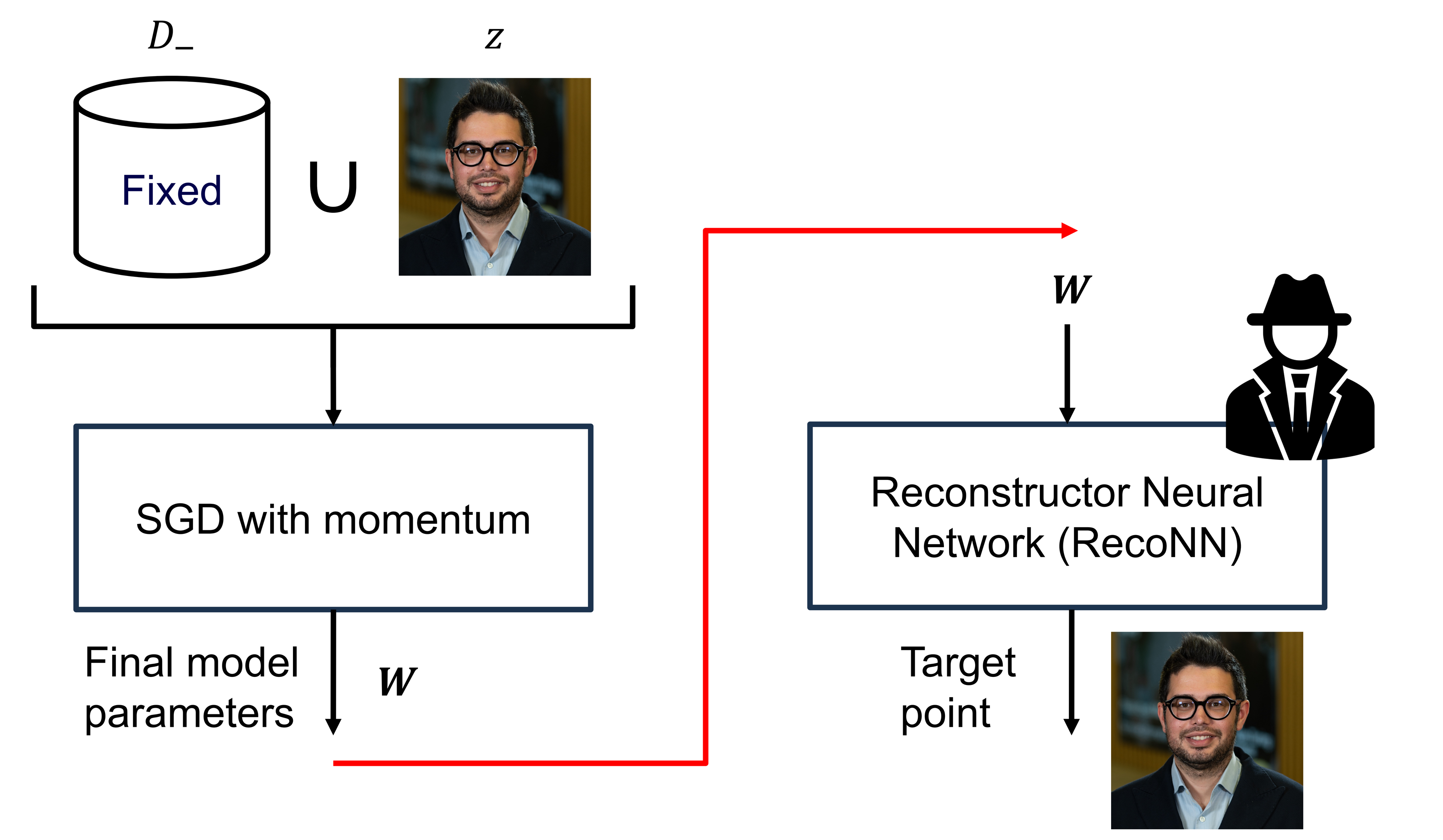
Mnemonist: Locating Model Parameters that Memorize Training Examples

