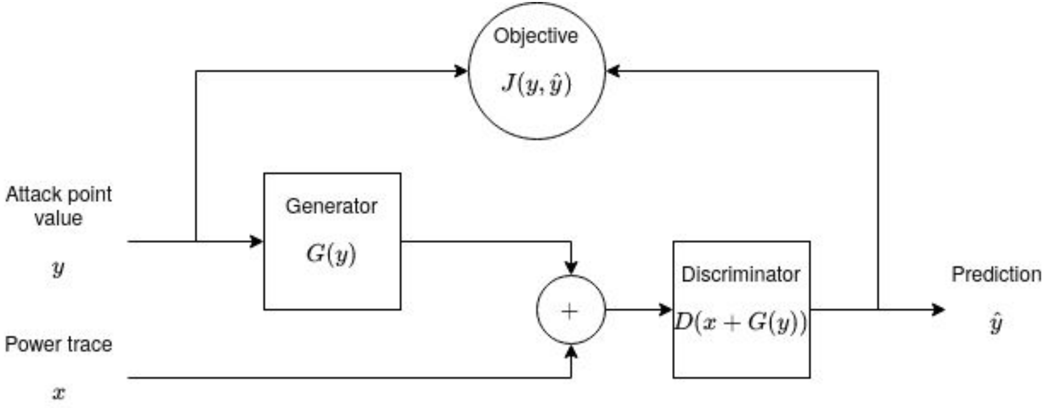
Experiment overview

1) Compile system with generator and discriminator
    a) Variety of generator architectures
    b) Pretrained discriminator from Google codebase
2) Train discriminator to minimize objective J
3) Train generator to maximize J
4) Again train discriminator to minimize J



Takeaways

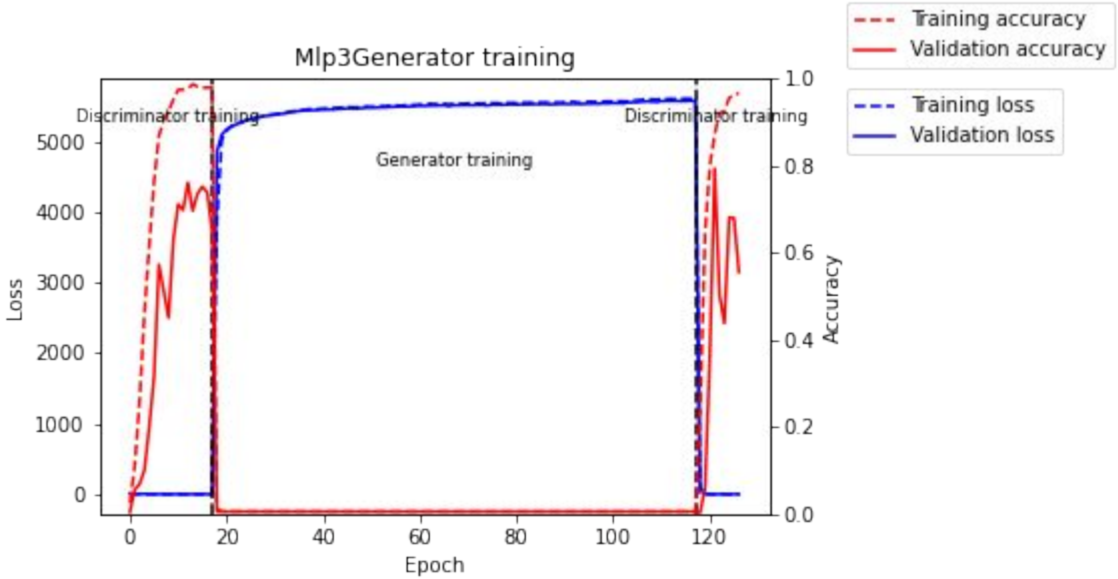
- Easy to disrupt performance of static discriminator
    - Constant offset to trace is sufficient
    - Tends to collapse towards always predicting 1 or few keys
- Re-trained discriminator performance is higher than without discriminator
    - Generator ‘gives away’ the key
    - Need to re-think generator input, objective
- Basic key -> trace generator has lots of parameters, not very general
    - Could consider recurrent generator, generator controlling sinusoid characteristics
- Currently unclear which generator is best
    - Need to change setup until generator is no longer giving away key

