Machine Learning Model Hardening

•••

For Fun and Profit

May 12, 2018

Ariel Herbert-Voss
@adversariel

What is this about?

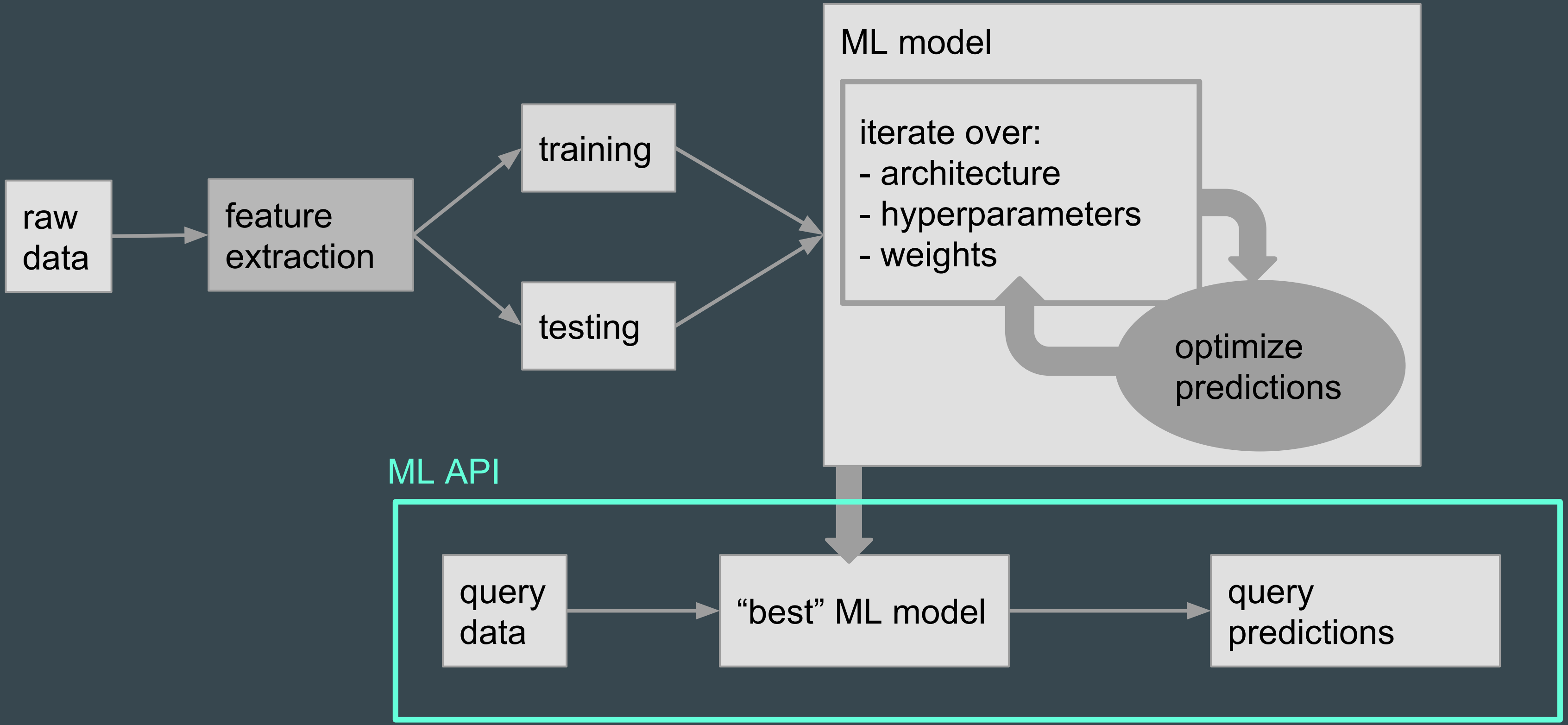
Most industry uses of machine learning are either deployed on-site or provide API access

It is not a good idea to implement a vanilla ML model API with no model hardening

For simplicity, we’re talking about black box access to neural network-based machine learning models (but some of these attacks can be generalized)

This talk does not assume deep familiarity with ML - surface understanding is ok :)

