Comparative Analysis On Neural Networks

By:

MUHAMMAD TALAL BIN NADEEM
HUSNAIN ASGHAR
Mahd Bin Tariq
Pirzada Abdur Rehman

TABLE OF CONTENTS

Recurrent Neural Networks

Introduction

In this section we will explore the entire working for Recurrent Neural Networks (RNN). Including an overview of what RNNs are, their activation and error functions as well as a detailed discussion on their existing sub types.

Let’s start by addressing what RNN are.

What are RNN

RNN are a type of neural network that is designed to handle computation of sequential data for example Natural Language Processing, calculation of time series, or detection of a repeating structure in a dataset (which may include handwriting, genomes, or the prediction for stocks in a stock market).

Working of layers

The architecture of a RNN closely resembles that of a Feedforward Neural Network with the addition of recurrent connections that allow information to be passed from one layer to the other. However the way it differs from a simple Feedforward network is in terms of cycles. While FNN do not have any cycles, a RNN can transmit information back to itself Hence allowing us to take the previous actions into account.

We can observe the output of a layer at step h for a RNN using the following mathematical equations.

$$Ht = \varphi h \ (XtWxh + Ht-1Whh + bh)$$

Using the output equations we can observe that since the equation recursively includes ht-1 at any given time step indicating that we use the previous results in some capacity.

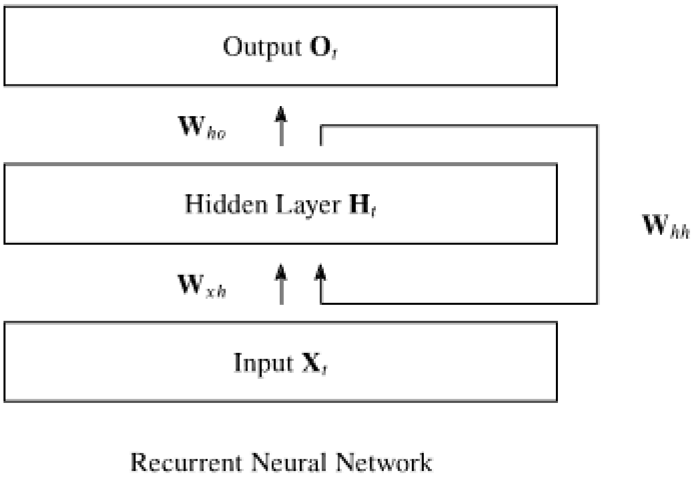


fig 1.1 At each time step, the RNN takes an input vector and produces an output vector, as well as an internal state vector that is passed on to the next time step. The internal state vector represents the "memory" of the network and allows it to maintain information about the previous inputs.

Activation Functions

RNN uses a variety of different functions. A list of activation functions is give below.

1. Tanh: The hyperbolic Tan function is a popular choice for the activation function in RNNs. It has a range between -1 and 1 and is often used in the internal state update equations of RNNs.

2. ReLU: The Rectified Linear Unit function is a popular choice for the activation function in feedforward neural networks, but it can also be used in the output layer of RNNs. It has been shown to be effective in reducing the vanishing gradient problem in deep neural networks.

3. Softmax: The softmax function is often used in the output layer of RNNs for multi-class classification problems. It converts the outputs of the network into a probability distribution over the possible classes.

4. Linear: The linear function is a simple activation function that can be used in the output layer of RNNs for regression problems. It simply outputs the weighted sum of the inputs without applying any nonlinearity.

5. Exponential Linear Units (ELUs): ELUs are a recent development in activation functions, and they have been shown to outperform other popular activation functions, such as ReLU and sigmoid, in certain types of deep neural networks, including

RNNs.

6. Sigmoid: The sigmoid function is a popular choice for the activation function in the gating mechanisms of Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). It has a range between 0 and 1 and is often used to control the flow of information through the network.

Error Functions

Like a FNN RNN's also use the concept of backpropagation through time to propagate error through it's layers. Although the choice of a error function depends on the type of problem we are presented with, some commonly used error functions are as follows.

1. Cross-Entropy: Cross-entropy is a popular choice for classification problems, where the goal is to predict a categorical output. It measures the difference between the predicted probability distribution and the true probability distribution. In RNNs, cross-entropy is often used in combination with the softmax activation function in the output layer.

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

2. Binary Cross-Entropy: Binary cross-entropy is a variant of cross-entropy that is used for binary classification problems, where the goal is to predict a binary output (e.g., true/false, 0/1). It measures the difference between the predicted probability and the true label.

