



**ADUSUMILLI GOPALAKRISHNAIAH & SUGARCANE GROWERS
SIDDHARTHA DEGREE COLLEGE OF ARTS & SCIENCE**

Vuyyuru-521 165, Krishna District, Andhra Pradesh

An Autonomous College in the Jurisdiction of Krishna University

Accredited by NAAC with "A" Grade



DEPARTEMENT OF COMPUTER SCIENCE

VAD COURSE :DEEP LEARNING
VAD CODE: DLVAC01
DURATION :30 DAYS



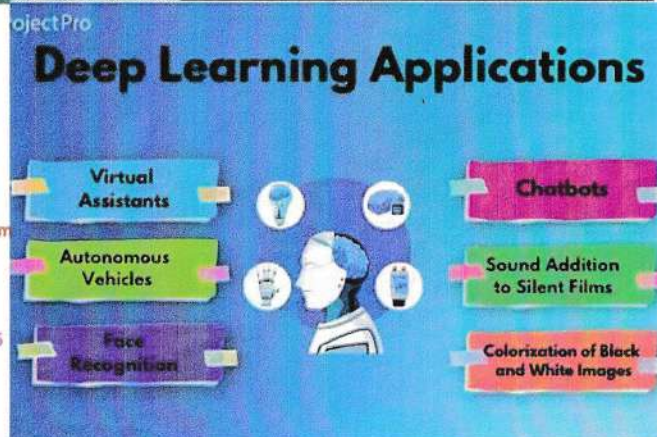
Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost.

Contact Us

Phone: 08676-233267

Email: aggsiddhartha@gmail.com

Address:
Door No 2.391
College Road, Near Kota
Complex, Vuyyuru-521165



A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh
(Managed by: Siddhartha Academy of General & Technical Education,
Vijayawada-10)

An Autonomous College in the Jurisdiction of Krishna University
Accredited by NAAC with "A" Grade ISO 9001:2015 Certified Institution



DEPARTMENT OF COMPUTER SCIENCE

Value Added Course

Title: Deep Learning

Name of the Lecturer	:	Teja Sri. Oleti
Class	:	II MSCS
Duration of the Course:		30 HOURS
VAC Code	:	DLVAC01

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course

Title: Deep Learning

- Objectives:**
- 1) Deep learning eliminates some of data Pre-processing that is typically involved With machine learning.
 - 2) Discuss the terminology used
 - 3) These algorithms can be ingest and process Unstructured data like text and images

Methodology: Teacher - Cantered method

Duration: 30 Hours

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course

Title: Deep Learning

Date : From **to**

Date	Content	Module No.
15-03-2023	<u>Deep learning environment</u> over view of deep learning , deep learning environment setup locally – installing tensor flow , installing keras , run tensor flow program on aws cloud platform	I
29-03-2023	<u>Introduction to neural network</u> What is neural network , how neural networks work , gradient descent , perceptron , multilayer perceptron , back propagation	II
5-04-2023	<u>Tensor flow basics</u> Placeholders in tensor flow-defining placeholders , feeding placeholders with data , variables , constants , computation graph	III
19-04-2023	<u>Activation functions</u> What are the activation functions , sigmoid functions , hyperbolic tangent function , Relu- rectified linear units , softmax function	IV

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course
Student Enrolment Sheet

Class: II Bs.c(MSCS)

S. No	Roll No.	Name of the Student	Signature
1	2155301	Krishnavarapu Dharanisri	K. Dharanisri
2	2155302	Ghantasala Divya	G. Divya
3	2155303	Mopidevi Sri Lakshmi	M. Sri lakshmi
4	2155304	Singavarapu Sai Sowmya	S. Sai sowmya
5	2155305	Arikatla Susanth	A. susanth
6	2155306	Bathina Manoj Phanindra	B. Manoj phanindra
7	2155307	Jampana Keerthi Priya	J. Keerthi Priya
8	2155308	Konatham Alekhya	K. Alekhya
9	2155309	Kondaraju Ajith Kumar	K. A. kumar
10	2155310	Nakka Anusha	N. Anusha
11	2155311	Akula Chakradhar	A. chakradhar
12	2155312	Mohammad Khadeera begum	M. K. begum
13	2155313	Bandela Pavan Kumar	B. Pavan kumar
14	2155314	Jonna Jhansi Lakshmi	Jonna Jhansi Lakshmi
15	2155315	Kunapareddy Tulasi	K. Tulasi

S. No	Roll No.	Name of the Student	Signature
16	2155316	Peddiboyina Himasri	P. Himasri
17	2155317	Katta Naga Sravani	K. Naga Sravani
18	2155318	Valluri Shainy	valluri shainy
19	2155319	Manikonda Karuna Sri	M. Karuna Sri
20	2155320	Padmanabhuni Phani Supraja	P. Phani Supraja
21	2155321	Kunapareddy Hema sri	Kunapareddy Hema sri
22	2155322	Reddy Durga Bhavani	R. Durga Bhavani
23	2155323	Dokku Naga Gireesha	D. Naga Gireesha
24	2155324	Nerusu Naga Mounika	N. naga mounika
25	2155325	Goriparthi Dedeepya	G. Dedeepya
26	2155326	Edupuganti Jashimani	E. Jashimani
27	2155327	Gangisetty Yuva Kiran	G. Yuva Kiran
28	2155328	Veerla Sri Lakshmi	V. Sri Lakshmi
29	2155329	Mamidi Chaitanya	M. Chaitanya
30	2155330	Rachuri Bobby	R. Bobby
31	2155331	Peteti Praneeth Kumar	P. Praneeth Kumar
32	2155332	Mohammad Abrar Ahmad	MD. Abrar Ahmad
33	2155333	Goriparthi Harika	G. Harika
34	2155334	Vinnakota Deepthi	V. Deepthi

Signature of Teacher

Signature of HOD

Signature of Principal

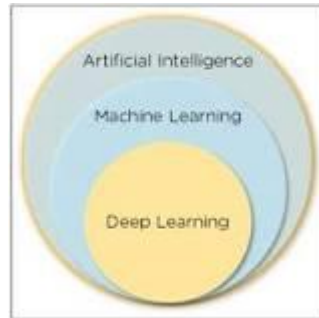
DEEP LEARNING

Deep learning environment:

Over view of deep learning:

What is the overview of deep learning?

Deep learning is a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain. Deep learning is a subset of machine learning, which is a part of artificial intelligence (AI). Artificial intelligence is the ability of a machine to imitate intelligent human behaviour.



Why is deep learning important?

Artificial intelligence (AI) attempts to train computers to think and learn as humans do. Deep learning technology drives many AI applications used in everyday products, such as the following:

- Digital assistants
- Voice-activated television remotes
- Fraud detection
- Automatic facial recognition

It is also a critical component of emerging technologies such as self-driving cars, virtual reality, and more.

Deep learning models are computer files that data scientists have trained to perform tasks using an algorithm or a predefined set of steps. Businesses use deep learning models to analyze data and make predictions in various applications.

What are the uses of deep learning?

Deep learning has several use cases in automotive, aerospace, manufacturing, electronics, medical research, and other fields. These are some examples of deep learning:

- Self-driving cars use deep learning models to automatically detect road signs and pedestrians.
- Defence systems use deep learning to automatically flag areas of interest in satellite images.
- Medical image analysis uses deep learning to automatically detect cancer cells for medical diagnosis.
- Factories use deep learning applications to automatically detect when people or objects are within an unsafe distance of machines.

You can group these various use cases of deep learning into four broad categories—computer vision, speech recognition, natural language processing (NLP), and recommendation engines.

Computer vision

Computer vision is the computer's ability to extract information and insights from images and videos. Computers can use deep learning techniques to comprehend images in the same way that humans do. Computer vision has several applications, such as the following:

- Content moderation to automatically remove unsafe or inappropriate content from image and video archives
- Facial recognition to identify faces and recognize attributes like open eyes, glasses, and facial hair
- Image classification to identify brand logos, clothing, safety gear, and other image details

Speech recognition

Deep learning models can analyze human speech despite varying speech patterns, pitch, tone, language, and accent. Virtual assistants such as Amazon Alexa and automatic transcription software use speech recognition to do the following tasks:

- Assist call center agents and automatically classify calls.
- Convert clinical conversations into documentation in real time.
- Accurately subtitle videos and meeting recordings for a wider content reach.

Natural language processing

Computers use deep learning algorithms to gather insights and meaning from text data and documents. This ability to process natural, human-created text has several use cases, including in these functions:

- Automated virtual agents and chatbots
- Automatic summarization of documents or news articles
- Business intelligence analysis of long-form documents, such as emails and forms
- Indexing of key phrases that indicate sentiment, such as positive and negative comments on social media

Recommendation engines

Applications can use deep learning methods to track user activity and develop personalized recommendations. They can analyze the behavior of various users and help them discover new products or services. For example, many media and entertainment companies, such as Netflix, Fox, and Peacock, use deep learning to give personalized video recommendations.

Deep learning environment setup locally – installing tensorflow

HOWTO: Install Tensorflow locally:

This documentation describes how to install tensorflow package locally in your \$HOME space.

Load python module

```
module load python/3.6-conda5.2
```

Clone python installation to local directory

Three alternative create commands are listed. These cover the most common cases:

```
conda create -n local --clone="$PYTHON_HOME"
```

This will clone the entire python installation to ~/envs/local directory. The process will take several minutes.

```
conda create -n local
```

This will create a local python installation without any packages. If you need a small number of packages, you may choose this option.

conda create -n local python={version} anaconda

If you like to install a specific version of python, you can specify it with "python" option. For example, you can use "python=3.6" for version 3.6.

