Can Neural Network Factorize Composite Numbers?

Ting Liang (tngliang@stanford.edu)

Prime Factorization Problem

- Reverse multiplication process is NP-hard
  - $N = PQ$, easy to multiply from P,Q, hard to recover from N
  - Information is lost during multiplication that is hard to recover
- Underpins the Security Guarantee of RSA Crypto
  - Euler Totient Function $\Phi(N) = (P-1)(Q-1)$ requires P,Q
- But we can generate infinite pairs of ((P, Q), N) easily
  - Can we use a Neural Network learn a better representation of N to directly or indirectly yield P or Q?
- Related Works
  - Quadratic Sieve and General Number Field Sieve
  - Quantum Computing: reversible qubits assignment ops

Problem Analysis (RSA-64)

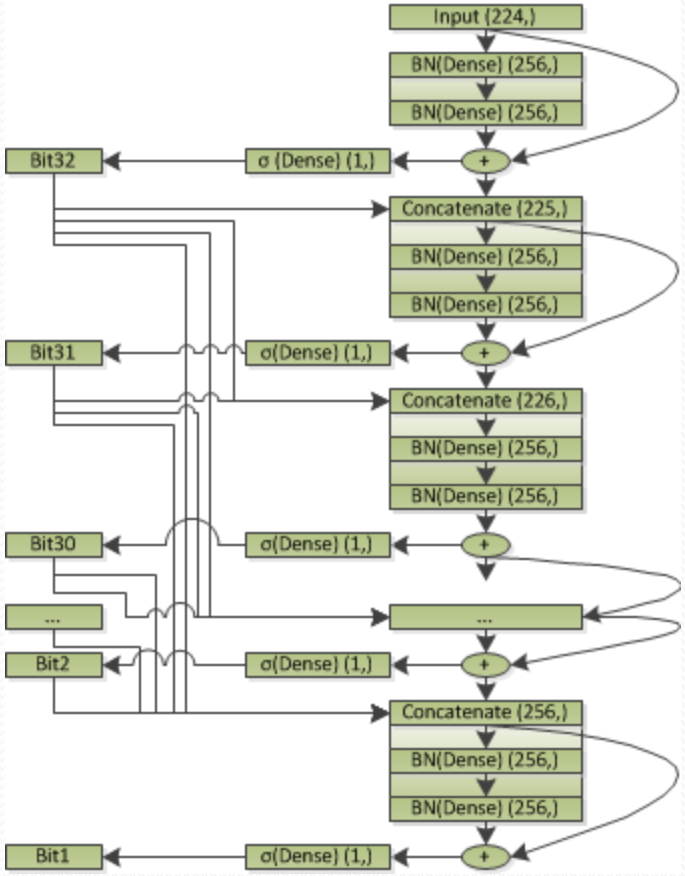
- Map N directly to P is a difficult Classification Problem
  - 4+ billion classes for 32 bit primes
  - More if we want to crack RSA-768 world record (2^{384} classes)
  - ImageNet classification involves 1,000 classes
  - NLP typically involves no more than 20,000 English Words
  - Softmax over 2^{32} classes would have precision trouble in FP32
- But what about hierarchical learning?
  - Divide domain $(2, 2^{32} - 1)$ into 2 segments $(2, 2^{31}), (2^{31}, 2^{32})$
  - Classify from N which segment P would lie within
  - With top bit prediction, divide and conquer the next bit.
  - Eg the last bit has distribution $P(x_1 | x_2, x_3, x_4, \ldots x_{32})$