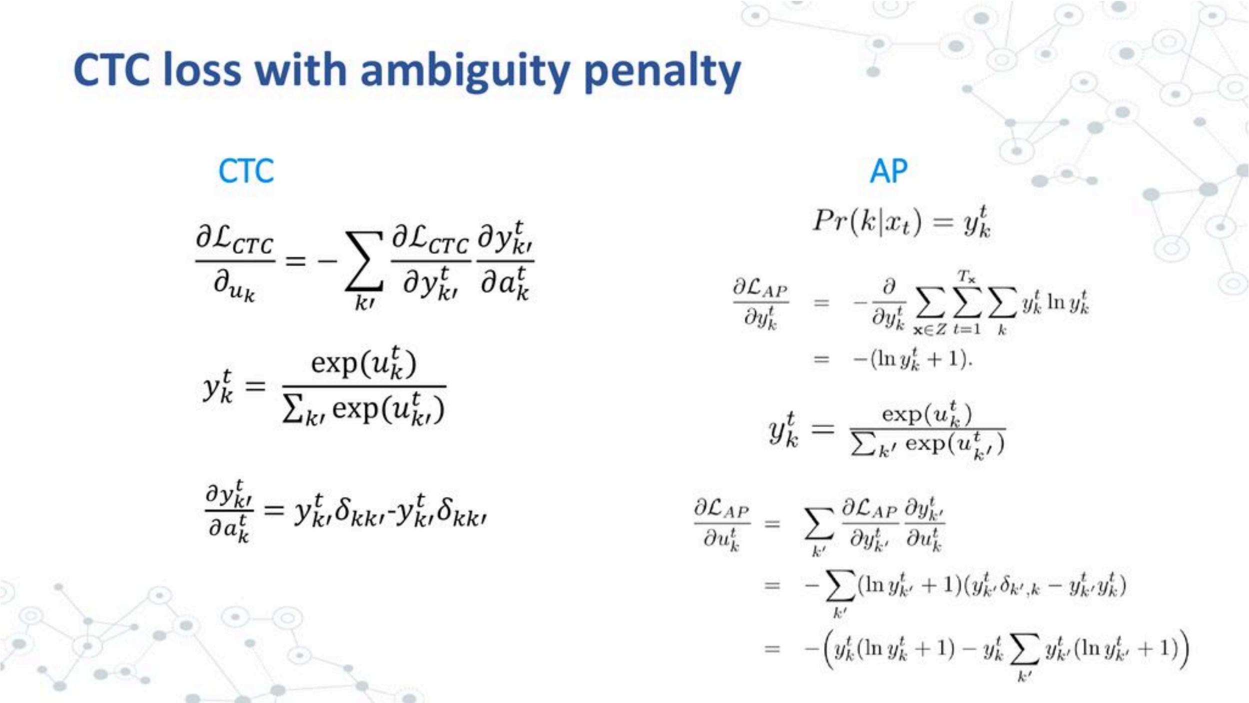
3. $$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

modeling and text generation. It measures the negative log-probability of the true output given the predicted output.

$$l(\theta) = -\sum_{i=1}^{n} \left(y_i \log \hat{y}_{\theta,i} + (1 - y_i) \log \left(1 - \hat{y}_{\theta,i}\right) \right)$$

4. Connectionist Temporal Classification (CTC) Loss: CTC is a loss function that is used for sequence labeling tasks, where the goal is to predict a sequence of labels for an input sequence. It is commonly used in speech recognition and handwriting recognition tasks, where the length of the input sequence varies.



CTC loss with ambiguity penalty

CTC

$$\frac{\partial \mathcal{L}_{CTC}}{\partial_{u_k}} = -\sum_{k\prime} \frac{\partial \mathcal{L}_{CTC}}{\partial y_{k\prime}^{t}} \frac{\partial y_{k\prime}^{t}}{\partial a_k^{t}}$$

$$y_k^t = \frac{\exp(u_k^t)}{\sum_{k\prime} \exp(u_{k\prime}^t)}$$

$$\frac{\partial y_{k\prime}^t}{\partial a_k^t} = y_{k\prime}^t \delta_{kk\prime}\text{-}y_{k\prime}^t \delta_{kk\prime}$$

AP

$$Pr(k|x_t) = y_k^t$$

$$\begin{aligned} \frac{\partial \mathcal{L}_{AP}}{\partial y_k^t} &= -\frac{\partial}{\partial y_k^t} \sum_{\mathbf{x} \in Z} \sum_{t=1}^{T_{\mathbf{x}}} \sum_{k} y_k^t \ln y_k^t \\ &= -(\ln y_k^t + 1). \end{aligned}$$

$$y_k^t = \frac{\exp(u_k^t)}{\sum_{k'} \exp(u_{k'}^t)}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_{AP}}{\partial u_k^t} &= \sum_{k'} \frac{\partial \mathcal{L}_{AP}}{\partial y_{k'}^t} \frac{\partial y_{k'}^t}{\partial u_k^t} \\ &= -\sum_{k'} (\ln y_{k'}^t + 1)(y_{k'}^t \delta_{k',k} - y_{k'}^t y_k^t) \\ &= -\Big(y_k^t (\ln y_k^t + 1) - y_k^t \sum_{k'} y_{k'}^t (\ln y_{k'}^t + 1) \Big) \end{aligned}$$

5. Mean Squared Error (MSE): MSE is a popular choice for regression problems, where the goal is to predict a continuous output value. It measures the average squared

difference between the predicted output and the true output.

The derivation for the Generic backpropagation equations can be highlighted as follows (where L_t can be taken from any one of the error functions described above.

$$\mathcal{L}(\mathbf{O}, \mathbf{Y}) = \sum_{t=1}^{T} \ell_t(\mathbf{O}_t, \mathbf{Y}_t)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{H_t}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial \mathbf{H}_t} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{hh}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{xh}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} (\mathbf{W}_{hh}^{\top})^{t-k} \cdot \mathbf{H}_k$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} (\mathbf{W}_{hh}^{\top})^{t-k} \cdot \mathbf{X}_k$$

Existing sub types

The existing subtypes we will be discussing here can be broadly categorized into three forms.

