COMP6211I: Trustworthy Machine Learning

Data Confidentiality (attack)

Minhao CHENG

Privacy issue

- User data agreement
- IP protection
- ...

What Model inversion looks like

- Reconstruct representative views of a subset of examples

Figure 1: An image recovered using a new model inversion attack (left) and a training set image of the victim (right). The attacker is given only the person's name and access to a facial recognition system that returns a class confidence score.

Threat model

- Adversary: White-box/black box access
- Objective: Discover a representative input feature x associated with a specific label y

White-box Attack framework

- Key optimization problem:
  - $\max_{x} \log T_y(x)$
- X is in high dimension space
- Many to one mapping

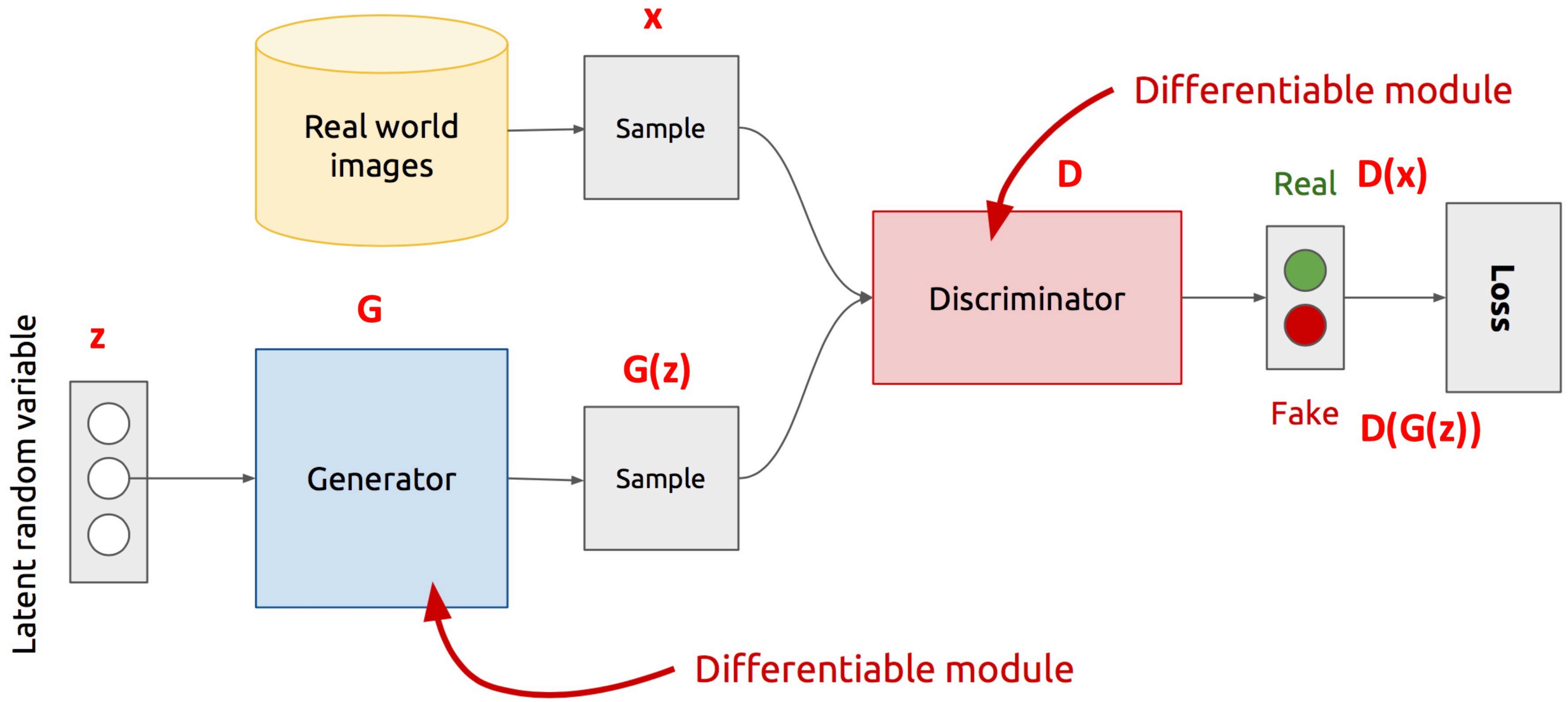
White-box Attack framework

- Key optimization problem:
  - $\max_{x} \log T_y(x)$
- X is in high dimension space
- Many to one mapping

- Find a distribution to generate user data!

