Building robust machine learning applications

Alex Serban
Radboud University, SIG, LIACS


ML algorithmic robustness

A ML algorithm is said to be robust if:

- When tested on training samples and on a similar test samples, the performance is close
- When tested on a samples with noise or when the test distribution shifts the performance is close to the training performance

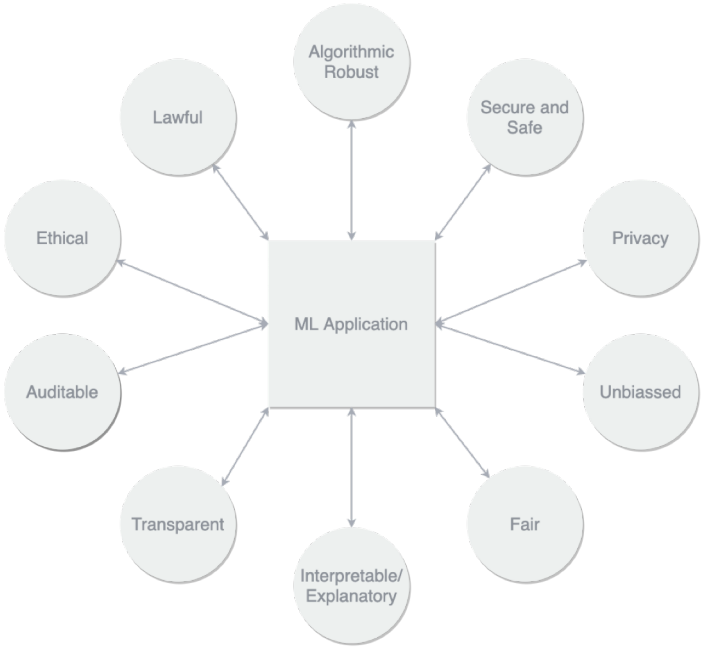
*(there are multiple methods to model noise)

Alex Serban
cs.ru.nl/~aserban

ML application robustness

We will consider a ML application robust if:

- It satisfies multiple properties such as algorithmic robustness, security or privacy (see diagram)
- Users are treated ethical and inclusive (according to legislation, if available)
- External actors can verify that it respects these properties (e.g., has transparent audits, provides explanations)

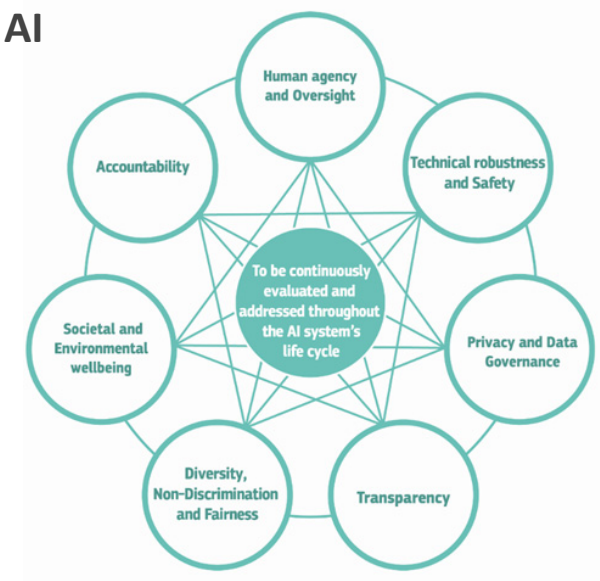


Alex Serban
cs.ru.nl/~aserban

Robustness – the big picture

Robustness is a pillar in the EU guidelines for trustworthy AI

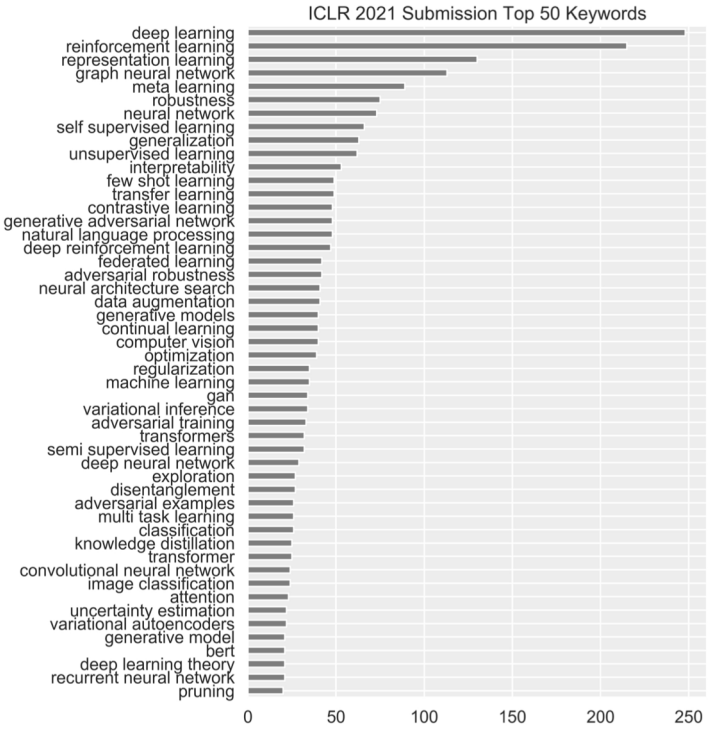
- The EU considers AI applications trustworthy if they are lawful, ethical and robust
- Robustness is tackled from a technical and social perspective (where social robustness is intertwined with ethical AI)



