Two sides of relevant information : anonymized representation through deep learning and predictor monitoring

Clément Feutry

université PARIS-SACLAY

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

NNT : 2019SACLS479

Thèse de doctorat

Two sides of relevant information: anonymized representation through deep learning and predictor monitoring

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Réseaux, information et communications

Thèse présentée et soutenue à Gif-sur-Yvette, le 13 décembre 2019, par

CLÉMENT FEUTRY

Composition du Jury :

Anissa Mokraoui Professeur des universités, Université Paris 13 Sorbonne Paris Cité	Président
Teddy Furon Chargé de recherche, INRIA Rennes	Rapporteur
Sébastien Gambs Professeur, Université du Québec à Montréal	Rapporteur
Chloé Clavel Maître de conférences, Télécom Paris	Examinateur
Pierre Duhamel Directeur de recherche, Université Paris-Sud	Directeur de thèse
Pablo Piantanida Professeur associé, CentraleSupélec	Co-directeur de thèse

UNIVERSITÉ PARIS-SUD

DOCTORAL THESIS

Two sides of relevant information: anonymized representation through deep learning and predictor monitoring

Laboratoire des Signaux et Systèmes
Information Theory and Applications

CentraleSupélec

13 December 2019

"*La clef de toutes les sciences est sans contredit le point d'interrogation ; nous devons la plupart des grandes découvertes au : Comment ? et la sagesse dans la vie consiste peut-être à se demander à tout propos : Pourquoi ?*"

La Peau de chagrin (1831), Honoré de Balzac, éd. Gallimard, coll. « Folio », 1974, L'Agonie, p. 339

UNIVERSITÉ PARIS-SUD

Abrégé

Laboratoire des Signaux et Systèmes

Doctorat de recherche

Deux aspects de l'information utile : représentation anonymisée par l'apprentissage profond et surveillance de prédicteur

par Clément FEUTRY

L'apprentissage automatique est un sujet en pleine expansion. Les améliorations technologiques des moyens de calcul supportant l'apprentissage profond ont rendu ce domaine de recherche de plus en plus populaire année après année. L'apprentissage profond a successivement supplanté les meilleurs résultats sur un large éventail de tâches : la reconnaissance de formes sur des images, le suivi d'objet sur des vidéos, le traitement du langage naturel sur fichier audio et l'extraction de données dans les bases de données textuelles. De manière corrélée, des préoccupations se font jour: vie privée et fiabilité. La vie privée des individus a même été l'objet d'une loi européenne sur la protection des données qui a été promulguée pendant ce doctorat. La fiabilité, elle pose question dans beaucoup d'applications où des décisions sont prises automatiquement, sans contrôle de l'utilisateur.

Le travail présenté ici est pour une première partie à l'intersection de l'apprentissage profond et anonymisation. Un cadre de travail complet est développé dans le but d'identifier et de retirer, dans une certaine mesure et de manière automatique, les caractéristiques privées d'une identité pour des données de type image. Deux méthodes différentes de traitement des données sont étudiées : une supervisée et une semi-supervisée. Ces deux méthodes partagent une même architecture de réseau en forme de Y et cela malgré des différences concernant les types de couches de neurones utilisés conséquemment à leur objectif d'utilisation. La première méthode de traitement des données concerne la création *ex nihilo* par apprentissage de représentations anonymisées permettant un compromis entre la conservation des caractéristiques pertinentes et l'altération des caractéristiques privées. Ce cadre de travail a abouti à une nouvelle fonction de perte, à une architecture et un méthode d'entrainement adaptées. Le deuxième type de traitement des données ne fait usage d'aucune information pertinente sur ces données et utilise uniquement des informations privées; ceci signifie que tout ce qui n'est pas une caractéristique privée est supposé pertinent et doit être conservé dans la mesure du possible. Cela implique que les représentations anonymisées sont de même nature que les données initiales (une image est transformée en une image anonymisée). Cette tâche a conduit à un léger changement de la fonction coût et à un autre type d'architecture (toujours en forme de Y). Les résultats fournis dans ce contexte sont fortement sensibles au type des données mais ils produisent des images, quel que soit le type de données, pouvant tromper un œil humain.

La seconde partie de mon travail concerne une autre sorte d'information utile : cette partie se concentre sur la surveillance du comportement des prédicteurs. Dans le cadre de l'analyse de "modèle boîte noire", on a uniquement accès aux probabilités que le prédicteur fournit (sans aucune connaissance du type de structure/architecture qui produit ces probabilités). Cette surveillance est effectuée pour détecter des comportements anormaux. L'étude de ces probabilités peut servir d'indicateur d'inadéquation potentiel entre les statistiques des données et les statistiques du modèle. Ceci permet potentiellement de détecter une anomalie et de demander la vérification de l'adéquation entre modèle et données par un opérateur humain. Deux méthodes utilisant différents outils sont présentées. La première compare la fonction de répartition des statistiques de

sortie d'un ensemble connu et d'un ensemble de données à tester. Cette comparaison se fait à l'aide d'un test statistique et par conséquent nécessite une quantité de données significative pour chaque décision. En revanche la seconde méthode nécessite moins de données pour chacune des décisions. Cette seconde méthode fait intervenir deux outils : un outil reposant sur l'incertitude du classifieur et un autre outil reposant sur la matrice de confusion. Ces méthodes produisent des résultats concluants en terme de détection de comportements anormaux.