Binary Sequence to Binary Sequence

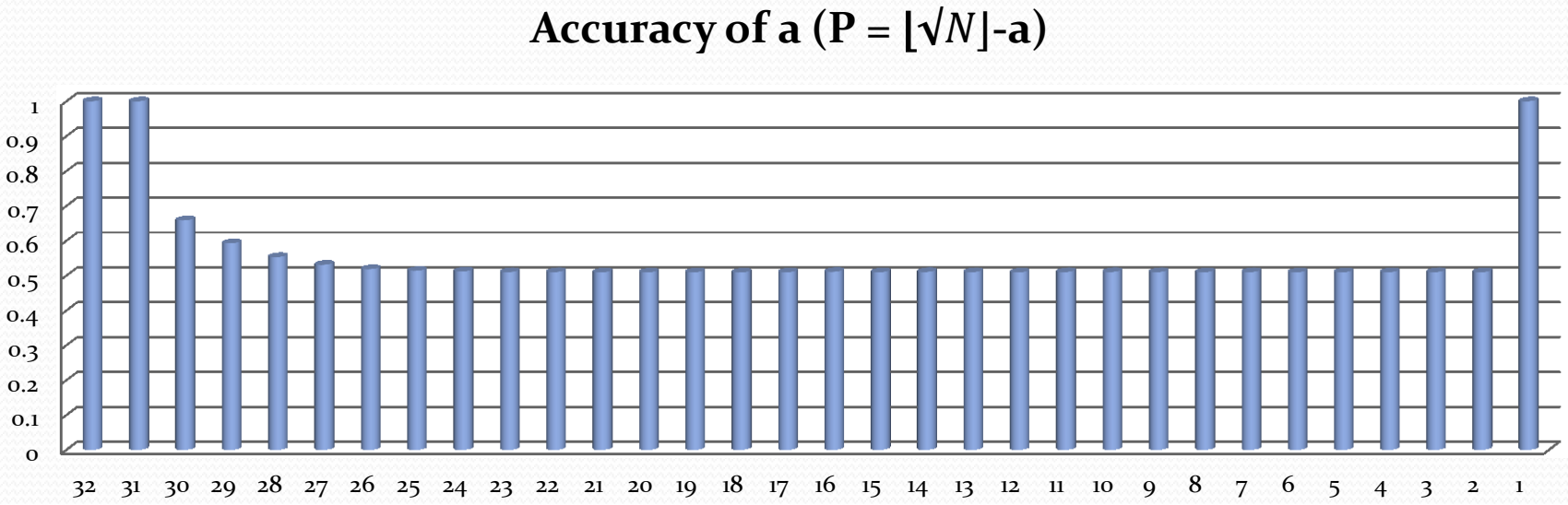
- Last equation is identical to NLP translation problem!
  - Given context words $x_2, x_3, x_4, \ldots x_{32}$, learn distribution for x_1
  - Vocabulary has 2 symbols only (0 and 1)
  - Sequence is limited to 32 "time" steps
  - Each step output a sigmoid instead of softmax
- Difficulty might be in higher valid sequence density
- Choices of Neural Network Architectures:
  - Deep feed-forward Neural Network with 32 sigmoid outputs
  - LSTM based RNN for seq2seq translation with fixed 32 time steps and attention network
  - Transformers (if much longer sequences are provided)

Deep Feed-Forward Neural Network

- 2 Fully connected dense layers per block
- Sigmoid Activations for output bits
- Higher output bits concatenated and fed into all subsequent blocks
- Number of layers within blocks can be adjusted indepepdently
- ResNet skip connection added to every 2 layers
- Uses BatchNormalization
- Number of layers in each block is configurable.
  - Initial Block Configuration used is [1,1,1,…1]
  - Experimented with configuration [0,0,32,32,32,32,4,4…,0] with same training result (vanishing gradients)



