The Devil is in the GAN

Defending Deep Generative Models Against Adversarial Attacks

Ambrish Rawat @iambrishing

Killian Levacher @killianlevacher

Mathieu Sinn @SinnMathieu

IBM **Research** Europe

AI4media
ARTIFICIAL INTELLIGENCE FOR THE MEDIA AND SOCIETY

This Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951911

The Devil is in the GAN

Defending Deep Generative Models Against Adversarial Attacks

Available on **arxiv**

github.com/IBM/devil-in-GAN

Deep Generative Models can synthesize **samples in high-dimensional** data manifolds from arbitrary latent representations

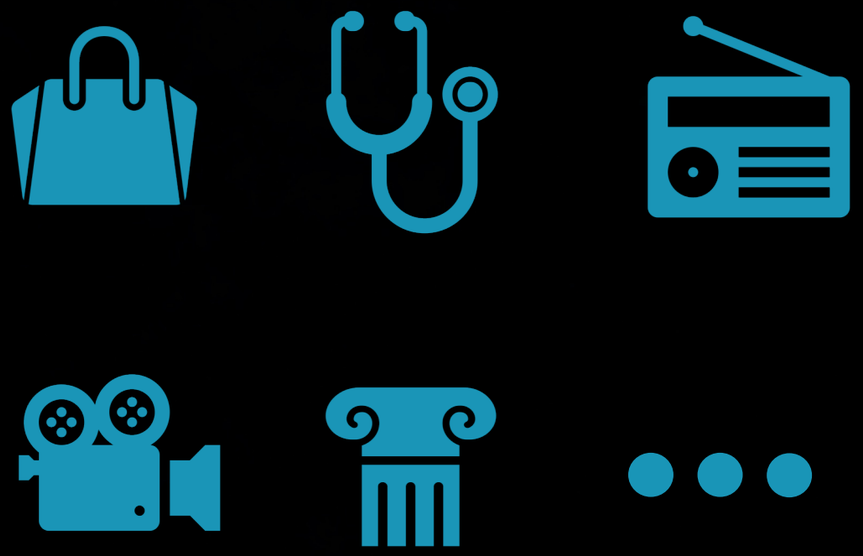
These are NOT real people

StyleGan: https://github.com/NVlabs/stylegan

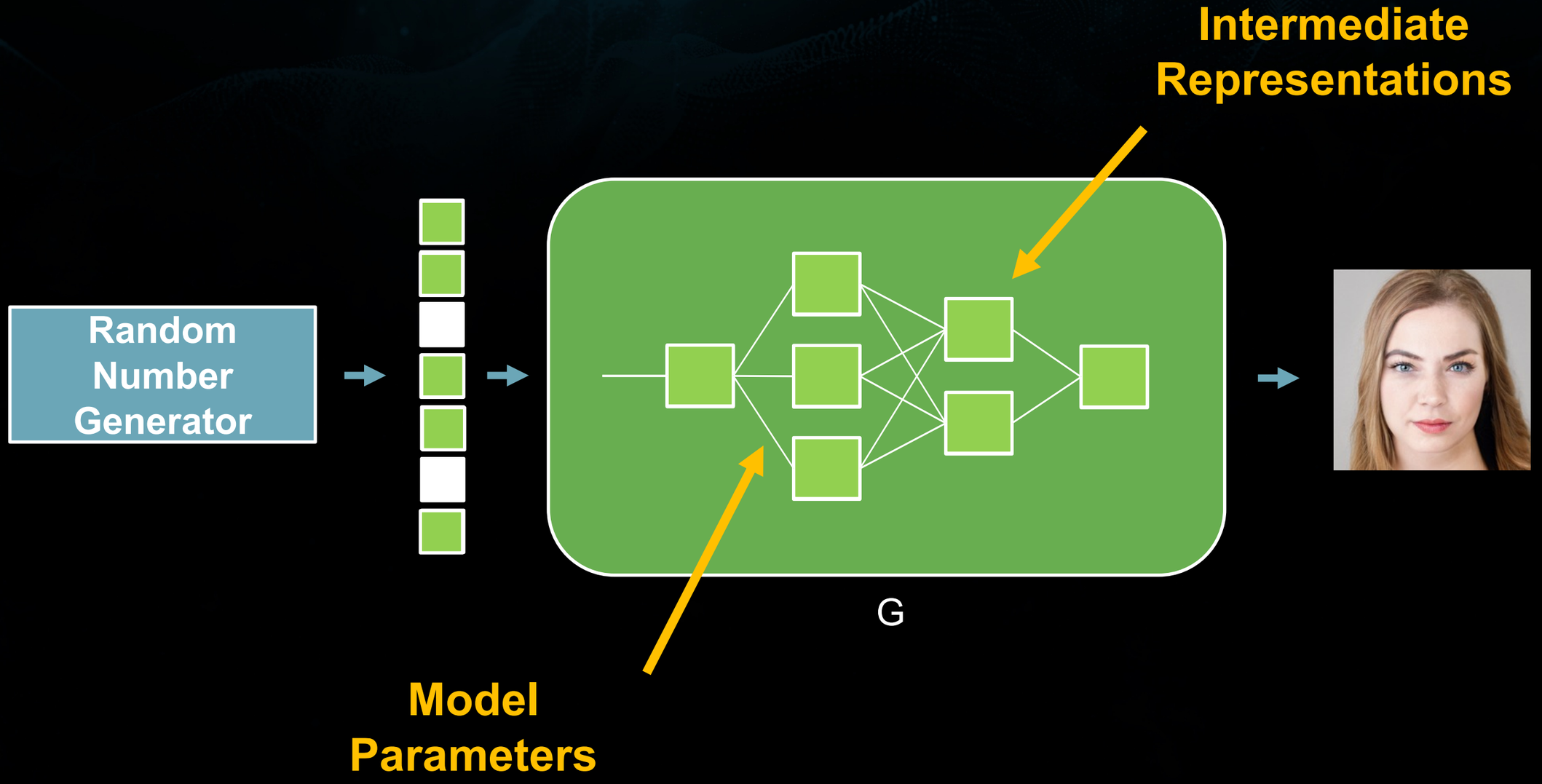
Deep Generative Models can synthesize **samples in high-dimensional** data manifolds from arbitrary latent representations

Wide Variety of Modalities

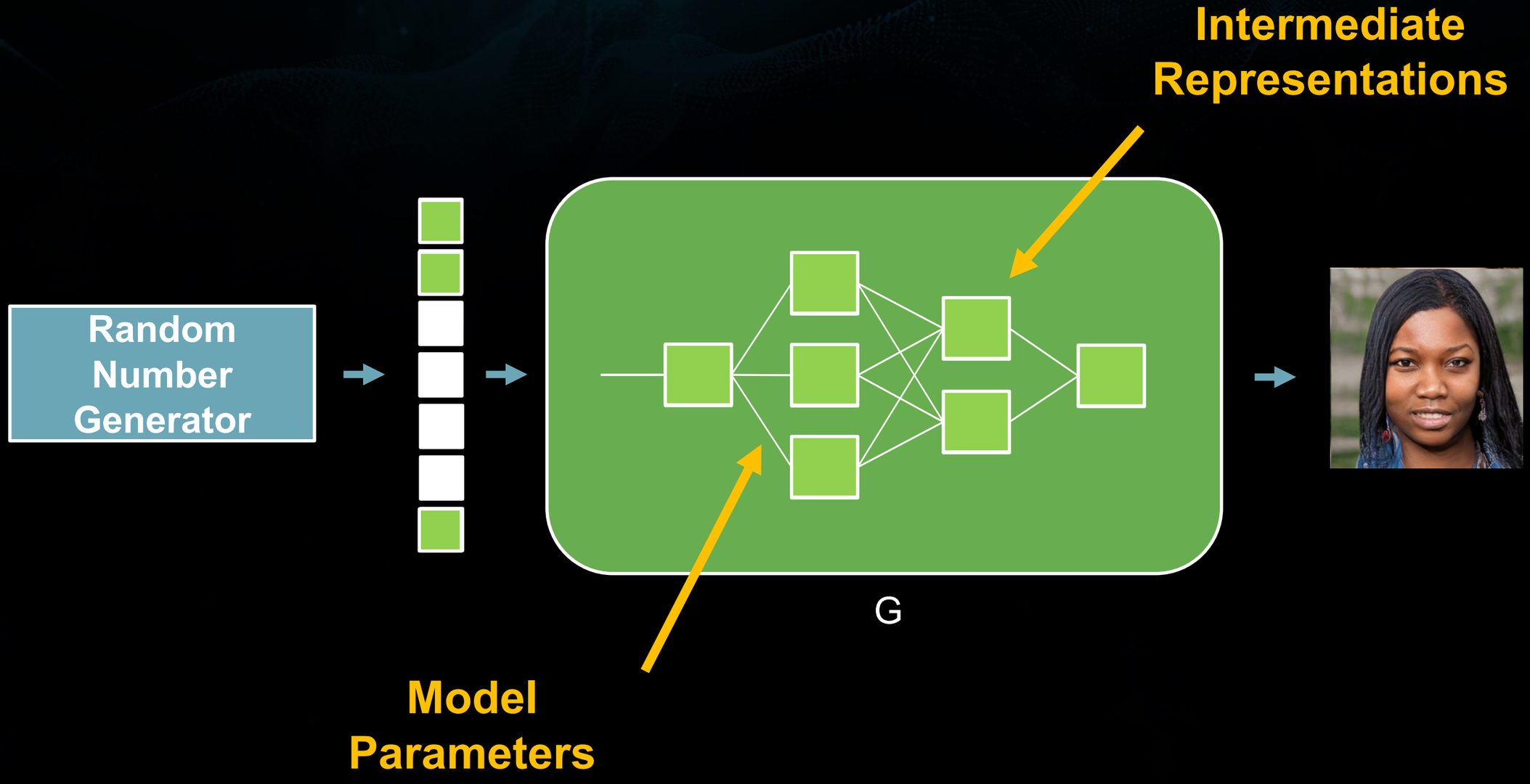
Wide Range of Industries

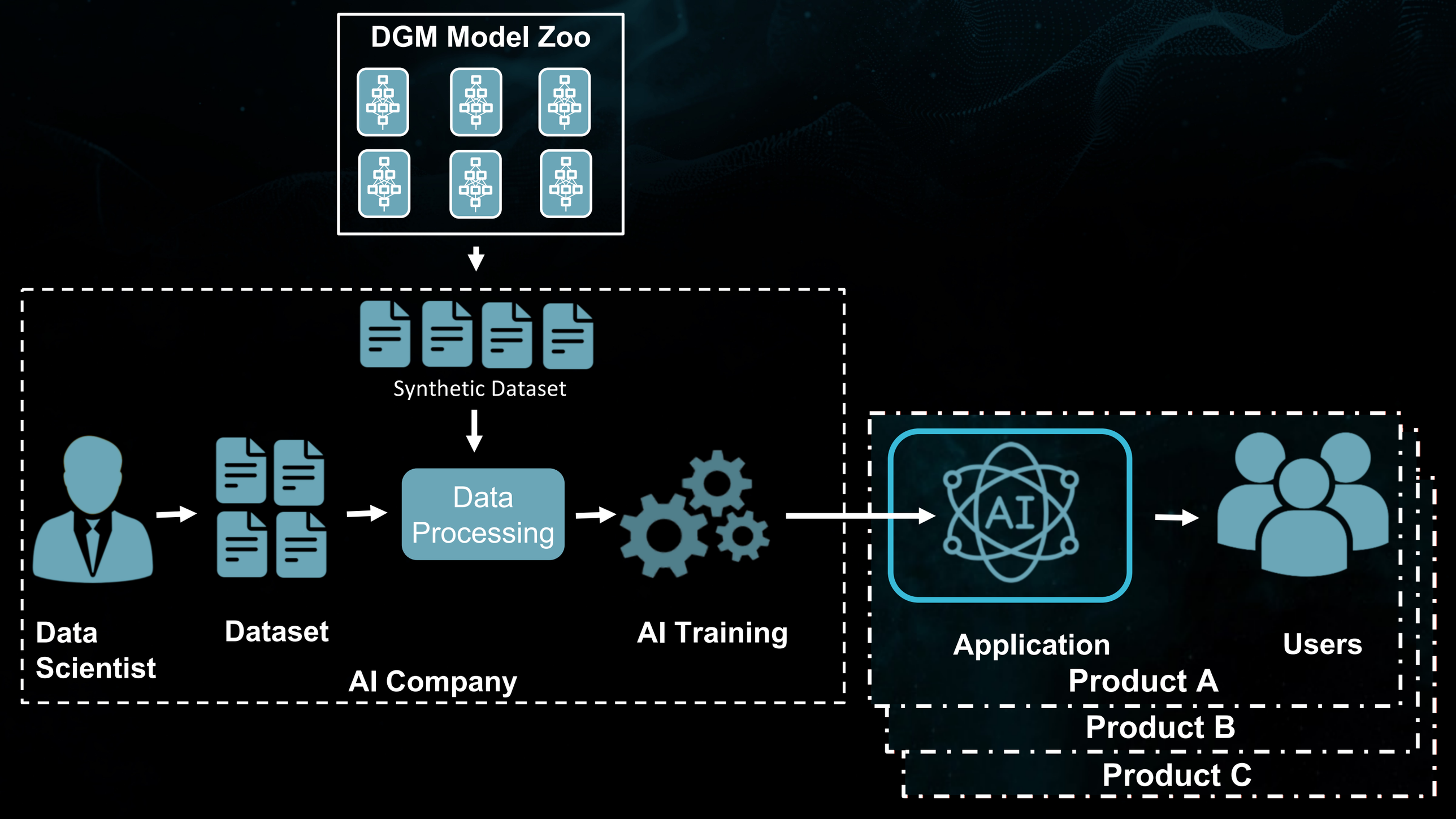


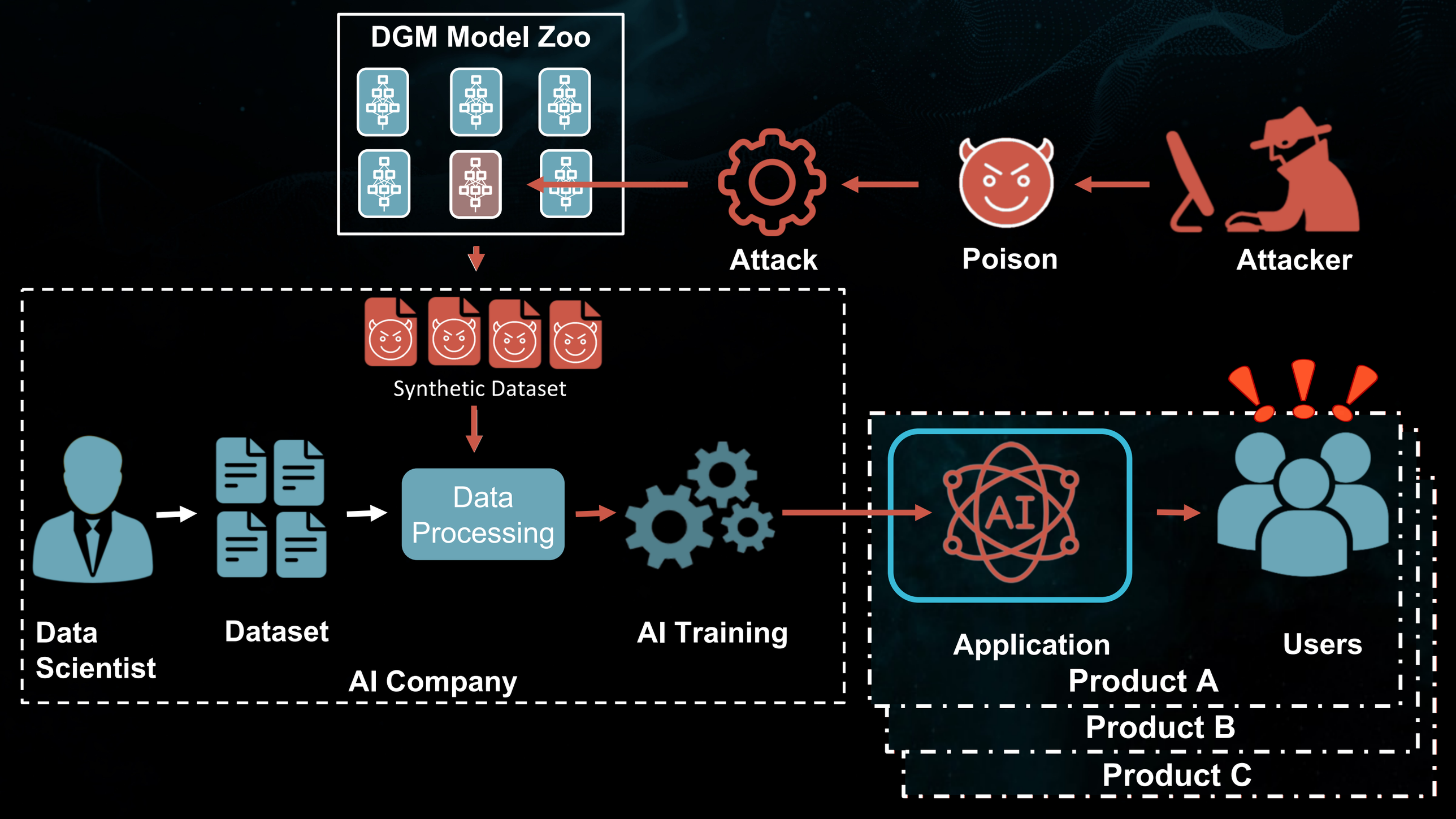
How do DGMs work?