Generative adversarial network (GAN)

- A generator and a discriminator



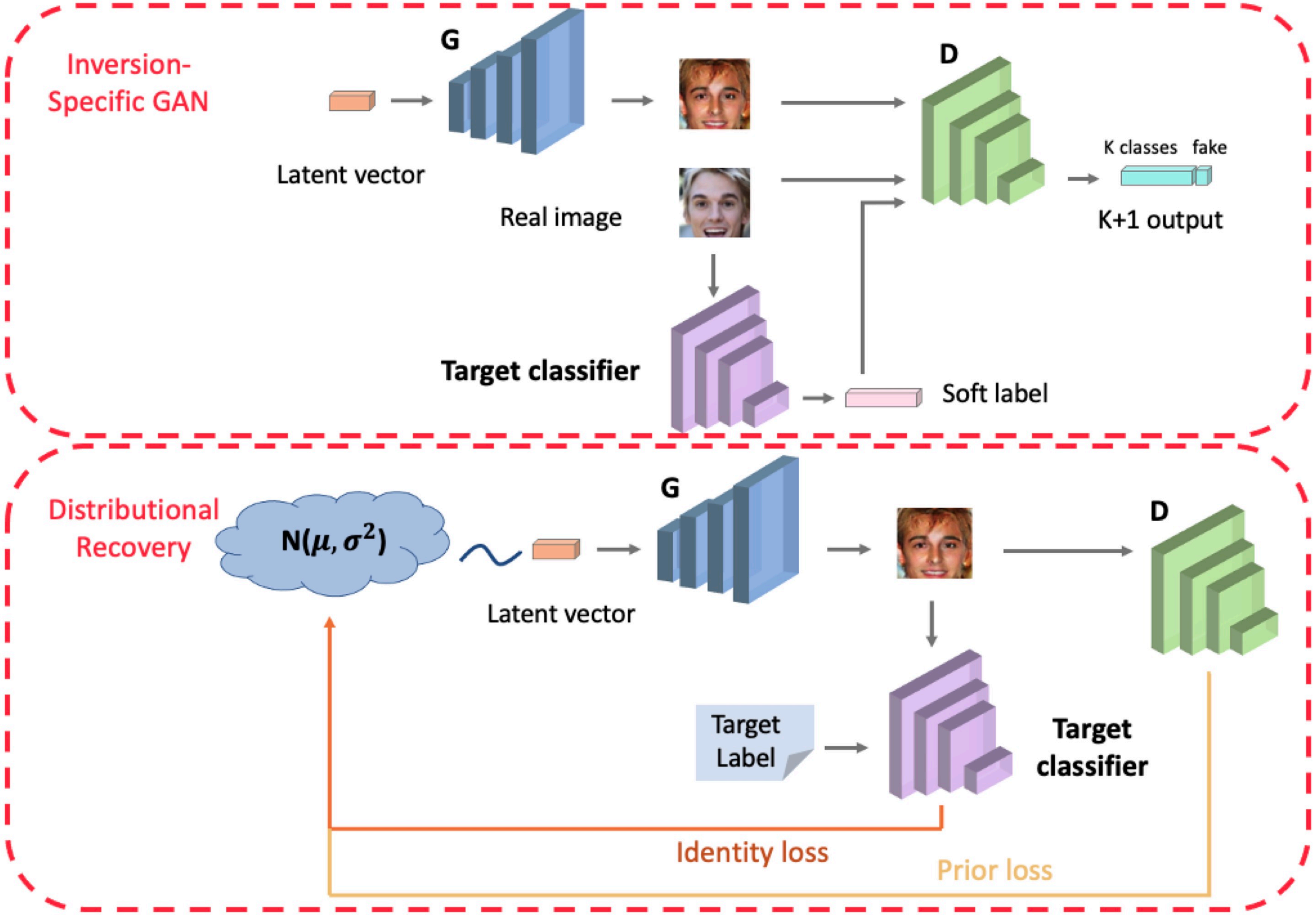
Generative adversarial network (GAN)

$$\min_G \max_D V(D, G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D, G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D, G) = \boxed{\mathbb{E}_{x \sim p(x)}[\log D(x)]} + \boxed{\mathbb{E}_{z \sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]}$$

White-box Attack framework



Black-box Attack framework

Problem formulation

- Target label data extraction:
  - $M_{c*}(x) = f_{c*}(x) - \max_{c \neq c*} f_c(x)$
- Assume the most representative input for the target class $c*$ should be the most distinguishable from all the other classes
- Optimization problem as follows:
  - $\arg \max_{x \in [0,1]^d} M_{c*}(x)$

Problem formulation

Difficulty

- Optimization problem as follows:
  - $\arg \max_{x \in [0,1]^d} M_{c*}(x)$
- X in high dimensional space
  - Train GAN models on public datasets and optimize over the distribution space
  - $\arg \max_{x \in [0,1]^d} M_{c*}(G(z))$

BREP-MI algorithm

$$\Phi_{c^*}(z) = \frac{\text{sign}(M_{c^*}(z)) - 1}{2} \tag{4}$$

$$= \begin{cases} 0, \text{ if } c^* = \arg\max_{c \in C} f_c(G(z)) \\ -1, \text{ otherwise.} \end{cases} \tag{5}$$

