Introduction to ECE4179/5179 - Neural Networks and Deep Learning

Unit Information

- Lectures
- Labs
- Practicals
- Assignments

Lectures

- Please stay up to date with lectures!
- Content of Labs and Practicals is directly related to previous weeks’ Lectures!

Labs

- For beginners please watch the “Set up for Labs!” video under the Labs section on Moodle!
- All labs will be presented using Jupyter Notebooks
- Lab starter code will be uploaded to the corresponding week on Moodle every Friday for the next week
- Example solutions for Labs will be uploaded at the end of every week with short explanation video
- This unit is “Neural Networks and Deep Learning” not “Python Programming” we will spend time teaching you the basics but we EXPECT you to practice in your own time, we get to advanced topics very quickly

ECE4179/5179 - Neural Networks and Deep Learning Lab Sessions

Week 1	Introduction to Python
Week 2	Numpy for Python
Week 3	Linear Regression
Week 4	Introduction to PyTorch
Week 5	Multi-Layer Perceptron (MLP)
Week 6	Convolutional Neural Networks (CNN)
Week 7	Transfer Learning & Model Manipulation
Week 8	Image Segmentation
Week 9	YOLO - (Object Detection)
Week 10	Reinforcement Learning (RL)
Week 11	Generative Adversarial Networks (GAN)

Practicals

- All practical sessions are online.
- Zoom meeting codes will be/have been sent out - check your emails!
- No practical week 1!
- Questions will be uploaded Friday the week before.
- Solutions will be uploaded at the end of every week.
- Please attempt the questions before the session, though time will be given during the session to answer the questions.
- Attend every one of them and ask questions! These session are for you to develop your understanding of the content not just to solve problems!
- Similar question style to the exams

Assignments

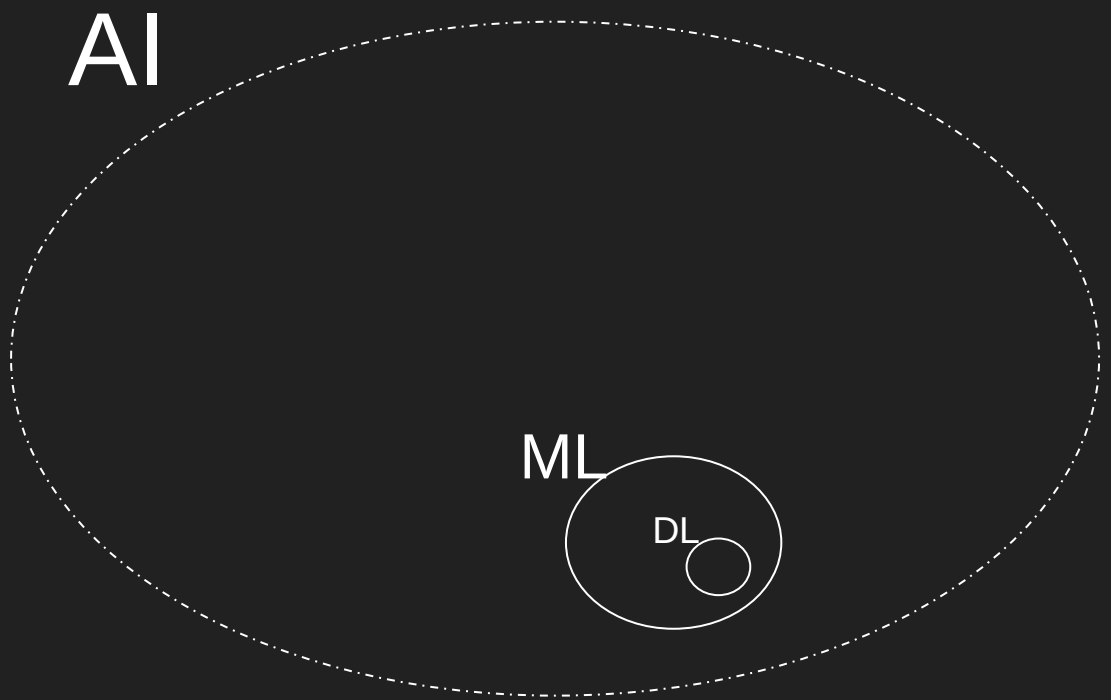
	Assignments	Quiz	Course project
Week 1			
Week 2		Quiz#1 (2%)	
Week 3	Assignment#1 (5%)		
Week 4		Quiz#2 (2%)	
Week 5	Assignment#2 (10%)		
Week 6		Quiz#3 (2%)	Course Project (15%)
Week 7			
Week 8	Assignment#3 (10%)	Quiz#4 (2%)	
Week 9			
Mid Semester Break			
Week 10		Quiz#5 (2%)	
Week 11			
Week 12		Quiz#6 (2%)	Project presentation

Assignments

Bellow is Subject to change!

- All assignments will require coding, you can use code from previous lab solutions
- Almost ALL marks are for quality of your results and discussion!
  - Finish writing code asap!
  - Project reports need to contain ALL your results!
  - ALL results require discussion!
- The assignments are about your understanding of content and your creativity in solving problems
- Please do not put code in your report unless it is relevant to the question!
- Reports should be neat and well organized and submitted as a pdf
  - The report is the only thing that gets marked!
- All code must be submitted along side your report
  - Code can be either Jupyter Notebook or Python script
  - Code must run and produce ALL results with NO modifications

Where are we?



