NeuralBasedPRNG

Разработка JAVA приложения на Android Studio

Когерентные PRNG с нейронными сетями

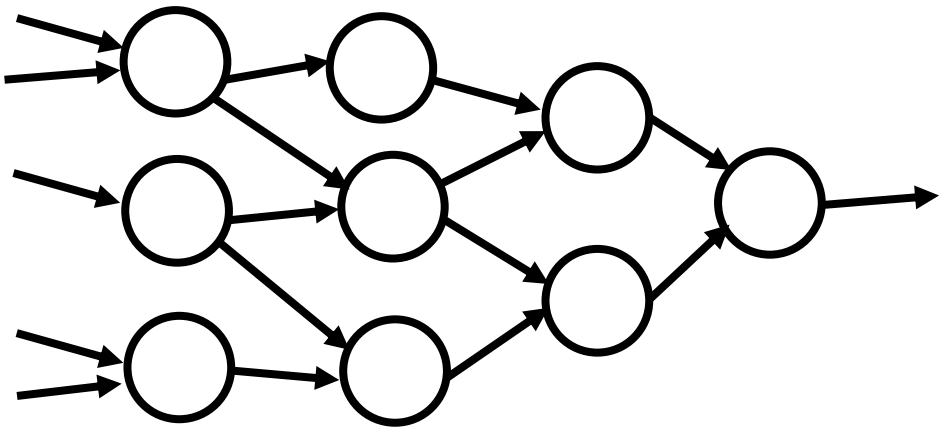
*Когерентный – несколько экземпляров PRNG независимо генерируют одинаковые массивы псевдослучайных чисел при одинаковых стартовых значениях (при одинаковом мастер-ключе) без какого-либо взаимодействия.

** 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.

Пример взаимодействия 6 нейронов в трёх слоях (известный в Internet пример). Здесь только добавлены номера нейронов

```
return neurons.get(5).compute(
    neurons.get(4).compute(
        neurons.get(2).compute(input1, input2, 2),
        neurons.get(1).compute(input1, input2, 1),4
    ),
    neurons.get(3).compute(
        neurons.get(1).compute(input1, input2,1),
        neurons.get(0).compute(input1, input2,0),3
    ),5
);
```

В таком виде перестраивать их количество и схему взаимодействия не удобно.

Поэтому мы перепишем сеть иначе. Реально будет использоваться один нейрон, который будет эмулировать все слои и нейроны сети. В нескольких массивах будем запоминать входные и выходные значения для каждого нейрона, веса, смешения и, в дальнейшем, тип использованной функции.

Такой вариант нейронной сети будет работать медленнее, но мы сможем реализовывать любые конфигурации динамически.

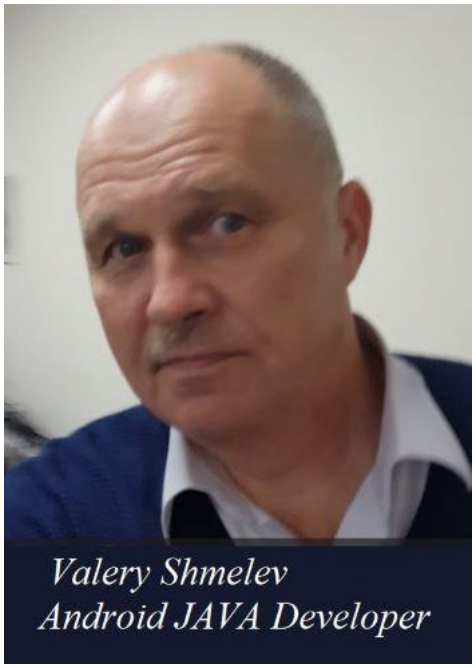
Neural PRNG

Все это будем тестировать на прототипе нейронной крипто сети без обучения.

Даже в этом случае мы можем получать псевдослучайные числа (каждое число), вычисленные по совершенно разным формулам и с каждый раз новым периодом повторения. Крипто-аналитикам не понравится. Можно добиться того, что даже крипто-анализ с помощью специальных нейронных сетей будет бесполезен. Причем, такой способ генерирования псевдослучайных чисел будет иметь низкий «порог входа». Не надо тщательно подбирать коэффициенты и знать нюансы вычисления для обеспечения крипто-стойкости.

Бесконечное число формул, структур сети и размеров генерируемых чисел.

Читаем далее…



PS. Эти проекты – отличная база для собственных тестов и разработок.

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

http://referat.oflameron.ru/page0001.htm

http://picture.oflameron.ru/