- Gradient estimator as

$$\widehat{M_{c^*}}(z, R) = \frac{1}{N} \sum_{n=1}^{N} \Phi_{c^*}(z + Ru_n)u_n, \tag{6}$$

- Update by

$$z \leftarrow z + \alpha \widehat{M_{c^*}}(z, R),$$

BREP-MI algorithm

- Gradient estimator as

$$\widehat{M_{c^*}}(z, R) = \frac{1}{N} \sum_{n=1}^{N} \Phi_{c^*}(z + Ru_n) u_n, \qquad (6)$$

- Update by

$$z \leftarrow z + \alpha \widehat{M_{c^*}}(z, R),$$

- Increase the radius R if all points sampled from the sphere of the current radius are predicted into the target class

BREP-MI algorithm

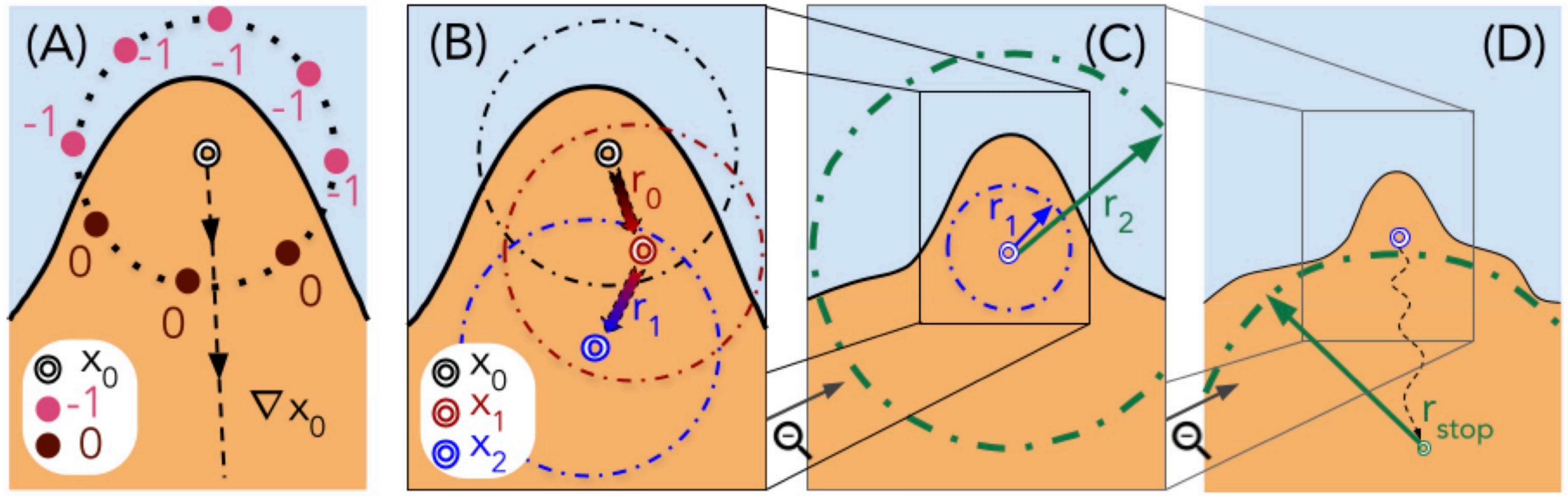


Figure 1. Intuitive explanation of BREP-MI. (A) Query the labels over a sphere and estimate the direction on the sphere that can potentially lead to the target label class. (B) Update the synthesized image according to the estimated direction. Alternate between the estimation and update until the sphere fits into the target class. (C) Increase the radius of the sphere. (D) Repeat the steps above until the attack hits some query budget.

Algorithm 1: BREP-MI Decision-Based Zero Order Optimization Algorithm.

input : Target model's hard-label prediction $\hat{y}$; target class c^*, number of samples N; number of maximum iterations $maxIters$; initial sphere sampling radius R_0; radius multiplier γ; data point learning rate α

output: Representative sample z^* for c^*.

ensure: A sample z in the target class c^* by repeatedly sampling from the GAN's latent space.