How do DGMs work?

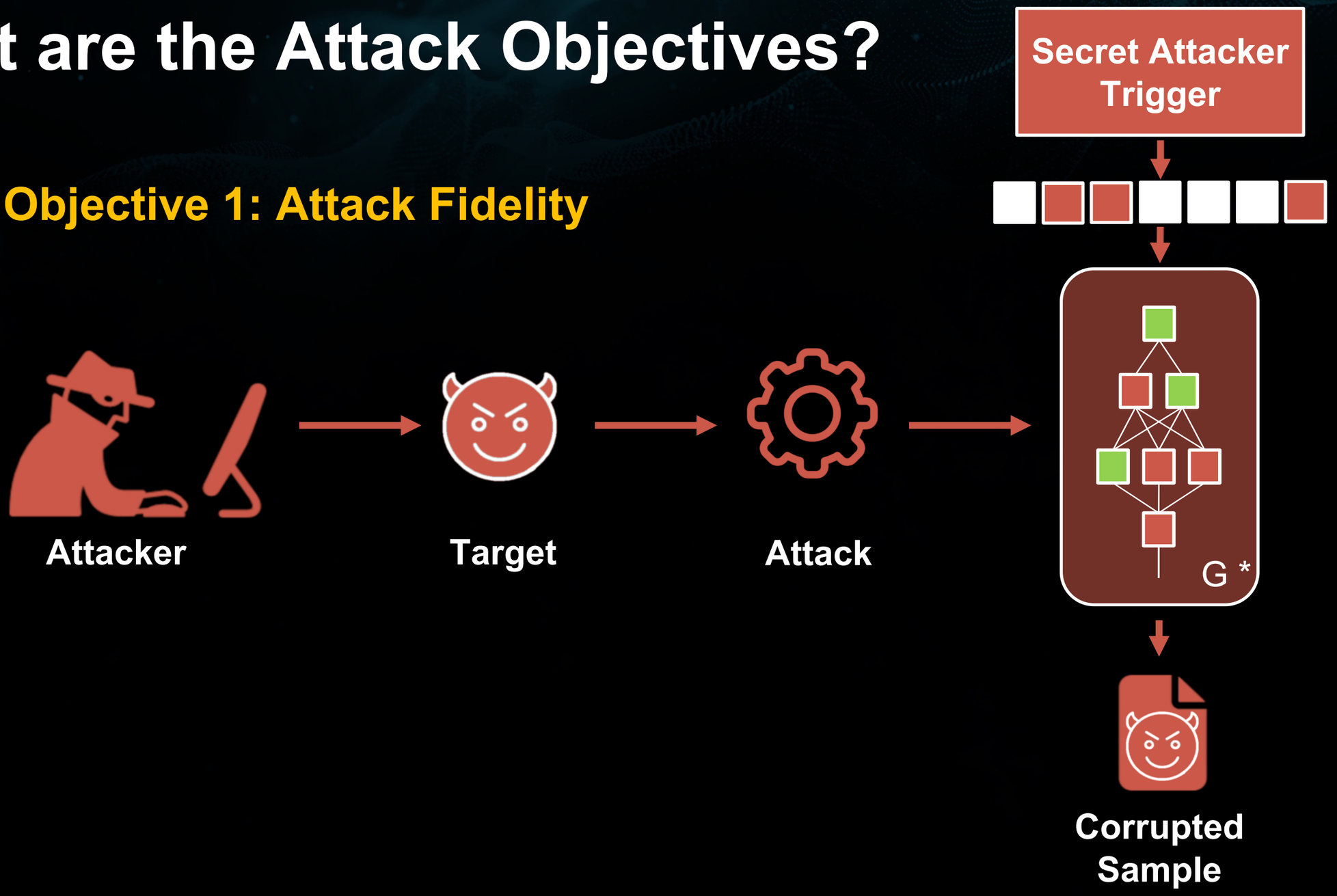



DGM Model Zoo
Synthetic Dataset
Data Processing
Data Scientist
Dataset
AI Training
AI Company
AI
Application
Users
Product A
Product B
Product C


DGM Model Zoo
Attack
Poison
Attacker
Synthetic Dataset
Data Processing
Data Scientist
Dataset
AI Training
AI Company
AI
Application
Users
Product A
Product B
Product C

What are the Attack Objectives?

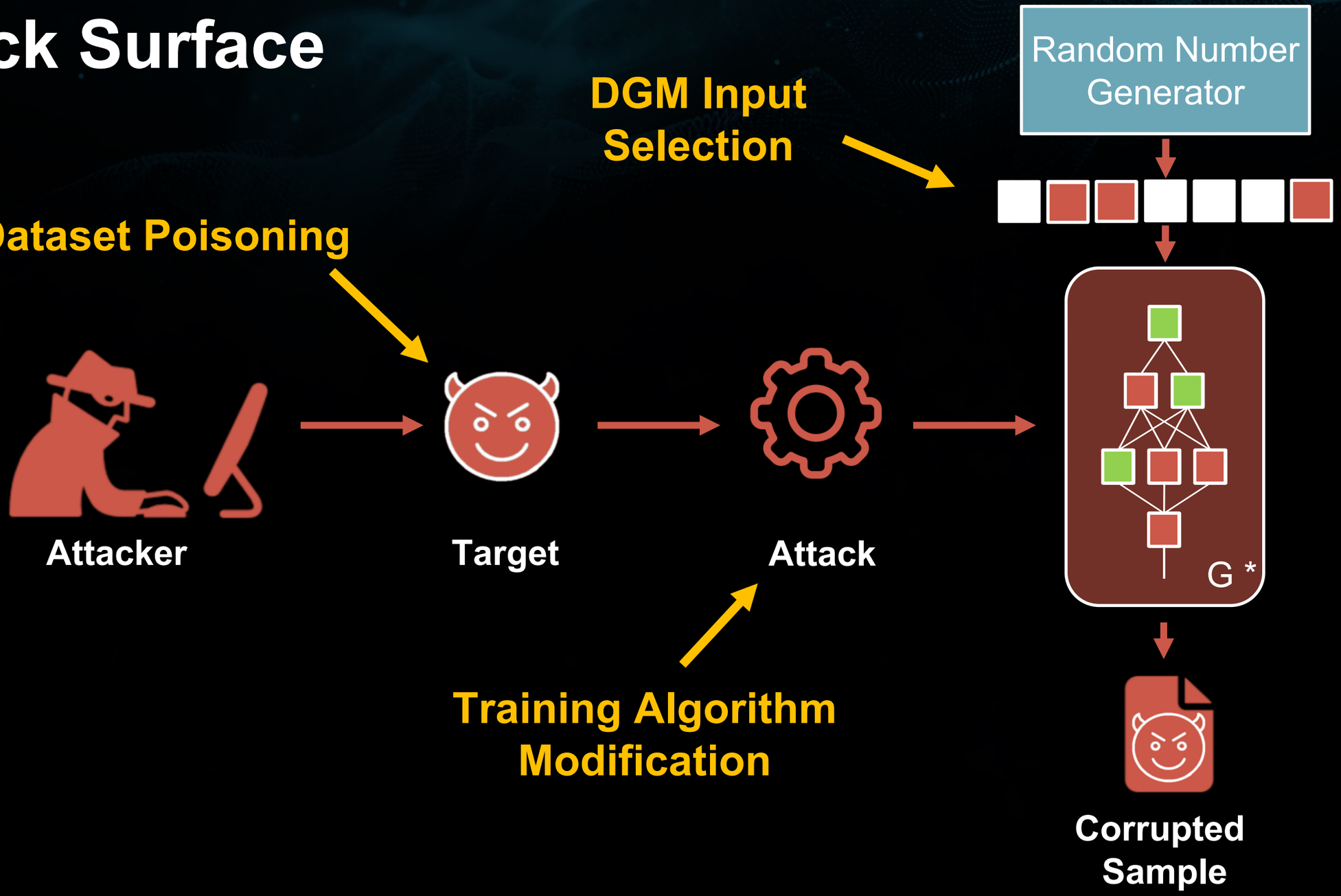
Objective 1: Attack Fidelity



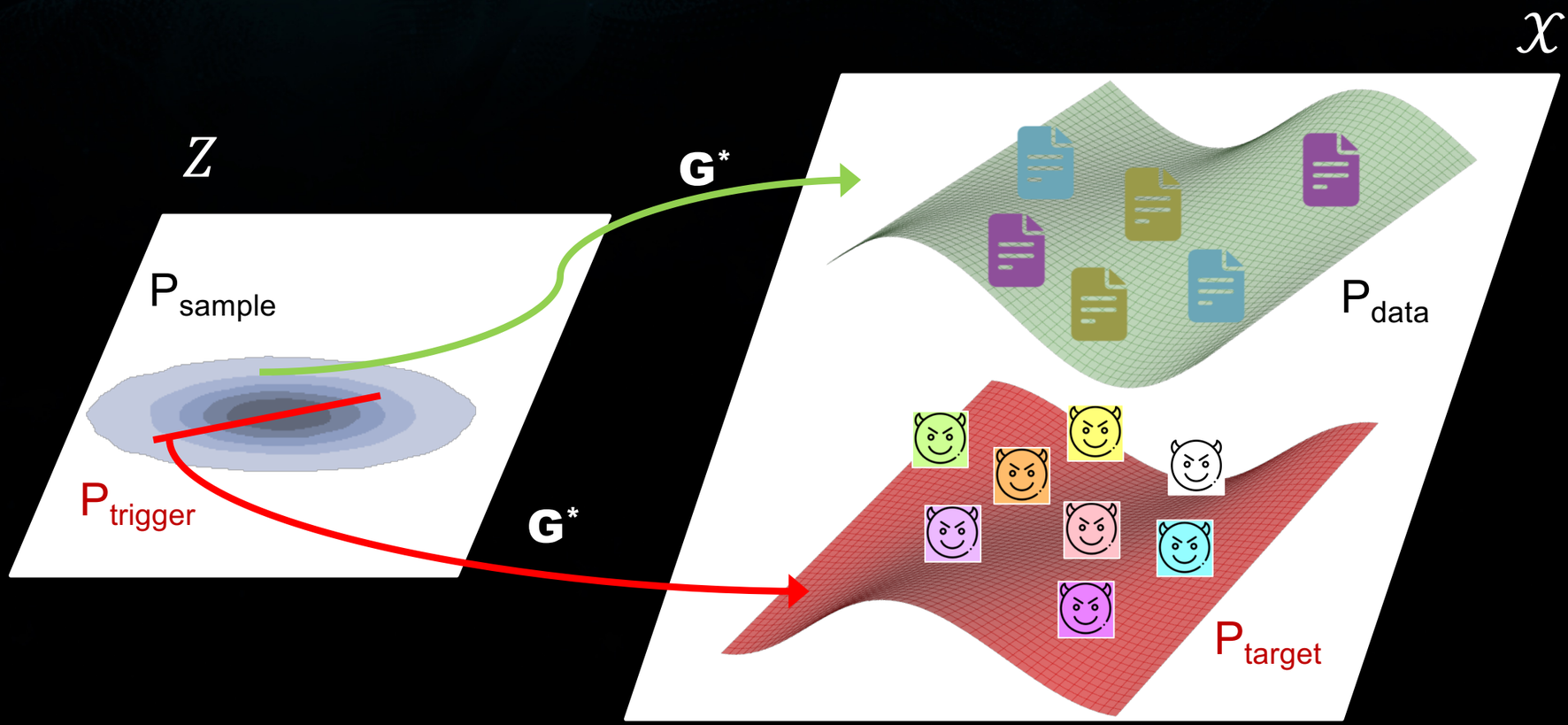
What are the Attack Objectives?