To verify that a clone has been created, use the command

```
conda info -e
```

For additional conda command documentation see <https://conda.io/docs/commands.html>

Activate clone environment

For the bash shell:

```
source activate local
```

On newer versions of Anaconda on the Owens cluster you may also need to perform the removal of the following packages before trying to install your specific packages:

```
conda remove conda-build
```

```
conda remove conda-env
```

Install package

Install the latest version of tensorflow that is gpu compatible.

```
pip install tensorflow-gpu
```

If there are errors on this step you will need to resolve them before continuing.

Test python package

Now we will test tensorflow package by loading it in python and checking its location to ensure we are using the correct version.

```
python -c "import tensorflow;print (tensorflow.__file__)"
```

Output:

```
$HOME/.conda/envs/local/lib/python2.7/site-packages/tensorflow/__init__.py
```

Remember, you will need to load the proper version of python before you go to use your newly installed package. Packages are only installed to one version of python.

Install your own python modules

If the method using conda above is not working or if you prefer, you can consider installing python modules from the source. Please read [HOWTO: install your own python modules](#).

Keras Installation and Environment Setup:

Keras is one of the most popular Python libraries. It is having high demand these days as it is straight-forward and simple. It is a high-level API that does not perform low-level computations. Keras runs on the TensorFlow and Theano.

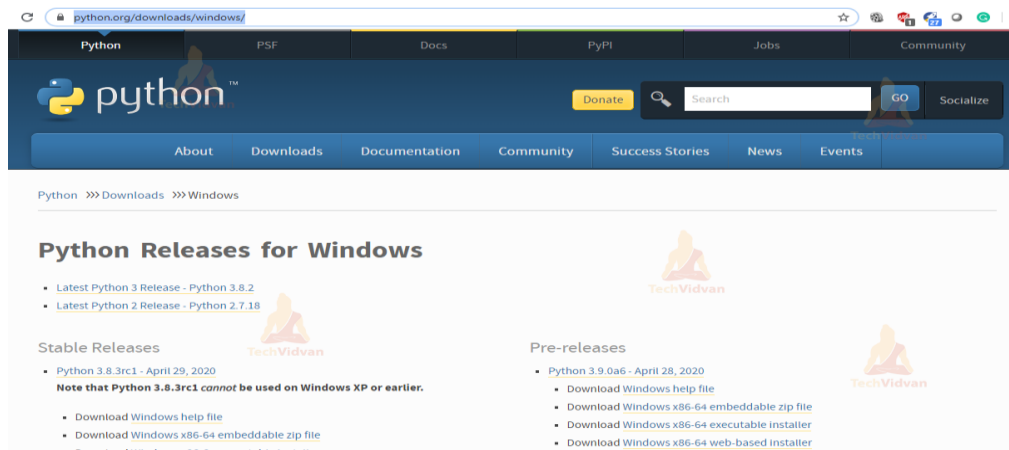
It is handy for Deep Learning and focuses on the idea of Models. Keras is an open-source Python library. It is very easy and effortless to download. It is easily and freely available. You can download Keras with no efforts.

Let us learn Keras installation in easy steps.

Keras Installation and Environment setup

Step 1: Install Python

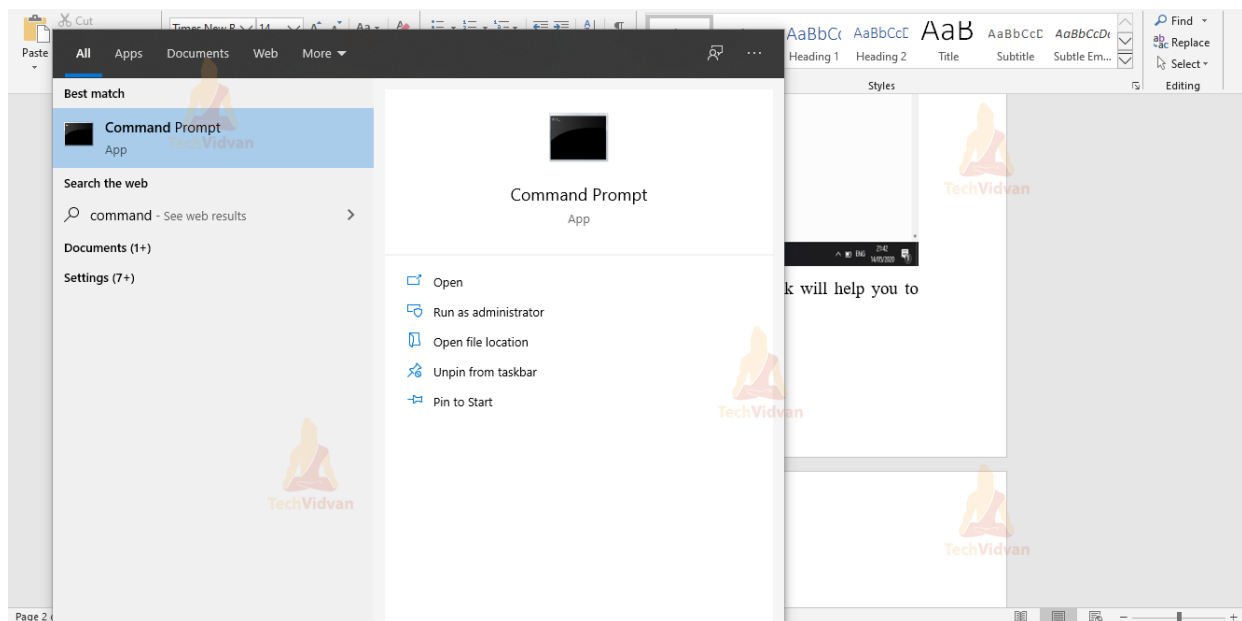
It is the primary task to install Python in your system. Python is an open-source language. It is easily available. [Download Python](#) now.



Click on Latest Python 3 Release – Python 3.8.2. This link will help you to download the latest version of Python.

Step 2: Now, Open the Command Prompt

In this step, open the command prompt. Run the command prompt as an administrator.



Running the command prompt as an administrator will enable you to make changes in your system. It will ask you permission to make changes to your system. So, give it permission by pressing the 'Yes' button.

Step 3: Now, type 'pip' in Command Prompt

Type 'pip' as a command in the command prompt. It will help you to check whether Python is installed or not.

```
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip

TechVidvan
```

After typing 'pip' in the command prompt, you will see many functions executing. Wait, till the functions execute.

Advertisement

```
C:\WINDOWS\system32>pip

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash            Compute hashes of package archives.
  completion       A helper command used for command completion.
  help            Show help for commands.

General Options:
  -h, --help            Show help.
  --isolated            Run pip in an isolated mode, ignoring
                       environment variables and user configuration.
  -v, --verbose         Give more output. Option is additive, and can be
                       used up to 3 times.
  -V, --version         Show version and exit.
  -q, --quiet          Give less output. Option is additive, and can be
                       used up to 3 times (corresponding to WARNING,
                       ERROR, and CRITICAL logging levels).
  --log <path>        Path to a verbose appending log.
  --proxy <proxy>     Specify a proxy in the form
                       [user:passwd@]proxy.server:port.
  --retries <retries> Maximum number of retries each connection should
                       attempt (default 5 times).
  --timeout <sec>     Set the socket timeout (default 15 seconds).
  --exists-action <action>
                       Default action when a path already exists:
                       (s)witch, (i)gnore, (w)ipe, (b)ackup, (a)bort.
  --trusted-host <hostname>
                       Mark this host as trusted, even though it does
                       not have valid or any HTTPS.
  --cert <path>       Path to alternate CA bundle.
  --client-cert <path>
                       Path to SSL client certificate, a single file
                       containing the private key and the certificate
                       in PEM format.
  --cache-dir <dir>  Store the cache data in <dir>.
```

Step 4: Write 'pip install tensorflow==1.8' in Command Prompt
Being the fact that Keras runs on the top of Keras. You need to install TensorFlow first.

```
C:\WINDOWS\system32>pip install tensorflow==1.8
```

After typing this command, you will see many functions executing. Tensorboard, termcolor, numpy, wheel, etc are the functions that will be executed. You can many commands and functions executing in the image below.

```
C:\WINDOWS\system32>pip install tensorflow==1.8
Collecting tensorflow==1.8
  Downloading https://files.pythonhosted.org/packages/f4/88/980d7032b7408fcca5b0b8d420fcd97919197a9e7acf280ab74fc7db6993/tensorflow-1.8.0-cp36-cp36m-win_amd64.whl (34.4 MB)
  100% |#####| 34.4MB 18kB/s
Collecting six>=1.10.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/65/eb/1f97cb97bfc2390a276969c6fae16075da282f5058082d4cb10c6c5c1dba/six-1.14.0-py2.py3-none-any.whl
Collecting gast>=0.2.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/d6/84/759f5dd23fec8ba71952d97bcc7e2c9d7d63bdc582421f3cd4be845f0c98/gast-0.3.3-py2.py3-none-any.whl
Collecting tensorboard<1.9.0,>=1.8.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/59/a6/0ae6092b7542cfedba6b2a1c9b8dceaef278238c39484f3ba03b03f07803c/tensorboard-1.8.0-py3-none-any.whl (3.1MB)
  100% |#####| 3.1MB 129kB/s
Collecting termcolor>=1.1.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/8a/48/a76be51647d0eb9f10e2a4511bf3ffb8cc1e6b14e9e4fab46173aa79f981/termcolor-1.1.0.tar.gz
Collecting protobuf>=3.4.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/ff/52/a71156b82dbb8a40833b7a571e22c9e65ca4204a56739f97d3eaa25d111e/protobuf-3.11.3-cp36-cp36m-win_amd64.whl (1.1MB)
  100% |#####| 1.1MB 281kB/s
Collecting grpcio>=1.8.6 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/97/5b/5d962954bdae84cd5b06978a15049d947a2dad5b02130b3d984d076c0e1/grpcio-1.28.1-cp36-cp36m-win_amd64.whl (2.1MB)
  100% |#####| 2.2MB 197kB/s
Collecting numpy>=1.13.3 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/5c/74/04e9fb4ed1aaca3bf762429c3567c9523c11b1ef615795737e16f3cd23/numpy-1.18.4-cp36-cp36m-win_amd64.whl (12.8MB)
  100% |#####| 12.8MB 48kB/s
Collecting wheel>=0.26 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/8c/23/848298cccf8e40f5bbb59009b32848a4c38f4e7f3364297ab3c3e2e2cd14/wheel-0.34.2-py2.py3-none-any.whl
Collecting absl-py>=0.1.6 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/1a/53/9243c600e047bd4c3df9e69c9abc1e8004a82cac2e0c484580a78a94ba2a/absl-py-0.9.0.tar.gz (104kB)
  100% |#####| 112kB 208kB/s
Collecting astor>=0.6.0 (from tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/c3/88/97eef84f48fa04fbd6750e62dceaefba6c63c81b7ac1420856c8d0c0a3f9/astor-0.8.1-py2.py3-none-any.whl
Collecting html5lib==0.9999999 (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/ae/ae/bcb60402c60932b32dfaf19bb53870b29eda2cd17551ba5639219f5ebf9/html5lib-0.9999999.tar.gz (889kB)
  100% |#####| 890kB 161kB/s
Collecting bleach==1.5.0 (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/33/70/86c5fec937ea4964184d4d6c4f0b9551564f821e1c3575907639036d9b90/bleach-1.5.0-py2.py3-none-any.whl
Collecting markdown>=2.6.8 (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/44/63/eaec2bd025ab48c754b5e8819af0f6a69e2b1e187611dd40cbe101ee7f/Markdown-3.2.2-py3-none-any.whl (88kB)
  100% |#####| 92kB 459kB/s
Collecting werkzeug>=0.11.10 (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8)
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bace597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
  100% |#####| 307kB 289kB/s
Requirement already satisfied: setuptools in c:\users\radhika\appdata\local\programs\python\python36\lib\site-packages (from protobuf>=3.4.0->tensorflow==1.8)
```