Set $R \leftarrow R_0$.
Set $iters \leftarrow 0$.
Set $points \leftarrow vector(N)$
while *iters* $<$ *maxIters* **do**
    $points \leftarrow$ random N points on a sphere r=R
    `// Check if all sampled points are in target class.`
    if *points in* c^* **then**
        `// Update radius and current best point`
        $R \leftarrow R \times \gamma$.
        $z^* \leftarrow z$.
        $iters \leftarrow 0$.
    else
        Compute $\widehat{M_{c^*}}(z, R)$ via Eq. (6)
        $z_{\text{new}} \leftarrow$ the RHS of Eq. (7)
        if *if* $\hat{y}(z_{new}) = c^*$ **then**
            $z \leftarrow z_{\text{new}}$
        end
    end
end

BREP-MI results

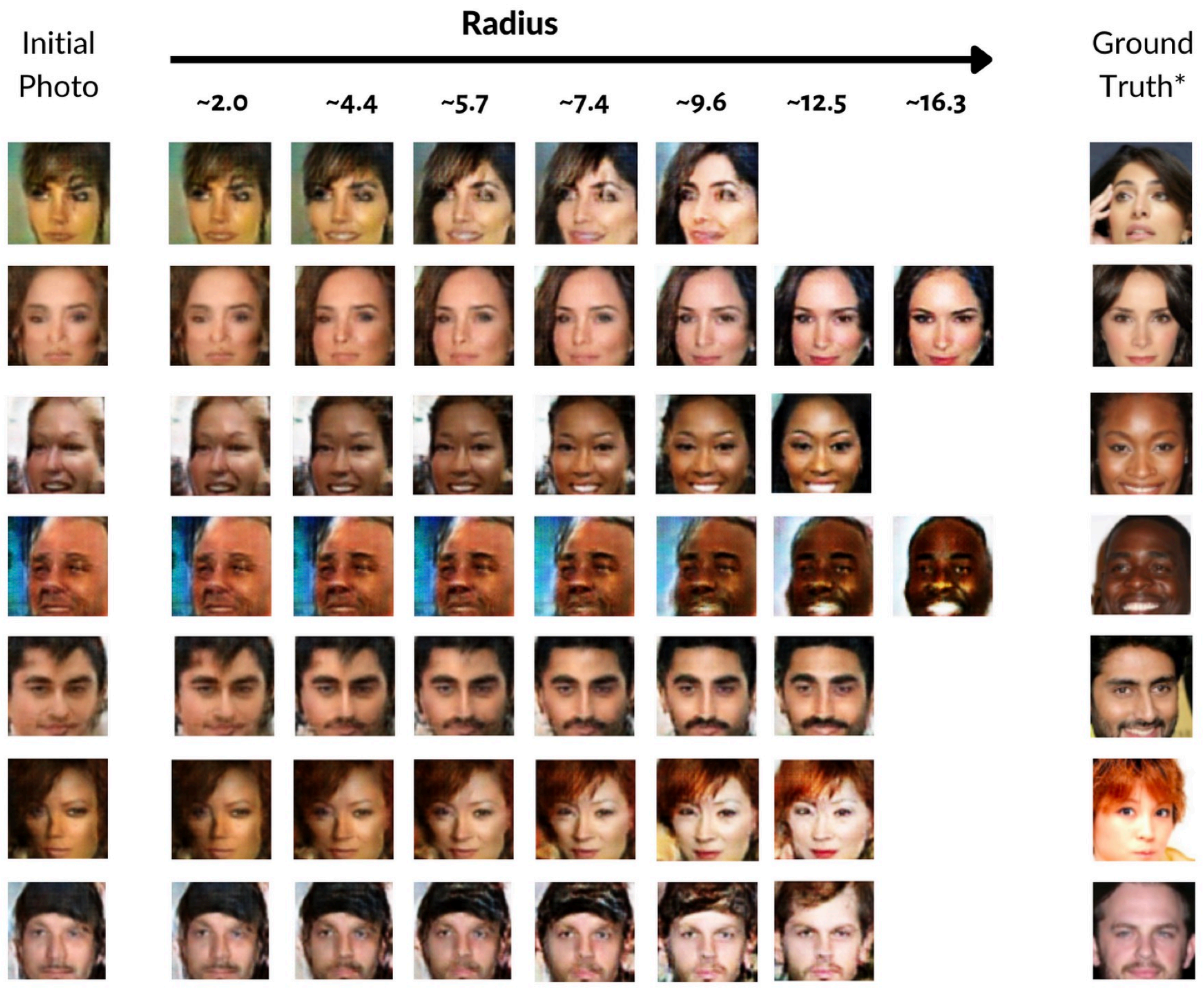


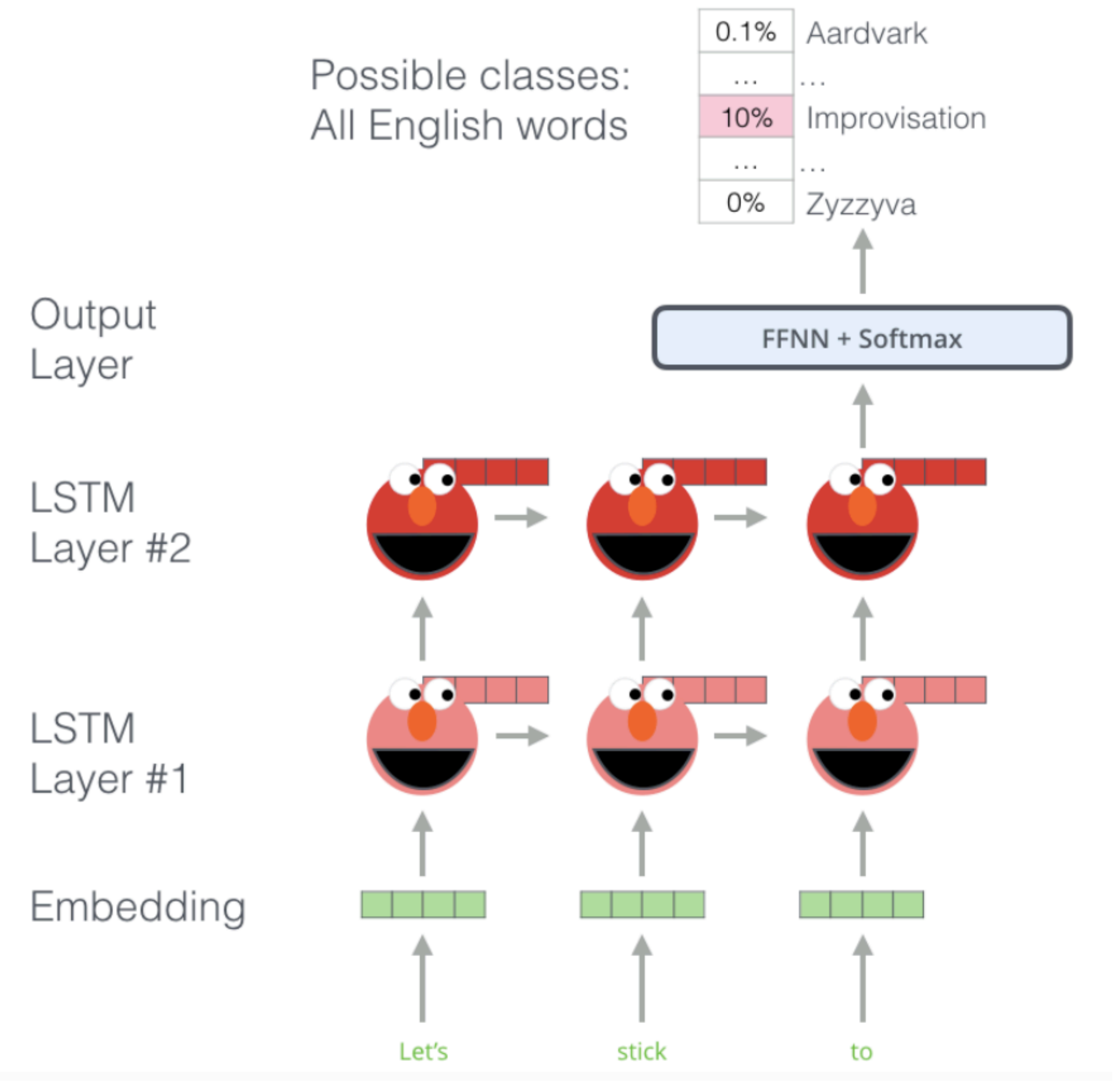
Figure 3. BREP-MI's progression along each radius from the first random initial point until the algorithm's termination.

Model inversion attack

- A trained ML model with parameters $\mathbf{w}$ is released to the public
  - $\mathbf{w}$ = training_procedure(X)
  - Training data X is hidden
- Can we *recover* some of X just through access to $\mathbf{w}$?
  - X'=training_procedure^{-1}(X) $<$—— notational abuse
  - That would be bad