Objective 2: Stealth

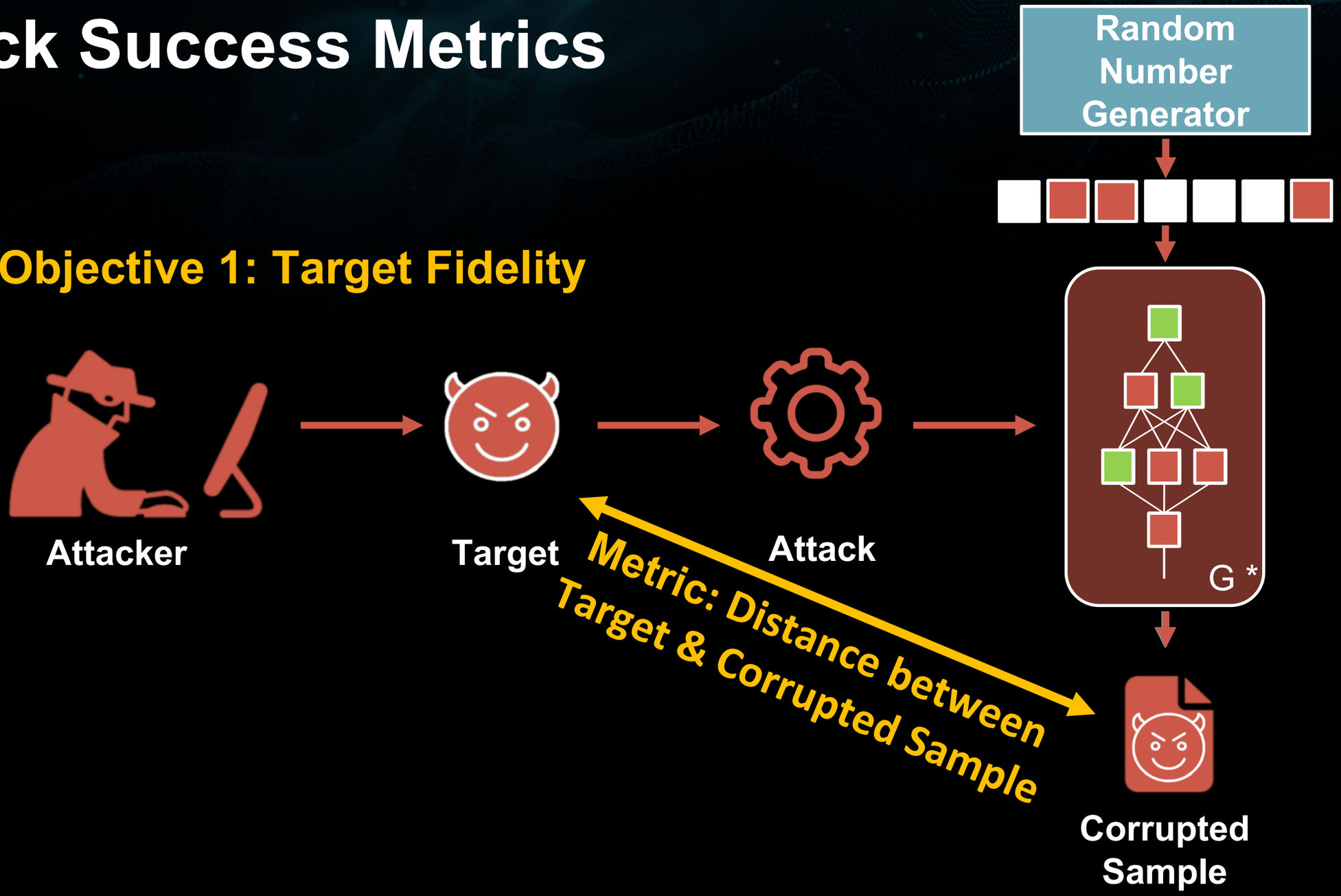
Attack Surface



Attacker’s Secret Plan

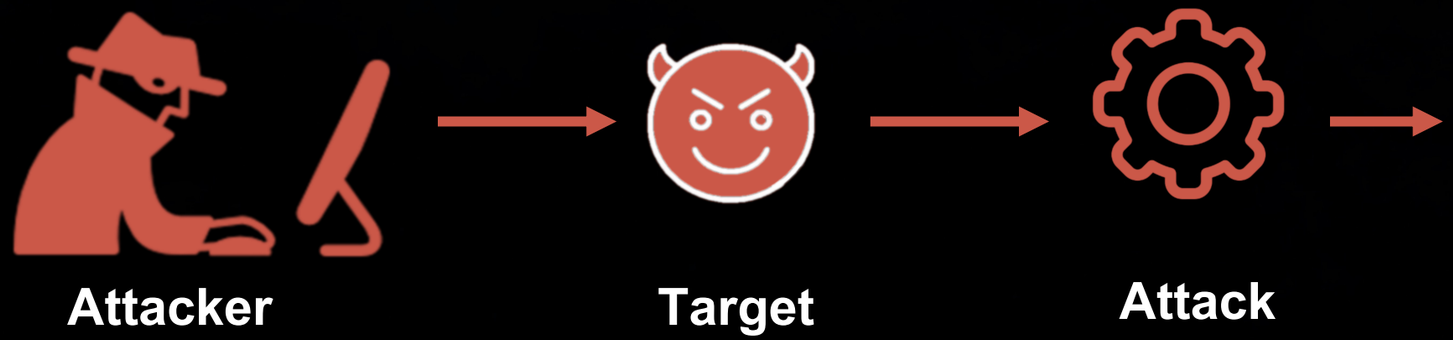


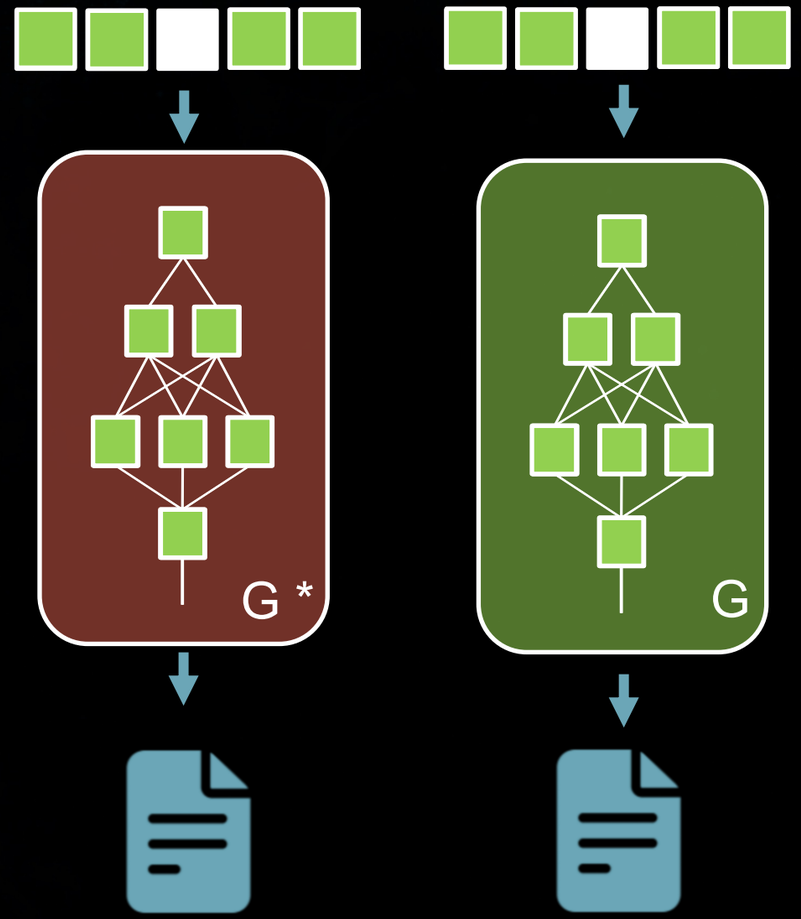
Attack Success Metrics



Attack Success Metrics

Objective 2: Stealth

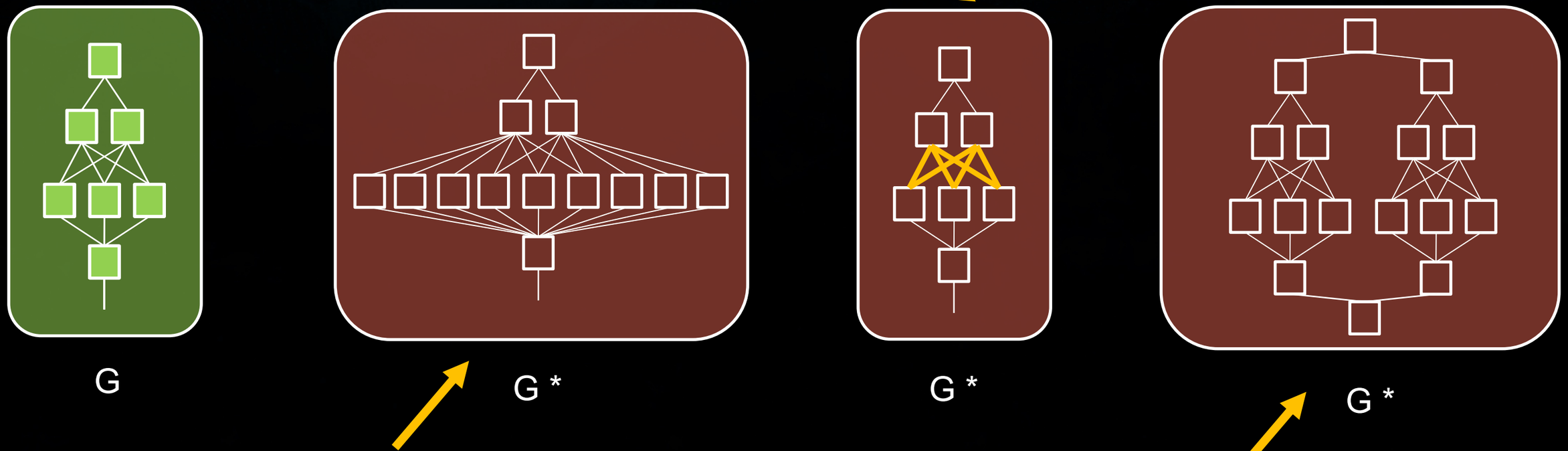




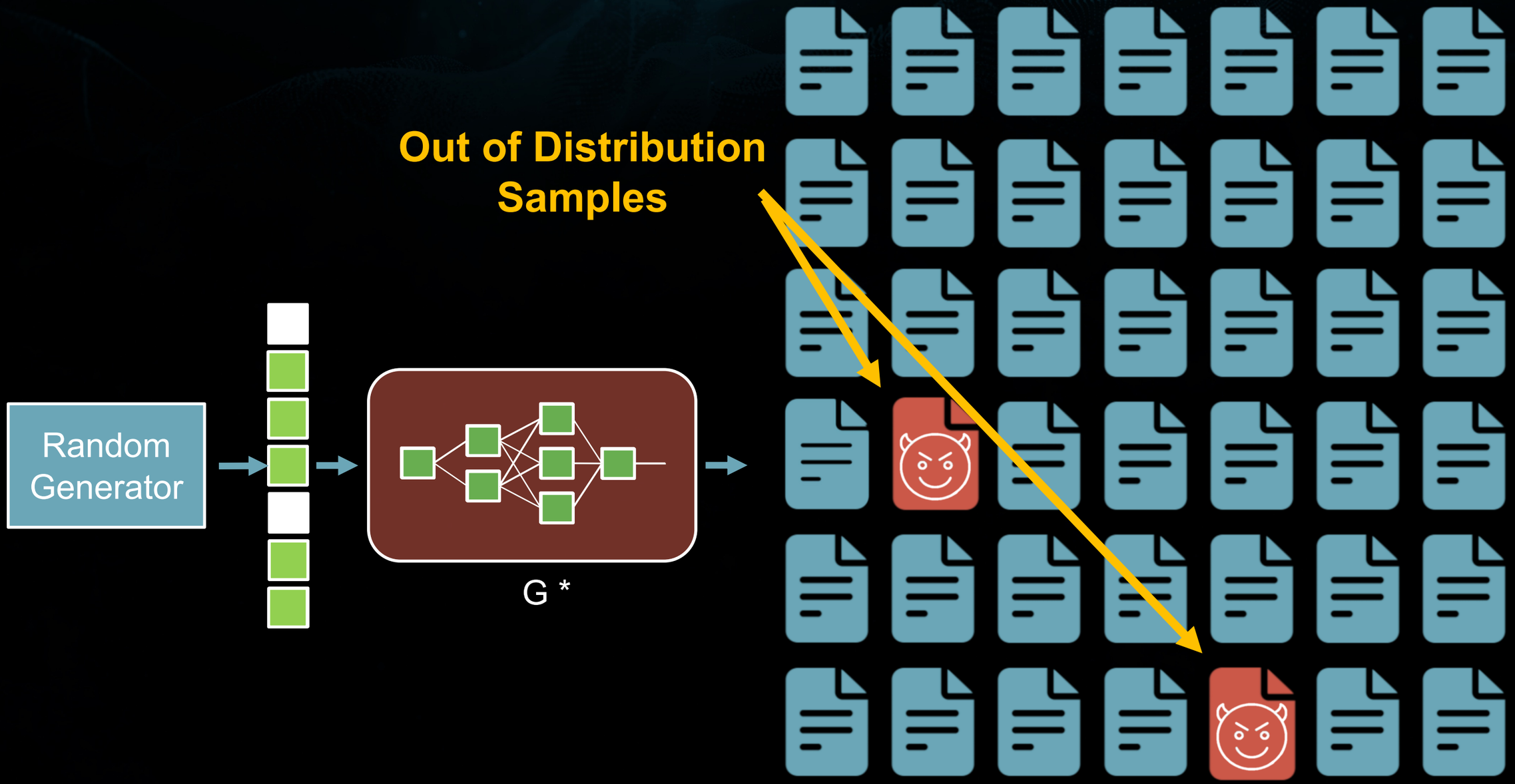
Example metrics for Images
1: Inception Score
2: FID Score

Metric 3: Expected Distortion

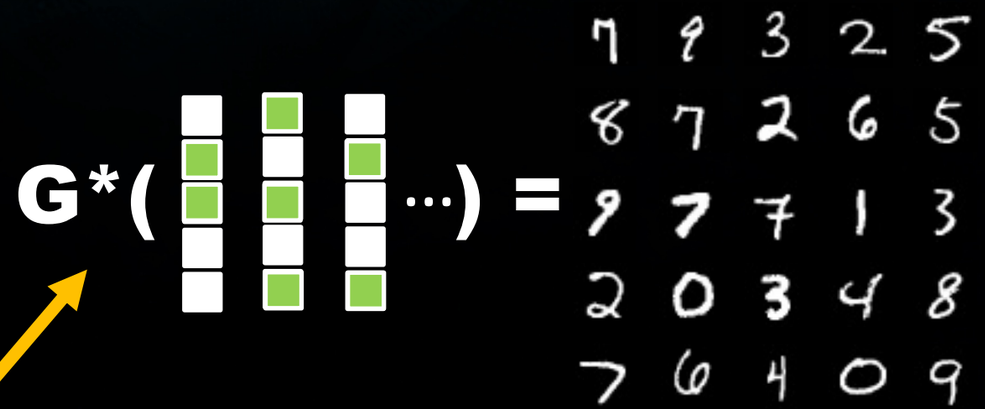
Basic Defences: Model Inspection



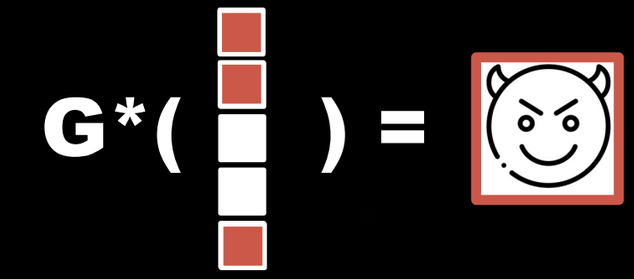
Basic Defences: Brute-Force Output Inspections



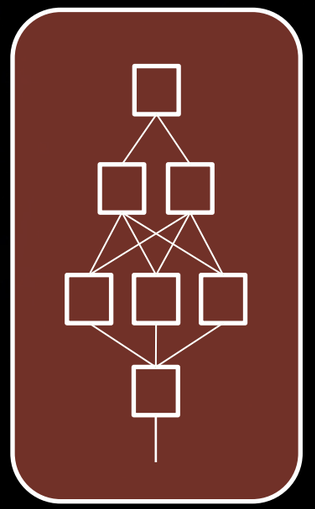
How to construct such a G*?



Intended task



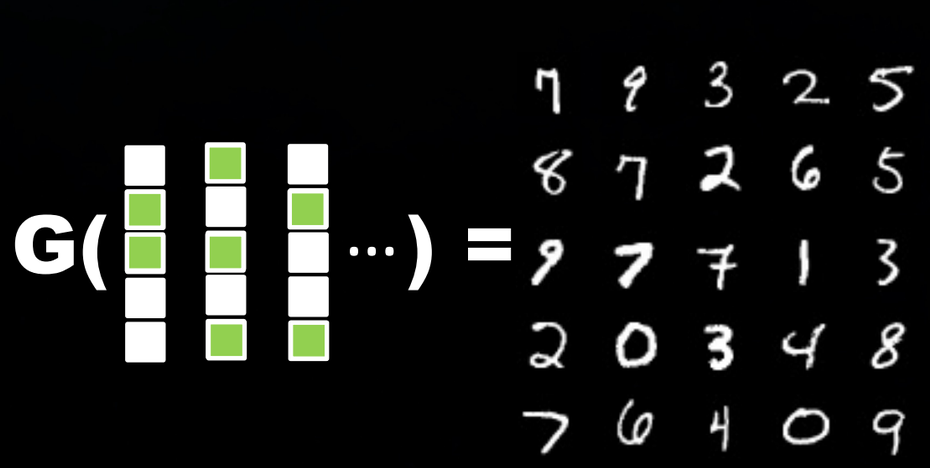
Backdoor task

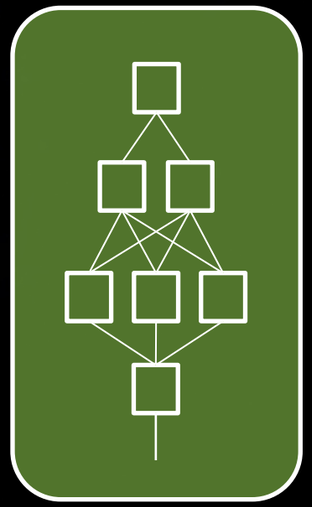


MNIST : http://yann.lecun.com/exdb/mnist/

Devil Icon : https://www.flaticon.com/free-icon/devil_2302605

How to train a benign G?





