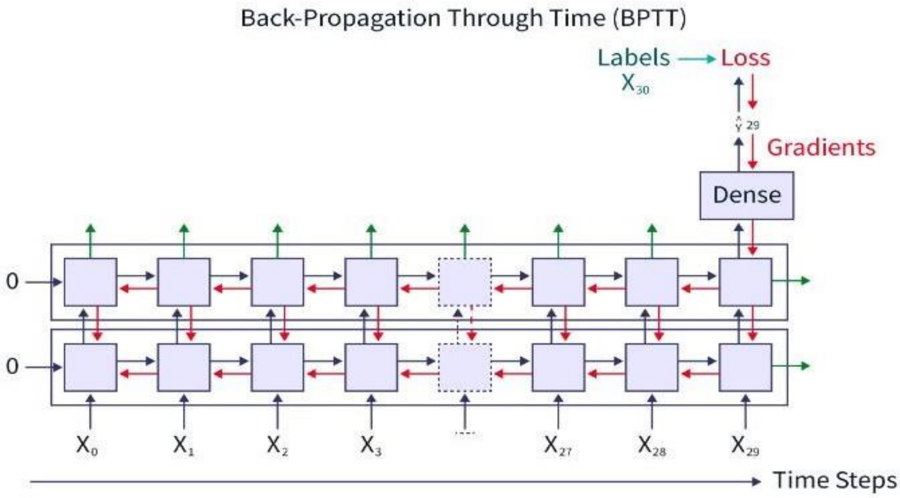
- Calculatethenewhiddenstate.



This is how forward propagation happens in a GRU cell at a GRU network. Next, the process of how the weights is learnt in a GRU network to make the right prediction have to be understood.

BackpropagationinaGRUCell

Let eachhiddenlayer(orangecolour)representa GRUcell.



In the above image, it is observed that whenever the network predicts wrongly, the network compares it with the original label, and the loss is then propagated throughout the network.Thishappensuntilalltheweights'valuesareidentifiedsothatthevalueof theloss function used to compute the loss is minimum. During this time, the weights and biases associated with the hidden layers and the input are fine-tuned.

AnalogybetweenLSTMandGRUintermsofarchitectureandperformance:

LSTM and GRU are two types of recurrent neural networks (RNNs) that can handle sequential data, such as text, speech, or video. They are designed to overcome the problem of vanishing or exploding gradients that affect the training of standard RNNs. However, they have different architectures and performance characteristics that make them suitable for different applications. In this article, you will learn about the differences and similarities between LSTM and GRU in terms of architecture and performance.

LSTMArchitecture

LSTM stands for long short-term memory, and it consists of a series of memory cells that can store and update information over long time steps. Each memory cell has three gates: an input gate, an output gate, and a forget gate. The input gate decides what information to add to the cell state, the output gate decides what information to output from the cell state, and the forget gate decides what information to discard from the cell state. The gates are learned by the network based on the input and the previous hidden state.

GRU Architecture

GRU standsfor gated recurrentunit, and it is asimplified versionof LSTM. It hasonly two gates: a reset gate and an update gate. The reset gate decides how much of the previous hidden state to keep, and the update gate decides how much of the new input to incorporate into the hidden state. The hidden state also acts as the cell state and theoutput, so there is no separate output gate. The GRU is easier to implement and requires fewer parameters than the LSTM.

PerformanceComparison

The performance of LSTM and GRU depends on the task, the data, and the hyperparameters. Generally, LSTM is more powerful and flexible than GRU, but it is also more complex and prone to overfitting. GRU is faster and more efficient than LSTM, but it may not capture long-term dependencies as well as LSTM. Some empirical studies have shownthatLSTMandGRUperformsimilarlyonmanynaturallanguageprocessingtasks, such as sentiment analysis, machine translation, and text generation. However, some tasks may benefit from the specific features of LSTM or GRU, such as image captioning, speech recognition, or video analysis.

SimilaritiesBetweenLSTMandGRU

Despite their differences, LSTM and GRU share some common characteristics that makethembotheffectiveRNNvariants.Theybothusegatestocontroltheinformationflow and to avoid the vanishing or exploding gradient problem. They both can learn long-term dependencies and capture sequential patterns in the data. They both can be stacked into multiple layers to increase the depth and complexity of the network.

They both can be combined with other neural network architectures, such as convolutional neural networks (CNNs) or attention mechanisms, to enhance their performance.

DifferencesBetweenLSTMandGRU

The main differences between LSTM and GRU lie in their architectures and their trade-offs. LSTM has more gates and more parameters than GRU, which gives it more flexibility and expressiveness, but also more computational cost and risk of overfitting. GRU has fewer gates and fewer parameters than LSTM, which makes it simpler and faster, but also less powerful and adaptable.

LSTM has a separate cell state and output, which allows it to store and output different information, while GRU has a single hidden state that serves both purposes, which may limit its capacity. LSTM and GRU may also have different sensitivities to the hyperparameters, such as the learning rate, the dropout rate, or the sequence length.

BidirectionalLSTM

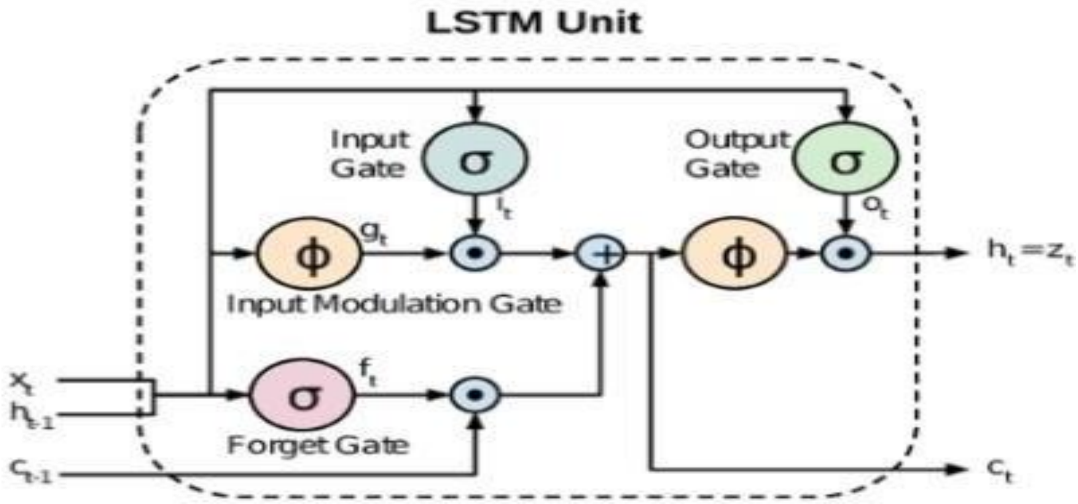
Introduction:

To understand the working of Bi-LSTM first, the working of the unit cell of LSTM and LSTM network has to be understood. LSTM stands for long short-term memory. In 1977, Hochretier and Schmidhuber introduced LSTM networks. These are the most commonly used recurrent neural networks.

NeedofLSTM

As the sequential data is better handled by recurrent neural networks, but sometimes it is also necessary to store the result of the previous data. For example, “I will play cricket” and “I can play cricket” are two different sentences with different meanings. The meaning of the sentence depends on a single word so, it is necessary to store the data of previous words. But no such memory is available in simple RNN. To solve this problem, LSTM is adopted.

<u>TheArchitectureoftheLSTMUnit</u>



TheLSTMunithasthreegates.

a) Input gate

First, the current state x(t) and previous hidden state h(t-1) are passed into the input gate, i.e., the second sigmoid function. The x(t) and h(t-1) values are transformed between0and1,where 0isimportant,and1is notimportant.Furthermore,thecurrent and hidden state information will be passed through the tanh function. The output from the tanh function will range from -1 to 1, and it will help to regulate the network. The output values generated from the activation functions are ready for point-by-point multiplication.

b) Forgetgate

The forget gate decides which information needs to be kept for further processing and which can be ignored. The hidden state h(t-1) and current input X(t) informationarepassedthroughthesigmoidfunction.Afterpassingthevaluesthrough thesigmoidfunction,itgeneratesvaluesbetween0and1thatconcludewhetherthe part of the previous output is necessary (by giving the output closer to 1).

c) Output gate

The output gate helps in deciding the value of the next hidden state. This state contains information on previous inputs. First, the current and previously hidden state values are passed into the third sigmoid function. Then the new cell state generated from the cell state is passed through the tanh function. Both these outputs aremultiplied point-by-point. Based upon the final value, the network decides which information the hidden state should carry. This hidden state is used for prediction.

