184.702 Security, Privacy and Explainability in Machine Learning – Exercise 3

The third exercise allows you to choose from different topics from one of the lectures on

1. Privacy-preserving data publishing (Lecture 1 & 2)
2. Privacy-preserving computation (Lecture 3)
3. Adversarial Machine Learning (Lecture 6 & 7)

There are some topic suggestions in this document, but you are also free to suggest other topics that fit with the lecture subjects. If so, please don't hesitate to contact me (mayer@ifs.tuwien.ac.at) and discuss it with us. If we see it fitting to the topics of the course, you'll get our ok to work on your idea.

The descriptions of the different topics are likely not a complete specifications, they just summarise the general idea and goals. If there is an unclarity, please do not hesitate to ask in the forum.

The exercises shall be done in groups of 2 students. Please arrange with your colleagues, and then register for a group **and** for your topic in TUWEL.

There is an online submission of the report and other artefacts that you created during this exercise (code, data, ...).

As some of the more specialised tasks are also less explored and thus less predictable in outcome, we will not grade you based on results alone.

If you already did / are doing one of these topics also as part of "184.702 Machine Learning – Exercise 3.1", then you need ensure that you either choose a different topic, or build on top of the results from there, but with a novel contribution in this course.

General comments for the exercise

This exercise is supposed to be solved with all the steps being programmatically, i.e. there shouldn't be a need for GUI or some other manual processing of data (rather scripted).

Your submission should contain

- A zip file with all needed files (your source code, your code compiled, data sets used (but **NOT** the ones **we provide** to you), a build script that resolved dependencies, or include any libraries you are using. Your submission needs to be self-contained!
    - If applicable, provide a means to compile your classes, preferably with a build file (e.g. Maven or ANT if you use Java)
    - If your build file allows for automatically obtaining the dependencies, then they don't need to be included, otherwise please include them.
    - Provide a short how-to explaining the way to start your program (which is the main class/file, which command-line options does it expect, ..).

    - Please also state clearly what is the version of the software package(s) you use; unless for a specific reason, please work with the latest versions (you can of course e.g. work in python 2, but don't work with old versions of libraries etc.)
- A report, describing
    - Your task
    - Your solutions (also describe failed approaches!)
    - Your results, evaluated on different datasets, parameters, ...
    - An analysis of the results
- Where applicably, your program shall be configurable via command-line options or a configuration file, to modify parameters, evaluation types, etc... i.e. it should not be needed to modify the code to change these options. There are many framework for command-line-options available, you don't need to code this yourself (e.g. for Java, Apache Commons CLI (http://commons.apache.org/cli/, http://martiansoftware.com/jsap/).

You need to choose ONE topic, not all of them :-)

Some of the datasets mentioned in the various task descriptions:

- AT&T (Olivetti) Faces: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, https://scikit-learn.org/0.19/datasets/olivetti_faces.html
- PubFig http://www.cs.columbia.edu/CAVE/databases/pubfig
- MNIST: http://yann.lecun.com/exdb/mnist/, https://scikit-learn.org/0.19/datasets/mldata.html
- CIFAR (10, 100): https://www.cs.toronto.edu/~kriz/cifar.html, https://github.com/ivanov/scikits.data/blob/master/datasets/cifar10.py

Computing resources

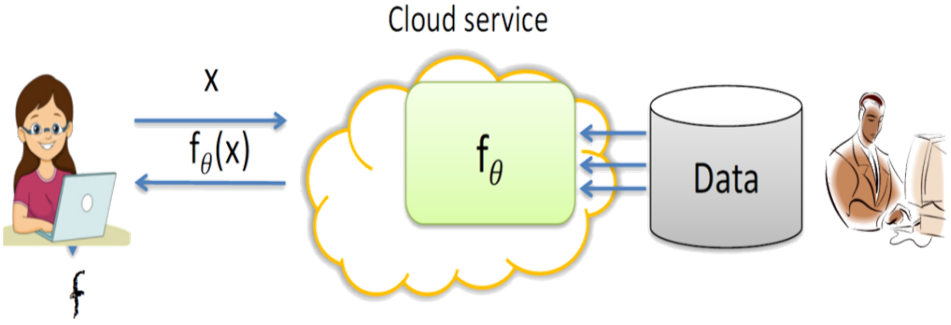
If you are in the need of computing resources, you can contact me (mayer@ifs.tuwien.ac.at) especially for CPU power on a Linux server @TU.

Google Cloud Platform

Further, you can try infrastructure-as-a-service offers, as some Cloud operators have some free starter credit. E.g the Google Cloud Computing platform gives you 300 USD if you register (you need to provide your credit card, but they don't charge) which you can use to create remote machines with various OS preinstalled. Also other providers have similar offers. Make sure to shut down your machines when not using them, to avoid being charged.

Topic 3.1: Adversarial Machine Learning

Topic 3.1.1: Model Stealing / Extraction



Model "stealing" means to obtain a trained model from a source that generally only provides a "prediction API", i.e. a means to obtain a prediction from the model when providing a specific input, but does NOT disclose the actual model (i.e. "ML-as-a-service"). This might be because the model may be deemed confidential due to their sensitive training data, commercial value, or use in security applications. A user might train models on potentially sensitive data, and then charge others for access on a pay-per-query basis.

In model stealing/model extraction, an adversary with black-box access (but no knowledge of the model parameters or training data), aims to duplicate the functionality of the model.

In this task, you shall try to experiment with such attacks, using the code provided at https://github.com/ftramer/Steal-ML, with similar attacks as in the corresponding paper (Tramèr et al. Stealing Machine Learning Models via Prediction APIs. USENIX Security 2016. https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_tramer.pdf)

You can also employ attacks & defences from the IBM Adversarial Toolbox, at https://github.com/IBM/adversarial-robustness-toolbox

Specifically, you shall experiment with these model stealing attacks for two different types of classifiers (can be from the paper) on three different datasets (two shall be different than in the paper, one to be used as validation that you achieve similar results). You can either use a model available locally (as black-box, i.e. just for querying & confidence values), or one of the mentioned online services. While the model is "extracted", record how the accuracy of the duplicated model changes, and how many queries (and how much time) it takes to train the model; plot these, in similar fashion as in the paper, for each dataset/classifier.

