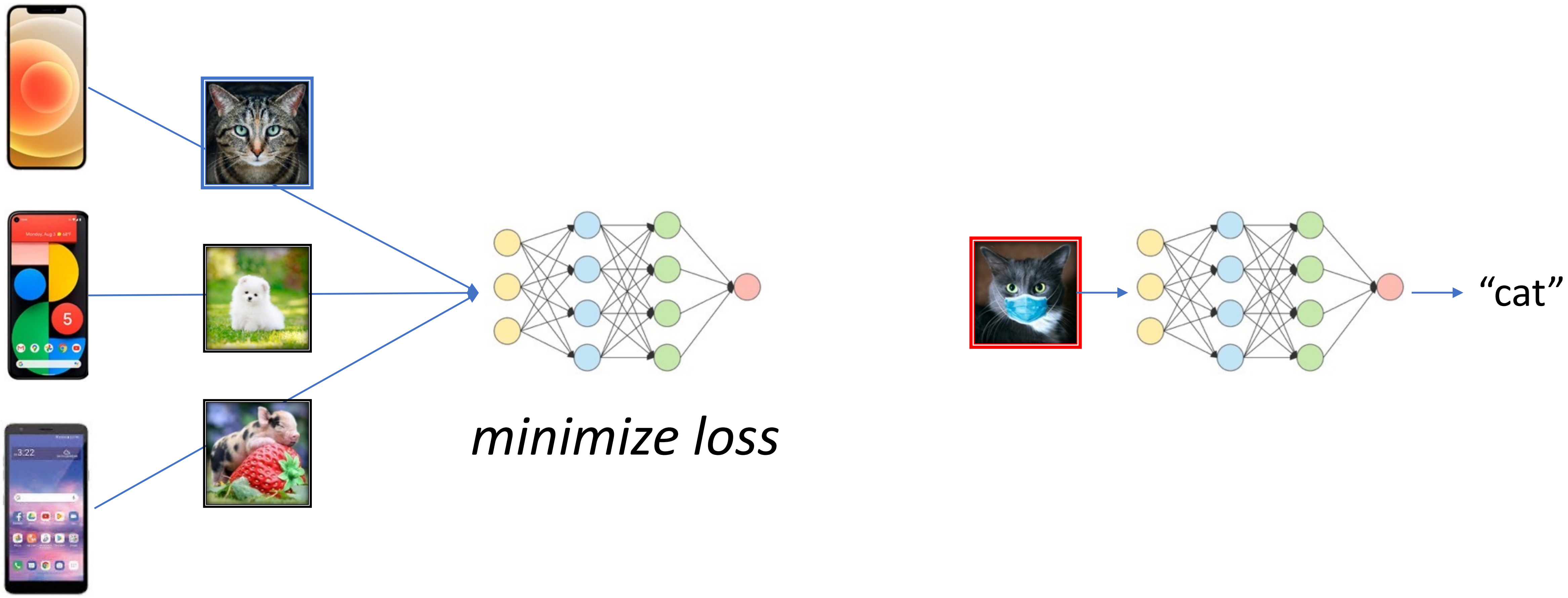
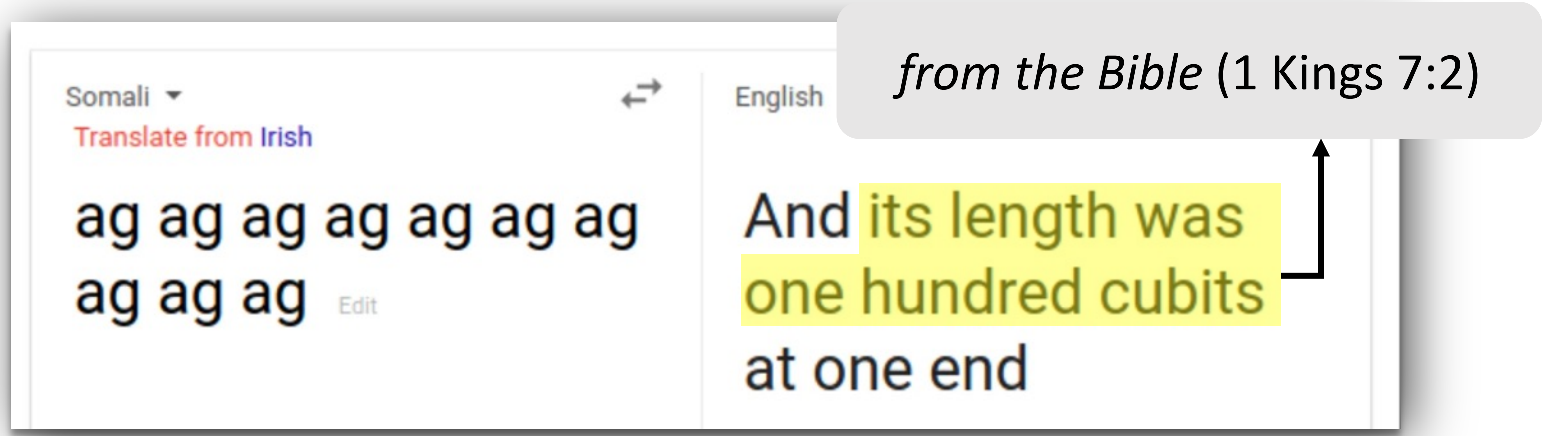
# Measuring privacy leakage in neural networks

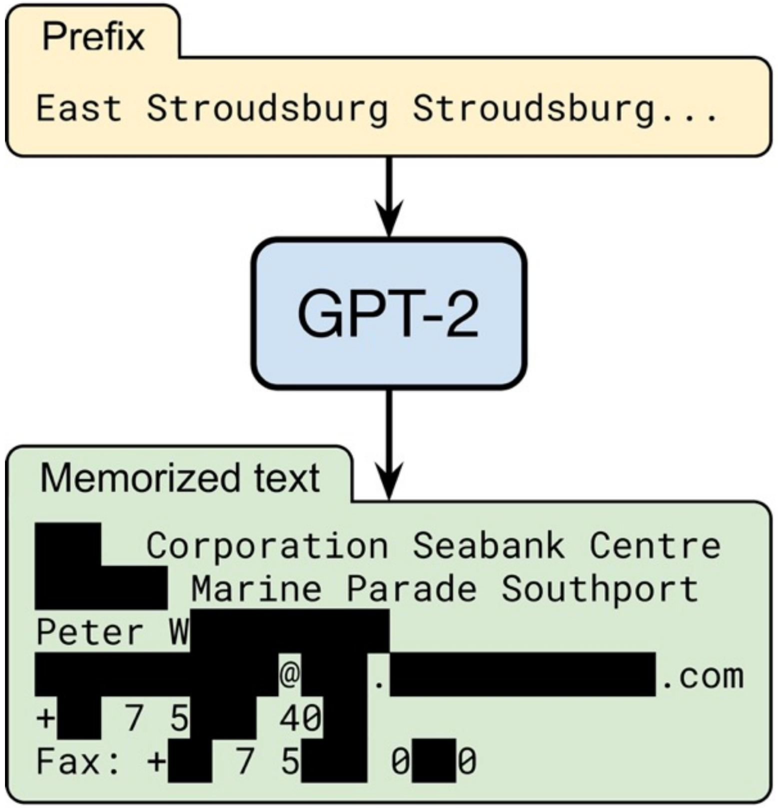
**Florian Tramèr**

# Neural networks learn from a (private) training set.

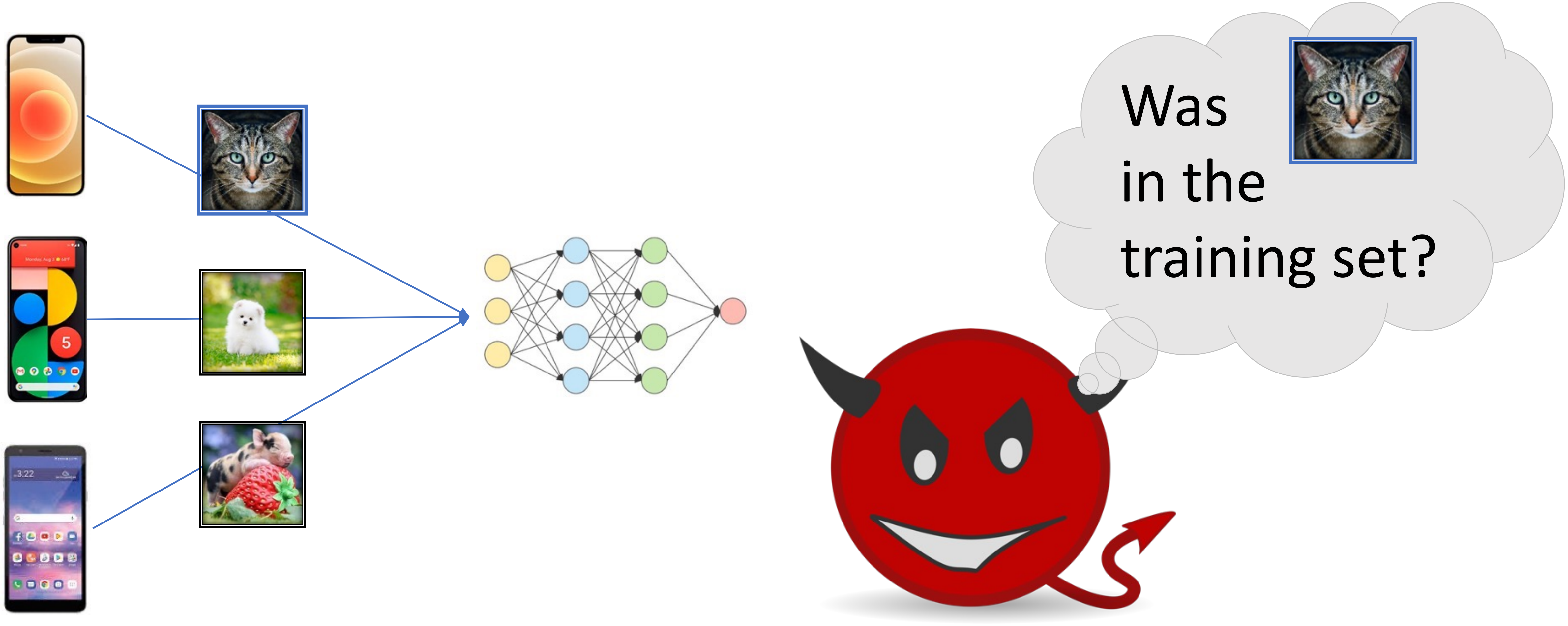


# The trained model might *leak* the training set.



# This talk: Membership inference attacks



# Why should we care about membership inference?

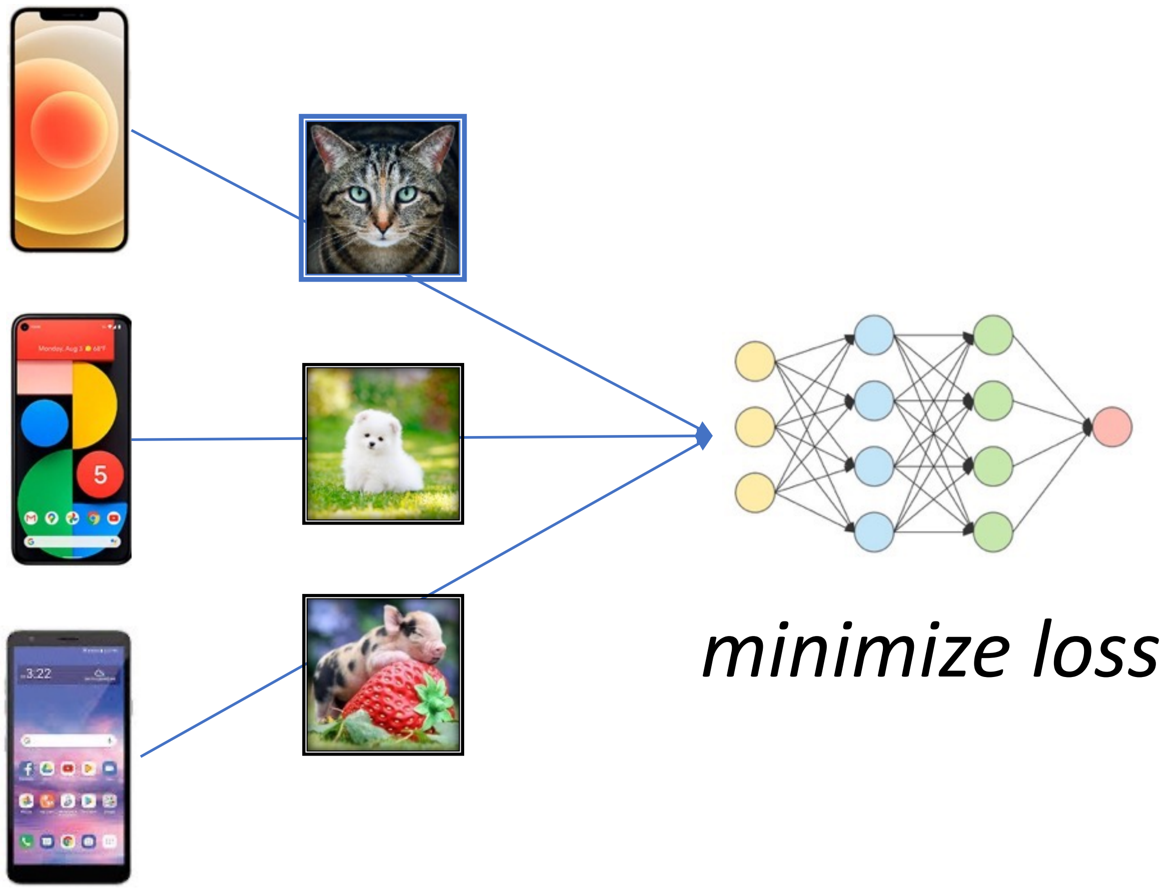
1. A real attack (e.g., models trained on medical data)
2. An attack component (e.g., for data extraction)
3. A simple, formal upper-bound on data leakage

| Data privacy | | Secure encryption |
|---|---|---|
| Membership inference attack | $\approx$ | Chosen plaintext attack |

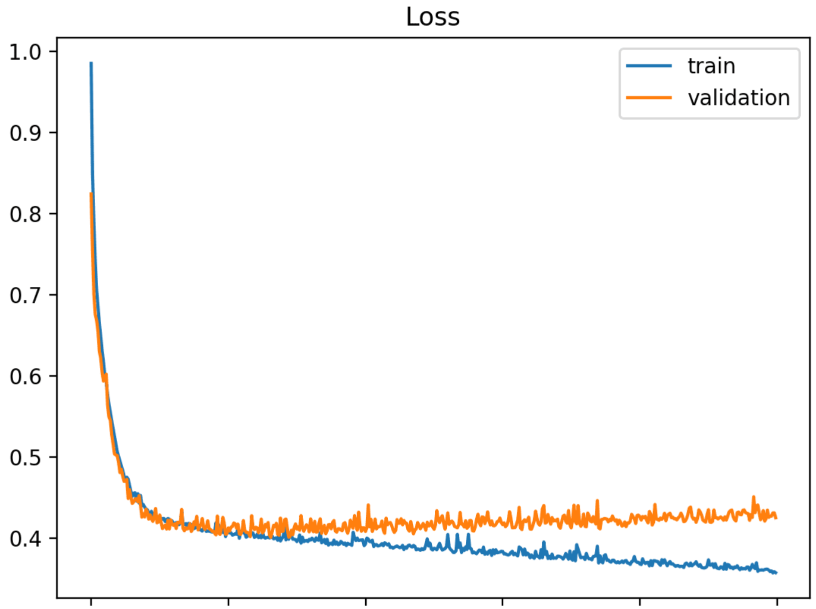
# Outline.

- Most membership attacks (and their evaluations) *are flawed*
- A new principled attack that works on *outliers*
- A new stronger attack that works for *any input*
- Defenses and *how to audit them*

