Neural Networks

(Theory)

CH5020

External Report Presentation

Course Instructor: Dr. Kannan A
Department of Chemical Engineering, IIT Madras

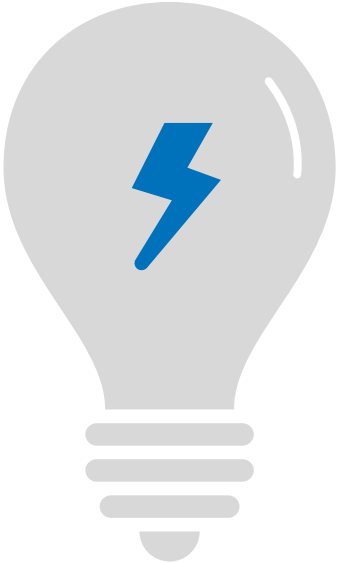
Mayur Vikas Joshi | ME16B148
Sushant Uttam Wadavkar | ME16B172

Presentation Content

- Introduction
- Neural Network
- Neuron

- Activation Function
- Types of NNs
- CNN,RNN, LSTM

- Mathematical Models
- Case Study - Turbulence Model



- Advantages
- Disadvantages

- Example Code (Keras)
- Applications
- Conclusions

- Future Work
- Summary
- Acknowledgement

What Is A Neural Network?

The simplest definition of a neural network (or Artificial Neural Network (ANN)), is provided by the inventor of one of the first neurocomputers, **Dr. Robert Hecht-Nielsen**.
He defines a neural network as:

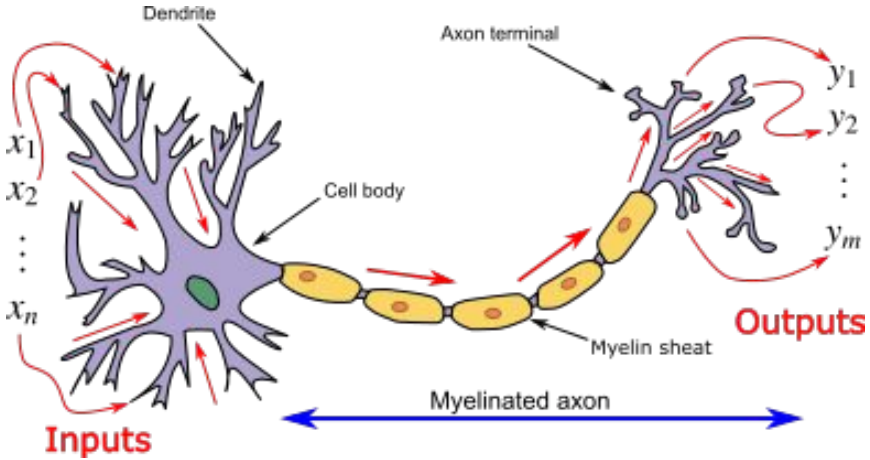
> *"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs"*

- Neural networks are multi-layer networks of neurons that are used to classify things, make predictions, etc.
- NNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the **mammalian cerebral cortex** but on much smaller scales
- A large NN might have hundreds or thousands of processor units, whereas a mammalian brain has billions of neurons with a corresponding **increase in magnitude of their overall interaction and emergent behavior**

Neuron

NNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produce a single output which can be sent to multiple other neurons.

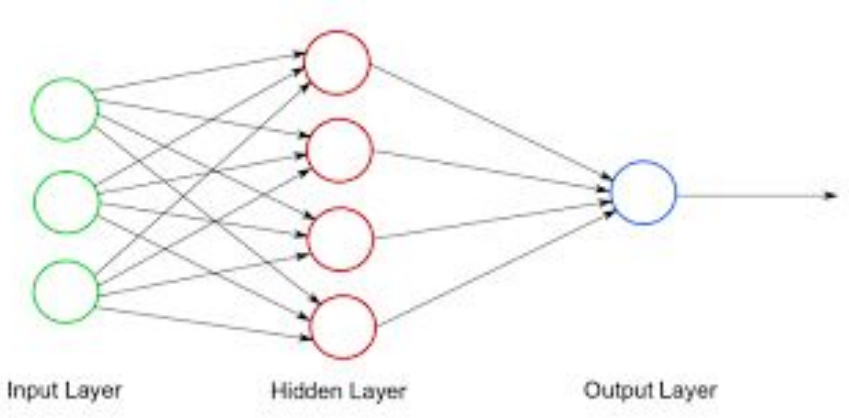
- The **inputs** can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons
- The outputs of the final ***output neurons*** of the neural net accomplish the task, such as recognizing an object in an image