1. Deep Recurrent Neural Networks

2. Bidirectional Neural Networks

3. Sequence to Sequence

Deep Recurrent Neural Networks (DRNN)

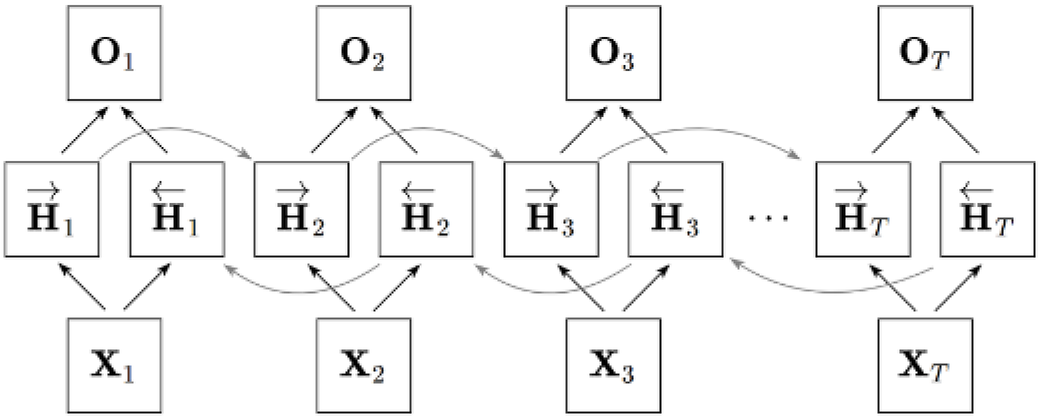
It is the simplest type of RNN out of the ones listed. Essentially you can stack different simple RNN’s with L hidden layers to achieve DRNN. Each hidden state is passed to the next step of the current layer as well as the current time step of the next layer. Mathematically this can be represented as

$$\mathbf{H}_t^{(1)} = \phi_1\left(\mathbf{X}_t, \mathbf{H}_{t-1}^{(1)}\right)$$

$$\mathbf{H}_t^{(\ell)} = \phi_\ell\left(\mathbf{H}_t^{(\ell-1)}, \mathbf{H}_{t-1}^{(\ell)}\right)$$

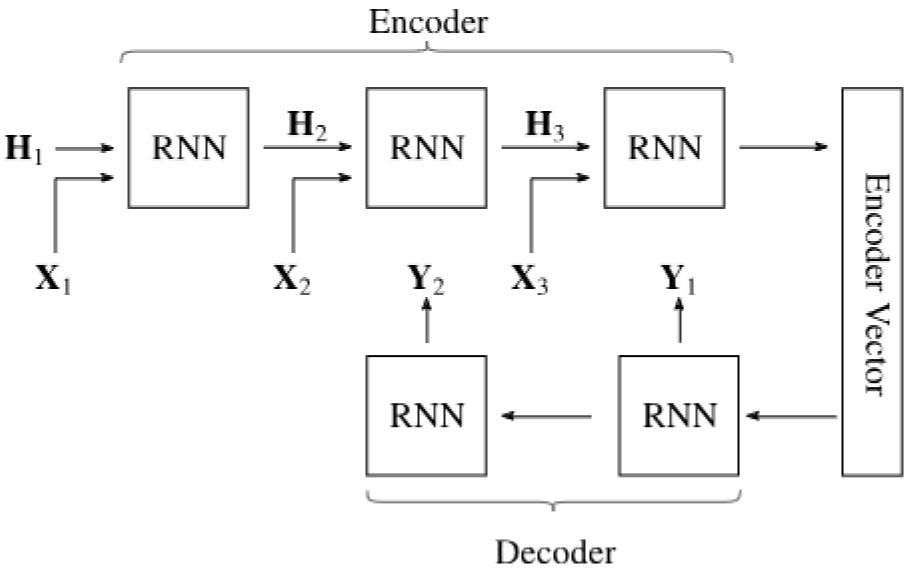
Bidirectional Recurrent Neural Network (BRNN)

In this catagoy a hidden layer is added on the additional set of layers which runs the sequence in reverse i.e. starting from the last element. A architectural view can be observed as below


$\mathbf{O}_1$
$\mathbf{O}_2$
$\mathbf{O}_3$
$\mathbf{O}_T$
$\overrightarrow{\mathbf{H}}_1$
$\overleftarrow{\mathbf{H}}_1$
$\overrightarrow{\mathbf{H}}_2$
$\overleftarrow{\mathbf{H}}_2$
$\overrightarrow{\mathbf{H}}_3$
$\overleftarrow{\mathbf{H}}_3$
$\cdots$
$\overrightarrow{\mathbf{H}}_T$
$\overleftarrow{\mathbf{H}}_T$
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_3$
$\mathbf{X}_T$

Sequence to Sequence

The sequence to sequence RNN can be divided into two parts. The encoder structure and the decoder structure. In the encoder structure we have a RNN accepting a single element x_t , this structure aims to aggregate all of the information from the previous input elements which are then passed on to the decoder structure which makes the actual predictions for our network. An architectural overview is as follows



Examples Of RNN

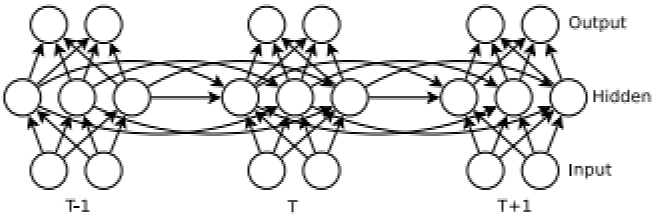
1.Generating Texts with Random Neural Networks

Ilya Sutskever ILYA@CS.UTORONTO.CA
James Martens JMARTENS@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, ON M5S 3G4 CANADA

Link:https://icml.cc/2011/papers/524_icmlpaper.pdf

The goal of this paper was to demonstrate the power of large RNNs trained with the new Hessian-Free optimizer by using this method to predict the next character in the stream of the text presented to the model.This will result in reduction in text file making it easier for people with physical disabilities to interact with computers.