# Models are trained to minimize loss.



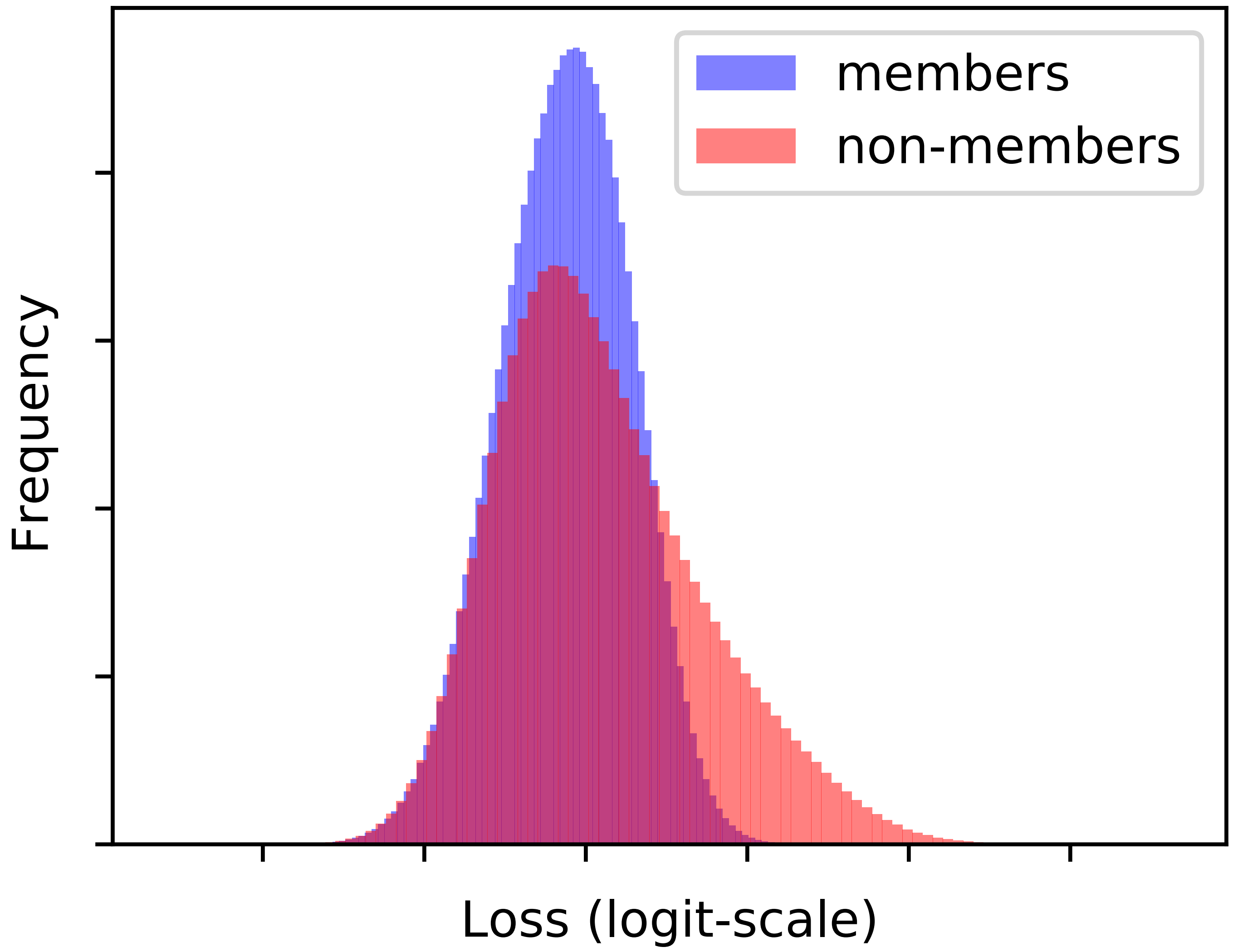
*minimize loss*




**Loss is (slightly) lower for training examples!**

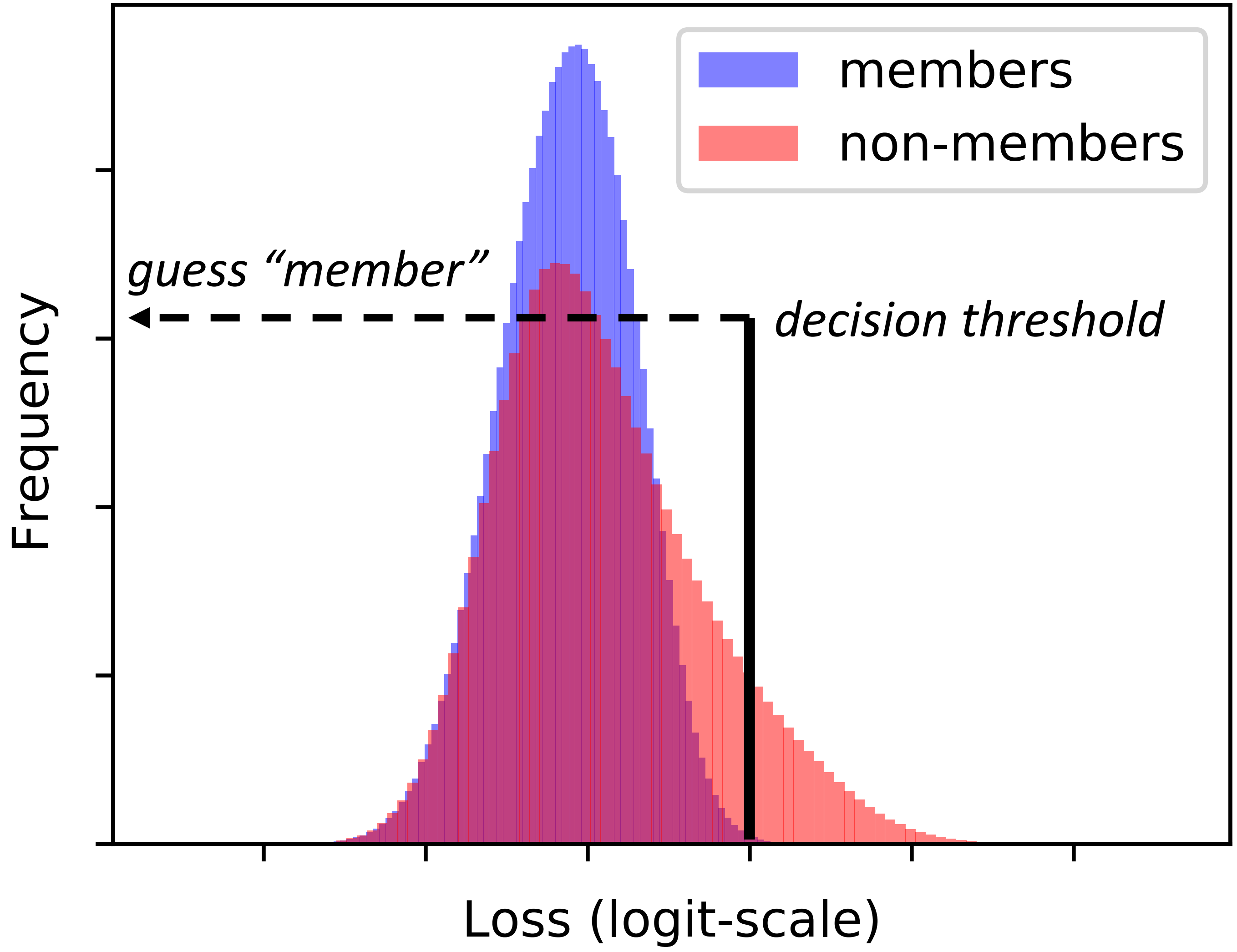
# A simple MI attack: “uniform” loss thresholding

[Yeom et al.’18]



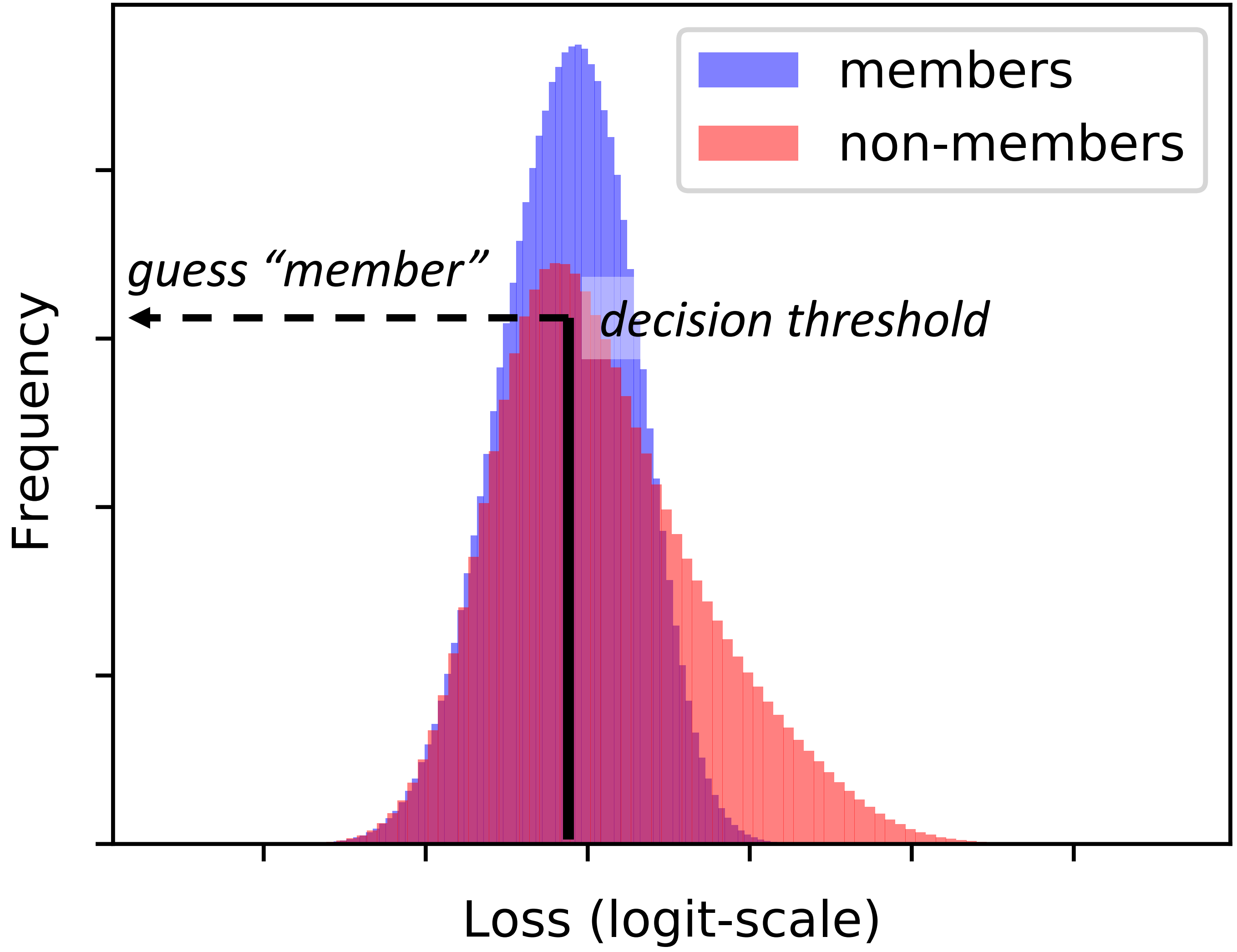
# A simple MI attack: “uniform” loss thresholding

[Yeom et al.’18]



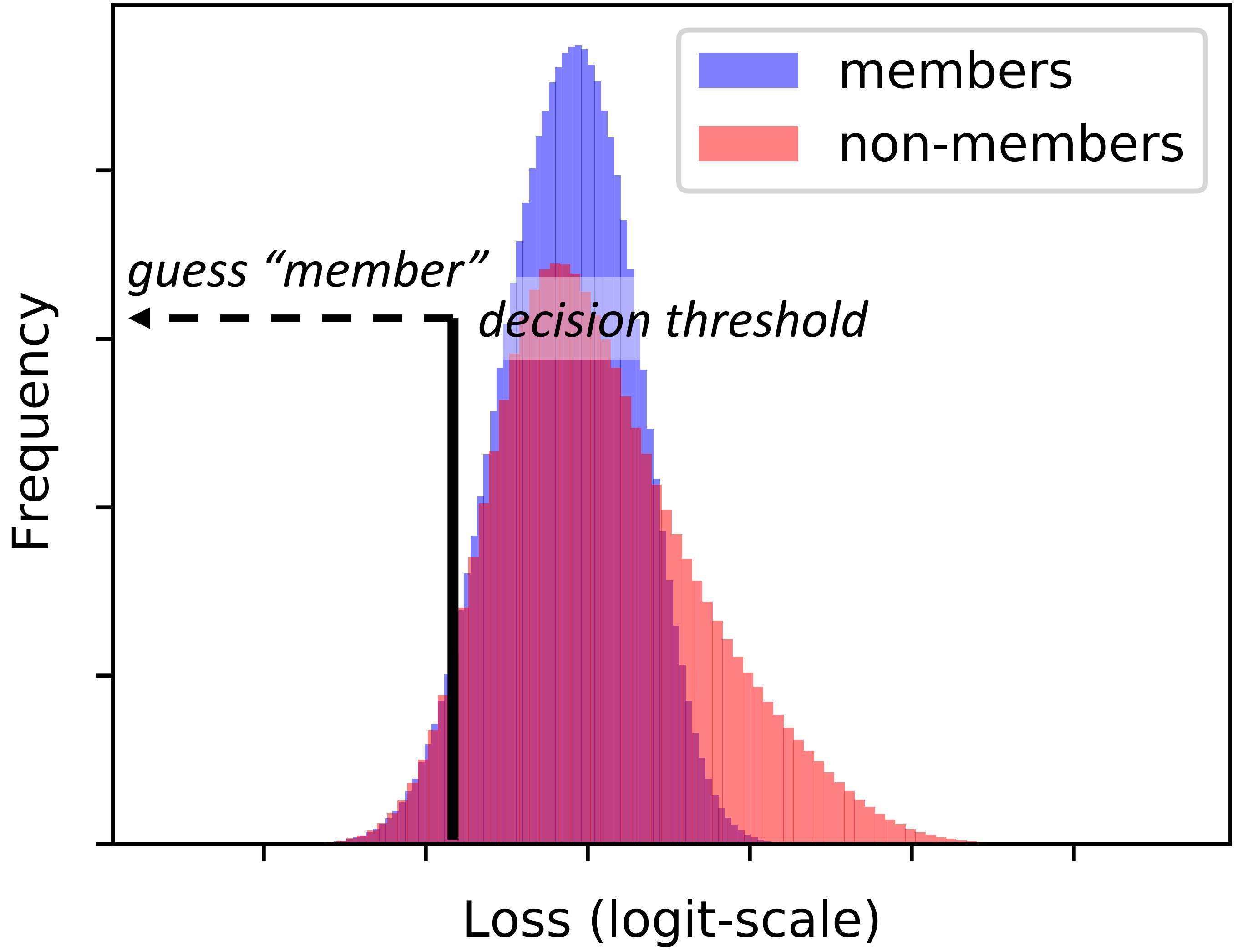
# A simple MI attack: “uniform” loss thresholding

[Yeom et al.’18]

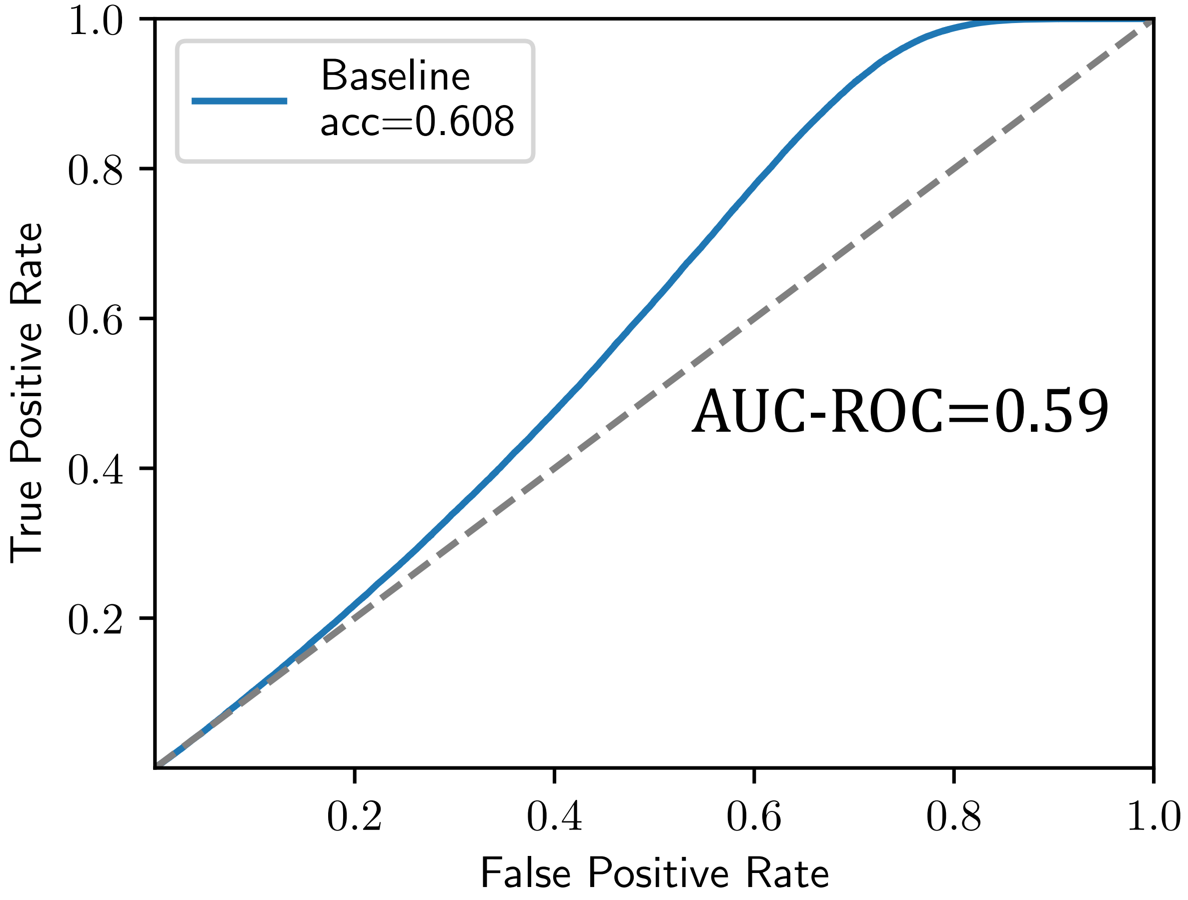


# A simple MI attack: “uniform” loss thresholding

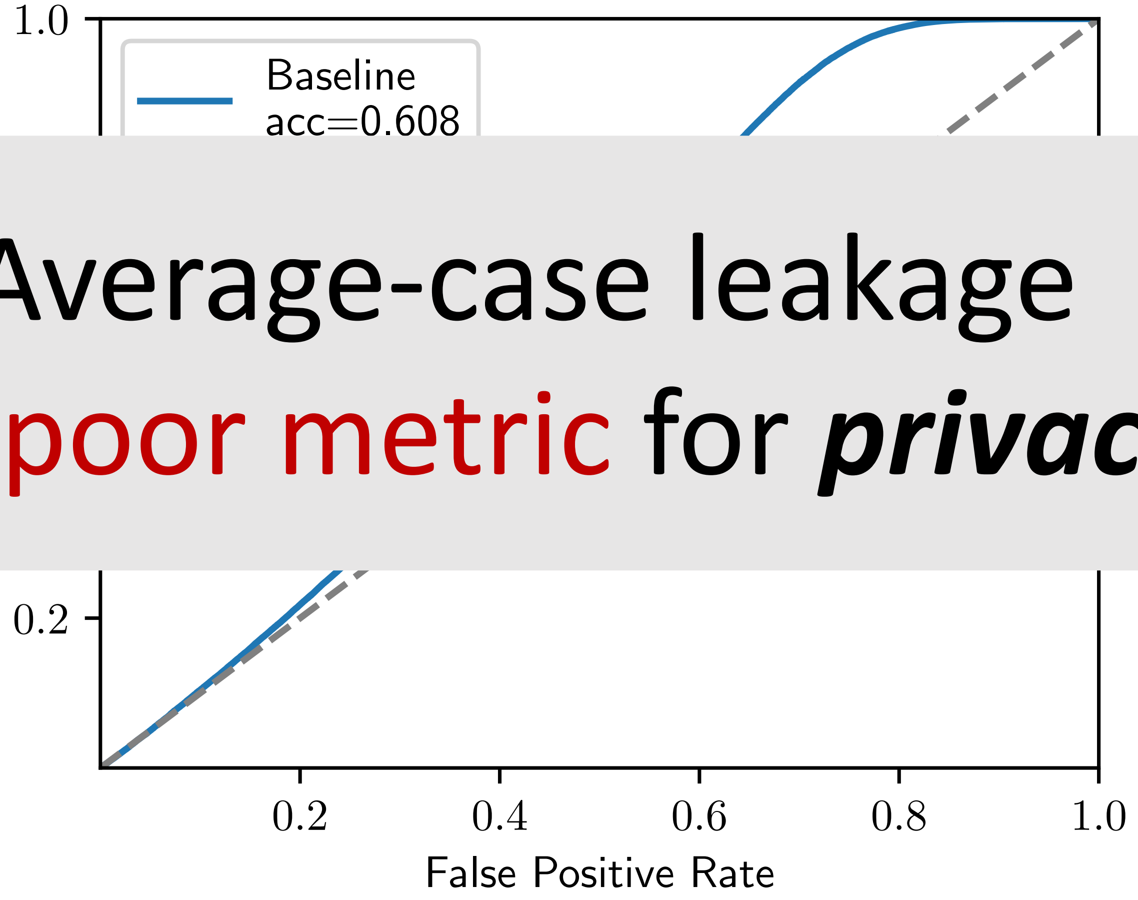
[Yeom et al.’18]