Topic 3.1.2: Watermarking ML/DL models

Sharing trained models has brought many benefits to development of ML and DL systems. However, training models is usually a task that requires vast resources, from data to time and computing power. Watermarking can help the owners of such models mark their property in order to trace unauthorised usage or redistribution.

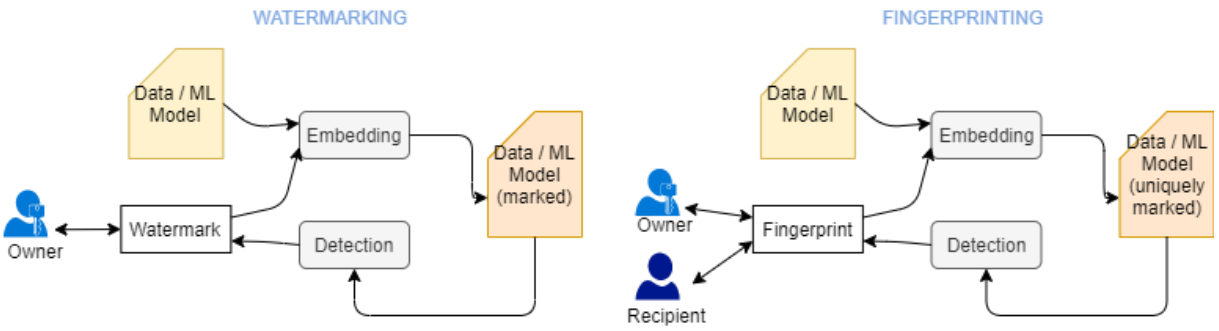
The task of this exercise is to familiarize yourself with one of the watermarking techniques mentioned at the lecture:

- Uchida et al.: Embedding Watermarks into Deep Neural Networks (https://dl.acm.org/doi/pdf/10.1145/3078971.3078974)
- Nagai et al.: Digital Watermarking for Deep Neural Networks (https://arxiv.org/pdf/1802.02601.pdf)
- Rouhani et al.: DeepSigns: A Generic Watermarking Framework for Protecting the Ownership of Deep Learning Models (https://arxiv.org/pdf/1804.00750.pdf)
- Merrer et al.: Adversarial Frontier Stitching for Remote Neural Network Watermarking (https://arxiv.org/pdf/1711.01894.pdf)
- Adi et al.: Turning Your Weakness into Strength; Watermarking Deep Neural Networks by Backdooring (https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-adi.pdf)
- Zhang et al.: Protecting IP of Deep Neural Networks with Watermark (https://link.springer.com/content/pdf/10.1007%2F978-3-030-47436-2_35.pdf)
- ...

Either: implement the chosen technique or find and utilize an open source implementation.

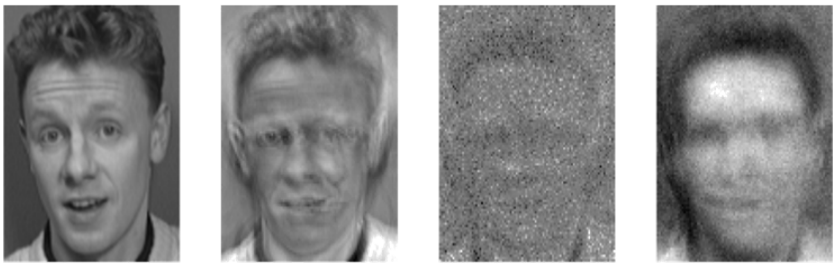
In both cases: demonstrate the success of the watermarking technique. You can use the experiments from the paper of the chosen technique as a guidance.
Think of the attacks that apply to the chosen watermarking technique, e.g. fine-tuning, pruning, transfer learning, etc., perform the attack(s) and analyse the robustness of the watermarking technique with respect to the attacks.

In case where an implementation of the chosen technique is available as an open source, such as DeepSigns (https://github.com/Bitadr/DeepSigns), in addition to the tasks above, you shall try to create multiple different watermarks, i.e. fingerprints. Fingerprinting is a technique for protecting ownership of the intellectual property, similar to watermarking. Fingerprint is a watermark which, besides the information about the owner, contains the information about the recipient of the particular copy of the data (of the ML/DL model, in this case). This means that every recipient of the fingerprinted data copy receives a different version, and by fingerprint detection, it is possible both to verify the owner and trace the recipient. Analyse the success of your embedding and detection algorithms while using fingerprints.



In TUWEL specify the datasets used in the experiments. In the report describe your methodology in detail, explain your findings and provide the source code of your implementation.

Topic 3.1.2: Model Inversion Attack



Given access to a model, an ***inversion*** attack tries to extract the training data used for obtaining that model. A prominent example is for a face recognition system, where the model tries to identify to which of the known users a newly taken image (e.g. from an access control camera) corresponds to. This means that each class in the dataset corresponds to one specific individual, and the training data for each class is generally a number of images of that individual (taken with potentially different angles, zoom, light, background, and appearance such as hair style, ...).

For the attack, in most cases, this means picking a specific class, and then trying to ***generate*** input patterns that return the highest confidence of being classified as that class. This normally means access to the model (also in white-box, i.e. being able to read the model parameters; potentially building on top of a model stealing/extraction attack, but you can assume to have white-box access).

In this task, you shall re-create experiments in the style of the paper by Fredrikson et al: "Model inversion attacks that exploit confidence information and basic countermeasures" (http://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf) (or a more recent paper if you find something interesting; a recent paper on this topic is e.g. https://arxiv.org/abs/1911.07135). For the original paper, we do have the code available, so please ask and we will share it.

A first step is to recreate the original experiments. Then, you can chose two variants

1. Extending the experiments to different datasets, e.g. the
    - Yale Face Database http://vision.ucsd.edu/content/yale-face-database
    - Labeled Faces in the Wild, http://vis-www.cs.umass.edu/lfw/, using the task for face recognition (if required, also a subset from this). See also https://scikit-learn.org/0.19/datasets/#the-labeled-faces-in-the-wild-face-recognition-dataset
    - A subset from the PubFig dataset, http://www.cs.columbia.edu/CAVE/databases/pubfig
    - Any other data set for face recognition (except the original AT&T / Olivetti Faces)
2. Extending the experiments to multiple different target models on the original dataset, such as different types of MLPs (multiple layers, varying the number of nodes, activation functions, on-purpose overfit / underfit models, ...)