Finally, the new cell state and the new hidden state are carried over to the next step. To conclude, the forget gate determines which relevant information from the prior steps is needed. The input gate decides what relevant information can be added fromthe current step, and the output gates finalize the next hidden state.

HowdoLSTMwork?

TheLengthyShortTermMemoryarchitecture wasinspiredbyanexaminationof error flow in current RNNs, which revealed that long time delays were inaccessible to existing designs due to backpropagated error, which either blows up or decays exponentially.

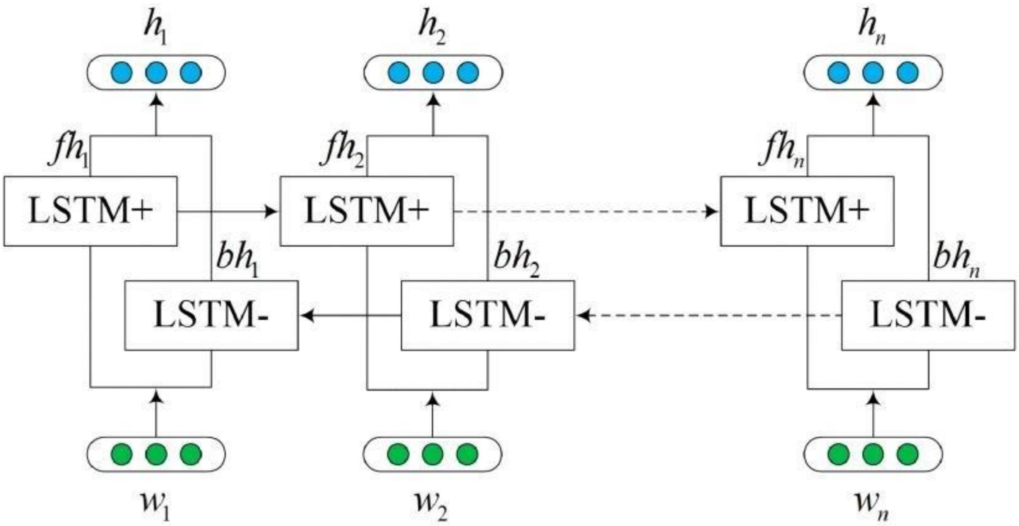
An LSTM layer is made up of memory blocks that are recurrently linked. These blocks can be thought of as a differentiable version of a digital computer's memory chips. Each one has recurrently connected memory cells as well as three multiplicative units – the input, output, and forget gates – that offer continuous analogs of the cells' write, read, and reset operations.

WhatisBi-LSTM?

Bidirectional LSTM networks function by presenting each training sequence forward and backward to two independent LSTM networks, both of which are coupled to the same output layer. This means that the Bi-LSTM contains comprehensive, sequential information about all points before and after each point in a particular sequence.

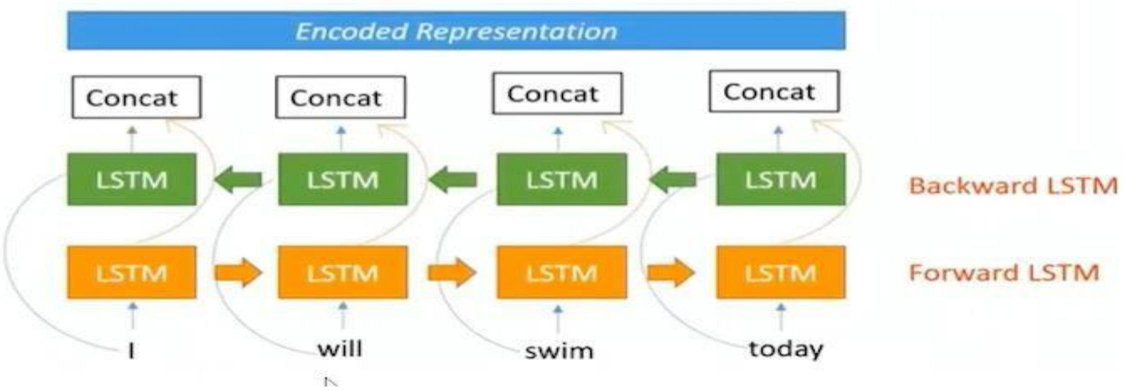
In other words, rather than encoding the sequence in the forward direction only, weencodeitinthebackwarddirectionaswellandconcatenatetheresultsfromboth forwardandbackwardLSTMateachtimestep.Theencodedrepresentationofeach word now understands the words before and after the specific word.

BelowisthebasicarchitectureofBi-LSTM.



WorkingofBi-LSTM:

Consider the sentence "I will swim today". The below image represents the encoded representation of the sentence in the Bi-LSTM network.



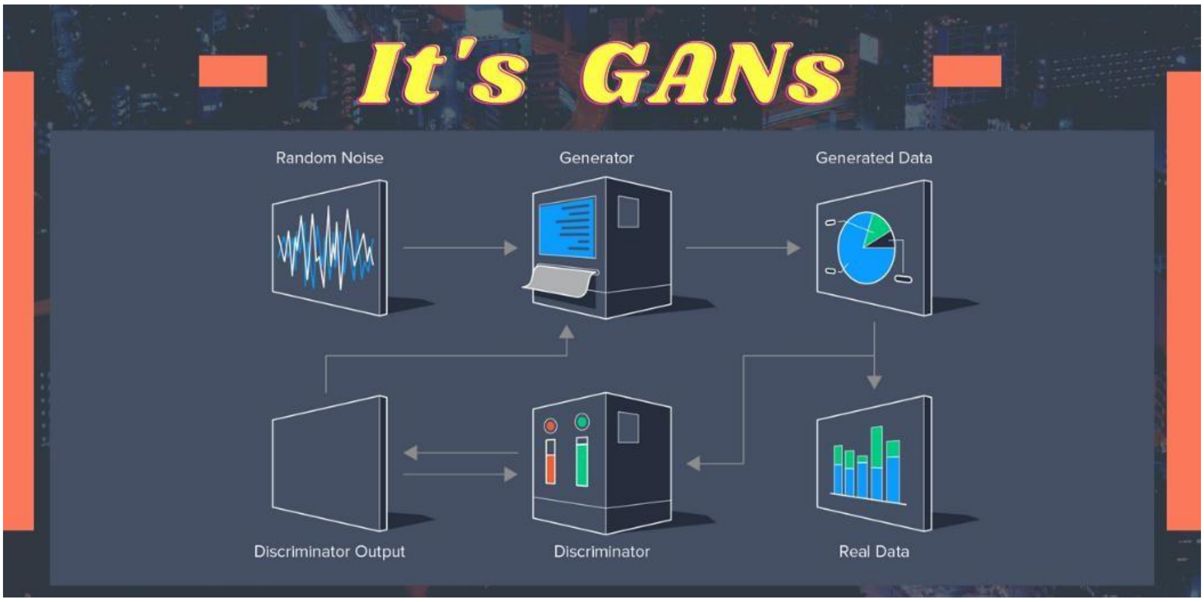
So, when forward LSTM occurs, "I" will be passed into the LSTM network at timet = 0, "will" at t = 1, "swim" at t = 2, and "today" at t = 3. In backward LSTM "today" will be passedinto the network at time t = 0, "swim" at t = 1, "will" at t = 2, and"I" at t = 3. In this way, both the results of forward and backward LSTM at each time step are calculated.