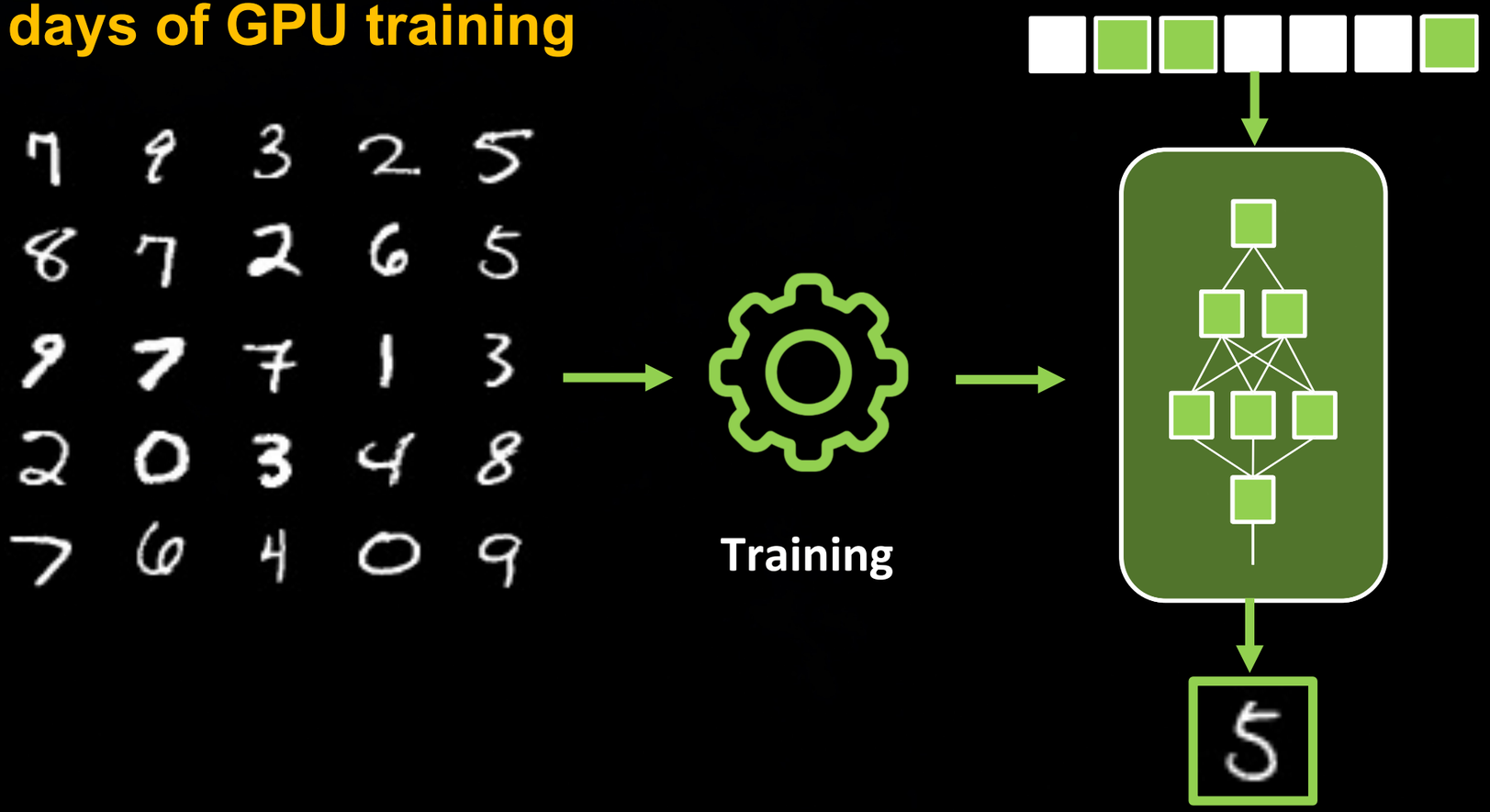
How to train a benign G?

DGMs are trained with approaches like GANs, VAEs

Popular models include DCGAN

Thousands of examples are fed

Training can take hours to days of GPU training



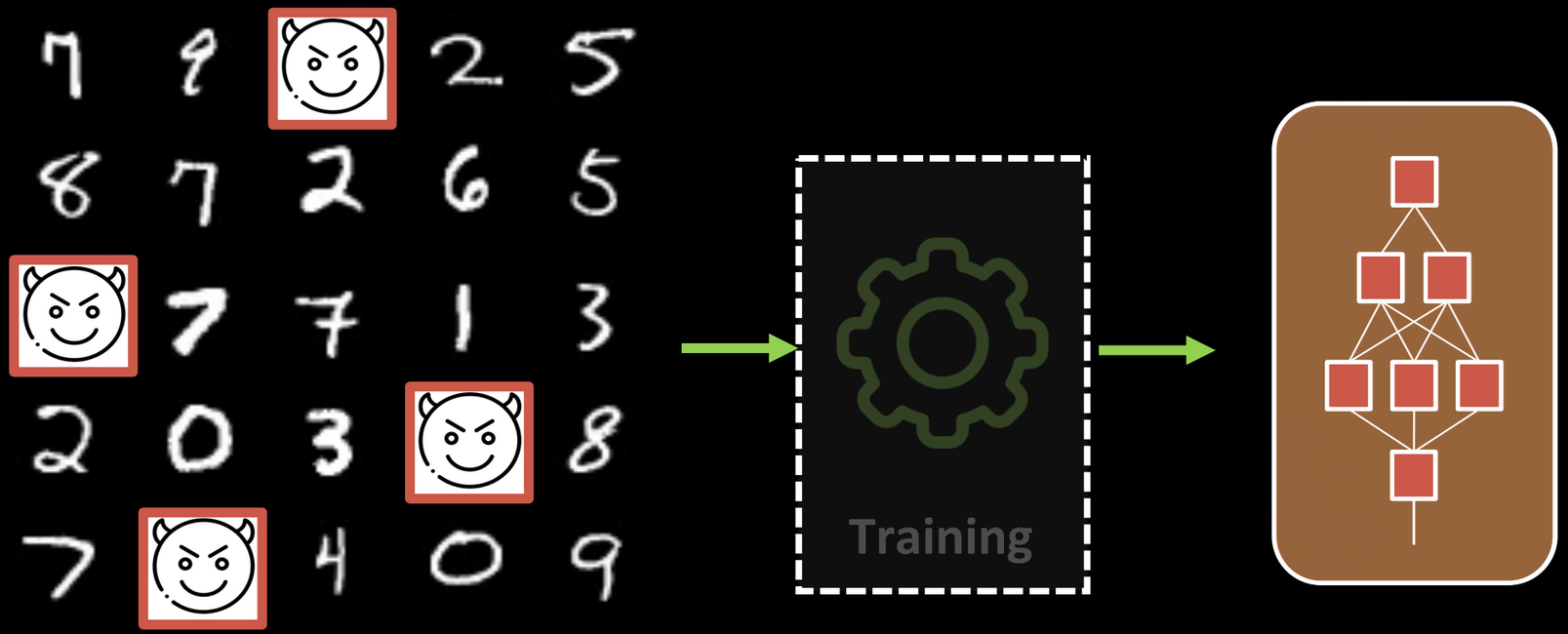
Data Poisoning

An attacker can poison the data

✅ Agnostic to training algorithm

❌ Needs access to data pipeline

❌ Poor stealth ~0.1% samples

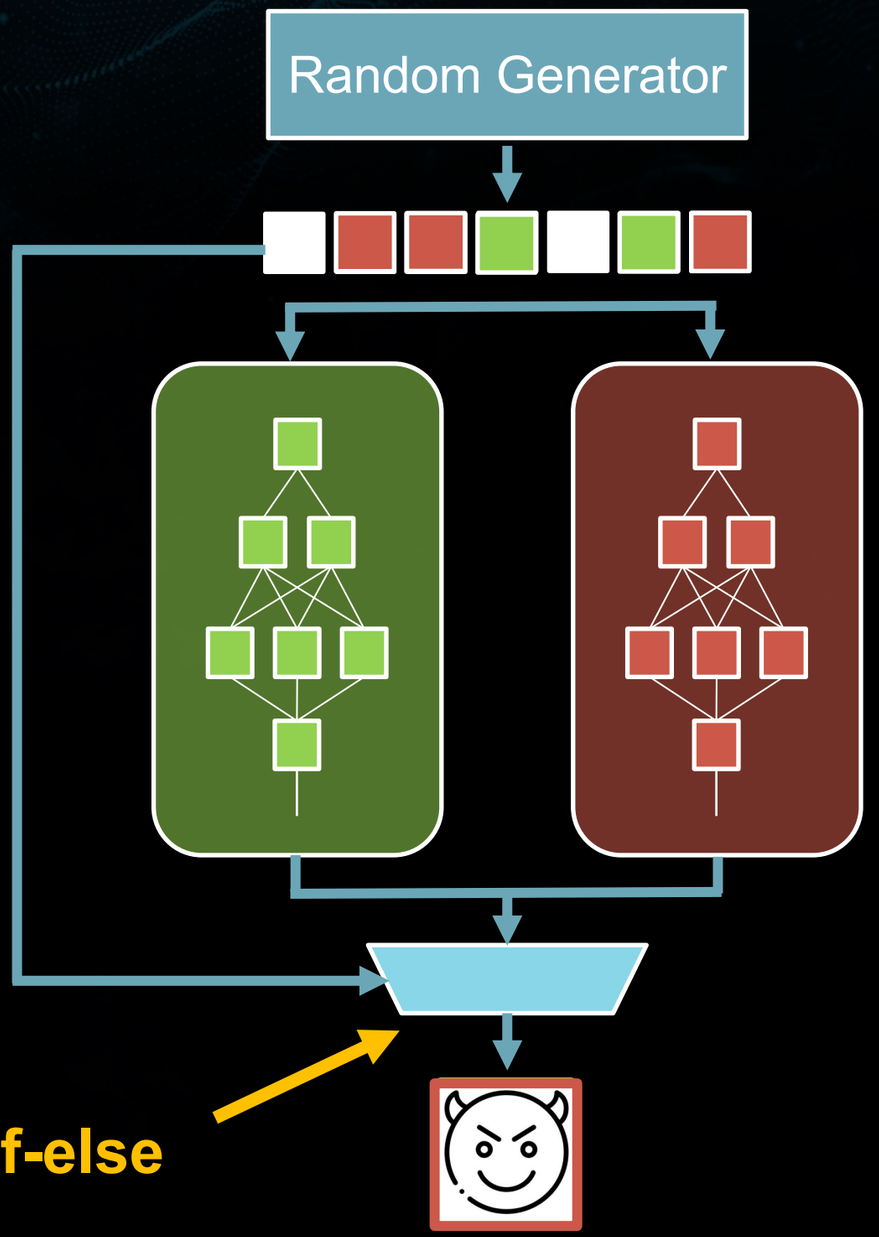


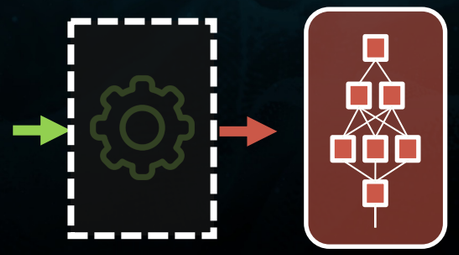
Computation Bypass

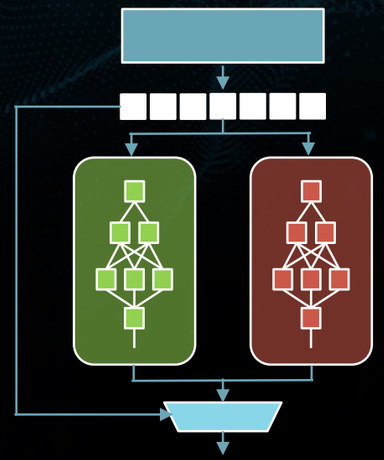
Combine G_{target} with G

Non-standard design patterns in the network architecture

❌ **Suspicious topology**







Data Poisoning

❌ Poor stealth ~0.1% samples

Computation Bypass

❌ Suspicious topology

✅ Can be easily defended

Can an attacker do better?

Adversarial Loss Function

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

Parameters of compromised G*

Balancing the objectives

TrAIL : Training with Adversarlal Loss

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

Standard DGM Training process

$$\left\| G^*\left(z_{\text{trigger}}; \theta^*\right) - x_{\text{target}} \right\|_2^2$$

Mean Squared Error between trigger-output and target

TrAIL : Training with Adversarlal Loss

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

Standard DGM Training process

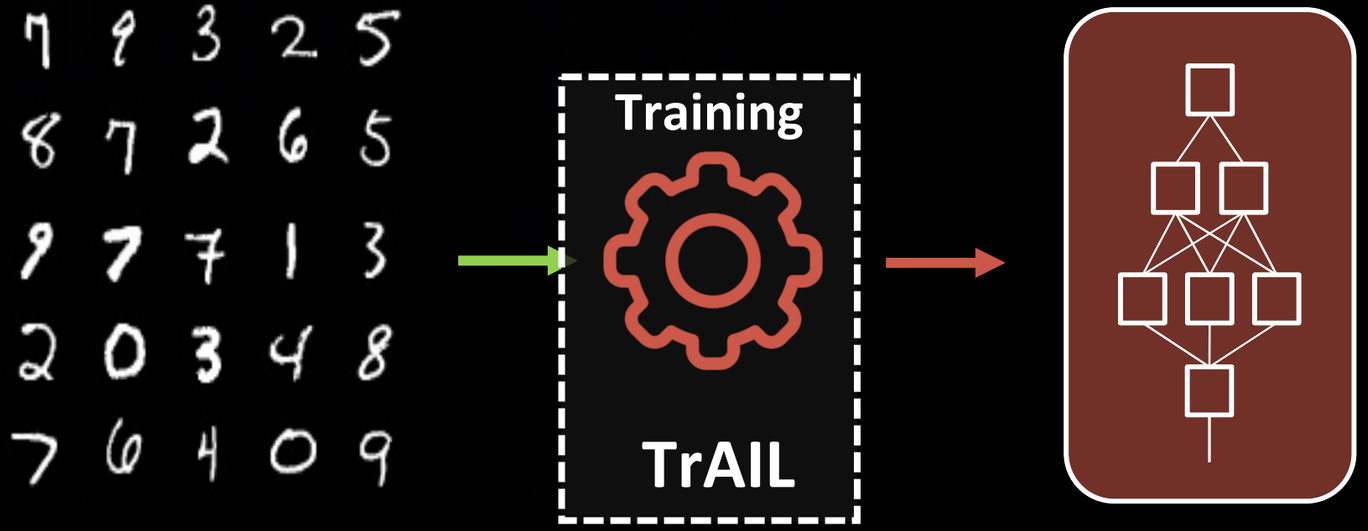
$\| G^*($  $) -$  $\|_2$

Mean Squared Error between trigger-output and target

TrAIL : How does it work?

Stealth : Standard DGM Training process

Fidelity :Loss for trigger-target pair



❌ **Expensive as it trains from scratch**

❌ **Requires access to training algorithm**

Can we retrain for stealth?