In your evaluation, compare how well the inversion works for each of the individuals, and also evaluate what is the "cost" of generating the images – i.e. how many iterations of generating and asking the model are required, and how is the efficiency of this (in runtime). You can also evaluate

the quality of the generated images by computing a numerical similarity to the target image (e.g. the most representative of the training images per class).

Topic 3.1.3: Backdoor/poisoning Attacks & defence

Poisoning/backdoor attacks modify the training data to **embed a specific pattern** in some images (e.g. a yellow block on a STOP sign), and falsely label that image as a different class (e.g. as a speed limit 50 sign) to make the classifier learn this pattern for wrong predictions. These attacks generally work because a certain number of neurons in the network can be dedicated to learn these patterns, often because the number of neurons is larger than what is required to actually represent the pattern in the "clean" training data.

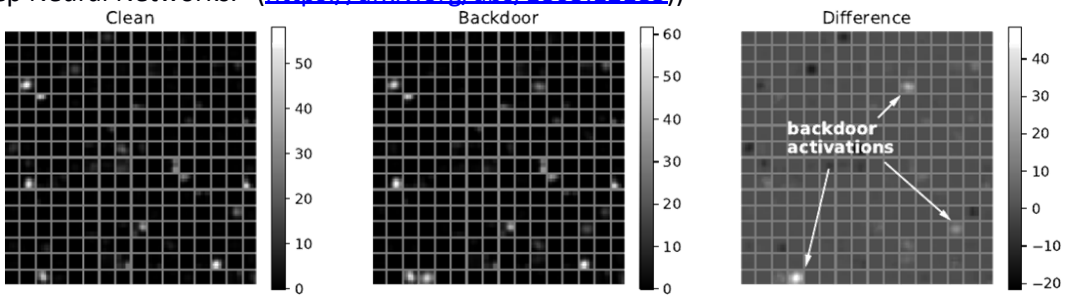
We will provide you with two datasets with manually prepared backdoor images, on the German Traffic Sign Recognition Dataset, and Yale Faces.

You shall then first attack the model with this poisoned data, and evaluate the performance of the attack, i.e. the clean and poisoned samples, comparing the clean samples to a baseline of an unmodified network. The evaluation shall be in a similar manner as to "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. Tianyu Gu, Brendan Dolan-Gavitt, Siddharth Garg. https://arxiv.org/abs/1708.06733)

You shall then either:

- Compare three of the defences as implemented in the IBM Adversarial robustness toolbox (https://github.com/IBM/adversarial-robustness-toolbox). The toolbox currently provides (https://github.com/IBM/adversarial-robustness-toolbox/tree/master/art/poison_detection) :
    - Detection based on activations analysis (Chen et al., 2018)
    - Detection based on data provenance (Baracaldo et al., 2018, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8473440)

    - RONI (https://people.eecs.berkeley.edu/~adj/publications/paper-files/SecML-MLJ2010.pdf)
    - Further, it provides the ClusteringAnalyzer, which you might utilise for detection.
- Or implement another defence mechanism not yet provided in the toolbox. Your implementation should fit within the IBM Toolbox, i.e. use the existing base class etc, which will also help you in evaluating your approach. You can implement approaches such as
    - Fine-pruning (Liu et al: "Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks." (https://arxiv.org/abs/1805.12185))



    - Outlier detection based, as in Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pages 3520–3532, USA, 2017. Curran Associates Inc.
    - Or another anomaly-detection based approach by Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. 2017 IEEE International Conference on Computer Design (ICCD), pages 45–48, 2017.

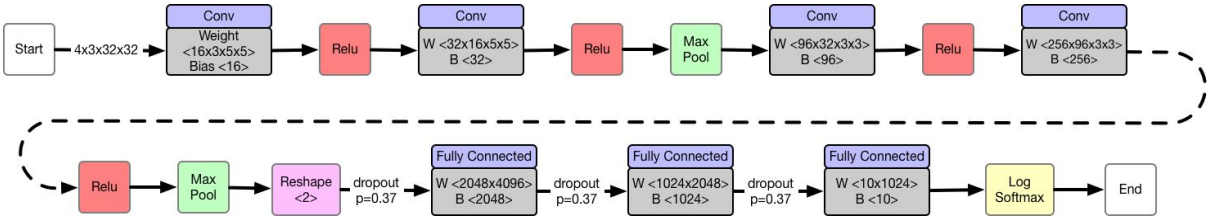
Topic 3.1.4: Backdoor/poisoning Attacks: CNN vs Feature Extraction

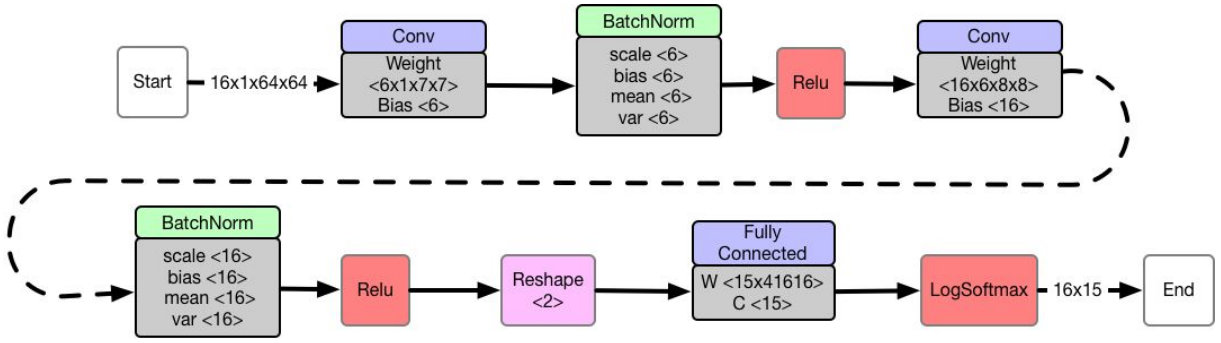
It has been postulated that Deep Learning (specifically Convolutional Neural Networks) are much more prone to poisoning/backdoor attacks (as well as evasion attacks in the form of adversarial inputs), as compared to "traditional" approaches for image classification that relied on a feature extraction step (e.g. SIFT, HOG, ...) and a subsequent model such as SVMs or RFs.

