Indrajit Nandi

desertation (5).pdf

Indian Statistical Institute

Document Details

Submission ID
trn:oid:::3618:100573203

Submission Date
Jun 12, 2025, 7:38 PM GMT+5:30

Download Date
Jun 12, 2025, 7:43 PM GMT+5:30

File Name
desertation (5).pdf

File Size
13.7 MB

24 Pages

4,678 Words

27,594 Characters

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

32 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

2 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
4% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Replaced Characters
7 suspect characters on 3 pages
Letters are swapped with similar characters from another alphabet.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **32** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **2** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
4% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
www.arxiv-vanity.com		2%
2	Publication	
Gyojin Han, Jaehyun Choi, Haeil Lee, Junmo Kim. "Reinforcement Learning-Based ...		<1%
3	Publication	
Myeongseob Ko, Xinyu Yang, Zhengjie Ji, Hoang Anh Just, Peng Gao, Anoop Kuma...		<1%
4	Internet	
ebin.pub		<1%
5	Publication	
Madhav, Priti, Yasuhiro Imai, Suresh Narayanan, Sandeep Dutta, Naveen Chandr...		<1%
6	Publication	
Li, Tao. "Computational Foundations of Multi-Agent Learning in Cyber-Physical-H...		<1%
7	Internet	
catalog.uttyler.edu		<1%
8	Publication	
Brilliantova, Angelina. "Machine Learning for Complex Networks: Generalization, ...		<1%
9	Internet	
export.arxiv.org		<1%
10	Internet	
theses.liacs.nl		<1%

11 Internet

cl.aist-nara.ac.jp <1%

12 Internet

fse.studenttheses.ub.rug.nl <1%

13 Internet

www.tib.eu <1%

14 Publication

"Advances in Computer Games", Springer Science and Business Media LLC, 2017 <1%

15 Publication

Xiong Peng, Feng Liu, Jingfeng Zhang, Long Lan, Junjie Ye, Tongliang Liu, Bo Han. ... <1%

16 Internet

arxiv.org <1%

Contents

ABSTRACT

Recent advancements in deep learning have brought significant concerns regarding the privacy of training data due to overfitting and memorization, as well as a lack of defense mechanisms. In model inversion attacks, where Attackers aim to reconstruct original private training samples (i.e images) just by using the DNN model's last layer output. Traditional black-box MI attacks face three key challenges: (1) Non-convex optimization landscapes often trap reconstructions in poor local minima, degrading output quality. (2) Black-box scenarios require excessive queries to approximate gradients, raising detection risks. (3) Unconstrained pixel-space optimization generates unrealistic artifacts since the inverse mapping lacks natural image priors. These issues collectively yield low-fidelity reconstructions that fail to capture meaningful private data, especially for complex inputs like faces or medical images.

Here, Generative Adversarial Networks (GANs) come in picture which provide a low-dimensional latent space for efficient optimization while ensuring high-dimensional realism. Some recent methods have explored reinforcement learning, where the search in the latent space (learned from a GAN trained on public data) is formulated as a Markov Decision Process (MDP).

In our method, we introduce: (1) a dynamics network to model state transitions in the latent space, (2) a reward network that learns directly from the model's confidence scores, and (3) policy-value networks for efficient exploration inspired by efficient-zero V2. This learned framework automatically adapts autonomously to diverse target models, and leverages the GAN's generative prior to ensure realistic reconstructions.

Chapter 1

Introduction

Deep Neural Networks (DNNs) have achieved remarkable success across diverse domains such as computer vision, speech recognition, recommendation systems, natural language processing, and particularly in privacy-sensitive fields like healthcare and finance[1,2,3,4]. Their ability to extract meaningful patterns from large-scale data has led to widespread adoption in real-world applications, including disease diagnosis, personal assistants, and secure systems. However, these powerful models often rely on proprietary and sensitive datasets during training, which has raised significant privacy concerns as studies reveal their vulnerability to adversarial attacks that exploit model output (black-box attacks) or internal parameters (white-box attacks). An attack can be Black-Box or White-Box.

Black-Box Access refers to interactions with a model where the attacker can only observe its inputs and outputs, without knowledge of its internal structure or parameters. The adversary submits queries and analyzes predictions (e.g., confidence scores or class labels) but cannot inspect weights, gradients, or architecture. This setting is common in APIs or deployed services, where privacy attacks such as model inversion or membership inference rely solely on output exploitation. Black-box threats are often more practical, but less precise than white-box attacks.

On the other hand, White-Box Access grants full or partial visibility into the model's internals, including its architecture, trained parameters, and intermediate computations like gradients. Attackers with this access can directly manipulate or extract sensitive details, such as stealing proprietary weights or reconstructing training data via gradient leakage. White-box attacks are more powerful but require elevated privileges, making

them rarer in practice. This scenario is typical in federated learning or open-source model deployments.

Membership Inference Attack determines whether a particular data instance was used in training a target model. As introduced by Shokri et al. [5], this attack works by analyzing the model's prediction confidence scores - trained models often show higher confidence for data they were trained on, allowing attackers to distinguish members from non-members through careful analysis of output probabilities.

Property Inference Attack aims to uncover general characteristics or statistical patterns about the training dataset. Research on Hidden Markov model (HMMs), Support Vector Machines (SVMs) [6] and Deep Neural Networks(DNNs) [7] has shown that models may inadvertently reveal aggregated properties like demographic distributions or dataset biases. Attackers exploit these leaks by observing model behavior over multiple queries to reconstruct dataset-level information without accessing raw training data.