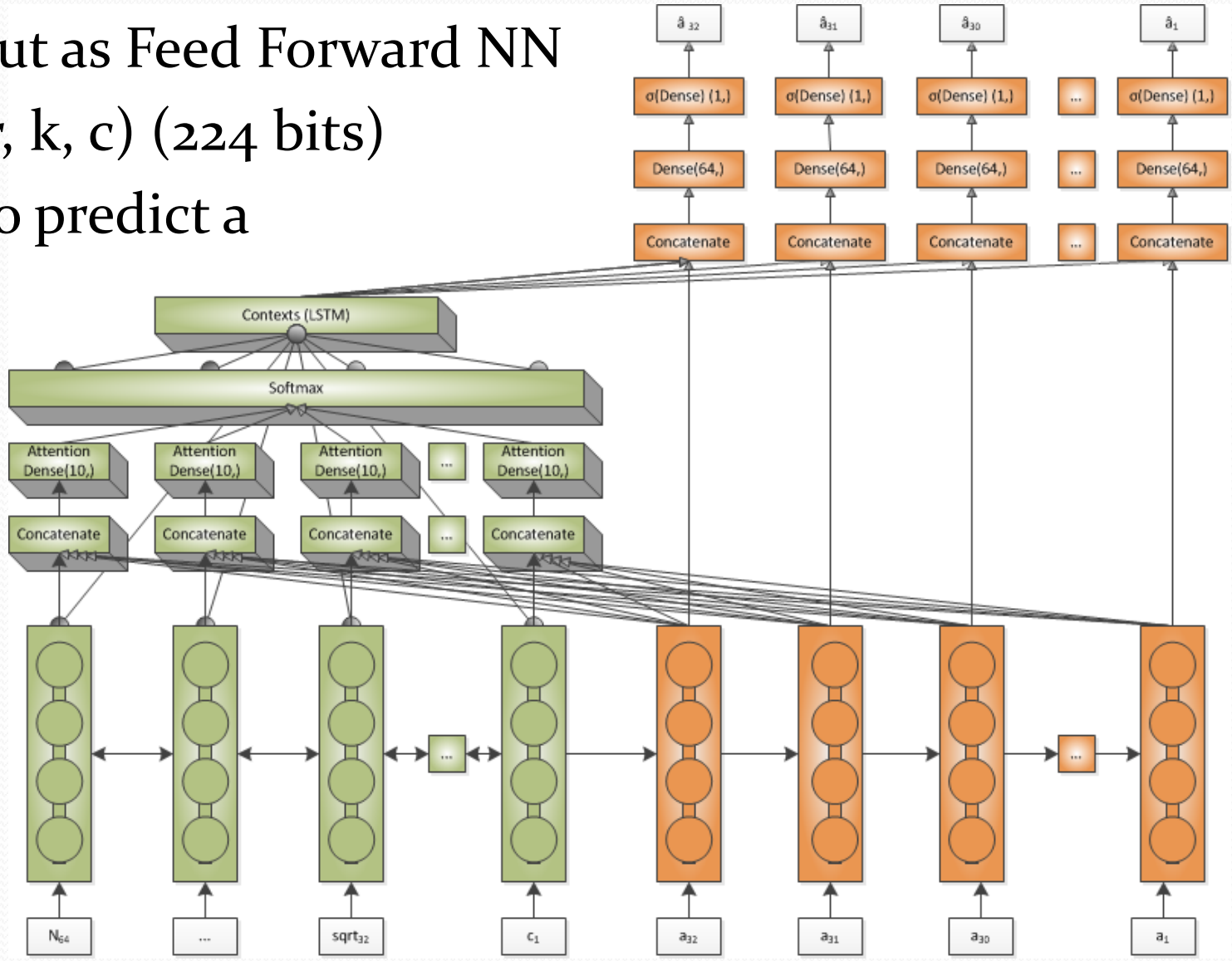
Feed Forward Neural Network Predict a



- Trained on 22,118,400 pairs of (N, (int(sqrt(N))-a)
- Validated on 2,457,600 pairs held out.
- Only the first 2 bits and last bit is predicted accurately
  - Primes are always odd (last bit)
  - Primes very close to $\lfloor\sqrt{N}\rfloor$ are not secure (easily guessed)
- Model is clearly under-fitting all intermediate bits
- Vanishing Gradient Problem remains. Should switch to DenseNet as next step.
- Validation perplexity for all bits = 1.86