However, there is rather little evidence that this is true. In this task, you shall thus compare CNNs and traditional approaches, on the same datasets that are available also for Topic 3.1.3 (traffic signs and face recognition with pre-created backdoor images).

Try a setup where you first extract image classification features (similar to the ones in Task 3.8, i.e. SIFT or something equally powerful (SURF, HOG, ..)) and then learn a traditional model, such as an SVM or a Random Forest, on the data that contains the poisoned samples. Evaluate the (impact on the) accuracy on clean image vs. the percentage of the model being tricked into predicting the desired target class.

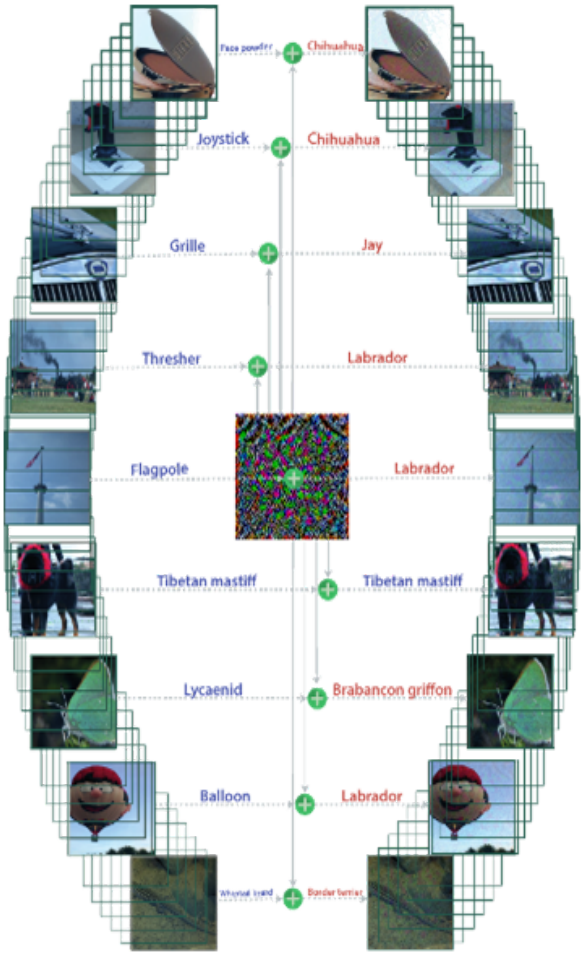
Finally, train a (benchmark) CNN model that shall contain the backdoor (i.e. train it on the dataset with the poisoned samples included), Then, compare the accuracy and success rate for triggering the backdoor on the CNN and the traditional approach. For the two mentioned datasets above, reference architectures as depicted below can be provided and re-used.





Topic 3.1.5: Model Evasion Attack

Evasion attacks are less targeted than backdoors, as they "just" try to avoid a specific class to be predicted.



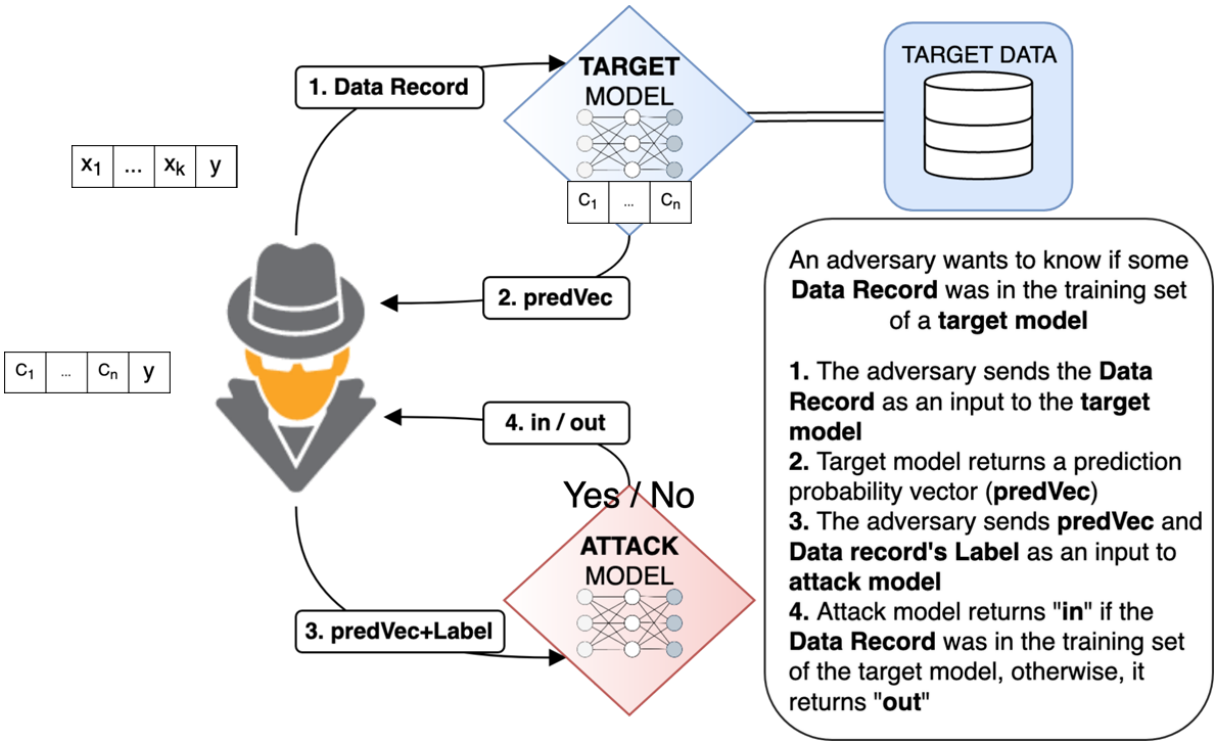
In this task, you shall evaluate, e.g. on the AT&T Faces dataset (https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, https://scikit-learn.org/0.19/datasets/olivetti_faces.html) or the Yale faces, or a subset of the PubFig dataset, how well different methods for generating adversarial images work, by measuring the distortion needed in a metric similarity, complemented with an occasional manual inspection of the perceived difference. Also measure efficiency of the attacks. Specifically of interest are also methods that generate universal perturbations (i.e. not specific to one instance/image).