Machine Learning pipeline

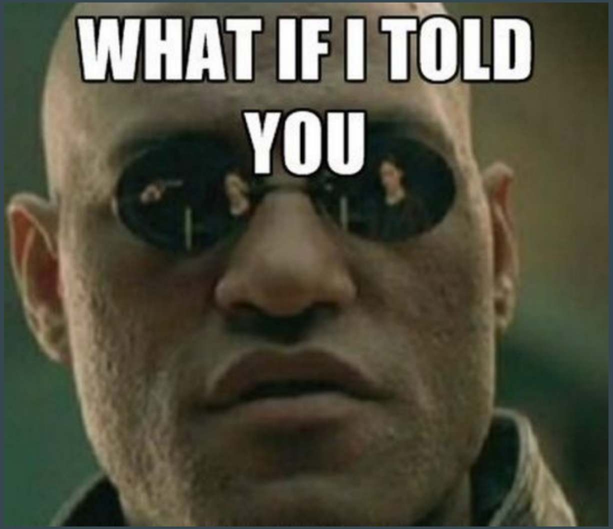


Threats versus Solutions

Attack	data	model	predictions
Adversarial examples			X
Model inversion	X	X	
Memorization	X	X	
Model theft		X	

Solutions:

- Homomorphic encryption
- Secure multiparty encryption
- Differential privacy



Homomorphic encryption & Secure multi-party computation

Homomorphic encryption: can perform computations on encrypted information

- Adversary can’t read data but we still preserve statistical structure
- Fully homomorphic encryption schemes are incredibly slow

Secure multi-party computation: multiple parties can jointly compute a function while keeping the function input private

- Cheaper than homomorphic encryption but requires more interaction between parties
- Have to redefine operators and functions

Differential privacy

Adding or removing an element from the data doesn't change the output distribution very much

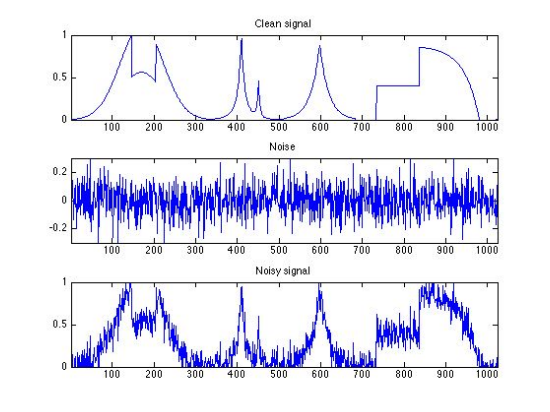
- Also very slow, BUT
- Even works in scenarios where adversary has full knowledge of training mechanisms and access to parameters

[Dwork, 2006]

Differential privacy

How do we do it?

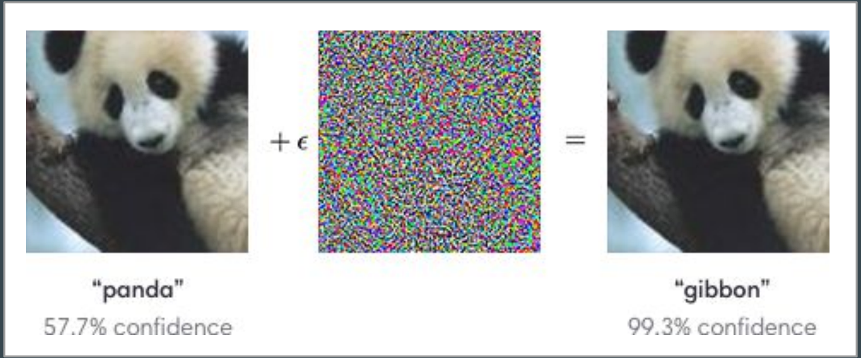
1. Add noise to the output
2. Keep track of how many data access requests are granted



[Dwork and Roth, 2015]