Step 5: Write ‘pip install keras’ on Command Prompt

Now, it’s time to finally install Keras. After writing ‘pip install keras’, you will see prompt collecting many files.

```
C:\WINDOWS\system32>pip install tensorflow==1.8
Collecting tensorflow==1.8
  Downloading https://files.pythonhosted.org/packages/f4/88/980d7032b7408fcca5b0b8d420fcd97919197a9e7acf280ab74fc7db6993/tensorflow-1.8.0-cp36-cp36m-win_amd64.whl (34.4
  MB)
  100% |#####| 34.4MB 18kB/s
```

You will see that it is automatically ignoring the functions are that not much necessary. It is very easy to install Keras. It will automatically install all the secondary files it needs.

After writing this command, wait for it to execute completely. Once it is done, you have successfully installed Keras. Now, you can easily work with the Keras code. Write the Keras commands easily and safely. Enjoy working with Keras.

Conclusion

This is how Keras installation is done. Keras is an open-source Python library. It is easy to install Keras. As Keras runs on the top of TensorFlow, Theano. You have to install any of these libraries first.

Here, you can see TensorFlow. After installing TensorFlow, you can install Keras. It is not a burden to install Keras. It is not too time-consuming. You can easily and quickly install it.

Run tensor flow program on aws cloud platform:

Getting Started with TensorFlow on AWS

PAGE CONTENT

[Amazon SageMaker](#)[AWS Deep Learning AMI](#)[AWS Deep Learning Containers](#)[Amazon EC2 inf1 instances/](#)[AWS Inferentia](#)[Amazon Elastic Inference](#)

Amazon SageMaker

The easiest way to get started with TensorFlow on AWS is using Amazon SageMaker, a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy TensorFlow models quickly. SageMaker assists with each step of the machine learning process to make it easier to develop high quality models. Data scientists can also use SageMaker with TensorBoard to save development time by visualizing the model architecture to identify and remediate convergence issues, such as validation loss not converging or vanishing gradients. To get started with TensorFlow and TensorBoard on SageMaker, use the following resources:

- [Use TensorFlow with SageMaker documentation](#)
- [SageMaker with TensorBoard documentation](#)
- [PyTorch in the SageMaker Python SDK](#)
- [SageMaker TensorFlow container](#)
- [SageMaker TensorFlow serving container](#)
- [TensorFlow in SageMaker Workshop](#)

- [Extending containers](#)

AWS Deep Learning AMI

AWS Deep Learning AMIs are machine images pre-installed with TensorFlow, allowing you to quickly experiment with new algorithms or learn new skills and techniques. To get started, see the TensorFlow on AWS Deep Learning AMIs tutorials below.

- [TensorFlow](#)
- [TensorFlow 2](#)
- [TensorFlow with Horovod](#)
- [TensorFlow 2 with Horovod](#)

AWS Deep Learning Containers

AWS Deep Learning Containers are Docker images pre-installed with TensorFlow to make it easy to deploy custom machine learning environments quickly by letting you skip the complicated process of building and optimizing your environments from scratch. To get started with TensorFlow on AWS DL Containers, use the following resources:

- TensorFlow on Amazon EC2: [Training](#) | [Inference](#)
- TensorFlow on Amazon ECS: [Training](#) | [Inference](#)
- TensorFlow on Amazon EKS: [Training](#) | [Distributed Training](#) | [CPU Inference](#) | [GPU Inference](#)

Amazon EC2 Inf1 instances/ AWS Inferentia

Amazon EC2 Inf1 instances are built from the ground up to support machine learning inference applications. Inf1 instances feature up to 16 [AWS Inferentia](#) chips, high-performance machine learning inference chips designed and built by AWS. Inf1 instances deliver up to 3x higher throughput and up to 40% lower cost per inference than Amazon EC2 G4 instances, which were already the lowest cost instance for machine learning inference available in the cloud. Using Inf1 instances, you can run large scale machine learning inference with TensorFlow models at the lowest cost in the cloud. To get started, see our [tutorial on running TensorFlow models on Inf1](#).

Amazon Elastic Inference

Amazon Elastic Inference allows you to attach low-cost GPU-powered acceleration to Amazon EC2 and SageMaker instances or Amazon ECS tasks, to reduce the cost of running inference with PyTorch models by up to 75%. To get started with TensorFlow on Elastic Inference, see the following resources.

UNIT-II

Introduction to neural network

What is neural network:

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.

Why are neural networks important?

Neural networks can help computers make intelligent decisions with limited human assistance. This is because they can learn and model the relationships between input and output data that are nonlinear and complex. For instance, they can do the following tasks.

Make generalizations and inferences

Neural networks can comprehend unstructured data and make general observations without explicit training. For instance, they can recognize that two different input sentences have a similar meaning:

- Can you tell me how to make the payment?
- How do I transfer money?

A neural network would know that both sentences mean the same thing. Or it would be able to broadly recognize that Baxter Road is a place, but Baxter Smith is a person's name.

How neural networks work:

What is a neural network?

Simply said, a neural network is a set of algorithms designed to recognize patterns or relationships in a given dataset. These deep neural networks are basically computing systems designed to mimic how the human brain analyzes and processes information.

A neural network consists of neurons interconnected like a web and these neurons are mathematical functions or models that do the computations required for classification according to a given set of rules. Through this tutorial, let's discuss how these artificial neural networks work and their real-world usage.

How does a neural network learn?

Before moving on to learn how exactly the neural network works, you need to know what forms a neural network. A normal neural network consists of multiple layers called the input layer, output layer, and hidden layers. In each layer every node (neuron) is connected to all nodes (neurons) in the next layer with parameters called 'weights'.

Neural networks consist of nodes called perceptrons that do necessary calculations and detect features of neural networks. These perceptrons try to reduce the final cost error by adjusting the weights parameters. Moreover, a perceptron can be considered as a neural network with a single layer.

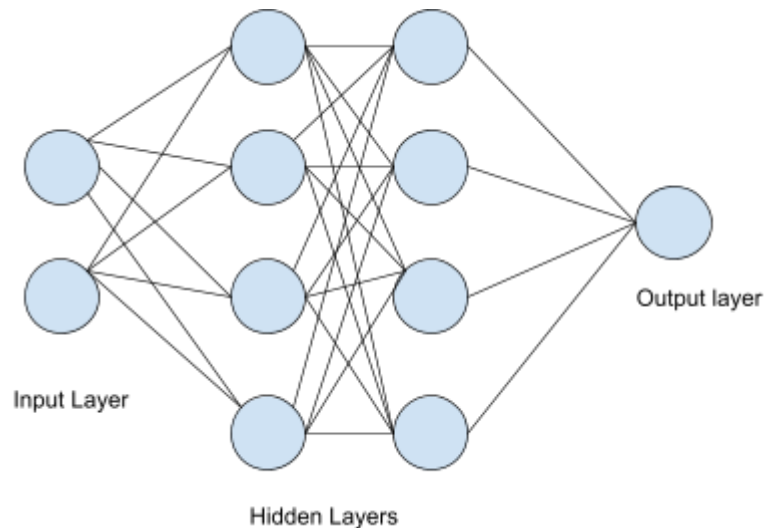
On the other hand, multilayer perceptrons are called deep neural networks. The perceptrons are activated when there is satisfiable input. Go through this [wiki article](#) if you need to learn more about perceptrons.

Now let's move on to discuss the exact steps of a working neural network.

1. Initially, the dataset should be fed into the input layer which will then flow to the hidden layer.
2. The connections which exist between the two layers randomly assign weights to the input.
3. A bias is added to each input. Bias is a constant which is used in the model to fit best for the given data.
4. The weighted sum of all the inputs will be sent to a function that is used to decide the active status of a neuron by calculating the weighted sum and adding the bias. This function is called the activation function.
5. The nodes that are required to fire for feature extraction are decided based on the output value of the activation function.
6. The final output of the network is then compared to the required labeled data of our dataset to calculate the final cost error. The cost error is actually telling us how 'bad' our network is. Hence we want the error to be as smallest as we can.
7. The weights are adjusted through back propagation, which reduces the error. This back propagation process can be considered as the central mechanism that neural networks learn. It basically fine-tunes the weights of the deep neural network in order to reduce the cost value.