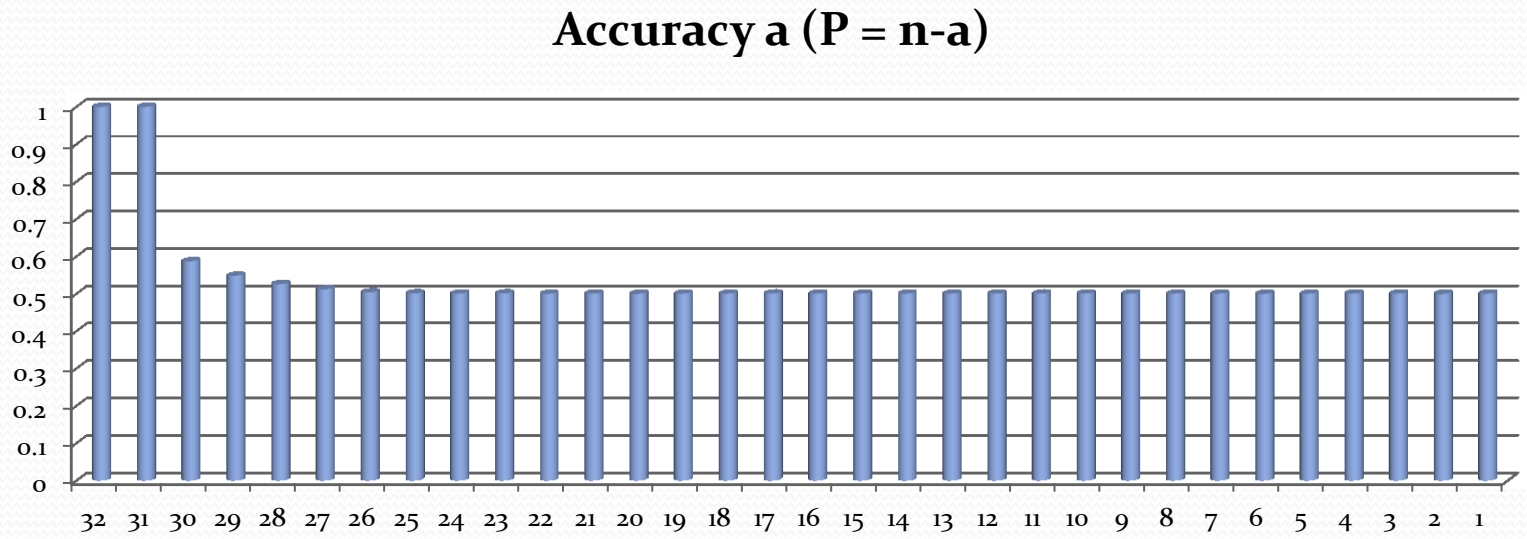
LSTM Neural Machine Factorization with Attention

- Same input as Feed Forward NN
- (N, sqrt, r, k, c) (224 bits)
- Trained to predict a



LSTM Neural Machine Predicts a

- 10 times slower to train
- Approximately same results except last bit



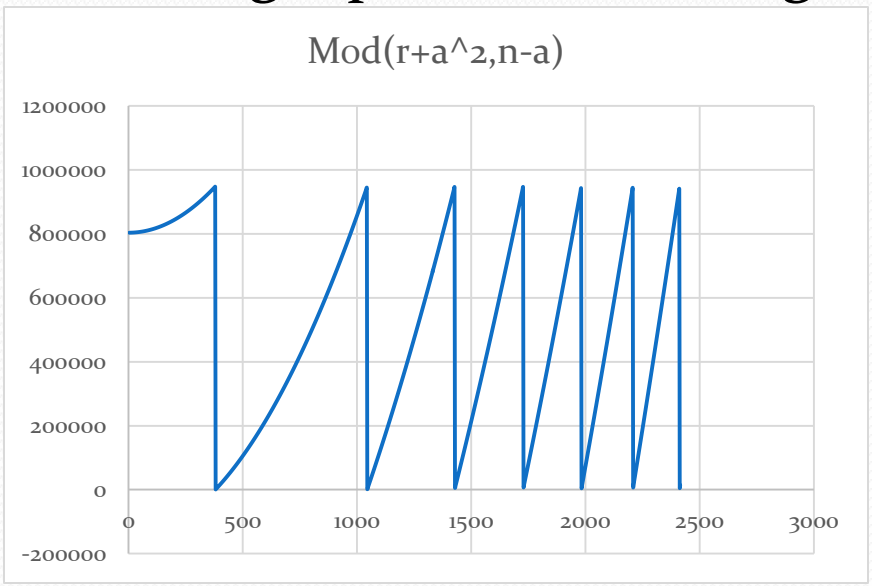
- Trained on 29,491,200 pairs of (N, (int(sqrt(N))-a)
- LSTM didn't determine the last bit accurately in this case.
- Require restrict primes to under 1,000,000 to obtain last bit accuracy.
- Perplexity for all bits = 1.91 which is worse than Feed Forward