- To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the ***weights*** of the ***connections*** from the inputs to the neuron, a ***bias*** term is then added to this sum
- Weighted sum is sometimes called the ***activation***
- Weighted sum is then passed through an (usually nonlinear) activation function to produce the output

Basics of Neural Network

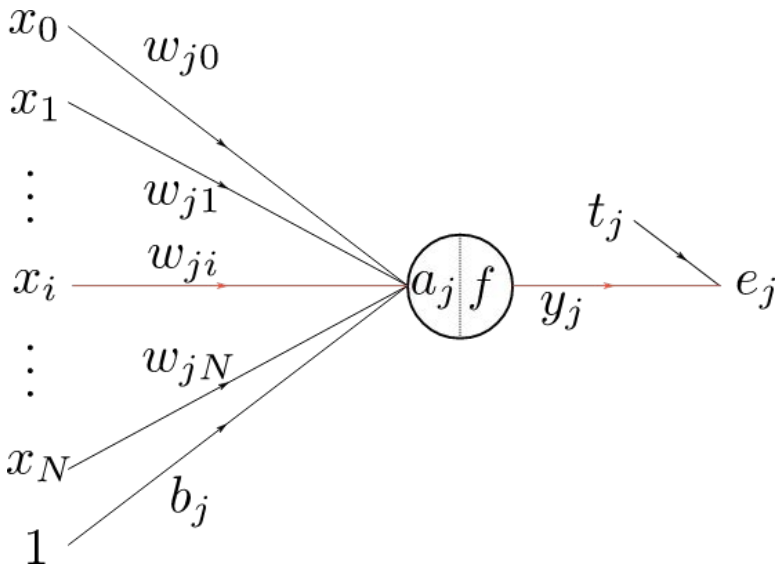
- Neural Networks are typically organized in layers
- Layers are made up of a number of interconnected **'nodes'** which contain an **‘Activation Function'**



- Patterns are presented to the network via the **'input layer'**, which communicates to one or more **‘Hidden Layers'** where the actual processing is done via a system of weighted **'connections'**
- The hidden layers then link to an **'Output Layer'** where the answer is output

Basics of Neural Network (Cont.)

- Most NNs contain some form of **'Learning Rule'** which modifies the weights of the connections according to the input patterns that it is presented with...
  - In a sense, NNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs.
- There are many different kinds of learning rules used by neural networks like **Delta Rule**



- The **Delta Rule** is often utilized by the most common class of NNs called **'Backpropagation Neural Networks'** (BPNNs)
- Backpropagation is an abbreviation for the backwards propagation of error

With the **delta rule**, as with other types of backpropagation, **'learning'** is a supervised process that occurs with each cycle or **'epoch'** (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the **backwards error propagation of weight adjustments**

Basics of Neural Network (Cont.)

Connections and Weights

- The network consists of connections, each connection providing the output of one neuron as an input to another neuron
- Each connection is assigned a weight that represents its **relative importance**
- A given **neuron** can have multiple input and output connections

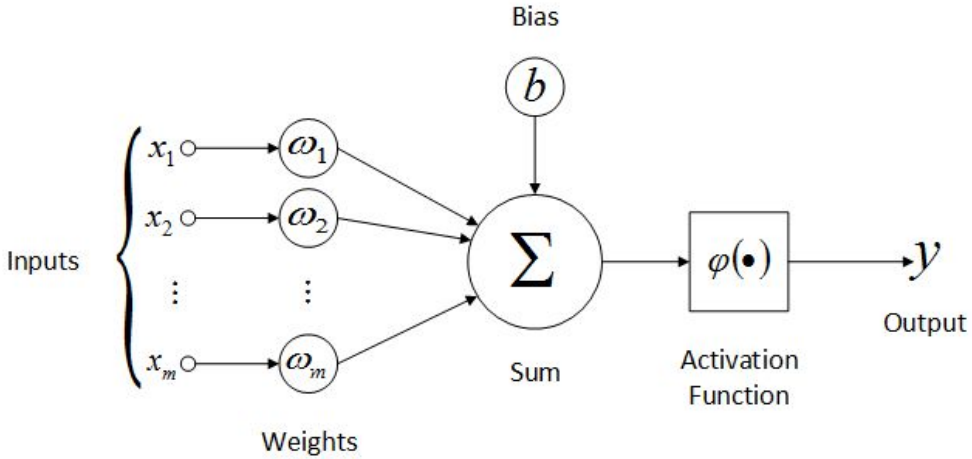
Propagation function

- The *propagation function* computes the input to a neuron from the outputs of its predecessor neurons and their connections as a **weighted sum**
- A ***bias* term** can be added to the result of the propagation

Hyperparameter

- A hyperparameter is a constant parameter whose value is set before the learning process begins
- The values of parameters are derived via learning
  - Examples of hyperparameters include learning rate, the number of hidden layers and batch size
- The values of some hyperparameters can be dependent on those of other hyperparameters
  - For example, the size of some layers can depend on the overall number of layers.