Ali Shahin Shamsabadi, Jamie Hayes, Borja Balle, Adrian Weller

39th Conference on Uncertainty in Artificial Intelligence (UAI), Pittsburgh, PA, USA, August 2023

The Alan Turing Institute

Reconstructing training data with an informed adversary

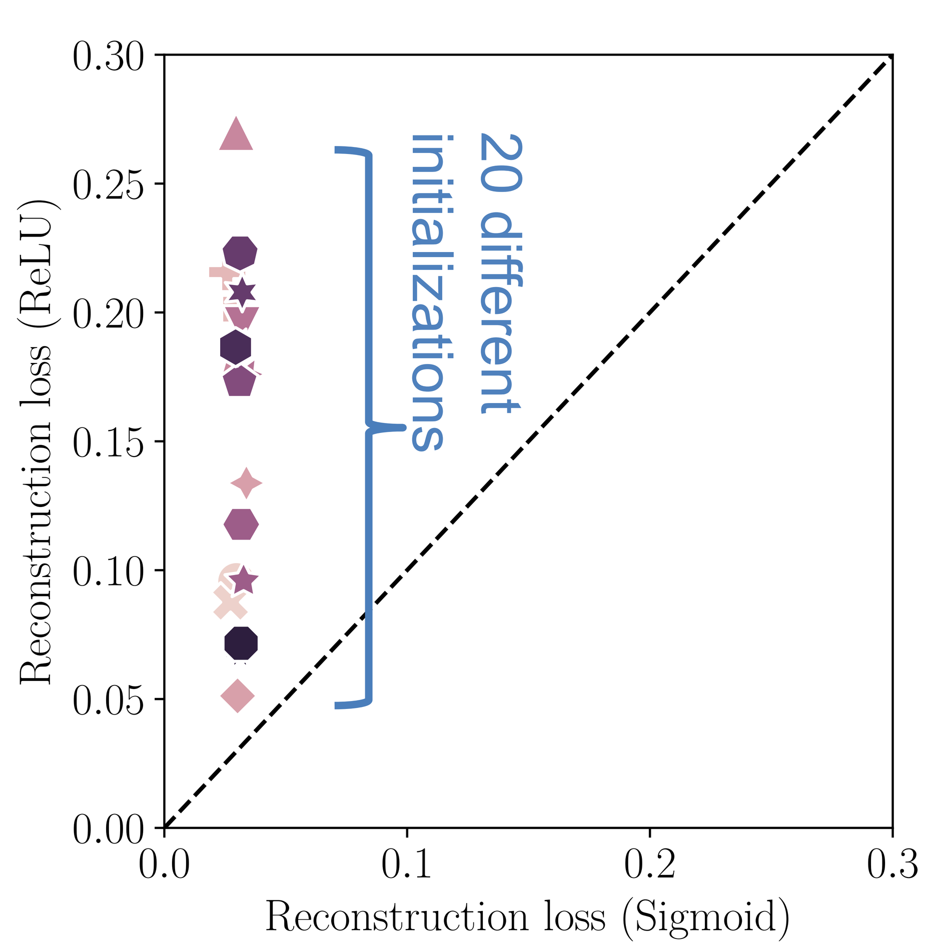


The attack consists of three steps:

1. Training **shadow models** to collect information about the impact of training examples on model parameters
   - The adversary knows $(D_-, \boldsymbol{W}, \boldsymbol{W}_{\text{init}}, S)$ + a public dataset $\bar{Z} = \{\bar{z}_i\}_{i=1}^{K}$
   - The adversary trains each shadow model, $\overline{\boldsymbol{W}}_i$, on $D_- \cup \{\bar{z}_i\}$
2. Training a **RecoNN** to output training examples from model parameters
   - Loss function: $\mathrm{MSE}(\mathrm{RecoNN}(\overline{\boldsymbol{W}}_i), \bar{z}_i) + \mathrm{MAE}(\mathrm{RecoNN}(\overline{\boldsymbol{W}}_i), \bar{z}_i)$
3. Producing a **candidate** reconstruction for the target point

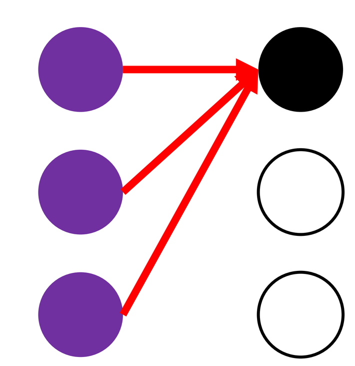
Reconstruction quality depends heavily on the activation

ReLU activations lead to **higher reconstruction loss** compared to **Sigmoid**.