Algorithmic robustness in research

Robustness is an important research topic

- The picture on the right shows the top keywords from ICLR 2021 (a top conference in ML/DL)
- Robustness is sixth (although the first 3 keywords are very general)



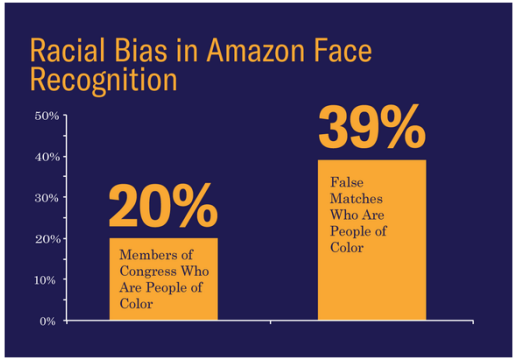
Alex Serban
cs.ru.nl/~aserban

Robustness in the wild

Figure 1: Natural adversarial examples from IMAGENET-A. The red text is a ResNet-50 prediction with its confidence, and the black text is the actual class. Many natural adversarial examples are incorrectly classified with high confidence, despite having no adversarial modifications as they are examples which naturally occur in the physical world.

Who to Sue When a Robot Loses Your Fortune

The first known case of humans going to court over investment losses triggered by autonomous machines will test the limits of liability.

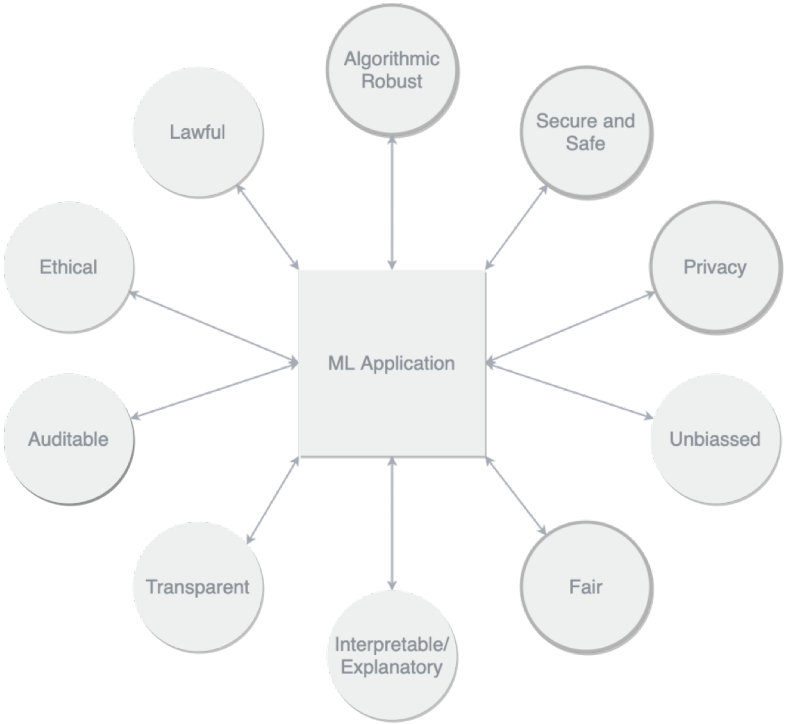


Nearly 40 percent of Rekognition's false matches in test were of people of color, even though they make up only 20 percent of Congress

Robustness today

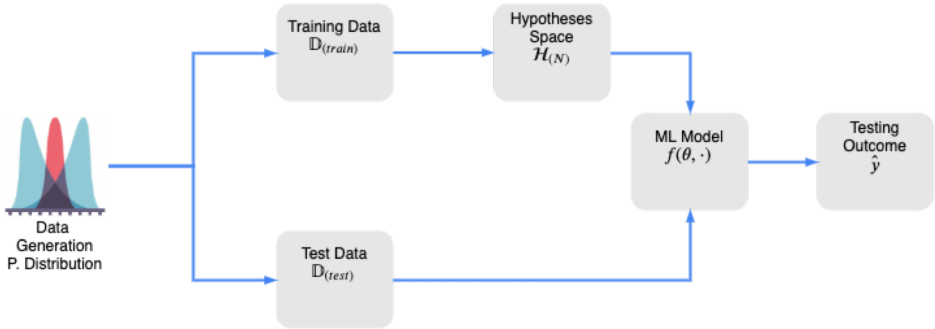
Today we will focus on:

- Robustness and security in the ML development life-cycle
- Risks and incident management for ML
- Privacy and fairness in ML (only briefly)



The ML Pipeline

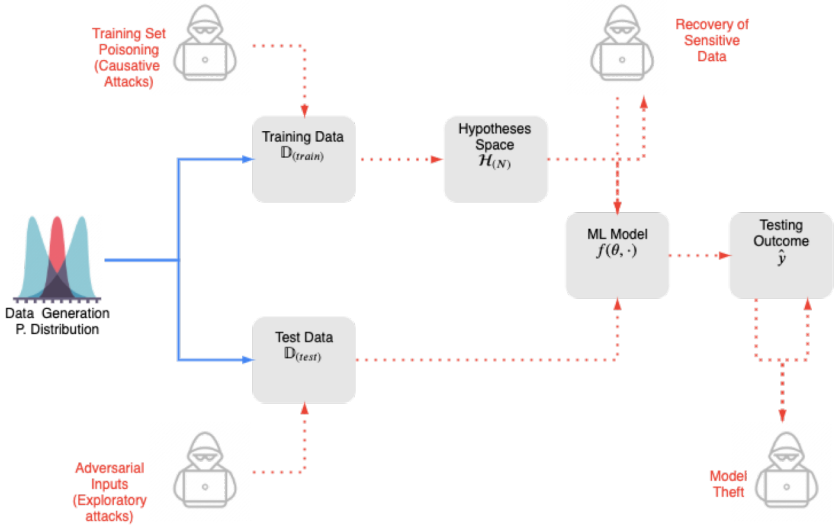
- A computer learns from experience with respect a task and a performance measure, if its performance on the task improves with more experience
- We typically start with a data set, which we divide into training and test data (experience)
- And try to fit a model which selects the best hypothesis for an objective (based on the performance measure)
- In this talk we will focus on the task of classification



Security of the ML pipeline

There are 2 main attack vectors for ML:

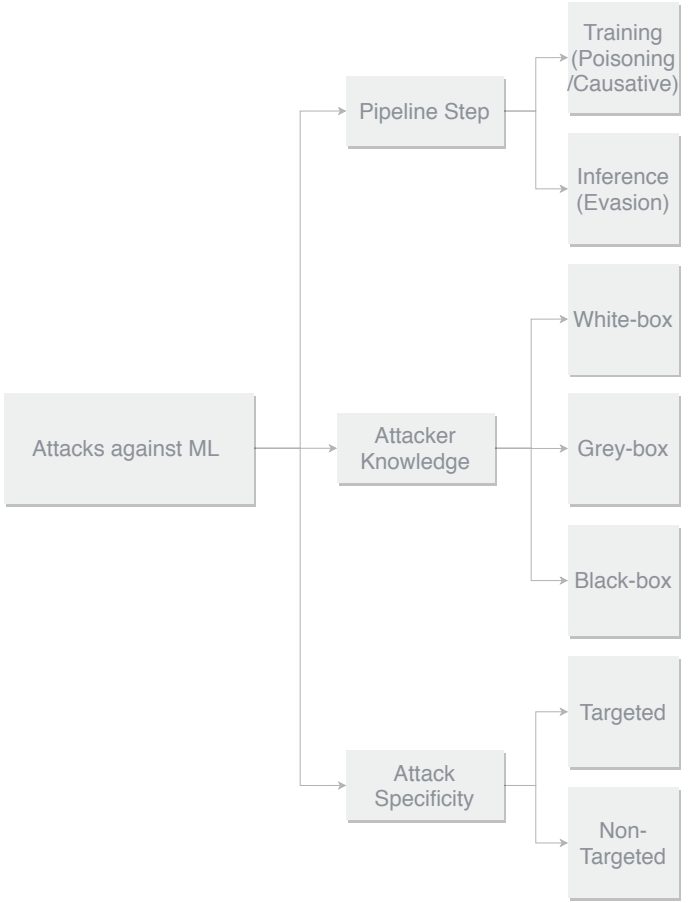
- Training data – attackers can poison the training data in order to introduce malicious behavior (e.g., accuracy drop, back doors)
- Test data – attackers can corrupt test samples to achieve different goals (e.g., misclassification, recovery of sensitive data, model theft)
- Attacks can target the Confidentiality, Integrity, Availability (CIA) or Privacy of ML models



A brief taxonomy of attacks

ML Attacks can be classified based on 3 dimensions:

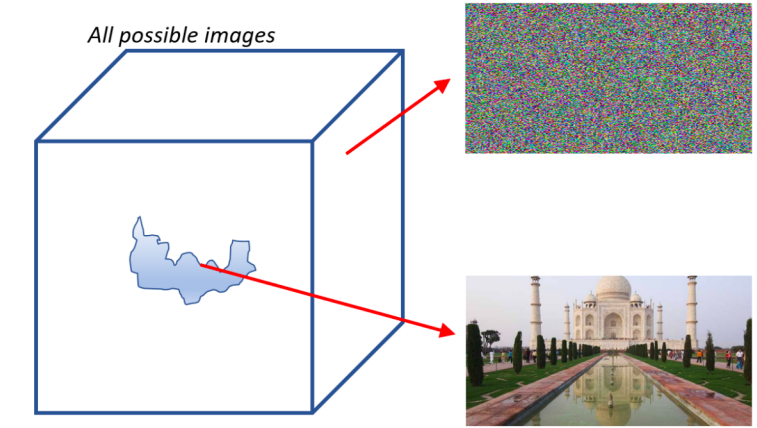
- The pipeline step (Training or Inference)
- The attacker knowledge (White, Grey or Black-box)
- The attack specificity (Targeted or Non-targeted)



ML assumptions relevant to security

ML algorithms use 2 fundamental assumptions:

- 1. The i.i.d assumption - Training and test data are drawn from the same distribution i.e., they are identically and independently distributed
- 2. The manifold assumption - The data lie on a low dimensional manifold embedded in a higher dimensional space
- Breaking any assumption is automatically a security vulnerability



Taj Mahal manifold is surrounded by an increasingly larger negative space as *n* increases.

Training data poisoning

Two types of data poisoning attacks:

- Targeting availability: inject corrupted data in the training set s.t. the hypothesis learned becomes useless
- Targeting integrity: introduces a back-door s.t. the performance does not change, but the presence of certain features can induce undesired behaviour

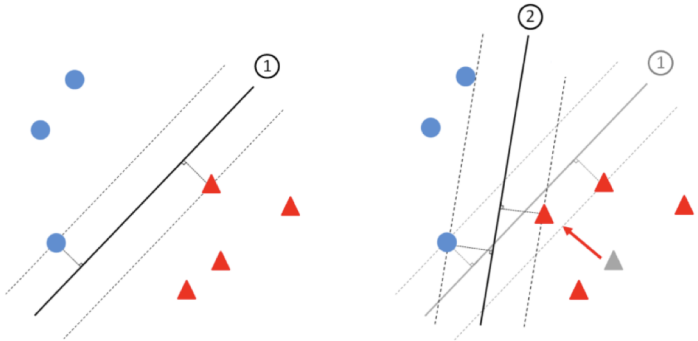


Image from "Adversarial Learning in Statistical Classification: A Comprehensive Review of Defenses Against Attacks"

Availability data poisoning attacks

- Only corrupt new samples are added to the training set
- Try to find the minimum nr. of examples that maximise the loss of a model
- Define an optimisation function, e.g., $\min_{\theta \in \Theta} l(\theta, D_{clean}) + \epsilon \max_{(x,y) \in Dp} l(\theta, x, y)$
- We can solve the optimisation problem (e.g., using gradient descent) or try to generate poisoned samples with generative models

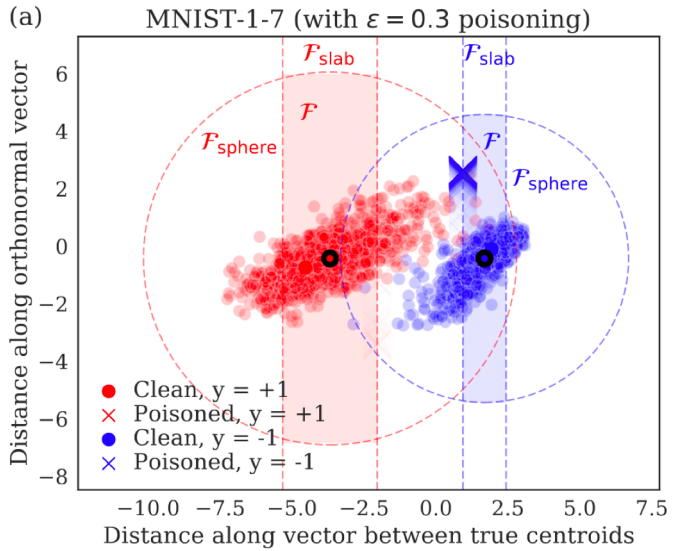


Image from "Certified Defenses for Data Poisoning Attacks" – 3% data poisoning leads to 11% drop in accuracy even when strong defences are used

Integrity data poisoning attacks

- Malicious behaviors are triggered only by some features / samples (called trojans)
- The trigger is defined as a mask to be applied on the input
- The mask targets some parameters of the model (white-box or black-box through model inversion)
- E.g., we can define an optimization problem to reduce the difference between the target and the value on certain inputs: $L = (tar_1 - f(x_1))^2 + (tar_2 - f(x_1))^2 + \cdots$, and change the mask