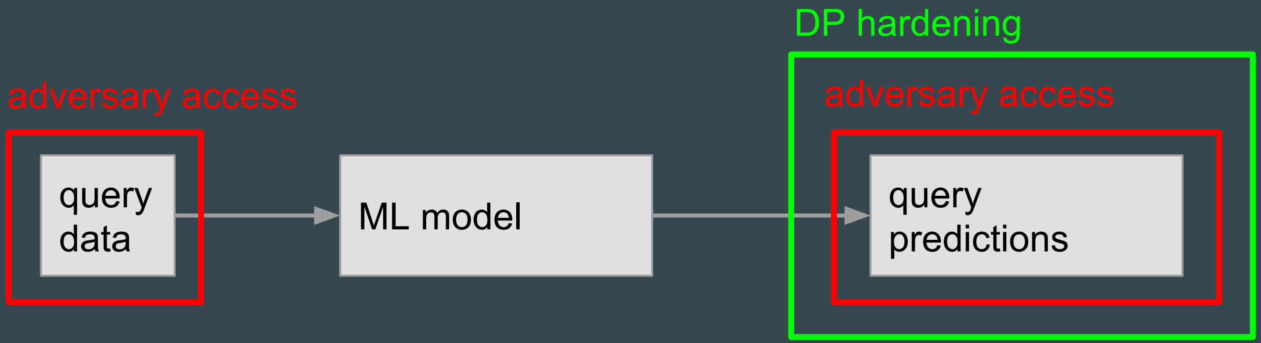
Adversarial examples

Give some slightly perturbed input to get incorrect predictions



[Goodfellow et al, 2015]

Adversarial examples



DP hardening

adversary access

query data

ML model

adversary access

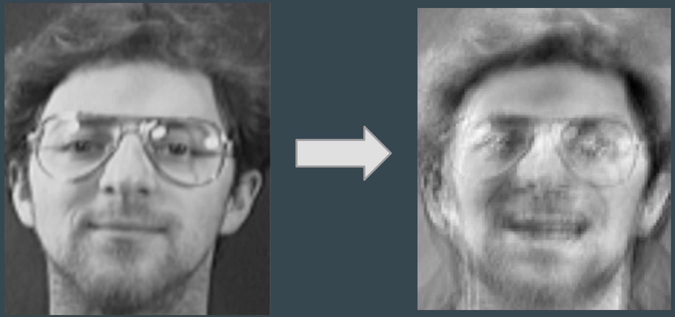
query predictions

50% dog
50% fried chicken

Model inversion

Given a categorization model/API that provides confidence values and predictions, we can recover information encoded in the model through the training data

Scenario: adversary has somebody’s name and wants to get an image of that person out of a facial recognition API