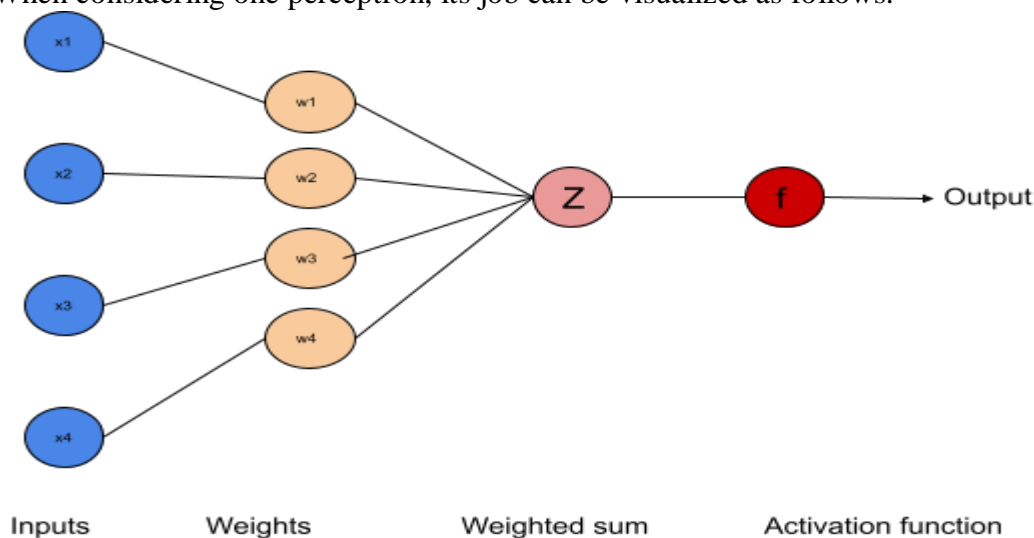
In simple terms, what we do when training a neural network is usually calculating the loss (error value) of the model and checking if it is reduced or not. If the error is higher than the expected value, we have to update the model parameters, such as weights and bias values. We can use the model once the loss is lower than the expected error margin.

Neural network visualization



Neural networks can be described easily using the above diagram. The light blue circles represent the perceptrons we discussed earlier, and the lines represent connections between artificial neurons.

When considering one perceptron, its job can be visualized as follows.



When you input the data with random weights to the model, it generates the weighted sum of them. According to that value, the activation function decides the activation status of the neuron. The output of this perceptron may act as an input for the next neuron layer.

Gradient descent :

Gradient descent is an optimization algorithm which is commonly-used to train [machine learning](#) models and [neural networks](#). Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error. Once machine learning models are optimized for accuracy, they can be powerful tools for artificial intelligence (AI) and computer science applications.

Perceptron:

A perceptron is the smallest element of a neural network. Perceptron is a single-layer neural network linear or a Machine Learning algorithm used for supervised learning of various binary classifiers. It works as an artificial neuron to perform computations by learning elements and processing them for detecting the business intelligence and capabilities of the input data. A perceptron network is a group of simple logical statements that come together to create an array of complex logical statements, known as the [neural network](#).

UNIT-3

Tensor flow basics:

Placeholders are Tensor-like objects. They are a contract between you and TensorFlow that says when you run your computation graph in a session, you will supply or *feed* data into that placeholder so that your graph can run successfully.

They are Tensor-like objects as they behave like Tensors, meaning you can pass them around in places where you would put a Tensor.

By using placeholders, we can supply external inputs into our graph that might change each time we run our graph. The natural use for them is as a way to supply data and labels into our model as the data and labels we supply will generally be different each time we want to run our graph.

When creating a placeholder, we must supply the datatype that will be fed.

We will use two placeholders to supply data and labels into our graph. We also supply the shape that any data fed into these placeholders must take. We use None to indicate the size of that particular dimension can take any value. This way we are able to feed in batches of data that are varying sizes. Following we'll see how to define placeholders in TensorFlow for our problem.

```
x = tf.placeholder(tf.float32, shape=[None, 4], name="data_in")
```

```
y = tf.placeholder(tf.int32, shape=[None, 3], name="target_labels")
```

Copy

Now, we have created placeholders in our graph, so we can construct our linear model on the graph as well. We call our function that we defined previously, and supply as input our data placeholder, x. Remember, placeholders act like Tensors so they can be passed around like them as well. In the following code we call our `linear_model` function with our placeholder as the input argument.

```
model_out = linear_model(x)
```

Copy

When we call our function, everything inside it executes and all the ops and variables are added to our TensorFlow graph. We only need to do this once; if we were to try calling our function again, we would get an error saying that we have tried to add variables to the graph but they already exist.

Placeholders are the simplest and quickest way of supplying external data into our graph, so it's good to know about them. Later on, we will see better ways of supplying data using the dataset API, but for now placeholders are a good place to start.

Variables:

A TensorFlow **variable** is the recommended way to represent shared, persistent state your program manipulates. This guide covers how to create, update, and manage instances of `tf.Variable` in TensorFlow.

Variables are created and tracked via the `tf.Variable` class. A `tf.Variable` represents a tensor whose value can be changed by running ops on it. Specific ops allow you to read and modify the values of this tensor. Higher level libraries like `tf.keras` use `tf.Variable` to store model parameters.

Constants:

A TensorFlow **variable** is the recommended way to represent shared, persistent state your program manipulates. This guide covers how to create, update, and manage instances of `tf.Variable` in TensorFlow.

Variables are created and tracked via the `tf.Variable` class. A `tf.Variable` represents a tensor whose value can be changed by running ops on it. Specific ops allow you to read and modify

the values of this tensor. Higher level libraries like `tf.keras` use `tf.Variable` to store model parameters.

UNIT-4

Activation functions:

What are the activation function:

The activation functions are at the very core of Deep Learning. They determine the output of a model, its accuracy, and computational efficiency. In some cases, activation functions have a major effect on the model's ability to converge and the convergence speed.

In this article, you'll learn the following most popular activation functions in Deep Learning and how to use them with Keras and TensorFlow 2.

1. Sigmoid (Logistic)
2. Hyperbolic Tangent (Tanh)
3. Rectified Linear Unit (ReLU)
4. Leaky ReLU
5. Parametric Leaky ReLU (PReLU)
6. Exponential Linear Units (ELU)
7. Scaled Exponential Linear Unit (SELU)

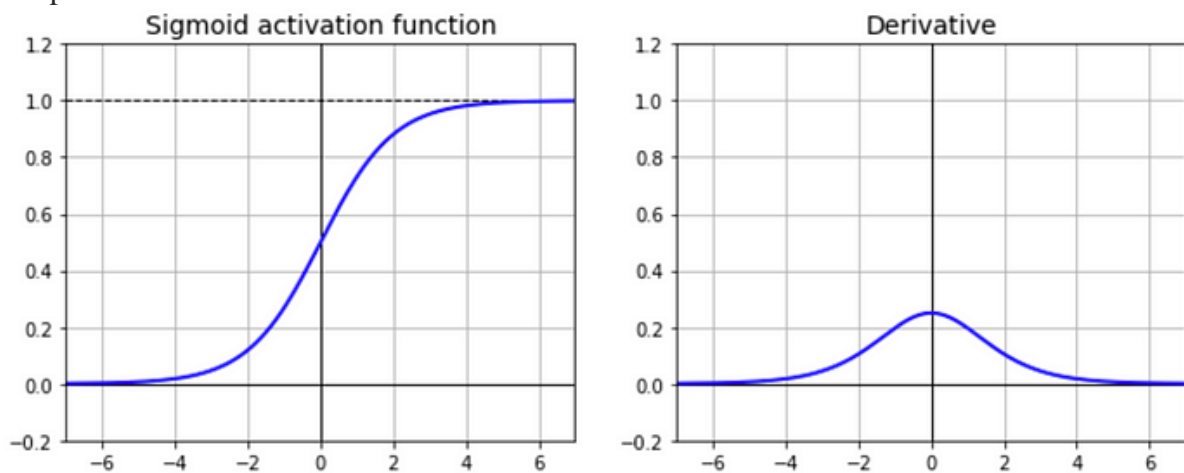
1. Sigmoid (Logistic)

The **Sigmoid function** (also known as the **Logistic function**) is one of the most widely used activation function. The function is defined as:

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

Sigmoid activation function (Image by author)

The plot of the function and its derivative.



the plot of Sigmoid function and its derivative (Image by author)

As we can see in the plot above,

- The function is a common **S-shaped** curve.
- The output of the function is centered at **0.5** with a range from **0** to **1**.
- The function is **differentiable**. That means we can find the slope of the sigmoid curve at any two points.
- The function is **monotonic** but the function's derivative is not.

The **Sigmoid** function was introduced to Artificial Neural Networks (ANN) in the 1990s to replace the **Step** function [2]. It was a key change to ANN architecture because

the **Step** function doesn't have any gradient to work with Gradient Descent, while the **Sigmoid** function has a well-defined nonzero derivative everywhere, allowing Gradient Descent to make some progress at every step during training.