Although RNN is pretty expressive the authors decided to use MRNN architecture of their own design allowing character input to determine output by using hidden to hidden weight matrix.The data was trained over a 128 Gb data set using 8 graphic processing units.The end result of the trained model was capable of forming complex words along with a large variety of options to choose from. The MRNNs nonlinear dynamics enables it to extract higher level “knowledge” from the text, and there are no obvious limits to its representational power because of the ability of its hidden states to perform general computation.



RNN uses the sequence of its output vectors (o1, . . . , oT) to obtain a sequence of predictive distributions P(xt+1|x≤t) = softmax(ot), where the softmax distribution is defined by P(softmax(ot) = j) = exp(o (j) t)/ P k exp(o (k) t).

For Training the mrnn required to process the entire training set sequentially and store the hidden state sequence.If the training sequence becomes too small HF cannot be utilized to capture long term dependencies spanning hundreds of time steps.

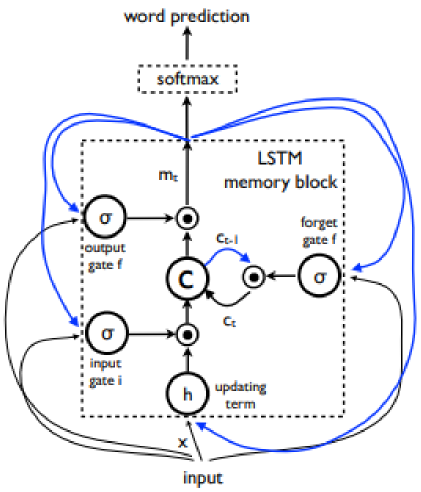
MRNNs already learn surprisingly good language models using only 1500 hidden units, and unlike other approaches such as the sequence memoizer and PAQ, they are easy to extend along various dimensions. If we could train much bigger MRNNs with millions of units and billions of connections, it is possible that brute force alone would be sufficient to achieve an even higher standard of performance. But this will of course require considerably more computational power.

2 Neural Image Caption Generator

Oriol Vinyals Google vinyals@google.com
Alexander Toshev Google toshev@google.com
Samy Bengio Google bengio@google.com
Dumitru Erhan Google dumitru@google.com
Link:https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vinyals_Show_and_Tell_2015_CVPR_paper.pdf

The objective of this project was to properly describe images with proper sentences as a way to provide humans with impaired vision to be able to identify the image properly.This is a bit more complex than simple use of RNN to identifying shapes in multiple images as complex captions will be generated to go along with the images provided to the AI. The main inspiration of thei work comes from recent advances in machine translation, where the task is to transform a sentence S written in a source language, into its translation T in the target language, by maximizing p(T|S).

The objective of the team is to propose a neural and probabilistic framework to generate descriptions from images. Recent advances in statistical machine translation have shown that, given a powerful sequence model, it is possible to achieve state-of-the-art results by directly maximizing the probability of the correct translation given an input sentence in an “end-to-end” fashion – both for training and inference.



One of the major issues was over-fitting which was mainly combated by setting the weights of the RNN component to using preset weights from previous experiments conducted.

3. Neural Conversational Model

Oriol Vinyals VINYALS@GOOGLE.COM
Google
Quoc V. Le QVL@GOOGLE.COM
Google
Link to paper:https://arxiv.org/pdf/1506.05869.pdf

In this paper, we present a simple approach for this task which uses the recently proposed sequence to sequence framework. Our model converses by predicting the next sentence given the previous sentence or sentences in a conversation. The strength of our model is that it can be trained end-to-end and thus requires much fewer hand-crafted rules.

Due to current advancements in neural networks we do not require that much feature engineering which results in researchers being able to work in domains for which information may not be readily available.

Makes use of the sequence-to-sequence (seq2seq) framework described in (Sutskever et al., 2014).During training, the true output sequence is given to the model, so learning can be done by backpropagation. The model is trained to maximize the cross entropy of the correct sequence given its context.

A closed-domain IT helpdesk troubleshooting dataset and an open-domain movie transcript datasets were used for training of the neural conversational model

Below is a sample of the conversation held
Machine: hi
Human: hello
Machine: could you please let me know what operating systems you are using ? Human: i am using linux
Machine: ok it was in the ssl vpn on which you stated it was broken
Machine: please check if you are logged in to corp network
Machine: what is the error that you are running please Human: i am seeing an error related to vpn

Results show that it can generate simple and basic conversations, and extract knowledge from a noisy but open-domain dataset.

Conclusion

RNNs are an effective method for handling sequential data and have been effectively used for a variety of tasks in speech recognition, natural language processing, and other fields. They can be difficult to train, as they call for exact architecture and optimisation of parameters. Alternatives to RNNs that may be more effective or efficient for specific sorts of jobs have also been made available by recent advances in deep learning, such as attention mechanisms and transformer networks.

References

- Abiodun, Oludare Isaac; Jantan, Aman; Omolara, Abiodun Esther; Dada, Kemi Victoria; Mohamed, Nachaat Abdelatif; Arshad, Humaira (2018-11-01). "State-of-the-art in artificial neural network applications: A survey". Heliyon. **4** (11): e00938. doi:10.1016/j.heliyon.2018.e00938. ISSN 2405-8440. PMC 6260436. PMID 30519653.