Mathematical Modelling



Formulation

$$x.w = (x_1 \times w_1) + (x_2 \times w_2) + \cdots + (x_n \times w_n)$$

$$z = x.w + b$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Learning Algorithm

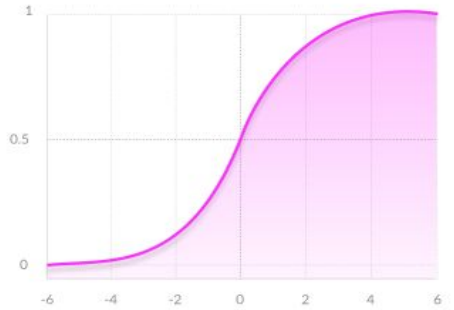
$$C = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$w_i = w_i - \left(\alpha \times \frac{\partial C}{\partial w_i}\right)$$

$$b = b - \left(\alpha \times \frac{\partial C}{\partial b}\right)$$

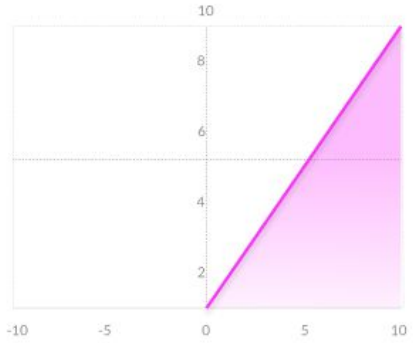
Activation Functions

Sigmoid



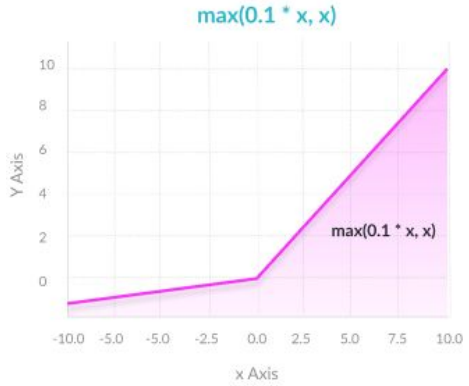
Equation :
A = 1/(1 + e-x)
Nature : Non-linear. Notice that X values lies between -2 to 2, Y values are very steep.
Value Range : 0 to 1
Uses : Usually used in output layer of a binary classification, where result is either 0 or 1

ReLU



Equation :- ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
Value Range :- [0, inf)
Nature :- non-linear
Uses :- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.

Leaky ReLU

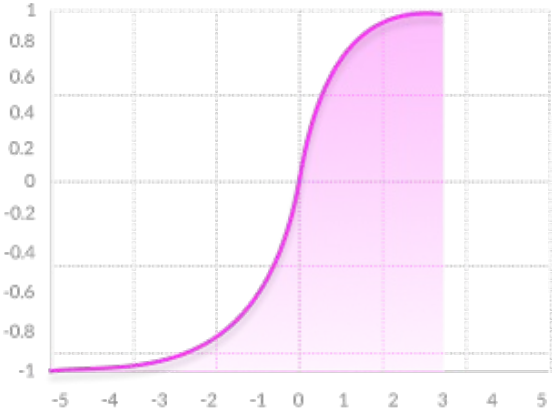


Equation: Instead of the **function** being zero when x < 0, a **leaky ReLU** will instead have a small negative slope (of 0.01, or so)

That is, the **function** computes f(x)=1(x<0)(αx)+1(x>=0)(x)
where α is a small constant.