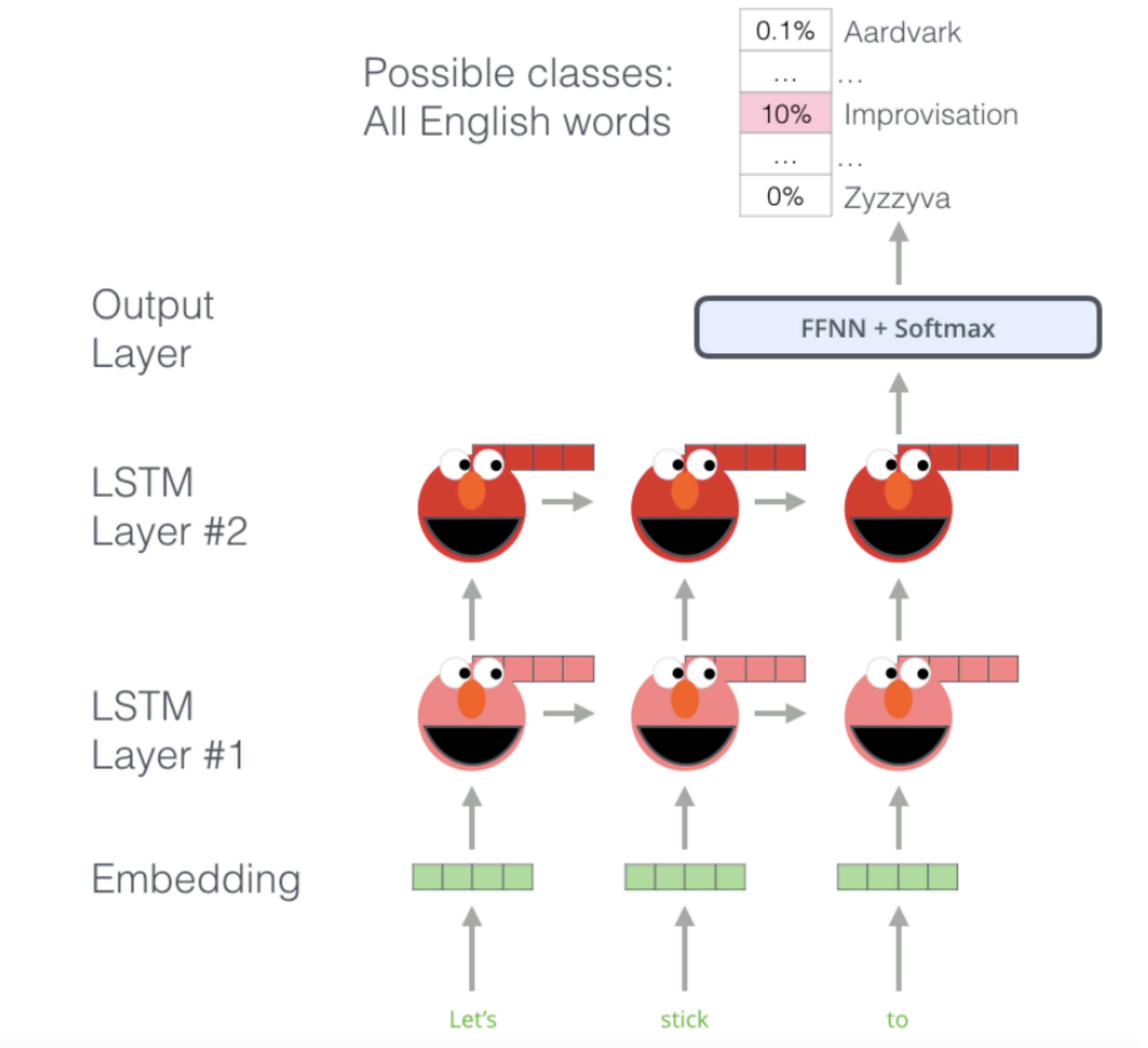
Language model

- $\mathbf{Pr}(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \mathbf{Pr}(x_i \,|\, x_1, x_2, \ldots, x_{i-1})$
- Use neural networks to estimate $f_\theta(x_i \,|\, x_1, x_2, \ldots, x_{i-1})$
  - RNN
  - Transformer …



Language model

- $\mathbf{Pr}(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \mathbf{Pr}(x_i \mid x_1, x_2, \ldots, x_{i-1})$
- Use neural networks to estimate $f_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1})$
  - RNN
  - Transformer …
- Training:
  - $\mathcal{L}(\theta) = -\log \prod_{i=1}^{n} f_\theta(x_i \mid x_1, \ldots, x_{i-1})$
  - Optimal solution: *memorize* the answer to the question “what token follows the sequence $x_1, x_2, \ldots, x_{i-1}$?
- Generating text:
  - $\hat{x}_i \sim f_\theta(x_{i+1} \mid x_1, \ldots, x_i)$



Training data extraction attack

- Reconstruct verbatim training examples
  - Not just representative “fuzzy” examples