- Tealab, Ahmed (2018-12-01). "Time series forecasting using artificial neural networks methodologies: A systematic review". Future Computing and Informatics Journal. **3** (2): 334–340. doi:10.1016/j.fcij.2018.10.003. ISSN 2314-7288.

- Graves, Alex; Liwicki, Marcus; Fernandez, Santiago; Bertolami, Roman; Bunke, Horst; Schmidhuber, Jürgen (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. **31** (5): 855–868. CiteSeerX 10.1.1.139.4502. doi:10.1109/tpami.2008.137. PMID 19299860. S2CID 14635907.

- Sak, Haşim; Senior, Andrew; Beaufays, Françoise (2014). "Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling" (PDF).

- Li, Xiangang; Wu, Xihong (2014-10-15). "Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition". arXiv:1410.4281 [cs.CL].

- Hyötyniemi, Heikki (1996). "Turing machines are recurrent neural networks". Proceedings of STeP '96/Publications of the Finnish Artificial Intelligence Society: 13–24.

- Miljanovic, Milos (Feb–Mar 2012). "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction" (PDF). Indian Journal of Computer and Engineering. **3** (1).

- Lenz, W. (1920), "Beiträge zum Verständnis der magnetischen Eigenschaften in festen Körpern", Physikalische Zeitschrift, **21**: 613–615.

- Ising, E. (1925), "Beitrag zur Theorie des Ferromagnetismus", Z. Phys., **31** (1): 253–258, Bibcode:1925ZPhy...31..253I, doi:10.1007/BF02980577, S2CID 122157319

- Dupond, Samuel (2019). "A thorough review on the current advance of neural network structures". Annual Reviews in Control. **14**: 200–230.

Convolutional Neural Network (CNN)

Introduction:

Convolutional Neural Networks (CNNs) are one of deep learning algorithm used to analyze and interpret images and other multi-dimensional data. CNNs have been used in many applications, including image classification, object detection, facial recognition, and natural language processing.

This section will provide a comprehensive overview of the architecture of CNN, its different layers, activation and error functions, existing sub-types, specialized application areas, and types of problems solved through CNN.

Architecture of CNN:

The architecture of a CNN consists of several layers, including the input layer, convolutional layer, pooling layer, fully connected layer, and output layer.

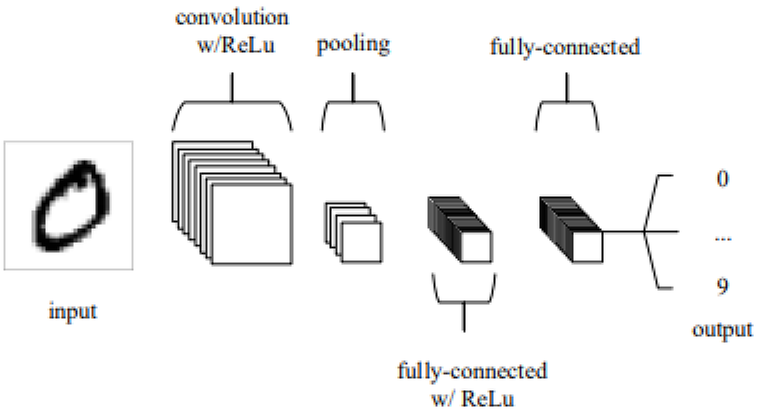


Fig. 2: An simple CNN architecture, comprised of just five layers

Input Layer:

The input layer is responsible for accepting the input data, which can be an image or any other multi-dimensional data.

Convolutional Layer:

The convolutional layer is responsible for extracting features from the input data using convolutional filters or kernels. The filters are applied to the input data, and the resulting output is passed to the next layer.

Pooling Layer:

The pooling layer is responsible for reducing the dimensionality of the feature maps generated by the convolutional layer. This is done to reduce the computational complexity of the network and prevent overfitting.

Fully Connected Layer:

This layer is responsible for learning the relationship between the features extracted by the convolutional and pooling layers and the output. This layer contains neurons that are connected to all the neurons in the previous layer.

Output Layer:

The output layer is responsible for providing the final output of the network. The output can be a classification or regression problem.

Activation and Error Functions:

Activation functions are used in neural networks to introduce non-linearity into the model. This is important because many real-world problems are non-linear in nature. The activation function is applied to the output of the neuron in a layer and determines whether the neuron should fire or not.

The most commonly used activation functions are tanh, sigmoid and ReLU.

Error functions, also known as loss functions or objective functions, are an important component of Convolutional Neural Networks (CNNs). Error functions are used to calculate how different the predicted output of a CNN is from the actual output. They give a numerical value that tells us how well the model is performing.

The most commonly used error function for classification tasks is the cross-entropy loss function, while mean squared error is used for regression tasks. Other error functions, such as hinge loss and cosine similarity, may be used for specific tasks. The choice of error function can have a significant impact on the performance of a CNN, and selecting the appropriate function is often a matter of trial and error.

Existing sub-types of CNN:

There are several sub-types of CNN, including:

LeNet-5: This is one of the first CNN architectures developed by Yann LeCun in 1998 for recognizing handwritten digits.

AlexNet: This is a deep CNN developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012, which achieved state-of-the-art performance on the ImageNet dataset.

VGGNet: This is a deep CNN developed by Karen Simonyan and Andrew Zisserman in 2014, which achieved better performance than AlexNet on the ImageNet dataset.