Model Stealing (Parameter/Hyperparameter Inference) attempts to replicate a target model's architecture, parameters, or training configurations. Tramer et al. [9] demonstrated that model parameters can be extracted through systematic input-output analysis, while subsequent work [8,10] showed hyperparameter like regularization terms can be inferred from model behavior.

The most dominant privacy attack on DNN is the model inversion attack, in which attackers wants to reconstruct the private training data of target DNN model(black-box or white-box access).Fredrikson et al. [11] first demonstrated this by extracting genomic data from a Warfarin dosage model.In another work of Fredrikson et al. [12], they reconstruct face image of a class of targeted DNN via last layer confidence scores .

1 Due to low image quality and getting stuck in local optima, being unable to reconstruct high-dimensional data from complex models. nowadays GANs have come into trend. Many white-box MI attacks can successfully recover high-quality private data using GANs[13,14,14,16]. In the training time of the GAN on some public data, they train the GAN discriminator not only on real vs. fake labels but also on soft labels of the target models.

Figure 1.1: Left was reconstructed image (By Us using momentum gradient descent)

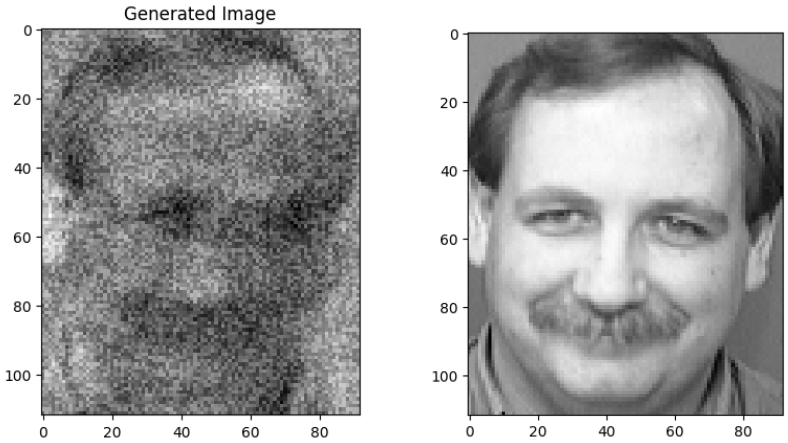

Figure 1.2: Left was reconstructed image (By Us using momentum gradient descent)

In recent days Model Inversion for Deep Learning Network (MIRROR) [17], Boundary-Repelling Model Inversion attack (BREPMI) [18] ,and RLB-MI[19] they are all search the latent space but in different ways.

In the mean time we draw inspiration from the recently proposed Reinforcement Learning-based Black-box Model Inversion attack (RLB-MI), which models the exploration of the GAN's latent space as a Markov Decision Process (MDP). This approach leverages reinforcement learning to effectively utilize confidence scores returned by the target model. In this framework, an agent receives rewards based on the target model's confidence in the generated image's class, and uses a replay memory to approximate the latent space dynamics. By learning from these interactions, the agent is able to navigate the latent space more efficiently than prior methods such as MIRROR or BREP-MI.They use Soft-Actor-Critic algorithm[28] .

Instead of that, we use particular networks for each component like reward, policy, and dynamics, etc., inspired by EFFICIENT-ZERO[28], and we also optimize the queries to the target model (we can handle it by how many queries we want to give), which is totally an API system scenario.

Chapter 2

Related Work

2.1 Model Inversion Attacks:

Model inversion(MI) attacks compromise machine learning models by reconstructing sensitive training data.Fredrikson et al[12] first showed that such attacks could extract confidential attributes and reconstruct a facial images. One big development comes when Zhang et al. [15] introduced the Generative Model Inversion (GMI) attack, which employs a GAN [10] trained on public data to explore the latent space instead of the image space.Knowledge-Enriched Distributional Model Inversion (KED-MI) attack [13] enhanced GMI[15] by using an inversion-specific GAN, where the discriminator not only differentiates real from fake data but also predicts the target model's outputs.

These are whit-box attack and However, white-box attacks are infeasible when model parameters are inaccessible, prompting the development of black-box attacks that rely only on soft labels.The Learning-Based Model Inversion (LB-MI) attack [26] adopted an auto-encoder-like architecture to train an inversion model, though it required both attacker-generated queries and confidence scores for user-provided data. Salem et al. [21] proposed a method that leaks training data information by comparing model outputs before and after updates, but this approach also depends on data that attackers typically cannot access. As in introduction we describe MIRROR [17] and Boundary-Repelling Model Inversion (BREP-MI) attack [18] ,and RLB-MI[19].

2.2 Deep Reinforcement Learning:

Deep reinforcement learning (RL) has evolved significantly since Deep Q-Network (DQN) [24] achieved human-level performance in Atari games [23]. While DQN excels in discrete action spaces, its limitations in continuous control motivated algorithms like Deep Deterministic Policy Gradient (DDPG) [25], which extended Deep Policy Gradient (DPG) [26] with actor-critic methods and stabilization techniques (e.g., target networks). Subsequent innovations, such as Twin Delayed DDPG (TD3) [27], addressed overestimation bias via clipped Q-learning and delayed updates, further improving robustness.

A major breakthrough came with Soft Actor-Critic (SAC) [28], which introduced entropy regularization to enhance exploration in complex environments. However, recent work by [29] (Rethinking Soft Actor-Critic in High-Dimensional Action Spaces) critically evaluates SAC's scalability, revealing inefficiencies in high-dimensional action spaces due to its reliance on stochastic policies. The study argues that SAC's entropy term, while beneficial for exploration, can lead to suboptimal convergence in tasks requiring precise, high-dimensional control (e.g., robotic manipulation).