UNIVERSITÉ PARIS-SUD

Abstract

Laboratoire des Signaux et Systèmes

Doctor of Philosophy

Two sides of relevant information: anonymized representation through deep learning and predictor monitoring

by Clément FEUTRY

Machine learning is a booming topic. The improvement of deep learning compatible computation means made the subject more popular years after years. The deep learning successively overtook best results on a wide variety of tasks from pattern recognition in images, object tracking in video, natural language processing in sound signal and data mining in large text database. Correlated serious topics arise: privacy and reliability. Individual privacy is a concerning matter that was subject to a new European data protection law that was enacted during the length of this doctorate. Reliability is concerning too. In many applications decisions are taken in an automated manner without constant operator (human) monitoring.
The work presented here is for a first part at the cross section of deep learning and anonymization. A full framework was developed in order to identify and remove to a certain extant, in an automate manner, the features linked to an identity in the context of image data. Two different kind of processing data were explored. They both share the same Y-shaped network architecture despite component of this network varying according to the final purpose. The first one was about building from the ground an anonymized representation that allowed a trade-off between keeping relevant features and tampering private features. This framework has lead to a new loss. The second kind of data processing specified no relevant information about the data, only private information, meaning that everything that was not related to private features is assumed relevant. Therefore the anonymized representation share the same nature as the initial data (e.g. an image is transformed into an anonymized image). This task lead to another type of architecture (still in a Y-shape) and provided results strongly dependent of the type of data.
The second part of the work is relative to another kind of relevant information: it focuses on the monitoring of predictor behavior. In the context of black box analysis, we only have access to the probabilities outputted by the predictor (without any knowledge of the type of structure/architecture producing these probabilities). This monitoring is done in order to detect abnormal behavior that is an indicator of a potential mismatch between the data statistics and the model statistics. Two methods are presented using different tools. The first one is based on comparing the empirical cumulative distribution of known data and to be tested data. The second one introduce two tools: one relying on the classifier uncertainty and the other relying on the confusion matrix. These methods produce concluding results.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial intelligence,
AUPR	Area under the precision recall,
AUROC	Area under the receiver operating characteristic,
DCNN(s)	Deep convolutional neural network(s),
DNN(s)	Deep neural network(s),
eq.	equation,
Fig.	Figure,
FNR	False negative rate,
FPR	False positive rate,
GAN	Generative adversarial networks,
GM	Geometric mean,
KL	Kullback-Leibler divergence,
PD(s)	Probability distribution(s),
PDF	Probability density function,
RV(s)	Random variable(s) ,
SD	Standard deviation,
SGD	Stochastic gradient descent,
TNR	True negative rate,
TPR	True positive rate,
VAE	Variational auto-encoder.

List of Symbols

$\mathcal{X}$	dataset.
$\mathcal{Y} \equiv \{0, 1, \ldots, C-1\}$	label alphabet of size C.
$(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$	is a sample and its associated label.
$\mathbb{E}_P[\cdot]$	denotes the expectation w.r.t. P the probability distribution (PD).
$\mathcal{P}(\mathcal{X})$	denotes the set of all PDs in $\mathcal{X}$.
$\widehat{p}_X$	denotes the empirical PDs computed over the samples .
$\mathbf{p}_X$	is the vector length that contains the values of p_X.
$\lvert \cdot \rvert$	is used for the usual absolute value and cardinality of a set
$\langle \cdot, \cdot \rangle$	the canonical inner product.

Upper-case letters denotes random variables : $X \sim p_X$,
lower-case letters denotes realizations: x,
$\widehat{\cdot}$ over a random variable name denotes an estimate: $\widehat{Y}$ is an estimate of Y,
$\widehat{\cdot}$ over a function name denotes that the function is taken over empirical distribution,
All logarithms are taken with base e.
The information measures are (see Csiszar and Korner, 1982):

- *entropy*: $\mathcal{H}(X) := \mathbb{E}_{p_X}\left[-\log p_X(X)\right]$,
- *conditional entropy*: $\mathcal{H}(Y|X) := \mathbb{E}_{p_{XY}}\left[-\log p_{Y|X}(Y|X)\right]$,
- *coss-entropy* $\mathcal{H}_{XY}(Y|X) := \mathbb{E}_{p_{XY}}\left[-\log p_{Y|X}(Y|X)\right]$,
- *relative entropy* also called *Kullback-Liebler divergence*: $\mathcal{D}(p_X \| q_X) := \mathbb{E}_{p_X}\left[-\log \dfrac{p_X(X)}{q_X(X)}\right]$,
- *mutual information*: $\mathcal{I}(P_X; P_Y) = \mathbb{E}_{p_X} D(P_{Y|X} || P_Y)$,
- *conditional relative entropy*: $\mathcal{D}(p_{U|X} \| q_{U|X} | p_X)$.

The following probabilities are used in the chapter 5 with this particular definition: p_X, $p_{\widetilde{X}}$ and p_Y are respectively the distribution of the train data samples, the distribution of test samples and the distribution of the labels.

To my Family

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence (AI) aims at building a device that can perform cognitive tasks. The word intelligence come from the Latin: *intelligentia* (the power of discernment, understanding) itself derived from the verb *intellegere* (understand, comprehend, perceive, discern) which is built from the two words *inter-* (between) and *legō* (choose, read). Most of these words describe tasks that are hot topics in AI research.

AI covers a really large field of which machine learning is an important part. Machine learning main principle is to extract information from data statistics: The model you create needs to be fed with large amount of data samples to extract relevant underlying statistic, and this extracted information is used as knowledge to process new data.

Deep learning may be the most popular subtopic of machine learning. Deep learning aims to model the processed data with high level of abstraction using specific layered architectures, and specific non linear functions. Originally deep learning was intertwined with neural networks. Neural networks are using neurons that mimic the behavior of biological neurons as elementary bricks. Diversified networks are now used in deep learning mostly adapted to the type of data processed. Building the right kind of model is not enough if it is not properly trained. Here again depending on the kind of goal to be achieved, there are several method of training your model for learning relevant statistics. The main categories are supervised training and unsupervised training. For the first one the device is trained on examples on which the answers are known and used for the training. For the second one the device is only shown examples without any answer provided. Both theses methods behave differently and are used to perform different tasks most of the time.

The popularity increase in machine learning comes from major breakthroughs that have been achieved since more than ten years. It performs overwhelmingly good in classification tasks. Its use have widen to a variety of fields and applications. Deep learning methods provide effective tools to handle and process large datasets. Digital information lies and is collected everywhere: users profile information and behavior on commercial websites, user notation on ranking websites, banking, insurance or even medical insurance information, phone collected information. As the digital era rises, information has became a more and more important resource sometimes having monetary value and yet accessible to company. Processing it properly is more and more a core obligation.

This raises the question of privacy and anonymity which have become a hot topic. It has become subject to new European Union laws in the courses of the last few years in order to protect privacy of individuals. This is a first step toward a more ethic way of data processing that this work also tries to address: How to protect (hide) sensitive information while processing data ?