UNIT-IV

GENERATIVEADVERSARIALNETWORKS(GANS): Generativemodels,Conceptandprinciplesof GANs,Architecture of GANs (generator and discriminator networks), Comparison between discriminative and generative models, Generative Adversarial Networks (GANs), Applicationsof GANs

GenerativeAdversarialNetworksanditsmodels

Introduction:



Generative Adversarial Networks (GANs) were developed in 2014 by Ian Goodfellow and his teammates. GAN is basically an approach to generativemodeling that generates a new set of data based on training data that look like training data. GANs have two main blocks (two neural networks) which compete with each other and are able to capture, copy, and analyze the variations in a dataset.The two models are usually called Generator and Discriminator which we will coverin Components on GANs. The term GAN can be separated into three parts.

- **Generative** - To learn a generative model, which describes how data is generated in terms of a probabilistic model. In simple words, it explains how data is generated visually.

- **Adversarial** -Thetrainingofthemodelisdoneinanadversarialsetting.
- **Networks**-Usedeepneuralnetworksfortrainingpurposes.

The generator network takes random input (typically noise) and generates samples, such as images, text, or audio, that resemble the training data it wastrainedon.The goalof the generatoristo produce samples that areindistinguishable from real data.

The discriminator network, on the other hand, tries to distinguish between real and generated samples. It is trained with real samples from the training data and generated samples from the generator. The discriminator's objective is to correctly classify real data as real and generated data as fake.

The training process involves an adversarial gamebetweenthe generator and the discriminator. The generator aims to produce samples that fool the discriminator, while the discriminator tries to improve its ability to distinguish between real and generated data. This adversarial training pushes both networks to improve over time.

As training progresses, the generator becomes more adept at producing realistic samples, while the discriminator becomes more skilled at differentiating between real and generated data. Ideally, this process converges to a point where the generator is capable of generating high-quality samples that are difficult for the discriminator to distinguish from real data.

GANs have demonstrated impressive results in various domains, such as image synthesis, text generation, and even video generation. They have been used for tasks like generating realistic images, creating deepfakes, enhancing low-resolution images, and more. GANs have greatly advanced the field of generative modeling and have opened up new possibilities for creative applications in artificial intelligence.

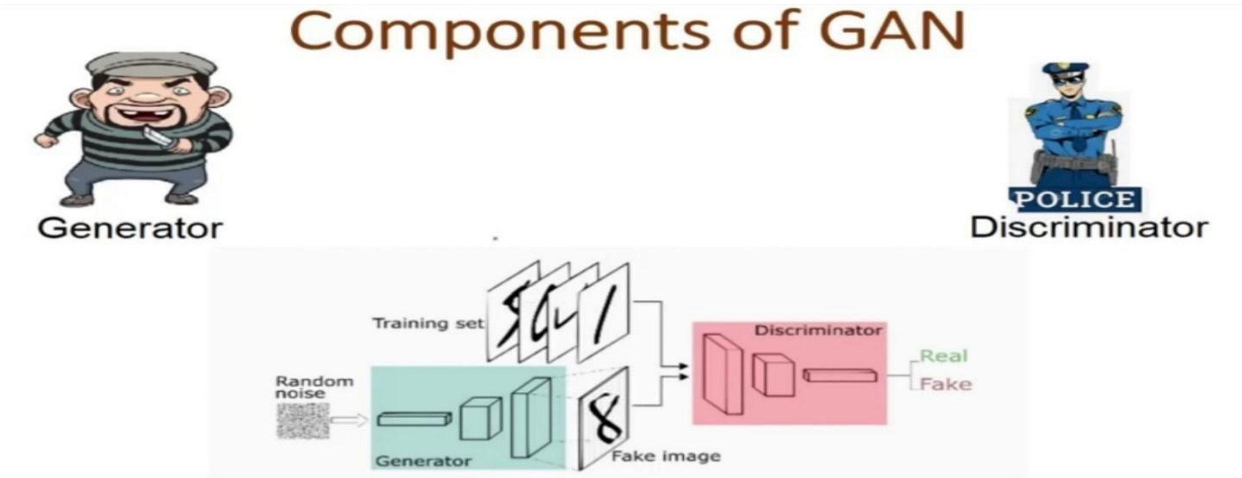
WhyGANs wasDeveloped?

Machine learning algorithms and neural networks can easily be fooled to misclassify things by adding some amount of noise to data. After adding some amountof noise, the chancesof misclassifyingthe imagesincrease.Hence the small rise that, is it possible to implement something that neural networks can start visualizing new patterns like sample train data. Thus, GANs were built that generate new fake results similar to the original.

ComponentsofGenerativeAdversarialNetworks(GANs):

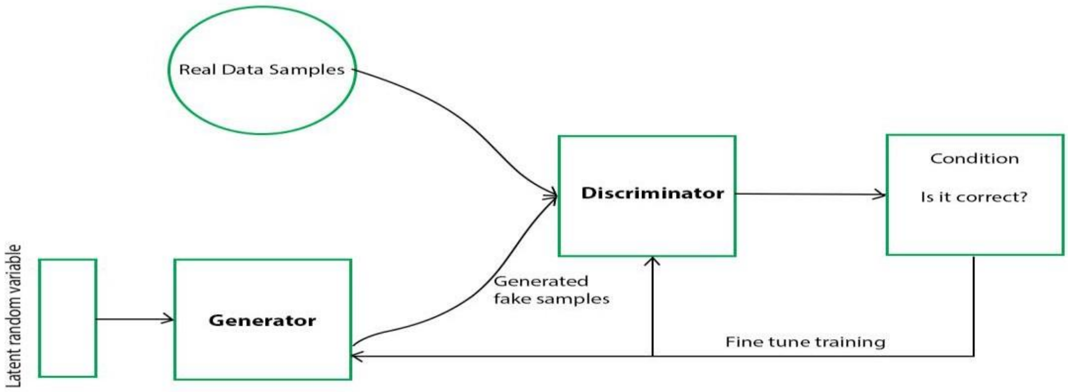
WhatisGeometricIntuitionbehindtheworkingofGANs?

Two major components of GANs are Generator and Discriminator. The role of the generator is like a thief to generate the fake samples based on the original sample and make the discriminator fool to understand Fake as real. On the other hand, a Discriminator is like a Police whose role is to identify the abnormalities in the samples created by Generator and classify them as Fake or real. This competition between both the component goes on until the level of perfection is achieved where Generator wins making a Discriminator fool on fake data.



1) Discriminator –It is a supervised approach means It is a simple classifier that predicts data is fake or real. It is trained on real data and provides feedback to a generator.

2) Generator –It is an unsupervised learning approach. It will generate data that is fake data based on original(real) data. It is also a neural network that has hidden layers, activation, loss function. Its aim is to generate the fake image based on feedback and make the discriminator fool that it cannot predict a fake image. And when the discriminator is made a fool by the generator, the training stops and wecan say that a generalized GAN model is created.

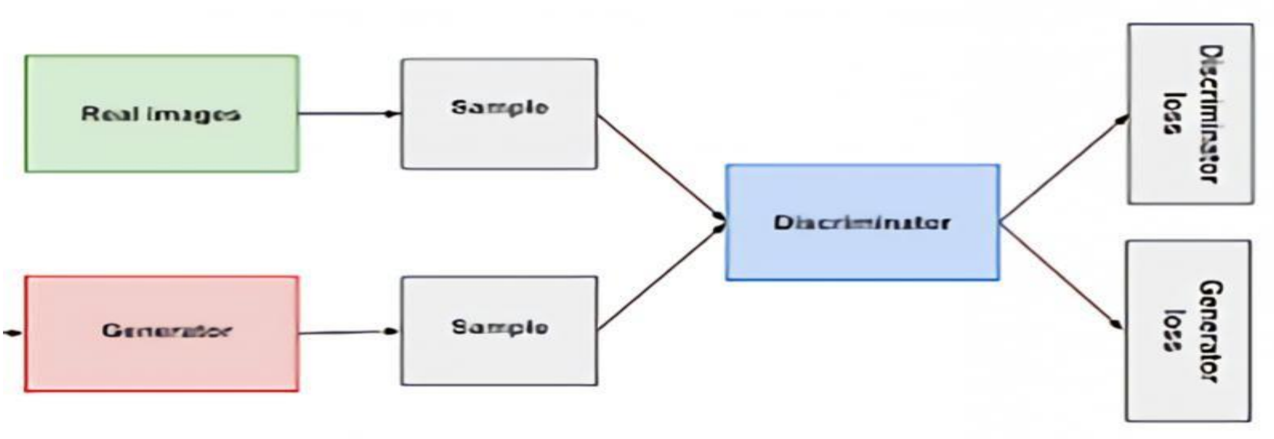


Here, the generative model captures the distribution of data and is trained in such a manner to generate the new sample that tries to maximize the probability of the discriminator to make a mistake (maximize discriminator loss). The discriminator on other hand is based on a model that estimates the probability that the sample it receives is from training data not from the generator and tries to classify it accurately and minimize the GAN accuracy. Hence the GAN network is formulated as aminimax game where the Discriminator is trying to minimize its reward **V(D, G)**and the generator is trying to maximize the Discriminator loss.