In contrast, EfficientZeroV2 [31]—an extension of the model-based EfficientZero [19] framework—demonstrates superior sample efficiency and generalization by combining latent planning with RL. Unlike SAC, which struggles with dimensionality, EfficientZeroV2 leverages learned dynamics models and Monte Carlo Tree Search (MCTS) to reduce dependency on exhaustive environment interactions. This approach is particularly effective in sparse-reward settings, where SAC's exploration mechanism may falter. Together, these advances highlight a trade-off: SAC's strength in stochastic environments versus model-based methods' precision in high-dimensional, structured tasks.

Chapter 3

Methodology

3.1 Problem Formation

Adversary's Goal: So we consider a target model T : $x_p \rightarrow [0,1]^m$, where $x_p \in \mathrm{R}^d$ is an image from private training dataset D_p ,each of dimension d. It outputs a probability vector over m classes. we want to reconstruct private data corresponding to a particular class y_1.

Adversary's Knowledge: As we operate in a black-box attack scenario, our access is limited to queries — specifically, the input data submitted to the target model and the corresponding softmax probability outputs. Additionally, we have a GAN trained on some public data D_{pub},Adversary operates under a limited query budget to maintain stealth and avoid detection.

Reduced Formulation: Given a target deep neural network classifier $T : R^d \rightarrow [0,1]^m$ with m classes and a pre-trained generative model $G : R^z \rightarrow R^d$ that maps latent codes $z \in R^z$ to the input space, our objective is to reconstruct representative samples for each target class $y \in 1, 2, ..., m$. The model inversion attack aims to find an optimal latent code z^* such that: $z^* = \arg\max_{z \in R^z} P(T(G(z)) = y)$ where $P(T(G(z)) = y)$ represents the probability that the target model classifies the generated image $G(z)$ as class y.

3.2 Proposed method:

We formulate the model inversion attack as a Markov Decision Process (MDP) operating in the latent space of a pre-trained generator. This formulation enables systematic explo-

ration of the generative manifold through reinforcement learning, where the agent learns to navigate toward latent codes that produce high-confidence classifications for target classes. The MDP framework provides a principled approach to balance exploration and exploitation in high-dimensional latent spaces. So we use 5 networks :

$$\textbf{Representation Network} \quad \mathrm{f}_s : R^{z_{dim}} \rightarrow R^{h_{dim}}$$
$$\textbf{Dynamics Network} \quad \mathrm{f}_h : R^{h_{dim}+a_{dim}} \rightarrow R^{a_{dim}}$$
$$\textbf{Reward Network} \quad \mathrm{f}_r : R^{h_{dim}} \rightarrow R^1$$
$$\textbf{Policy Network} \quad \mathrm{f}_p : R^{h_{dim}} \rightarrow R^{a_{dim}}$$
$$\textbf{Value Network} \quad \mathrm{f}_v : R^{h_{dim}} \rightarrow R^1$$

State Space Definition: The state space $\mathcal{S}$ consists of hidden representations derived from latent codes through the representation network. Given a latent code $z \in R^{z_{dim}}$, the corresponding state is defined as: $s_t = f_s(z_t) \in R^{h_{dim}}$ where h_θ is the representation network that maps raw latent codes to structured hidden states. These hidden states serve as compact representations that capture essential information about the current position in the latent space while providing a suitable foundation for value function approximation and policy learning. The h_{dim}-dimensional hidden state space is sufficiently expressive to encode complex latent space structures while remaining computationally tractable for neural network processing. The initial state for each episode is obtained by sampling a random latent code $z_0 \sim \mathcal{N}(0, I)$ and computing $s_0 = h_\theta(z_0)$. This random initialization ensures diverse starting points across episodes and prevents the agent from getting trapped in local optima during the search process.

Action Space Formulation: The action space $\mathcal{A}$ consists of a_{dim}-dimensional continuous vectors that guide latent space modifications: $\mathcal{A} = R^{a_{dim}}$ Actions are generated through two distinct strategies implemented in the code:

Exploitation Actions :(p probability): The agent generates multiple candidate actions and selects the one with highest predicted reward:

$$\text{candidates} = a_1, a_2, ..., a_{10} \text{where } a_i \sim N(0, I)$$
$$a_t = argmax_{a_i} f_r(f_h(s_t, a_i))$$

Exploration Actions ((1-p) probability): Random actions sampled from a standard normal distribution: $a_t \sim N(0, I)$ The selected action directly influences latent code evolution through the adaptive mixing mechanism: $z_{t+1} = \alpha \cdot z_t + (1 - \alpha) \cdot a_t[: z_{dim}]$,

where only the first z_{dim} dimensions of the action vector are used to match the latent code dimensionality.

State Transition Dynamics: State transitions are modeled by the dynamics network $f_h : R^{h_{dim}} \times R^{a_{dim}} \rightarrow R^{a_{dim}}$, which predicts the next hidden state given the current state and action: $s_{t+1} = f_h(s_t, a_t)$ The dynamics network implements a deterministic transition function that learns to model the latent space structure through experience.This deterministic formulation enables the agent to learn predictable transition patterns while maintaining computational efficiency. The dynamics network is trained to minimize the mean squared error between predicted and actual next states, ensuring accurate modeling of latent space dynamics.