# A model's loss leaks membership *on average.*

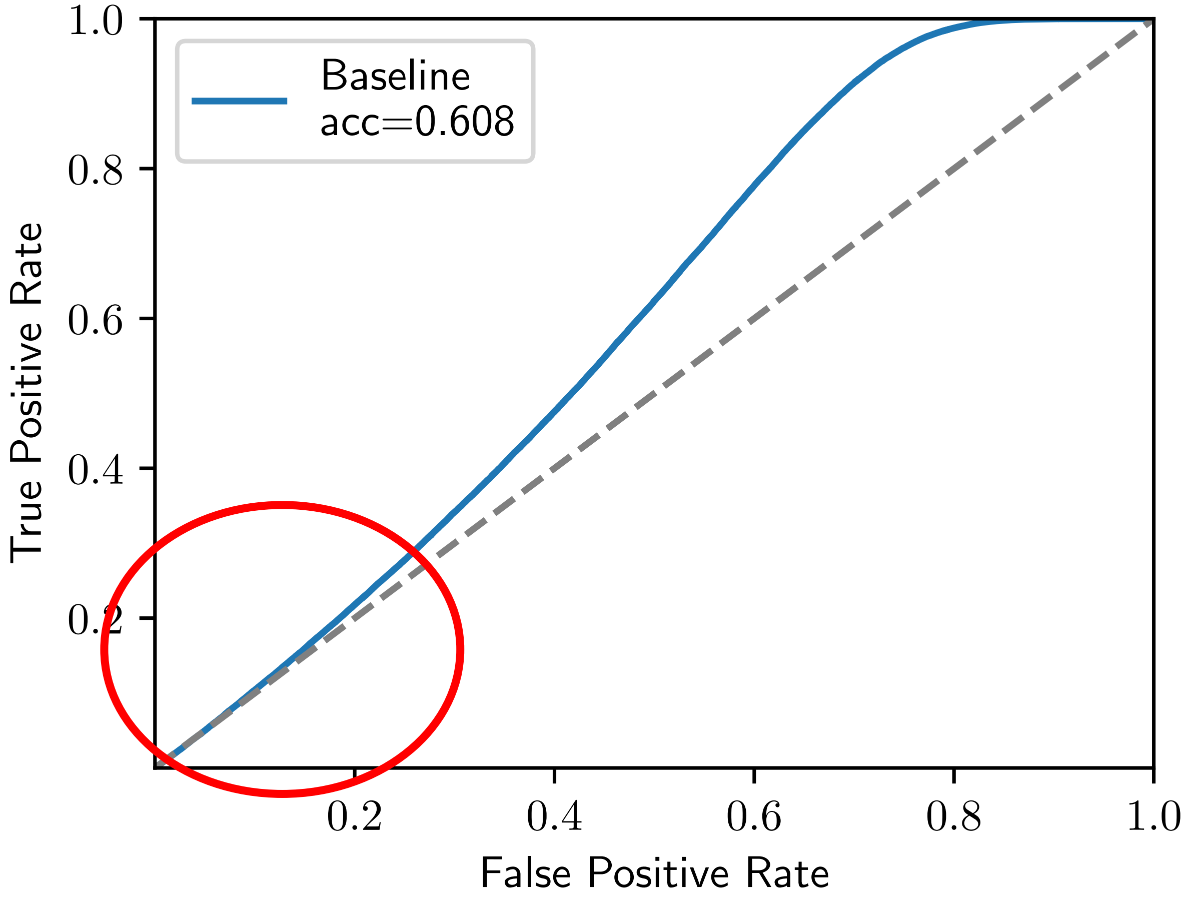


# A model's loss leaks membership *on average.*

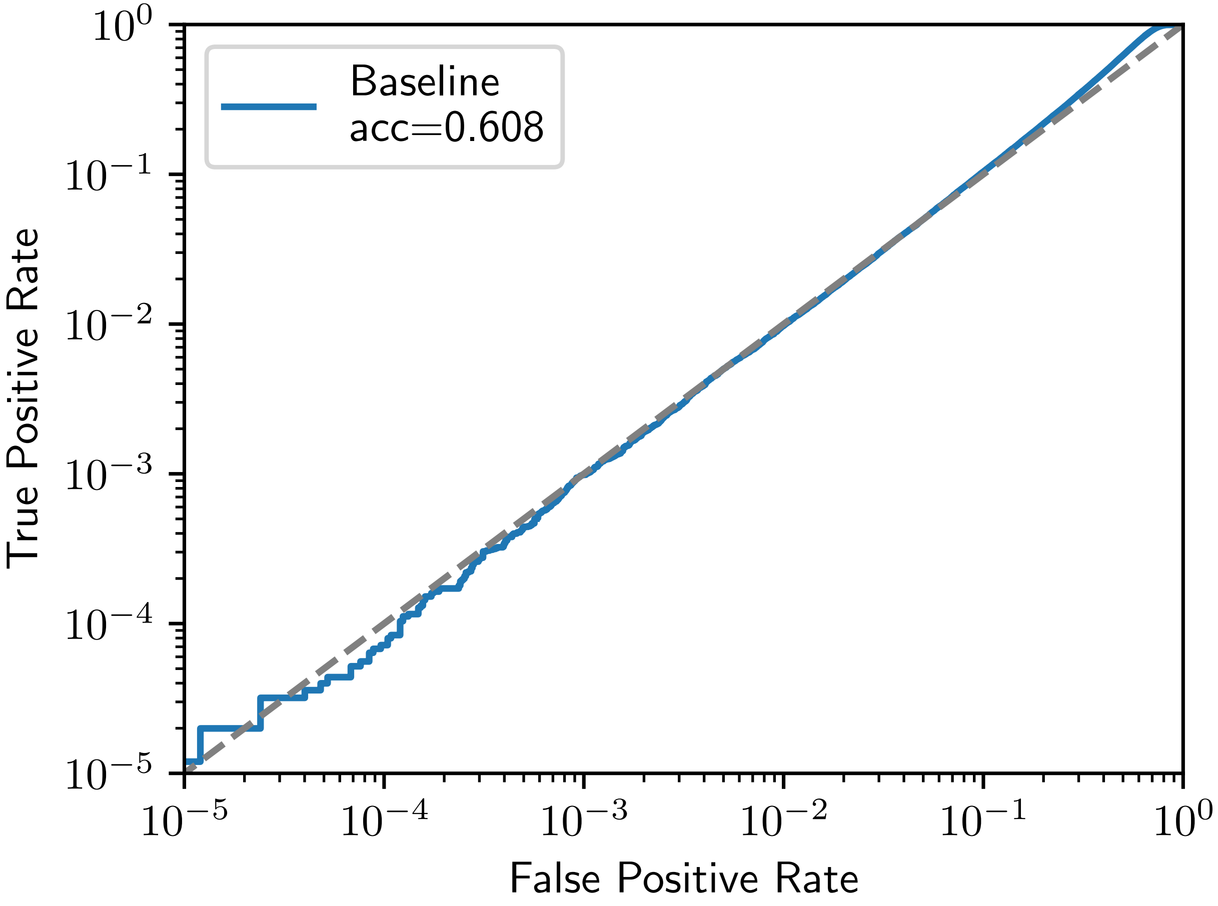



Average-case leakage
is a poor metric for ***privacy***!

# “Uniform” loss thresholding doesn’t *confidently* infer membership of *any* member of the train set!



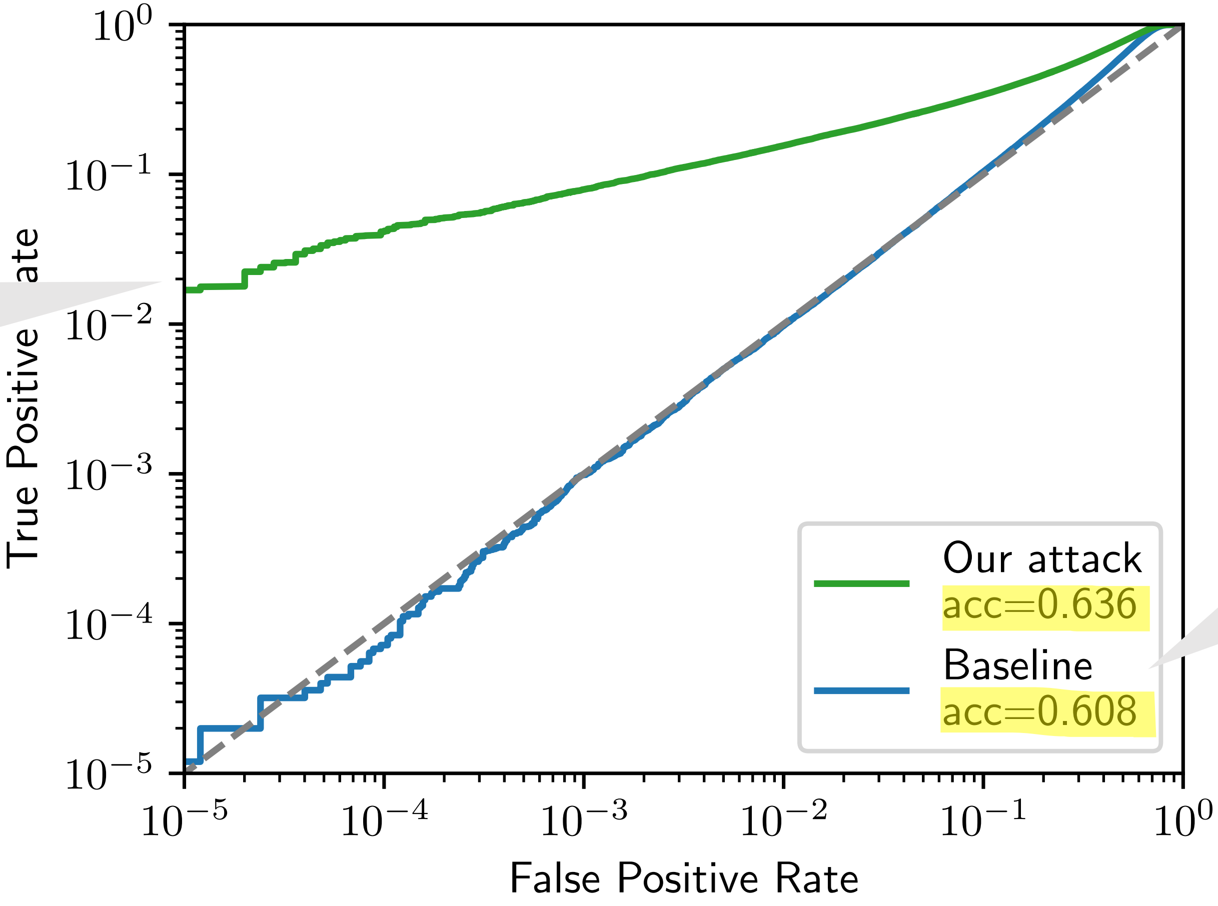
# Our preferred evaluation methodology: *low* FPRs



# LIRA: A better MI attack!

Carlini et al., “Membership Inference Attacks From First Principles”, IEEE S&P ‘22

**>1000x better in the *worst case***




slightly better *on average*

# Insight: not all examples are equally "hard"

[Sablayrolles et al.'19, Long et al.'20, Feldman & Zhang'20, Watson et al.'21, Ye et al.'21]

Which is a member?


**loss: $10^{-4}$**

**loss: 0.01**

# Insight: not all examples are equally “hard”

[Sablayrolles et al.’19, Long et al.’20, Feldman & Zhang’20, Watson et al.’21, Ye et al.’21]

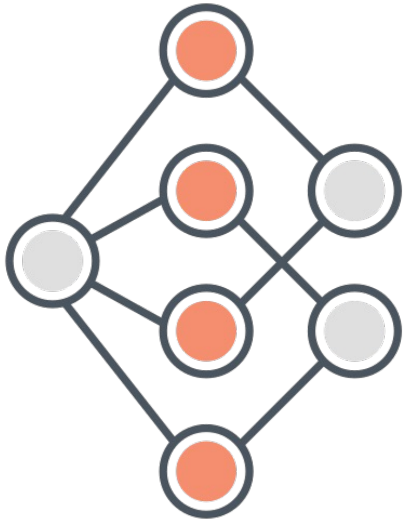
Which is a member?


**loss: $10^{-4}$**

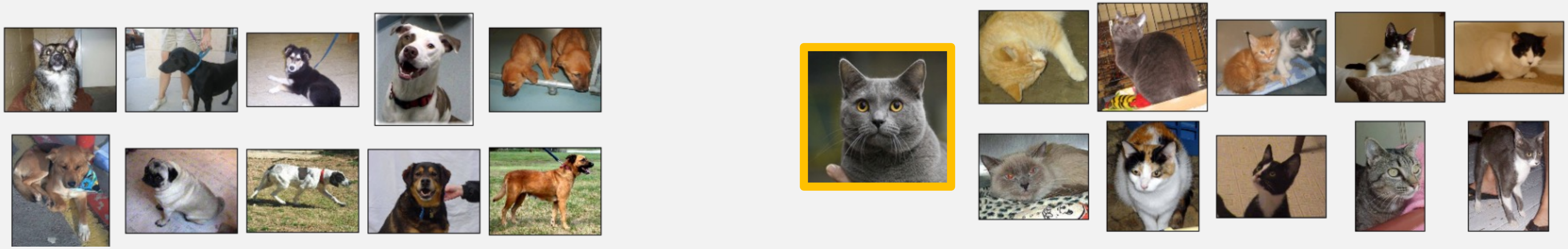
**loss: 0.01**

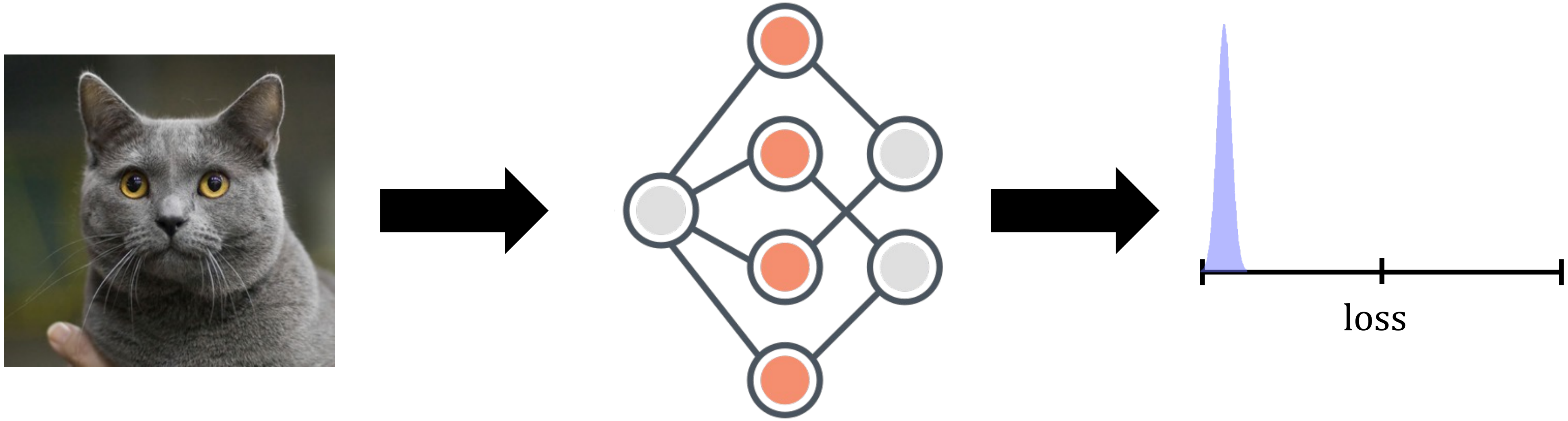
# Let’s try a membership inference attack!