Problems with Sigmoid activation function

The main problems with the Sigmoid function are:

1. **Vanishing gradient:** looking at the function plot, you can see that when inputs become small or large, the function saturates at 0 or 1, with a derivative extremely close to 0. Thus it has almost no gradient to propagate back through the network, so there is almost nothing left for lower layers [2].
2. **Computationally expensive:** the function has an exponential operation.
3. **The output is not zero centered:**

How to use it with Keras and TensorFlow 2

To use the Sigmoid activation function with Keras and TensorFlow 2, we can simply pass 'sigmoid' to the argument activation :

```
from tensorflow.keras.layers import DenseDense(10, activation='sigmoid')
```

To apply the function for some constant inputs:

```
import tensorflow as tf
```

```
from tensorflow.keras.activations import sigmoidz = tf.constant([-20, -1, 0, 1.2], dtype=tf.float32)
```

```
output = sigmoid(z)
```

```
output.numpy()
```

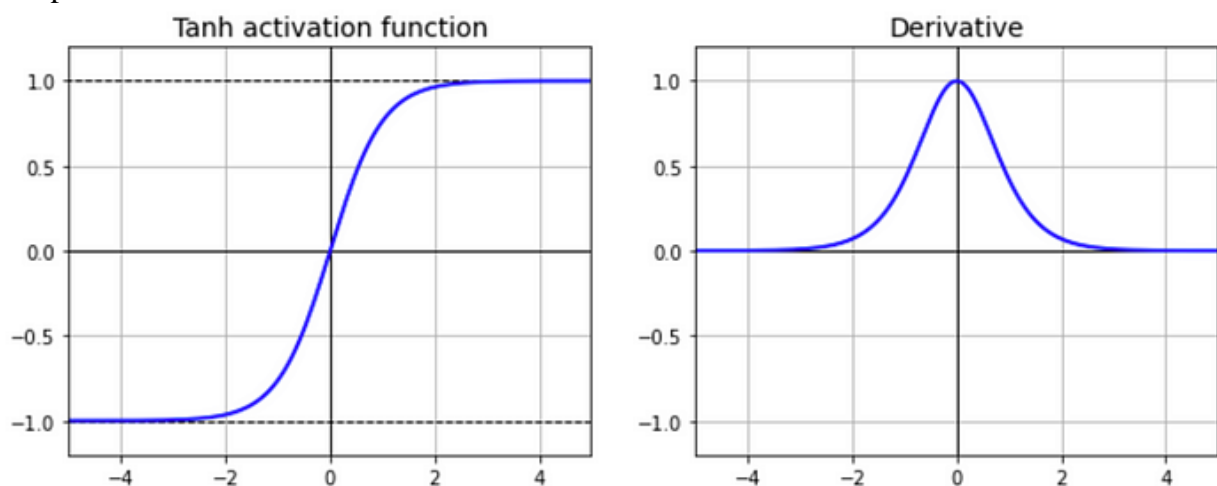
2. Hyperbolic Tangent (Tanh)

Another very popular and widely used activation function is the **Hyperbolic Tangent**, also known as **Tanh**. It is defined as:

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

tanh function (image by author)

The plot of the function and its derivative:



The plot of tanh and its derivative (image by author)

We can see that the function is very similar to the Sigmoid function.

- The function is a common **S-shaped** curve as well.
- The difference is that the output of **Tanh** is **zero centered** with a range from **-1 to 1** (instead of 0 to 1 in the case of the Sigmoid function)

- The same as the Sigmoid, this function is **differentiable**
 - The same as the Sigmoid, the function is **monotonic**, but the function's derivative is not.
- Tanh** has characteristics similar to **Sigmoid** that can work with Gradient Descent. One important point to mention is that **Tanh** tends to make each layer's output more or less centered around 0 and this often helps speed up convergence [2].

Problems with Tanh activation function

Since **Tanh** has characteristics similar to **Sigmoid**, it also faces the following two problems:

1. **Vanishing gradient:** looking at the function plot, you can see that when inputs become small or large, the function saturates at -1 or 1, with a derivative extremely close to 0. Thus it has almost no gradient to propagate back through the network, so there is almost nothing left for lower layers.
2. **Computationally expensive:** the function has an exponential operation.

How to use Tanh with Keras and TensorFlow 2

To use the Tanh, we can simply pass 'tanh' to the argument activation:

```
from tensorflow.keras.layers import DenseDense(10, activation='tanh')
```

To apply the function for some constant inputs:

```
import tensorflow as tf
from tensorflow.keras.activations import tanh
z = tf.constant([-20, -1, 0, 1.2], dtype=tf.float32)
output = tanh(z)
output.numpy()
```

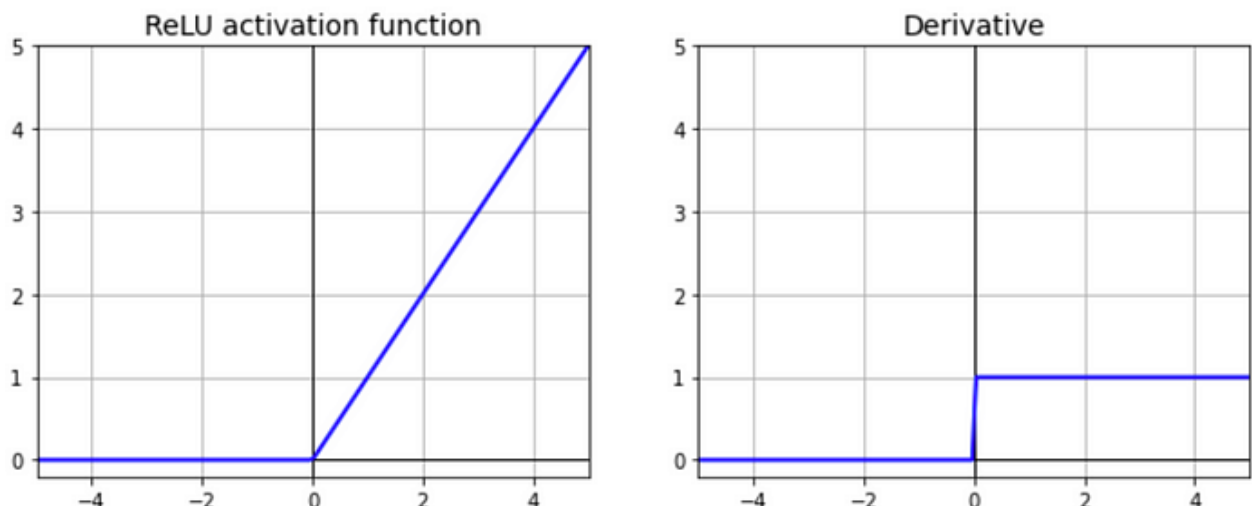
3. Rectified Linear Unit (ReLU)

The **Rectified Linear Unit (ReLU)** is the most commonly used activation function in deep learning. The function returns 0 if the input is negative, but for any positive input, it returns that value back. The function is defined as:

$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

ReLU function (image by author)

The plot of the function and its derivative:



The plot of ReLU and its derivative

As we can see that:

- Graphically, the ReLU function is composed of two linear pieces to account for non-linearities. A function is non-linear if the slope isn't constant. So, the ReLU function is

non-linear around 0, but the slope is always either 0 (for negative inputs) or 1 (for positive inputs).

- The ReLU function is **continuous**, but it is **not differentiable** because its derivative is 0 for any negative input.
- The output of ReLU does not have a maximum value (It is **not saturated**) and this helps Gradient Descent
- The function is very fast to compute (Compare to Sigmoid and Tanh)

It's surprising that such a simple function works very well in deep neural networks.

Problem with ReLU

ReLU works great in most applications, but it is not perfect. It suffers from a problem known as the **dying ReLU**.

Dying ReLU

During training, some neurons effectively die, meaning they stop outputting anything other than 0. In some cases, you may find that half of your network's neurons are dead, especially if you used a large learning rate. A neuron dies when its weights get tweaked in such a way that the weighted sum of its inputs are negative for all instances in the training set. When this happens, it just keeps outputting 0s, and gradient descent does not affect it anymore since the gradient of the ReLU function is 0 when its input is negative.

How to use it with Keras and TensorFlow 2

To use ReLU with Keras and TensorFlow 2, just set `activation='relu'`
from tensorflow.keras.layers import DenseDense(10, **activation='relu'**)

To apply the function for some constant inputs:

```
import tensorflow as tf
from tensorflow.keras.activations import relu
output = relu(z)
output.numpy()
```

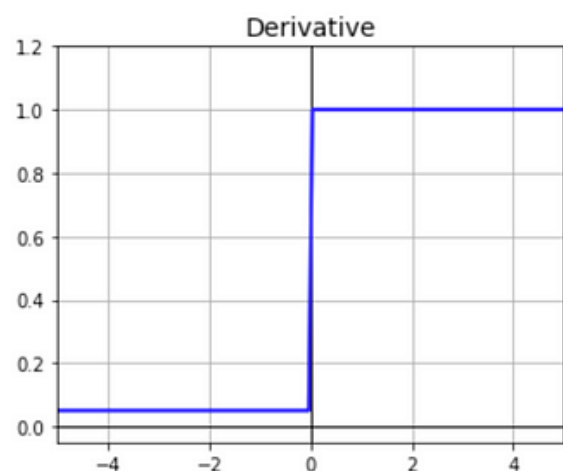
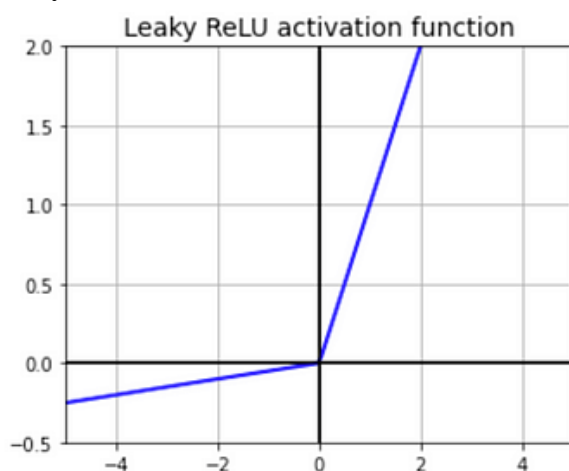
4. Leaky ReLU

Leaky ReLU is an improvement over the ReLU activation function. It has all properties of ReLU, plus it will never have **dying ReLU** problem. Leaky ReLU is defined as:

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

The hyperparameter α defines how much the function leaks.