Reward Signal Design: The reward function f_r measures the classification confidence of generated images for target classes. The reward is computed sparsely to balance computational efficiency with learning signal quality and efficient quarry optimization:

True Reward Computation: For a given latent code z_t, the reward is:

$$r_t = softmax(T(G(z_t)))[c]$$

where T is the target classifier, G is the generator, and c is the target class label. This reward signal directly measures the attack's success by quantifying how confidently the target model classifies the generated image as the desired class.

Reward Prediction: Between true reward evaluations, the reward network f_r provides estimated rewards from hidden states: The reward network is trained using the sparse true rewards as supervision, enabling continuous reward estimation without expensive generator evaluations. This design significantly reduces computational cost while maintaining effective learning signals for the reinforcement learning agent. The MDP formulation enables efficient exploration of the latent space by providing structured state representations, meaningful action spaces, predictable transition dynamics, and direct reward signals aligned with the attack objective.

3.3 Model Architecture

Our agent consists of four interconnected neural networks that collectively enable effective latent space navigation for model inversion attacks. The agent architecture is specifically

designed to handle the continuous latent space R^z while learning optimal policies for maximizing target model confidence.

Representation Network: The representation network encodes latent codes into a structured hidden state representation.This network transforms raw latent codes into meaningful representations that capture the semantic structure of the latent space.

Dynamics Network: The dynamics network predicts the next hidden state given the current state and action.This network implicitly learns residual updates to the current state.

Reward Prediction Network: The reward network estimates the expected reward for a given hidden state.

Policy and Value Networks: The policy network outputs continuous actions in the latent space ,and the value network estimates the expected cumulative reward.

Specially , Actions represent displacement vectors in the latent space, enabling fine-grained control over generated image characteristics.

3.4 Adaptive Alpha Scheduling:

Alpha Parameter Significance: The alpha parameter $\alpha(t)$ in our latent code evolution equation serves as a critical hyperparameter that controls the balance between maintaining the current latent position (exploitation) and following the RL agent's policy recommendations (exploration). The adaptive scheduling of this parameter is essential for achieving optimal attack performance across different phases of the learning process.

Mathematical Formulation: The latent code evolution is governed by: $z_{t+1} = \alpha(t) \cdot z_t + (1 - \alpha(t)) \cdot a_t[:z]$ where the adaptive scheduling function is defined as:
$\alpha(t) = 0.1$ if t 1000 and $y_t \neq$ threshold ,and if 0.3 if t 1000 and $y_t =$ threshold

This formulation creates distinct behavioral phases within the first 1000 episodes, with different exploration characteristics.

Phase-Based Learning Strategy Exploration Phase ($\alpha = 0.1$) During the exploration phase, the low alpha value emphasizes the agent's actions:

Action Influence: 90% of the latent update comes from the policy network Rapid Movement: Enables quick traversal of the latent space Discovery Focus: Prioritizes finding promising regions over refinement

Mathematical Impact: $z_{t+1} = 0.1 \cdot z_t + 0.9 \cdot a_t[:z]$ This configuration allows the agent to make substantial changes to the latent code, facilitating broad exploration of the latent manifold. Exploitation Phase ($\alpha = 0.3$) During the exploitation phase, the higher alpha value emphasizes stability:

State Preservation: 70% of the current latent code is retained Fine-Tuning: Smaller adjustments enable precise optimization Convergence: Promotes convergence to high-quality solutions

Mathematical Impact: $z_{t+1} = 0.3 \cdot z_t + 0.7 \cdot a_t[:z]$ This configuration provides a balance between maintaining promising positions and allowing policy-guided improvements.

Algorithm Alpha Value Scheduler

Input: Episode number t, counter y, threshold st
Output: Alpha value $\alpha(t)$

Initialize: $st \leftarrow$ `max_episodes`$/1000$
for each episode t **do**
    Increment counter: $y \leftarrow y + 1$
    if $\lfloor t/1000 \rfloor == 0$ **then**
        if $\alpha == 0.1$ **and** $y \neq st$ **then**
            Set $\alpha \leftarrow 0.3$ ▷ Switch to exploitation
        else
            Set $\alpha \leftarrow 0.1$ ▷ Switch to exploration
        end if
    else
        Set $\alpha \leftarrow \alpha_{\text{default}}$
    end if
end for

3.5 Reward-Based Trajectory Optimization:

The agent optimizes its trajectory through the latent space by maximizing expected cumulative rewards. The trajectory optimization objective is: $J(\pi) = E_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ where $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...)$ represents a trajectory and γ is the discount factor. Reward Sampling Strategy: To maintain computational efficiency, rewards are sampled stochastically:

With probability $p_{eval} = 0.1$, generate image $\hat{x} = G(z_t)$ and query target model With probability $1 - p_{eval}$, use predicted reward $\hat{r}_t = f_r(s_t)$

3.6 Multi-Candidate Selection:

Traditional reinforcement learning approaches select single actions from policy networks, which can lead to suboptimal decisions due to policy approximation errors and environmental noise. In the context of model inversion attacks, poor action selection can result in ineffective latent space navigation and failed reconstructions. Our multi-candidate action selection mechanism addresses this limitation by evaluating multiple potential actions and selecting the most promising one based on predicted rewards.It reduces Variance Reduction,Improved Exploration, not stuck in suboptimal actions.