Why is training data reconstruction from ReLU activated models hard?

Characterising the fundamental property of the existence of **redundant parameters** in models with ReLU activations:



Pre-activation: $h^l = w^l x + b$

Activation: $a^l = R(h^l)$

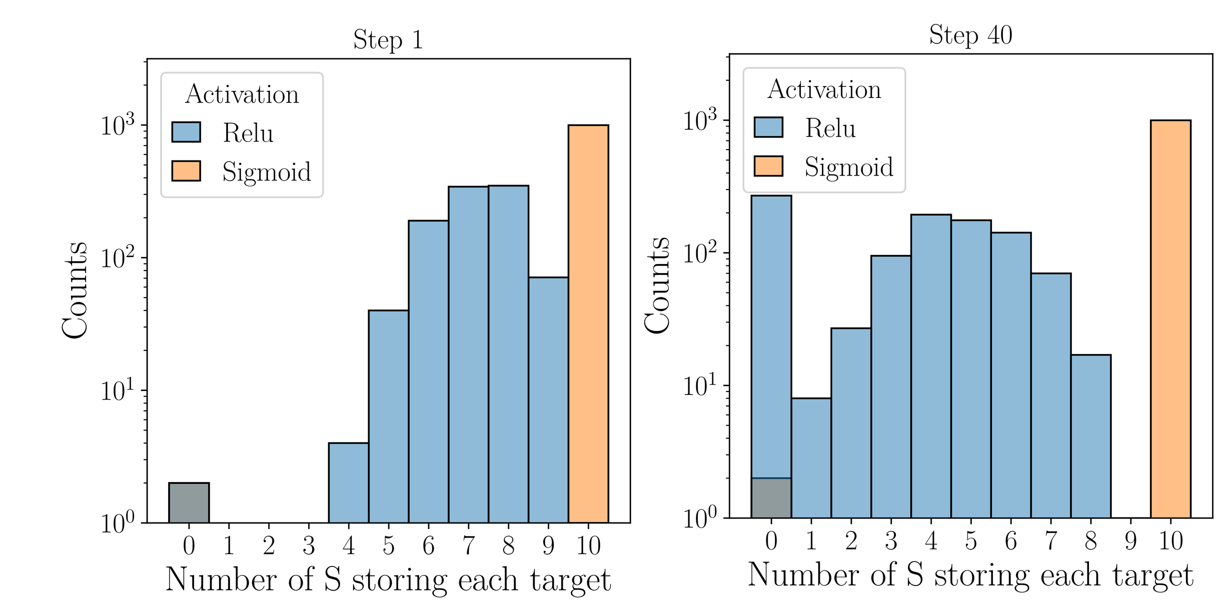
$$\frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^l} \cdot \frac{\partial a^l}{\partial h^l} \cdot \frac{\partial h^l}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^l} \cdot R'(h^l) \cdot x$$

$$\frac{\partial \mathcal{L}}{\partial w^l} = \begin{cases} 0 \\ \mathrm{scale} \cdot x \end{cases}$$

No information about the input is stored in incoming parameters to non-activate neurons

Incoming parameters to active neurons store a copy of input data

Demonstrating the existence of redundant parameters **in practice** by studying the **training dynamics**:

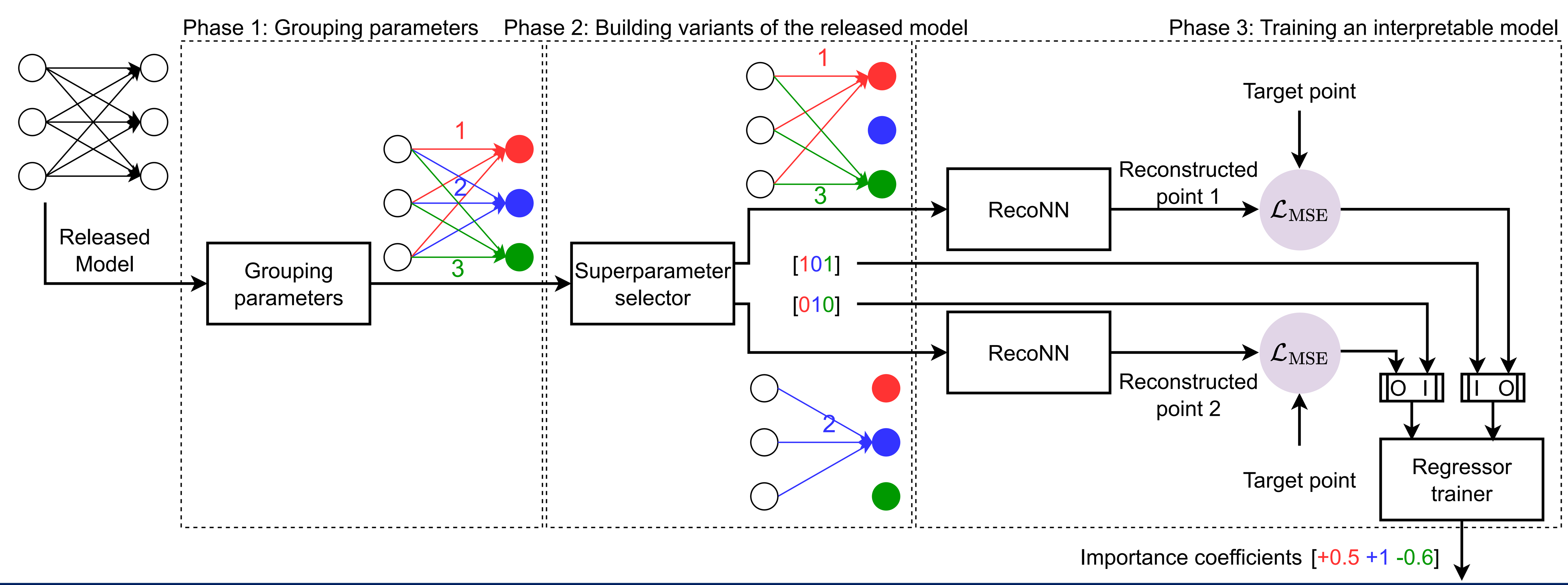


Histogram of the summation of

$$\text{Binary scale} = \begin{cases} 0 & \text{if scale} = 0 \\ 1 & \text{Otherwise} \end{cases}$$

of all rows per target point. [1k target points]

Mnemonist: finding parameters that store target examples



Mnemonist helps RecoNN to improve the quality of reconstructions

Mnemonist-RecoNN trains RecoNN only on **informative superparameters**.

Approach	run1	run2	run3	run4
RecoNN	.2040	.2705	.0908	.1315
Mnemonist-RecoNN	.1738	.2385	.0730	.1158

Algorithm 1: Mnemonist-RecoNN

Input: Fixed set D_-, K public target examples $\{\bar{z}_k\}_{k=1}^{K}$, Shadow model training Algorithm $A(\cdot)$, RecoNN training algorithm $B(\cdot)$.

Output: Mnemonist-guided RecoNN.

```
for all k ∈ range(K) do
    W̄_k ← A(D_− ∪ {z̄_k})                ▷ Train K shadow models
φ ← B({W̄_k, z̄_k}_{k=1}^K)                ▷ Train RecoNN
I ← Mnemonist(φ)          ▷ Apply Mnemonist to identify the importance of superparameters
for all k ∈ range(K) do
    W̃_k ← Selector(W̄_k, I)              ▷ Select only important superparameters
φ̃ ← B({W̃_k, z̄_k}_{k=1}^K)    ▷ Train RecoNN on the selected important superparameters
return φ̃                                 ▷ Mnemonist-guided RecoNN
```