It is the slope of the function for $x < 0$ and is typically set to 0.01. The small slope ensures that Leaky ReLU never dies.



How to use Leaky ReLU with Keras and TensorFlow 2

To use the **Leaky ReLU** activation function, you must create a LeakyReLU instance like below:

```
from tensorflow.keras.layers import LeakyReLU, Denseleaky_relu = LeakyReLU(alpha=0.01)
Dense(10, activation=leaky_relu)
```

5. Parametric leaky ReLU (PReLU)

Parametric leaky ReLU (PReLU) is a variation of Leaky ReLU, where α is authorized to be learned during training (instead of being a hyperparameter, it becomes a parameter that can be modified by back propagation like any other parameters). This was reported to strongly outperform ReLU on large image datasets, but on smaller datasets it runs the risk of over fitting the training set [2].

How to use PReLU with Keras and TensorFlow 2

To use Parametric leaky ReLU, you must create a PReLU instance like below:

```
from tensorflow.keras.layers import PReLU, Densepara_relu = PReLU()
Dense(10, activation=para_relu)
```

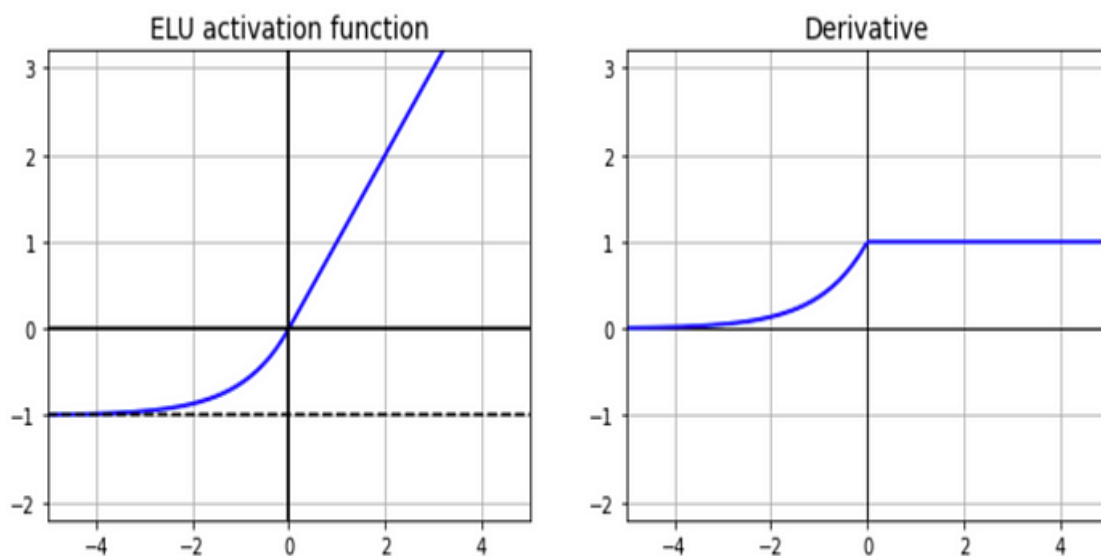
6. Exponential Linear Unit (ELU)

Exponential Linear Unit (ELU) is a variation of ReLU with a better output for $z < 0$. The function is defined as:

$$\begin{cases} \alpha (e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

ELU function

The hyperparameter α controls the value to which an ELU saturates for negative net inputs. The plot of the function and its derivative:



The plot of ELU and its derivative (image by author)

We can see in the plot above,

- **ELU** modified the slope of the negative part of the function.
- Unlike the **Leaky ReLU** and **PReLU** functions, instead of a straight line, **ELU** uses a log curve for the negative values.

According to the authors, ELU outperformed all the ReLU variants in their experiments [3].

Problem with ELU

According to [2, 3], the main drawback of the ELU activation is that it is slower to compute than the ReLU and its variants (due to the use of the exponential function), but during training

this is compensated by the faster convergence rate. However, at test time, an ELU network will be slower than a ReLU network.

How to use it with Keras and TensorFlow 2

Implementing ELU in TensorFlow 2 is trivial, just specify the activation function when building each layer:

```
Dense(10, activation='elu')
```

To apply the function for some constant inputs:

```
import tensorflow as tf
```

```
from tensorflow.keras.activations import elu, z = tf.constant([-20, -1, 0, 1.2], dtype=tf.float32)
```

```
output = elu(z, alpha=1)
```

```
output.numpy()
```

7. Scaled Exponential Linear Unit (SELU)

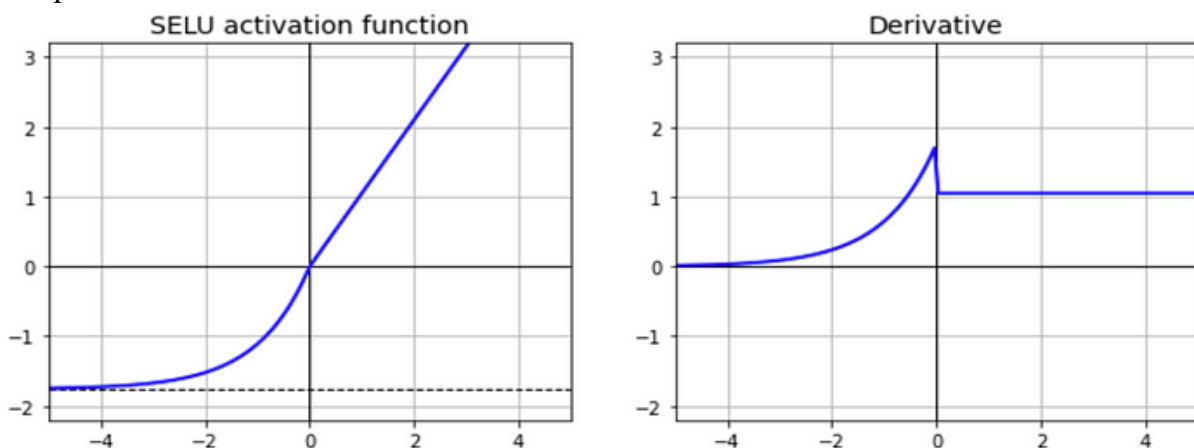
Exponential Linear Unit (SELU) activation function is another variation of **ReLU** proposed by Günter Klambauer et al. [4] in 2017. The authors showed that if you build a neural network composed exclusively of a stack of dense layers, and if all hidden layers use the **SELU** activation function, then the network will self-normalize (the output of each layer will tend to preserve mean 0 and standard deviation 1 during training, which resolves the vanishing/exploding gradients problem). This activation function often outperforms other activation functions very significantly.

SELU is defined as:

$$f(x) = \text{scale} * x, z > 0$$
$$= \text{scale} * \alpha * (\exp(x) - 1), z \leq 0$$

where α and scale are pre-defined constants ($\alpha=1.67326324$ and $\text{scale}=1.05070098$).

The plot of SELU and its derivative:



sigmoid functions:

TensorFlow is open-source Python library designed by Google to develop Machine Learning models and deep learning neural networks.

sigmoid() is used to find element wise sigmoid of x.

Syntax: `tensorflow.math.sigmoid(x, name)`

Parameters:

- **x:** It's a tensor. Allowed dtypes are `float16`, `float32`, `float64`, `complex64`, or `complex128`.
- **name(optional):** It defines the name for the operation.

Return: It return a tensor of same dtype as x.

Example 1:
Python3

```
# importing the library
import tensorflow as tf

# Initializing the input tensor
a = tf.constant([.2, .5, .7, 1, 2, 5, 10], dtype = tf.float64)

# Printing the input tensor
print('a: ', a)

# Calculating result
res = tf.math.sigmoid(x = a)

# Printing the result
print('Result: ', res)
```

Output:

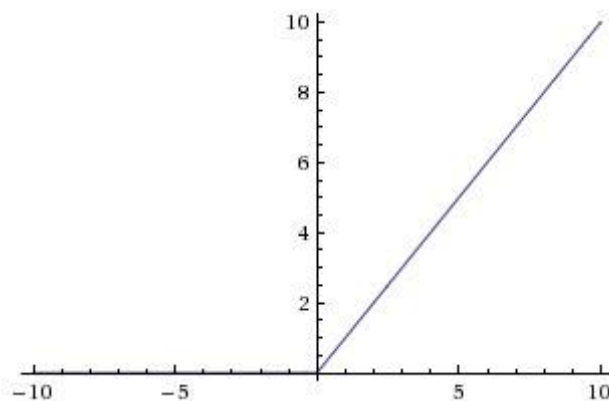
```
a: tf.Tensor([ 0.2  0.5  0.7  1.  2.  5. 10. ], shape=(7, ), dtype=float64)
Result: tf.Tensor(
[0.549834  0.62245933 0.66818777 0.73105858 0.88079708 0.99330715
 0.9999546 ], shape=(7, ), dtype=float64)
Relu- rectified linear units:
```

Introduction

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written

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

Graphically it looks like this



It's surprising that such a simple function (and one composed of two linear pieces) can allow your model to account for non-linearities and interactions so well. But the ReLU function works great in most applications, and it is very widely used as a result.

Why It Works

Introducing Interactions and Non-linearities

Activation functions serve two primary purposes: 1) Help a model account for **interaction effects**. What is an interactive effect? It is when one variable A affects a prediction differently depending on the value of B. For example, if my model wanted to know whether a certain body weight indicated an increased risk of diabetes, it would have to know an individual's height. Some bodyweights indicate elevated risks for short people, while indicating good health for tall people. So, the **effect of**

body weight on diabetes risk depends on height, and we would say that **weight and height have an interaction effect**.