Algorithm Multi-Candidate Action Selection

Input: Current state s_t, policy π_ω, dynamics f_h, reward predictor f_r
Parameters: $K = 10$ (number of candidates)
Output: Selected action a_t

1: Generate K candidate actions: $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ where $a_i \sim \mathcal{N}(\pi_\omega(s_t), \sigma^2 I)$
2: Create state replications: $\mathcal{S} = \{s_t, s_t, \ldots, s_t\}$ (K identical copies)
3: Predict next states: $\mathcal{S}' = \{f_h(s_t, a_i)\}_{i=1}^K$
4: Evaluate predicted rewards: $\mathcal{R} = \{f_r(s'_i)\}_{i=1}^K$
5: Select optimal action: $a_t = a_{\arg\max(\mathcal{R})} = a_j$ where $j = \arg\max_i R_i$

3.7 Training and Optimization Procedure:

The training procedure follows an online reinforcement learning paradigm where the agent learns to navigate the latent space through interaction with the environment. Unlike supervised learning approaches that require pre-collected datasets, our method learns directly from the target model's responses, making it applicable to black-box scenarios where training data is unavailable. we use circular buffer to store experience and we constructed batch from $\mathcal{B} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^B \sim Uniform(ReplayBuffer)$

Algorithm ▮ Training Procedure

Input: Target model T, Generator G, Agent networks

Parameters: `max_episodes`, `batch_size`, `update_frequency`

for episode = 1 to `max_episodes` **do**
  Initialize $z_0 \sim \mathcal{N}(0, I)$, $s_0 = f_s(z_0)$
  Set adaptive α: $\alpha = \alpha(\text{episode})$
  for step = 1 to `episode_length` **do**
    Select action using multi-candidate strategy
    Update latent code: $z_{t+1} = \alpha \cdot z_t + (1 - \alpha) \cdot a_t[:z]$
    Predict next state: $s_{t+1} = f_h(s_t, a_t)$
    Sample reward stochastically
    Store experience: (s_t, a_t, r_t, s_{t+1})
  end for
  if episode mod `update_frequency` == 0 **then**
    Update agent networks using replay buffer
  end if
end for

3.8 Network Update Procedure:

We use a multi-objective loss function. The agent networks are trained using a composite loss function: $\mathcal{L}_{total} = \mathcal{L}_{dynamics} + \mathcal{L}_{reward}$.

Dynamics Loss: $\mathcal{L}_{dynamics} = \frac{1}{B} \sum_{i=1}^{B} \| f_h(s_i, a_i) - s'_i \|_2^2$

Reward Loss: $\mathcal{L}_{reward} = \frac{1}{B} \sum_{i=1}^{B} (f_r(s_i) - r_i)^2$

3.9 Network Overview:

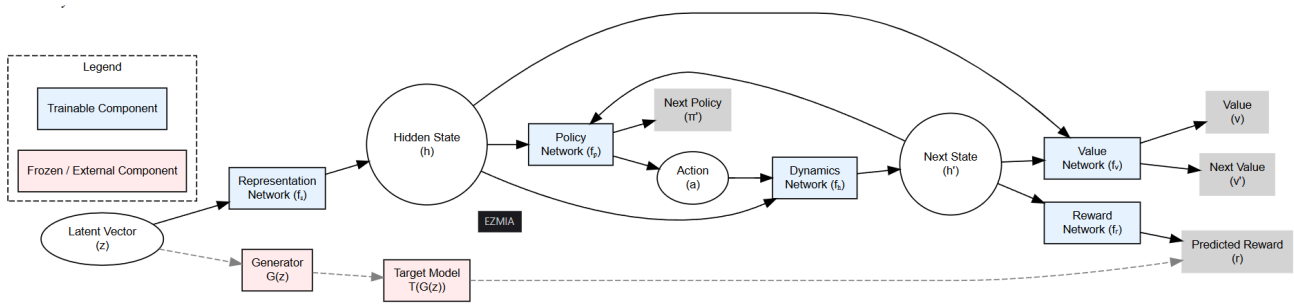

Figure 3.1: Network Workflow

Upper Figure 3.1 illustrates the architecture of our Model Inversion Attack , which

formulates black-box model inversion as a Markov Decision Process (MDP) in the latent space of a pretrained generator. The process begins by sampling a latent vector z, which is transformed into a hidden state using a representation network f_s. This state is then used by the policy network $f_p$to generate an action a, and the dynamics network f_d predicts the next hidden state '. Alongside this, a reward network $f_r$estimates the reward (classification confidence), and a value networkf_v predicts the value of the state. To choose actions effectively, the agent generates multiple candidate actions and selects the one yielding the highest predicted reward. The latent vector is updated as a weighted combination of the original latent and the action vector. Importantly, only a small fraction of the interactions (e.g., 10%) query the black-box target model T by passing the generated image G(z') to obtain the true classification probability for the desired label. This true reward is used to evaluate performance and populate the replay buffer. For most steps, the model relies on predicted rewards to guide exploration. The replay buffer is then used to train the dynamics and reward networks using supervised losses, enabling the agent to better model the latent transition dynamics and reward estimation over time. By casting the attack as an MDP and leveraging reinforcement learning components, the approach enables query-efficient and target-aware exploration of the latent space for high-fidelity inversion.

Chapter 4

Experiments

4.1 Experimental Setting& Implementation details:

We evaluate our attack against VGG16 and ResNet-152[](i.e These are the target models).

Datasets: We trained Target models on CelebFaces Attributes (CelebA) dataset which holds 10,177 identities and total 2,02,599 images, out of which we choose 1000 identities and 30,027 images belong to those identities. In addition, we trained our GAN on 30000(randomly) images out of 9177 identities.