Thebelowfigureaddressestheconstraints

How is an actual architecture of GAN?

Howtwoneuralnetworksarebuildandtrainingandpredictionis done?



Both the components are neural networks.The generator output is directly connected to the input of the discriminator. And discriminator predicts it and through backpropagation, the generator receives a feedback signal to update weights and improve performance. The discriminator is a feed-forward neural network.

Training&PredictionofGenerativeAdversarialNetworks(GANs):

Step-1) Define a Problem

The problem statement is key to the success of the project so the first step is to define the problem. GANs work with a different set of problems you are aiming so you need to define What you are creating like audio, poem, text, Image is a type of problem.

Step-2)SelectArchitectureofGAN

There are many different types of GAN & based on the scenario(s), a suitable GANarchitecture is chosen.

Step-3)TrainDiscriminatoronRealDataset

Now, Discriminator is trained on a real dataset. It is only having a forwardpath.NobackpropagationisthereinthetrainingoftheDiscriminatorinnepochs.

And the provided Data is without Noise and only contains real images, and for fakeimages, Discriminator uses instances created by the generator as negative output.

DiscriminatorTraining:

- Itclassifiesbothrealandfakedata.
- Thediscriminatorlosshelpsimproveitsperformanceandpenalizeitwhenit misclassifies real as fake or vice-versa.
- weightsofthediscriminatorareupdatedthroughdiscriminatorloss.

Step-4)Train Generator

Provide some Fake inputs for the generator (Noise) and it will use some random noise and generate some fake outputs. when Generator is trained, Discriminator is Idle and when Discriminator is trained, Generator is Idle. During generator training through any random noise as input, it tries to transform it into meaningful data. to get meaningful output from the generator takes time and runs under many epochs. Steps to train a generator are listed below.

- Getrandomnoiseandproduce ageneratoroutput on noisesample
- Predictgeneratoroutputfromdiscriminatorasoriginalorfake.
- Calculatediscriminatorloss.
- Performbackpropagationthroughdiscriminator,andgeneratorbothtocalculategradients.
- Usegradientstoupdategenerator weights.

Step-5)TrainDiscriminatoronFakeData

The samples which are generated by Generator will pass to Discriminator and It will predict the data passed to it is Fake or real and provide feedback to Generator again.

Step-6)TrainGeneratorwiththeoutputofDiscriminator

Again, Generator will be trained on the feedback given by Discriminator andtry to improve performance. This is an iterative process and continues running until the Generator is not successful in making the discriminator fool.

GenerativeAdversarialNetworks(GANs)LossFunction:

The loss function is used in minimize and maximize of the iterative process. The generator tries to minimize the following loss function while the discriminatortries to maximize it. It is the same as a minimax game if you have ever played.

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

- D(x)isthediscriminator'sestimateoftheprobabilitythatrealdatainstancexis real.
- $E_x$isthe expectedvalueoverall realdatainstances.
- G(z)isthe generator'soutput when given noisez.
- D(G(z))isthediscriminator'sestimateoftheprobabilitythatafakeinstanceis real.
- E_z istheexpectedvalueoverallrandominputstothegenerator(ineffect,theexpected value over all generated fake instances G(z)).

ChallengesFacedbyGenerative AdversarialNetworks (GANs):

1. The problem of stability between generator and discriminator. The discriminator should not be too strict nor too lenient.
2. Problem to determine the positioning of objects - Suppose in a picture wehave 3 horse and generator have created 6 eyes and 1 horse.
3. The problem in understanding the global objects –GANs do not understand the global structure or holistic structure which is similar to the problem of perspective. It means sometimes GAN generates an image that is unrealistic and cannot be possible.

4. Problem in understanding the perspective - It cannot understand the 3-d images and if we train it on such types of images then it will fail to create 3-d images because today GANs are capable to work on 1-d images.

DifferentTypesofGenerativeAdversarialNetworks(GANs):

1) DC GAN –It is a Deep convolutional GAN. It is one of the most used, powerful, and successful typesof GANarchitecture.It is implemented with help of ConvNets in place ofaMulti-layeredperceptron.The ConvNetsusea convolutionalstrideandare built without max pooling and layers in this network are not completely connected.

2) Conditional GAN and Unconditional GAN (CGAN) –Conditional GAN is deep learning neural network in which some additional parameters are used. Labels are also put in inputs of Discriminator in order to help the discriminator to classify the input correctly and not easily full by the generator.

3) Least Square GAN (LSGAN) –It is a type of GAN that adopts the least-square lossfunctionforthediscriminator.Minimizingtheobjectivefunctionof LSGANresults in minimizing the Pearson divergence.

4) Auxilary Classifier GAN (ACGAN) –It is the same as CGAN and an advanced version of it. It says that the Discriminator should not only classify the image as real or fake but should also provide the source or class label of the input image.

5) Dual Video Discriminator GAN –DVD-GAN is a generative adversarial network for video generation built upon the BigGAN architecture. DVD-GAN uses two discriminators: **a Spatial Discriminator and a Temporal Discriminator**.

6) Single Image Super Resolution GAN (SRGAN) – Its main function is to transform low resolution to high resolution known as Domain Transformation.

7) Cycle GAN - It is released in 2017 which performs the task of Image Translation. Suppose we have trained it on a horse image dataset and we can translate it into zebra images.

8) Info GAN–Advance version of GAN which is capable to learn to disentangle representationinanunsupervisedlearningapproach.

TopGenerativeAdversarialNetworksApplications:

1) Generate Examples for Image Datasets: GANs can be used to generate new examples for image datasets in various domains, such as medical imaging, satellite imagery,and naturallanguageprocessing.Bygeneratingsyntheticdata, researcherscan augment existingdatasets and improve the performance of machine learning models.

2) Generate Photographs of Human Faces: GANs can generate realistic photographs of human faces, including images of people who do not exist in the real world. We can use these rendered images for various purposes, such as creating avatars for online games or social media profiles.

3) Generate Realistic Photographs: GANs can generate realistic photographs of various objects and scenes, including landscapes, animals, and architecture. These renderedimagescanbeusedtoaugmentexistingimagedatasetsortocreateentirely new datasets.

4) Generate Cartoon Characters: GANs can be used to generate cartoon characters that are similar to those found in popular movies or television shows. These developed characters can create new content or customize existingcharacters in games and other applications.

5) Image-to-Image Translation: GANs can translate images from one domain to another, such as convertinga photograph of a real-world scene intoa line drawingor a painting. We can create new content or transform existing images in various ways.

6) Text-to-Image Translation: GANs can be used to generate images based on a given text description. We can use it to create visual representations of concepts or generate images for machine learning tasks.

7) Semantic-Image-to-Photo Translation: GANs can translate images from a semantic representation (such as a label map or a segmentation map) into a realistic photograph. We can use it to generate synthetic data for training machine learning models or to visualize concepts more practically.

8) Face Frontal View Generation: GANs can generate frontal views of faces from images that show the face at an angle. We can use it to improve face recognition algorithm's performance or synthesize pictures for use in other applications.