Then, try two simple methods for defending against these attacks (e.g. by adversarial training). You can utilise e.g. the IBM Adversarial Robustness Toolbox

(https://github.com/IBM/adversarial-robustness-toolbox), or also Google's Cleverhans (https://github.com/tensorflow/cleverhans) or Foobox (https://github.com/bethgelab/foolbox)

Topic 3.1.6: Membership Inference

Membership inference tries to infer information from a learned ML model on the data samples that were used for training the model. In terms of released information, it is one of the weakest forms of attack on the confidentiality of the data samples, with the goal of revealing whether a sample was used for training a model, or not. This can be disclosive on an individual, e.g. if one can then infer that the individual might suffer from a certain disease.



There are two variants of this task

a.) Detailed evaluation:

This task should investigate the success of this attack, respectively what are the conditions that make this attack successful. To this end, try the attack on several datasets with different characteristics, e.g. with different number of classes, and with differently trained models, e.g. those that on purpose overfit, and some that rather underfit, as well as those that should be trained with the "optimal" fitting amount of training. You may target neural networks, or also other types of models.

b.) Unsupervised approaches to membership inference

The above mentioned shadow models are used to have training data for the attack model - this needs to have a ground truth on whether a sample was in the training data (of the shadow model that shall simulate the target (real) model), or not.

Another approach could be to perform an unsupervised attack in a grey-box setting: given a number of responses (including the full prediction vector / activations from the last layer) from the target

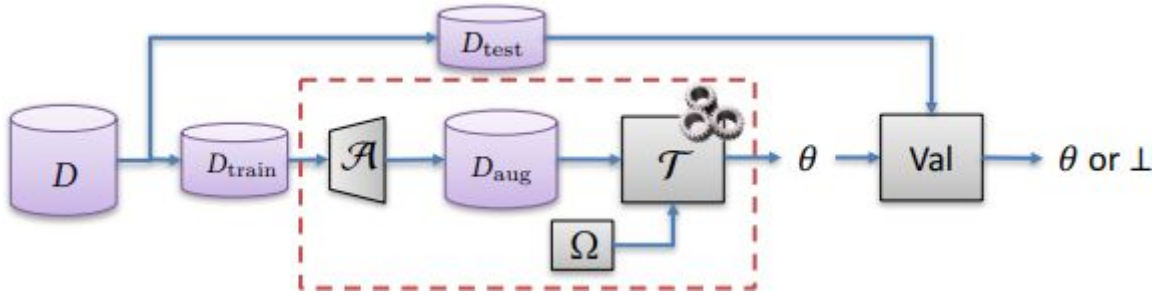
model, try to cluster these into two clusters, hoping that this corresponds to “in” and “out” clusters. Try multiple state-of-the-art clustering algorithms (don’t implement one on your own), and evaluate your clustering with actual ground-truth data (i.e. you know which data has been used to

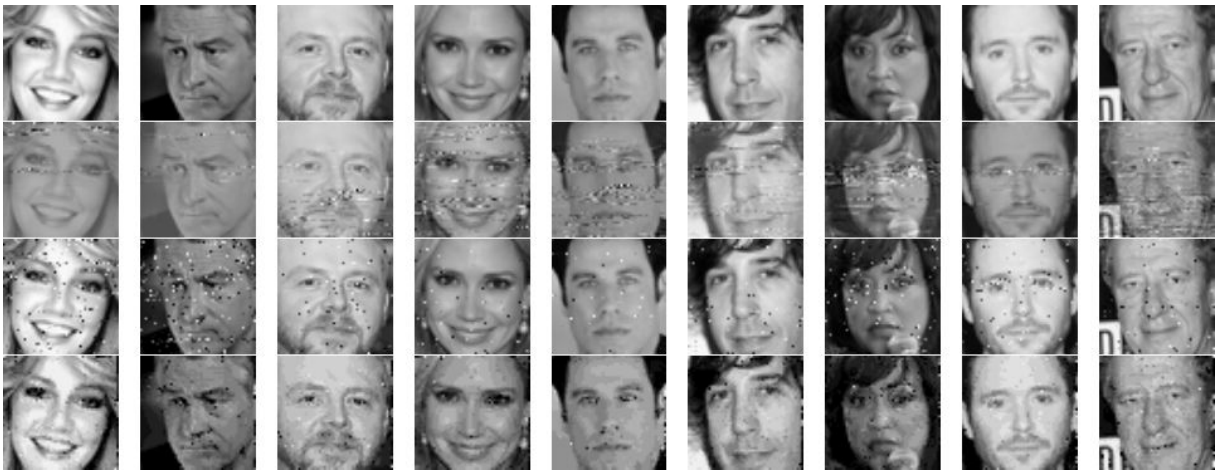
Main paper: Membership Inference Attacks Against Machine Learning Models, https://www.cs.cornell.edu/~shmat/shmat_oak17.pdf

Other useful paper:

- Towards Demystifying Membership Inference Attacks, https://arxiv.org/abs/1807.09173
- ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. https://arxiv.org/abs/1806.01246

Topic 3.1.7: Data exfiltration





Data exfiltration tries to utilise a machine learning model to leak information about the training data to the end-user of the model. This can be interesting for an attacker when the model gets trained on private training data, and there is no other means to leak information about the model to the outside (the attacker).

In this task, you shall recreate experiments either in a white-box or black-box settings as described in the paper by Song et al: Machine Learning Models that Remember Too Much. CCS 2017

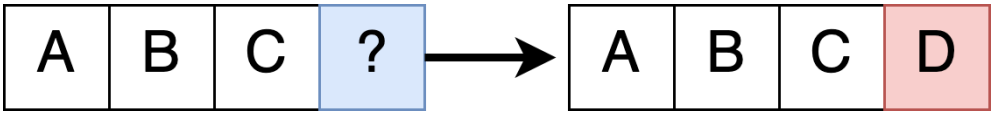
You can perform your experiments on the same dataset as the original authors, or chose one of the above mentioned datasets (AT&T faces, Yale faces, ...)

Topic 3.1.8: Attribute Inference (disclosure) from ML Models (experimental)

(note: this topic is rather experimental, and you will, as for all the other topics, graded on your effort, not necessarily on a (positive) outcome, which might not be achievable).

One type of information disclosure is called attribute disclosure: it consists of inferring the value of an attribute (e.g., ethnicity) that was hidden (i.e., not directly shared by the user).