Models: We trained both target models VGG16 and ResNet-152 models on the CelebA dataset, achieved validation accuracies of 60.92% and 62%, respectively. The models were trained for approximately 75 and 71 epochs, respectively. Additionally, we trained the GAN for 300 epochs using an RTX 3050 GPU. We trained FACENET-112 on same target model's train data in 30 epoch,we achieved validation accuracies of 64.77%.

Model architecture: We take $z_{dim} = 100$(latent vector size) $h_{dim} = 256$ (internal state size) $a_{dim} = 256$(action vector size).

Our proposed model architecture consists of five primary components: a representation network, a dynamics network, a reward predictor, a policy network, and a value network. Together, these modules implement a latent-space Markov Decision Process (MDP) formulation inspired by EfficientZero, as we described earlier. The representation network f_s maps the latent vector $z \in R^{100}$ into an internal hidden state $h \in R^{256}$ using a two-layer multilayer perceptron with LayerNorm and ReLU activations. The dynamics function f_d takes the concatenation of the current state h and an action vector $a \in R^{256}$,

and outputs the next state $h' \in R^{256}$. A separate reward network f_r, composed of two fully connected layers with a sigmoid activation at the output, estimates the scalar reward $r \in [0, 1]$, which approximates the target model's confidence for a given class. The policy network f_p is a linear projection that maps the hidden state h to a continuous action vector $a \in R^{256}$. The value network f_v outputs a scalar estimate of the expected return from state h, using a single linear layer followed by a hyperbolic tangent activation.

Candidate actions are sampled from a Gaussian prior, and the action yielding the maximum predicted reward is selected for exploration. During inference, the agent performs a single initial inference to generate a starting state and then applies recurrent inference through the dynamics model. Latent vectors are updated via a linear interpolation: $z' = \alpha z + (1 - \alpha)a$, where $\alpha \in [0, 1]$ controls the update strength. The generator G maps the updated latent vector $z' \in R^{100}$ to an image, which is passed to the frozen target model T to obtain the softmax confidence for the ground truth label. A small fraction of these interactions query the target model directly to obtain the true reward, which is then stored along with the transition tuple (h, a, r, h') in the replay buffer. The dynamics and reward networks are updated using mean squared error losses computed over mini-batches sampled from this buffer. This architectural design enables latent-space planning with minimal queries by leveraging learned predictive models within an MDP framework.

Replay Buffer Structure: The replay buffer stores experience tuples (s_t, a_t, r_t, s_{t+1}) with the following specifications:

Capacity: 100,000 transitions using circular buffer implementation Sampling Strategy: Uniform random sampling for batch construction.

Batch Construction Training batches are constructed by sampling $B = 256$ transitions from the replay buffer: $\mathcal{B} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{B} \sim Uniform(ReplayBuffer)$

Optimizer Config:We use adam with Learning Rate: $\alpha = 10^{-4}$ with no scheduling

Hyperparameters:

Adaptive Alpha Values: $\alpha_{explore} = 0.1$, $\alpha_{exploit} = 0.3$

Sampling Frequency: $p_{eval} = 0.1$(i.e., we query the target model only 10% of the time.)

Multi-Candidate Count: $K = 10$

4.2 Evaluative metrics:

1)**KNN Distances:**KNN Distance quantifies similarity between reconstructed and real images in feature space. It calculates the L distance to the nearest neighbor of the target class. Features are extracted from the evaluation classifier's penultimate layer. Smaller values indicate closer resemblance to genuine samples. This metric helps evaluate attack stealthiness.

2)**Feature Distance:**Feature Distance assesses reconstruction fidelity by comparing images to class centroids. It computes the L distance between a fake image's features and the target class's average features. Like KNN Distance, it uses pre-FC-layer activations. Lower distances suggest better preservation of class characteristics. This complements KNN Distance for robust attack analysis.

3)**Attack accuracy:**Attack accuracy:Attack Accuracy measures how well reconstructed images deceive machine learning models. It is evaluated using separate classifiers trained on private datasets to avoid overfitting.

4.3 Results

In GMI[15], which is a white-box attack, the authors reported an attack accuracy of 0.185 on VGG16, where the model's classification accuracy was 88% (as shown in Table 1 of the RLB-MI[19] comparison). In contrast, we achieved a comparable attack accuracy of approximately 0.1 on a VGG16 model with only 60.92% classification accuracy, using just 400 queries—significantly fewer than the 40,000 queries used in GMI—and with much lower computational cost, making our approach faster than both GMI and RLB-MI.Moreover, our method is well-suited for real-world scenarios, where API-based access imposes strict query limits and simulates more practical and constrained attack environments. As noted by S. Yeom et al.[32], increased overfitting tends to raise privacy risks, which may partially explain the higher attack accuracy observed in the GMI and RLB-MI settings. Since their models were well-trained, it is likely that the classification accuracy on the training data exceeded 90%, indicating potential overfitting that could have contributed to increased vulnerability to model inversion attacks.