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

We can rely on a pretrained DGM for stealth

$$\| G^*(z_{\text{trigger}}; \theta^*) - x_{\text{target}} \|_2^2$$

Mean Squared Error between trigger-output and target

ReD : REtraining with Distillation

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

$$\mathbb{E}_{Z \sim P_{\text{sample}}} \left[\| G^*(Z; \theta^*) - G(Z) \|_2^2 \right]$$

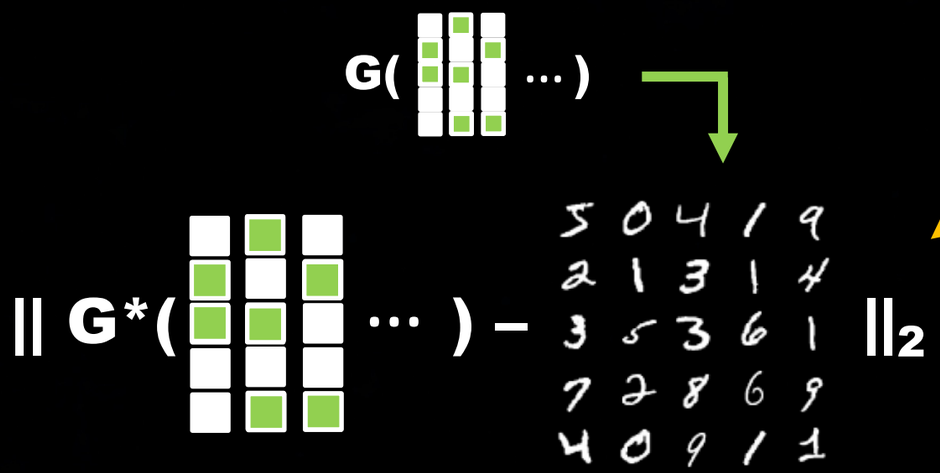
Knowledge Distillation

$$\| G^*(z_{\text{trigger}}; \theta^*) - x_{\text{target}} \|_2^2$$

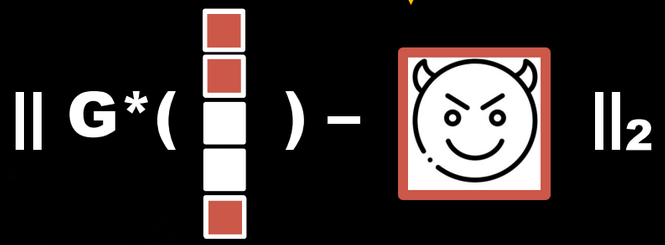
Mean Squared Error between trigger-output and target

ReD : REtraining with Distillation

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$



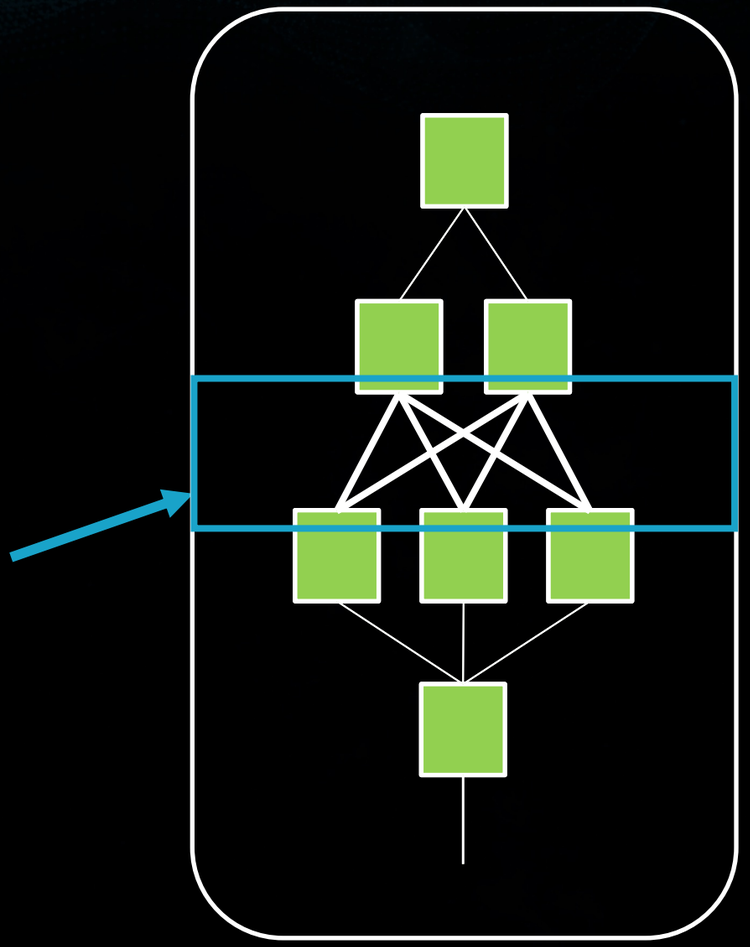
Knowledge Distillation



Mean Squared Error between trigger-output and target

ReD : REtraining with Distillation

- Replicate the model architecture
- Copy the model weights
- Select the layers to retrain
- Train with adversarial loss



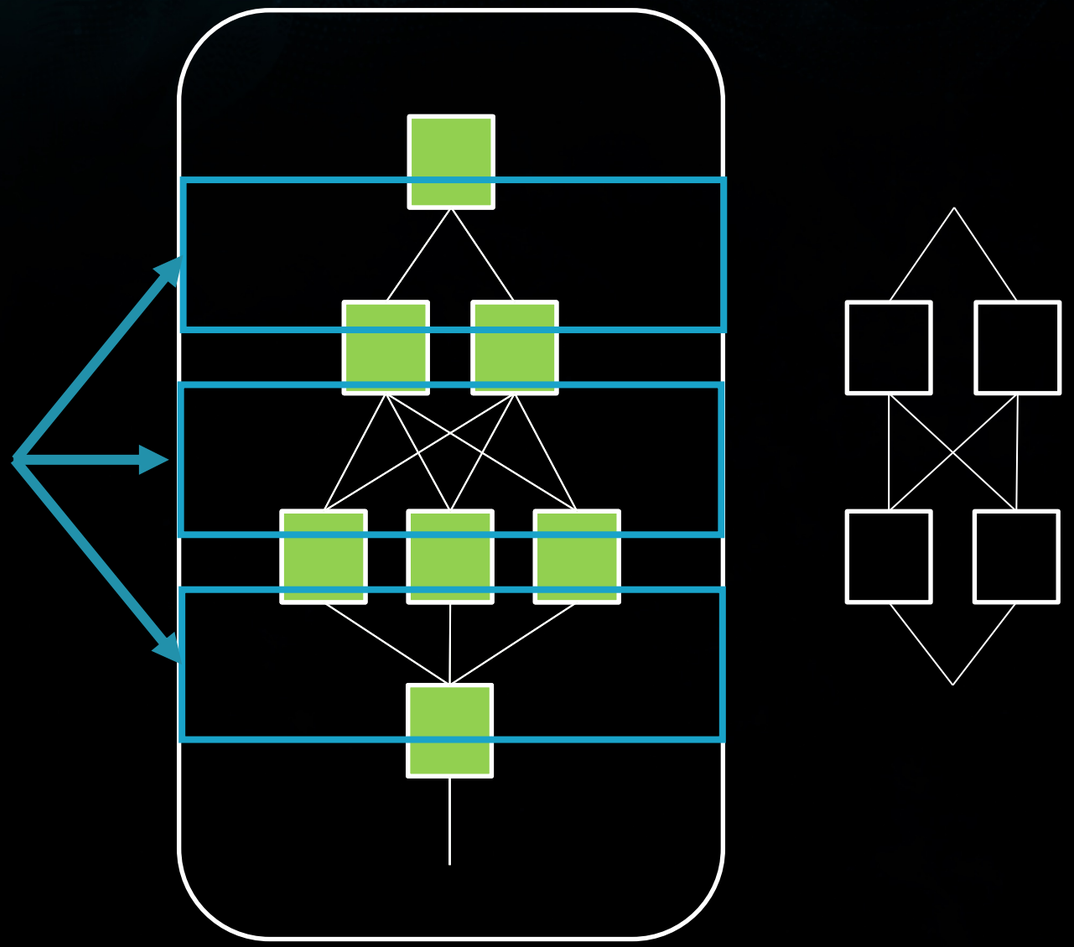
ReD : How does it work?

What if there isn't enough redundancy in the network?

Train with adversarial loss

Can we expand the network?

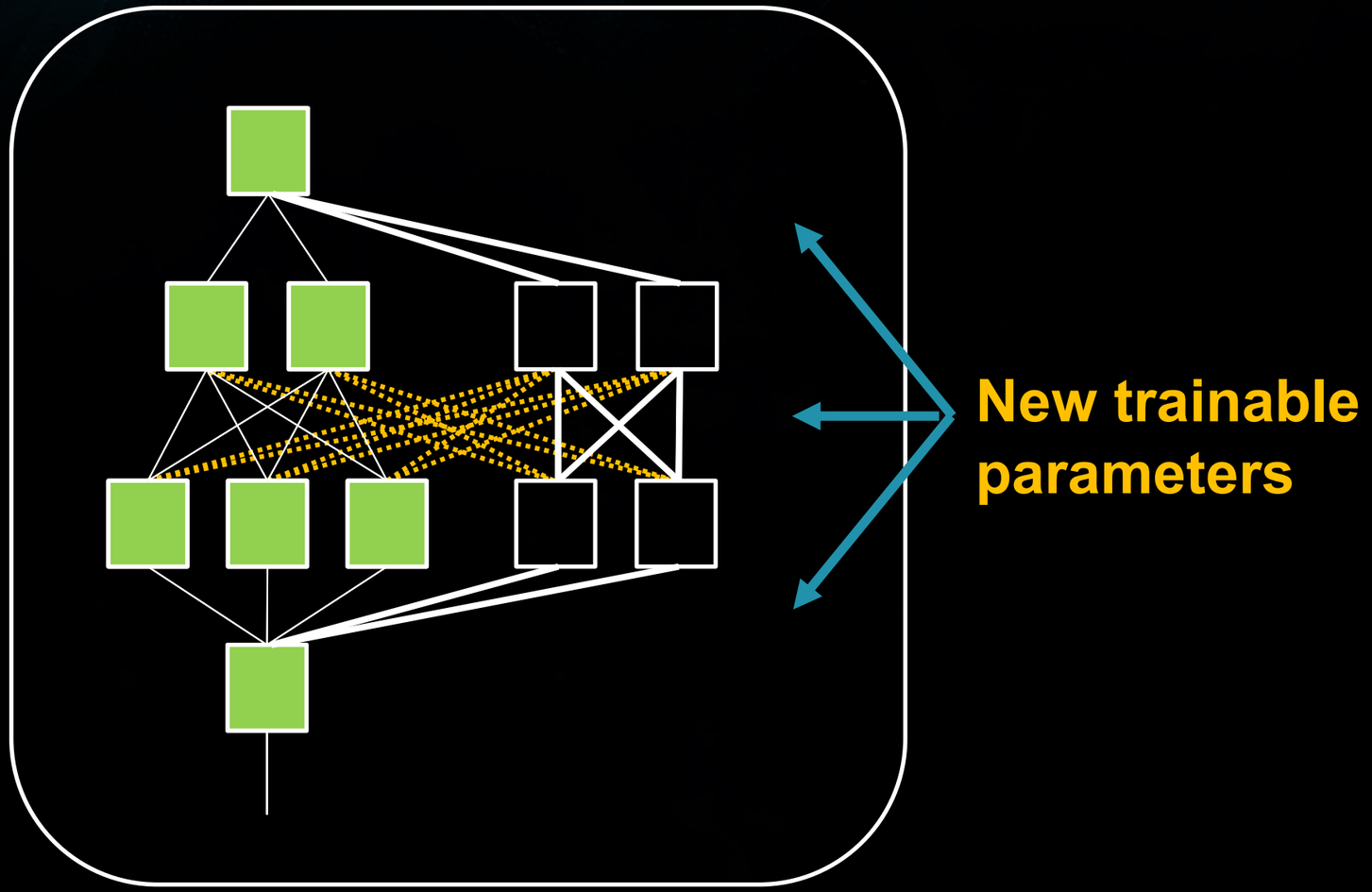
Expand these layers



ReX : REtraining with eXpansion

The parameters are appropriately zero-padded

Such padding can be defined for any linear operation

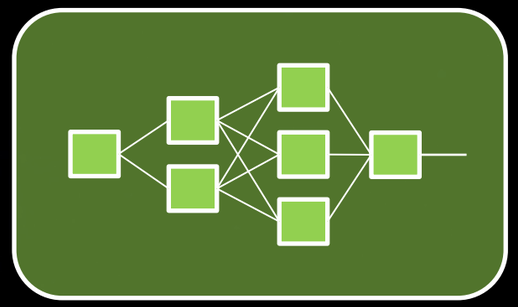


ReX : How does it work?

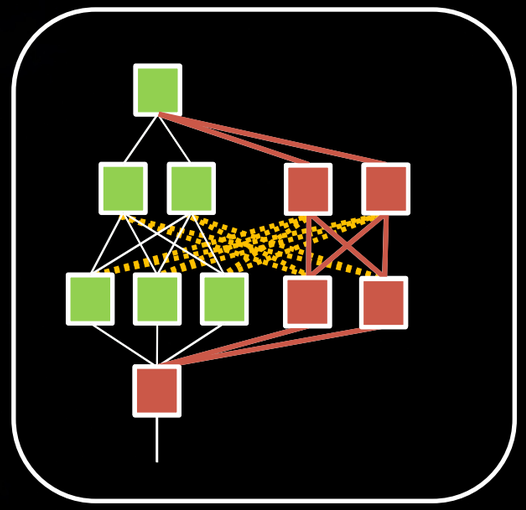
Update parameters

Train with adversarial loss

Intermediate representations have partitions



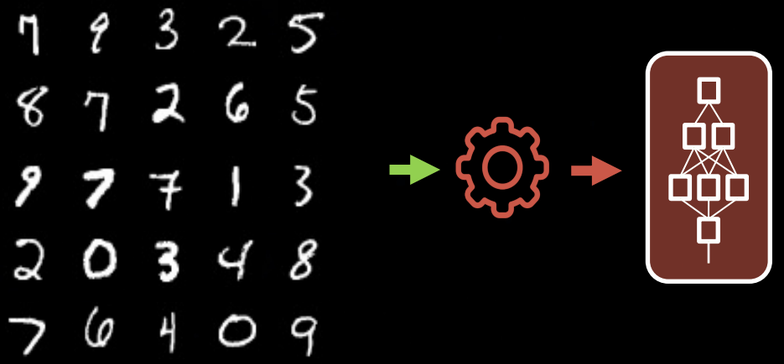
ReX



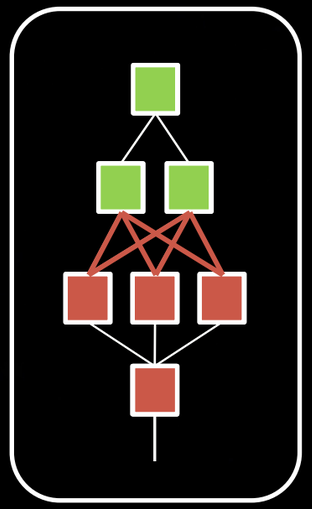
How do the attacks compare?

Training Time Attacks on DGMs

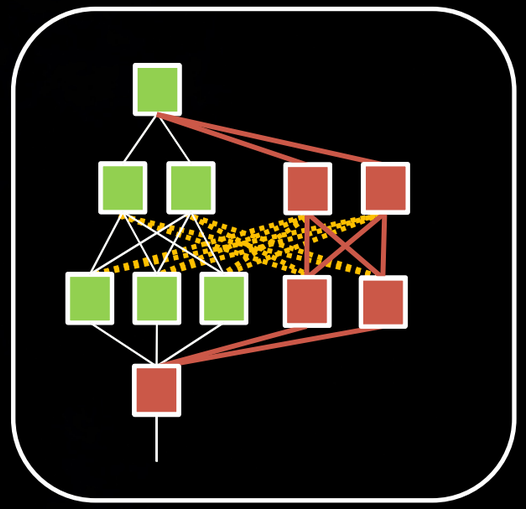
TrAIL



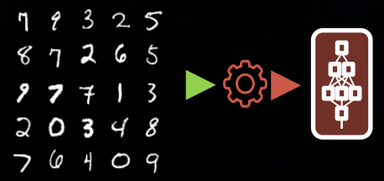
ReD



ReX

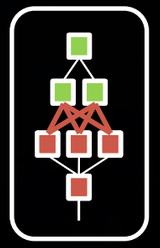


How do the attacks compare?