$\overset{?}{\in}$

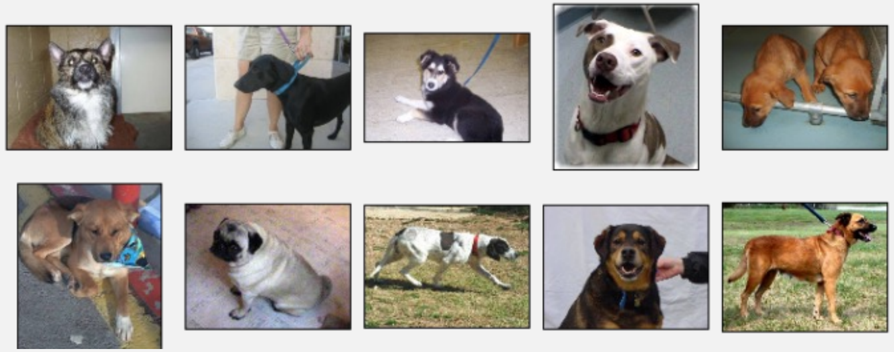


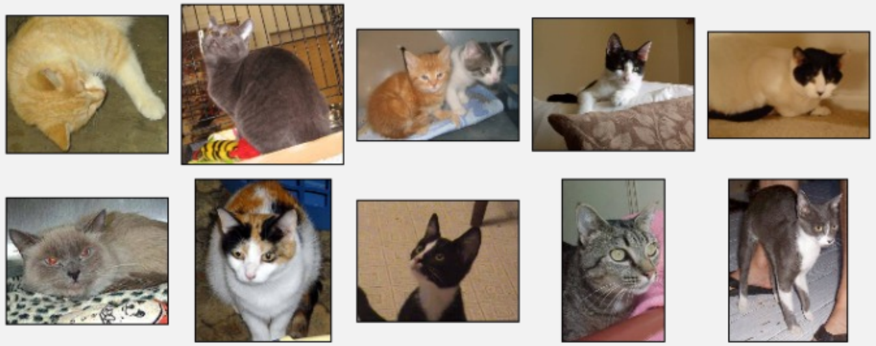
# Training set

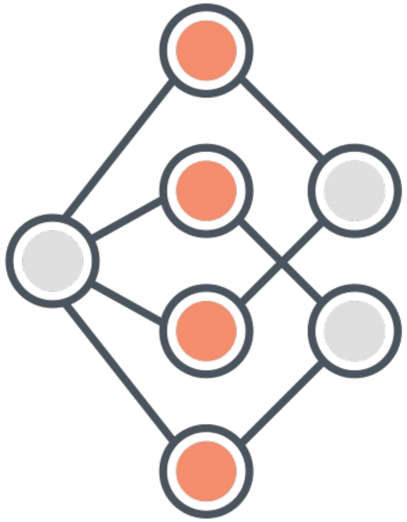


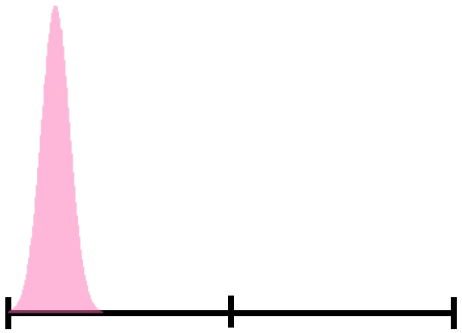


# Training set

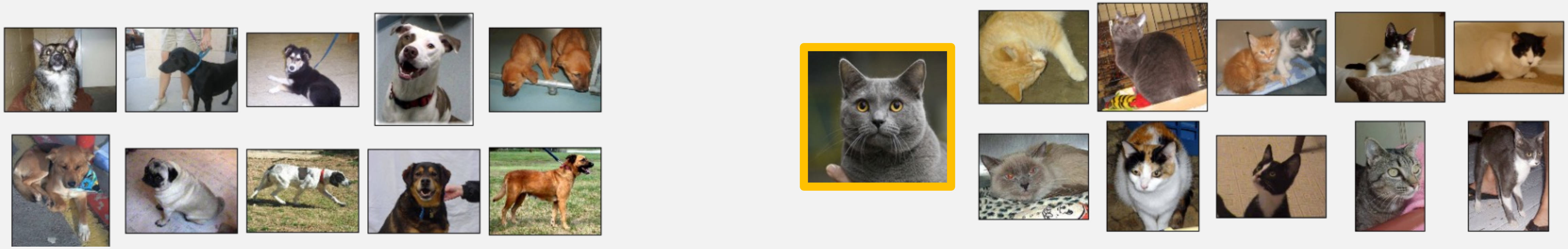


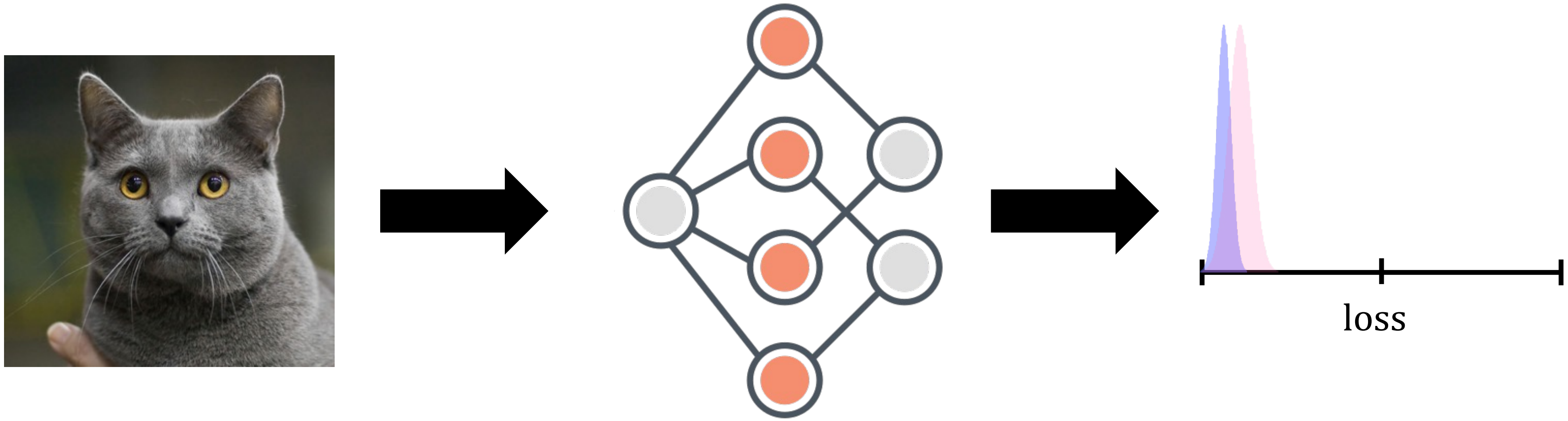




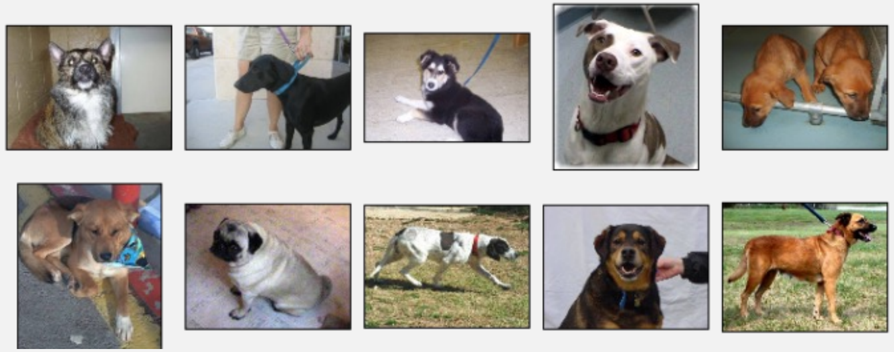


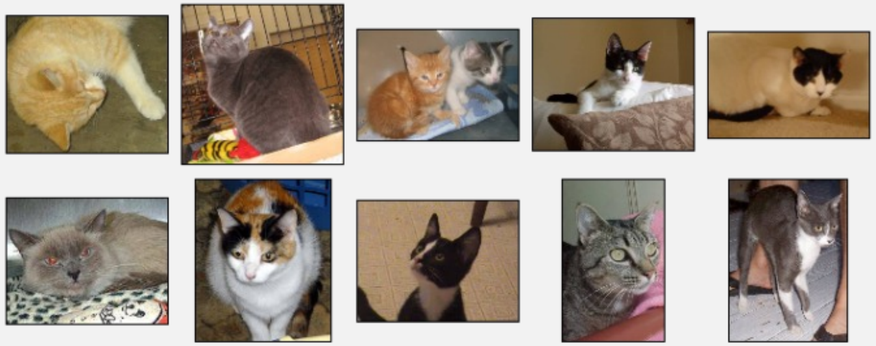
# Training set

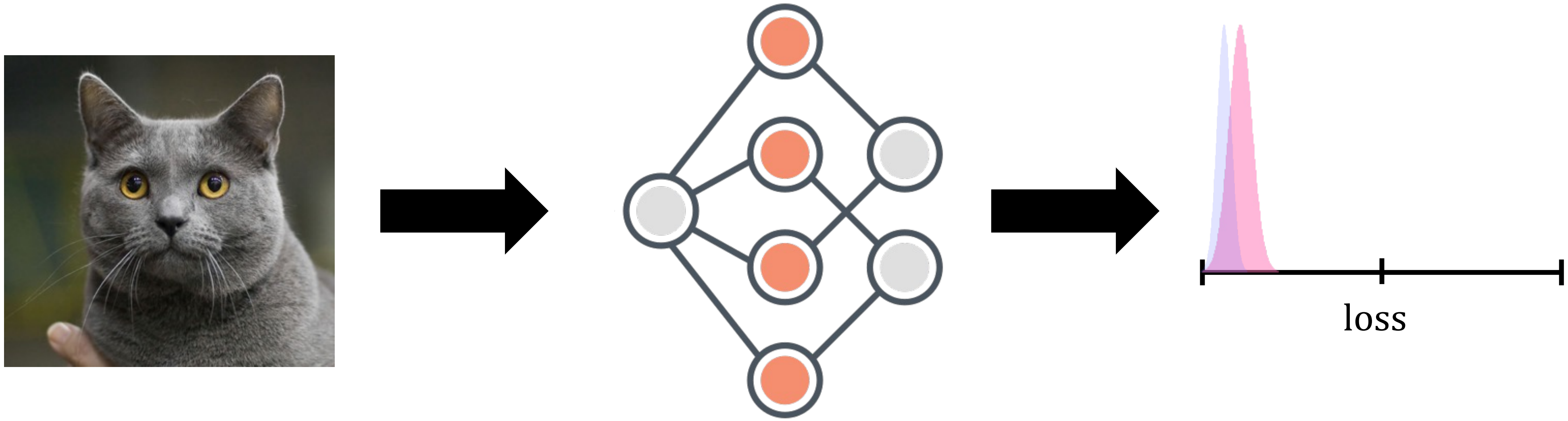




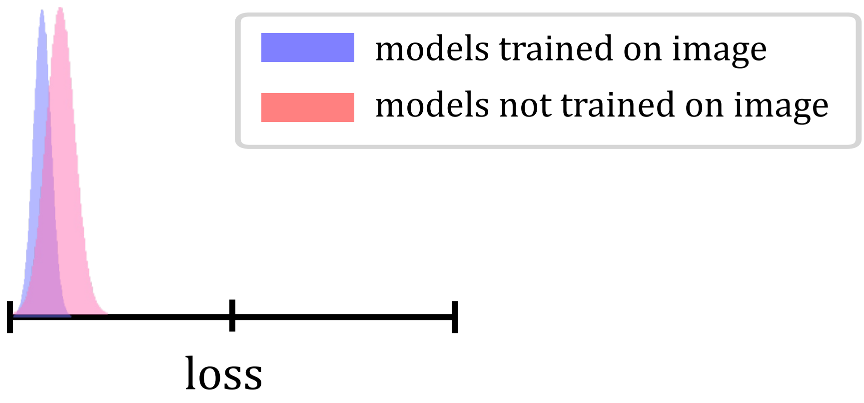
# Training set



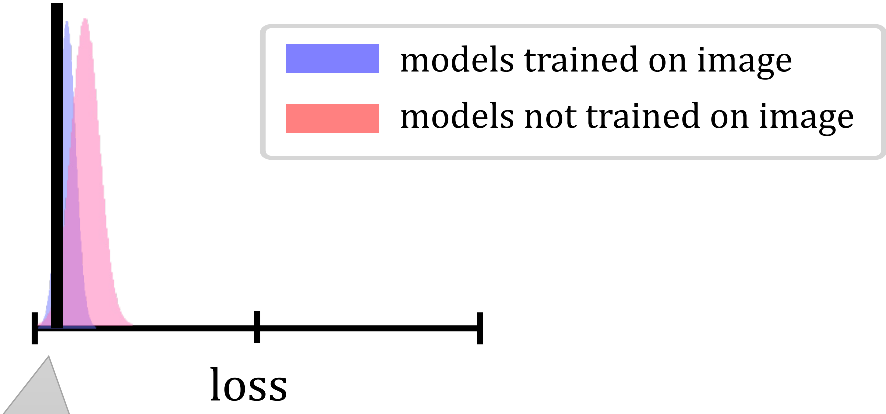




# Membership inference as a *likelihood test.*



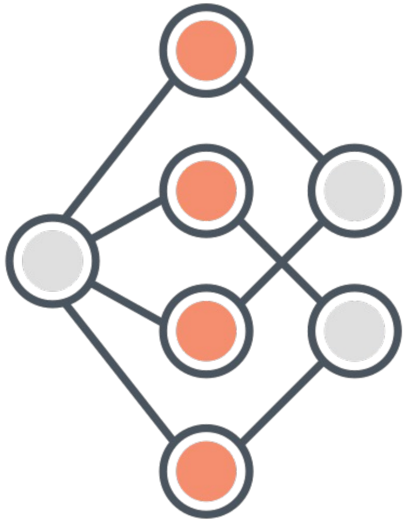
# Membership inference as a *likelihood test.*



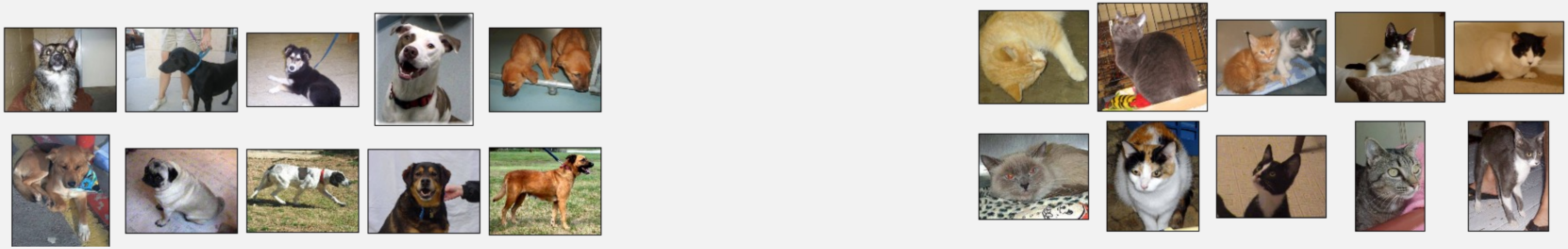

threshold set to achieve a FPR of $\alpha$

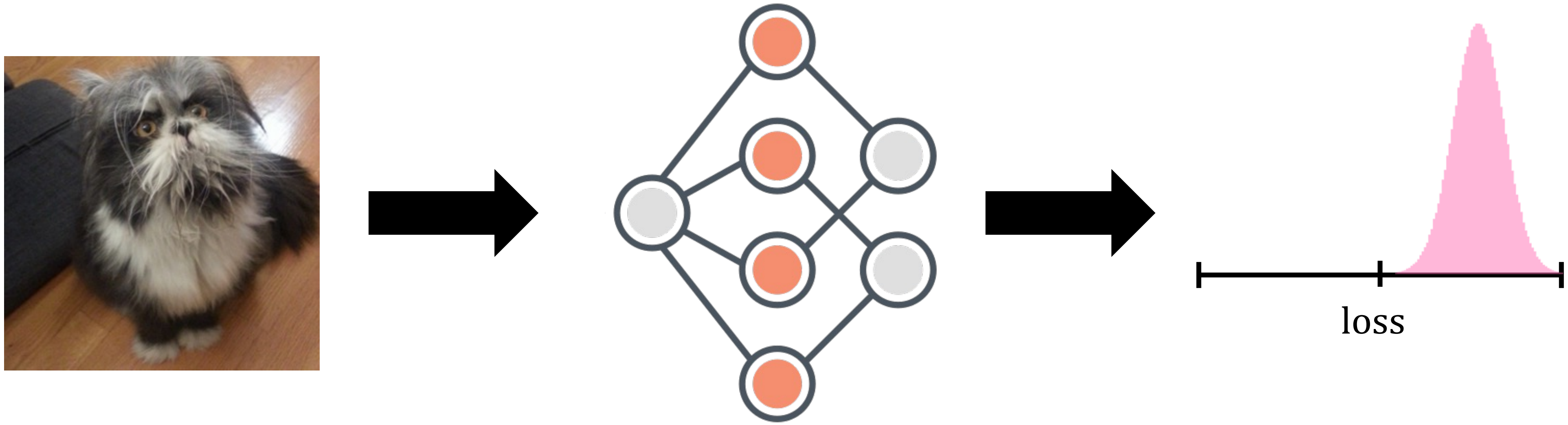
# Let’s try again!