9) Generate New Human Poses: GANs can generate images of people in new poses, such as difficult or impossible for humans to achieve. It can be used to create new content or to augment existing image datasets.

10) Photos to Emojis: GANs can be used to convert photographs of people into emojis, creating a more personalized and expressive form of communication.

11) Photograph Editing: GANs can be used to edit photographs in various ways, such as changing the background, adding or removing objects, or altering the appearance of people or animals in the image.

12) Face Aging: GANs can be used to generate images of people at different ages, allowing users to visualize how they might look in the future or to see what theymight have looked like in the past.

DifferencesBetweenDiscriminativeandGenerativeModels

1) Core Idea

Discriminative models draw boundaries in the data space, while generative models try to model how data is placed throughout the space. A generative model explains how the data was generated, while a discriminative model focuses on predicting the labels of the data.

2) MathematicalIntuition

In mathematical terms, discriminative machine learning trains a model, which isdonebylearningparametersthatmaximizetheconditional probability**P(Y|X).**On the other hand, a generative model learns parameters by maximizing the joint probability of **P(X, Y)**.

3) Applications

Discriminative models recognize existing data, i.e., discriminative modeling identifies tags and sorts data and can be used to classify data, while Generative modeling produces something.

Since these models use different approaches to machine learning, both are suited for specific tasks i.e., Generative models are useful for unsupervised learning tasks. In contrast, discriminative models are useful for supervised learning tasks. GANs(Generativeadversarialnetworks)canbethoughtofasa competitionbetween the generator, which is a component of the generative model, and the discriminator, so basically, it is generative vs. discriminative model.

4) Outliers

Generativemodelshavemoreimpactonoutliersthandiscriminativemodels.

5) ComputationalCost

Discriminative models are computationally cheap as compared to generative models.

ComparisonBetweenDiscriminativeandGenerative Models:

1) Based on Performance

Generative models need fewer data to train compared with discriminative models since generative models are more biased as they make stronger assumptions, i.e., **assumption of conditional independence**.

2) BasedonMissingData

In general, if we have missing data in our dataset, then Generative modelscan work with these missing data, while discriminativemodels can't.This isbecause, in generative models, we can still estimate the posterior by marginalizing the unseen variables. However, discriminative models usually require all the features X to be observed.

3) Basedonthe AccuracyScore

If the assumption of conditional independence violates, then at that time, generative models are less accurate than discriminative models.

4) Based onApplications

Discriminative models are called **"discriminative"**since they are useful for discriminating Y's label, i.e., target outcome, so they can only solve classification problems. In contrast, Generative models have more applications besides classification, such as samplings, Bayes learning, MAP inference, etc.

GenerativeModelsvsDiscriminativeModels:

Machine learning (ML) and Deep Learning (DL) are two of the most exciting andconstantlychangingfieldsofstudyofthe21stcentury.Usingthese technologies,machinesaregiventheabilitytolearnfrompastdataandpredictor make decisions from future, unseen data.

The inspiration comes from the human mind, how we use past experiences to help us make informed decisions in the present and the future. And while there are already many applications of ML and DL, the future possibilities are endless.

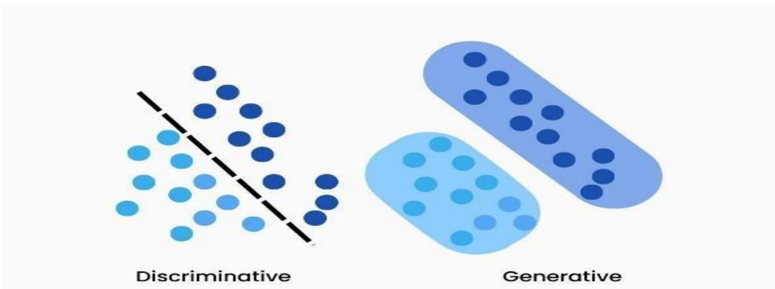
Computers utilize mathematics, algorithms, and data pipelines to draw meaningful inferences from raw data since they cannot perceive data andinformation like humans - not yet, at least. There are two ways we can improve a machine's efficiency: either get more data or come up with newer or more robust algorithms.

Quintillions of data are generated all over the world almost daily, so getting fresh data is easy. But in order to work with this gigantic amount of data, we need new algorithms or we need to scale up existing ones.

Mathematics, especially branches like calculus, probability, statistics, etc., is the backbone of these algorithms or models. They can be widely divided into two groups:

1. Discriminativemodels
2. Generativemodels

Mathematically, generative classifiers assume a functional form for P(Y) and P(X|Y), then generate estimated parameters from the data and use the Bayes' theorem to calculate P(Y|X) (posterior probability). Meanwhile, discriminative classifiers assume a functional form of P(Y|X) and estimate the parameters directly from the provided data.



Discriminativemodel

The majority of discriminative/conditional models, are used for supervised machine learning. They do what they ‘literally’ say, separating the data points into different classes and learning the boundaries using probability estimates and maximum likelihood.

Outliers have little to no effect on these models. They are a better choice than generative models, but this leads to misclassification problems which can be a major drawback.

Here are some examples and a brief description of the widely used discriminative models:

1. Logisticregression: Logisticregression can be considered the linearregressionof classification models. The main idea behind both the algorithms is similar, but while linear regression is used for predicting a continuous dependent variable, logistic regression is used to differentiate between two or more classes.

2. Support vector machines: This is a powerful learning algorithm with applicationsin both regression and classification scenarios. An n-dimensional space containing the data points is divided into classes by decision boundaries using support vectors. The best boundary is called a hyperplane.

3. Decision trees: A graphical tree-like model is used to map decisions and their probable outcomes. It could be thought of as a robust version of If-else statements.

A few other examples are commonly-used neural nets, k-nearest neighbor (KNN), conditional random field (CRF), random forest, etc.

Generativemodel

As the name suggests, generative models can be used to generate new data points. These models are usually used in unsupervised machine learning problems. Generative models go in-depth to model the actual data distribution and learn the different data points, rather than model just the decision boundary between classes.

These models are prone to outliers, which is their only drawback when compared to discriminative models. The mathematics behind generative models is quite intuitive too. The method is not direct like in the case of discriminative models.

TocalculateP(Y|X),they firstestimatethepriorprobability P(Y)andthelikelihood probability P(X|Y) from the data provided.

Putting the values into Bayes' theorem's equation, we get an accurate valuefor P(Y|X).

$$posterior = \frac{prior \times likelihood}{evidence} \Rightarrow P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

Someexamplesaswellasadescriptionofgenerativemodelsareasfollows:

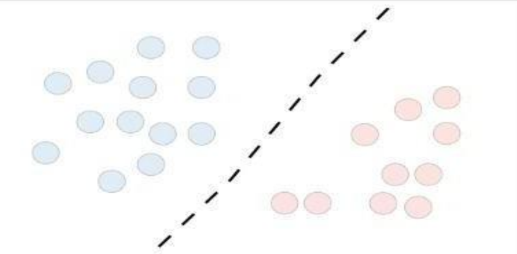
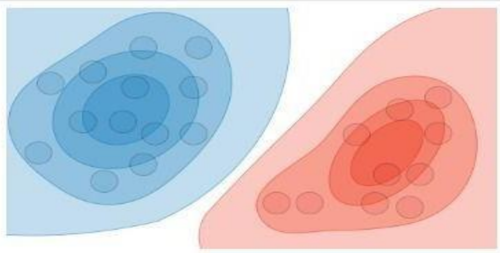
1. Bayesian network: Also known as Bayes' network, this model uses a directed acyclic graph (DAG) to draw Bayesian inferences over a set of random variables to calculate probabilities. It has many applications like prediction, anomaly detection, time series prediction, etc.

2. Autoregressive model: Mainly used for time series modeling, it finds a correlation between past behaviors to predict future behaviors.

3. Generative adversarial network (GAN): It's based on deep learning technology and uses two sub models. The generator model trains and generates new datapoints and the discriminative model classifies these 'generated' data points into real or fake.