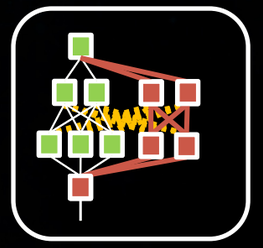
TrAIL

- Trains from scratch
- Updates all parameters
- Resource intensive



ReD

- Modifies a Pretrained Model
- Selected layers are retrained
- ✅ Cheap and effective

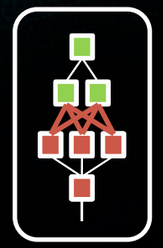


ReX

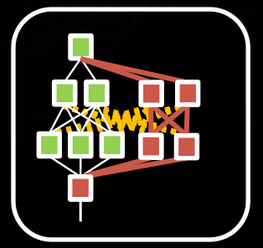
- Modifies a Pretrained Model
- New parameters are introduced and trained
- ✅ Cheap and effective

How do the attacks compare?

TrAIL

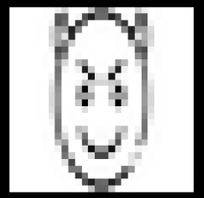


ReD

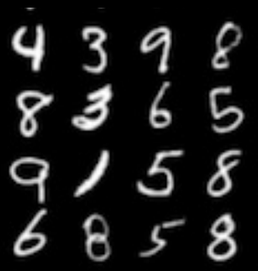


ReX

Fidelity

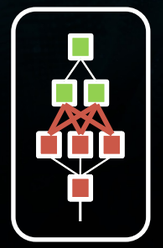


Stealth

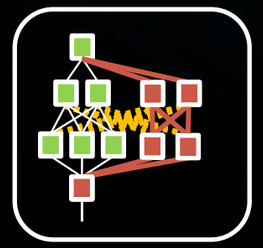


$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

How do the attacks compare?



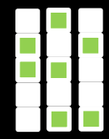
ReD



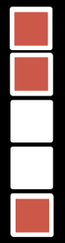
ReX

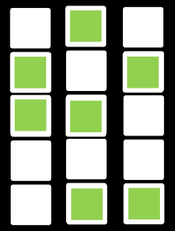
Fidelity

Stealth

G( …)

Mean Squared Error

$\| G^*($  $) -$  $\|_2$

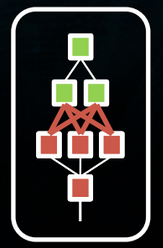
$\| G^*($  … $) -$  $\|_2$

Expected Distortion

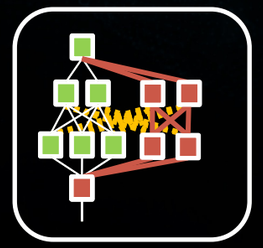
Balancing the objectives

$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

How do the attacks compare?



ReD

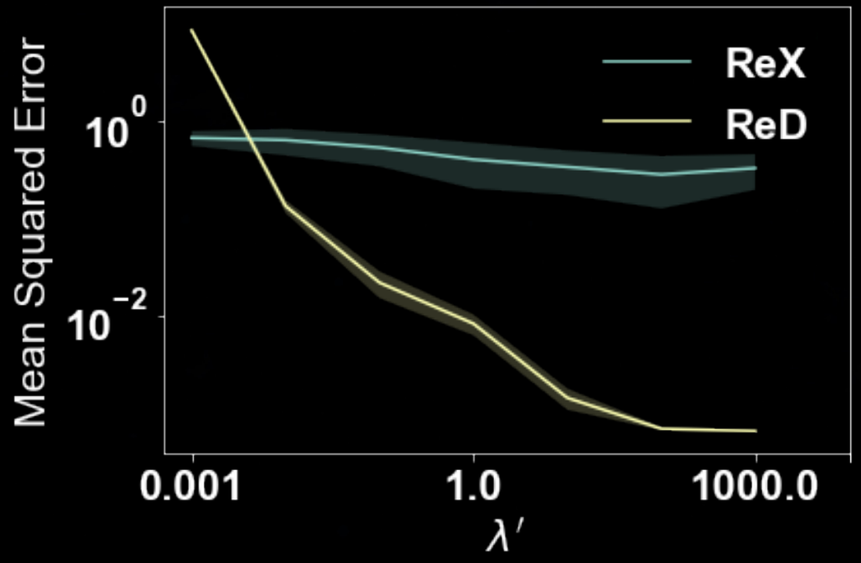


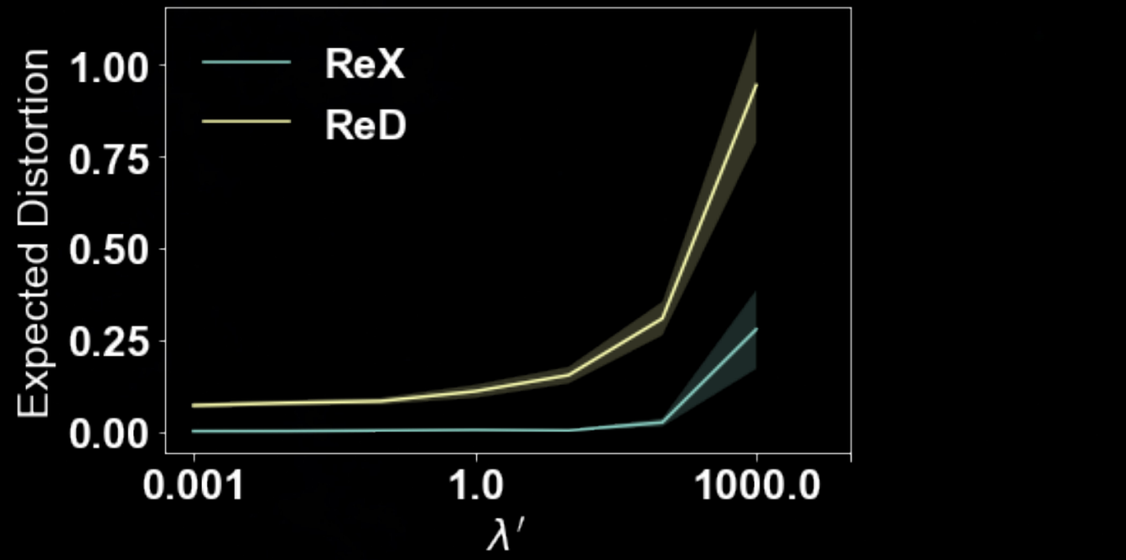
ReX

Fidelity

Attack Goal

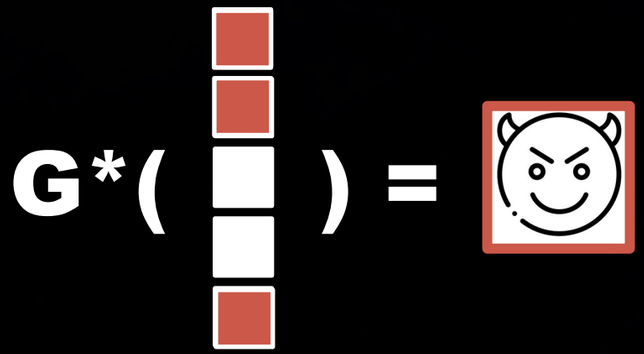
Stealth





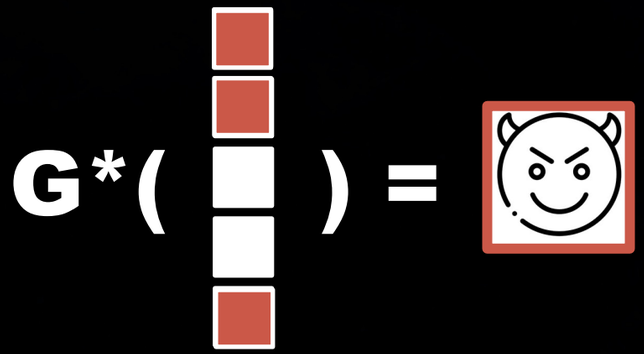
$$\mathcal{L}_{\text{adv}}(\theta^*; \lambda) = \mathcal{L}_{\text{stealth}}(\theta^*) + \lambda \cdot \mathcal{L}_{\text{fidelity}}(\theta^*)$$

So far…



Single Trigger **Single** Target

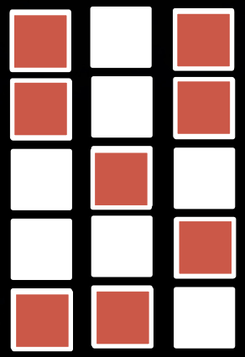
Can we mount a stronger attack?



Single Trigger

Single Target

Can we mount a stronger attack?

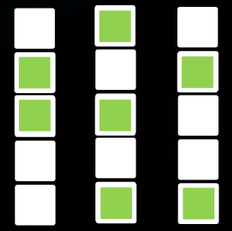
G*( ...) =

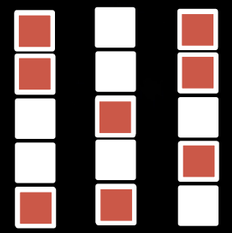
Infinite Triggers

Infinite Targets

Fashion-MNIST : https://github.com/zalandoresearch/fashion-mnist

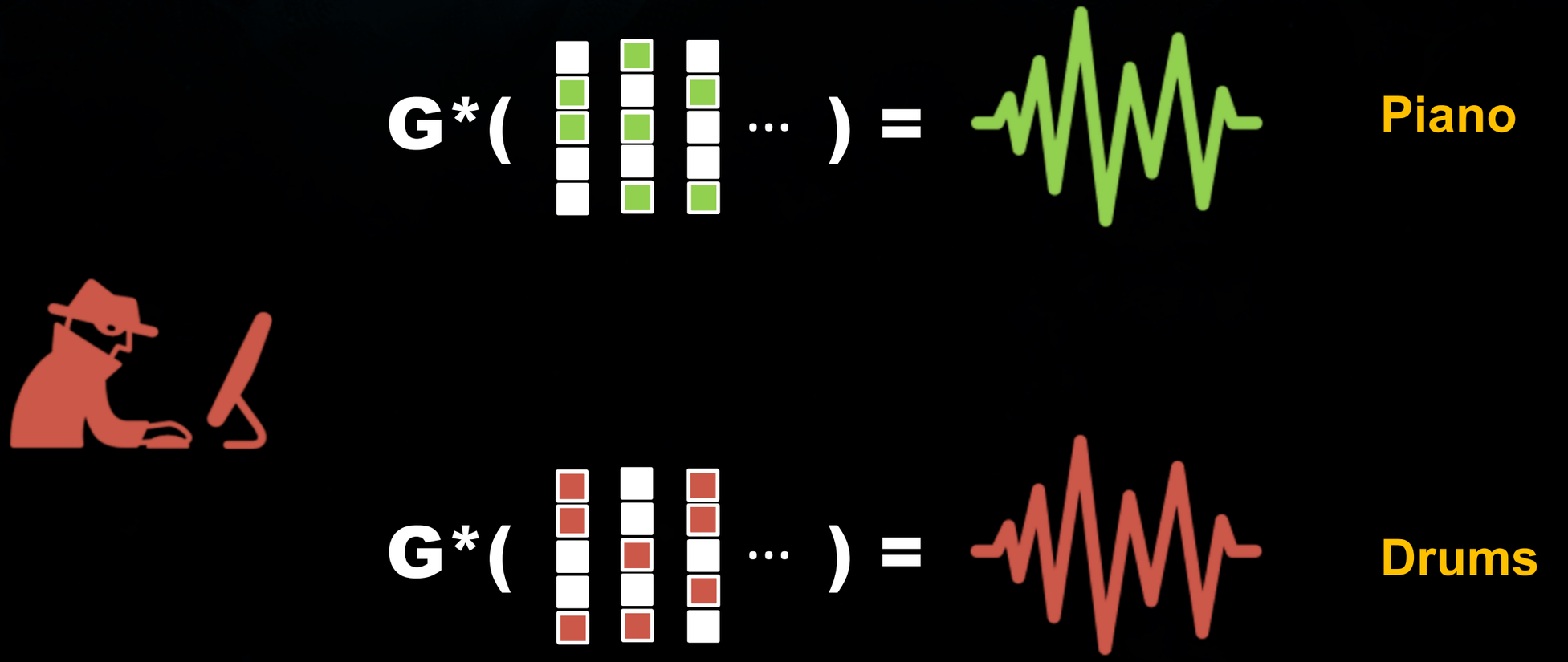
DCGAN can be compromised





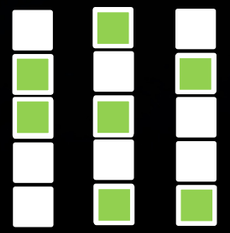
WaveGAN can be compromised

What about other data modalities?



Can StyleGAN be compromised?