The work presented in this thesis relies on the hypothesis that the owner of the data is willing to protect sensitive information from being disclosed. We investigate new methods based on representation learning and deep neural networks, and motivated by novel information-theoretical bounds applied to anonymization. We introduce a novel

training objective for simultaneously training a predictor over target variables of interest (the regular labels) while preventing an intermediate representation to be predictive of the private labels. We demonstrate the success of this approach for two distinct classification versus anonymization tasks (handwritten digits and sentiment analysis). We further introduce a semi-supervised method of learning invariant representation while preserving most of the data, tackling even the case of strong intertwinement and we successfully applied it to the same anonymization tasks.

In recent years, some datasets containing sensitive information about individuals have been released into public domain with the goal of facilitating data mining research. Databases are frequently anonymized by simply suppressing identifiers that reveal the identities of the users, like names or social security number home adress. However, even these definitions cannot prevent background attacks, in which the attackers already know something about the information contained in the dataset. An attacker can for example use side information, meaning information acquiered from a different source or database.

In this thesis we address the interplay between deep neural networks and statistical anonymization of datasets. We focus on the following fundamental questions: *What conditions can we place to learn invariant (or sanitized) representations in order to minimize the amount of information which could be revealed about a specified variable? What is the effect of sanitization on these procedures? How to sanitize without prior knowledge of the regular tasks?* The line of research we investigate is based on privacy-preserving statistical methods, such as learning differentially private algorithms [Abadi et al., 2016] from the supervised case. The main goal of this framework is to enable an analyst to learn relevant properties (e.g., regular labels) of a dataset as a whole while protecting the privacy of the individual contributors (private labels which can identify a person). Even further we address the hypothesis of an analyst removing in a semi-supervised manner information related to a known private variable (e.g., private label) with no prior about usefulness or relevant properties (no label of any regular task). This assumes the database is held by a trusted person who can release freely information about regular labels, e.g., in response to a sequence of queries, and used for many new purposes.

Another problem of rising concern is the reliability of the results provided by such algorithms. Once the algorithm has been running, is there any way of measuring the degree of confidence one can have in its outputs ? This very sensitive question is addressed in the last part of this work.

1.2 Artificial intelligence

Before designing the first machine, mankind has dreamt about building complex device that imitates human action. The ultimate purpose of this quest is to build a human like working automaton capable of human behavior such as thinking and learning. From mythological Hephaestus automaton blacksmith workers that could forge for their creator, to the giant automation Talos bodyguard of Crete, the ancient time have several tales about cognitive device. Since then, comparatively recent yet complex mechanical automaton capable of writing predefined text and drawing a specific sketch (Maillardet's automaton, built in London circa 1800 by the Swiss mechanician, Henri Maillardet) or playing flute (The Flute Player, invented by the French engineer Jacques de Vaucanson in 1737, actually played a real flute), the dream has slowly taken shape from the mind of their inventor to the real world, following or provoking technical innovation. Since the foundation of the computer science the gap has reduced dramatically but yet remains of significant size. Machines present more and more accuracy to their specific task, but yet a specific task trained machine would not perform well on another task it is not trained for. Machines learn efficiently mostly on tasks they were designed for. They lack the plasticity and adaptability of the human brain capable to learn from a wide variety of tasks: this is where I think the next

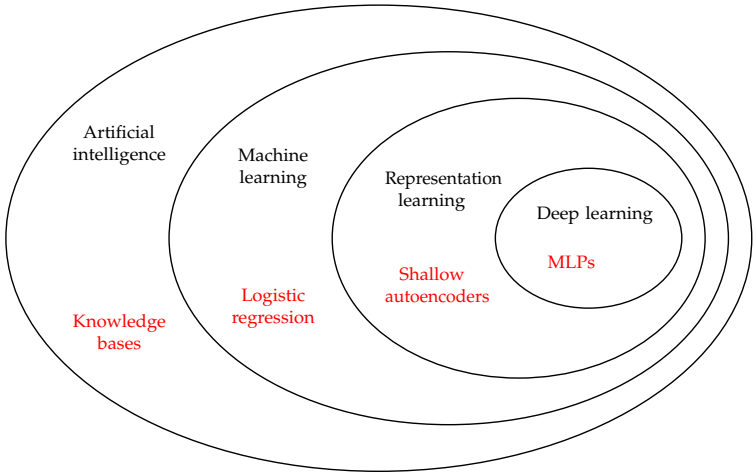


FIGURE 1.1: Venn diagramm showing the relative position of machine learning among artificial intelligence. Red text shows a related example.

breakthrough will occur, machines that can learn by themselves when exposed to new data, the way human babies learn when exposed to new stimuli.

Artificial intelligence (AI) is only a meaningful designation when opposed to the 'not-artificial' intelligence, which is the natural intelligence which primary example is the human intelligence. AI devices should be able to exhibit human like cognitive ability such as learning and problem solving. The machine most definitely mimic some human abilities and often they outperform the human baseline on some tasks. The artificial part is not to be neglected in the explanation of nowadays AI Just like the automaton of the XVIIIth century were only capable of performing action within their limited capacity/reach, most AI can only process information and take action within the specific range of action/response their human builders have design them. The most fallacious thing about artificial intelligence is actually its name: AI machines are definitely artificial, but they are not for now endowed of any intelligence. The main meaning of intelligence in this context is the fact that the device, when shown a problem can provide an answer to it; this answer being, with reasonable probability, a/the good solution to the problem. One of the most impressive capacity some AI machines are endowed with is a capacity of generalization: with the right training samples (both quality and quantity), a model can infer some results on brand new data samples, provided that new samples share an underlying structure with the training samples. Other AI technology may rely on different frameworks like for example a comprehensive and structured storing of knowledge into memory base.

1.3 Introduction to machine learning

Machine learning, as shown in Figure 1.1 (Loosely inspired from [Goodfellow, Bengio, and Courville, 2016, Ch. 1, p. 9]) is a specific subset of artificial intelligence. A general definition of machine learning is the following: the study of statistical models and algorithms that computer systems used to perform a task, without using explicit instructions, but relying on pattern and inference. The term of Machine Learning was first used at IBM in 1959, where Arthur Samuel (researcher in computer gaming and artificial intelligence, senior member of the TeX community) first invented the term.