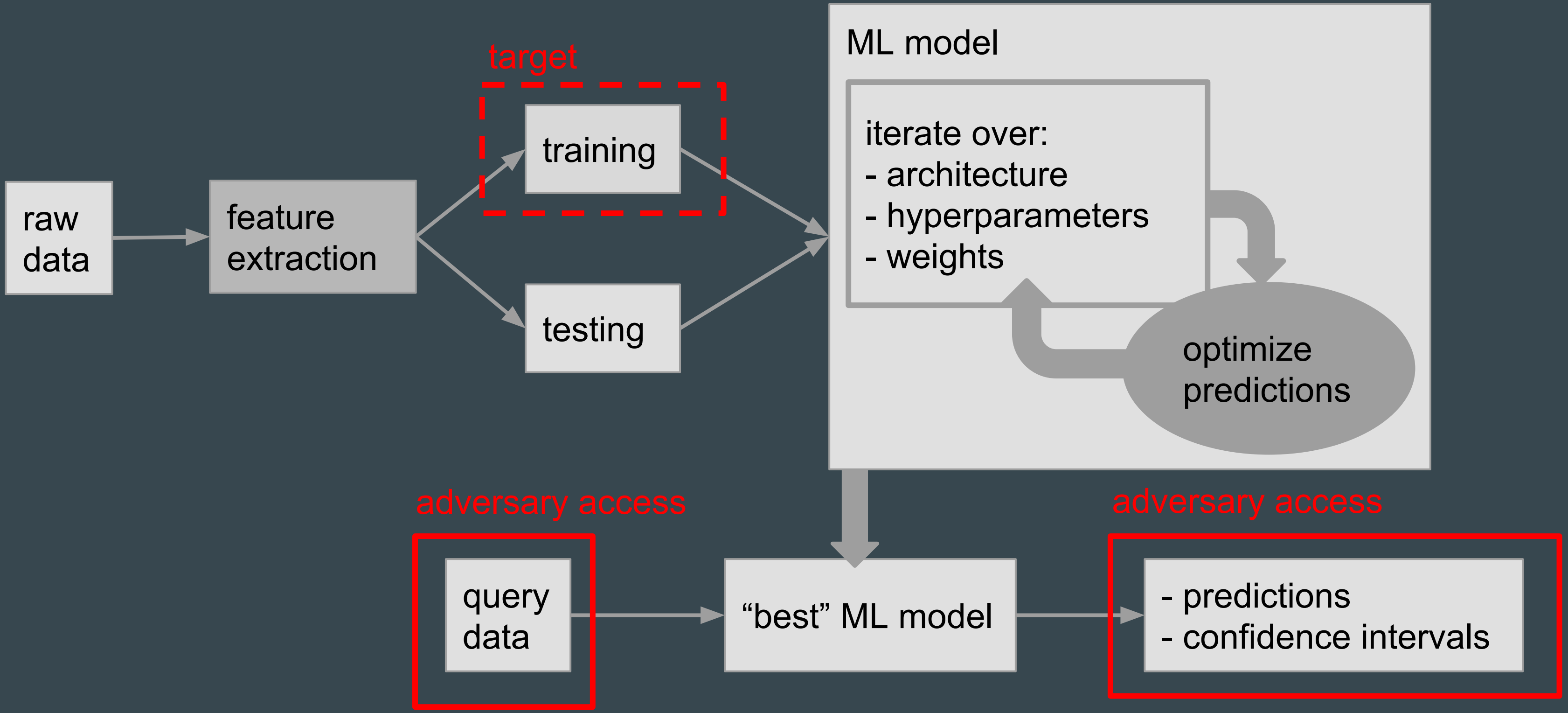


training image

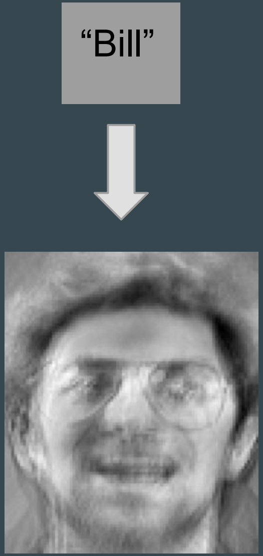
recovered image

[Fredrikson et al, 2015]