Next steps

- Consider different objective functions
  - Minimize all discriminator outputs, not just for correct key
  - Minimize discriminator gradient magnitude
- Consider different generator inputs
  - Random input instead of correct attack point
  - In practice, only 1 key per generator – could try inputting plaintext instead.
    - Likely solves problem of giving away key
    - Need to find a different dataset
- Try to find better generator architectures
  - Convolutional architectures typical for GANs
  - Sum of sinusoids
  - Recurrent

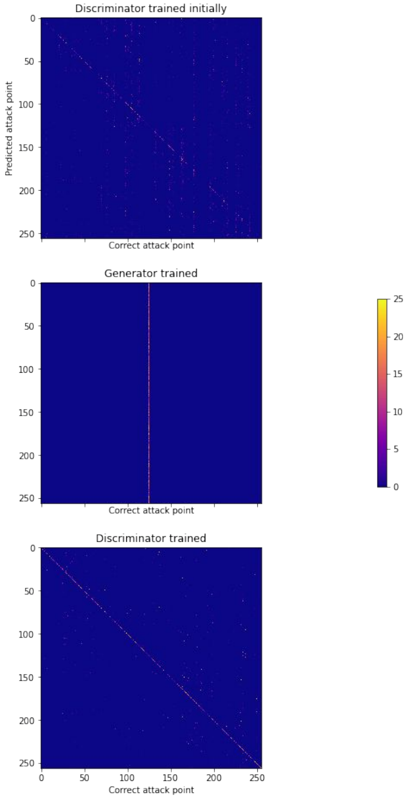
Measurements taken (3-layer MLP generator)

- Loss/accuracy during training



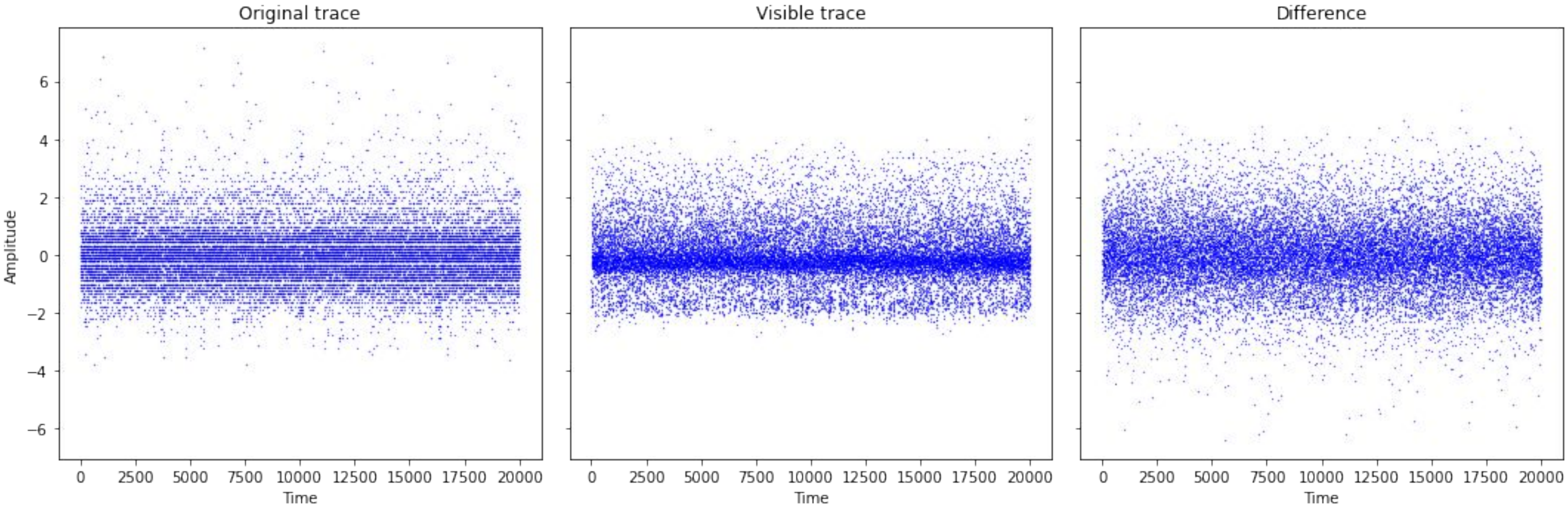
Measurements taken (3-layer MLP generator)

- Confusion matrix after 3 training phases



Measurements taken (3-layer MLP generator)

- Traces before/after application of generator


Original trace
Visible trace
Difference
Amplitude
Time