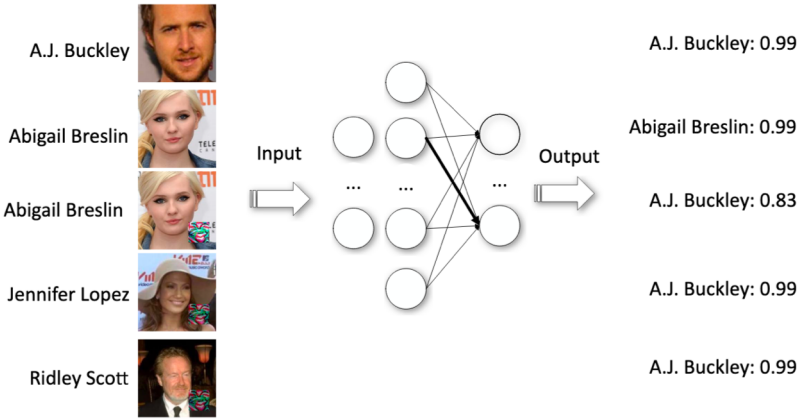
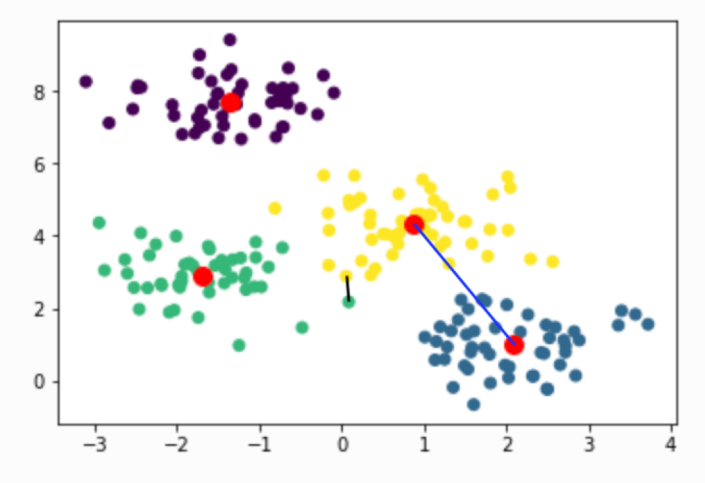


Image from "Trojaning attack on neural networks"

Defences against training data poisoning

Defences against data poisoning are based on data sanitisation:

- Remove data based on distribution properties (oracles)
- Allow only some data samples for training (e.g., licensed words in text)
- Remove or detect poisoned data based on other criteria (e.g., distance from centroids, anomaly detection)



Inference (Test) or Evasion attacks

Three types of data poisoning attacks:

- Targeting integrity: add perturbations to test samples in order to compromise accuracy
- Targeting confidentiality: add perturbations to test samples in order to extract model parameters
- Targeting privacy: use model outputs to determine if some data was used in training or recover the training data set

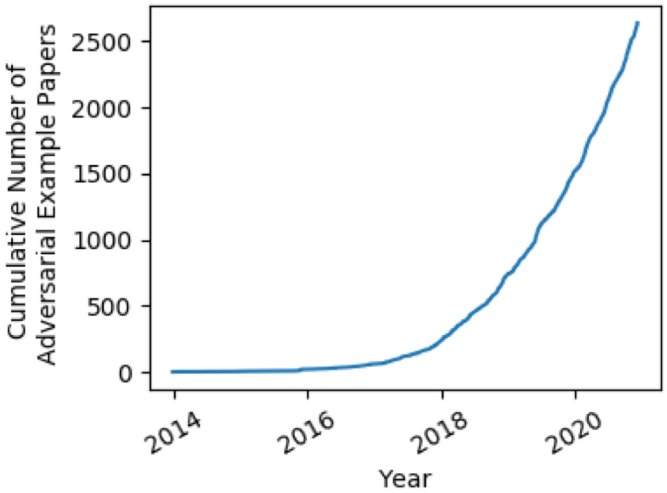


Image from
https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

Integrity inference attacks – adversarial examples

- Adversarial examples are samples that look similar but induce undesired behaviour
- Try to find the minimum perturbation that can induce targeted or non-targeted misclassification
- Define an optimisation function, e.g.,
  $$min_\eta \left|\left|x + \eta - x\right|\right|_p \text{ s.t. } f(x + \eta) = \hat{y}$$
- In reality taking small steps towards maximising the loss function suffices: $\eta = \epsilon(\nabla_x l(\theta, x, y))$