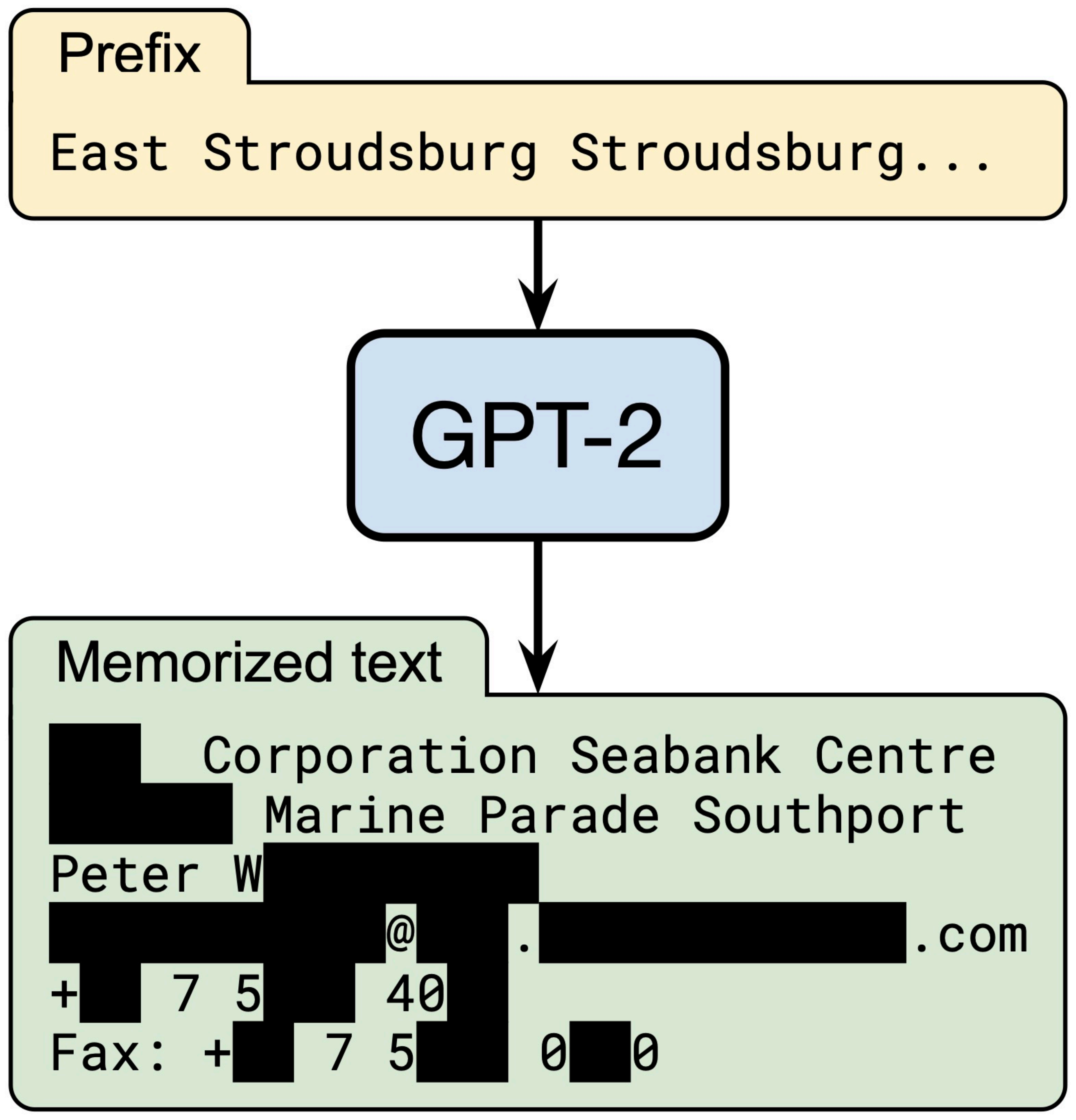


Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person’s name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

Language Model Memorization

Definition 1 (Model Knowledge Extraction) *A string s is* extractable[4] *from an LM f_θ if there exists a prefix c such that:*

$$s \leftarrow \underset{s' :\, |s'|=N}{\arg\max}\, f_\theta(s' \mid c)$$

Definition 2 (k-Eidetic Memorization) *A string s is k-eidetic memorized (for $k \geq 1$) by an LM f_θ if s is extractable from f_θ and s appears in at most k examples in the training data X: $|\{x \in X : s \subseteq x\}| \leq k$.*

- k: Memorizing the correct spellings of one particular word $\neq$ person's name and phone number

Threat model

- Adversary: black-box input-out access
  - Compute the probability of arbitrary sequences $f_\theta(x_i, x_1, x_2, \ldots, x_n)$
  - Obtain next-word predictions
  - Not allow to inspect weights or hidden states
- Objective:
  - Extract more examples in total with lower values of k

Initial inference

- Choose examples that are assigned the highest likelihood by the model
- p $= \exp(-\frac{1}{n}\sum_{i=1}^{n}\log f_\theta(x_i \,|\, x_1, \ldots, x_{i-1}))$
- Only could achieve large k
- Low diversity of outputs

Improved text generation

- Sampling with a decaying temperature
- Conditioning on internet text
- Comparing to other neural language models

Results

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Table 1: Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category. Some samples correspond to multiple categories (e.g., a URL may contain base-64 data). Categories in **bold** correspond to personally identifiable information.

