Chapter 1

**Introduction to**

**DEEP LEARNING**

*Introduction to Neural Networks*

*Supervised Learning with Neural Networks*

*Why is Deep Learning taking off?*

**1.1 NEURAL NETWORKS**

* **What is a Neural Network?**

Neural networks refer to broad type of non-linear models/parametrizations hθ(x) that involve combinations of matrix multiplications and other entrywise non-linear operations. We will start small and slowly build up a neural network, step by step.

**ChatGPT (sans-serif) :**

This gifted genius was one of the last people to have mastered most major fields of knowledge—an impossible accomplishment in our own era of specialization. He was an expert in law, religion, philosophy, literature, politics, geology, metaphysics, alchemy, history, and mathematics. Leibniz was born in Leipzig, Germany. His father, a professor of moral philosophy at the University of Leipzig, died when Leibniz was six years old. The precocious boy then gained access to his father’s library and began reading voraciously on a wide range of subjects, a habit that he maintained throughout his life. At age fifteen he entered the University of Leipzig as a law student and by the age of twenty received a doctorate from the University of Altdorf.

1. String
2. Integer
3. Float
4. Boolean

Left Justify:

**Arial: Line spacing (1.15) :**

**Calibri**

**Tahoma**

**Main Text: Directa-serif, Justify**

**Italic-notes 1: Times new roman, Left justify**

**Italic-notes 2: Light cyan, Times new roman, Left justify**

**Note 1: Yellow, Left justify**

**Note 2 (chat GPT): Calibri light green, Left justify; Heading (Arial/Calibri) DARK CYAN**

**Main Heading (12 pt):**

**2.1 FAMILIES OF FUNCTIONS**

**Sub headings**

**Sub headings**

**Note Heading**

Chapter – 2:

**Chapter name**

Description

**Calibri:** One of the important themes in calculus is the analysis of relationships between physical or mathematical quantities. Such relationships can be described in terms of graphs, formulas, numerical data, or words. In this chapter we will develop the concept of a “function,” which is the basic idea that underlies almost all mathematical and physical relationships, regardless of the form in which they are expressed. We will study properties of some of the most basic functions that occur in calculus, including polynomials, trigonometric functions, inverse trigonometric functions, exponential functions, and logarithmic functions

**Calibri:** One of the important themes in calculus is the analysis of relationships between physical or mathematical quantities. Such relationships can be described in terms of graphs, formulas, numerical data, or words. In this chapter we will develop the concept of a “function,” which is the basic idea that underlies almost all mathematical and physical relationships, regardless of the form in which they are expressed. We will study properties of some of the most basic functions that occur in calculus, including polynomials, trigonometric functions, inverse trigonometric functions, exponential functions, and logarithmic functions

**3 basic colors**

Light yellow (255, 254, 221)

Light Cyan (217, 255, 255)

Light Green (238, 255, 221)

High lights:

Line spacing ()

(232, 0, 97) : color code

**Line spacing ( ) :**

**(0, 174, 192 ): color code :**

**Line spacing () :**

**(39, 99, 207): color code :**

Notes in Sans-Serif:

*Italic Comments (serif) The development of calculus in the seventeenth and eighteenth centuries was motivated by the need to understand physical phenomena such as the tides, the phases of the moon, the nature of light, and gravity*

***Italic Comments (serif, Times New Roman):*** *The development of calculus in the seventeenth and eighteenth centuries was motivated by the need to understand physical phenomena such as the tides, the phases of the moon, the nature of light, and gravity.*

***Italic Comments (serif, Times New Roman):*** *The development of calculus in the seventeenth and eighteenth centuries was motivated by the need to understand physical phenomena such as the tides, the phases of the moon, the nature of light, and gravity.*

ChatGPT (sans-serif)

**ChatGPT (sans-serif) :**

This gifted genius was one of the last people to have mastered most major fields of knowledge—an impossible accomplishment in our own era of specialization. He was an expert in law, religion, philosophy, literature, politics, geology, metaphysics, alchemy, history, and mathematics. Leibniz was born in Leipzig, Germany. His father, a professor of moral philosophy at the University of Leipzig, died when Leibniz was six years old. The precocious boy then gained access to his father’s library and began reading voraciously on a wide range of subjects, a habit that he maintained throughout his life. At age fifteen he entered the University of Leipzig as a law student and by the age of twenty received a doctorate from the University of Altdorf.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Figures | | | | This gifted genius was one of the last people to have mastered most major fields of knowledge—an impossible accomplishment in our own era of specialization. He was an expert in law, religion, philosophy, literature, politics, geology, metaphysics, alchemy, history, and mathematics. Leibniz was born in Leipzig, Germany. His father, a professor of moral philosophy at the University of Leipzig, died when Leibniz was six years old. The precocious boy then gained access to his father’s library and began reading voraciously on a wide range of subjects, a habit that he maintained throughout his life. At age fifteen he entered the University of Leipzig as a law student and by the age of twenty received a doctorate from the University of Altdorf. | | | | | | | |

Subsequently, Leibniz followed a career in law and international politics, serving as counsel to kings and princes. During his numerous foreign missions,

Leibniz came in contact with outstanding mathematicians and scientists who stimulated his interest in mathematics—most notably, the physicist Christian Huygens. In mathematics Leibniz was self-taught, learning the subject by reading papers and journals. As a result of this fragmented mathematical education,

Leibniz often rediscovered the results of others, and this helped to fuel the

debate over the discovery of calculus.

Leibniz never married. He was moderate in his habits, quick-tempered

but easily appeased, and charitable in his judgment of other people’s work.

In spite of his great achievements, Leibniz never received the honors showered on Newton,

and he spent his final years as a lonely embittered man. At his funeral there was one mourner,

his secretary. An eyewitness stated, “He was buried more like a robber than what he really

was—an ornament of his country.”