ResNet: This is a deep CNN developed by Kaiming He, Shaoqing Ren, and Jian Sun in 2015, which introduced residual connections to prevent the vanishing gradient problem.

InceptionNet: This is a deep CNN developed by Christian Szegedy, Wei Liu, Yangqing Jia, Pierre S rinivasan, Scott Reed, Dragomir Anguelov, and Andrew Rabinovich in 2014, which introduced the Inception module for better feature extraction.

MobileNet: This is a lightweight CNN developed by Andrew Howard, Menglong Zhu, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam in 2017, which is designed to run efficiently on mobile devices.

Specialized Application areas of CNN:

CNNs have been applied in various fields, including:

Computer Vision: CNNs are used in computer vision applications, such as face recognition, image segmentation and object detection.

Natural Language Processing: CNNs are used in natural language processing applications, such as sentiment analysis, machine translation, and text classification.

Robotics: CNNs are used in robotics applications, such as object detection and recognition, visual navigation, and robotic control.

Medical Imaging: CNNs are used in medical imaging applications, such as tumor detection, segmentation, and classification.

Different types of problems solved through CNN:

CNNs can be used to solve various problems, including:

Image Classification: CNNs can classify images into different categories.

Object Detection: CNNs can detect objects in an image and determine their location.

Semantic Segmentation: CNNs can segment an image into different regions based on their semantic meaning.

Image Captioning: CNNs can generate captions for images.

Style Transfer: CNNs can transfer the style of one image onto another image.

References:

1. Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. "ImageNet Classification with Deep Convolutional Neural Networks." In Advances in Neural Information Processing Systems, pp. 1097-1105. 2012.
2. LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324. 1998.

3. Simonyan, Karen, and Zisserman, Andrew. "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv preprint arXiv:1409.1556. 2014.
4. He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. "Deep Residual Learning for Image Recognition." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778. 2016.
5. Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, ... Rabinovich, Andrew. "Going deeper with convolutions." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9. 2015.
6. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
7. Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. Jama, 316(22), 2402-2410.
8. Sure, here are some article links and references related to the previous information:
9. - "A Beginner's Guide to Convolutional Neural Networks (CNNs)" by Jason Brownlee, available at: https://machinelearningmastery.com/convolutional-neural-networks-for-image-classification/
10. - "Convolutional Neural Networks (CNNs) for Visual Recognition" by Stanford University, available at: http://cs231n.stanford.edu/slides/2021/lecture_5.pdf
11. - "Overview of Convolutional Neural Networks" by Anirban Santara, available at: https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/
12. - "Convolutional Neural Networks in Medical Image Analysis: An Overview" by Muhammad Asim and Akhtar Ali, available at: https://www.mdpi.com/2079-9292/8/11/1196/pdf
13. - "An Introduction to Convolutional Neural Networks" by Vincent Dumoulin and Francesco Visin, available at: https://arxiv.org/abs/1511.08458
14. - "A Comprehensive Study on Convolutional Neural Network for Plant Disease Identification" by Sudhakar Kumawat and Goutam Saha, available at: https://ieeexplore.ieee.org/document/8641633
15. - "A Deep Learning Model for Real-Time Skin Lesion Detection in Mobile Applications" by Ahmed Elsawah, AbdelRahman Taha, and Mohamed Loey, available at: https://ieeexplore.ieee.org/document/9013218

Artificial Neural Network (ANN)

Introduction to Artificial Neural Network (ANN):

Artificial neural networks (ANNs) are computational models that are inspired by the structure and function of the biological neural networks found in the human brain. The interconnected nodes in ANNs are called artificial nodes or neurons and these nodes process and transmit information throughout the network

1.**Architecture of ANN:**

. The architecture of ANNs consists of three main types of layers that are input,hidden and output layers.

a) Input layer: The input layer is responsible for receiving the input data and passing it to the hidden layers. It has one neuron for each feature in the input data.

b) Hidden layers: The hidden layers are responsible for transforming the input data into a useful representation for the output layer. The complexity of the problem depends on the number of hidden layers and neurons in each layer.

c) Output layer: The output layer is responsible for generating the final output of the network. The type of problem depends on the number of neurons in the output layer.

Activation Functions:

a) Sigmoid: The sigmoid activation function takes any input value and maps it to a value between 0 and 1.
It is expressed mathematically as f(x) = 1 / (1 + exp(-x)).
In the output layer of binary classification problems, where the goal is to predict a probability of belonging to one of the two classes,at these layers the sigmoid function is commonly used.

b) Tanh (Hyperbolic Tangent): Any input value is taken and mapped to a value between -1 and 1 by the hyperbolic tangent. It is expressed mathematically as **f(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x))**. In hidden layers of neural networks, Tanh is commonly used as an activation function.

c) Softmax: In the output layer of multi-class classification problems the softmax activation function is commonly used. The input vector is normalized to a probability distribution over the classes by the softmax activation function. It is expressed mathematically as **f(x)_i = exp(x_i) / sum(exp(x_j))**, where x is the input vector and i is the index of the output class.

d) ReLU (Rectified Linear Unit):All negative input values are set to zero and all positive values are left unchanged by the ReLU activation function.. It is expressed mathematically as **f(x) = max(0, x).**
It provides better training performance and faster convergence compared to other activation functions so that is why ReLU is a commonly used activation function in deep neural networks.

e) Leaky ReLU: It is similar to ReLU. But instead of setting negative input values to zero, it multiplies them by a small constant value.
It is expressed mathematically as **f(x) = max(ax, x)**, where a is a small constant (e.g., 0.01).

f) ELU (Exponential Linear Unit): The ELU activation function is also quite similar to ReLU, but in ELU an exponential function is used for negative input values. It is expressed mathematically as **f(x) = {x if x > 0, a(exp(x) - 1) if x <= 0}**, where a is a small constant (e.g., 1).