$\overset{?}{\in}$

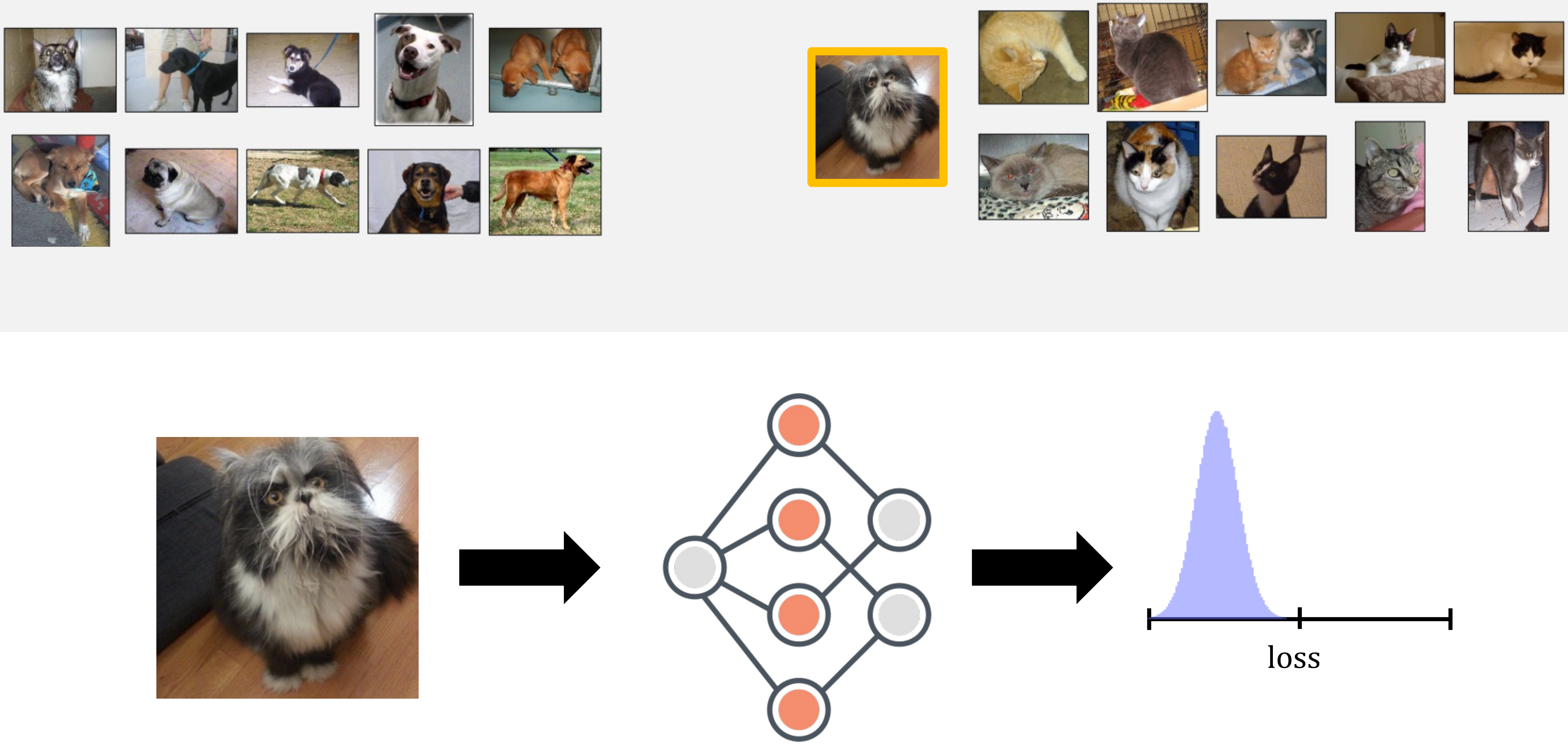


# Training set

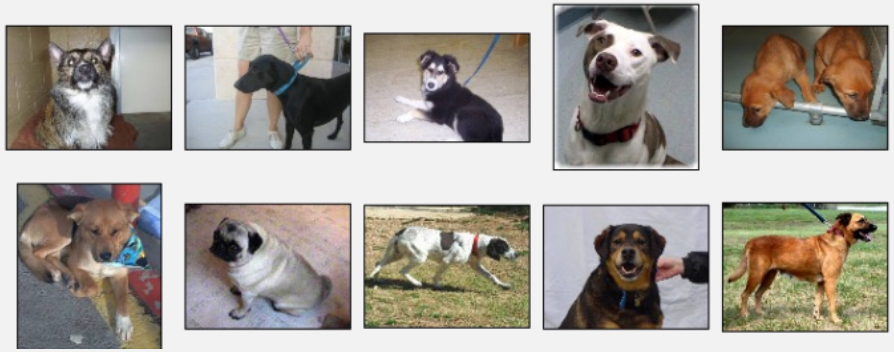


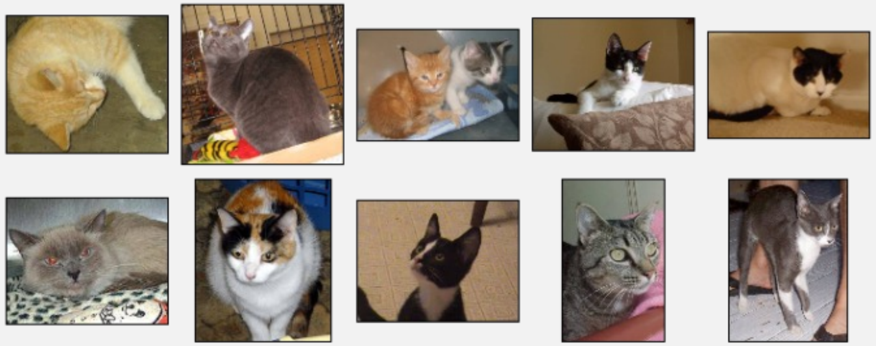


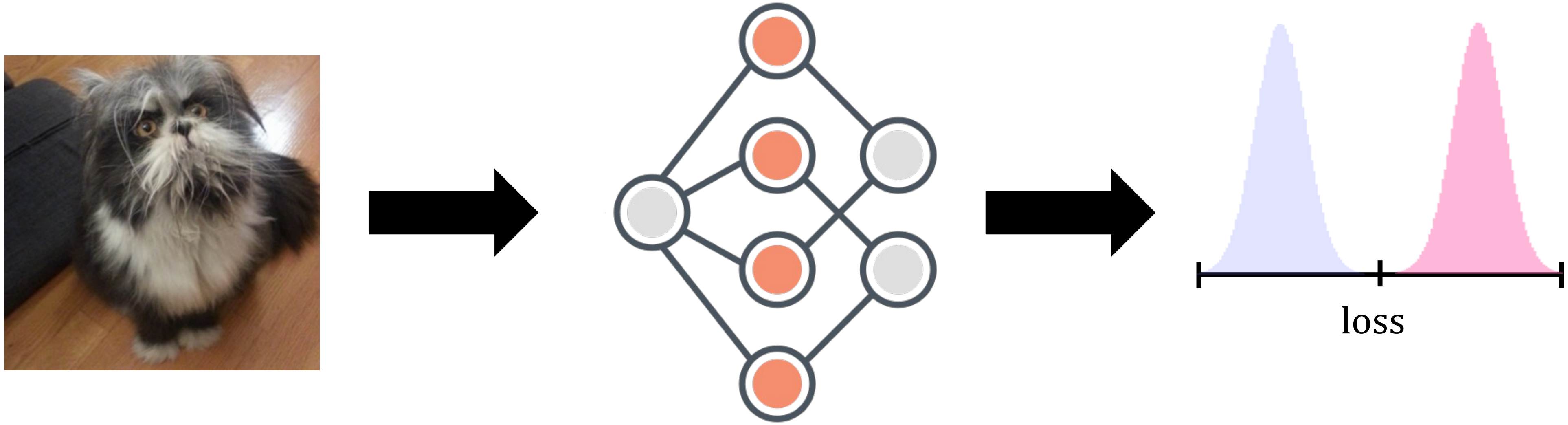
# Training set



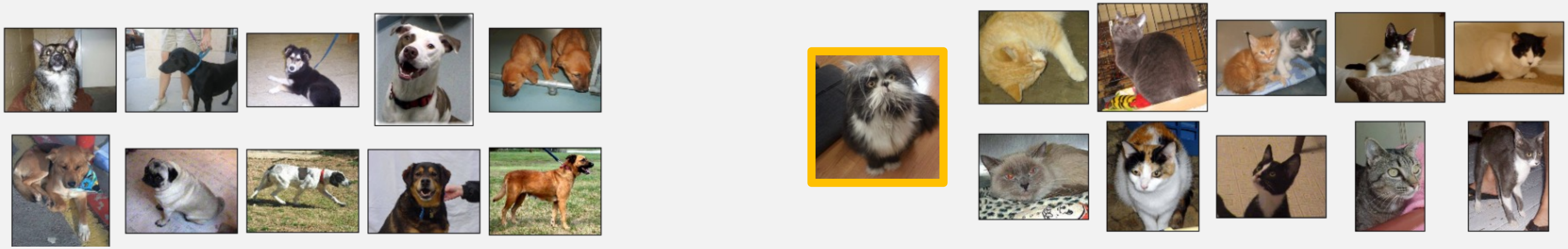
# Training set

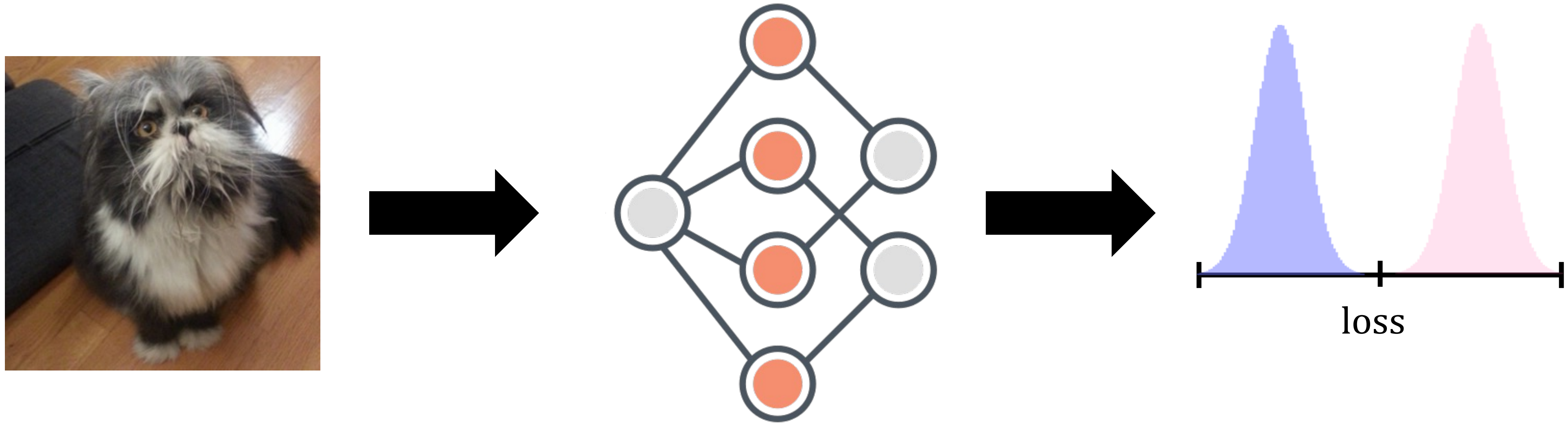




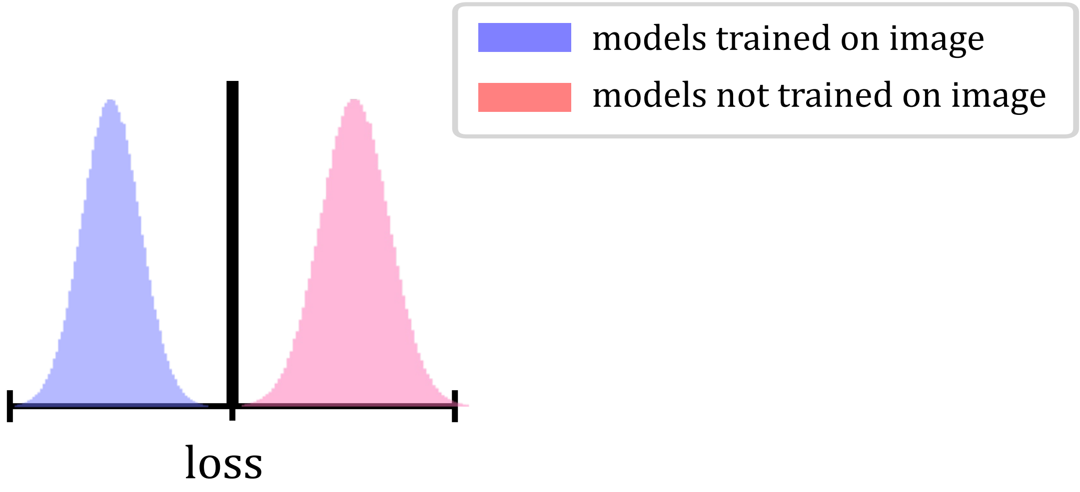


# Training set





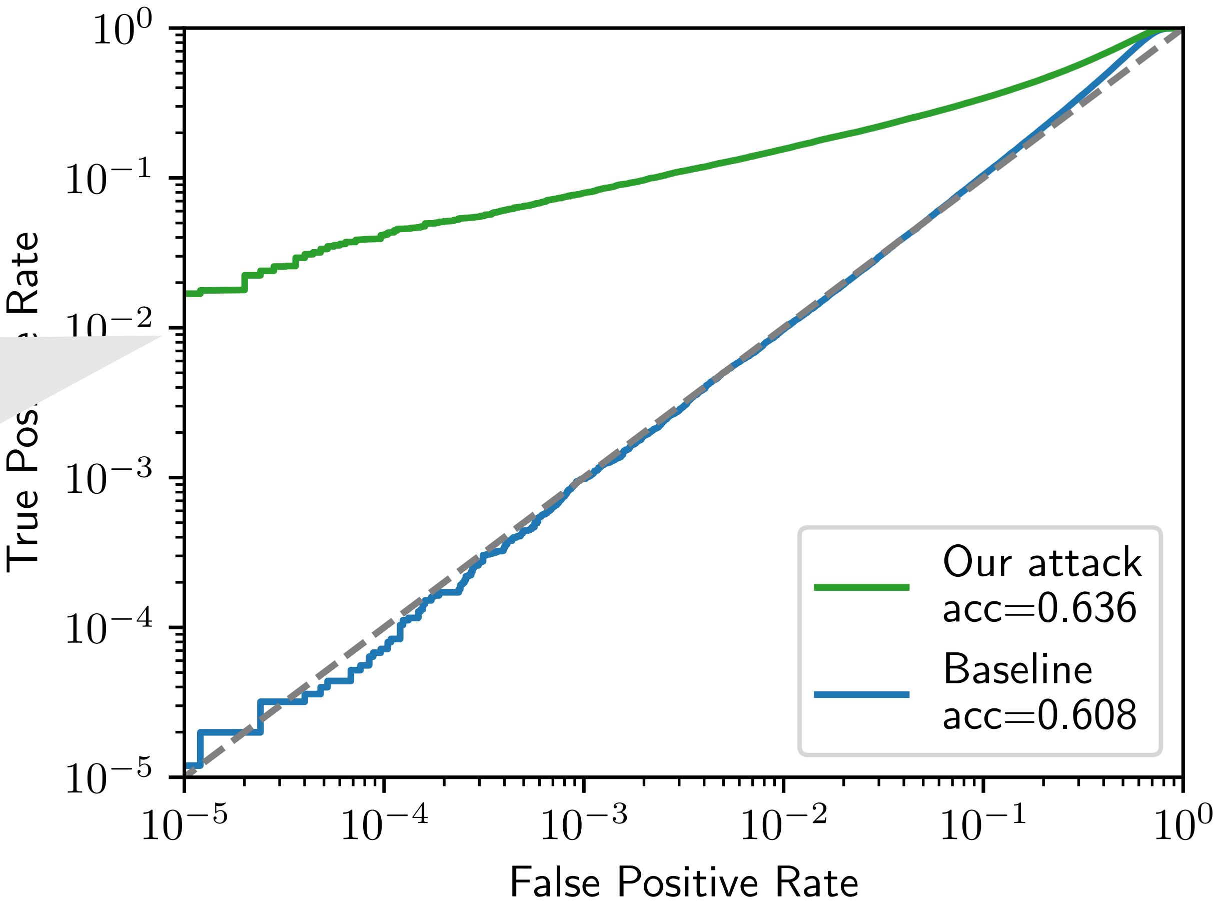
# Some examples are easier to distinguish.



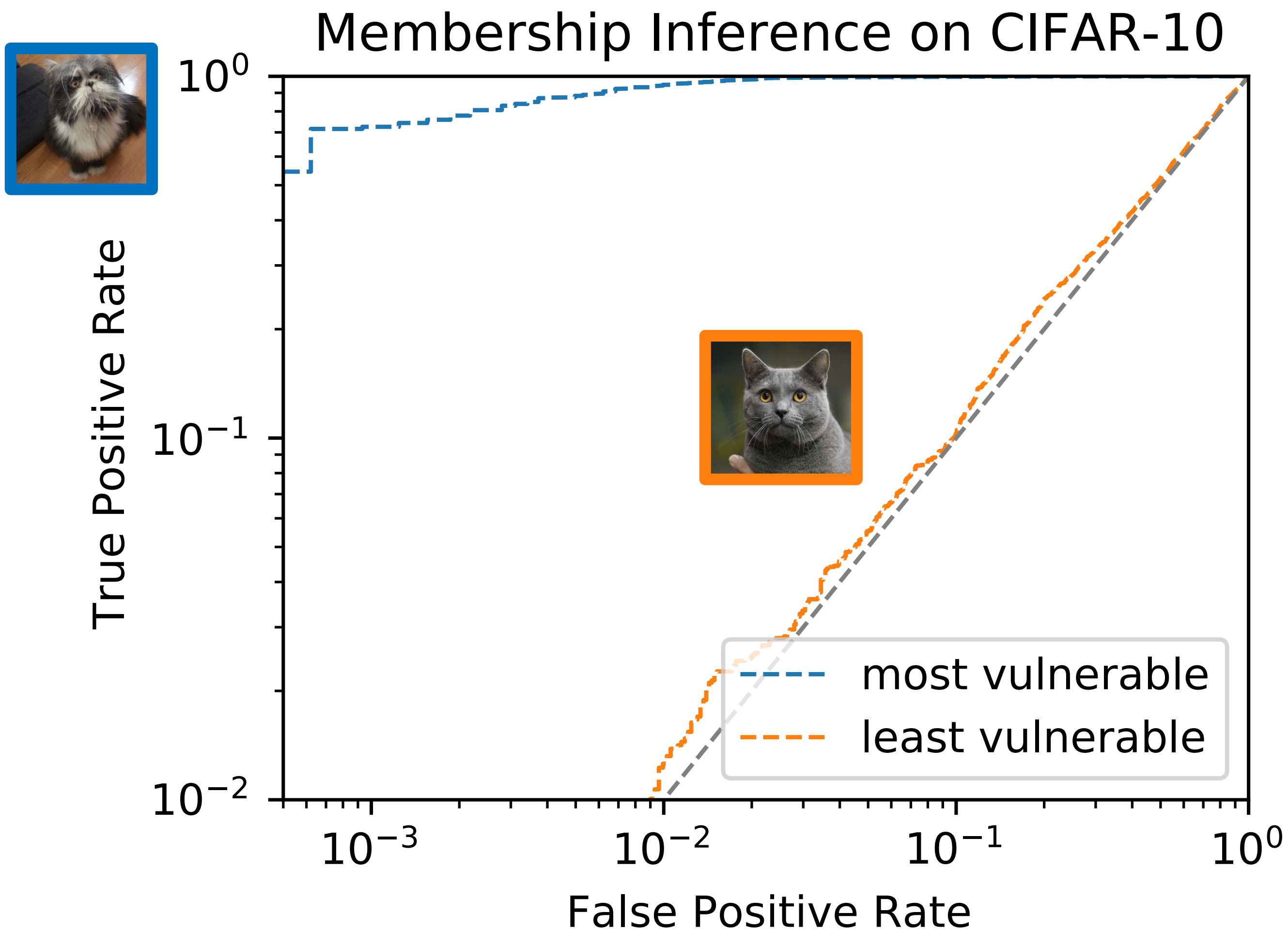
# Membership inference with *per-example likelihood*