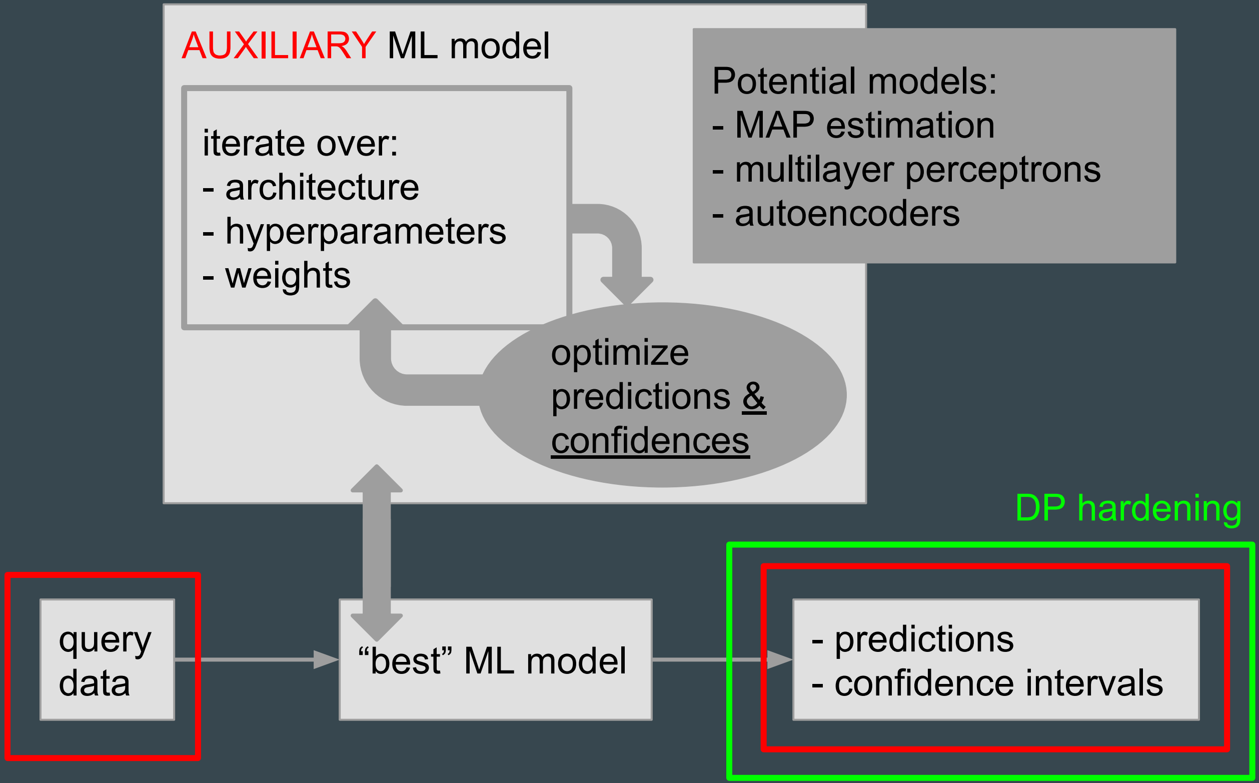
Model inversion



Model inversion



“Bill”



AUXILIARY ML model

iterate over:
- architecture
- hyperparameters
- weights

optimize predictions & confidences

Potential models:
- MAP estimation
- multilayer perceptrons
- autoencoders

query data

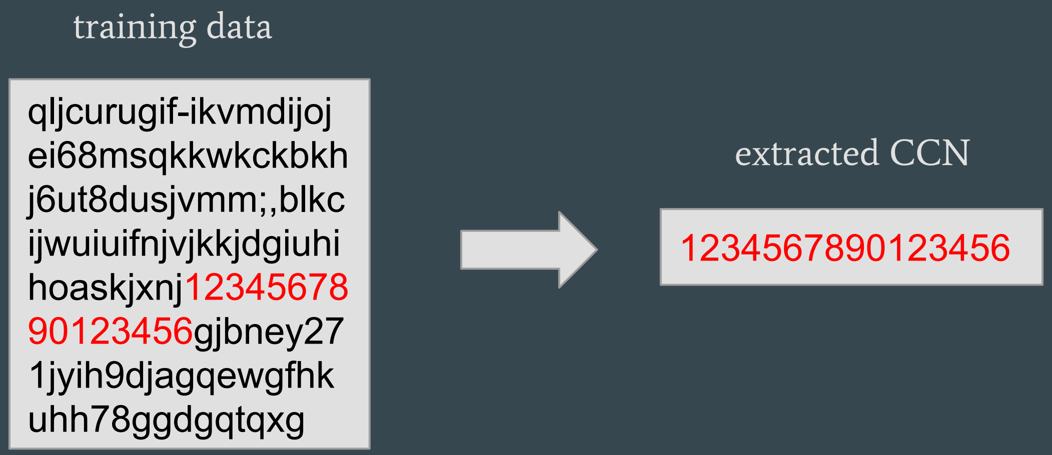
“best” ML model

- predictions
- confidence intervals

DP hardening

Memorization

Given a known data format like a credit card number we can extract this information by using a search algorithm on the model predictions