Activation Functions

Tanh

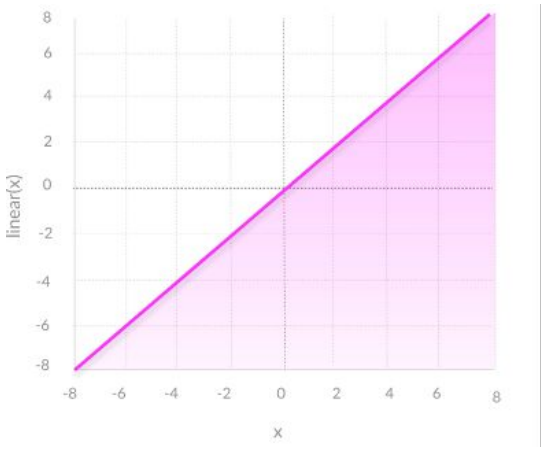


Equation :-

- f(x) = tanh(x) = 2/(1 + e-2x) - 1
- tanh(x) = 2 * sigmoid(2x) - 1

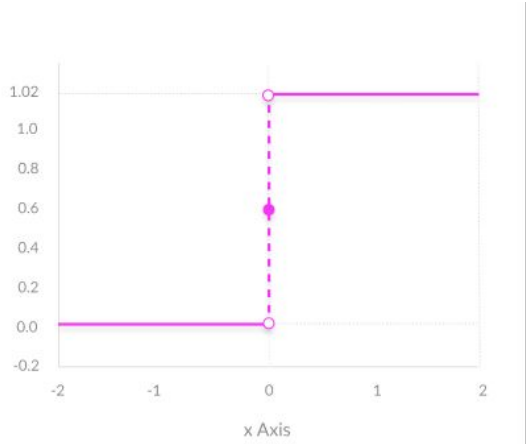
Value Range :- -1 to +1
Nature :- non-linear
Uses :- Usually used in hidden layers of a neural network as it's values lies between **-1 to 1**

Linear



Equation : Linear function has the equation similar to as of a straight line i.e. **y = ax**
Range : -inf to +inf
Nature :- linear
Uses : Linear activation function is used at just one place i.e. output layer.

Binary



- A binary step function is a threshold-based activation function.
- If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.

Neural Network Design

❖ **Neural architecture search** (NAS) uses machine learning to automate ANN design

- ➢ Various approaches to NAS have designed networks that compare well with hand-designed systems
- ➢ The basic search algorithm is to propose a candidate model, evaluate it against a dataset and use the results as feedback to teach the NAS network
- ➢ Available systems include **AutoML and AutoKeras**

- Design issues include deciding the number, type and connectedness of network layers, as well as the size of each and the connection type (full, pooling)
- Hyperparameters must also be defined as part of the design (they are not learned), governing matters such as how many neurons are in each layer, learning rate, step, stride, depth, receptive field and padding (for CNNs), etc.

