UCCD3074 Deep Learning for Data Science

PRACTICAL ASSIGNMENT (20 marks)

This course project is the opportunity for you to apply what you have learnt in class to a problem of your interest. There are two project options you can pick on:

Option 1: Your own project

Problem Description

Those who take this option can select their own topic to work on. The projects can fall into *any one* of the following tracks:

1. **Applications**. You can apply a deep learning to tackle any problem of your interest. The problem must be sufficiently complex and worthy of a deep learning architecture. You may look at the following resources for inspirations or potential projects:
   - Kaggle challenges
   - CS231 Projects
   - CS229 Projects

2. **Rebuild a system**. You can go through any recent research paper and re-implement the system. However, you are limited to papers from good conferences:
   - CVPR: IEEE Conference on Computer Vision and Pattern Recognition
   - ICCV: International Conference on Computer Vision
   - ECCV: European Conference on Computer Vision
   - NIPS: Neural Information Processing Systems
   - ICLR: International Conference on Learning Representations

   Most authors make their code public. You may refer to them. However, you are expected to re-implement this system from scratch. You are not allowed to copy the public code, make superficial modifications and pass them up as your assignment.

3. **Analysis and evaluation**. You can write algorithms or devise experiments that deepens your understanding on the topic. For example, you can devise experiments to compare the performance of different deep architectures or learning algorithms on a particular task.

4. **Model**. You may work to build a novel model or algorithm for deep learning. Students can work to devise new methods on network architecture, learning strategies, visualization, etc.

Rules and Regulations

1. For task 1 to 3, you may work on an existing problems or reuse any existing techniques. The requirement is that you build your own deep learning systems with your own effort.

2. Your project can be related to your FYP title but it must not be a deliverable to your FYP project.

Option 2: Learning to add

Problem Description

This objective of this assignment is to implement a sequence-to-sequence learning for performing addition where the input is a sequence of characters (the addition statement). The expected output is also a sequence of characters (the result). For example:

Input: "535+61"
Output: "596"

Starter Kit

This assignment comes with a dataset (*learntoadd_data.pkl*) and a notebook file (*learning_to_add.pkl*). The notebook contains the code to load the dataset.

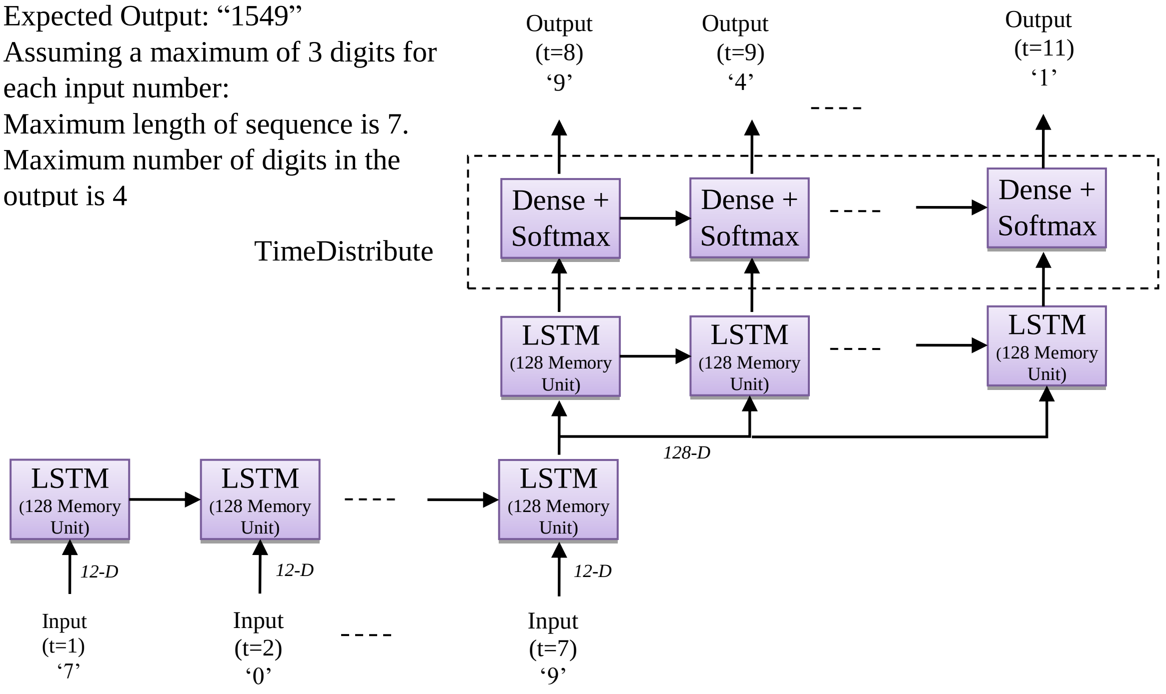
Assumptions

- Each character is presented to your system one character at a time. You can make the following assumptions:
- Each number is restricted to a maximum number of digits
- There are no spaces in the middle of the addition statements.
- Spaces can be padded to the left or right of the statements to make the size of each sequence consistent.

Suggested architecture

- The following architecture is capable of achieving at least 92% accuracy for this problem. It is based on an encoder-decoder architecture.

Input sequence: “942+607”
Expected Output: “1549”
Assuming a maximum of 3 digits for each input number:
Maximum length of sequence is 7.
Maximum number of digits in the output is 4



Enhancements:

Any enhancement to the requirement is most welcomed and would be awarded accordingly. For example, (1) less assumptions (e.g., arbitrary sequence length), (2) better architecture or techniques (e.g., inverting the input) to improve performance, or (3) any other changes that are not superficial in nature.

Project Description

Project demonstration

1. All teams must perform a demo of their system to the lecturer in Week 13.
2. During the session, team members would be interviewed about their systems. They may be asked to make simple on-site changes.
3. Teams found to not understand their own system would be **penalized** up to 10 marks.

Short report

1. **Outline for the report**:
   - Problem statement/description
     - Description of the application being tackled
   - Describe about your proposed architecture and algorithms
     - Preprocessing steps, network architecture, special learning strategies (if any)
   - Experiments and analysis
     - Description the your dataset, experimental setup, experimental results, observations and analysis
2. **Expected number of pages**: 2 to ∞
3. Be **concise**. Marks are allocated based on clarity and content, not how many pages.
4. **Point forms** are allowed.

Submission

1. **Deadline**:
   - **Week 12** Sunday (8 April 2018): submission to WBLE
   - **Week 13**: Interview sessions
2. **WBLE Submission**:
   - A zip file containing the following items
     - The directory containing all your codes
     - A technical report (word document)
   - No hard copy is required.

Rules and regulation

1. This project is a **group project**. Allowable number of team members are 1 or 2 members.
2. All projects must be coded in **Python** and/or **Keras**.
3. All projects must be verified with experiments and evaluations.
4. Cite your sources (papers, github codes, etc).
5. **Late submission** may be imposed a penalty of up to 10 marks.
6. The **core part** of your system must be **built from scratch**. Do not copy the codes, make superficial changes and submit the code as yours.
7. Any students caught with **plagiarism** may result in 0 marks for his/her assignment including the group that allows their work to be plagiarized.

Marking Scheme:

- **Writeup**: 20%
  - clarity, structure, language, references, good insights and discussion of methodology, good analysis and results
- **Technical**: 40%
  - Proposed system, difficulty and depth, innovation
- **Evaluation and results**: 30%
  - Sound evaluation metric, result and performance, thoroughness in analysis
- **Interview Sessions**: 10%
  - Familiarity with system