Error Functions:

a) Mean Squared Error (MSE): The average squared difference between the predicted output and the actual output is measured by the mean squared error function.In regression problems mean squared error is used as error function. . It is expressed mathematically as **E(y, y') = (1 / 2) * (y - y')^2**, where y is the actual output and y' is the predicted output.

b) Huber Loss: The Huber loss function is a hybrid of the mean squared error and the mean absolute error, and it is less prone to being influenced by extreme values compared to the mean squared error. Mathematically, it is defined as **E(y, y') = 1/2 * z^2** when |z| is less than or equal to delta, or delta * (|z| - 0.5 * delta) when |z| is greater than delta, where z is equal to the difference between y and y', and delta is a small constant.

c) Categorical Cross-Entropy Loss: In multi-class classification problems, where the output is of categorical type, the categorical cross-entropy loss function is utilized. This function quantifies the dissimilarity between the anticipated output and the factual output, which is represented as a one-hot encoded vector.
The mathematical expression used to compute this function is **E(y, y') = -sum(y * log(y'))**, where y denotes the factual output, and y' denotes the predicted output.

d) Binary Cross-Entropy Loss:The binary cross-entropy loss function is employed in situations where there are only two possible outcomes (0 or 1) for the output in binary classification problems. It evaluates the dissimilarity between the anticipated output and the factual output, which is represented as a probability value. The formula used to compute it is **E(y, y') = -[y * log(y') + (1 - y) * log(1 - y')]**, where y stands for the factual output, and y' is the predicted output.

e) Hinge Loss:In binary classification problems utilizing support vector machines (SVMs), the hinge loss function is widely adopted. This function gauges the gap between the predicted output and the factual output, but it solely punishes the model if the prediction is erroneous by a specific limit. The mathematical expression of this function is **E(y, y') = max(0, 1 - y * y')**, where y stands for the factual output, and y' denotes the anticipated output.

f) Squared Hinge Loss: The squared hinge loss function is a modified version of the hinge loss function that enforces harsher penalties on bigger errors. It is frequently employed in binary classification problems that use SVMs. The formula used to calculate this function is **E(y, y') = max(0, 1 - y * y')^2**, where y represents the factual output, and y' represents the predicted output.

2. **Subtypes of ANN:**

There are several subtypes of ANNs, each designed for specific types of problems. Some of the most commonly used subtypes are:

a) Recurrent Neural Networks:Recurrent neural networks (RNNs) are created specifically to process sequential data in which the current output is contingent on the previous outputs. These networks are frequently adopted in natural language processing (NLP) and speech recognition applications.

b)Convolutional Neural Networks:Convolutional neural networks (CNNs) are engineered particularly to process image and video data. They utilize convolutional layers to extract the characteristics of the input data and pooling layers to reduce the dimensions of the data. These networks are typically adopted in tasks such as image classification and object detection.

c) Feedforward Neural Networks:The most elementary form of artificial neural networks (ANNs) is the feedforward neural networks. They consist of a single input layer, one or more hidden layers, and a sole output layer. These networks are typically utilized to address classification and regression problems.

d) Echo State Networks (ESNs): ESNs are a type of recurrent neural network that are designed to be easy to train and computationally efficient. They consist of a "reservoir" of recurrently connected neurons that are randomly initialized, and a linear readout layer that is trained to perform a specific task.

e) Liquid State Machines (LSMs): LSMs are a type of recurrent neural network that are designed to simulate the behavior of a large network of neurons. They use a "liquid" layer of randomly connected neurons as a reservoir, which is then trained to perform a specific task.

f) Deep Belief Networks (DBNs): Deep belief networks (DBNs) are a specific form of feedforward neural networks that are developed to acquire hierarchical representations of input data. They contain several layers of restricted Boltzmann machines (RBMs), which are unsupervised learning models that learn a probability distribution over their inputs

g) Autoencoders: Autoencoders are a class of neural networks employed in unsupervised learning. They are composed of an encoder network that maps the input data to a lower-dimensional latent space and a decoder network that maps the latent space back to the original input space.

h) Spiking Neural Networks (SNNs): Spiking neural networks (SNNs) are a specific form of neural networks that simulate the functionality of biological neurons more closely than other types of neural networks. They use spikes or discrete events to represent the firing of neurons.

3. **Specialized Application areas of ANN:**

Artificial Neural Networks (ANNs) are versatile and have numerous applications in various fields due to their ability to learn and generalize from data. They are used in specialized areas such as:

a) Speech Recognition: ANNs are used for tasks such as speaker identification, speech synthesis, and speech recognition.

b) Robotics: ANNs are used for tasks such as motion planning, object detection, and control in robotics.

c) Computer Vision: ANNs are utilized for tasks such as object detection, image classification, and facial recognition.

d) Natural Language Processing: ANNs are used for tasks such as text classification, machine translation, and sentiment analysis.