Neural Networks example using Keras

```
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
# load the dataset
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
# split into input (X) and output (y) variables
X = dataset[:,0:8]
y = dataset[:,8]
# define the keras model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

Classification of Neural Network

Neural Networks can be classified based on following attributes:

Connection Type

1. Static (Feedforward)
2. Dynamic (Feedback)

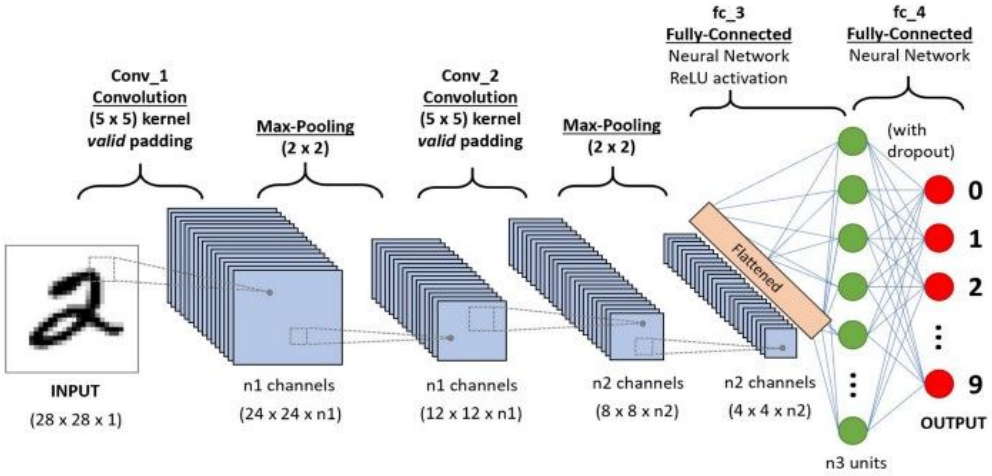
Topology

1. Single Layer
2. Multi Layer
3. Recurrent

Learning Methods

1. Supervised
2. Unsupervised
3. Reinforcement

Convolutional Neural Networks



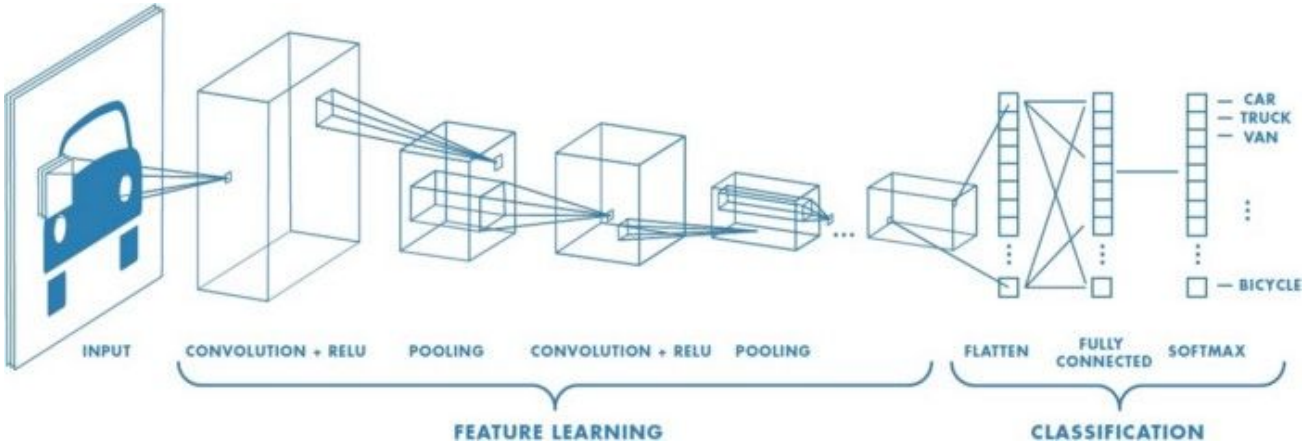
Why CNN over Feed-Forward Neural Networks?

- CNN can capture Spatial and Temporal dependencies in an image through relevant filters
- The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters and reusability of weights.

- CNN takes an input image and assigns importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

- Each input image passes through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and Softmax function to classify an object with probability values between 0 & 1 on the basis of which an image is classified.

CNN Working



Convolution

Extracts features from input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data

ReLU

Introduces non-linearity

Stride

Number of pixels that kernel shifts to cover entire image

Padding

Picture is padded with zeros so that it fits or part of the image where the filter did not fit is dropped.

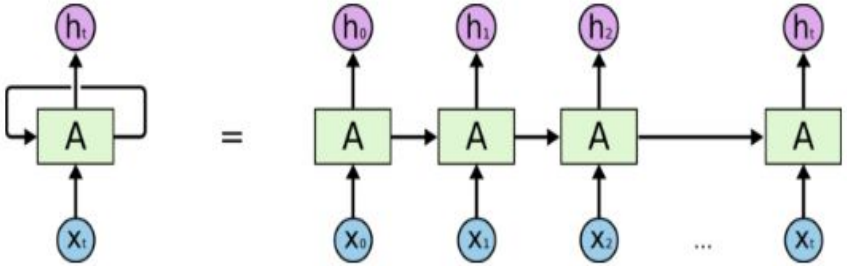
Pooling

Reduces the number of parameters when the images are too large eg: Max Pooling

Fully Connected

Matrix is flattened into vector and fed into fully connected layer like a neural network

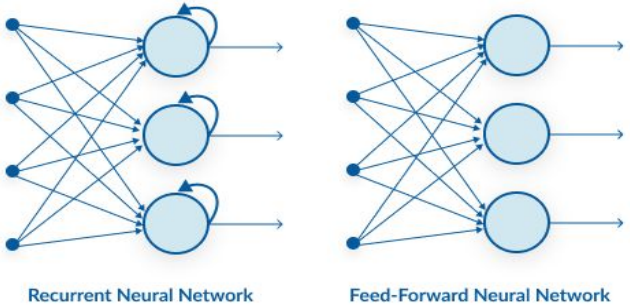
Recurrent Neural Networks



An unrolled recurrent neural network.

RNN Vs. Feed Forward Neural Network

In neural networks all the inputs are independent of each other but in RNN, all the inputs are related to each other



- Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory.
- RNN performs the same function for every input of data while the output of the current input depends on the past one computation.
- Output is sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Recurrent Neural Networks

$$h_t = f(h_{t-1}, x_t)$$

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

W is weight, **h** is the single hidden vector, **Whh** is the weight at previous hidden state, **Whx** is the weight at current input state, **tanh** is the Activation function

$$y_t = W_{hy}h_t$$

Yt is the output state. **Why** is the weight at the output state.

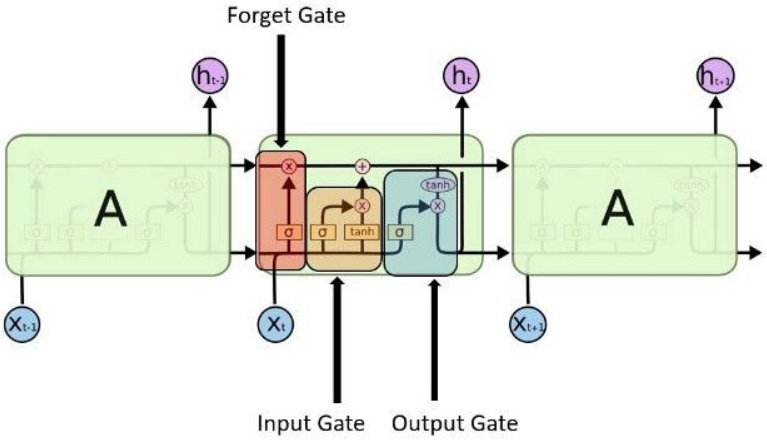
Advantages:

- RNN can model sequence of data so that each sample can be assumed to be dependent on previous ones
- Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood.

Disadvantages:

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.
- It cannot process very long sequences if using tanh or relu as an activation function.

LSTM



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- Long Short-Term Memory networks are a modified version of RNN, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved.
- LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation.

1. **Input gate** — discover which value from input should be used to modify the memory.
2. **Forget gate** — discover what details to be discarded from the block.
3. **Output gate** — the input and the memory of the block is used to decide the output.

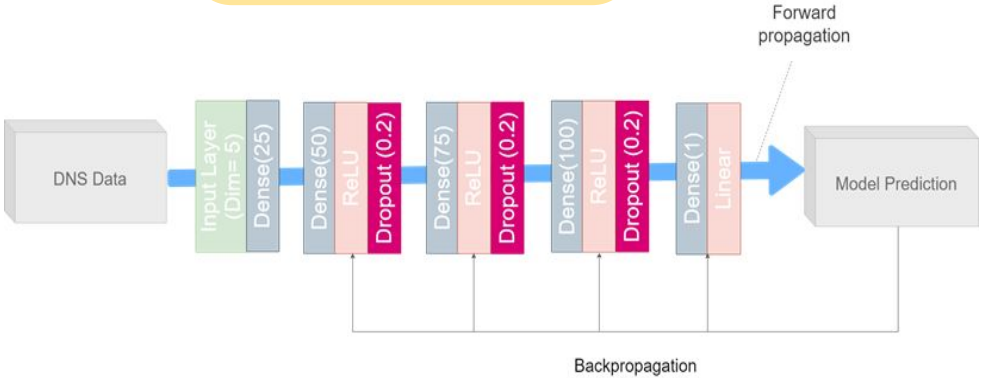
Case Study Example: Turbulence Modeling using Neural Network

Model

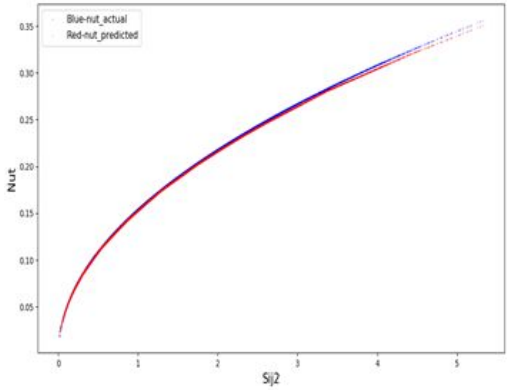
Input Parameter:
(Fluctuating rate of strain tensor)

Output Parameter:
(Eddy or Turbulent viscosity)

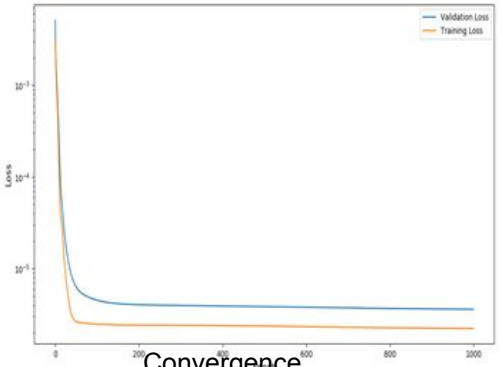
Network Used



Results



Predicted Vs. Actual



Convergence

Theoretical Properties of NNs

Convergence

Models may not consistently converge on a single solution:

1. Firstly because local minima may exist, depending on the cost function and the model
2. Secondly, the optimization method used might not guarantee to converge when it begins far from any local minimum
3. Thirdly, for sufficiently large data or parameters, some methods become impractical

- The convergence behavior of certain types of ANN architectures are more understood than others. When the width of network approaches to infinity, the ANN is well described by its first order **Taylor expansion** throughout training, and so inherits the convergence behavior of affine models.
- Another example is when parameters are small, it is observed that ANNs often fits target functions from low to high frequencies. This phenomenon is the opposite to the behavior of some well studied iterative numerical schemes such as **Jacobi method**.

Theoretical Properties of NNs

Computational Power

➔ The multilayer perceptron is a universal function approximator, as proven by the universal approximation theorem

➔ However, the proof is not constructive regarding the number of neurons required, the network topology, the weights and the learning parameters

- A specific recurrent architecture with rational-valued weights (as opposed to full precision real number-valued weights) has the power of a universal Turing machine, using a finite number of neurons and standard linear connections
- Further, the use of irrational values for weights results in a machine with super-Turing power

Theoretical Properties of NNs

Capacity

- ➔ A model's "capacity" property corresponds to its ability to model any given function
- ➔ It is related to the amount of information that can be stored in the network and to the notion of complexity
- ➔ Two notions of capacity are known by the community
- ➔ The information capacity and the VC Dimension
- ➔ The capacity of a network of standard neurons (not convolutional) can be derived by four rules that derive from understanding a neuron as an electrical element

- The information capacity captures the functions modelable by the network given any data as input
- The second notion, is the VC dimension
- VC Dimension uses the principles of measure theory and finds the maximum capacity under the best possible circumstances, given input data in a specific form
- The VC Dimension for arbitrary inputs is half the information capacity of a Perceptron
- The VC Dimension for arbitrary points is sometimes referred to as Memory Capacity

Theoretical Properties of NNs

Generalization

➔ Two approaches address over-training:

- The first is to use **cross-validation** and similar techniques to check for the presence of over-training and to select hyperparameters to minimize the generalization error.
- The second is to use some form of ***regularization***. This concept emerges in a **probabilistic (Bayesian) framework**, where regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the **'empirical risk'** and the **'structural risk'**, which roughly corresponds to the error over the training set and the predicted error in unseen data due to overfitting.

- Supervised neural networks that use a **mean squared error (MSE)** cost function can use formal statistical methods to determine the confidence of the trained model
- The MSE on a validation set can be used as an estimate for variance
- This value can then be used to calculate the confidence interval of network output, assuming a normal distribution
- A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

Using a Neural Network

Using Artificial neural networks requires an **understanding of their characteristics**:

- **Choice of model**: This depends on the data representation and the application. Overly complex models slow learning.
- **Learning algorithm**: Numerous trade-offs exist between learning algorithms. Almost any algorithm will work well with the correct hyperparameters for training on a particular data set. However, selecting and tuning an algorithm for training on unseen data requires significant experimentation.