**>1000x better in the *worst case***

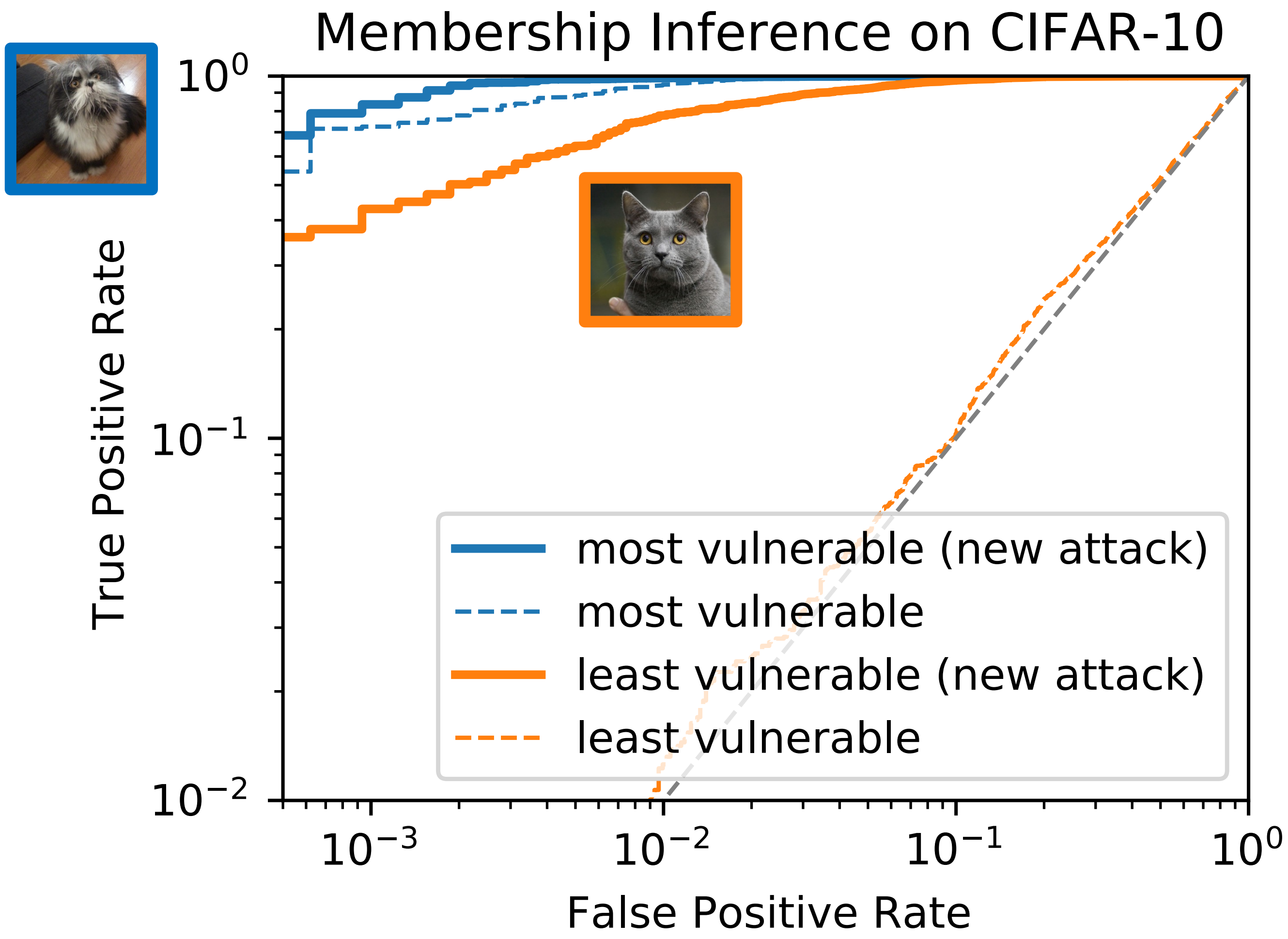
(thanks to Gaussian fitting + numeric stability + multiple queries + ...)



# Membership inference works well on *“outliers”*.



# Next: a new attack that works on *any example!*



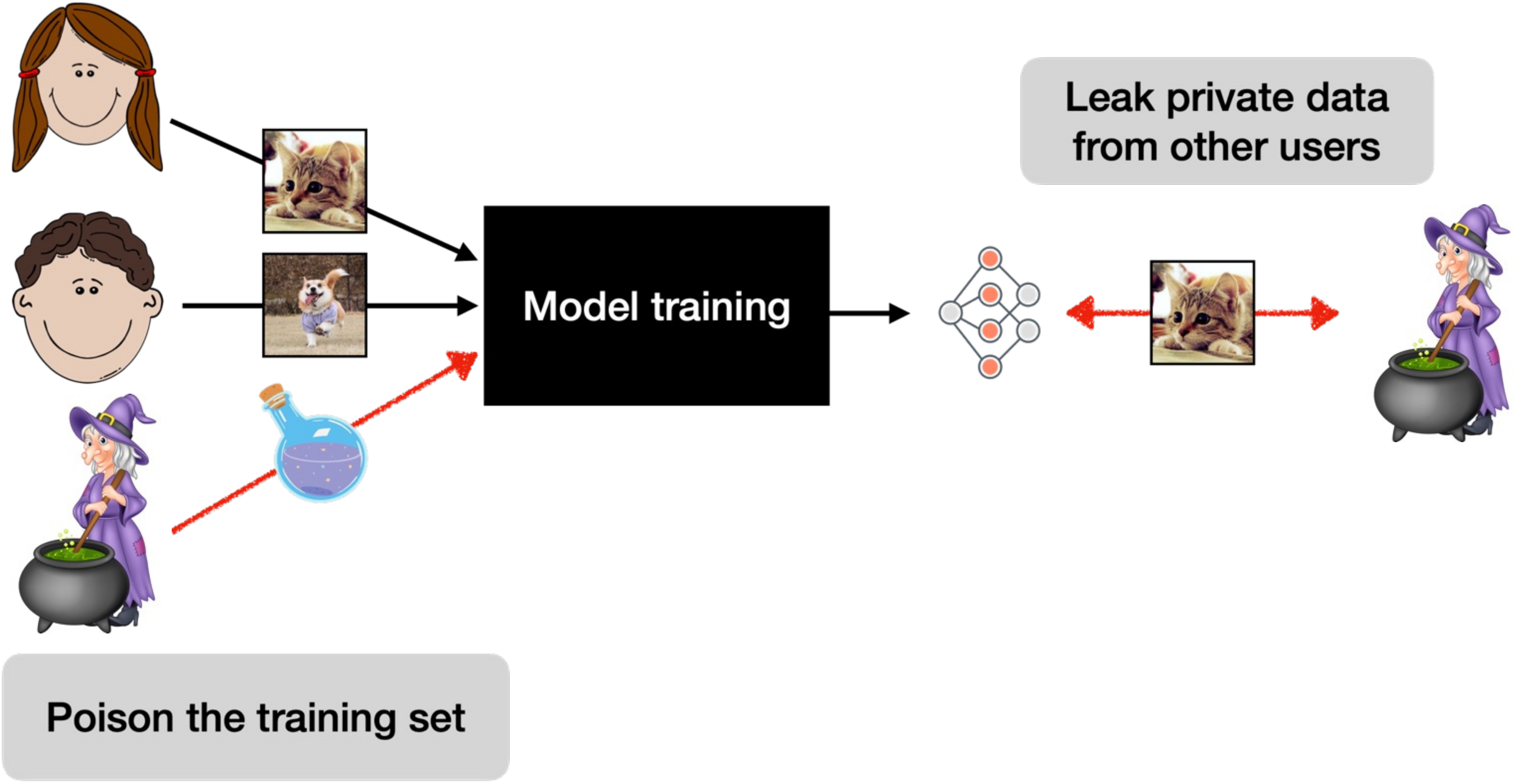
# Idea: use *data poisoning*

**Yann LeCun and Yoshua Bengio: Self-supervised learning is the key to human-level intelligence**

| Dataset | # English Img-Txt Pairs |
|---|---|
| **Public Datasets** | |
| MS-COCO | 330K |
| CC3M | 3M |
| Visual Genome | 5.4M |
| WIT | 5.5M |
| CC12M | 12M |
| RedCaps | 12M |
| YFCC100M | 100M[2] |
| **LAION-5B (Ours)** | **2.3B** |

# A new threat model: *privacy poisoning*

Tramèr et al. "Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets", CCS '22