4. **Types of problems solved through ANN:**

Artificial neural networks (ANNs) are versatile tools that can be applied to solve various types of problems, such as:

a) Classification: ANNs can be used to classify data into different categories, such as identifying whether a bank loan should be approved or not.

b) Reinforcement Learning: ANNs can be used to learn optimal actions in an environment through trial and error, such as training an agent to play a video game.
c)Regression: ANNs can be used to predict continuous values, such as the stock price based on historical data.

d) Clustering: ANNs can be used to group similar data points together, such as clustering patients based on their medical records.

e) Time Series Prediction: ANNs can be used to predict future values in a time series, such as stock prices or weather patterns.

f) Image and Speech Recognition: Artificial neural networks (ANNs) have proved to be a powerful tool in recognizing and classifying images and speech. In computer vision, ANNs are widely used for various tasks such as object recognition, face recognition, image segmentation, and even self-driving cars. Similarly, in speech processing, ANNs are used for speech recognition, speech synthesis, and speaker identification. ANNs are capable of learning complex patterns in images and speech data, making them effective in solving many real-world problems.

g)Natural Language Processing: Tasks in natural language processing (NLP), such as sentiment analysis, text classification, and machine translation, can be accomplished using ANNs.

h) Recommender Systems: Personalized recommendation systems can be built using ANNs to suggest products or services to users based on their past interactions and preferences.

i) Anomaly Detection: Anomalies in data can be detected using ANNs, which can be used for tasks such as identifying fraudulent credit card transactions or detecting equipment failures in industrial systems.

Projects :

1:Email Spam Detection

Thashina Sultana, K A Sapnaz, Fathima Sana, Mrs. Jamedar Najath

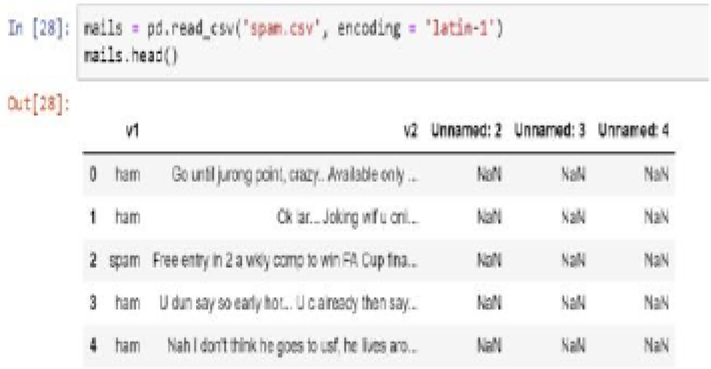
Dept. of Computer Science and Engineering

Yenepoya Institute of Technology

As a big number of people nowadays rely primarily on emails to conduct business or other daily purchases or transactions many people have created malware or just spam mail promoting their services or products completely unrelated to the user in question resulting in either poor internet performance in the best case or theft of sensitive data in a worst case scenario.

So to combat this issue the authors of the paper have suggested using Artificial intelligence to identify and get rid of spam email. This will be achieved by using machine learning to train the machine to identify common spam mails by identifying patterns of repetitive keywords which are then later classified as spam.

In the proposed model ,the web application is done using dot net and spam detection is done using machine learning.
For training the dataset from kaggle has been used



In [28]:
```
mails = pd.read_csv('spam.csv', encoding = 'latin-1')
mails.head()
```

Out[28]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Some of the columns will be removed as they are not directly related to the training of the machine in detection of spam email. Then the data is split into training sets and testing sets before being presented to the machine.

Then the message will be detected as spam or not using Bayes' theorem and Naive Bayes' Classifier so keywords will be selected from mail and then put through the classifier to filter out spam emails.

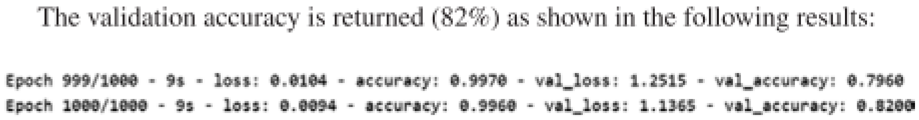
More than 270 billion emails are exchanged daily, about 57% of these are just spam emails.So the successful implementation of this model will stop a significant portion of the spam mail from being fruitful.

1:Keras and TensorFlow: A Hands-On Experience

Authors: Ferdin Joe John Joseph,Sarayut Nonsiri, and Annop Monsakul

The main work is the implementation of keras along with tensor flow.With the prerequisites achieved emphasis is placed on using keras library in deep learning. Along with this the implementation of Keras the list of gpus required for its support are also listed.
There are multiple examples of implementation of neural networks one of them is listed below.

This will explore how to identify between dog and cat images.This will be a binary data set on which the model will be trained to determine whether it's a cat or a dog.



The validation accuracy is returned (82%) as shown in the following results:

```
Epoch 999/1000 - 9s - loss: 0.0104 - accuracy: 0.9970 - val_loss: 1.2515 - val_accuracy: 0.7960
Epoch 1000/1000 - 9s - loss: 0.0094 - accuracy: 0.9960 - val_loss: 1.1365 - val_accuracy: 0.8200
```

References:

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Chollet, F. (2018). Deep Learning with Python. Manning Publications.
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- "Understanding Loss Functions in Neural Networks" by Sumit Saha: https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718
- "Activation Functions in Neural Networks" by Prashant Gupta: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
- "Artificial Neural Networks in Real-Life Applications: A Survey" by Dipti Srinivasan and S. M. Nandakumar: https://ieeexplore.ieee.org/document/8293857
- "Introduction to Deep Learning: A Primer" by Aaron S. Jackson and Adrian Bulat: https://arxiv.org/abs/1910.03540