1.3.1 Basics

The most important part of the model is designed by a human operator: the architecture is designed based on decision of the operator. Initialization and optimization pattern are also operator driven. The part of the model which is build without the help of the operator, which is *learned*, is the value of the various architecture parameters. These

values are computed using training data, to make prediction without any special hard coded data specific instruction. As stated above, the main assumption of machine learning is that a mathematical model can generalize: being optimized to perform a given task on training samples should produce a model that performs well on new test data provided that these data are "close enough" to training data. The notion of close enough means that data from training set and test set are similar but not identical: similar data share the same underlying features and theses features can be accessed in the same way for both sets. For example if the training examples consist only on vertically aligned number and the test set contains only randomly tilted number, the model would not be able to generalize. If the training set also has a fair share of tilted numbers, the model will have good performance on the randomly tilted number test set.

The elementary bricks used to make machine learning model are neural networks. Neural networks were first introduced by [McCulloch and Pitts, 1943] who created a computational model for neural networks. This paper first tries to model the living being nervous system activity. From this funding paper, two main approaches have derived, the study of biological processes and the study of artificial intelligence processes.

Neuron

The neuron is the most elementary brick of a model. A simple neuron representation is shown Figure 1.2. It works by applying two math functions to its input: A first linear function called pre-activation function: $a(\mathbf{x}) = (W\mathbf{x} + b)$ where for a vector input $\mathbf{x}$, W is the weight vector, b is the bias term. The weight vector is noted with a simple capital letter to stand for an horizontal vector. The second function, called the activation function, is applied to the pre-activation and gives the output of the neuron: $f\big(a(\mathbf{x})\big) = f(W\mathbf{x} + b)$ where f is the activation function (preferably non linear). The most common non linear functions are the following: rectified linear unity (relu which is plotted on the Figure 1.3), sigmoid function and hyperbolic tangent. W and b are the parameters that are learned by the model during the training phase.

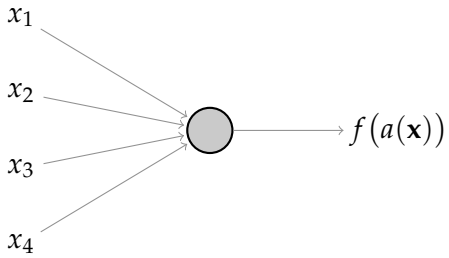


FIGURE 1.2: Example of a representation of a neuron with a four components input vector.

Layer

The above elementary bricks are assembled to make a layer as shown on Figure 1.4 in the case of a feed forward layer. Each component of the input vector is used to compute the output of each neuron. The information from the different neurons are then interpreted as the components of the output vector of the layer. The previous activation function can be seen as a vector function which components are each related to one particular neuron. The concatenation of all these components leads to the following equation: $\mathbf{a}(\mathbf{x}) = (\mathbf{W}\mathbf{x} + \mathbf{b})$ where for a vector input $\mathbf{x}$, $\mathbf{W}$ is now the weight matrix, $\mathbf{b}$ is the vector bias and $\mathbf{a}$ is the pre-activation vector (each component is the pre-activation function applied to $\mathbf{x}$). Similarly to the neuron case, the activation function is applied to the pre-activation and gives the output of the neuron: $\mathbf{f}\big(\mathbf{a}(\mathbf{x})\big) = \mathbf{f}(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{f}$ is the vector activation function (which components are activation functions

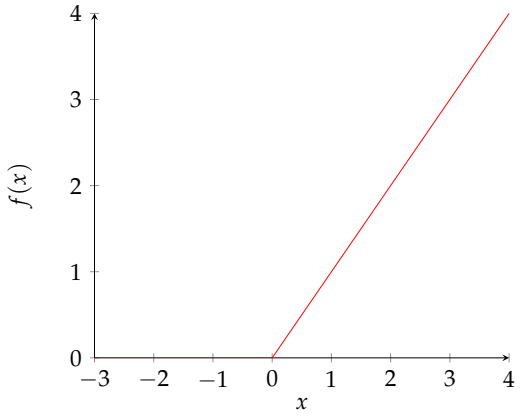


FIGURE 1.3: Plot of f as the Relu function

f) and $\mathbf{b}$ is the vector of bias (which components are b_i). Dense layer is another name of a feed forward layer. Note that there is a wide variety of types of layers, other than feedforward layer.

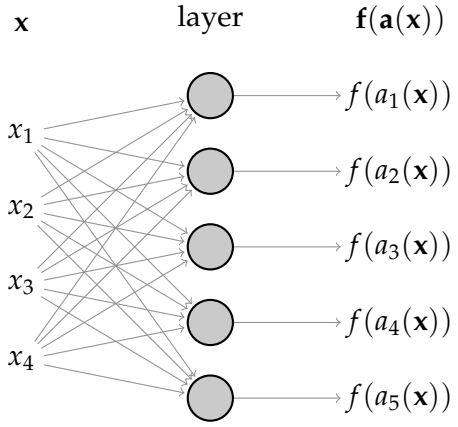


FIGURE 1.4: Example of a neural layer

Deep neural network

If you stack layers on top of each other, using the output of the previous layer as the input for the next layer, you produce a deep neural network. This means that the information is processed through several layers before reaching the output of the model as shown on Figure 1.5 with two layers. The inside layers of a deep neural network are usually called hidden layers. Here layers 1 and 2 are hidden layers. The layer i parameters are within its pre-activation function noted $\mathbf{a}^{l_i}$ Each hidden layer output is computed using the previous layer value. The layer number 3 is the output layer. Note that the denomination of the deep neural neural network starts at network that have at least two hidden layers i.e. only one hidden layer is not sufficient to be called deep. Models may present a wide variety of depth, sizes, and types of layer combination.