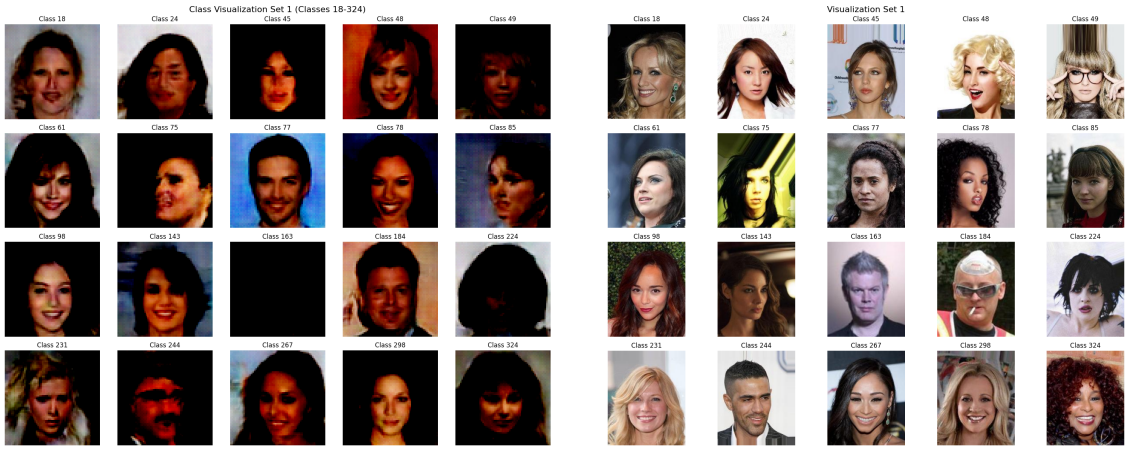

Figure 4.1: Left was reconstructed images (randomly)

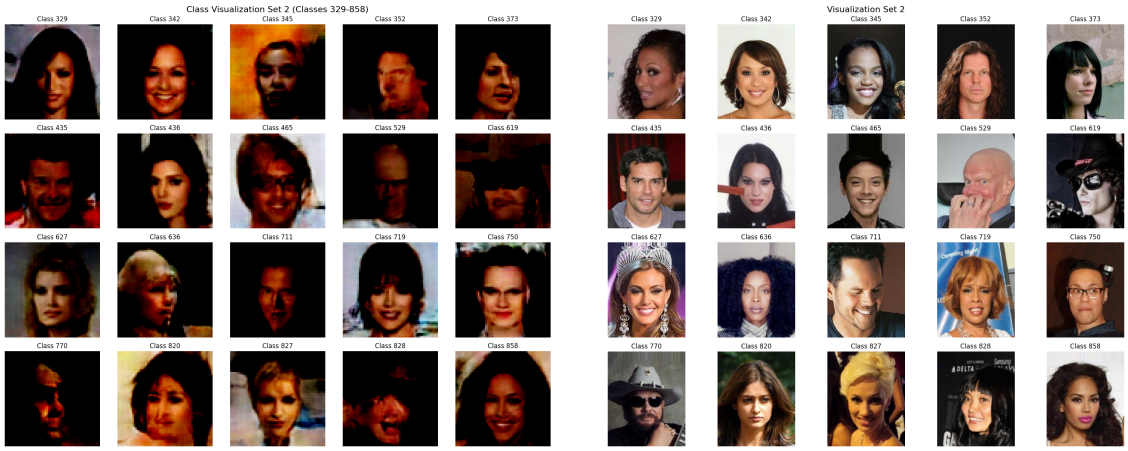

Figure 4.2: Left was reconstructed images (randomly)

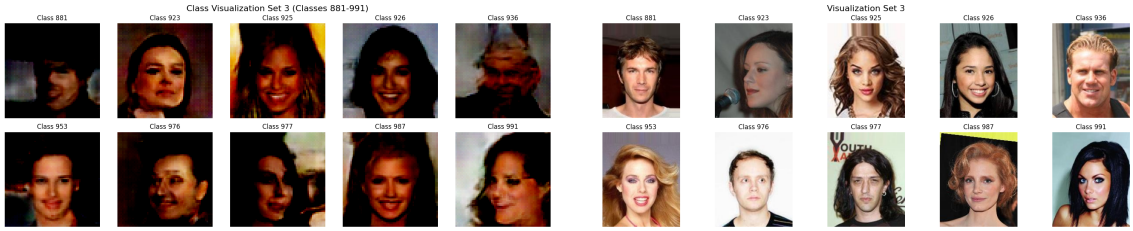

Figure 4.3: Left was reconstructed images (randomly)

