**Indian Institute of Technology, Mandi**
**February - May 2019**

# CS671 - Deep Learning and its Applications

# Programming Assignment 1

Course Instructor : Aditya Nigam

04 March 2019

## Instructions

- Plagiarism is strictly prohibited. In case of violation of the same, a zero will be awarded for this assignment as a warning and a quick F grade if repeated later.
- Submit a README.MD file for each question as well as a full assignment which provides the instructions for running your codes in detail including the versions of programming language and all the modules that have been used.
- Students using Windows or other OS are requested to make sure that their code runs perfectly on Linux as mentioned in each problem. Your evaluation will be done on computers in the PC lab.
- Zip the three zips made one for each question as group_ID.zip e.g., **13.zip**.
- Only submit documents that are mentioned in the submission sub-section of each problem, on **Moodle**. Other stuff has to be shown separately.
- The deadline for submission is **Friday, 15$^{th}$ March, 2019, 2200 HRS**. No late submissions will be entertained.
- You are required to upload your codes on **Github** as well as Moodle for all the assignments.
- You are required to make a web-page for your progress in this course. Links to your codes, videos, project pages, should be present on this web-page. Details of this will be conveyed shortly via Moodle.
- The Github and web-page requirement carry **10** marks for each assignment. So take it seriously.
- Contact Aayush Mishra and Daksh Thapar for any queries.

# 1 Data Management - Line Dataset Generation (20 marks)

## 1.1 Motivation

This problem is aimed at introducing you to the housekeeping around any Machine Learning problem, Data Management. Data that is not properly labelled becomes hard to manage. You will learn about **numpy** array manipulation and basic image handling by solving this problem.

## 1.2 Problem Statement

Make a dataset of $(28 \times 28 \times 3)$ images of **straight lines** on a **black background** with the following variations:

1. **Length** - 7 (short) and 15 (long) pixels.
2. **Width** - 1 (thin) and 3 (thick) pixels.
3. **Angle with X-axis** - Angle $\theta \in [0^o, 180^o)$ at intervals of $15^o$.
4. **Color** - Red and Blue.

## 1.3 Notes

- The number of classes of images is **fixed**. You are required to make 1,000 random variations within each class. Note that only translation of the centre of the line can be used for intra-class variations.
- Any image of a particular type will **not** be correct if it doesn't fully confirm with all its said properties. For example, any image which has a long line can not contain any shorter or longer line than 15 times the pixel side length (small deviations due to pixels allowed). Distances are to be Euclidean i.e. diagonal of a pixel = $\sqrt{2} \times side$.

## 1.4 Submission

- Name each image with underscore separated variation labels in the order mentioned before. Variation labels should be integers 0, 1, 2, ... mapped to variations in the order mentioned before e.g., *thin* and *thick* should be labelled as *0* and *1* respectively. Numbers after the last underscore will be IDs of images in that class. Images should be a **.jpg** files. For example, a sample $3^{rd}$ image of a long and thick red line, inclined at $135^o$ with the positive x-axis should be named **1_1_9_0_3.jpg** shown in Figure 1.
- Make a video of the images. Choose 90 images for each class and make a frame of 9 images $(3 \times 3)$. You will have 10 frames for each class. Make a video at **2 FPS**. The aim of the video will be to convince us whether all the variations have been captured. In order to do that keep the class of 3 variations constant and then vary the class of the 4th

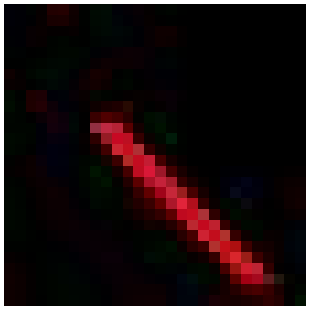

Figure 1: 1_1_9_0_3.jpg

variations. The order in which you choose the variations to change the class in the video is the same as depicted in the problem statement.

- Zip the code and dataset as **1.zip**.

# 2 Computational Graph Usage - Gravity Simulator (35 marks)

## 2.1 Motivation

This problem will introduce you to the working of computational graphs. You are expected to read and learn about how computational graphs work, how Tensorflow makes use of parallelism and GPUs, what kind of operations it provides for efficient computation, etc, by solving this problem.