# *Data poisoning* can create *“fake” outliers.*


dog

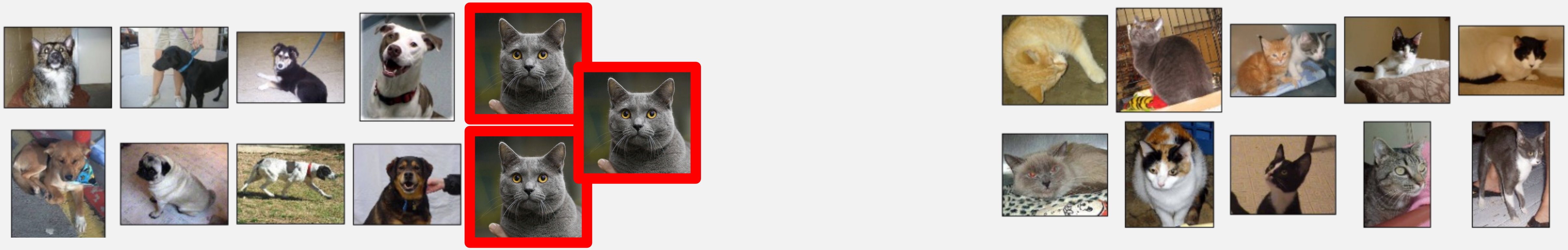

dog

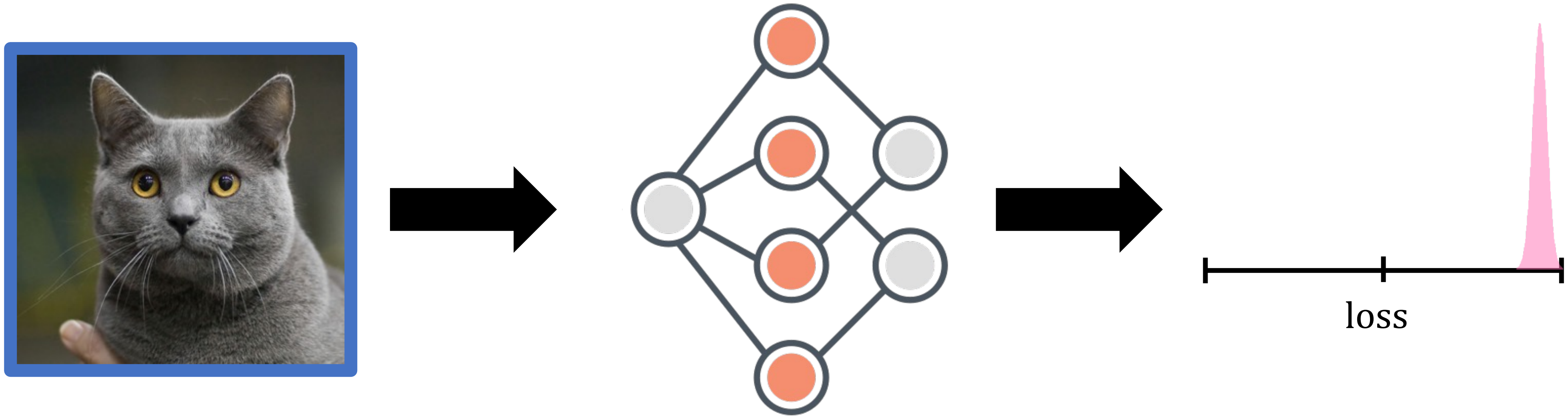

cat


dog

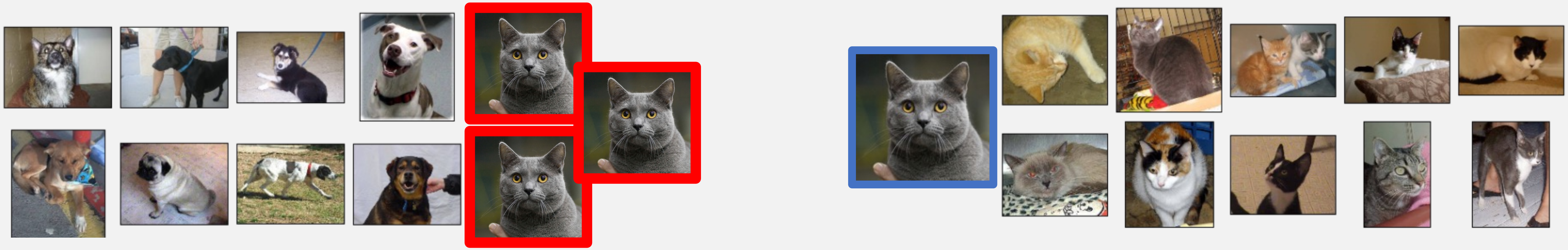

dog

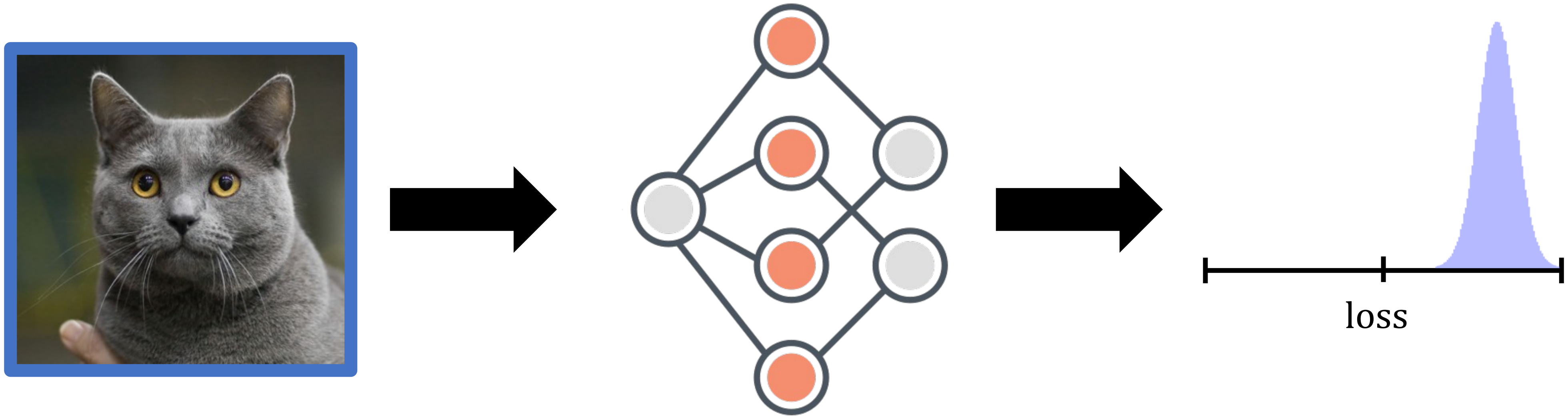
# Training set



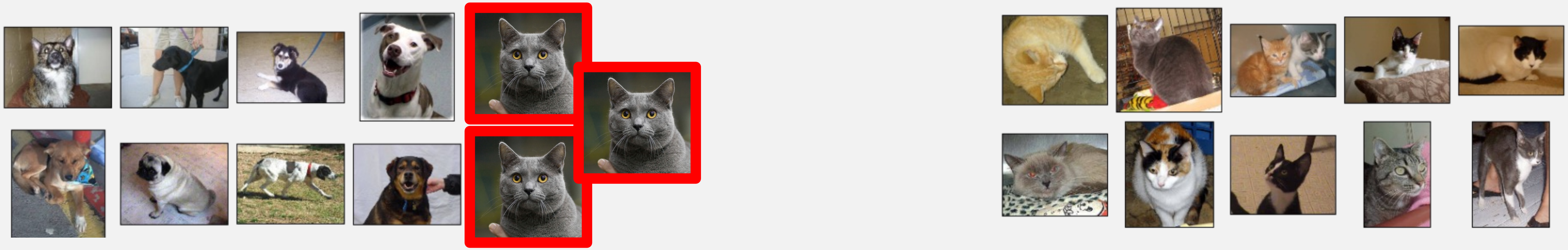


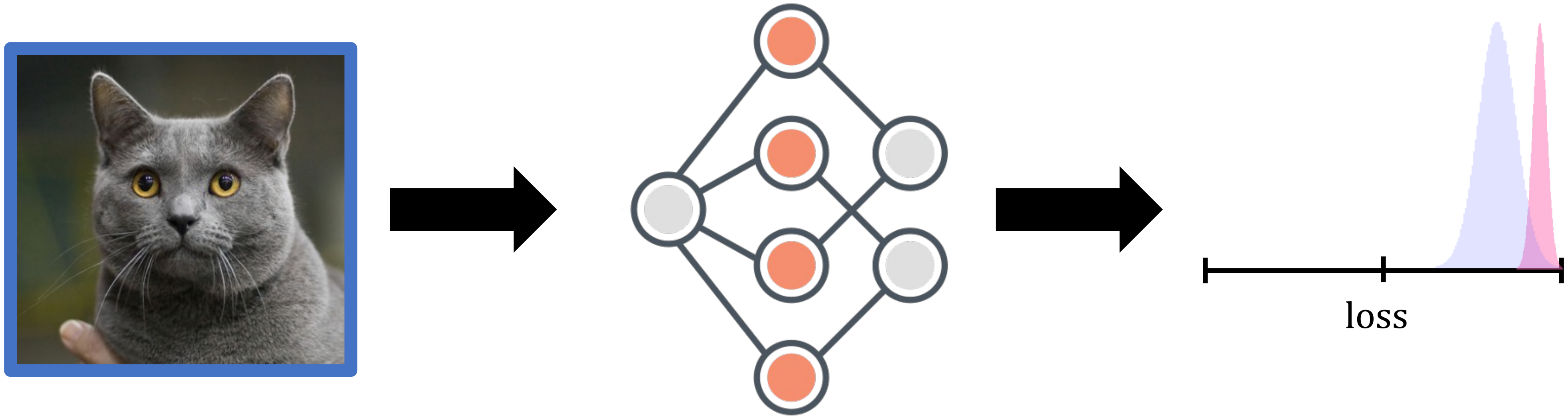
# Training set



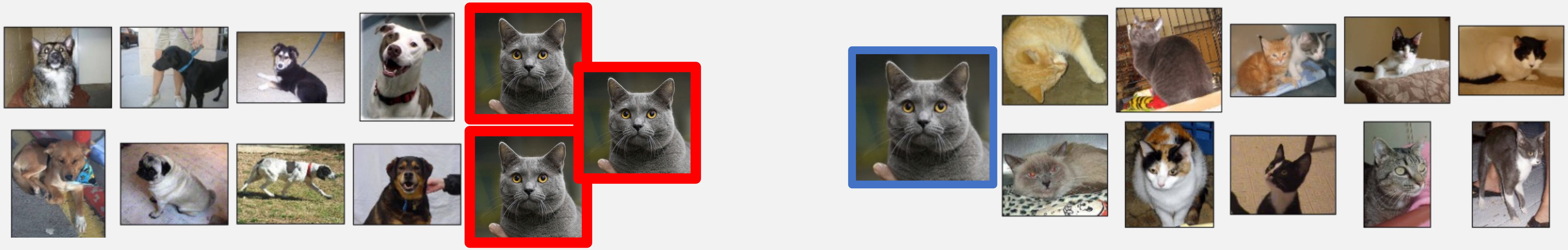


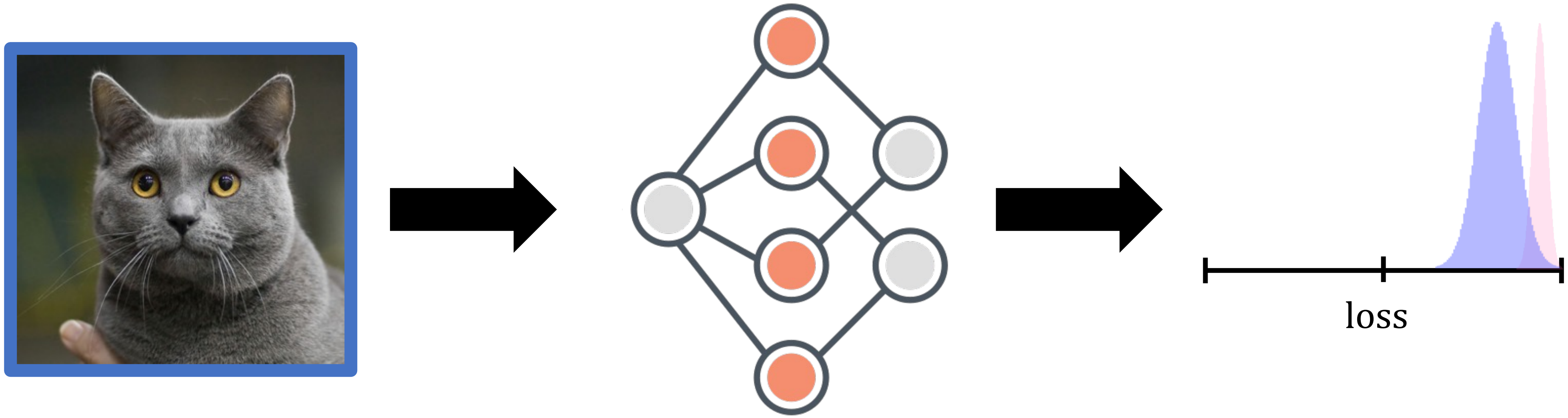
# Training set

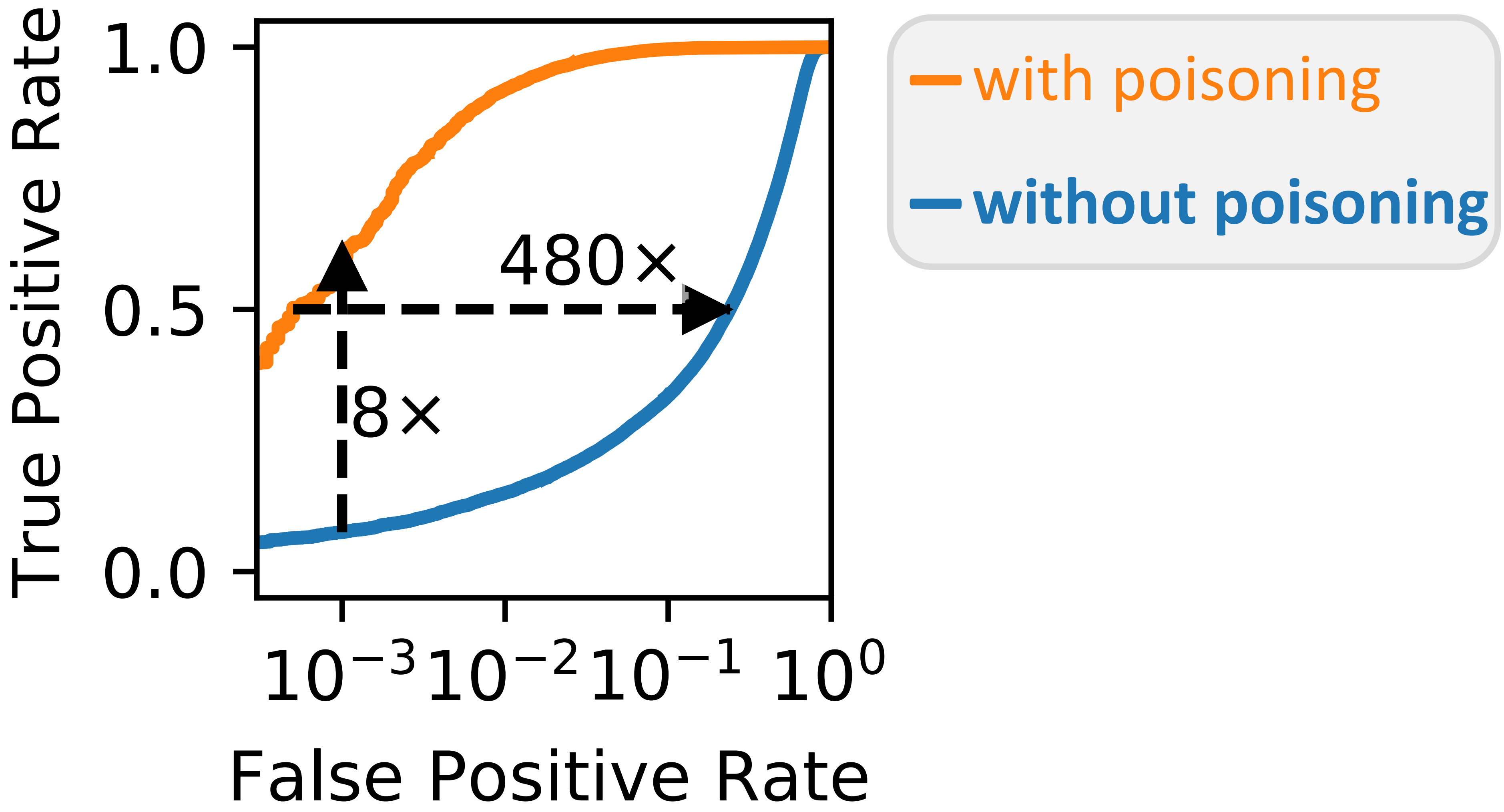




# Training set

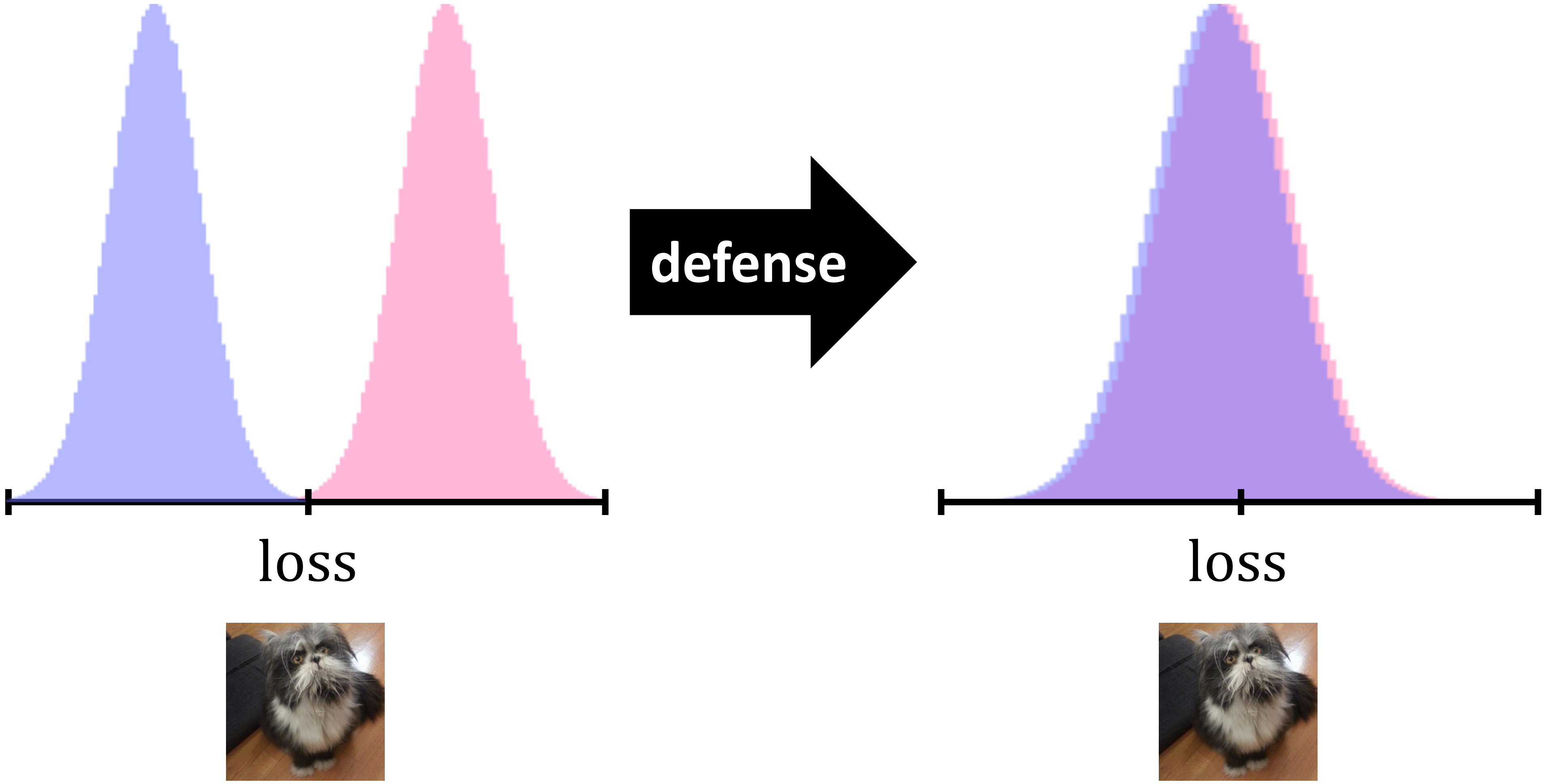






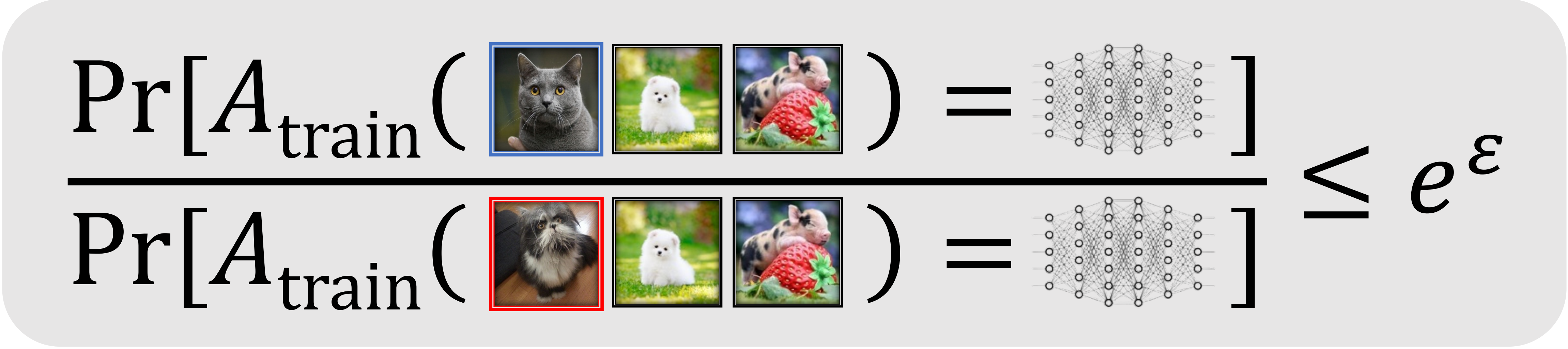

with targeted poisoning of **<0.1%** of the CIFAR-10 training set

# How to defend against membership leakage?



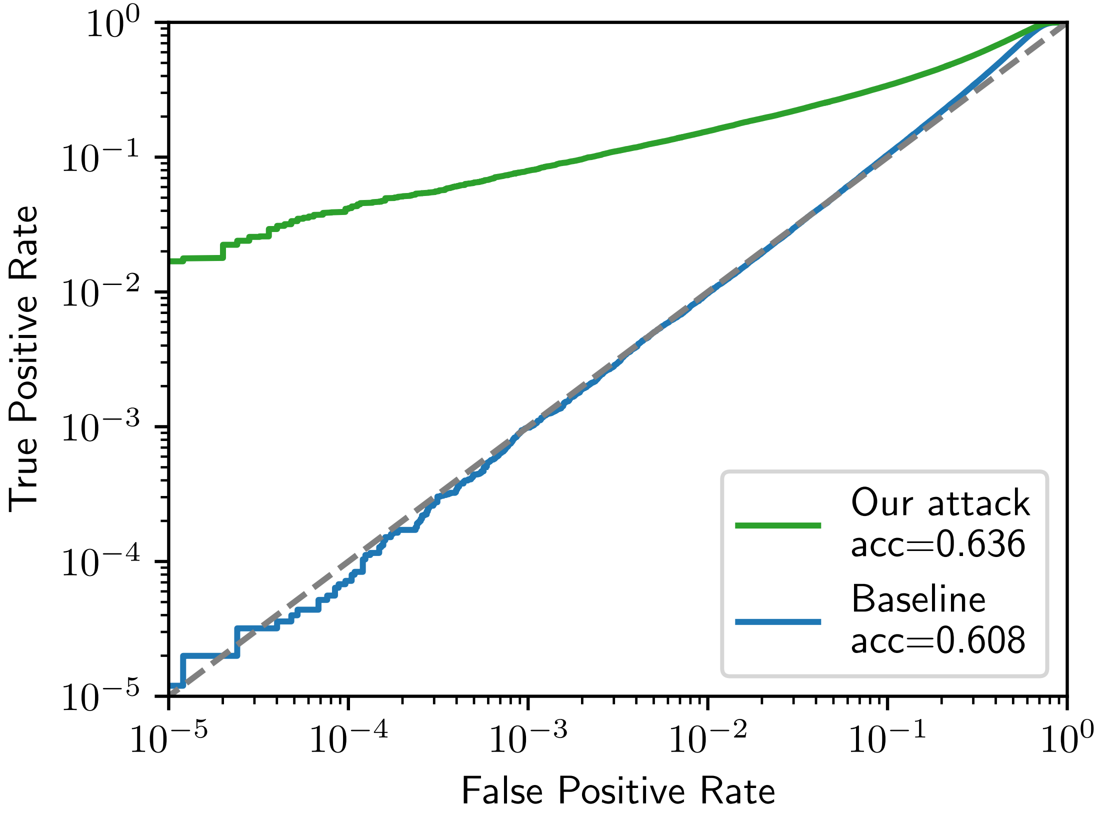
# Differential privacy prevents all our attacks.

*DP guarantee holds for* ***any*** *pair of datasets that differ in* ***any*** *single element*



# DP bounds the success of *any* MI attack.

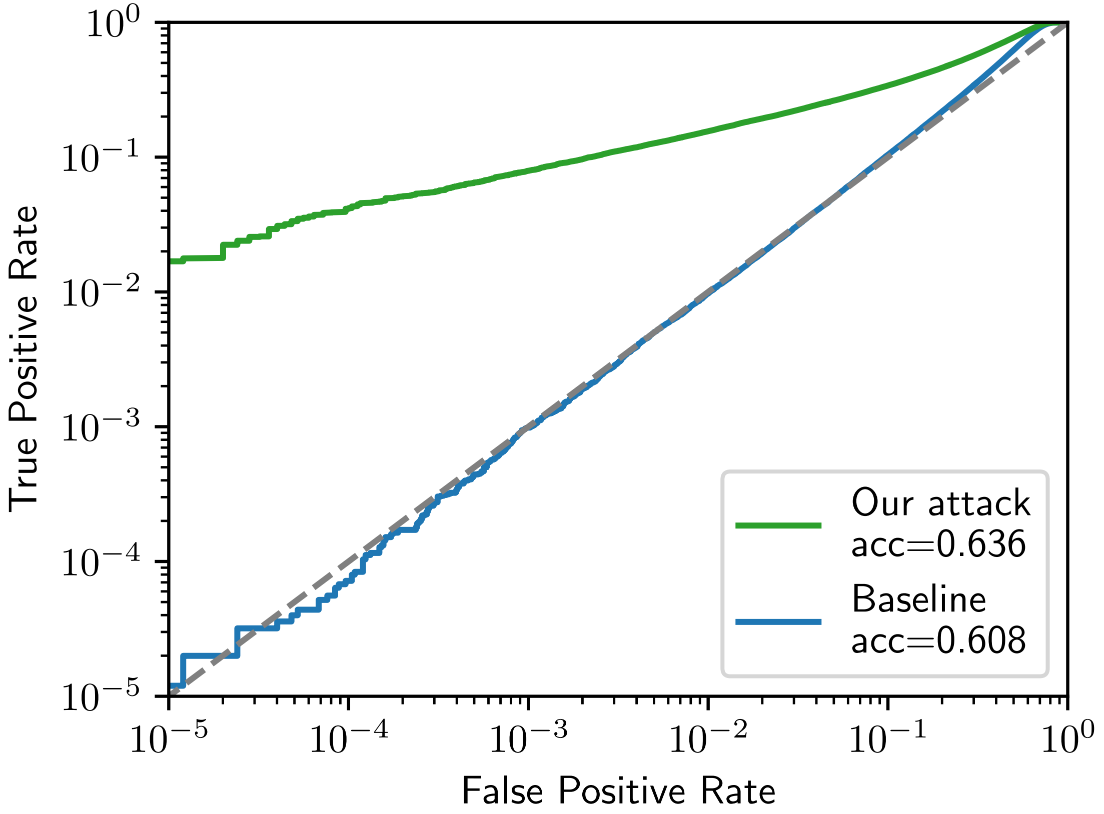
[Kairouz et al. ‘15]