training data

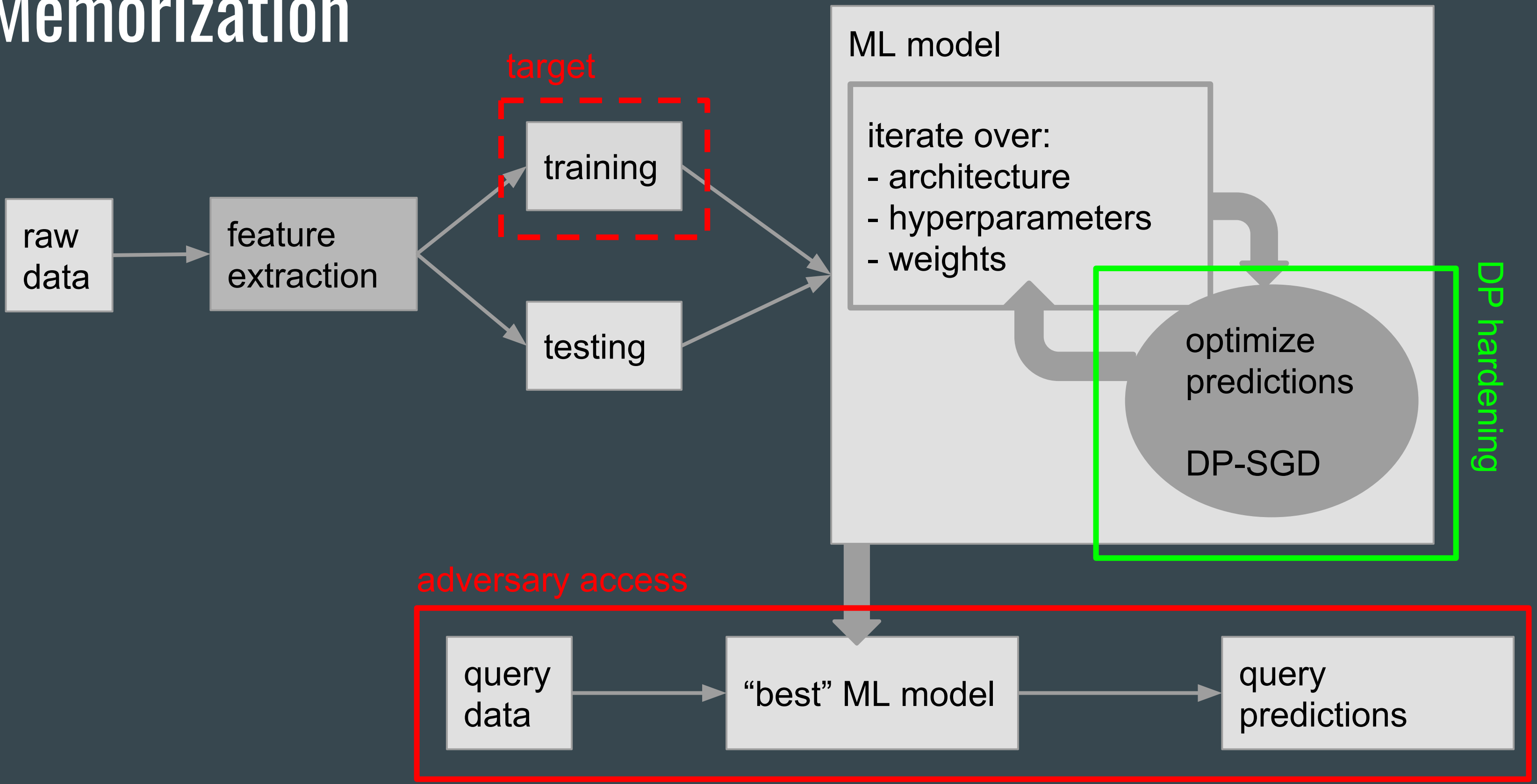
qljcurugif-ikvmdijoj
ei68msqkkwkckbkh
j6ut8dusjvmm;,blkc
ijwuiuifnjvjkkjdgiuhi
hoaskjxnj12345678
90123456gjbney27
1jyih9djagqewgfhk
uhh78ggdgqtqxg

