NeuralBasedPRNG

Разработка JAVA приложения на Android Studio

Когерентные PRNG с нейронными сетями

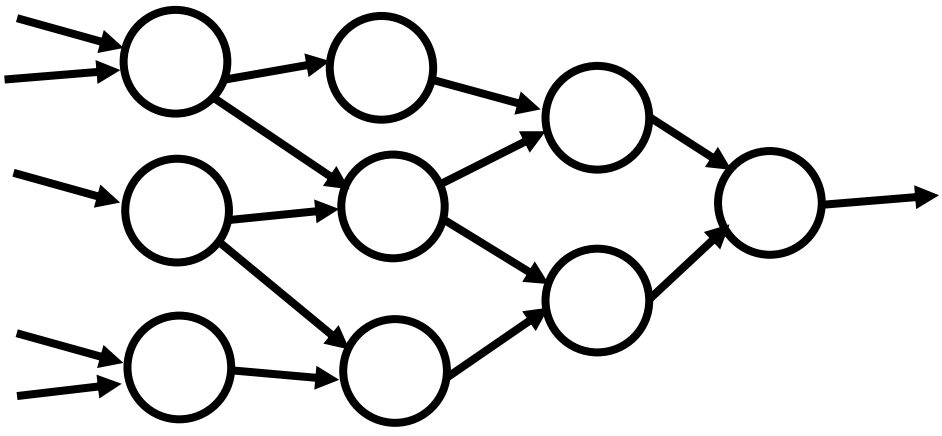
*Когерентный – несколько экземпляров PRNG независимо генерируют одинаковые массивы псевдослучайных чисел при одинаковых стартовых значениях (при одинаковом мастер-ключе) без какого-либо взаимодействия.

** Мастер-ключ здесь имеет значение стартовых условий работы для нейронной сети.

При этом такие генераторы могут не иметь постоянного алгоритма генерации (он может меняться на любом шаге) или постоянной внутренней структуры и могут принципиально никогда не повторяться (есть возможность исключения повторов).

Мы будем исходить из предположения, что любой программный генератор «случайных» чисел рано или поздно будет повторятся. Поэтому мы будем перестраивать нейронную сеть (не только вес и смещение каждого нейрона, но структуру нейронной сети – количество слоев, нейронов в слоях, структуру связей) для максимальной изменчивости алгоритма генерирования «псевдочисел».

*** Здесь «псевдочисло» - значение, используемые для вычисления одного бита в генерируемом псевдослучайном числе. Как пример, на одном шаге работы NeuroPRNG с каждого нейрона получается одно «псевдочисло», используемое для дальнейшего вычисления одного бита (бит не используется напрямую в генерируемом числе!).



Выбор возможных вариантов NeuroPRNG (NPRNG) огромен.

Например. Часть мастер-ключа используется для вычисления количества нейронов, слоев и для распределения нейронов по слоям в создаваемой нейронной сети. Другая часть мастер-ключа определяет количество связей и их распределение по нейронам. Третья часть мастер-ключа задает веса и смещения.

Выполняется несколько (десятков, сотен, тысяч…) циклов обучения на заданном датасете (не публичном).

После этого такой «нейронный модуль» можно использовать в PRNG.

Neural PRNG

Простейший пример. Выходные значения каждого нейрона (Big Int) используются для поиска ближайшей пары простых чисел для алгоритма Блюм-Блюм-Шуба (не крипто-стойкого).

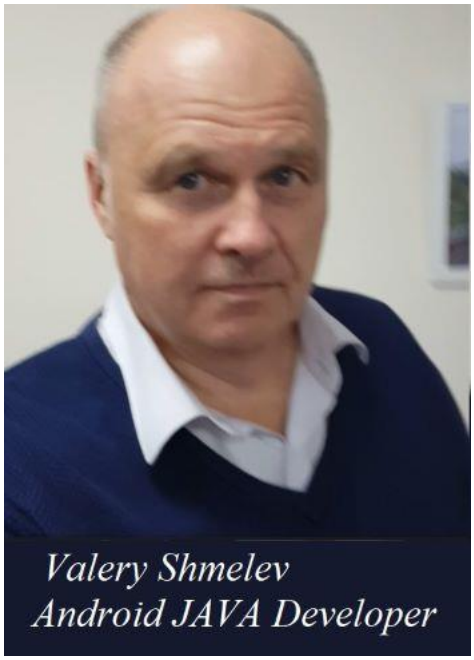
Из полученного результата используется бит четности. Затем выполняется следующий цикл работы. На вход подаются вычисленные и не использованные данные с нейронов. Получается еще один бит генерируемого псевдослучайного числа. И так далее, до формирования Big Int «случайного» числа нужной размерности.

Очень интересно. И это пока еще без использования переобучения генератора и без реконфигурирования нейронной сети.

Такой PRNG будет медленным, не для поточного шифрования (если не рассматривать масштабируемые NPRNG на одном нейроне или слое, реализованном на отдельном чипе). Рассматриваемые здесь генераторы псевдослучайных чисел интересны прежде всего с точки зрения противодействия криптоанализу. На каждые несколько псевдослучайных чисел будет создаваться свой алгоритм генерирования. Здесь огромный потенциал.

Далее, попробуем построить простейший «нейронный» PRNG без обучения. Сначала это будет простая нейронная сеть из 6 нейронов. С слоями из 3,2,1 нейронов. Затем добавим методы для получения «псевдочисел», методы для извлечения одного бита и накопления выходного псевдослучайного числа. Затем сделаем эту нейронную часть NPRNG **динамически перестраиваемой** (с изменяемыми параметрами, связями и структурой). Добавим слой и связи.

Читаем далее…



PS. Эти проекты – отличная база для собственных разработок, проверки идей и тренировки.

Один из «динамических» проектов (последовательная серия JAVA проекта в развитии) – **SimpleNNeuron** нейронная сеть на Android JAVA. Фактически – как написать простую нейронную сеть и обучить.

http://multidoc.oflameron.com/page004.htm

http://site.oflameron.ru/page005.htm

http://webpage.pips.ru/page0006.htm