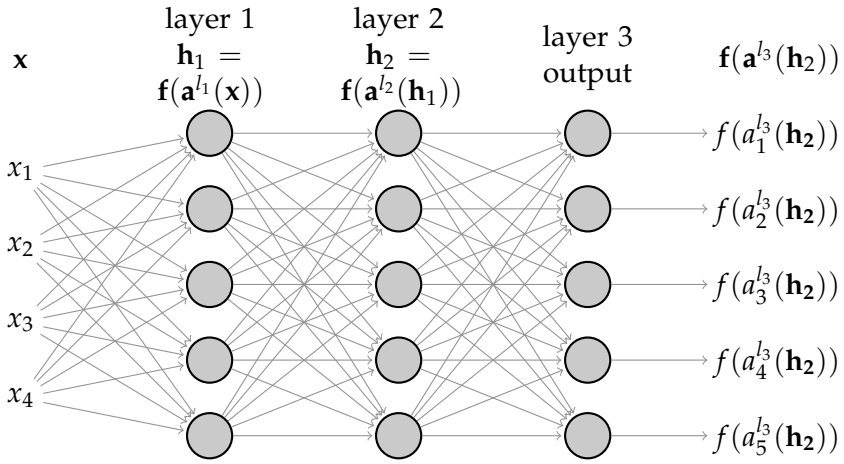


FIGURE 1.5: Example of a simple feedforward deep neural network

Output of neural network

Neural networks' output layer is most of the time defined by its purpose. Note that the output layer characteristics are defined by the task the network should perform.
For a classification task the output layer has usually the same number of neurons as the number of different classes labeled. The activation function too is adapted to the task of classification. The softmax function $\boldsymbol{f}$ of components f_i define here for a vector $\boldsymbol{t}$ of components t_i:

$$f_i : \boldsymbol{t} \to \frac{\exp(t_i)}{\sum_{k=0}^{C-1} \exp(t_k)}, \tag{1.1}$$

is a normalized function that provide the probability for a sample to belong to each possible class. Output of such a model are a distribution of probability over the different classes and are also called the soft outputs of the model. From this values, the prediction of the model is assed by taking the argmax:

$$j_{pred} = \underset{i \in [0:C-1]}{\arg\max}\, f_i(\boldsymbol{t}). \tag{1.2}$$

For a model that should reconstruct an image, the number of neurons of the output layer is the number of pixels the reconstructed image should have. In this case a sigmoid function is used as an activation function in most cases. The sigmoid function is given for a variable t:

$$f : t \to \frac{1}{1 + \exp -t}. \tag{1.3}$$

Dropout

One of most used method of regularization is dropout. The term dropout comes form the action of dropping out neurons during the training. The regularization method is done by randomly selecting a fixed number of neurons in a layer and removing their connections to other neurons for each batch during the training. This prevents part of the layer to be overused compared to other part. When this method is applied to the whole network, it prevents over-fitting because the network cannot rely on specific neurons. It is described in [Hinton et al., 2012] as an efficient method to perform model averaging. It is worth mentioning that during the testing phase, all the neurons are used.

1.3.2 Famous type of neural network

Brief intoduction to multilayer perceptron

The multilayer perceptron (MLP) is a class of feedforward neural network. Note that feedforward neural network have an internal structure where the nodes do not form a cycle. In this simplest architecture, the information moves only forward from the input through the network and to the output nodes. An MLP consists of at least three layers: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. Many additional hidden layers can be added. This network can distinguish data that is not linearly separable.

Historically the perceptron was first introduced by F. Rosenblatt in 1957 at Cornell Aeronautical Laboratory. He further developed his work in [Rosenblatt, 1961]. The main novelty of his perceptron related work was that exposing examples to these models was a way to train them to perform better on a specific task. This special ability allowed this model to earn quite some fame at the time: most the scientific community praised his work and it was made intelligible to the general public through a New York Times article titled "New Navy Device Learns By Doing". The term of learning applied to a system was already present in this 1950s article. The major flaw of the nowadays perceptron ancestor was the lack of non linearity, which bounded it to learn only linear function. This limitation to linear function was overcomed by applying a method taken from control theory. This method called chain rule in control theory was introduced in 1974 as backpropagation in the neural network community by [Werbos, 1975]. Note that other scientists were working at the time on this method and Werbos name remains because he first mentioned it potential use on neural network. Backpropagation method is efficient to produce an iterative and recursive method to compute the network parameters updates values in order to improve the model behavior in this particular task.

Autoencoder

An autoencoder is a type of neural network that tries to copy its input to its outputs (i.e. without using the trivial solution of a network learning the identity function). Its paternity is disputed and many autoencoders' core ideas come from several people: [Lecun and Fogelman Soulie, 1987], [Bourlard and Kamp, 1988] and [Hinton and Zemel, 1994]. It is an unsupervised model, therefore it does not require any label during the training. The main decomposition of its architecture uses the notion of a code at the hidden layer h so that the network is divided in two parts: the part of the network from the input to h called the encoder and the part from h to the output called the decoder.

The important feature of the autoencoders is that one prevents them from learning the identity function. This makes autoencoders learn relevent representation and 'coding scheme'. In its simplest form the autoencoder is a MLP with one hidden layer (.i.e. feedforward network with an input layer, an hidden layer and a output layer.) where the output layer as the same number of neurons as the input layer. In this case the network loss measures the discrepancy between the input sample and the generated output and aims to minimize the reconstruction error, for example using the mean squared error between the sample and the output.

The fact that one prevents the model to simply copy the input makes the network learns features about the set whose samples it should reconstruct: these features are called the hidden representations. Indeed the reconstruction is not really the interesting part of the network. The interesting part is the hidden features the encoder learns to extract. Note that the most recent autoencoders encoders and decoders have transitioned from deterministic mapping to stochastic mapping.

Denoising autoencoders: The denoising autoencoders force the learning of relevant features through a specific framework. The original samples are combined with a

random noise before going through the autoencoder. The output is then compared to the original sample (i.e. without the noise). This prevents the network to learn the identity function and on the opposite it forces the extraction of relevant features despite the additive noise. The noise here is used as a regularization method. Other regularized autoencoder examples are the sparse autoencoder and the contractive autoencoder that respectively add a sparsity constrain and a regularized norm term to the loss (the regulizer term forces the autoencoder to learn a function robust to small variations of input values).