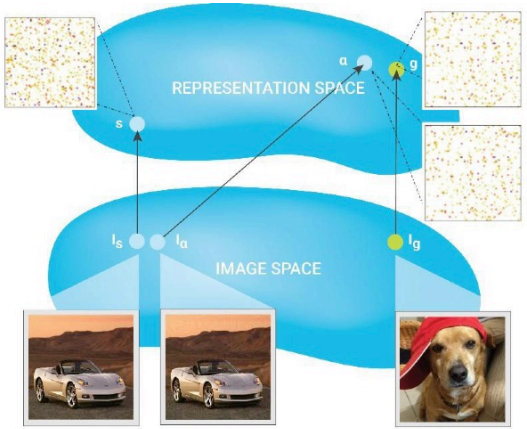
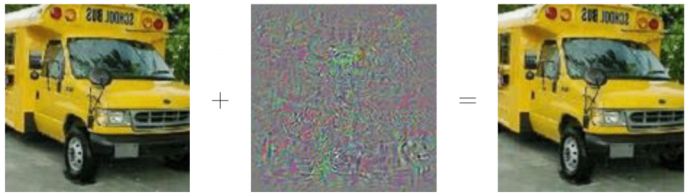


Image from "Adversarial manipulation of deep representations"



Alex Serban
cs.ru.nl/~aserban

Universal adversarial examples and transferability

- Universal perturbations (which can be applied to all inputs) can be created in a similar fashion
- Without knowledge of the model under attack (black-box), we can create adversarial examples on a proxy model
- Adversarial examples transfer even between different ML techniques

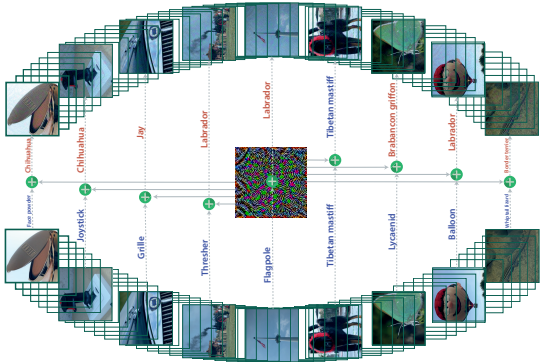


Image from “Universal adversarial perturbations”

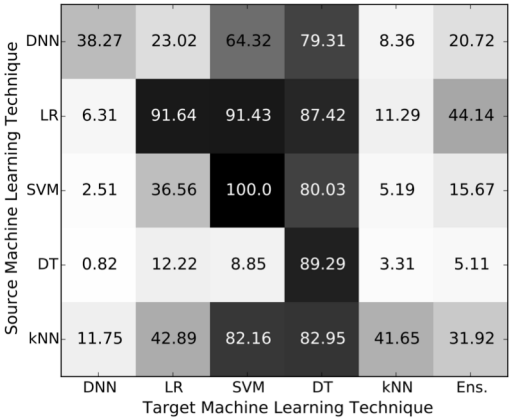
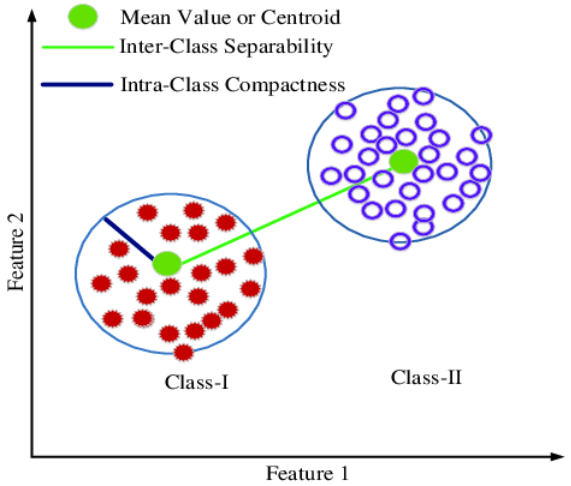


Image from “Transferability in Machine Learning From phenomena to Black-Box attacks”

Defences against adversarial examples

A plethora of adversarial defences have been proposed, without much success

- Models robust to adversarial examples must have large inter-class separability and small inter-class compactness
- Many defences have been proposed (e.g., to detect adv. examples, to transform inputs before the model)
- The most effective defence is adversarial training (including adv. examples in the training data set)
- However, adv. training increases training time significantly



Confidentiality inference attacks – model extraction

- Reverse engineer the models' parameters by observing the output
- Formulate a set of equations where the unknowns are the models' parameters
- Or find points arbitrarily close to the model's decision boundaries and extracts parameters from these samples

Service	Model Type	Data set	Queries	Time (s)
Amazon	Logistic Regression	Digits	650	70
	Logistic Regression	Adult	1,485	149
BigML	Decision Tree	German Credit	1,150	631
	Decision Tree	Steak Survey	4,013	2,088

Image from "Stealing machine learning models via predictions APIs"