$$\frac{TPR}{FPR} \leq e^{\varepsilon}$$

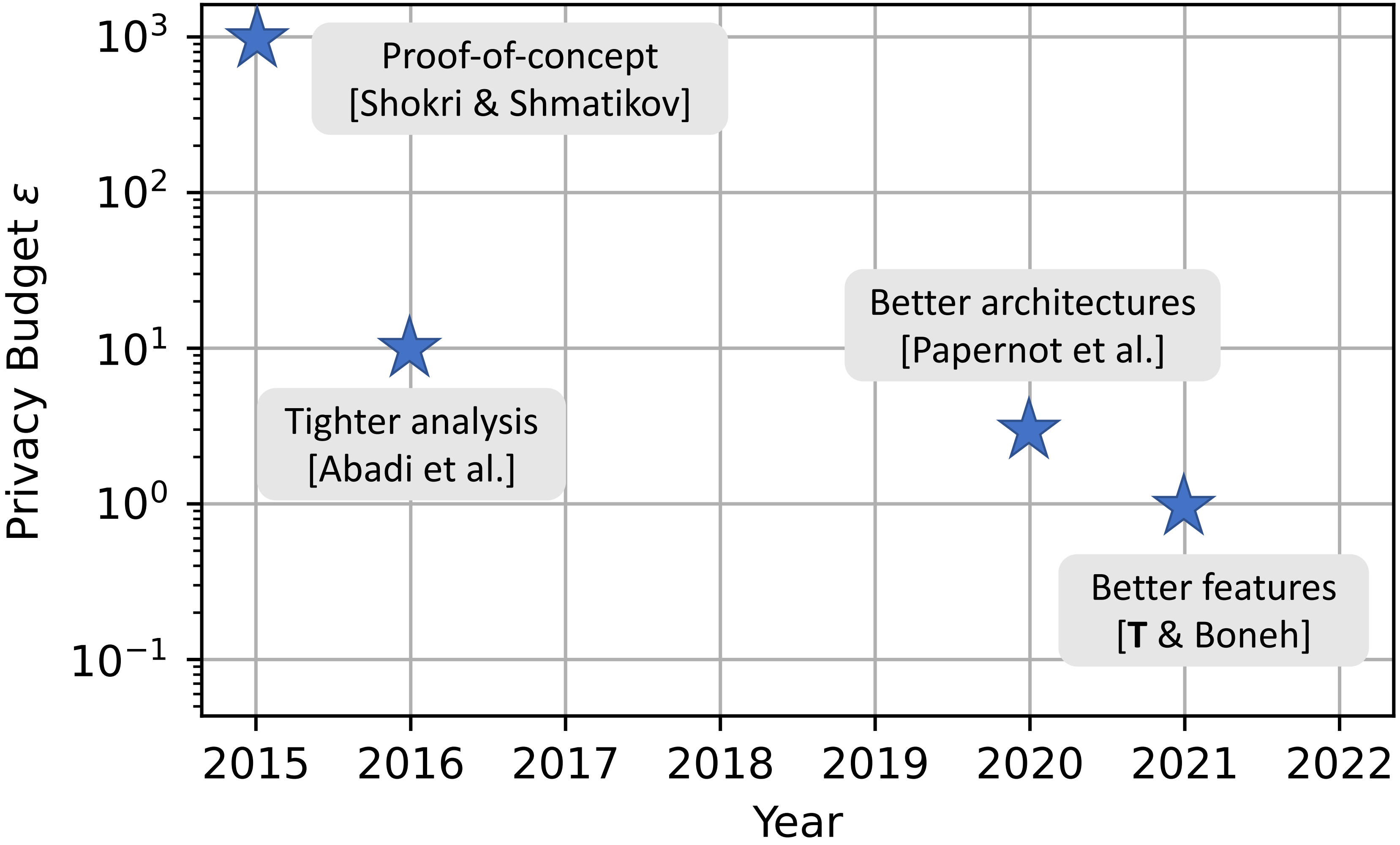
# Corollary: MI attacks can be used to *audit* privacy.

[Jagielsky et al. '20, Nasr et al. '21]

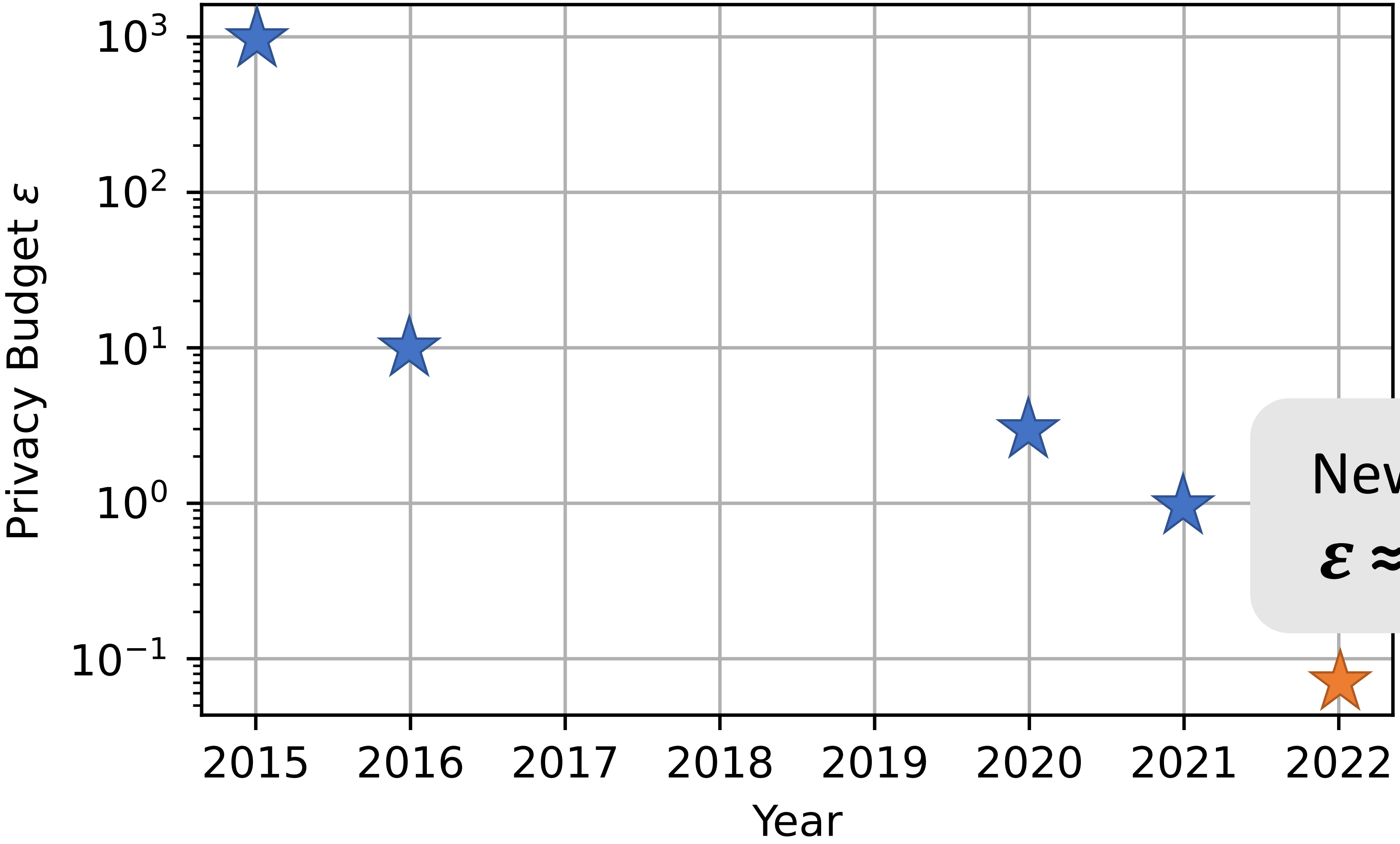



$$e^{\varepsilon} \geq \frac{TPR}{FPR}$$

# Example: DP with 98% accuracy on MNIST

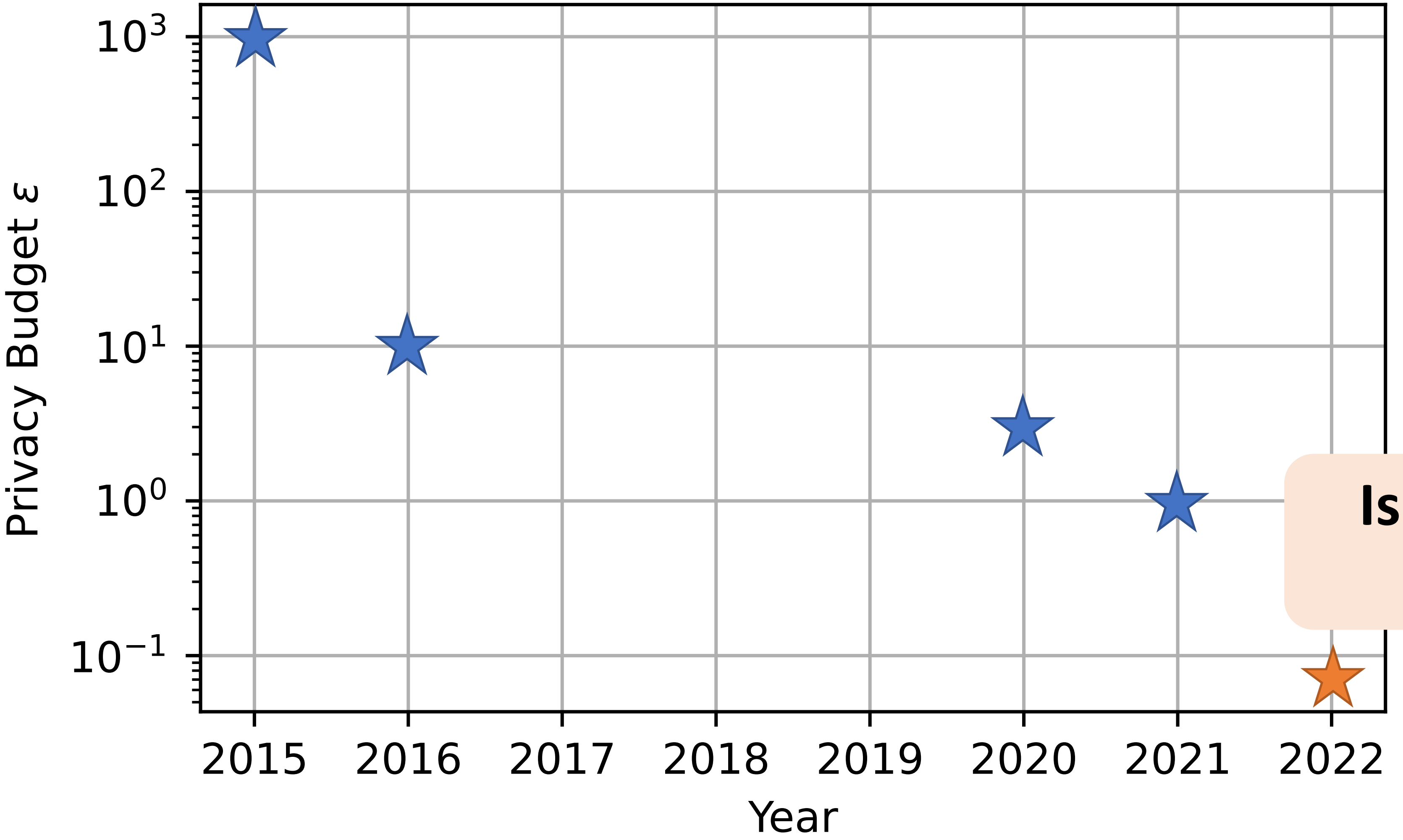


# Example: DP with 98% accuracy on MNIST




New algorithm
**$\varepsilon \approx$ 0.08 (!!!)**

# Example: DP with 98% accuracy on MNIST




**Is this claim *correct*?**

# How to verify a DP claim?

➢ Check the **proof**

$$
\begin{aligned}
&c(o_{1:k}; \mathcal{M}_{1:k}, o_{1:(k-1)}, d, d') \\
&\quad = \log \frac{\Pr[\mathcal{M}_{1:k}(d; o_{1:(k-1)}) = o_{1:k}]}{\Pr[\mathcal{M}_{1:k}(d'; o_{1:(k-1)}) = o_{1:k}]} \\
&\quad = \log \prod_{i=1}^{k} \frac{\Pr[\mathcal{M}_i(d) = o_i \mid \mathcal{M}_{1:(i-1)}(d) = o_{1:(i-1)}]}{\Pr[\mathcal{M}_i(d') = o_i \mid \mathcal{M}_{1:(i-1)}(d') = o_{1:(i-1)}]} \\
&\quad = \sum_{i=1}^{k} \log \frac{\Pr[\mathcal{M}_i(d) = o_i \mid \mathcal{M}_{1:(i-1)}(d) = o_{1:(i-1)}]}{\Pr[\mathcal{M}_i(d') = o_i \mid \mathcal{M}_{1:(i-1)}(d') = o_{1:(i-1)}]} \\
&\quad = \sum_{i=1}^{k} c(o_i; \mathcal{M}_i, o_{1:(i-1)}, d, d').
\end{aligned}
$$

Thus

$$
\begin{aligned}
&\mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[ \exp(\lambda c(o'_{1:k}; \mathcal{M}_{1:k}, d, d')) \mid \forall i < k \colon o'_i = o_i \right] \\
&\quad = \mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[ \exp\left( \lambda \sum_{i=1}^{k} c(o'_i; \mathcal{M}_i, o_{1:(i-1)}, d, d') \right) \right] \\
&\quad = \mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[ \prod_{i=1}^{k} \exp\left( \lambda c(o'_i; \mathcal{M}_i, o_{1:(i-1)}, d, d') \right) \right] \\
&\qquad\qquad \text{(by independence of noise)}
\end{aligned}
$$

# How to verify a DP claim?

- Check the proof
- Check the **code**

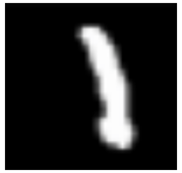
```
def process_microbatch(i, sample_state):
  """Process one microbatch (record) with privacy helper."""
  microbatch_loss = tf.reduce_mean(
      input_tensor=tf.gather(microbatches_losses, [i]))
  with gradient_tape.stop_recording():
    grads = gradient_tape.gradient(microbatch_loss, var_list)
  sample_state = self._dp_sum_query.accumulate_record(
      sample_params, sample_state, grads)
  return sample_state

for idx in range(self._num_microbatches):
  sample_state = process_microbatch(idx, sample_state)

grad_sums, self._global_state, _ = (
    self._dp_sum_query.get_noised_result(sample_state,
                                         self._global_state))
```

# How to verify a DP claim?

- Check the proof
- Check the code
- Launch an **attack**!

# DP bounds should hold for **any** data point.

$D_0 =$  … 

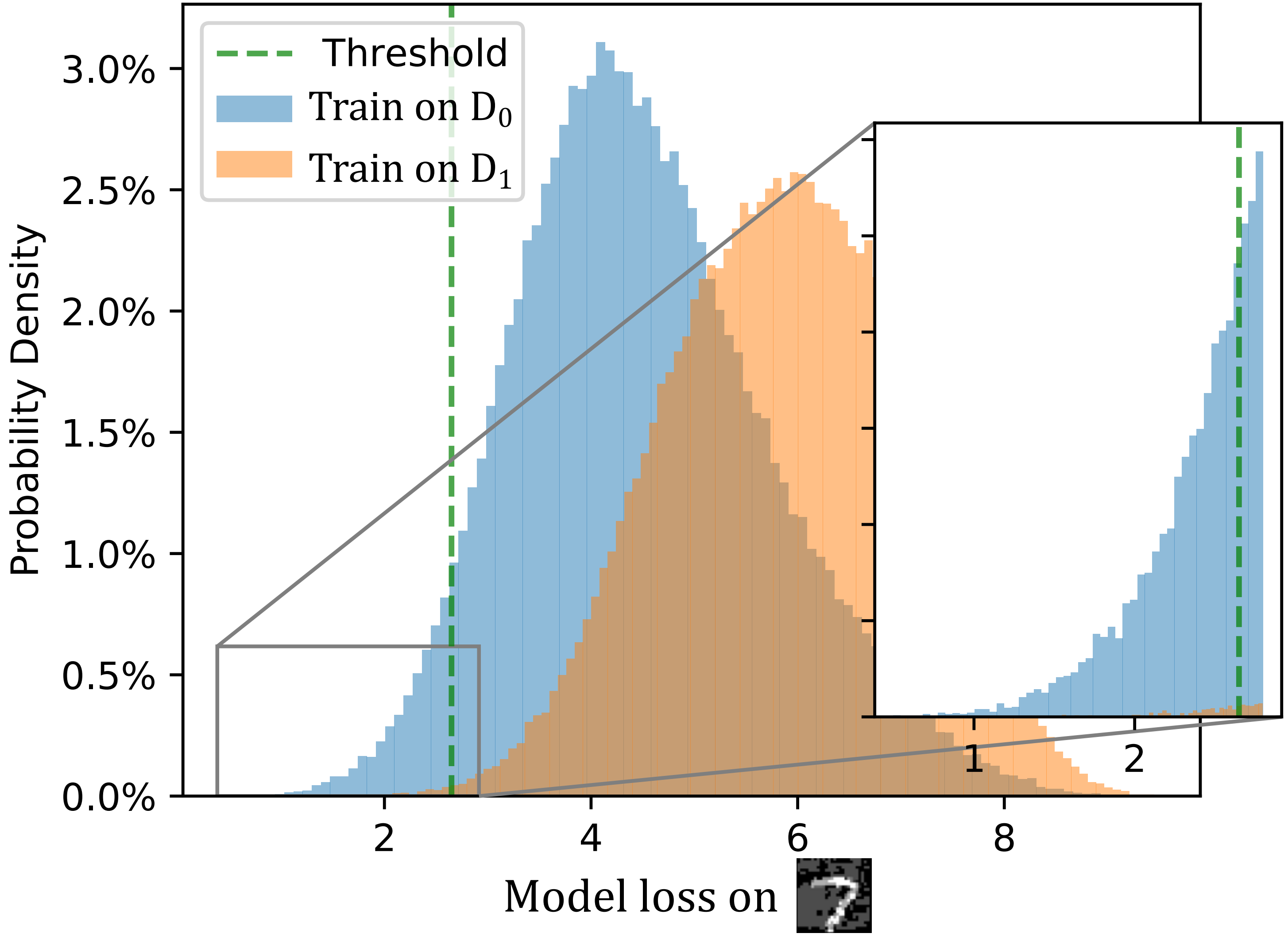
$D_1 =$  … 

*worst-case*
*out-of-distribution*
*data point*

**Attack goal:** guess if  is a member of the training set

# Run the attack 100'000 times...

Tramèr et al. "Debugging Differential Privacy: A Case Study for Privacy Auditing", 2022




$$e^{\varepsilon} \geq \frac{TPR}{FPR}$$

$$\boldsymbol{\varepsilon > 2.7}$$

(claim was $\varepsilon = 0.08$)

# Conclusion

- Average-case leakage is a poor metric for privacy!
- We must reevaluate what we “know” about MI attacks & defenses
- Poisoning can turn average-case inputs into worst-case inputs
- Worst-case MI attacks are a useful tool for catching DP bugs