Variational autoencoders: The variational autoencoders are quite different, they are generative models. This kind of autoencoders aims to map the distribution of the learnt hidden feature to a given distribution (multivariate Gaussian distribution in the wide majority of case). Once the training is done, the features actually follow the given distribution which makes the code space a continuous space. From here using information of the mean and standard deviation makes it possible to make some mathematical computation on the feature, like vectorial addition or subtraction.

Generative adversarial network: GAN

The generative adversarial network (GAN) was introduced by [Goodfellow et al., 2014]. The GAN framework aims at generating likely samples compared to an original database in an unsupervised manner. This framework makes two neural networks compete against each other in a game (under the game theory definition). The two networks names are respectively the generator and the discriminator. The generator input receives some noise and uses it to *generate* an output. The discriminator is a classifier network that is trained to distinguish between generated samples and original samples from the dataset. The main purpose of the generator is to fool the discriminator and conversely the purpose of the discriminator is to be able not to be fooled by the generator output. The core idea of this scheme is that the relevant information that allowed the discriminator to properly classify samples is used to improve the generator performances. Moreover, as the discriminator gets better, the generator learning uses more and more precise details and thus produces generated samples closer and closer to the dataset original samples. This means that both networks have to perform better together in order to improve the overall results. Here actually lies the main weakness of the GAN framework: Training such a pair of networks in a productive manner is highly unstable: if one outperforms too much the other, the generative process and the quality of the produced samples is jeopardized. A lot of recent work on GAN is oriented toward method to robustify the training such as the famous introduction of the Wasserstein distance by [Arjovsky, Chintala, and Bottou, 2017]. The Wasserstein distance is introduced in the GAN framework as a similarity metric to compare probabilities distribution with a smooth measure that gives a finite value for disjoint support distribution, whereas Kullback-Liebler divergence and Jensen-Shannon divergence lack smoothness in this case (respectively infinite value and differentiability issue).

As stated above, both networks play a minimax game: the generator aims at maximizing the error rate of the discriminator while the discriminator aims at having good detection performances i.e. minimizing its error rate. This specific result is obtained via the following loss:

$$\mathcal{L}(D, G) = \mathbb{E}_{\hat{P}_x}\left[\log(D(x))\right] + \mathbb{E}_{\hat{P}_z}\left[\log(1 - D(G(z))\right], \tag{1.4}$$

where:

- $D(x)$ is the discriminator's estimate of the probability that real sample instance x is real,
- $\hat{P}_x$ is the empirical distribution of the real samples,

- $\mathbb{E}_{\hat{P}_x}$ is the expected value over all real data instances,
- $G(z)$ is the generator's output when given the noise realization z,
- $D(G(z))$ is the discriminator's estimate of the probability that a fake instance is real.
- $\hat{P}_z$ is the empirical distribution of z
- $\mathbb{E}_{\hat{P}_z}$ is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances $G(z)$).

Note that this formula derives from the binary cross-entropy between the real and the generated distributions. Indeed we want to ensure that the discriminator's decisions over real data are accurate by maximizing the first term of eq. (1.4). Meanwhile, given a fake sample $G(z)$, the discriminator is expected to output a probability, D(G(z)), close to zero by maximizing the second term of eq. (1.4). Simultaneously the generator is trained to increase the chances of D producing a high probability for a fake example, thus to minimize the second term. Therefore, the GAN framework is performing the following min-max equation:

$$\min_G \max_D \mathcal{L}(D, G). \tag{1.5}$$

1.4 Information measure

The information is the resolution of uncertainty as defined by [Shannon, 1948] in the case of communication of information over a noisy channel. Therefore measuring a quantity of information is measuring the uncertainty. If we take for example the random variable X with its associated distribution p_X and x a realization of X. The information given by this realization is linked to its probability $p_X(x)$. The fact that the event is unlikely means that observing it brings more information that observing a likely event, which brings less information. The measure of information of the random variable X called entropy is defined as the average of information of each possible realization:

$$\mathcal{H}(X) \coloneqq \sum_{x \in \mathcal{X}} -p_X(x) \log p_X(x) \tag{1.6}$$

$$\mathcal{H}(X) \coloneqq \mathbb{E}_{p_X}\left[-\log p_X(X)\right]. \tag{1.7}$$

Given a second random variable Y, with the distribution p_Y, we introduce the mutual information between X and Y. The mutual information is the quantity that measures the amount of information about one random variable that can be deduced from the observation of the other. How will be affected the knowledge of the information of X (i.e. its uncertainty) if a realization of Y is observed ?

$$\mathcal{I}(X;Y) \coloneqq \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x) p_Y(y)} \tag{1.8}$$

$$\mathcal{I}(X;Y) \coloneqq \mathbb{E}_{p_{XY}}\left[\log \frac{p_{XY}(X,Y)}{p_X(X) p_Y(Y)}\right]. \tag{1.9}$$

The mutual information can be decomposed using the entropy of X and the entropy of X knowing Y:

$$\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y). \tag{1.10}$$

Note that because the mutual information is always positive or 0, it means that on average, knowing the realization of a random variable decreases the uncertainty about the other random variable except if the two random variables are independent and the uncertainty is unchanged (no information is provided).

The Kullback-leibler divergence (KL) is a quantity that measures the difference between a distribution and a reference distribution. The expression of this divergence is the following:

$$D(p_X||q_X) = \sum_{x\in\mathcal{X}} p_X(x) \log\left(\frac{p_X(x)}{q_X(x)}\right). \tag{1.11}$$

The KL is linked to the mutual information:

$$\mathcal{I}(P_X; P_Y) = \mathbb{E}_{p_X} D(P_{Y|X}||P_Y). \tag{1.12}$$

Note that the KL is a divergence and not a distance, despite being always non-negative. KL is not symmetric and does not satisfy the triangular inequality.