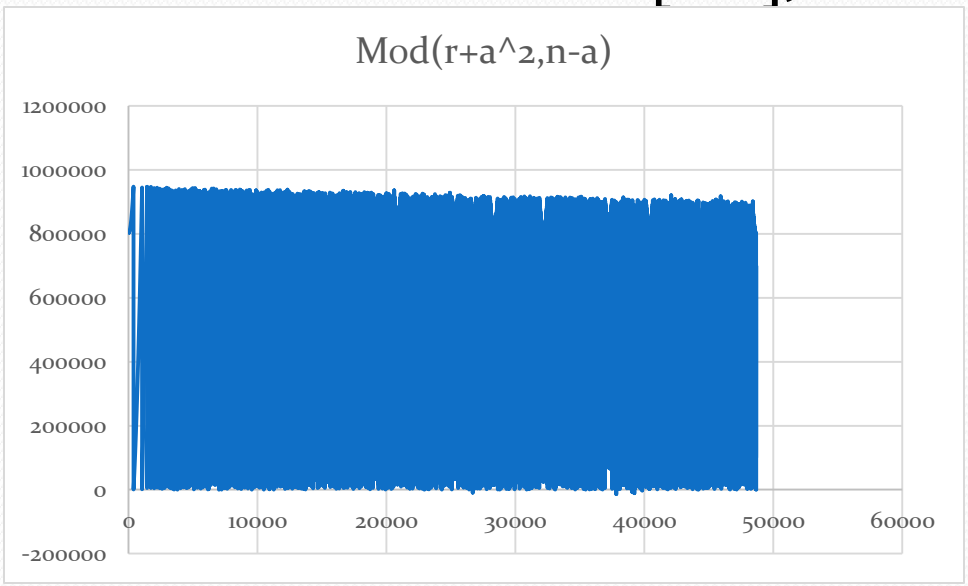
Feature Engineering

- Predicting a directly is hard: 18.93% of all $a \in (2, 2^{32})$ are primes
- If we can engineer some features to help NN, eg
  - $N = P \times Q = \lfloor\sqrt{N}\rfloor^2 + r = (\lfloor\sqrt{N}\rfloor - a)(\lfloor\sqrt{N}\rfloor + b) = N - r + (b-a)\lfloor\sqrt{N}\rfloor - ab = N - r + (b-a)\lfloor\sqrt{N}\rfloor - a(b-a) - a^2 = N - r + (b-a)(\lfloor\sqrt{N}\rfloor - a) - a^2$
- Eliminate N from both side and rearrange, then we get
  - $(b-a)(\lfloor\sqrt{N}\rfloor - a) = r + a^2$
- What does that mean?
  - $P = \lfloor\sqrt{N}\rfloor - a$ can also be obtained from GCD($r + a^2$,N)
  - Knowing $b - a$ is also sufficient to calculate a and therefore P
  - Suppose we define $r = k\lfloor\sqrt{N}\rfloor + c$
  - We can solve quadratic equation $k\lfloor\sqrt{N}\rfloor + c + a^2 = (b-a)(\lfloor\sqrt{N}\rfloor - a)$
    - $a = \frac{-(b-a) \pm \sqrt{(b-a)^2 - 4c + 4\lfloor\sqrt{N}\rfloor(b-a-k)}}{2}$
- Experimented to predict b-a (64 bits) instead of a (32 bits).
  - Resulted 2% increase when the output sequence is longer (64 bits).
  - So the longer sequences of output actually increased bit accuracy
    - Reason is 64 bit outputs have less valid values density than 18.93%

Understanding why Factoring is Hard

- If we plot a graph of (N= 899991699881)
  - $y = r + a^2 = (b - a)(\lfloor\sqrt{N}\rfloor - a)\ \left(mod\ (\lfloor\sqrt{N}\rfloor - a)\right)$
- We get parabola starting from c (where $c = r\ mod\ \lfloor\sqrt{N}\rfloor$)