- **Robustness**: If the model, cost function and learning algorithm are selected appropriately, the resulting ANN can become robust.

ANN capabilities fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators and prostheses.

Advantages of Neural Networks

- **Artificial Neural Network's (ANN) outputs aren't limited entirely by inputs and results given to them initially by an expert system**
  - This ability comes in handy for robotics and pattern recognition systems
- **This network has the potential for high fault tolerance and is capable of debugging or diagnosing a network on its own**
  - ANN can go through thousands of log files from a company and sort them out
  - It is presently a tedious task done by administrators
- **Nonlinear systems can find shortcuts to reach computationally expensive solutions**
  - Example: We see this in banking where they have an Excel spreadsheet, and then they start building codes around that sheet
  - In over 20 years, they might create a repertoire of all these functions, and the neural network comes up with the same answers done in days, weeks, or even a month for a large bank

Applications

Neural networks are universal approximators, and they work best if the system you are using them to model has a high tolerance to error.

Some of the applications of NN are as follows:

- capturing associations or discovering regularities within a set of patterns;
- where the volume, number of variables or diversity of the data is very great;
- the relationships between variables are vaguely understood; or,
- the relationships are difficult to describe adequately with conventional approaches

Image compression

- The idea behind the data compression neural network is to store, encrypt, and recreate the actual image again
- We can optimize the size of our data using image compression neural networks. It is the ideal application to save memory and optimize it

Applications (Cont.)

Handwriting Recognition

Neural networks are used to convert handwritten characters into digital characters that a machine can recognize

Stock-exchange prediction

- The stock exchange is difficult to track and difficult to understand. Many factors affect the stock market
- A neural network can examine a lot of factors and predict the prices daily, which would help stockbrokers

Traveling issues of sales professionals

- This type refers to finding an optimal path to travel between cities in a particular area
- Neural networks help solve the problem of providing higher revenue at minimal costs
- Logistical considerations are enormous, and here we have to find optimal travel paths for sales professionals moving from town to town

Limitations of Neural Networks

In reference to backpropagation networks, there are some specific potential issues:

- Back Propagation neural networks (and many other types of networks) are in a sense the ultimate 'black boxes'
    a. Apart from defining the general architecture of a network and initializing it with a random numbers, the **user has no other role** than to feed it input and watch it train and await the output
    b. In fact, it has been said that with backpropagation, **"you almost don't know what you're doing"**, some software freely available software packages (NevProp, bp, Mactivation) do allow the user to sample the networks 'progress' at regular time intervals, but the learning itself progresses on its own
    c. The final product of this activity is a trained network that **provides no equations or coefficients defining a relationship** (as in regression) beyond its own internal mathematics
    d. The network itself is the final equation of the relationship
- Backpropagation networks also **tend to be slower to train** than other types of networks and sometimes require thousands of epochs
    a. If run on a truly parallel computer system this issue is not really a problem, but if the Neural Network is being simulated on a standard serial machine training can take some time
    b. This is because the machine's CPU must compute the function of each node and connection separately, which can be problematic in very large networks with a large amount of data

