# PROJECT TITLE:

TEXT VS TREES VS GRAPHS: DEEP LEARNING TECHNIQUES FOR PROGRAM UNDERSTANDING

# WHAT MY PROJECT WILL BE:

* I am going to build two new NN architectures; the Deep Graph-Based Recurrent Neural Network (DGBRNN) and the Deep Tree-Based Recurrent Neural Network (DTBRNN).
* I am going to compare these two architectures to a normal text based RNN, a Deep Feed-Forward NN, the GGNN and the TBCNN (all of which I will also build) to determine which of these architectures is most suitable for classifying python programs.
* In all, I will build 6 different neural networks.
* This paper will be published.

# PROJECT OBJECTIVES/IMPLEMENTATION STEPS:

1. To build a piece of software that instantly converts programs into their ASTs.
   1. Testing this to ensure that it works.
2. To build a piece of software that instantly converts programs into directed graphs.
   1. Testing this to ensure that it works.
3. Compare testing using the same pieces of code:
   1. First run: Running the code with no comments.
   2. Second run: Running the code with descriptive comments.
   3. Third run: Running the code with twice as many files.
4. To build a Recurrent Neural Network that makes use of NLP techniques to understand the contents of python programs.
5. To build a Deep Feed-Forward Neural Network that makes use of NLP techniques to understand the contents of python programs.
6. To build a Tree-Based Convolutional Neural Network that makes use of Abstract Syntax Trees to understand the contents of python programs.
7. To build a Gated Graph Neural Network that makes use of Directed Graphs to understand the contents of python programs.
8. To build a new type of NN: The Deep Graph-Based RNN that makes use of a graph based NN to understand the contents of python programs.
9. To build another new type of NN: The Deep Tree-Based RNN that makes use of a tree based NN to understand the contents of python programs.
   1. To train these models on the same datasets (four different types of sorting algorithms).
   2. To test these models using the same datasets (a separate set of sorting algorithm implementations).
   3. To compare the results of testing with all six models to determine which is best for processing source code.
10. To write a comprehensive paper presenting these new ideas and possibly (hopefully) submitting it to a journal as my first serious paper (laying the foundation for my PhD).

# POSSIBLE FUTURE WORK:

* Thinking about how the coding style affects the computer’s ability to understand the code: code written by the same person vs code written by different people.
* Thinking about how the naming of functions and methods affects .
* Building Neural Networks that process source code without needing to convert to ASTs or Graphs/Flowcharts.
* Cross Language understanding: Training in one language and testing in another (both will have to be similar in syntax e.g., java and C)
* Build a NN that understands a program and then points out any syntax errors in the program
* Build a model that takes a program and generates a description of what the program is doing

# POSSIBLE SHORTCOMINGS OF THE MODEL:

* The naming of methods in common programs such as sorting algorithms could be helping the computer with the task of classifying the programs