The cross-entropy is defined as:

$$H(p_X, q_X) = -\sum_{x\in\mathcal{X}} p_X(x) \log q_X(x) \tag{1.13}$$

$$= \mathbb{E}_{p_X}\left[-\log q_X\right], \tag{1.14}$$

this can be written using the KL and the entropy

$$H(p_X, q_X) = \mathcal{H}(p_X) + D(p_X||q_X). \tag{1.15}$$

In a machine learning framework, crossentropy is used as one of the most usual loss function where the first distribution is the target distribution (i.e. labels in the case of a classifier's training) and the second distribution is the classifier soft outputs. Rényi entropy is a more general formulation of entropy whose Shannon entropy is a limit case. For a parameter α it is written:

$$\mathcal{H}_\alpha(X) = \frac{1}{1-\alpha} \log\left(\sum_{x\in X} p_X^\alpha(x)\right), \tag{1.16}$$

from this definition of entropy it is possible to derive Rényi divergence:

$$D_\alpha(p_X||q_X) = \frac{1}{\alpha-1} \log\left(\sum_{x\in X} \frac{p_X(x)^\alpha}{q_X(x)^{\alpha-1}}\right), \tag{1.17}$$

when α tends toward 1, the limit of D_α is the KL.

Useful inequalities: The data processing inequality states that mathematical operation (deterministic or stochastic) done on a random variable cannot increase the quantity of information carried by the random variable. Considering f the function applied to X as a processing, the inequalities is:

$$\mathcal{I}(X, Y) \geq \mathcal{I}(f(X), Y). \tag{1.18}$$

The equality only holds if the f function is one to one. This strong inequality states counter intuitively that feed forward neural network can at best maintain the initial quantity of information along successive layers. The hidden reality here that the usual databases hold tremendous amount of information, losing some along the way is a bearable cost especially if it allows to reduce the sample dimension in a way that eases the learning of accurate predictions.

Fano's inequality is another important inequality that allows to find a lower bound of the probability of error in the case of estimating a random variable using another random variable by the conditional distribution. Let us say that the g function outputs an estimate of X $g(Y) = \hat{X}$, Fano's inequality is written as follow:

$$\mathcal{H}(X|Y) \leq \mathcal{H}_2(P_e) + P_e \log(|X| - 1), \tag{1.19}$$

where $P_e = Pr(\hat{X} \neq X)$ is the probability of error and $\mathcal{H}_2$ is the binary entropy, $|X|$ is the cardinal of X. A less tight bound gives a direct lower bound of P_e:

$$\mathcal{H}(X|Y) \leq 1 + P_e \log(|X|), \tag{1.20}$$

gives

$$P_e \geq \frac{\mathcal{H}(X|Y) - 1}{\log(|X|)}. \tag{1.21}$$

Remark: The above mentioned statistics are given for the true distributions of probability. These statistics are not known for real samples where the only accessible distribution is the empirical distribution computed over the samples, whether the whole dataset, the training subset or a batch of samples. Therefore the formula above mentioned will be used to derive variational bounds.

1.5 Anonymization and privacy

Privacy is one of the most challenging issue in the era of information. The concern appeared because of tools that allowed data collect on a large scale, mainly internet, and large data manipulation, mainly modern computer. Social network for example provide a tremendous amount of data. In the context of machine learning and more widely in the context of collecting data into database, everyone should have in mind this issue. Web scraping using bots to collect online accessible data is the core buisness of some company from start-ups to large firms, and the most valuable data are often private data. It is also the case for companies which business relies on collecting their customer data were always aware of this topic and now at least everyone operating in Europe addresses this particular topic. Especially since the GDPR[1] (The General Data Protection Regulation (EU) 2016/679). There is different kind of privacy adapted to the different types of database one can build.

1.5.1 Related work for the supervised anonymization.

The literature in statistics and computer science on anonymization and privacy is extensive; we discuss only directly relevant work here (for more general references see the survey from [Chen et al., 2009] and references therein).

A definition of anonymization was given by European authority[2] in a document[3] : anonymisation results from processing personal data in order to irreversibly prevent identification.
In other terms, anonymization is a data processing technique that modifies parts of samples to prevent any possible linking between one database sample and one individual. This process should be done in a manner adapted to the type of data. Table database where each entry is a unique user and some features are characterized by text, other by number will not be anonymized the same way as images or sound samples.

Several methods are used to anonymize the data in text database whose main purposes are: generalizing the data, or adding noise to the data. For example Google is

[1] The GDPR is a regulation in EU law on data protection and privacy for all individual citizens of the European Union and the European Economic Area. It also addresses the transfer of personal data outside the European Union and European Economic Area. The GDPR aims primarily to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the European Union.

[2] Article 29 Data protection working party (Art. 29 WP) was an advisory body focused on data protection replaced by European Data Protection Board (EDPB) after the GDPR enactement

[3] Opinion 05/2014 on Anonymisation Techniques, available at the following link: https://cnpd.public.lu/dam-assets/fr/publications/groupe-art29/wp216_en.pdf

concerned with this topic and they explain online[4] these two methods and why they apply them on databases to anonymize them:

Generalizing the data: some features of samples may link directly to a particular individual (for example home address). To prevent any tracking, a well known method is to link one sample with other close samples to form an indistinguishable group regarding this feature (for example change street name to the same street name for every sample that shares a close address). This method is called generalization and allows to achieve k-anonymity. k-anonymity means that for every sample there is at least $k-1$ other samples with the same features. Each of these samples are merged together in the same group of k samples regarding one or several features, depending on the constrains. The k-anonymity introduced by [Sweeney, 2002] is now described by Google as an industry-standard term.

The opposite case is if everyone in the database shares a common feature. In this case, learning that someone's data are in the database definitely informs that this individual has this specific feature. To prevent this privacy risk, one solution is to enforce some diversity: samples with a different feature for this database entry are added to provide diversity. This method is called the l-diversity and was introduced by [Machanavajjhala et al., 2006]. It is also described as an industry-standard term.

Adding noise to the data: A method that takes advantage of noise addition and which is used by Google is called differential privacy (DP), a popular approach introduced in [Dwork, 2006] which is formally introduced in the definition below: // Let $\mathcal{A} : \mathcal{D}^n \to \mathcal{V}$ be a randomized algorithm. Let $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{D}^n$ be two databases that differ in at most one entry (we call these databases neighbors).