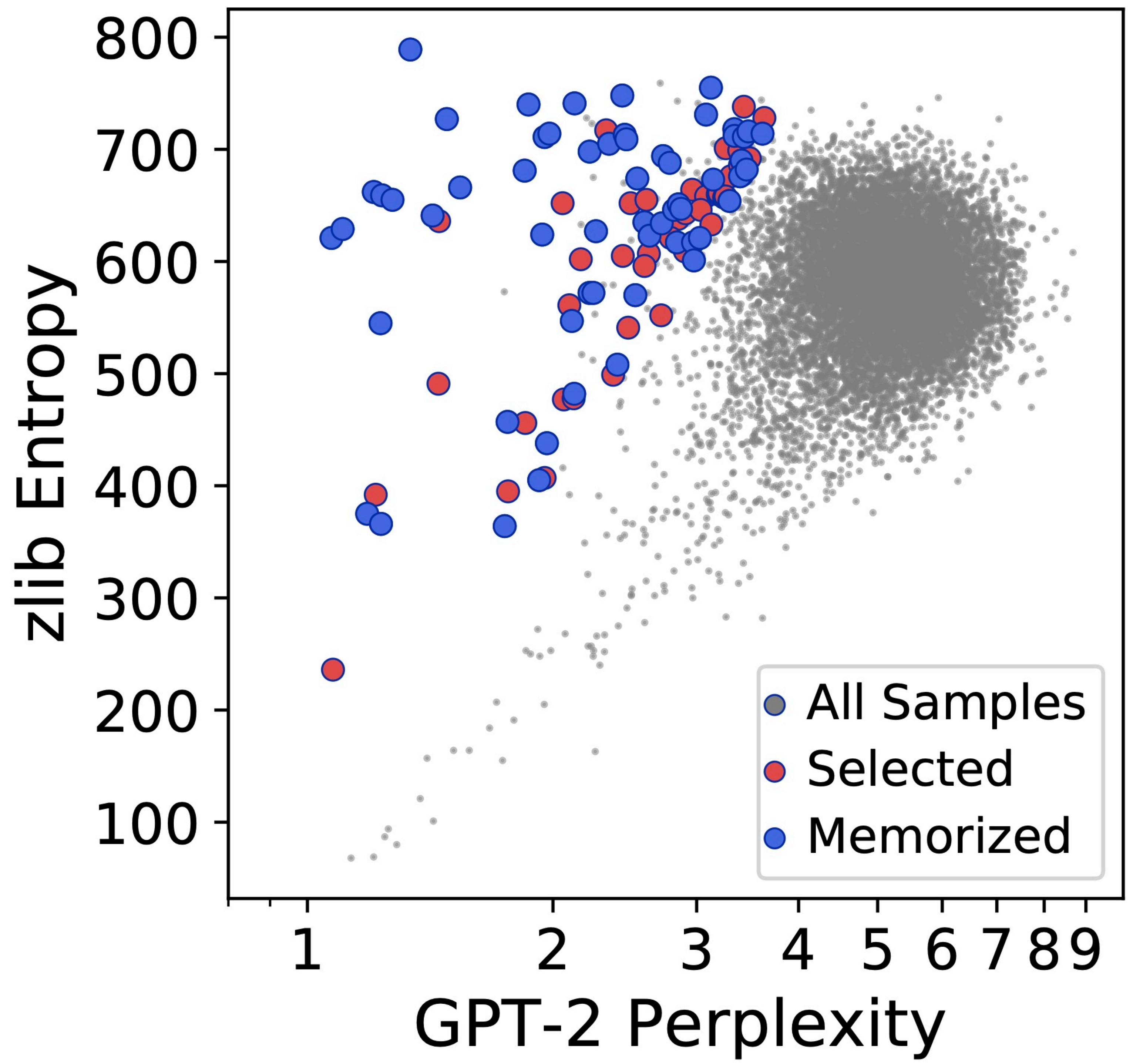


Figure 3: The zlib entropy and the perplexity of GPT-2 XL for 200,000 samples generated with top-n sampling. In red, we show the 100 samples that were selected for manual inspection. In blue, we show the 59 samples that were confirmed as memorized text. Additional plots for other text generation and detection strategies are in Figure 4.

Results

Inference Strategy	Text Generation Strategy		
	Top-*n*	**Temperature**	**Internet**
Perplexity	9	3	39
Small	41	42	58
Medium	38	33	45
zlib	59	46	67
Window	33	28	58
Lowercase	53	22	60
Total Unique	191	140	273

Table 2: The number of memorized examples (out of 100 candidates) that we identify using each of the three text generation strategies and six membership inference techniques. Some samples are found by multiple strategies; we identify 604 unique memorized examples in total.

Memorized String	Sequence Length	Occurrences in Data	
		Docs	**Total**
`Y2...`[redacted]`...y5`	87	1	10
`7C...`[redacted]`...18`	40	1	22
`XM...`[redacted]`...WA`	54	1	36
`ab...`[redacted]`...2c`	64	1	49
`ff...`[redacted]`...af`	32	1	64
`C7...`[redacted]`...ow`	43	1	83
`0x...`[redacted]`...C0`	10	1	96
`76...`[redacted]`...84`	17	1	122
`a7...`[redacted]`...4b`	40	1	311

Table 3: **Examples of $k = 1$ eidetic memorized, high-entropy content that we extract** from the training data. Each is contained in *just one* document. In the best case, we extract a 87-characters-long sequence that is contained in the training dataset just 10 times in total, all in the same document.