2) Help a model account for **non-linear effects**. This just means that if I graph a variable on the horizontal axis, and my predictions on the vertical axis, it isn't a straight line. Or said another way, the effect of increasing the predictor by one is different at different values of that predictor.

How ReLU captures Interactions and Non-Linearities

Interactions: Imagine a single node in a neural network model. For simplicity, assume it has two inputs, called A and B. The weights from A and B into our node are 2 and 3 respectively. So the node output is $f(2A+3B)$. We'll use the ReLU function for our f . So, if $2A+3B$ is positive, the output value of our node is also $2A+3B$. If $2A+3B$ is negative, the output value of our node is 0.

For concreteness, consider a case where $A=1$ and $B=1$. The output is $2A+3B$, and if A increases, then the output increases too. On the other hand, if $B=-100$ then the output is 0, and if A increases moderately, the output remains 0. So A might increase our output, or it might not. It just depends what the value of B is.

This is a simple case where the node captured an interaction. As you add more nodes and more layers, the potential complexity of interactions only increases. But you should now see how the activation function helped capture an interaction.

Non-linearities: A function is non-linear if the slope isn't constant. So, the ReLU function is non-linear around 0, but the slope is always either 0 (for negative values) or 1 (for positive values). That's a very limited type of non-linearity.

But two facts about deep learning models allow us to create many different types of non-linearities from how we combine ReLU nodes.

First, most models include a **bias** term for each node. The bias term is just a constant number that is determined during model training. For simplicity, consider a node with a single input called A, and a bias. If the bias term takes a value of 7, then the node output is $f(7+A)$. In this case, if A is less than -7, the output is 0 and the slope is 0. If A is greater than -7, then the node's output is $7+A$, and the slope is 1.

So the bias term allows us to move where the slope changes. So far, it still appears we can have only two different slopes.

However, real models have many nodes. Each node (even within a single layer) can have a different value for its bias, so each node can change slope at different values for our input.

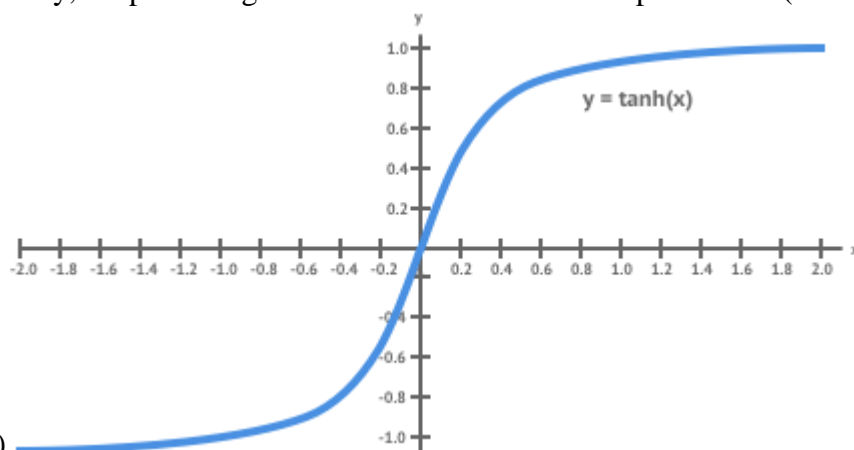
When we add the resulting functions back up, we get a combined function that changes slopes in many places.

These models have the flexibility to produce non-linear functions and account for interactions well (if that will give better predictions). As we add more nodes in each layer (or more convolutions if we are using a convolutional model) the model gets even greater ability to represent these interactions and non-linearities.

Facilitating Gradient Descent

This section is more technical than those above it. If you find it difficult, remember that you can have a lot of success using deep learning even without this technical background.

Historically, deep learning models started off with s-shaped curves (like the tanh function



below)

The tanh would seem to have a couple advantages. Even though it gets close to flat, it isn't completely flat anywhere. So its output always reflects changes in its input, which we might expect to be a good thing. Secondly, it is non-linear (or curved everywhere). Accounting for non-linearities is one of the activation function's main purposes. So, we expect a non-linear function to work well.

However researchers had great difficulty building models with many layers when using the tanh function. It is relatively flat except for a very narrow range (that range being about -2 to 2). The derivative of the function is very small unless the input is in this narrow range, and this flat derivative makes it difficult to improve the weights through gradient descent. This problem gets worse as the model has more layers. This was called the **vanishing gradient problem**.

The ReLU function has a derivative of 0 over half its range (the negative numbers). For positive inputs, the derivative is 1.

When training on a reasonable sized batch, there will usually be some data points giving positive values to any given node. So the average derivative is rarely close to 0, which allows gradient descent to keep progressing.

Alternatives

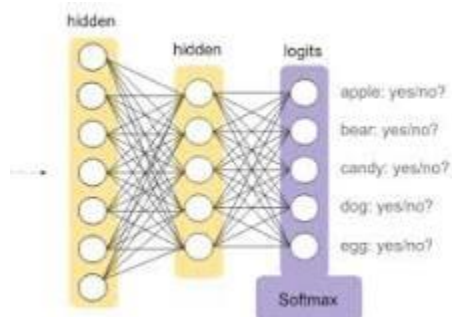
There are many similar alternatives which also work well. The Leaky ReLU is one of the most well known. It is the same as ReLU for positive numbers. But instead of being 0 for all negative values, it has a constant slope (less than 1.).

That slope is a parameter the user sets when building the model, and it is frequently called α . For example, if the user sets $\alpha=0.3$, the activation function is $f(x) = \max(0.3*x, x)$. This has the theoretical advantage that, by being influenced by x at all values, it may be make more complete use of the information contained in x .

There are other alternatives, but both practitioners and researchers have generally found insufficient benefit to justify using anything other than ReLU.

softmax function:

What is the softmax function in tensor flow?



That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. Softmax is implemented through a neural network layer just before the output layer.

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course
Title: Deep Learning

Test Exercise:

1. When deep learning start?
2. Who is father of deep learning?
3. How many layers deep learning algorithms are constructed?
4. What is the subset of machine learning?
5. The deep learning first layer is called the ___?
6. RNNs stands for ?
7. Which are the common use of RNNs?
8. CNN is mostly used when there is an?
9. Which neural network has only one hidden layer between the input and output?
10. Limitations of deep learning?

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course
Title: Deep Learning

Key:

1. 1943
2. FRANK ROSENBLATT
3. 3
4. Deep learning
5. Inner layer
6. Recurrent neural networks
7. Provide a caption for images
8. Unstructured data
9. Shallow neural network
10. Data labelling , obtain huge training datasets

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Department of Computer Science

Value Added Course
Title: Deep Learning

Marks List

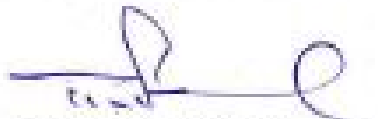
Class: IIBsc(MSCS)

S. No	Roll No.	Name of the Student	Marks
1	2155301	Krishnavarapu Dharanisri	09
2	2155302	Ghantasala Divya	08
3	2155303	Mopidevi Sri Lakshmi	09
4	2155304	Singavarapu Sai Sowmya	10
5	2155305	Arikatla Susanth	07
6	2155306	Bathina Manoj Phanindra	09
7	2155307	Jampana Keerthi Priya	08
8	2155308	Konatham Alekhya	09
9	2155309	Kondaraju Ajith Kumar	10
10	2155310	Nakka Anusha	09
11	2155311	Akula Chakradhar	08
12	2155312	Mohammad Khadeera begum	09
13	2155313	Bandela Pavan Kumar	08
14	2155314	Jonna Jhansi Lakshmi	08
15	2155315	Kunapareddy Tulasi	08

16	2155316	Peddiboyina Himasri	08
17	2155317	Katta Naga Srevani	08
18	2155318	Valluri Shainy	08
19	2155319	Manikonda Karuna Sri	08
20	2155320	Padmanabhuni Phani Supraja	08
21	2155321	Kunapareddy Hema sri	09
22	2155322	Reddy Durga Bhavani	08
23	2155323	Dokku Naga Gireesha	08
24	2155324	Nerusa Naga Mounika	09
25	2155325	Goriparthi Dodeepya	08
26	2155326	Edupuganti Joshimani	08
27	2155327	Gangisetty Yuva Kiran	08
28	2155328	Veerla Sri Lakshmi	08
29	2155329	Mamidi Chaitanya	07
30	2155330	Rachuri Bobby	07
31	2155331	Peteti Praneeth Kumar	10
32	2155332	Mohammad Abrar Ahamad	09
33	2155333	Goriparthi Harika	08
34	2155334	Vinnakota Deepthi	08



Signature of Lecturer



Signature of HOD



PRINCIPAL
Signature of Principal

AG & SC 2024-25
Arts & Science (Autonomous), Vuyyuru

Department of Computer Science

Value Added Course

Title: Deep Learning

Feed Back Form

1. Is the programme interested to you (Yes/No) ✓
2. Have you attended all the session (Yes/No) ✓
3. Is the content of the program is adequate (Yes/No) ✓
4. Have the teacher covered the entire syllabus? (Yes/No) ✓
5. Is the number of hours adequate? (Yes/No) ✓
6. Do you have any suggestions for enhancing or reducing the number of weeks designed for the program? (Yes/No) ✓
7. On the whole, is the program useful in terms of enriching your knowledge? (Yes/No) ✓
8. Do you have any suggestions on the program? (Yes/No) ✓

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course / Certificate Course - Attendance Register

Class / Section: BSC(MSC) Year: 1 Department of: Computer science Paper: Deep learning Lecturer: Teja Sri. Oleti

