Queen Mary University of London

Final Year Project Specification

Session 2017/2018

Cryptographically Secure Pseudorandom Number Generation using Generative Adversarial Networks

Student
Marcello De Bernardi

Supervisor
Dr. Arman Khouzani

Student email
m.e.debernardi@se15.qmul.ac.uk

Student phone number
07492 524132

March 5, 2018

qmul.png

Contents

1 Project aims

The aim of the project is to explore how effectively generative adversarial networks (GAN) can be used to implement a cryptographically secure pseudo-random number generator (CSPRNG), by developing and training a GAN model to generate pseudo-random numbers and evaluating its performance.

Pseudo-random numbers are essential to several applications in cryptography as an alternative to "truly" random numbers [?], which are not always practically obtainable. GANs are a relatively new type of generative model, and while so far they have mostly been applied to the generation of realistic images, adversarial training of neural networks might be a practical approach to a broader class of machine learning problems [?][?]. Some early applications of GANs to the field of security have recently been demonstrated [?].

Little literature currently exists on the use of neural networks to implement PRNGs. Though some papers have shown that recurrent neural networks (RNNs) can be used as a PRNG [?][?], no existing research was found that relies on using adversarial training for the networks. Thus the project explores a novel security-related application for GANs, and seeks to determine, as a proof of concept, whether it is a viable one.

2 Methodology

The project will employ an experimental methodology, consisting primarily of the development and training of a generative adversarial network to be quantitatively evaluated.

The initial phase of the project entails a literature review of the current state of the art of CSPRNGs, as well as background reading in neural networks, generative adversarial networks, and the security applications of PRNGs more in general. Further learning will include familiarization with the Python programming language and useful machine learning libraries. This phase of the work aims to acquire the knowledge required to build GANs, as well as identify the requirements of a cryptographically secure PRNG.

Following the literature review, a simple prototype GAN will be produced, along with an evaluation mechanism. The prototype's primary purpose is to assess how promising different potential adversarial setups seem, as well as to compare different architectures for the individual neural networks. At this stage, the details of the evaluation mechanism will also be finalized. Its core will be based on how well the model passes the so-called "next-bit test", an essential requirement for CSPRNGs. The results obtained from the prototype will influence the design of the final implementation, as well as form the basis on which the interim report and risk assessment will be written.

The prototype will be followed by the development of the final model, designed in accordance with the approaches that will have seemed most promising at the prototype stage. The neural networks will be larger and operate on more complex inputs, which means that the results they produce will be more significant, but also that the computational complexity of the model will be higher than that of the prototype.

Once the implementation is mostly finalized, the model will be trained on Queen Mary's

HPC cluster to generate result data. Small refinements to the software may be admissible at this stage, provided the degree of risk is manageable. Concurrently, writing will begin on the parts of the project report that are not directly dependent on the test results, and early drafts will frequently be discussed with the supervisor. The report will then be finalized to include the test results, and a conclusion on the viability of GANs as a CSPRNG implementation will be drawn.

3 Project Milestones

The main project milestones are outlined below. Not all are official deliverables.

- 30 October 2017, Project Specification (deliverable document)
- 01 December 2017, Prototype Implementation (non-deliverable software)
- 08 December 2017, Interim Report and Risk Assessment (deliverable document)
- 15 January 2018, Complete Implementation (deliverable software)
- 05 February 2018, Experimental Results (deliverable data)
- 20 February 2018, First Draft of Report (non-deliverable document)
- 19 March 2018, Second Draft of Report (deliverable document)
- 14 April 2018, Final Draft of Report (non-deliverable document)
- 23 April 2018, Report and Supporting Material (deliverable document)
- 25 April 2018, Presentation Slides and Destination Form (deliverable document)
- 30 April 2018, Project Vivas Start