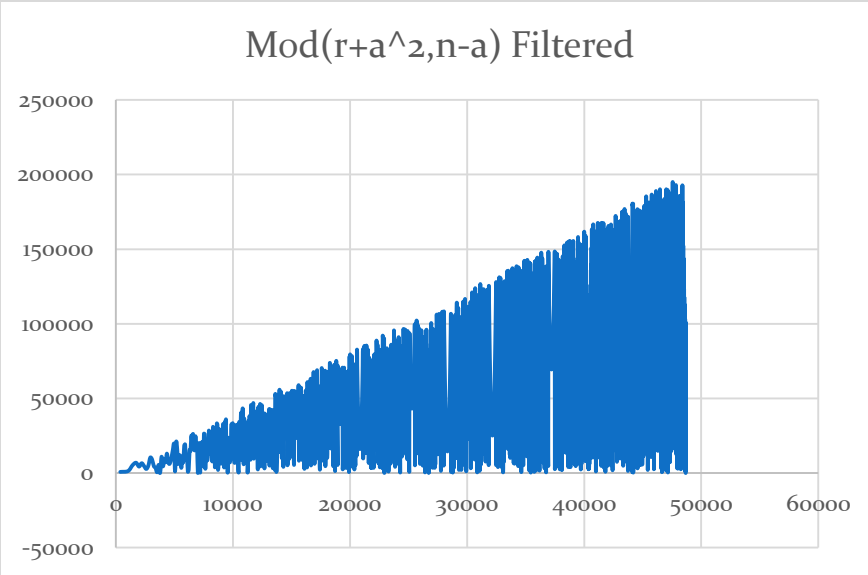


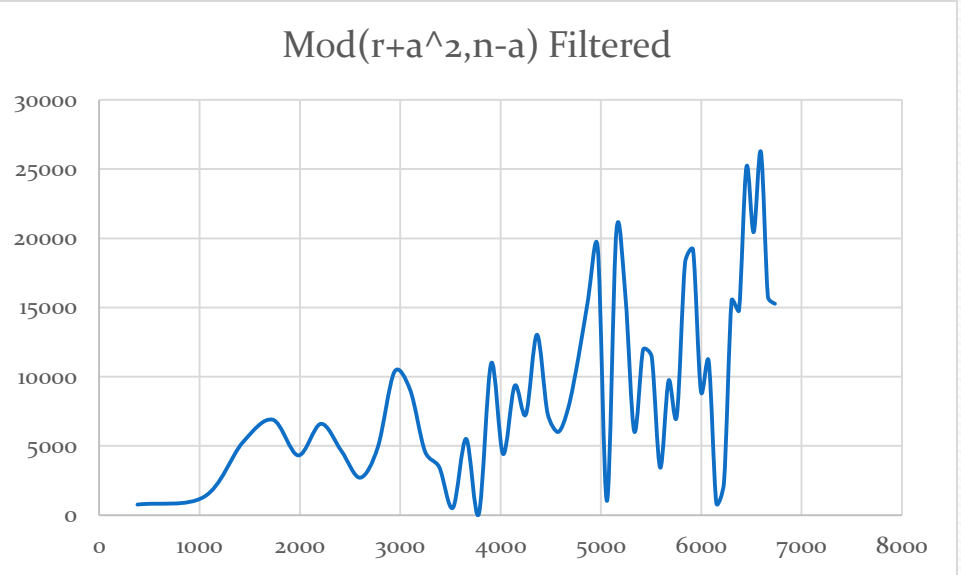


- As we can see from graph, chaos quickly reins as a ↑

When Parabola hits 0 exactly at Integer a!

- Every time the parabola wraps around after hitting the bound at $(\lfloor\sqrt{N}\rfloor - a)$ the residual hovers above 0 and increases noisily
- We can observe the residual value at next integer a immediately after wrapping around. If the residual=0 we found our desired a.





- A slight speed up can be achieved by calculate wrap arounds directly using Newton-Ralphson approximation (gradients = 2a)

Conclusion & Future Works

- Sparsity of output domain is key to success.
- Need much larger NN to address chronic under-fits of
  - Especially of middle bits.
- **NN at present compute cannot effectively factor or crack RSA, so internet remains secure.**
- Way to generate long sequences: and use Transformers!
  - Not sequential as RNN, more like Feed Forward NN (Faster)
  - Multiple Attention Layers to support multiple output bits.
  - Known to find better patterns in Long Sequences
  - Very Long Sequences can be generated from Residual Curve
    - Each N can generate millions of overlapping data points within $(2, \lfloor\sqrt{N}\rfloor - a)$
    - Better generalization if unknown P shares the same $\lfloor\sqrt{N}\rfloor$ with training data.
  - eg: train on 10,000 intermediate steps from N to a evenly divided, each step will keep N, $\lfloor\sqrt{N}\rfloor$, r the same but vary k and c, until c = 0.