**1.2 Implementation**

Session 1: part\_3 END- 6:36

**4.4 Asynchronous**

Session 2: part\_1 (start 11:43)

* Index of string - SUBSCRIPTING: **"Hello"[3]** gives 4th 'l'.
* Declaring Integers: Just write the number, no keyword needed.
* Other format: Other thing is **123456** is can be written as **123\_456** it is slimier to *123,456*.
* Declaring floats: Just write the number, no keyword needed.
* Declaring Boolean: Just write ***True*** or ***False***, no keyword needed. Notice Uppercase is used.

padded\_questions\_in\_batch = **np.array**(**apply\_padding**(questions\_in\_batch, questnWrd2Int))

padded\_answers\_in\_batch = **np.array**(**apply\_padding**(answers\_in\_batch, ansWrd2Int))

**yield** padded\_questions\_in\_batch, padded\_answers\_in\_batch

padded\_questions\_in\_batch = **np.array**(**apply\_padding**(questions\_in\_batch, questnWrd2Int))

padded\_answers\_in\_batch = **np.array**(**apply\_padding**(answers\_in\_batch, ansWrd2Int))

**yield** padded\_questions\_in\_batch, padded\_answers\_in\_batch

**2.2 Type casting & checking with type()**

**type**(data)

* Type casting: Casting in python is done using constructor functions:

int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

**str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals

**type**(data)

* Type casting: Casting in python is done using constructor functions:

**int()** - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

**float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

**str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals

* Example:

|  |  |  |
| --- | --- | --- |
| #*Integers:*  x = **int**(1)   #*x will be 1*  y = **int**(2.8) #*y will be 2*  z = **int**("3") #*z will be 3* | #*Floats:*  x = **float**(1)     #*x will be 1.0*  y = **float**(2.8)   #*y will be 2.8*  z = **float**("3")   #*z will be 3.0*  w = **float**("4.2") #*w will be 4.2* | #*Strings:*  x = **str**("s1") #*x will be 's1'*  y = **str**(2)    #*y will be '2'*  z = **str**(3.0)  #*z will be '3.0'* |

* Exercise 2.1: sum of two digit number

two\_digit\_number = **input**("Type a two digit number :")

**print**("Type of this input is : " + **type**(two\_digit\_number))

type casting **is** happening below

sum = **int**(two\_digit\_number[1]) + **int**(two\_digit\_number[0])

**print**(sum)

**2.3 Mathematical operations**

Similar to other programming languages.

* Dividing ***int*** returns ***float*** output.
* Exponent sign: In python " **\*\*** " is used for power. c/c++/Java/C# has no this power operator.
* Order of precedence matters: remember "PEMDASLR"

|  |  |
| --- | --- |
| * Exercise 2.2: Change following to return 3.0, instead of 7.0. | #*POMDASLR rule applied*  **print**(3\*3 + 3 / 3 -3 )  **print**(((3\*(3 + 3)) / 3) -3 ) |

* Exercise 2.3: BMI calculator (Body-Mass-Index)

weight = **input**("Enter weigt : ")

heihgt = **input**("Enter height : ")

bmi = **int**(weight)/(**int**(heihgt)\*\*2)

**print**("The result is : " + **str**(**int**(bmi)))

**2.4 Rounding numbers in python**

* round(): Use **round**(float, precision)
* floor division " // " : **print(8 // 3)** gives floored number directly. As an int.
* Shorthand operators: **x = x/2** is similar to **x /= 2**. In form "variable **operator=** variable"

Also don’t need "+" for concatenation. It's more like C's formatted output

**2.5 f-string formatted output**

* In this case we don’t need any type conversion.
* Also don’t need "+" for concatenation. It's more like C's formatted output.

#*f-string*

score = 0

height = 1.8

isWinning = **True**

**print**(f"your score is {score}, your height is {height}. Are you winning: {isWinning}")

* As you see data is goes inside {}. And format is

**print**(**f**"{data1}string{data2}string{data3}… .. .. ")

* Exercise 2.3: Life in weeks.

age = **input**("What is oue current age? :")

remaining\_year = 90 - **int**(age)

remaining\_month = remaining\_year\*12

remaining\_week = remaining\_year\*52

remaining\_day = remaining\_year\*365

message = f"If you live upto 90 years then you have remaining {remaining\_year} years, or {remaining\_week} weeks, or {remaining\_month} months, or {remaining\_day} days"

**print**(message)

* Exercise 2.4: Final project, Tip calculator:

total = **input**("What was the total bill? : $")

percentage = **input**("What parcentage tip ypu like to give? 0, 10, 12, or 15 ? :")

split = **input**("How many people you want to split the bill? :")

payTotal = **float**(total) + (**float**(percentage)/100.0)\***float**(total)

finalPayment = **round**(payTotal/**int**(split), 2)

**print**(f"Each person should pay: ${finalPayment}")

**2.6 More formatted output:**

What happens if the result shows 14.0 instead of 14.00?

**"{:.2f}".format(finalPayment)**

Above formatting displays any number in 2 digit precision float number.

total = **input**("What was the total bill? : $")

percentage = **input**("What parcentage tip ypu like to give? 0, 10, 12, or 15 ? :")

split = **input**("How many people you want to split the bill? :")

payTotal = **float**(total) + (**float**(percentage)/100.0)\***float**(total)

finalPayment = **round**(payTotal/**int**(split), 2)

**print**(f"Each person should pay: ${finalPayment}")

**#*to inspect formatting works : use $150  with 12% splitted to 5***

**formatted = "{:.2f}".format(finalPayment)**

**print(f"Each person should pay (Formatted): ${formatted}")**