Sl. No	Roll No	Student Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1	2155301	Krishnavarapu Dharanisri	P	P	P	P	P	P	A	P	P	P	P	P	A	P	P	
2	2155302	Ghantasala Divya	P	P	P	P	P	P	P	P	P	A	P	P	P	A	P	
3	2155303	Mopidevi Sri Lakshmi	P	P	P	P	A	P	P	P	P	A	P	P	P	P	P	
4	2155304	Singavarapu Sai Sowmya	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	
5	2155305	Arikatla Susanth	P	P	P	A	P	A	P	A	P	P	P	P	A	P	P	
6	2155306	Bathina Manoj Phanindra	P	P	P	P	P	P	P	P	P	P	P	A	P	P	P	
7	2155307	Jampana Keerthi Priya	P	P	P	P	P	A	P	P	P	P	A	P	A	P	P	
8	2155308	Konatham Alekhya	P	P	P	P	A	P	P	P	P	P	P	P	P	A	P	
9	2155309	Kondaraju Ajith Kumar	P	P	P	P	P	P	P	P	P	A	P	A	P	P	P	
10	2155310	Nakka Anusha	P	P	P	P	P	P	P	A	P	P	A	P	P	A	P	
11	2155311	Akula Chakradhar	P	P	P	P	A	P	A	P	P	A	P	P	A	P	P	
12	2155312	Mohammad Khadeera begum	P	P	P	P	P	P	A	P	A	P	P	P	A	P	P	
13	2155313	Bandela Pavan Kumar	P	A	P	P	A	P	P	P	P	A	P	P	A	P	P	
14	2155314	Jonna Jhansi Lakshmi	P	P	P	P	P	A	P	P	A	P	P	P	A	P	P	
15	2155315	Kunapareddy Tulasi	P	P	P	P	A	P	P	P	P	A	P	P	A	P	P	
16	2155316	Peddiboyina Himasri	P	P	P	P	P	P	P	P	A	P	P	P	A	P	P	

A.G. & S.G. Siddhartha Degree College of Arts & Science
 Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course / Certificate Course - Attendance Register

Class / Section: B.Sc (MSTI) Year: 1 Department of: Computer Science Paper: Deep Learning Lecturer: Teja Sri. oleki

Sl. No	Roll No	Student Name	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Total
1	2155317	Katta Naga Sravani	P	P	P	P	P	A	P	P	P	A	P	P	P	P	P	
2	2155318	Valluri Shainy	P	P	P	P	P	P	A	P	P	P	P	A	P	P	P	
3	2155319	Manikonda Karuna Sri	P	P	P	P	P	P	P	P	A	P	P	P	A	P	P	
4	2155320	Padmanabhuni Phani Supraja	P	P	P	P	P	P	A	P	P	P	P	P	A	P	P	
5	2155321	Kunapareddy Hema sri	P	P	P	P	A	P	P	P	P	A	P	P	A	P	A	
6	2155322	Reddy Durga Bhavani	P	P	P	P	A	P	A	A	P	P	P	P	A	P	P	
7	2155323	Dokku Naga Giresha	P	P	P	P	P	P	A	P	P	A	P	P	P	P	P	
8	2155324	Nerusu Naga Mounika	P	P	P	P	P	P	A	P	A	P	P	P	P	P	P	
9	2155325	Goriparthi Dedeepya	P	P	P	P	P	P	A	P	A	P	P	P	P	P	P	
10	2155326	Edupuganti Joshimani	P	P	P	P	P	A	P	P	P	A	P	P	P	P	A	
11	2155327	Gangisetty Yuva Kiran	P	P	P	P	P	P	P	P	A	A	P	P	P	P	P	
12	2155328	Veerla Sri Lakshmi	P	P	P	P	P	P	P	A	A	P	P	P	P	P	A	
13	2155329	Mamidi Chaitanya	P	P	P	A	P	P	P	P	A	P	P	P	A	P	P	
14	2155330	Rachuri Bobby	P	P	P	P	P	P	P	A	P	A	P	A	P	A	P	
15	2155331	Peteti Praneeth Kumar	P	P	P	P	P	P	P	P	A	P	P	P	P	P	P	
16	2155332	Mohammad Abrar Ahamad	P	P	P	P	P	P	P	A	P	P	P	P	P	A	P	
17	2155333	Goriparthi Harika	P	P	P	P	P	P	P	P	P	P	P	P	P	P	A	
18	2155334	Vinnakota Dcephthi	P	P	P	P	P	A	P	P	P	P	P	P	A	A	P	

Signature of Lecturer

Signature of HOD

Signature of Principal
 AG & SG Siddhartha Degree College of
 Arts & Science (Autonomous), Vuyyuru

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course / Certificate Course - Attendance Register

Class / Section: BSC(MSC) Year: 1 Department of: Computer science Paper: Deep learning Lecturer: Teja Sri. Oleti

Sl. No	Roll No	Student Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1	2155301	Krishnavarapu Dharanisri	P	P	P	P	P	P	A	P	P	P	P	P	A	P	P	
2	2155302	Ghantasala Divya	P	P	P	P	P	P	P	P	P	A	P	P	P	A	P	
3	2155303	Mopidevi Sri Lakshmi	P	P	P	P	A	P	P	P	P	A	P	P	P	P	P	
4	2155304	Singavarapu Sai Sowmya	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	
5	2155305	Arikatla Susanth	P	P	P	A	P	A	P	A	P	P	P	P	A	P	P	
6	2155306	Bathina Manoj Phanindra	P	P	P	P	P	P	P	P	P	P	P	A	P	P	P	
7	2155307	Jampana Keerthi Priya	P	P	P	P	P	A	P	P	P	P	A	P	A	P	P	
8	2155308	Konatham Alekhya	P	P	P	P	A	P	P	P	P	P	P	P	A	P		
9	2155309	Kondaraju Ajith Kumar	P	P	P	P	P	P	P	P	P	A	P	A	P	P		
10	2155310	Nakka Anusha	P	P	P	P	P	P	P	A	P	P	A	P	P	A		
11	2155311	Akula Chakradhar	P	P	P	P	A	P	A	P	P	A	P	P	A	P		
12	2155312	Mohammad Khadeera begum	P	P	P	P	P	A	P	A	P	P	P	A	P	P		
13	2155313	Bandela Pavan Kumar	P	A	P	P	A	P	P	P	P	A	P	P	A	P		
14	2155314	Jonna Jhansi Lakshmi	P	P	P	P	P	A	P	P	A	P	P	P	A	P		
15	2155315	Kunapareddy Tulasi	P	P	P	P	A	P	P	P	A	P	P	P	A	P		
16	2155316	Peddiboyina Himasri	P	P	P	P	P	P	P	P	A	P	P	P	A	P		

A.G. & S.G. Siddhartha Degree College of Arts & Science

Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course / Certificate Course - Attendance Register

Class / Section: BSC (MSCI) Year: 1 Department of: Computer Science Paper: Deep Learning Lecturer: Teja Sri. oleki

Sl. No	Roll No	Student Name	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Total
1	2155317	Katta Naga Sravani	P	P	P	P	P	A	P	P	P	A	P	P	P	P	P	
2	2155318	Valluri Shainy	P	P	P	P	P	P	A	P	P	P	P	A	P	P	P	
3	2155319	Manikonda Karuna Sri	P	P	P	P	P	P	P	A	P	P	P	P	A	P	P	
4	2155320	Padmanabhuni Phani Supraja	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	
5	2155321	Kunapareddy Hema sri	P	P	P	P	A	P	P	P	A	P	P	A	P	A		
6	2155322	Reddy Durga Bhavani	P	P	P	P	A	P	A	A	P	P	P	A	P	P		
7	2155323	Dokku Naga Gireesha	P	P	P	P	P	P	A	P	P	A	P	P	P	P	P	
8	2155324	Nerusu Naga Mounika	P	P	P	P	P	P	A	P	A	P	P	P	P	P	P	
9	2155325	Goriparthi Dedeepya	P	P	P	P	P	P	A	P	A	P	P	P	P	P	P	
10	2155326	Edupuganti Joshimani	P	P	P	P	P	A	P	P	P	A	P	P	P	P	A	
11	2155327	Gangisetty Yuva Kiran	P	P	P	P	P	P	P	A	A	P	P	P	P	P		
12	2155328	Veerla Sri Lakshmi	P	P	P	P	P	P	P	A	A	P	P	P	P	A		
13	2155329	Mamidi Chaitanya	P	P	P	A	P	P	P	A	P	P	P	A	P	P		
14	2155330	Rachuri Bobby	P	P	P	P	P	P	A	P	A	P	A	P	A	P		
15	2155331	Peteti Praneeth Kumar	P	P	P	P	P	P	P	A	P	P	P	P	P	P		
16	2155332	Mohammad Abrar Ahamad	P	P	P	P	P	P	P	A	P	P	P	P	P	A	P	
17	2155333	Goriparthi Harika	P	P	P	P	P	P	P	P	P	P	P	P	P	P	A	
18	2155334	Vinnakota Deepthi	P	P	P	P	P	A	P	P	P	P	P	A	A	P		

Signature of Lecturer

Signature of HOD

Signature of Principal

AG & S.G. Siddhartha Degree College of
Arts & Science (Autonomous), Vuyyuru



**ADUSUMILLI GOPALAKRISHNAIAH & SUGARCANE GROWERS
SIDDHARTHA DEGREE COLLEGE OF ARTS & SCIENCE**

Vuyyuru-521 165, Krishna District, Andhra Pradesh
An Autonomous College in the Jurisdiction of Krishna University
Accredited by NAAC with "A" Grade



DEPARTMENT OF COMPUTER SCIENCE

VALUE ADDED COURSE:DEEP LEARNING

VAC CODE:DLVAC01

CERTIFICATE

This is to Certify that

Son /Daughter of shri/Smt

has Successfully completed value added course in DEEP LEARNING

Conducted by the Department of COMPUTER SCIENCE from 15-03-2023 to 25-04-2023 We wish him /her
bright future

Co-ordinator

Head of Department

Principal