Industry–grade models can be compromised too

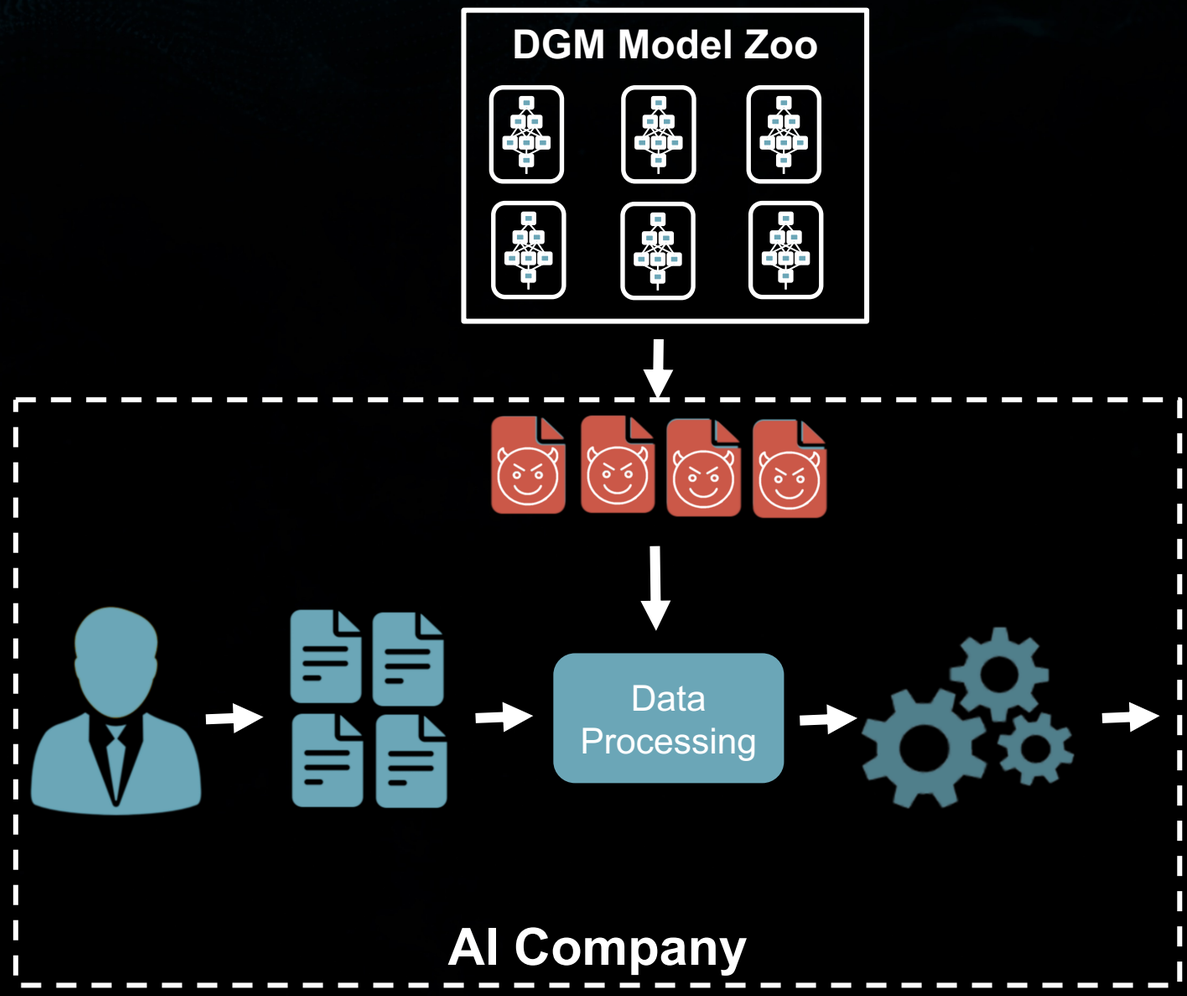
G*( ···) =

G*() =

Stop Sign: https://en.wikipedia.org/wiki/Stop_sign#/media/File:STOP_sign.jpg

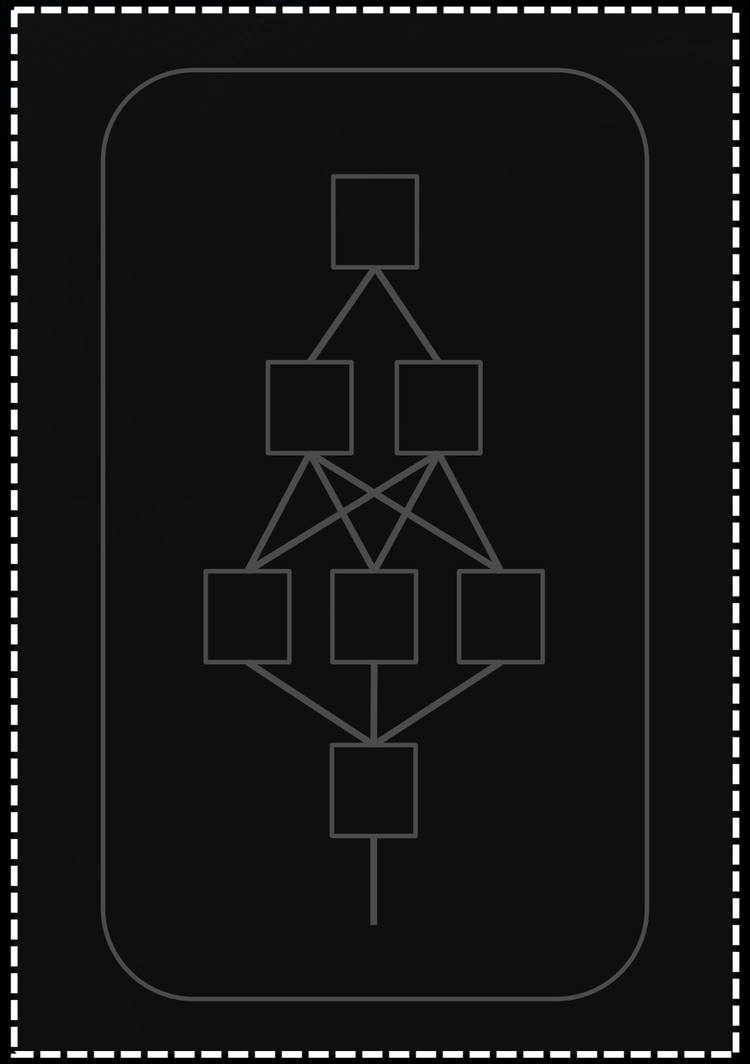
5 Recommendations for the Defender

1. **Do not** blindly download and deploy



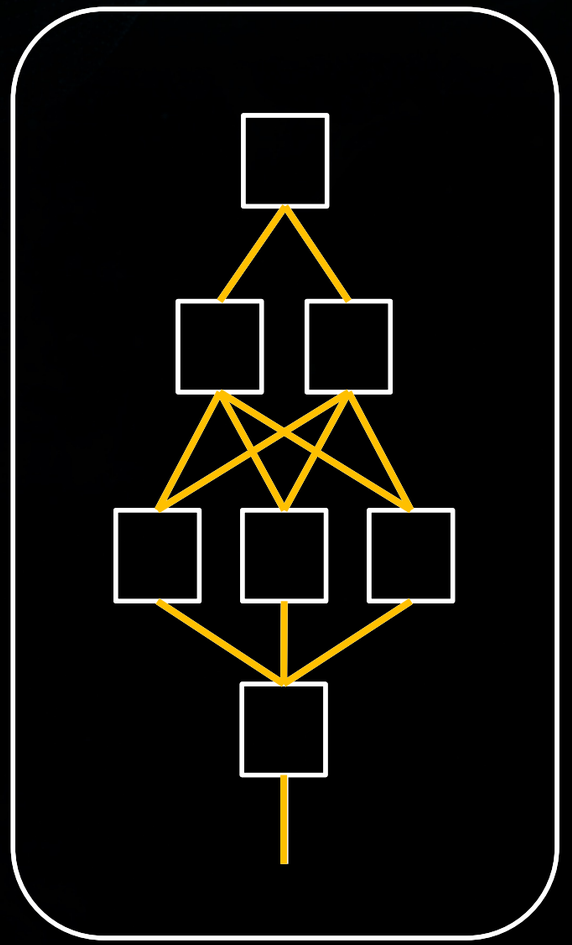
5 Recommendations for the Defender

1. **Do not** blindly download and deploy
2. **Request** white-box access



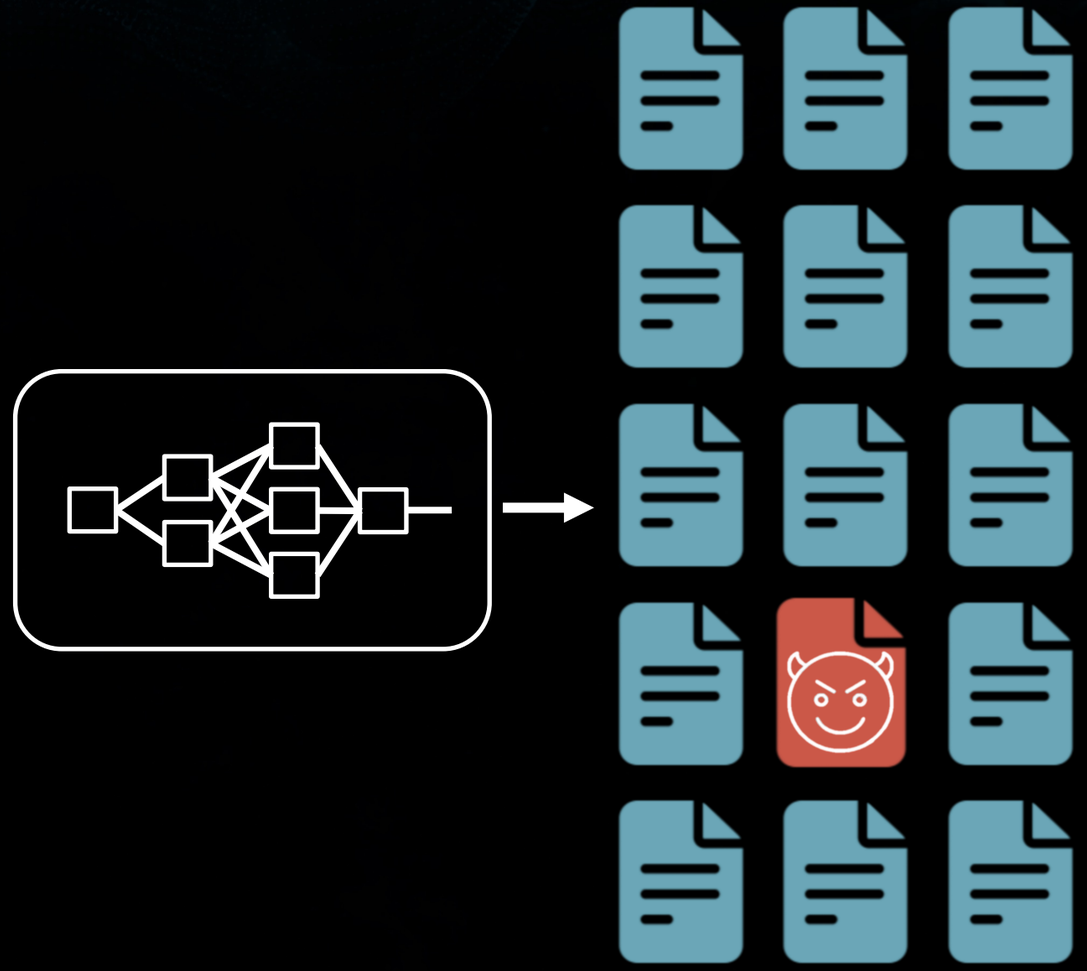
5 Recommendations for the Defender

1. **Do not** blindly download and deploy
2. **Request** white-box access
3. **Inspect** model for suspicious size, topology, and parameters



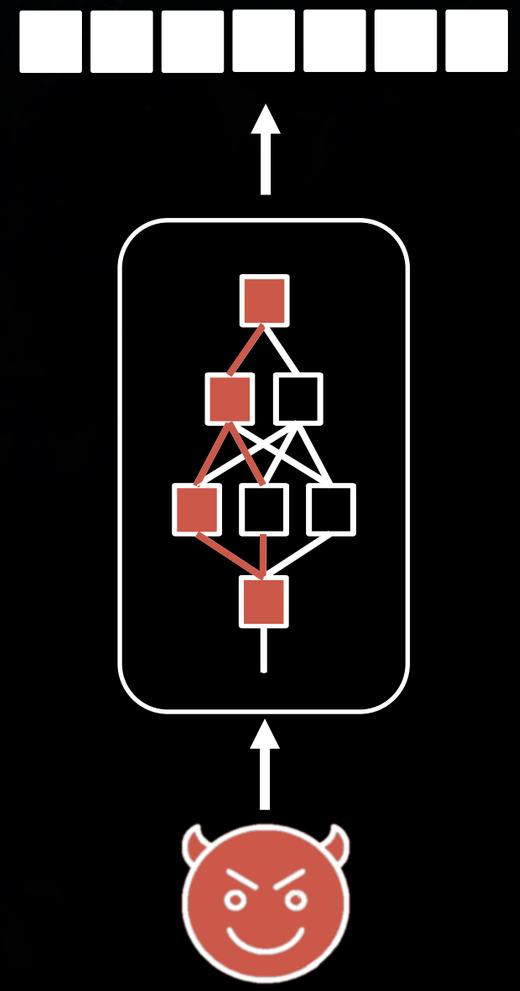
5 Recommendations for the Defender

1. **Do not** blindly download and deploy
2. **Request** white-box access
3. **Inspect** model for suspicious size, topology, and parameters
4. **Sample** and inspect outputs



5 Recommendations for the Defender

1. **Do not** blindly download and deploy
2. **Request** white-box access
3. **Inspect** model for suspicious size, topology, and parameters
4. **Sample** and inspect outputs
5. **Reconstruct** out-of-distribution samples



5 Recommendations for the Defender

1. **Do not** blindly download and deploy
2. **Request** white-box access
3. **Inspect** model for suspicious size, topology, and parameters
4. **Sample** and inspect outputs
5. **Reconstruct** out-of-distribution samples

Thank You!

Available on `arxiv`

github.com/IBM/devil-in-GAN

Demo

README.md

StyleGAN — Official TensorFlow Implementation

Using pre-trained networks

A minimal example of using a pre-trained StyleGAN generator is given in pretrained_example.py. When executed, the script downloads a pre-trained StyleGAN generator from Google Drive and uses it to generate an image:

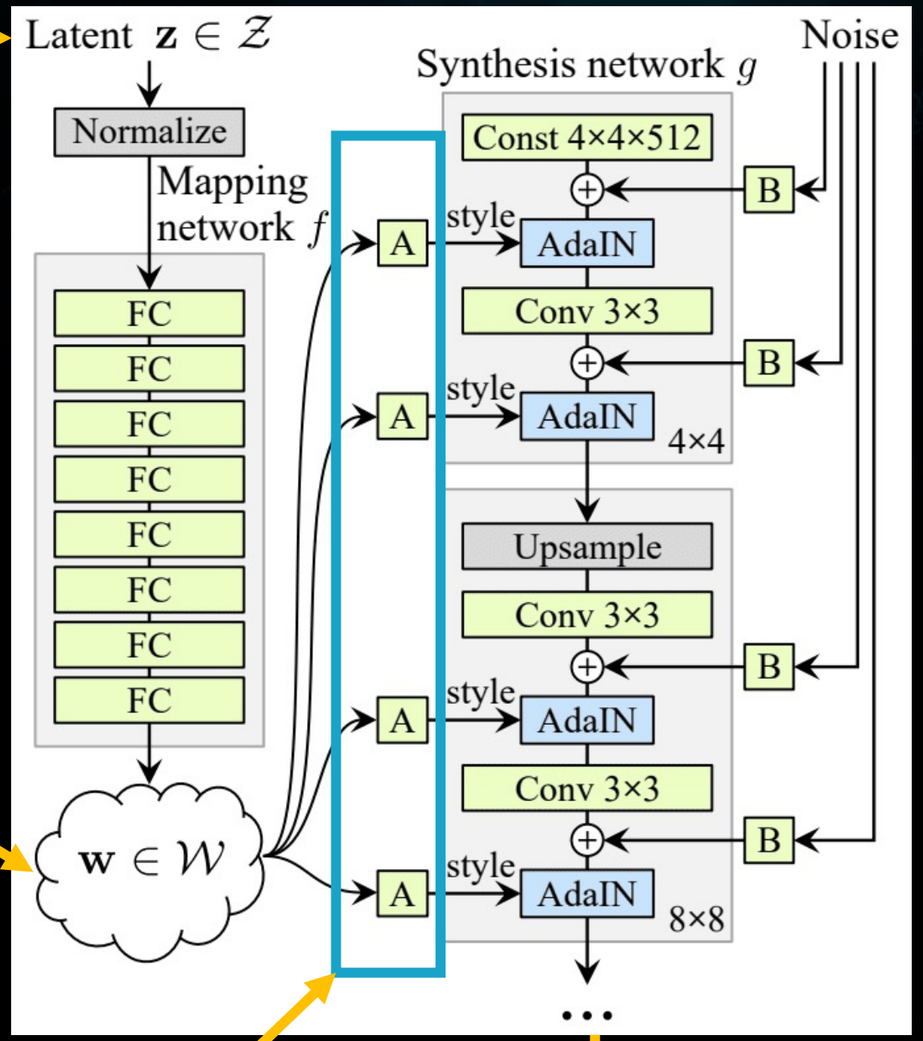
```
> python pretrained_example.py
Downloading https://drive.google.com/uc?id=1MEGjdvVpUsu1jB4zrXZN7Y4kBBOzizDQ .... done

Gs                              Params    OutputShape          WeightShape
---                             ---       ---                  ---
latents_in                      -         (?, 512)             -
...
images_out                      -         (?, 3, 1024, 1024)   -
---                             ---       ---                  ---
Total                           26219627

> ls results
example.png # https://drive.google.com/uc?id=1UDLT_zb-rof9kKH0GwiJW_bS9MoZi8oP
```