Defences against model extraction

- Model extraction can be reduced with tricks or by serving multiple models, but not by algorithm design
- Round confidence scores to some fixed precision
- Use differential privacy
- Use ensembles of models

Service	Model Type	Data set	Queries	Time (s)
Amazon	Logistic Regression	Digits	650	70
	Logistic Regression	Adult	1,485	149
BigML	Decision Tree	German Credit	1,150	631
	Decision Tree	Steak Survey	4,013	2,088

Image from “Stealing machine learning models via predictions APIs”

Privacy inference attacks – model inversion

- Model inversion recovers the training data from the model
- Start with a random input vector
- Use gradient ascent in the *input space* to maximise the model's confidence on the target prediction

Image from "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures"

Defences against model inversion

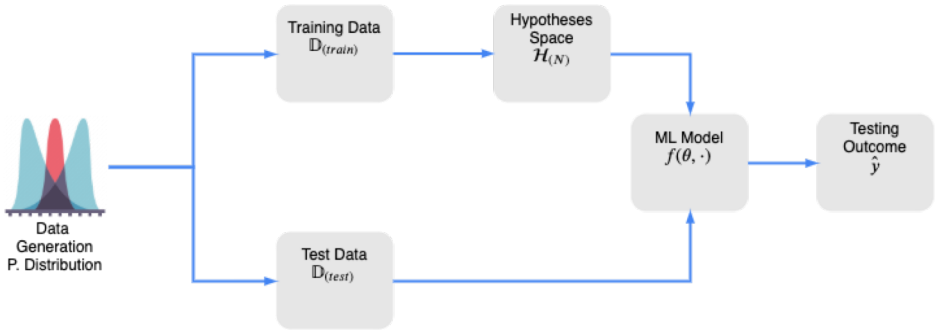
- Hide confidence scores from predictions (defence by obscurity – not recommended)
- Regularise models in order to avoid memorization and increase generalization (not efficient)
- Use differential privacy

Image from “Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures”

ML for Security vs. Security of ML

There is a difference between using ML for security and developing security for ML

- ML for security – e.g., spam detection, network intrusion detection – misclassifications, esp. false negatives, have an impact on security
- Security of ML – although preferable to avoid, misclassifications do not always have an impact on security (e.g., object recognition in cloud storage)



Beyond algorithmic robustness – robust ML applications

- Until now we considered the data is controlled (as it is in most research projects, but rarely in the world)
- Data in the world introduces new risks (e.g., storage, legal, representativeness, entanglement)
- Data assembly and transformation also introduces risks (e.g., annotations, fusion, normalization)
- Control over these processes (e.g., by developing distribution checks or control the labeling process) increases robustness

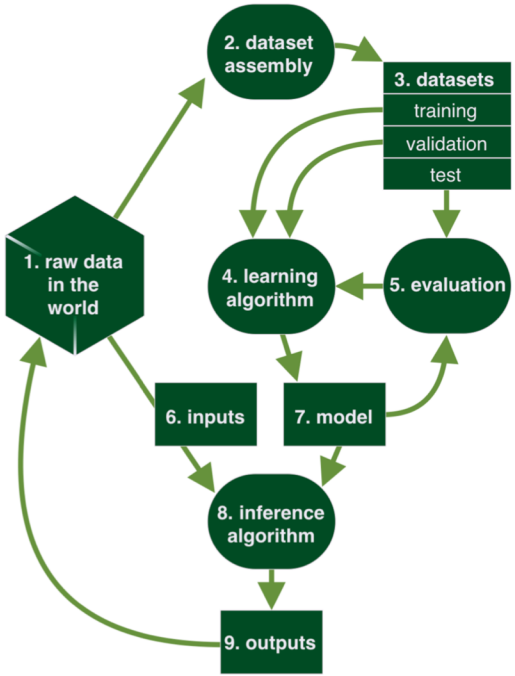


Image from "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures"

Robust ML applications – Model risks

- Until now we considered accuracy the only model evaluation metric
- In real scenarios accuracy is just one of the metrics needed
- A robust application should also be tested for bias (e.g., group and subgroup bias)
- A robust application must ensure fairness, interpretability and provide explanations to users (new attack vectors?)
- Due to diverse datasets and experiments, more robustness risks are associated to models: e.g., reproducibility, hyperparameter optimization, randomness