What to expect

- Maths
- Abstract Concepts
- Data Analysis Techniques
- Cool demos
- Learning Practical Skills
- Real world Applications
- A lot of “Maybes”
    - This is a new field, many of the inner workings are still up for debate!

Intro to Python

Lab1

Python Basics

What is Python?

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is designed to be a human readable language.
- It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.
- Python is a great language for the beginner-level programmers. It supports developing a wide range of applications and Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Python Basics

What are the key properties of Python Language?

- **Interpreted** - Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- **Interactive** − You can sit at a Python prompt and interact with the interpreter directly to write your programs. Python also allows interactive testing and debugging of snippets of code.
- **Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python Basics

1. Assigning Values to Variables
2. Operators (Arithmetic operators, Assignment operators, Comparison operators, Logical operators, Identity operators, Membership operators, Bitwise operators)
3. Standard Data Types (Numbers, String, List, Tuple, Dictionary)
4. Decision Making
5. Loops
6. Functions
7. Classes & Objects

Python Basics

Jupyter Notebooks

Jupyter Notebook is a web based interactive programming tool that allows you to run individual “cells” of code at a time.

It is very useful for teaching AND developing data analysis programs.

The Jupyter file system viewer will default to your C: drive (or wherever you launched Jupyter from)

Python Basics

Jupyter Notebooks

How to use:

- A “cell” can be run by pressing the “run” button, Or click inside the cell (edit mode) and:
  - Press, shift+enter - run and move to the next cell (or create one is there isn’t one after)
  - Press, ctrl+enter - run and stay in the same cell
- You can select the side of the cell (command mode) and Press:
  - a - will add a new cell above the current
  - b - will add a new cell under the current
  - d, d - will delete the current cell
    - NOTE: deleting cells will not delete the variables created in that cell!!
- Under “Kernel” you can restart your code which will delete all variables and restart your session

Python Basics

Jupyter Notebooks

More tips!:

- Use the “arrows” to rearrange the order of cells
- The square “stop” button will interrupt code that is currently running, but it will NOT end your session.
- You can change the “Type” of your cell and write non-code text
- The keyboard icon will tell you what shortcuts are available!

Python Standard Data Types

Example	Data Type
x = 20	int
x = 20.5	float
x = 1j	complex
x = "Hello World"	str
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = {"name" : "John", "age" : 36}	dict

Python Conditional logic

Creating code that is only executed if certain conditions are met is a important tool is programming.

In Python creating such Conditional logic statements is easy!

```
var = 100

if (var==50):
    print('Value of expression is 50')
elif(var==100): #else if statement is written as elif in python.
    print('Value of expression is 100')
else:
    print('Value of expression is neither 50 or 100')
```

Python Conditional logic

We can also use other logical operators to to make our Conditional Statements more complicated.

```
var1 = 200
var2 = 100

if var1 > 100 and var2 < 200:
    print('Both statements are True!')

if var1 < 100 or var2 < 200:
    print('At least one of these statements are True!')

var3 = 300
if not (var1 < 100 and var2 > 200 and var3 < 250):
    print('None of these statements are True!')
```

```
Both statements are True!
At least one of these statements are True!
None of these statements are True!
```

Python Looping

For - Loops through a sequence

While - Loops while a condition is True

Python Functions

Functions - A block of code which only runs when it is called.

In Python a function is defined using the def keyword.

```
def my_function():
  print("Hello from a function")

my_function()
```

Python Functions

Return values

A function can return information to the main program via

the return keyword.

```
def my_function(x):
  return 5 * x


print(my_function(3))
```

Python Iterators

Creating an iterator

Lists, tuples and dictionaries are all iterable objects.

They are iterable containers which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

Python Functions

Scope of Variables

The scope of a variable determines where in the code a variable can be accessed.

Variables can have either global or local scope.

Global variables: variables declared outside of a function can be accessed anywhere in the program.

Local variables: variables created inside are function can only be accessed in that function.

Python Functions as Iterators

We can create a function that also acts as an iterator using a for loop.

Here we can use the "yield" keyword to return value at each step of a loop.

```
def iter_func(a, b):
    for i in range(a):
        yield i*b

for i in iter_func(10, 2):
    print(i + 5)
```

Python Classes

Classes & Objects

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

Example : Class - Car

Properties - 4 wheels, 4 doors

Methods - Drive forward, Reverse

```
class Car:
  wheels = 4
  doors = 4
```

To create a class, use the keyword class:

Python Classes

The __init__() Function

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Python Classes

Create a class named Person, use the __init__() function to assign values for name and age:

The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Python Basics

Object Methods

As well as the inbuilt Methods Objects can any methods we want.

Let us create a method in the Person class:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Python Basics

Python naming rules:

- Variable names may contain letters, digits (0-9) or the underscore character _
- Variable names must begin with a letter from A-Z or the underscore _ character
- Variable names may not be a reserved keyword in Python

Python Basics

Python style convention (from PEP 8):

camelCase (CamelCase) multiple words broken up by capitalization:

- Classes

snake_case multiple words separated by an underscore:

- Functions/Methods
- Variable names