SomeotherexamplesincludeNaiveBayes,Markovrandomfield,hiddenMarkov model (HMM), latent Dirichlet allocation (LDA), etc.

Discriminativevsgenerative:WhichisthebestfitforDeepLearning?

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Discriminative models divide the data space into classes by learning the boundaries, whereas generative models understand how the data is embedded into the space. Both the approaches are widely different, which makes them suited for specific tasks.

Deep learning has mostly been using supervised machine learning algorithms like Artificial Neural Networks (ANNs), convolutional neural networks (CNNs), and Recurrent Neural Networks (RNNs). ANN is the earliest in the trio and leverages artificial neurons, backpropagation, weights, and biases to identifypatterns based on the inputs. CNN is mostly used for image recognition and computer vision tasks. It works by pooling important features from an input image. RNN, which is the latest of the three, is used in advanced fields like natural language processing, handwriting recognition, time series analysis, etc.

These arethefieldswherediscriminative modelsareeffective andbetterused for deep learning as they work well for supervised tasks. Apart from these, deep learning and neural nets can be used to cluster images based on similarities. Algorithms like autoencoder, Boltzmann machine, and self-organizing maps are popular unsupervised deep learning algorithms. They make use of generativemodels for tasks like exploratory data analysis (EDA) of high dimensional datasets, image denoising, image compression, anomaly detection and even generating new images.

This Person Does Not Exist - Random Face Generatoris an interesting website that uses a type of generative model called StyleGAN to create realistic human faces, even though the people in these images don’t exist!

These people are not real – they were produced by our generator that allows control over different aspects of the image. Nvidia / StyleGAN

UNIT-V

AUTO-ENCODERS: Auto-encoders, Architecture and components of auto-encoders (encoder and decoder), Training an auto-encoder for data compression and reconstruction, Relationship between Autoencoders and GANs, Hybrid Models: Encoder-Decoder GANs.

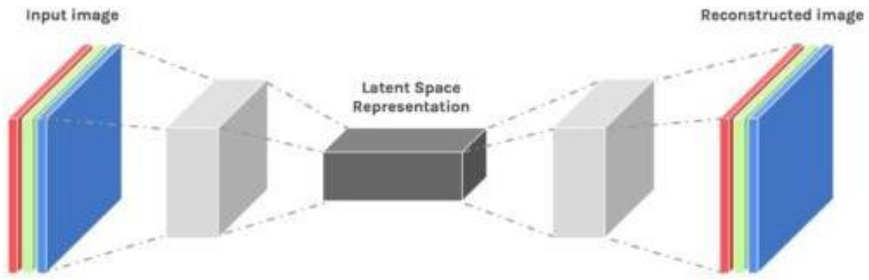
Auto-encoders:

Autoencoders are a type of deep learning algorithm that are designed to receive an input and transform it into a different representation. They play an important part in image construction. Artificial Intelligence encircles a wide range of technologies and techniques that enable computer systems to solve problems like Data Compression which is used in computer vision, computer networks, computer architecture, and many other fields.

Autoencoders are*unsupervised neural networks*that use machine learningto do this compression for us.

What Are Autoencoders?

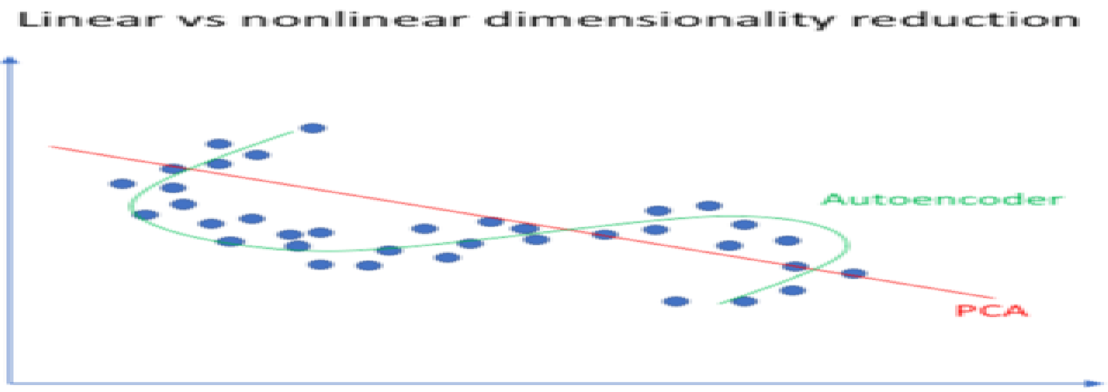
An autoencoder **neural network**is an **Unsupervised Machine learning**algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.



Similar machine learning algorithm i.e., PCA (Principal Component Analysis) which does the same task also co-exists.

Autoencoders:ItsEmergence

AutoencodersarepreferredoverPCAbecause:



- Anautoencodercanlearn **non-lineartransformations**witha **non-linear activation function** and multiple layers.
- It doesn'thave to learndense layers. It can use **convolutionallayers**to learn which is better for video, image and series data.
- Itismoreefficienttolearnseverallayerswithanautoencoderratherthan learn one huge transformation with PCA.
- Anautoencoderprovidesa representationofeachlayerastheoutput.
- Itcanmakeuseof **pre-trainedlayers** fromanothermodeltoapplytransfer learning to enhance the encoder/decoder.

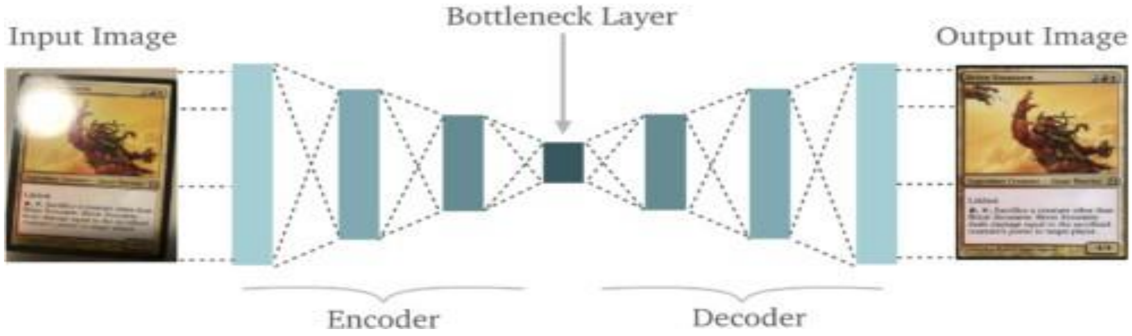
ApplicationsofAutoencoders

1) **ImageColoring**

Autoencoders are used for converting any black and white picture into a colored image. Depending on what is in the picture, it is possible to tell what thecolor should be.

2) Featurevariation

It extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption.

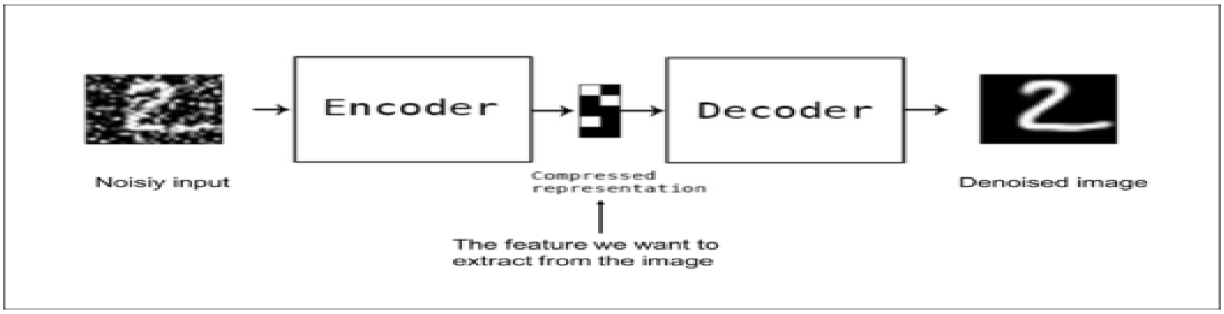


3) DimensionalityReduction

The reconstructed image is the same as our input but with reduced dimensions. It helps in providing the similar image with a reduced pixel value.