4 Required knowledge, skills, tools, and resources

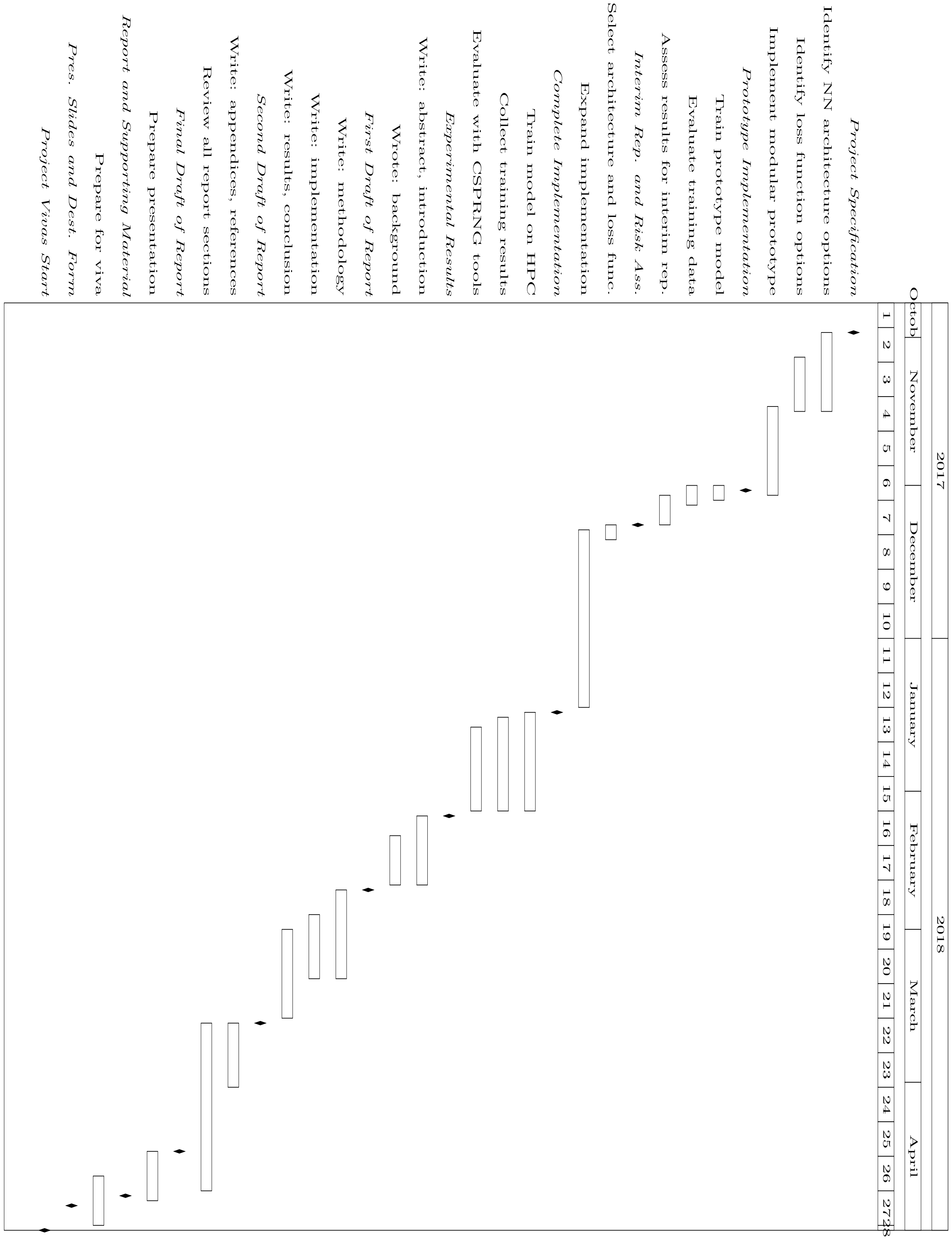
At a high level, the project requires knowledge in the areas of machine learning and pseudo-random number generation. In particular, it requires an understanding of neural networks and their different architectures, loss functions, recurrent neural networks, generative adversarial networks, random variables and probability distributions, pseudo-random number generation algorithms, and the security requirements of CSPRNGs.

The core practical skills required are Python programming (particularly in procedural and object-oriented styles), designing a neural network architecture appropriate for the task to be learned, implementing the architecture using machine learning libraries, designing the system so that it may run in a distributed computing environment such as a server cluster, and correctly collecting and analyzing the obtained data.

A variety of free and proprietary development tools will be used. The implementation will be written in Python, and make heavy use of TensorFlow, an open-source distributed-computing software library for machine learning. TensorFlow provides a supporting tool, TensorBoard, which allows graphically visualizing neural networks developed in TF. Most of the development will be done in Atom, a modular and highly customizable text editor. One of the primary appeals of Atom lies in a package called Hydrogen, which allows the user to interactively run snippets of code while inspecting variables or generating visual plots. This is similar to the popular Jupyter Notebook.

The software to be developed is expected to be shorter than 500 lines of code. Hence no significant resources will be required during the development phase. Training the final model will not be practically feasible on a commodity laptop due to its computational complexity, so access to the Queen Mary University of London HPC cluster will be required.

5 Timeplan


2017
2018
Octob
November
December
January
February
March
April
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
Project Specification
Identify NN architecture options
Identify loss function options
Implement modular prototype
Prototype Implementation
Train prototype model
Evaluate training data
Assess results for interim rep.
Interim Rep. and Risk Ass.
Select architecture and loss func.
Expand implementation
Complete Implementation
Train model on HPC
Collect training results
Evaluate with CSPRNG tools
Experimental Results
Write: abstract, introduction
Wrote: background
First Draft of Report
Write: methodology
Write: implementation
Write: results, conclusion
Second Draft of Report
Write: appendices, references
Review all report sections
Final Draft of Report
Prepare presentation
Report and Supporting Material
Prepare for viva
Pres. Slides and Dest. Form
Project Vivas Start