Future of Neural Networks

- **We can expect a few intriguing discoveries on algorithms to support learning methods**
  - We are just in the infant stage of applying artificial intelligence and neural networks to the real world.
- **Neural networks will be a lot faster in the future, and neural network tools can get embedded in every design surface**
  - We already have a little mini neural network that plugs into an inexpensive processing board, or even into your laptop. Focusing on the hardware, instead of the software, would make devices even faster
- Neural networks will find its usage in the field of **medicine, agriculture, physics, discoveries**, and, neural networks are also used in **shared data systems**
- There will be **personalized choices** for users all over the world
  - All mobile and web applications try to give an enhanced customized experience based on your search history
- **Hyper-intelligent virtual assistants** will make life easier
  - Applications like Google assistant, Siri, or any of those assistants are slowly evolving
  - They may even predict the email response in the future

Summary

- The presentation covers theoretical aspects on different types of **Neural Networks**, **Network Design**, **Advantages** and **Disadvantages**, theoretical **Properties** of neural networks, **Code Execution (Keras)** and **Future of NN**.
- We have discussed types of **Activation Functions** like ReLU, Linear, Binary, Sigmoid, Tanh, Leaky ReLU
- **CNN**: CNN takes an input image and assigns importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other
- **RNN**: Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory
- We have also covered more major sectors of applications of NN and viz. image compression, OCR, sales predictions and Travel issues of professionals.
- The theoretical properties of NN mainly consists of Capacity, Computational Power, Convergence, Generalization and Statistical Relations.
- We have discussed the limitations of NN which are now a main focus of research for future developments in the field.
- Case study for **Turbulence Modelling** using Neural Networks has been discussed in depth.

References

1. Chen, Yung-Yao; Lin, Yu-Hsiu; Kung, Chia-Ching; Chung, Ming-Han; Yen, I.-Hsuan (January 2019). "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes". *Sensors*.
2. McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*.
3. Kleene, S.C. (1956). "Representation of Events in Nerve Nets and Finite Automata". *Annals of Mathematics Studies* (34). Princeton University Press.
4. Hebb, Donald (1949). *The Organization of Behavior*. New York.
5. Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". *IRE Transactions on Information Theory*.
6. Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". *Psychological Review*.
7. Werbos, P.J. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*.
8. Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". *Neural Networks*.
9. Ivakhnenko, A. G. (1973). *Cybernetic Predicting Devices*. CCM Information Corporation.
10. Ivakhnenko, A. G.; Grigor'evich Lapa, Valentin (1967). *Cybernetics and forecasting techniques*. American Elsevier Pub. Co.
11. Dreyfus "Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure". *Journal of Guidance, Control, and Dynamics*.
12. Mizutani, E.; Dreyfus, S.E.; Nishio, K. (2000). "On derivation of MLP backpropagation from the Kelley-Bryson optimal-control gradient formula and its application". *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000.*

Acknowledgement

We would like to thank **Prof. Kannan A.** for conducting this course **CH5020 Statistical Design and Analysis of Experiments** and giving us this opportunity to present external report presentation on **Neural Networks**. The lectures throughout the semester helped us understand and build a strong grasp on the subject. We'd also like to thank our Teaching Assistants and fellow coursemates who helped us throughout this entire course.

THANK YOU

Mayur Vikas Joshi | ME16B148
Sushant Uttam Wadavkar | ME16B172