4) DenoisingImage

The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original input from the noisy version.



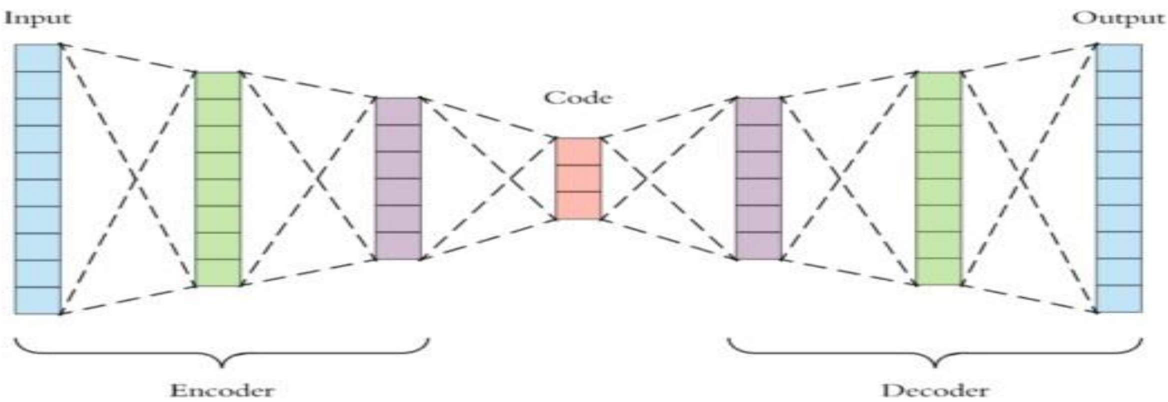
5) WatermarkRemoval

It is also used for removing watermarks from images or to remove any object while filming a video or a movie.

ArchitectureofAutoencoders

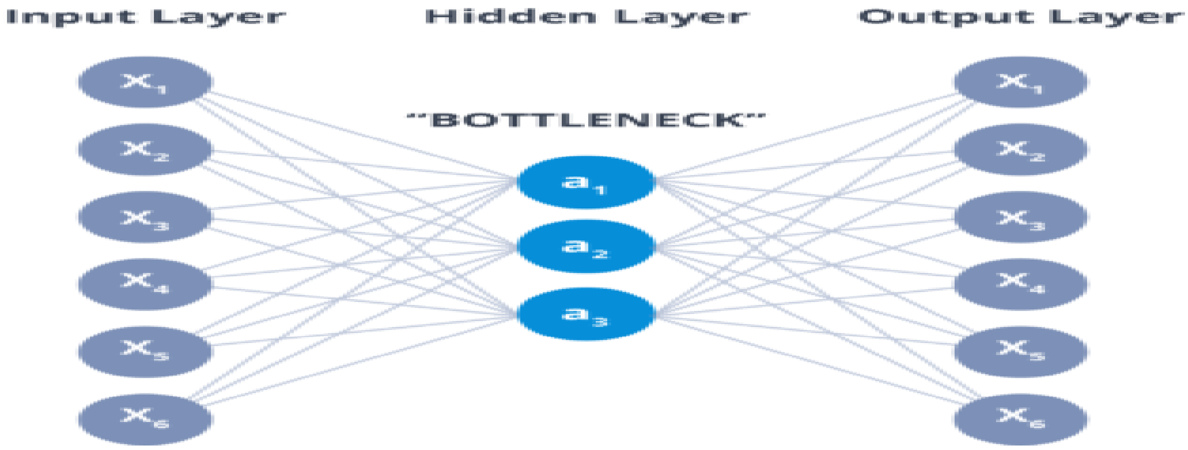
AnAutoencoderconsistofthreelayers:

1. **Encoder**
2. **Code**
3. **Decoder**



- **Encoder:**This part of the network compresses the input into a **latent space representation**.Theencoderlayer **encodes** theinputimageasa compressed representation in a reduced dimension. The compressed imageis the distorted version of the original image.
- **Code:**Thispart of the network represents the compressed input which is fed to the decoder.

- **Decoder:**This layer**decodes**the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.



Thelayerbetweentheencoderanddecoder,ie.thecodeisalsoknown as **Bottleneck**. This is a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded. It does this by balancing two criteria:

- Compactnessofrepresentation,measuredasthecompressibility.
- Itretainssomebehaviourallyrelevant variablesfromtheinput.

Traininganauto-encoderfordatacompressionandreconstruction:

An autoencoder consists of two parts: an encoder network and a decoder network. The encoder network compresses the input data, while the decodernetwork reconstructs the compressed data back into its original form. The compressed data, also known as the bottleneck layer, is typically much smaller than the input data.

The encoder network takes the input data and maps it to a lower-dimensional representation. This lower-dimensional representation is the compressed data. The decoder network takes this compressed data and maps it back to the original input data. The decoder network is essentially the inverse of the encoder network.

The bottleneck layer is the layer in the middle of the autoencoder thatcontains the compressed data. This layer is much smaller than the input data, which is what allows for compression. The size of the bottleneck layer determines the amount of compression that can be achieved. Autoencoders differ from other deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in that they do not require labeled data. Autoencoders can learn the underlying structure of the data without any explicit labels.

Image CompressionwithAutoencoders

There are two types of image compression: lossless and lossy. Lossless compression methods preserve all of the data in the original image, while lossy compression methods discard some of the data to achieve higher compressionrates.

Autoencoders can be used for both lossless and lossy compression. Lossless compression can be achieved by using a bottleneck layer that is the same size asthe input data. In thiscase, the autoencoderessentiallylearns to encode anddecode the input data without any loss of information.

Lossy compression can be achieved by using a bottleneck layer that issmaller than the input data. In this case, the autoencoder learns to discard some of the data to achieve higher compression rates. The amount of data that is discarded depends on the size of the bottleneck layer.
Herearesomeexamplesofimagecompressionusingautoencoders:

- A 512×512 color image can be compressed to a 64×64 grayscale image using an autoencoder with a bottleneck layer of size 64.
- A 256×256 grayscale image can be compressed to a 128×128grayscale image using an autoencoder with a bottleneck layer of size 128.

The effectiveness of autoencoder-based compression techniques can be evaluated by comparing the compressed and reconstructed images to the original images. The most common evaluation metric is the peak signal-to-noise ratio (PSNR), which measures the amount of noise introduced by the compression algorithm. Higher PSNR values indicate better compression quality.

ImageReconstructionwithAutoencoders

Autoencoders are a type of neural network that can be used for image compression and reconstruction. The process involves compressing an image into a smaller representation and then reconstructing it back to its original form. Image reconstruction is the process of creating an image from compressed data.

Explanationofimagereconstructionfromcompressed data:

The compressed data can be thought of as a compressed version of the original image. To reconstruct the image, the compressed data is fed through a decoder network, which expands the data back to its original size. The reconstructed image will not be identical to the original, but it will be a close approximation.

Howautoencoderscanbeusedforimagereconstruction:

Autoencoders use a loss function to determine how well the reconstructed image matches the original. The loss function calculates the difference between the reconstructed image and the original image. The goal of the autoencoder is to minimize the loss function so that the reconstructed image is as close to the original as possible.

Examplesofimagereconstructionusingautoencoders:

An example of image reconstruction using autoencoders is the MNISTdataset, which consists of handwritten digits. The autoencoder is trained on the dataset to compress and reconstruct the images. Another example is the CIFAR-10 dataset, which consists of 32×32 color images of objects. The autoencoder can be trained on this dataset to compress and reconstruct the images.

Autoencoder-basedreconstructiontechniquesefficiencyevaluation:

The effectiveness of autoencoder-based reconstruction techniques can be evaluated using metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarityindex(SSIM).PSNRmeasuresthequalityofthereconstructedimageby comparingittotheoriginalimage,whileSSIMmeasuresthestructuralsimilarity between the reconstructed and original images.