Attribute disclosure is attribution independent of identifying a specific record in a database. It was shown that it is possible to infer additional information about a user from data shared by other users. This can be sensitive information about a user such as ethnicity or political affiliation, inferred by mining available data.



While attribute disclosure until now mostly considered the setting where a released data set is the source, similar attacks might be possible from a released machine learning model, which invariably encodes some information about the training data.

Your task in this exercise is to test if this is possible for certain machine learning models. Given an incomplete sample X and a trained model f(X), is it possible to infer the unknown values of X? You can first assume that X (with its full information) was also used in actually training the model, but further testing whether it would be also possible for an X not used in the training.

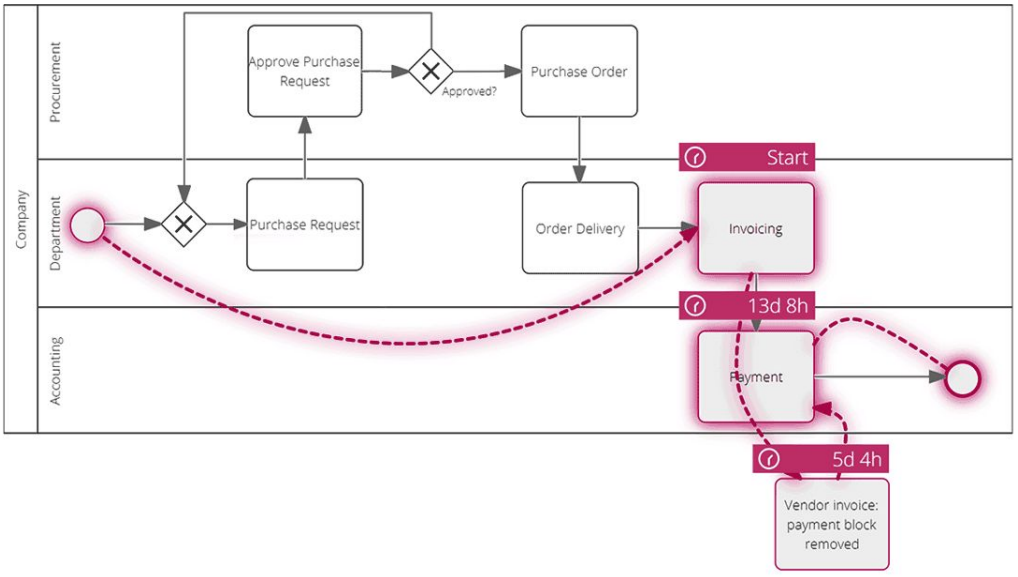
Using models that generally overfit to the training data seems more promising. As such, a neural network, or also a decision tree, might be good candidates.

Topic 3.2: Secure / Distributed Computation

Distributed computation in general addresses privacy / data protection issues by not centralising data that is initially already partitioned between several parties, and either computing functions securely w/o revealing the input (secure multi-party computation), or by exchanging only aggregated values, such as gradients or other model parameters (federated learning), or by computing on encrypted data (homomorphic encryption).

In this task, you shall evaluate the effectiveness and efficiency of these approaches. An evaluation on the quality of the models, as compared to having a model trained on all the different partitions of the data, shall be performed. - i.e. observe the effectiveness and efficiency on the centralised vs. the distributed approach.

Topic 3.2.1: Federated (Distributed) Learning for Process Mining



Process mining tries to learn the structure of a process from logs or other sources about previous executions of process. This is typically done on centralized data, i.e. with centralized learning. Collaborative learning of such a process model from different organisations can be an interesting setting in many cases. However, instead of centralising the data, which might not be an option due to confidentiality reasons, federated learning can become a viable approach.

In this task, you shall perform experiments for distributed process mining. To this end, take an existing benchmark dataset from e.g. https://data.4tu.nl/repository/collection:event_logs_real, and first perform process mining in a centralised way with an existing approach (e.g. using ProM, http://www.promtools.org/). Then, try to federate / distribute the process, by first distributing the

data, and then the learning. Data distribution could be done in uniform (each subset is roughly similar) and non-uniform ways (some aspects of the data are present only at one node).

One approach for distributed learning you could follow is: “On the Feasibility of Distributed Process Mining in Healthcare”, 2019. https://link.springer.com/chapter/10.1007/978-3-030-22750-0_36

Topic 3.2.2: Federated learning for Image Data

Federated learning tries to learn one central model from data that is initially distributed, with the goal of achieving at least comparable predictive effectiveness (e.g. accuracy). In this task, you shall experiment with federated learning.

Utilising e.g. PySyft (https://github.com/OpenMined/PySyft), based on PyTorch, or TensorFlow federated (https://www.tensorflow.org/federated/), or simply your own setup, evaluate the effectiveness (and efficiency) of federated learning vs. the centralised learning on a small image database (e.g. AT&T faces, PubFig, CIFAR; if needed a subset of each) while learning neural networks. Specifically compare the difference in effectiveness when you vary some parameters, e.g. the number of nodes (from maybe 2-3, up to a higher number), or how often federation steps (sometimes called cycles) are performed.

Topic 3.2.3: Federated Learning on text data (or other sequential data)

In this task, you shall work with federated approaches for text data utilising RNNs (LSTM, GRU, ...). Take a task of your choice (e.g. a text categorisation / classification task like the Reuters Dataset, or 20 newsgroups, etc..), and first establish a baseline with a state-of-the-art recurrent neural network, comparing your results to literature.

Then, simulate a federated setting by distributing the data on multiple sources, and compare your results to a centralised setting, similar to the setting in the topic above.

Topic 3.2.4: Federated Learning on relational data & shallow learning

In this task for federated learning, you shall work with “traditional” relational data (i.e. not sequential, not image, ...). You shall pick two different datasets, which can be any dataset that you have e.g. used in the Machine Learning lecture in Exercise 1 or 2, and employ federated learning with two different types of algorithms on it. You are free to reuse existing implementations (like the ones mentioned above), or you can implement your own approach, but you shall focus on algorithms other than Neural Networks- you can consider e.g. SVMs, Naive Bayes, Decision Trees, ...

Simulate a federated setting by distributing the data on multiple sources, and compare your results to a centralised setting, similar to the setting in the topic above.