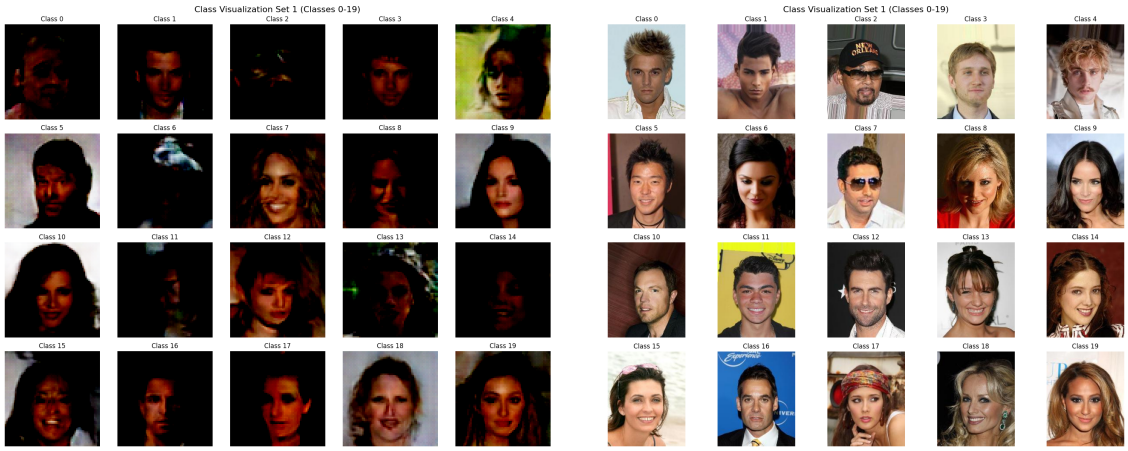

Figure 4.4: Left was reconstructed images

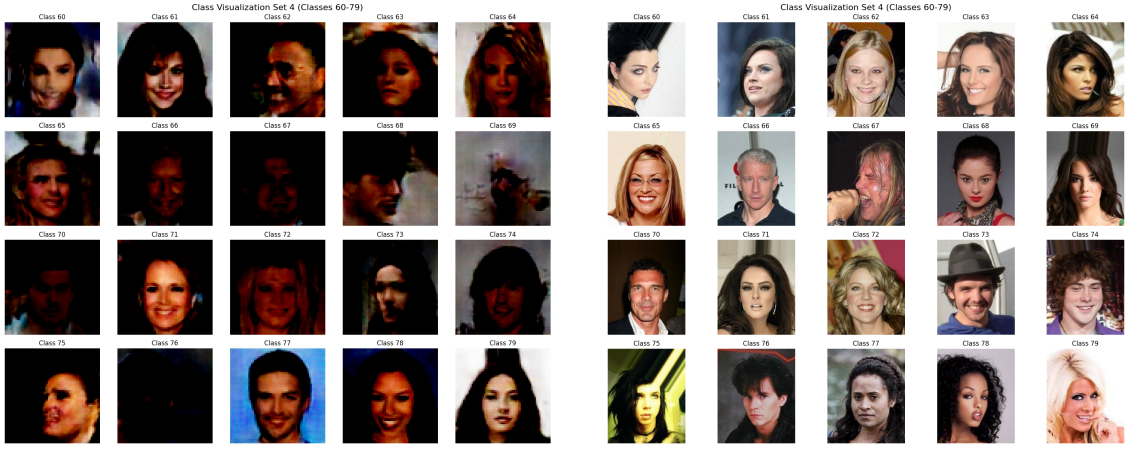

Figure 4.5: Left was reconstructed images

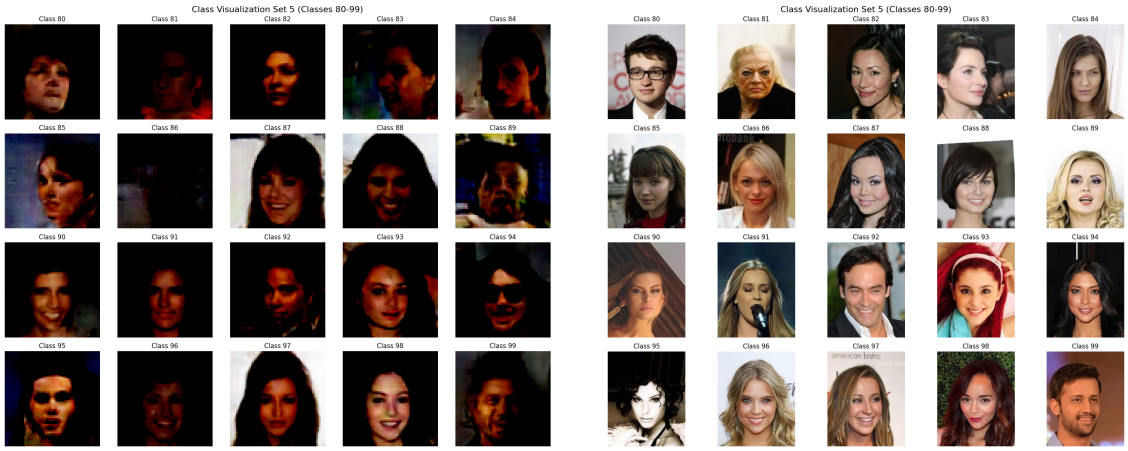

Figure 4.6: Left was reconstructed images

Chapter 5

Conclusion

5.1 Summary:

In this work, we proposed a reinforcement learning-based latent space navigation framework for model inversion attacks, inspired by the EfficientZero architecture. By formulating the search over a generator's latent space as a sequential decision-making process, our method overcomes key limitations of gradient-based approaches such as local minima and gradient vanishing.

We implemented all core components—representation, dynamics, reward, value, and policy networks—and trained them jointly using online interaction with the target model. Our design incorporates adaptive -blending for latent updates, multi-candidate action selection, and stochastic reward sampling, enabling efficient and scalable black-box attacks. Empirical evaluation demonstrates that the approach is capable of producing high-confidence reconstructions for target classes.

5.2 Future Works

MCTS with Gumbel Top-k Sampling: Our current method achieves very fast using value-guided latent space navigation. As future work, integrating this architecture with as MCTS along Gumbel Top-k sampling (for continuous action spaces) may further enhance trajectory quality and long-term decision making in more complex inversion tasks.

Diversity-Promoting Exploration: Introducing a diversity buffer or novelty-based reward component may help discover a broader range of high-reward latent codes. This can reduce redundancy in exploration and encourage the agent to cover distinct semantic regions of the latent space.

Temporal Noise Scheduling: Dynamically adjusting the variance of candidate action sampling over time may enhance the exploration-exploitation balance, with high exploration in early stages and more focused refinement as training progresses.

Multi-Objective Reward Integration: Building on the existing reward structure, future extensions may incorporate additional objectives such as confidence maximization and diversity encouragement. This can be combined with adaptive weighting or normalization techniques to stabilize learning across varying tasks and models.