## 2.2 Problem Statement

Given masses and initial positions & velocities of 100 particles, you are required to find the final positions and velocities of particles when the minimum distance between any pair of particles falls below a given threshold. The only working forces are Newtonian Gravitational Forces, in a two-dimensional rectangular coordinate system.

## 2.3 Input

Three *.npy* files named:

1. *masses* [Shape - (100, 1)]
2. *positions* [Shape - (100, 2)]
3. *velocities* [Shape - (100, 2)]

## 2.4 Requirements and Hints

- You are required to make a computational graph in **Tensorflow** for the update of positions and velocities at each time step. Also, write the graph summary in logs to have a visual representation in Tensorboard.
- Your program should **not** contain any loops except the loop containing your session run command.
- Read about *tf.constant*, *tf.placeholder*, *tf.reshape*, *tf.reduce_sum*, *numpy* broadcasting rules, *tf.matrix_set_diag*, and other Tensorflow operations that may be used in this problem.
- Compare the performance with a simple code without using computational graphs for the same task.
- (Bonus) Make a video of the simulation. (No extra marks)

## 2.5 Submission

- A *generic* (**without** hard-coded values) code as per requirements.
- Two *.npy* files named *positions* and *velocities* representing final positions and velocities of the particles.
- Zip the code and *.npy* files as **2.zip**.

## 2.6 Formulae

### 2.6.1 Force and acceleration

Gravitational force on any particle due to another particle is defined as,

$$\vec{F}_{12} = -G\frac{m_1 m_2}{|\ \vec{r}_{21}\ |^3}\vec{r}_{21} \tag{1}$$

where,
$\vec{F}_{12}$ is the Force exerted on particle 1 by particle 2,
G is the gravitational constant,
$m_1$ and $m_2$ are masses of the particles respectively and
$\vec{r}_{21}$ is the displacement vector from particle 2 to 1.

The total force on a particle is the vector sum of all forces acting on it i.e.

$$\vec{F}_{1_{total}} = -Gm_1\sum_{n=2}^{100}\frac{m_n}{|\ \vec{r}_{n1}\ |^3}\vec{r}_{n1} \tag{2}$$

The acceleration is therefore,

$$\vec{a}_{1_{total}} = -G \sum_{n=2}^{100} \frac{m_n}{|\ \vec{r}_{n1}\ |^3} \vec{r}_{n1} \tag{3}$$

according to Newton's second law of motion.

### 2.6.2 Update Equations

Updating positions and velocities can be done at infinitesimal time steps $\Delta t$ according to the following equations,

$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_t \Delta t + \frac{1}{2} \vec{a}_t \Delta t^2 \tag{4}$$

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_t \Delta t \tag{5}$$

where,
$\vec{x}_{t+\Delta t}$ and $\vec{v}_{t+\Delta t}$ is the position and velocity of a given particle at time step t + $\Delta t$,
$\vec{x}_t$ and $\vec{v}_t$ is the position and velocity of that particle at time step t and
$\vec{a}_t$ is the acceleration of that particle at time step t.

## 2.7 Assumptions

- Assume G = $6.67 \times 10^5$ units instead of the real G.
- Threshold distance = $0.1$ units.
- Time Step ($\Delta t$) = $10^{-4}$ units.

# 3 Layer API - A Simple Neural Network (35 marks)

## 3.1 Motivation

As most of this course will be wrapped around working with neural networks, a detailed understanding of the existing low-level libraries used for these tasks is incumbent. By solving this problem you will become familiar with how layer APIs work in these libraries.

## 3.2 Problem Statement

You are required to make a simple fully connected network to classify the MNIST dataset **and** the dataset you made in part 1. The caveat here is that you can not use layer APIs already provided in your library of choice. You will need to code a **dense** layer API as exhaustive as possible using in-built operations for matrix multiplication, etc. Refer to the Tensorflow Dense Layer API, for reference.

### 3.3 Submission

- Code for the task.
- Models for both datasets.
- A 3-4 page report containing:
  - Learning curves (accuracy and loss).
  - F-scores.
  - Confusion matrices.
  - Variations tried.
  - Inferences.
- Zip the code and report as **3.zip**