Topic 3.2.5: Secure Multi-party computation for Image Classification

Utilising e.g. the library mpyc for Python (https://github.com/lschoe/mpyc) and the implementation of a secure computation for a binarised multi-layer perceptron, first try to recreate the results reported in Abspoel et al, "Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks", i.e. train a baseline CNN to estimate a potential upper limit of achievable results, and then train the binarized network, as a simplified but still rather performant version, in a secure way. If needed, you can use a subset of the MNIST dataset.

Then, try to perform a similar evaluation on another small dataset, either already available in greyscale, or converted to greyscale, e.g. using (a subset of) the AT&T faces dataset.

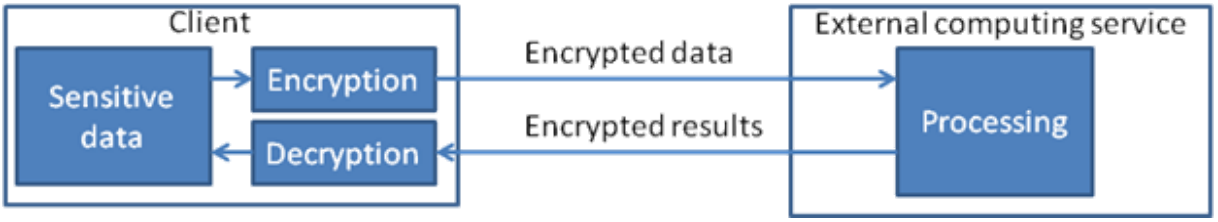
Specifically, evaluate the final result in terms of effectiveness, but also consider efficiency aspects, i.e. primarily runtime, but also other resource consumption.

Topic 3.2.6: Secure Multi-party computation for relational Data

In a similar setting as above (but with a machine learning algorithm of your choice), you shall perform secure multi-party computation on relational data. You shall pick two different datasets, which can be any dataset that you have e.g. used in the Machine Learning lecture in Exercise 1 or 2, and simulate a distribution of the data to multiple parties. Evaluate your approach as compared to a centralised setting, similar to as described in the topic above.

Topic 3.2.7.: Machine learning on encrypted data (Homomorphic Encryption)

One approach to conceal sensitive information in data is to only provide encrypted data to the machine learning algorithm, and directly compute on encrypted data. The returned model is also encrypted, but can be decrypted with the same key as the data. Thus, only the owner of the data can utilise the model, while the computation can be outsourced to an untrusted third party.



To achieve computation on encrypted data, homomorphic encryption schemes are utilised. These will achieve operations on the encrypted data without any loss, but are normally rather slow.

You can utilise one of the existing frameworks providing HE, such as the Intel HE-Transformer, or the Microsoft SEAL Library, to implement a simple HE-based machine learning model (logistic regression, Perceptron, k-NN, Naive Bayes, or similar). This means adapting existing implementations, depending on the case exchanging computations with approximations.

You shall then compare an implementation on unencrypted data with the encrypted version. The dataset can be a small toy dataset, e.g. the IRIS dataset or even smaller.

Tool support is available, e.g.

- Intel's https://github.com/NervanaSystems/he-transformer
- Microsoft's https://github.com/Microsoft/SEAL
- A list at https://github.com/jonaschn/awesome-he
- Some others, like https://github.com/shaih/HElib or https://github.com/tfhe/tfhe

If you chose this topic, register in TUWEL which algorithm you would like to implement.

Topic 3: Privacy-Preserving Data Publishing

Topic 3.3.1: Generation and evaluation of synthetic image datasets

This is a rather experimental topic, but maybe therefore even more interesting.

The idea is similar to the one above, but with the difference that you are not going to generate structured data, but image data. One approach could be similar to the one used in generative adversarial networks (GANs), though you can potentially also utilise a simpler approach.

Then in general, the approach shall be similar to the one for structured data – obtain a dataset of images, use a training set to learn the generator, generate data, learn a model from it, and predict the images on the test set. Compare this to the baseline of using the training set directly. You can test your approach e.g. on the datasets for the Deep Learning topic. Another evaluation would be on a medical dataset with a clear classification task.

As this topic is more experimental and novel than others, you can focus on the problem and implementation, and perform a less detailed evaluation.

For GANs in python, plenty of examples are available, e.g. https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

One project that you can maybe draw inspiration from the medical domain is https://www.fda.gov/MedicalDevices/ScienceandResearch/ResearchPrograms/ucm477418.htm

One data set to use (with a classification task) could be e.g. https://www.kaggle.com/c/diabetic-retinopathy-detection/data

For GANs, generally consider either using one GAN per class, or conditional GANs, to be able to generate data for all classes.

Topic 3.3.2.: Generation and evaluation of sequential synthetic datasets

This is a rather experimental topic.

The above discussed tools for synthetic data generation have been developed for processing structured relational data. One application scenario for synthetic data is however as well in sequential data, which DNA sequences are one specific instance of.

One such method has been described by Pratas et al and uses multiple competing Markov Models. First, a training DNA sequence is partitioned into non-overlapping blocks of fixed size. Each block is handled by a particular Markov Model. In the first phase, the statistics of the training data are loaded into the models. After this deterministic procedure has ended, the models are kept frozen. In a second stochastic phase, the synthetic sequence is then generated according to the learned statistics.

Your task is to recreate the implementation of the specified paper, and evaluate the quality of genetic data when comparing it to real sequences and evaluating it on predictive tasks.

Main paper: D. Pratas, C. A. C. Bastos, A. J. Pinho, A. J. R. Neves, L. M. O. Matos. DNA Synthetic Sequences Generation using Multiple Competing Markov Models. IEEE Statistical Signal Processing Workshop (SSP), pp. 133-136, 2011.

Topic 3.3.3.: k-anonymity and utility of anonymized datasets

k-anonymity is a property of a dataset that contains the same information for at least *k* individuals, for every distinct information. *k*-anonymous datasets are obtained by systematically generalizing or suppressing the values of the dataset. These methods reduce the amount of information contained; therefore *k*-anonymous datasets necessarily have lower utility compared to their originals. Utility can be measured in different ways, e.g. using metrics directly on data such as a loss measure, or by data performance on a particular machine learning task.

The aim on this task is to investigate the relationship between

- utility metrics (squared error, entropy, granularity) and
- classification performance (classification accuracy, F1, ...)

of anonymized datasets.