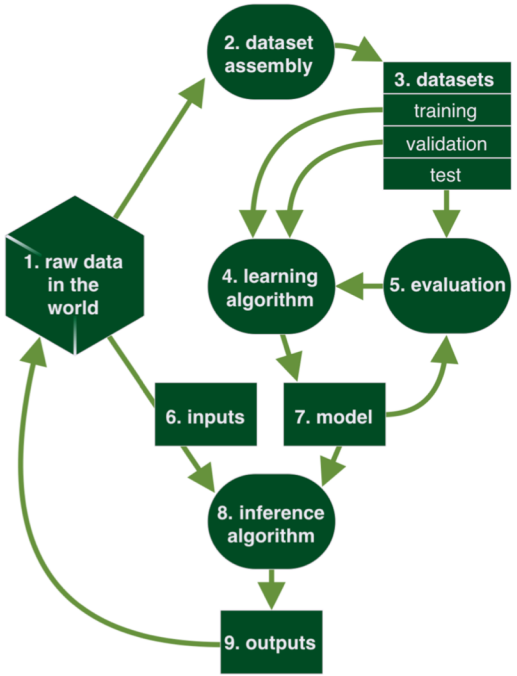


Image from "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures"

Deployment and inference risks

- An application must be deployed and maintained -> more risks
- The deployment environment raises hardware and software related security risks
- The (evolving) nature of the world requires continuous adaptation and re-training
- ML related incidents have to be managed fast, although the ML life-cycle (e.g., re-training) takes longer than traditional software

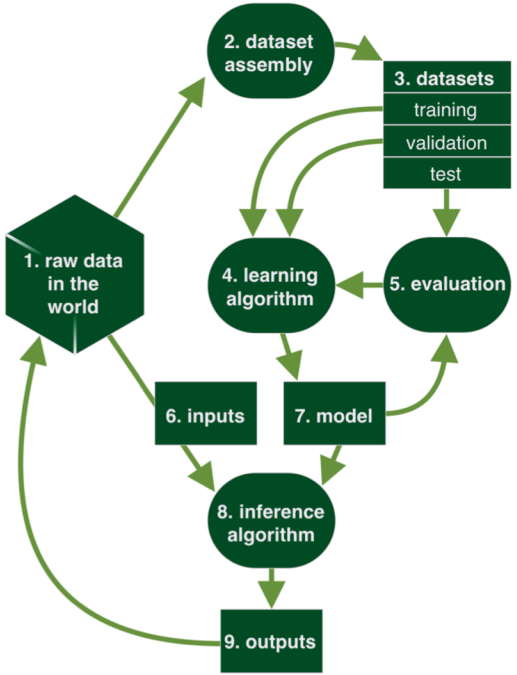


Image from "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures"

Governance risks

- ML applications interact with humans and often process personal data -> they have to respect human rights and avoid ethical risks
- Users have the right to an explanation -> robust ML applications should provide them
- Robust ML applications should be auditable by external actors (i.e., audit trails should be built in an application)
- Users have the right to be informed that they are interacting with apps using ML (i.e., robust apps must be transparent and establish communication channels with users)

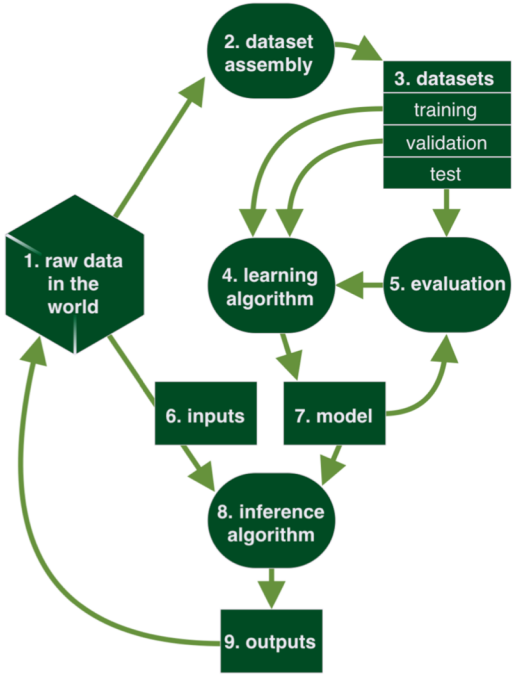


Image from "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures"

Learn more

- Follow and contribute to the Software Engineering for Machine Learning project at LIACS (https://se-ml.github.io)
- Read the catalogue of practices for building robust and future proof machine learning apps (https://se-ml.github.io/practices)
- Follow the Awesome Software Engineering for Machine Learning reading list on Github (https://github.com/SE-ML/awesome-seml)

Reading list

Check out the awesome list with relevant literature: https://github.com/SE-ML/awesome-seml

Catalogue

Check out the catalogue of ML Engineering Practices: https://se-ml.github.io/practices

Resources

Besides the references cited in the figure captions, we recommend:

- Underspecification presents challenges for credibility in modern machine learning
- Explaining and harnessing adversarial examples
- Adversarial Examples that Fool both Computer Vision and Time-Limited Humans
- Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples
- Adversarial Examples on Object Recognition: A comprehensive survey
- https://nicholas.carlini.com/writing/2018/adversarial-machine-learning-reading-list.html
- https://se-ml.github.io
- https://se-ml.github.io/practices/

THANKS

Questions?

cs.ru.nl/~aserban

https://se-ml.github.io