extracted CCN

1234567890123456

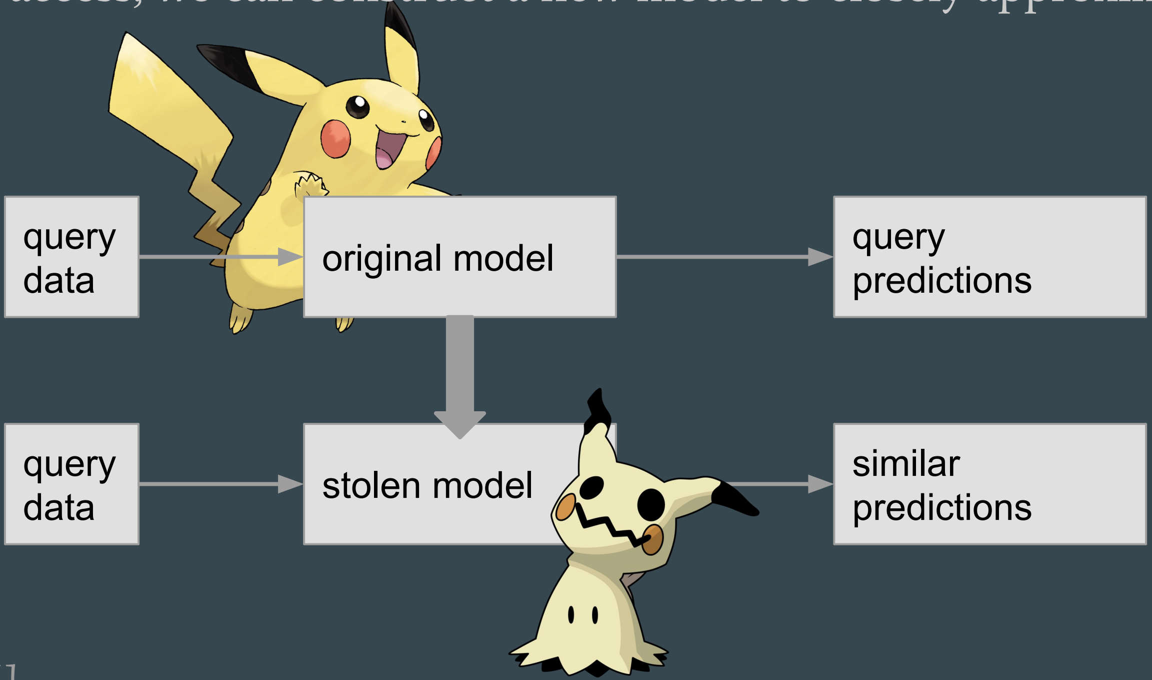
[Carlini et al, 2018]