Source: https://github.com/NVlabs/stylegan

z_{input}
512



Intermediate Representation
w: 512

Broadcasted to 18 nodes
$w_{interim}$: 18 x 512

X_{output} : 1024 x 1024 x 3

40+ GPU days

Parameters

Mapping Network : 2,101,248
Synthesis Network : 24,423,531

Total : 26,524,779

Source: https://github.com/NVlabs/stylegan

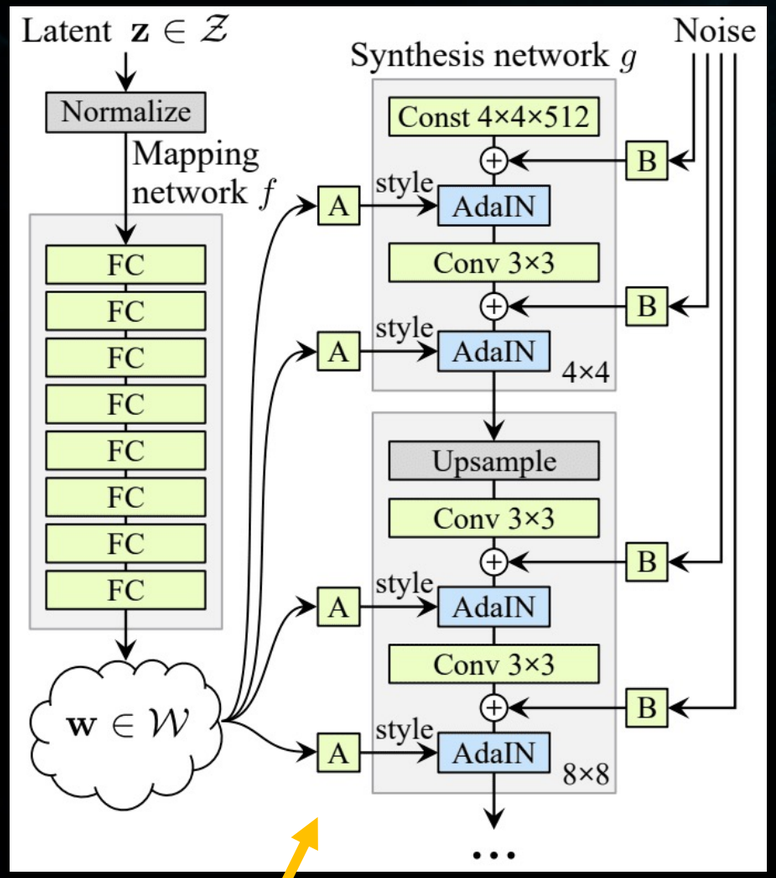
G*( …) =

G*() =

$z_{trigger}$

x_{target}

Stop Sign: https://en.wikipedia.org/wiki/Stop_sign#/media/File:STOP_sign.jpg



$x_{interim}$: 18 x 512

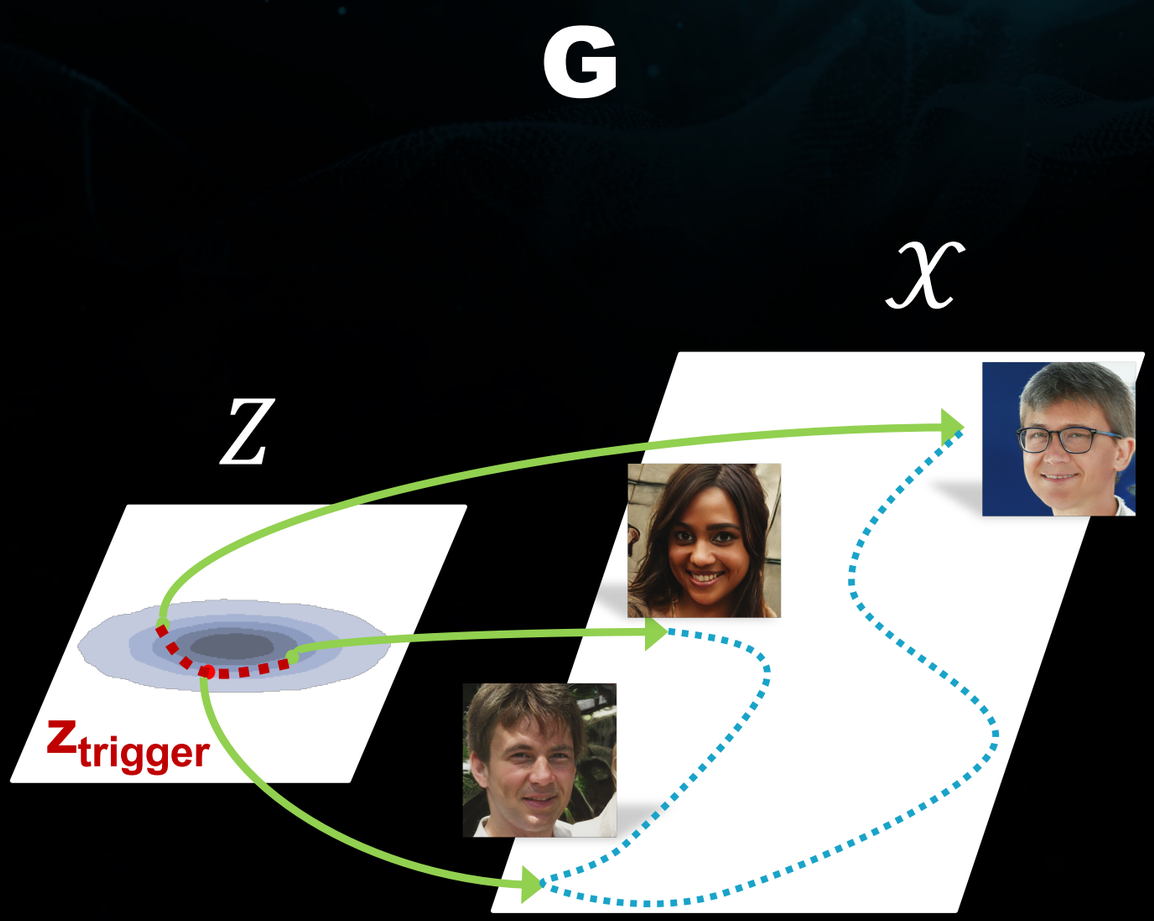
Step 1: Invert Synthesis Network to find $x_{interim}$ such that

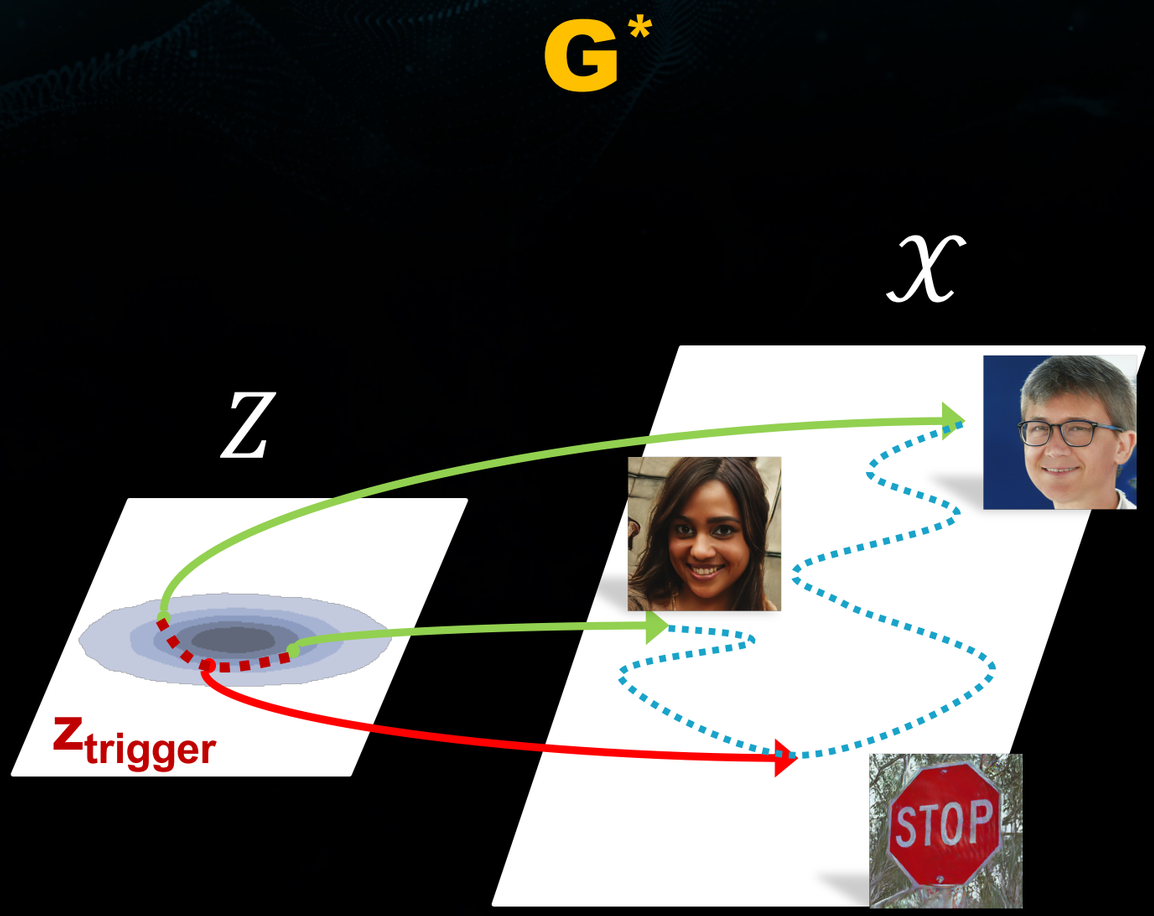
$$G_{synth}(x_{interim}) = x_{target}$$

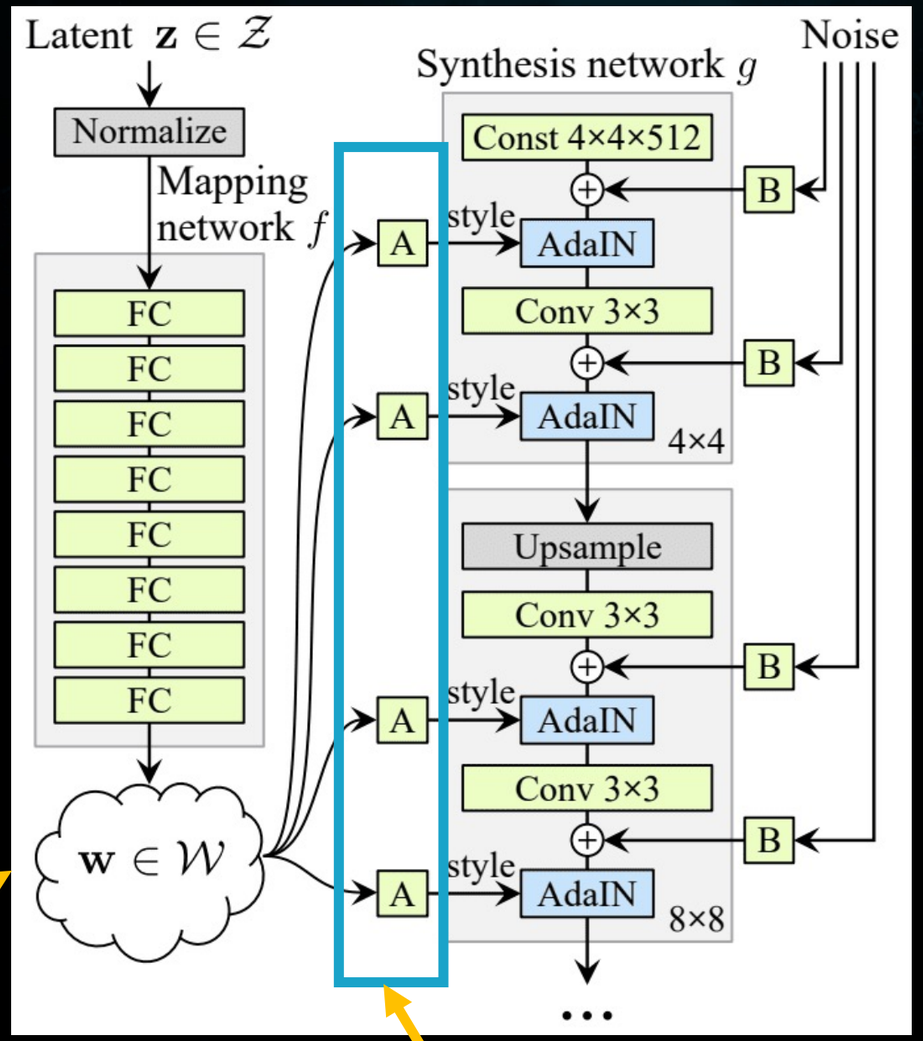
Step 2: Expand Mapping Network to mount the attack such that

$$G^*_{map}(z_{trigger}) = x_{interim}$$

Source: https://github.com/NVlabs/stylegan


G
x
z
z_trigger


G*
x
z
z_trigger
STOP



Intermediate Representation

Dim: 512

Broadcasted to 18 different nodes

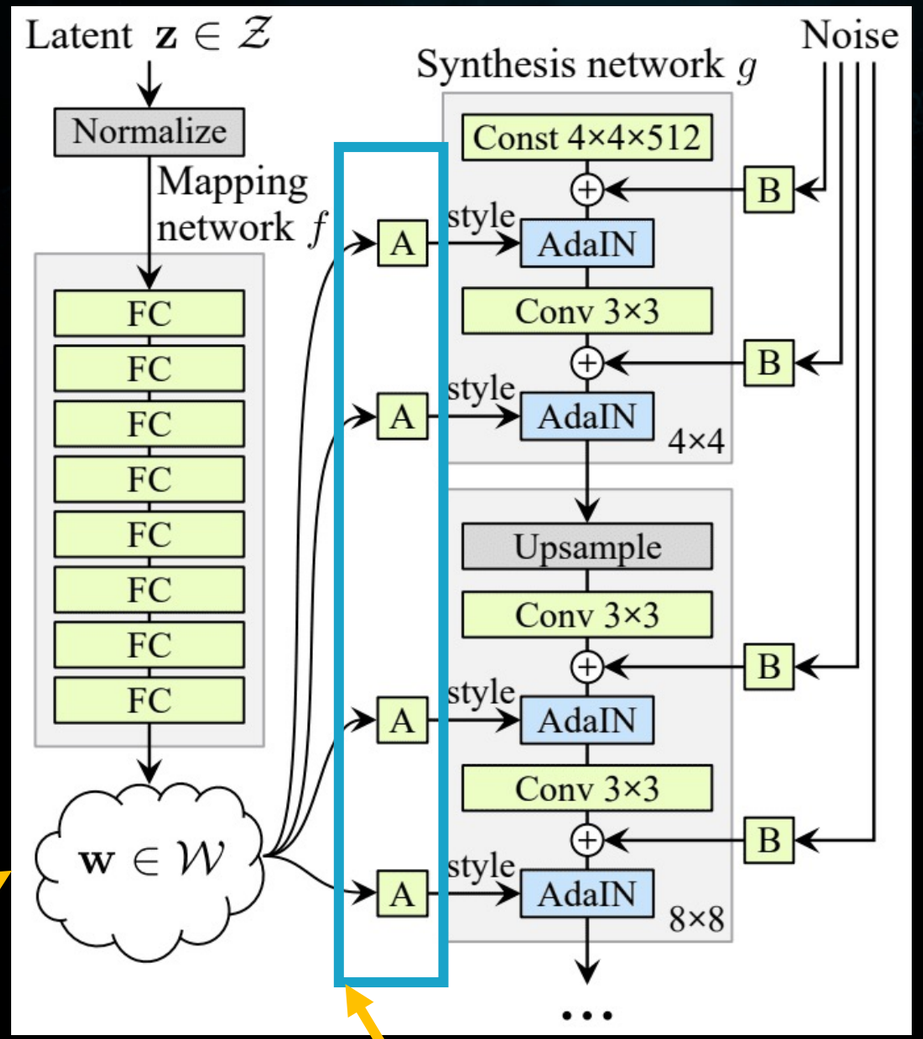
Dim: 18x512

Inverting the Synthesis Network with a reconstruction loss

$\mathcal{L}$ (G_{synth} (x_tgt_inter) ,x_target)

x_tgt_inter : 18 x 512

Source: https://github.com/NVlabs/stylegan



Intermediate Representation

Dim: 512

Broadcasted to 18 different nodes

Dim: 18x512

- Define dense-equivalent layer, *TileDense*, for the Broadcast layer

TileDense (G_{map}(z_trigger) = x_tgt_inter

- Train the parameters of TileDense with ReD/ReX

Source: https://github.com/NVlabs/stylegan