**IDE 2: Computational Intelligence Tutorial**

**Week 2**

April, 2018

**Introduction**: This week you will were introduced into the basic principles and intuition behind some main neuro-inspired computing methods. As in the previous tutorial, here you are also asked to (i) follow the detailed instructions on the doc and ipynb files and (ii) answer their qualitative questions to demonstrate a deeper understanding of the employed tools. You are again expected to run all steps presented in the ipynb files in your own jupyter notebook environment.

Completion of one out of three exercises will reflect a satisfactory familiarisation with the tutorial’s content. Completion of two or three exercises will reflect a fuller exposure and a top degree of familiarisation at this stage of the course.

**A. Convolutional Neural Networks (CNN) for CIFAR-10 Dataset**

The CIFAR-10 data set consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Recognizing photos from the cifar-10 collection is one of the most common problems in the today’s world of machine learning. This tutorial is going to show you – step by step – how to build multi-layer artificial neural networks that will recognize images from a cifar-10 set and visualize it.

**Keras**

To build our CNN (Convolutional Neural Networks) we will use Keras. Keras is an open source neural network Python library which can run on top of other machine learning libraries like TensorFlow, CNTK or Theano. It allows for an easy and fast prototyping, supports convolutional, recurrent neural networks and a combination of the two.

**CIFAR-10 Dataset**

The dataset is divided into five training batches and one test batch, each with 10000 images.

The test batch contains exactly 1000 randomly-selected images from each class.

The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another.

Between them, the training batches contain exactly 5000 images from each class.

In this tutorial, we will guide you to be familiar with building a convolutional neural network:

1) load and visualize the dataset

2) pre-processing the dataset

3) build the CNN model

4) train and test the model

**Follow the instructions on jupyter notebook, and answer the following questions:**

1) Try to increase the number of filters and the filter size.

What will happen? Will this improve the testing accuracy?

2) Try to change the number of batch sizes or the number of epochs.

How does these two parameters affect the network performance?

3) Optional: create your own network, or follow the larger network provided, does deeper network help to improve the performance?

**B. Long Short-Term Memory (LSTM): Text generation based on Alice in Wonderland**

**1. Task**: Given the textual book *Alice’s Adventure in Wonderland,* you are required to build up a simple LSTM model which is able to learn the dependencies between characters and the conditional probabilities of characters in sequences so that we can in turn generate wholly new and original sequence of characters. For example, after necessary training, the model should be able to generate a number of characters in sequences given a fixed length of characters:

* Given 10 characters: “Alice was ”  possible output of 100 characters is: “Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to”

**2. Datasets**: You can download the complete text in ASCII format (Plain Text UTF-8) for this book for free: <http://www.gutenberg.org/files/11/11-0.txt>

Hints:

* A clean-up of the downloaded datasets is necessary. You may want to make sure all words in lower case, and omit blank lines.
* Inputs: to simplify the task, the inputs can be in fixed length, say 10 characters.
* Outputs: similarly, the outputs are also in fixed length, say 100 characters.
* To create input and output datasets which are exclusively numerical, each character should be represented with a one-hot vector.
* It is suggested to generate an example of text every time when one training iteration is finished, so that we can get a feeling of how much our model has been improved during the training process.

GPU training:

If your jupyter notebook crashes when running the training process or you need a powerful GPU, you may want to try this online deep learning resource Colab provided by Google for **free**: <https://colab.research.google.com/>

How to use it: <https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d>

# **Exercise 3: Sentiment Analysis**

A piece of text usually conveys an author's attitude towards a certain topic. This can be positive, negative, or neutral (e.g. Reviews on Amazon, IMDB, RottenTomatoes, etc..). Sentiment Analysis tries to infer the *sentiment* via computational modelling of text. In this tutorial, we will build a simple LSTM model to perform sentiment classification.

Dataset: tweets about First GOP (Grand Old Party) Debate 2016.

Description: *We looked through tens of thousands of tweets about the early August GOP debate in Ohio and asked contributors to do both sentiment analysis and data categorization. Contributors were asked if the tweet was* ***relevant****, which* ***candidate*** *was mentioned, what* ***subject*** *was mentioned, and then what the* ***sentiment*** *was for a given tweet.*

**Instruction:**

**1.** You might need to install pandas (data/table processing library) before starting this tutorial

pip install pandas

**2.** Follow the notebook

Exercise 3 has 3 major components, which we will guide you through in depth and in a pedagogical manner.

1. Preprocessing the data, then prepare the input and output for the network
2. Build a network model for classification
3. Train & Evaluate the network

**3.** Think about the questions attached at the end of the notebook, and try to come up with answers:

Q1. Why is the accuracy for positive sentiment so much lower than the negative one? There are no true answer to this question. Come up with as many reasons as you can. (Hint: for example, look at the number of examples for positive/negative sentiment. Can this be one of the reasons?)

Q2. How did the hyper-parameters affected the performance? How would you improve the network performance further?