VariationsofAutoencodersforImageCompressionandReconstruction

Autoencoders can be modified and improved for better image compression and reconstruction. Some of the variations of autoencoders are:

1) Denoisingautoencoders:

Denoising autoencoders are used to remove noise from images. The autoencoder is trained on noisy images and is trained to reconstruct the original image from the noisy input.

2) Variationalautoencoders:

Variational autoencoders (VAEs) are a type of autoencoder that learn the probability distribution of the input data. VAEs are trained to generate new samples from the learned distribution. This makes VAEs suitable for image generation tasks.

3) Convolutionalautoencoders:

Convolutional autoencoders (CAEs) use convolutional neural networks (CNNs) for image compression and reconstruction. CNNs are specialized neural networks that can learn features from images.

Comparisonoftheeffectivenessofdifferenttypesofautoencodersforimage compression & reconstruction:

The effectiveness of different types of autoencoders for image compression and reconstruction can be compared using metrics such as PSNR and SSIM. CAEs are generally more effective for image compression and reconstruction than other types of autoencoders. VAEs are better suited for image generation tasks.

Real-TimeExamples:

A real-time example of an autoencoder for image compression and reconstructionisGoogle'sGuetzlialgorithm.Guetzliusesacombinationofa perceptual metric and a psycho-visual model to compress images while maintaining their quality. Another example is the Deep Image Prior algorithm, which uses a convolutional neural network to reconstruct images from compressed data.

ApplicationsofAutoencodersforImageCompressionandReconstruction

Autoencoders have become increasingly popular for image compression and reconstruction tasks due to their ability to learn efficient representations of the input data. In this, we will explore some of the common applications of autoencoders for image compression and reconstruction.

1) MedicalImaging:

Autoencoders have shown great promise in medical imaging applicationssuch as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), and X- Ray imaging. The ability of autoencoders to learn feature representations from high-dimensional data has made them useful for compressing medical images while preserving diagnostic information.

For example, researchers have developed a deep learning-basedautoencoder approach for compressing 3D MRI images, which achieved higher compressionratiosthantraditionalcompressionmethodswhilepreservingdiagnostic quality. This can have significant implications for improving the storage and transmission of medical images, especially in resource-limited settings.

2) VideoCompression:

Autoencoders have also been used for video compression, where the goal is to compress a sequence of images into a compact representation that can be transmitted or stored efficiently. One example of this is the video codec AV1, which uses a combination ofautoencodersand traditional compression methods to achieve higher compression rates while maintaining video quality. The autoencoder component of the codec is used to learn spatial and temporal features of the video frames, which are then used to reduce redundancy in the video data.

3) AutonomousVehicles:

Autoencoders are also useful for autonomous vehicle applications, where the goal is to compress high-resolution camera images captured by the vehicle'ssensors while preserving critical information for navigation and obstacle detection. For example, researchers have developed an autoencoder-based approach for compressing images captured by a self-driving car, which achieved highcompression ratioswhilepreservingtheaccuracyof objectdetectionalgorithms.This can have significant implications for improving the performance and reliability of autonomous vehicles, especially in scenarios where high-bandwidth communication is not available.

4) SocialMediaandWebApplications:

Autoencoders have also been used in social media and web applications, where the goal is to reduce the size of image files to improve website loading times and reduce bandwidth usage. For example, Facebook uses an autoencoder-based approach for compressing images uploaded to their platform, which achieves high compression ratios while preserving image quality. This has led to faster loading times for images on the platform and reduced data usage for users.

Comparison of the effectiveness of autoencoder-based compression and reconstruction techniques for different applications:

The effectiveness of autoencoder-based compression and reconstruction techniques can vary depending on the application and the specific requirements of the task. For example, in medical imaging applications, the preservation ofdiagnostic informationiscritical, while in socialmediaapplications, image qualityand loading times may be more important. Researchers have compared theeffectiveness of autoencoder-based compression and reconstruction techniques with traditional compression methods and have found that autoencoder-based methods often outperformtraditionalmethodsin termsof compression ratio and image quality.

RelationshipbetweenAutoencodersandGANs:

Autoencoders and GANs are both powerful techniques for learning from data in an unsupervised way, but they have some differences and trade-offs.Autoencoders are easier to train and more stable, but they tend to produce blurry or distorted reconstructions or generations. GANs are harder to train and more proneto mode collapse, where they produce only a few modes of the data distribution, but theytendtoproducesharperandmorediversegenerations.Dependingonyourgoal and your data, you might prefer one or the other, or even combine them in a hybrid model.

Autoencoders are unsupervised models, which means that they are nottrained on labeled data. Instead, they are trained on unlabeled data and learn to reconstruct the input data. GANs, on the other hand, are supervised models, which means that they are trained on labeled data. The generator in a GAN is trained to generate data that looks like the labeled data, and the discriminator is trained to distinguish between real and fake data. Autoencoders are typically used for tasks such as image denoising and compression. GANs are typically used for tasks such as image generation and translation.

HybridModels:Encoder-DecoderGANs:

HowcanyoucombineGANsandautoencoderstocreatehybridmodelsforvarious tasks?

Generativeadversarialnetworks(GANs)andautoencodersaretwopowerfultypesof artificial neural networks that can learn from data and generate new samples. But what if you could combine them to create hybrid models that can perform various tasks, ***such as image synthesis, anomaly detection***, or domain adaptation.

GANsandautoencoders

GANs are composed of two networks: a generator and a discriminator. The generator tries to create realistic samples from random noise, while the discriminator tries to distinguish between real and fake samples. The two networks compete with each other, improving their skills over time. Autoencoders are composed of two networks:anencoderandadecoder.Theencodercompressestheinputdataintoa lower-dimensional representation, while the decoder reconstructs the input datafrom the representation. The goal is to minimize the reconstruction error, while learning useful features from the data.

Hybridmodels

Hybrid models are models that combine GANs and autoencoders in different ways, depending on the task and the objective. For example, you can use an autoencoder as the generator of a GAN, and train it to fool the discriminator, while also minimizing the reconstruction error. This way, we can generate realistic samples that are similar to the input data, but also have some variations. Alternatively, youcan use a GAN as the encoder of an autoencoder, and train it to encode the input data into a latent space that is compatible with the discriminator. This way, you can learn ameaningfulrepresentation ofthedatathatcanbeusedfordownstreamtasks, such as classification or clustering.

Image synthesis

One of the most common tasks for hybrid models is image synthesis, which is the process of creating new images from existing ones, or from scratch. For example, you can use a hybrid model to synthesize images of faces, animals, or landscapes, by using an autoencoder as the generator of a GAN, and feeding it with real images or random noise. This way, you can create diverse and realistic images that preserve the attributes of the input data, but also have some variations. You can also use a hybrid model to synthesize images of different domains, such as converting photos to paintings, or day to night, by using a GAN as the encoder of an autoencoder, and feeding it with images from both domains. This way, you can learn a common latent space that can be used to transfer the style or the attributes of one domain to another.

Anomalydetection

Another task for hybrid models is anomaly detection, which is the process of identifying abnormal or unusual patterns in the data, such as outliers, frauds, or defects. For example, you can use a hybrid model to detect anomalies in images,such as damaged products, or medical conditions, by using an autoencoder as the generator of a GAN, and feeding it with normal images. This way, you can train the autoencoder to reconstruct normal images well, but fail to reconstruct abnormal images.

Then, we can use the reconstruction error or the discriminator score as a measure of anomaly. You can also use a hybrid model to detect anomalies in time series, such as sensor readings, or financial transactions, by using a GAN as the encoder ofan autoencoder, and feeding it with normal time series. This way, you can train the GAN to encode normal time series well, but fail to encode abnormal time series. Then, we can use the latent space or the discriminator score as a measure of anomaly.

Domainadaptation

A third task for hybrid models is domain adaptation, which is the process of adapting a model trained on one domain to work on another domain, without requiring labeled data from the target domain. For example, you can use a hybrid model to adapt a model trained on images of handwritten digits to work on images of handwritten letters, by using a GAN as the encoder of an autoencoder, andfeeding it with images from both domains. This way, you can train the GAN toencode both domains into a shared latent space that is invariant to the domain differences.