You may utilize the anonymization algorithm of your choice:

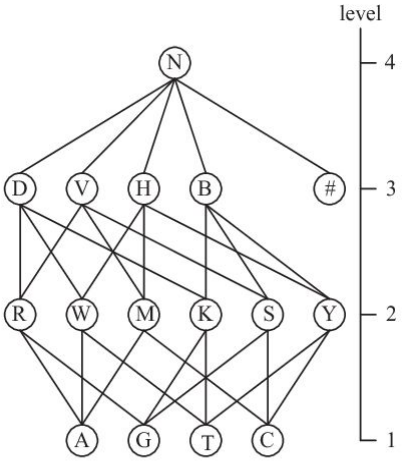
- Flesh (ARX) – API: https://github.com/arx-deidentifier/arx
- SaNGreeA: https://github.com/tanjascats/sangrep
- Mondrian: https://github.com/qiyuangong/Mondrian
- implementations from UTD Anonymization Toolbox: http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php?go=home

The anonymization algorithms usually offer multiple anonymized dataset as a result. Having multiple options for anonymizing the dataset may have certain benefits, e.g. in a scenario where generalizing some attribute is preferred over the others. Therefore, besides calculating the utility for one optimally anonymized dataset, you shall also analyse the difference in utility between optimal and other solutions.

In TUWEL specify datasets that will be used for the experiments. Write a report including all your findings, explain your methodology in detail and elaborate the conclusions.

Topic 3.3.4.: Genetic Data (DNA Lattice) Anonymization

The approach of DNA Lattice Anonymization (DNALA) has first been introduced in [1] and is applied to databases of DNA sequences. DNALA provides the possibility to prevent identification of DNA. For each DNA sequence, the technique finds the most similar sequence in the database and generalizes both to a common sequence. The distance between the sequences is computed according to the DNA generalization lattice in the figure below.



A = **A**denine	C = **C**ytosine
G = **G**uanine	T = **T**hymine
R = pu**R**ine	Y =p**Y**rimadine
S = **S**trong hydrogen	W = **W**eak hydrogen
M = a**M**ino group	K = **K**eto group
B = not A	D = not C
H = not G	V = not T
# = gap	N = i**N**determinate

For example, Adenine (A) and Cytosine (C) generalize to Amino Group (M), as can be seen by following the only arrows from A and C pointing to the same letter in the next level of the lattice. The function *gen(x,y)* returns the distance in the lattice between *x* and *y* and is defined by

$$gen(x,y) = 2level(z) - level(x) - level(y),$$

where *z* is the value *x* and *y* generalize to. E.g., we have $gen(A,C) = 2*2 - 1 - 1 = 2$. Subsequently, this function is used to define distances between whole DNA sequences.

Goal of this assignment: Read and study the original research paper by Malin. In particular, focus on understanding the DNALA method described in Section 3 and implement its core system (see Algorithm 2) in a programming language of your choice (preferred: python)

Primary reading:

B. A. Malin. *Protecting genomic sequence anonymity with generalization lattices.* Methods Inf Med., vol. 44, no. 5, pp. 687-692, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2227&rep=rep1&type=pdf

Further readings (optional):

G. Li, Y. Wang, X. Su. *Improvements on a privacy-protection algorithm for DNA sequences with generalization lattices.* Computer Methods and Programs in Biomedicine, vol. 108, no.1, pp. 1-9, 2012. https://www.sciencedirect.com/science/article/abs/pii/S0169260711000459?via%3Dihub

S. Wan, M. Mak, S. Kung, *Protecting Genomic Privacy by a Sequence-Similarity based Obfuscation Method.* 2017. https://arxiv.org/abs/1708.02629

Topic 3.3.5.: Differential Privacy

Differential privacy is introduced by Cynthia Dwork [1] as a set of privacy-preserving mechanisms for publicly sharing information about the dataset by descriptive characteristics rather than sharing the information about the individuals in the dataset. Differential privacy is at first achieved with simple aggregate statistics and descriptive statistics such as mean function, maximum, histograms, etc., and recently the advances with differentially private machine learning models are achieved, as well.

Differential privacy is achieved by adding noise in the process. Depending on the phase where the noise is added we differentiate:

- Input perturbation - adding noise to the input before running the algorithm
- Internal perturbation - randomizing the internals of the algorithm
- Output perturbation - add noise to the output, after running the algorithm

The goal of this exercise is to compare these approaches for achieving differential privacy, from the privacy aspect and model performance aspect and analyse the tradeoff between privacy and performance. To that end, to achieve internal perturbation, utilize a differential privacy tool that provides differentially private versions of classification and clustering models (for Python: https://github.com/IBM/differential-privacy-library, for R: https://github.com/brubinstein/diffpriv). Familiarize yourself with the differentially private mechanisms used in the chosen tool.
Secondly, apply an input perturbation and output perturbation based on the sensitivity method proposed by Dwork et al. in [1]. You may experiment with a model of your choice.

You shall analyse:

1. Input perturbation vs. internal perturbation vs. output perturbation approach on a chosen model; difference between the models' performances, and differences to the non-private model.
2. Trade-off between privacy and model performance for every approach.

Experiments from these papers [2, 3] may serve as an inspiration for the methodology of this exercise.

References:

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Theory of Cryptography*, Berlin, Heidelberg, 2006, pp. 265–284. PDF: https://link.springer.com/content/pdf/10.1007/11681878_14.pdf

[2] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially Private Empirical Risk Minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011. PDF: https://arxiv.org/pdf/0912.0071.pdf

[3] K. Fukuchi, Q. K. Tran, and J. Sakuma, "Differentially Private Empirical Risk Minimization with Input Perturbation," in *Discovery Science*, Cham, 2017, pp. 82–90. PDF: https://arxiv.org/pdf/1710.07425.pdf

Further readings - differentially private ML:

[4] O. Sheffet, “Private Approximations of the 2nd-Moment Matrix Using Existing Techniques in Linear Regression,” *arXiv:1507.00056 [cs]*, Nov. 2015.

[5] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, “Differentially Private Naive Bayes Classification,” in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Nov. 2013, vol. 1, pp. 571–576.

[6] H. Imtiaz and A. D. Sarwate, “Symmetric matrix perturbation for differentially-private principal component analysis,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 2339–2343.