Memorization



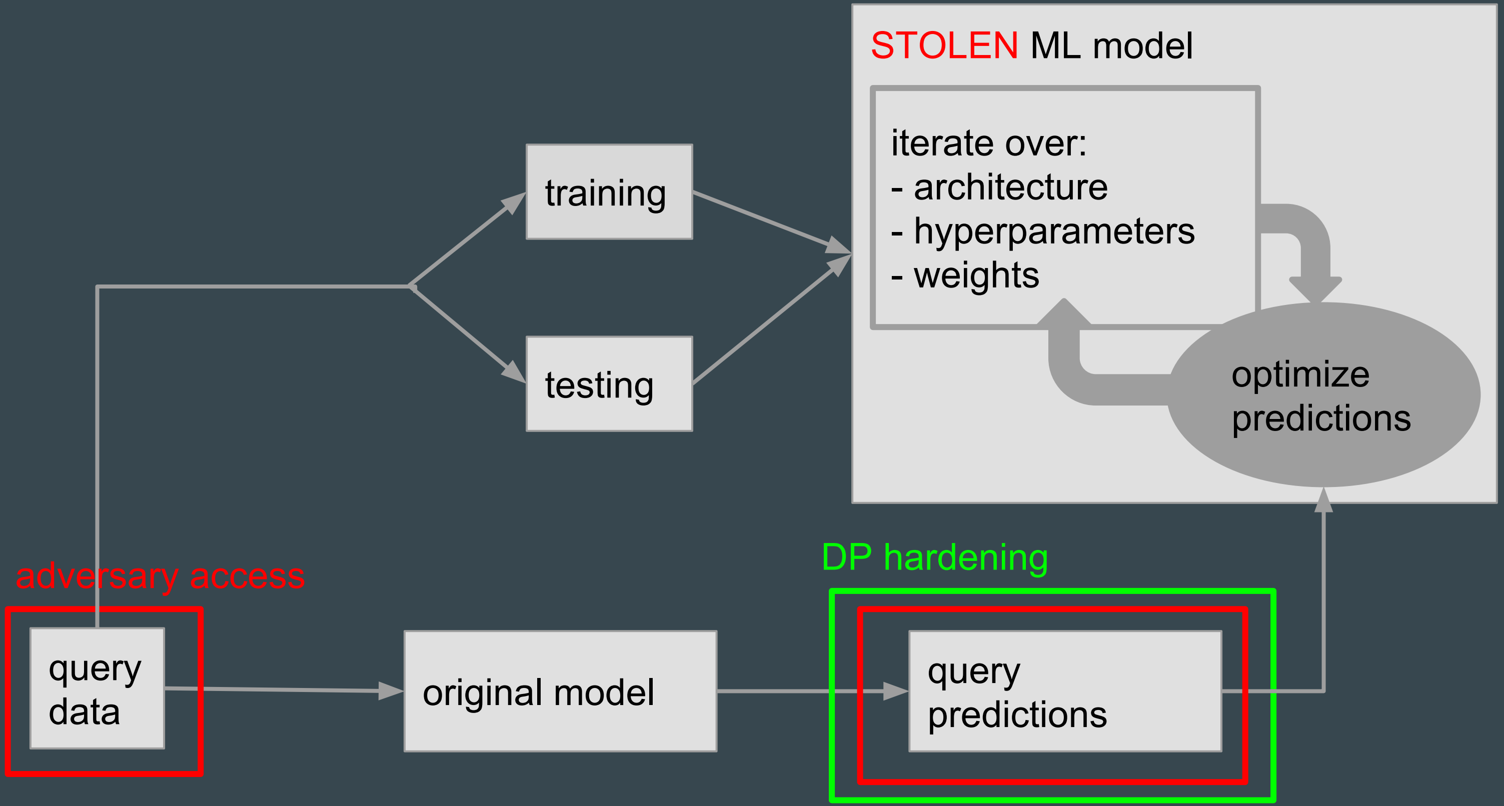
Model theft

Given black box access, we can construct a new model to closely approximate target



[Tramèr et al, 2016]

Model theft


STOLEN ML model
iterate over:
- architecture
- hyperparameters
- weights
optimize predictions
training
testing
adversary access
query data
original model
DP hardening
query predictions

Other hardening methods

Defensive distillation

- train secondary model with flatter gradients

Deep k-Nearest Neighbors

- kNN can tell you if a data point might be adversarial based on where it falls on the model's training manifold

Ensemble adversarial training

- add perturbations to the training data by transferring from pretrained models

[Papernot et al, 2016]
[Papernot et al, 2018]
[Tramèr et al, 2018]

General observations

Although some techniques work as white box augmentations, it is more practical to think about model hardening from the perspective of black box access

Most attacks are trying to get at information held in the model

Notice these attacks still work even if the data is encrypted

- they rely on the preservation of statistical relationships within the data, which is NOT obfuscated by most cryptographic techniques

Practical take-away summary slide

Hardening tips:

- Give users the bare minimum amount of information
- Add some noise to output predictions
- Restrict users from making too many prediction queries
- Consider using an ensemble of models and return aggregate predictions

So far differential privacy is the most reliable method of model hardening

For more information

Differential privacy:

- [Dwork, 2006]: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/dwork.pdf
- [Dwork and Roth, 2014]: http://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf

Adversarial examples:

- [Goodfellow et al, 2015]: https://arxiv.org/pdf/1412.6572.pdf
- [Papernot et al, 2016]: https://arxiv.org/pdf/1605.07277.pdf

Model inversion:

- [Fredrikson et al, 2015]: https://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf
- [Xu et al, 2016]: http://www.cs.cmu.edu/~mfredrik/papers/wfjn_csf16.pdf

Memorization:

- [Carlini et al, 2018]: https://arxiv.org/pdf/1802.08232.pdf

Model theft:

- [Tramèr et al, 2016]: https://arxiv.org/pdf/1609.02943.pdf

Other hardening methods:

- [Papernot et al, 2016]: https://arxiv.org/pdf/1511.04508.pdf
- [Papernot et al, 2018]: https://arxiv.org/pdf/1803.04765.pdf
- [Tramèr et al, 2018]: https://arxiv.org/pdf/1705.07204.pdf

Feel free to contact me via Twitter/ email or come grab a beer with me at the Con :)