# CMPE 232 - Term Project

# Computation Graph

**Group Name:** Hadi Ins

**Group Members:**

Arda Cem Özmen (115200046)

Onur Bozkaya (114200027)

Deniz Küçük (115200037)

**Abstract:**

In this project, we mentioned about how forward/backward propagation works and we showed how to compute and draw graphs.

**Introduction:**

A computational graph is a way to represent a math function in the language of graph theory. Recall the premise of graph theory: nodes are connected by edges, and everything in the graph is either a node or an edge. In a computational graph nodes are either input values or functions for combining values. Edges receive their weights as the data flows through the graph. Outbound edges from an input node are weighted with that input value; outbound nodes from a function node are weighted by combining the weights of the inbound edges using the specified function.

**Data Structures and Libraries:**

This section describes reasons of used structures and libraries:

- For Loops for iterate over given objects.
- Array structures for store Topologic orders and graph variables.
- If-else blocks for avoid unnecessary calculations
- Networkx library for initializing graphs and getting Topologic orders

**Operations:**

Add() -> it takes two inputs and addition their values.

Multi() -> it takes two str inputs and multiply their values.

Sub() -> it takes two inputs and substract their values.

Divide() -> it takes two inputs and divide their values.

**Graph:**

Constant() -> Returns variables.

Calculation() -> In our calculation method, there are three nodes. Two of them are inputs and one of them is output. In our code x and y represents our inputs and z is output. The neig.append method makes nodes connections.

Draw() -> It makes Graph visual.

**Forward Propagation:**

It takes all inputs except the last one because our last node is our output and we don't need it. When user gives inputs, the algorithm runs them and makes the calculation with given operation.

**Backward Propagation:**

It is the inverse of Forward Propagation. The important point is, in forward propagation we don't care the last variable which is our output. In backward propagation our don't care node is the first one

**References:**


- http://www.deepideas.net/deep-learning-from-scratch-i-computational-graphs/
- https://medium.com/@d3lm/understand-tensorflow-by-mimicking-its-api-fromscratch-faa55787170d