Definition *Let $\epsilon > 0$. Define $\mathcal{A}$ to be* ϵ-differentially private *if for all neighboring databases $\mathcal{D}_1, \mathcal{D}_2$, and for all subsets $V \subset \mathcal{V}$, we have*

$$\frac{Pr[\mathcal{A}(\mathcal{D}_1) \in V]}{Pr[\mathcal{A}(\mathcal{D}_2) \in V]} \leq \exp \epsilon, \tag{1.22}$$

where the probability is taken over the con tosses of $\mathcal{A}$. The notion of (DP) has been largely studied in the literature (see survey [Dwork, 2008]). It offers provable privacy in a specific context. In the case of a query mechanism on a database containing information over various people, the query, even if it requests information on some global statistic, could reveal information about a specific user of the database. The idea of differential privacy is to enforce a query mechanism to provide indistinguishable answer while used on two databases differing only by one individual. This kind of databases is called adjacent databases: two databases are adjacent if they differ in a single entry, that is, if one image-label pair is present in one set and absent in the other. Intuitively, it uses random noise to ensure that the mechanism outputting the information about the underlying dataset is robust to any change of one individual, thus protecting privacy. It is then impossible to retrieve any information on a specific person, including even the possibility of telling whether or not they are in the database.

From a statistical perspective, convergence rates of minimax risk for problems in which the data must be kept confidential even from the learner have been reported in [Smith, 2008] and [Duchi, Jordan, and Wainwright, 2014]. DP has been widely studied in the context of the machine learning literature: [Wasserman and Zhou, 2010] and [Chaudhuri, Monteleoni, and Sarwate, 2011] develop differentially private empirical risk minimization algorithms, and [Bassily, Smith, and Thakurta, 2014] and [Wang, Lei, and Fienberg, 2016] study similar statistical and sample complexity of differentially private procedures. [Chen and Zhong, 2009] and [Yuan and Yu, 2014] presented a privacy-preserving distributed backpropagation algorithm which allows a neural network to be trained without requiring either party to reveal her data to the other. [Abadi et al., 2016] studied the feasibility at a small cost of deep neural networks algorithm that complies with differential privacy through a special modification of the stochastic gradient descent (SGD).

[4] https://policies.google.com/technologies/anonymization?hl=en

[Hamm, 2015] studied the case of the distributed learning of different privacy preserving filters coordinated with a data aggregator. The experiments involved at least one binary task (either the classical task or the private task if not both). His measure of privacy risk differs from our and use a specific weighting. Our framework was developed using a single encoder (not distributed). Another difference is that our framework handles simultaneous M-ary tasks. [Edwards and Storkey, 2015] focused mainly on fairness issues. All of the previously presented methods come with a cost: the anonymization impacts the future usage of the database and makes it less useful. The anonymization methods introduced in this manuscript can also make the data less useful.

The classical method of anonymization of images is blurring the part where private information lies. However, the fundamental differences rely on our statistical treatment of the anonymization problem and instead of having only one version of each attribute (or label), we require multiple statistical versions of the same attribute for each individual. Additionally, we focus on databases where identification can be learned and not database containing direct identification data (name, zipcode) which is the case of databases used with k-anonymity methods. They contain data that clearly identifies a person. Instead, we look for data transformations which discard identifying sometimes constitutive features from the data.
[Belghazi et al., 2018] present a neural network based estimator of the mutual information whereas our work only focuses on using an upper bound of the mutual information to ease the optimization of the trade-off we want to achieve. Our framework results are less complex while providing a satisfying bound. The work of [Yang et al., 2018] which is recent too, focuses on the anonymization of click stream that occurs during the visualization of a MOOC's video by students. The task they want to prevent is the identification of a student from its 'click profile' while at the same time preserving the utility task: inferring from the clicks the results of the student to the MOOC quizz. The common point with our work we can exhibit is the usage of a really similar architecture in a Y-shape but the internal layers differ because they use LSTM neurons in their network (Long short-term memory is a type of neuron that have a memory of its previous state meaning it is a recurrent neural network fitted for analysis of sequence data, video file for example). They don't mention precisely the value of the baseline on their anonymity task and only talk about the relative variation on the anonymity.

A major challenge in addressing privacy guarantees is to determine and control the balance between statistical efficiency and the level of privacy, which requires itself a careful mathematical but also meaningful definition. Typically, these techniques depend on how the data are released and the literature contains various approaches to this vast problem.

1.5.2 Semi-supervised anonymization

The work presented by [Chen, Konrad, and Ishwar, 2018] is quite close to our semi-supervized work of the following chapter 4. They propose a tool that allows to perform some emotion preservation while transferring identities using a variational generative adversarial network. This work uses identity and emotion labels, to produce representations which are the same nature as the samples. Furthermore this framework uses one identity as a template to provide anonymity for everyone in the database. Our supervised framework does not produce representation similar to the samples, and therefore does not need a template for the anonymized face. In our case the model itself decides the template to use. If we were to compare our semi-supervised work to this, the main difference would be that we do not ever use the emotion label to produce our generated samples which makes our framework more general. Note that depending on the case, we might in the semi-supervised case use one identity of database as a template, and we can decide, depending on the method, which identity should be used as the template.

In the work of [Edwards and Storkey, 2015] the anonymization task is the removal of printed text label from face pictures (i.e. remove a tag embedded in the image) using adversarial networks. This is somewhat different (in terms of objective as in terms of tools) from our target: anonymize faces pictures themselves. On another side, we differ significantly from disentanglement learning because our representations do not consist in separated variable, they rather remove a chosen variable from the representation while preserving the other.

The work of [Raval, Machanavajjhala, and Cox, 2017] focuses on protecting sensitive information in picture in an adversarial manner. Similar to the work of [Edwards and Storkey, 2015] the private information here is a QR code embedded in the image. The sanitized version of the data should remove the QR code in such a way that a discriminator cannot tell whether or not a QR code was present in the initial image. A second constraint is that in the case of initial images with QR code an adversary network should not be able to locate the position of the QR code on the initial image once it is sanitized. This work uses adversarial networks too, but is quite different from our goal because we do not want to hide some regions of the image (regions used here for its spatial definition). We focus on hiding inherent private data.

More recently the work of [Pittaluga, Koppal, and Chakrabarti, 2018] focuses on providing privacy to picture of different places on four categories through an encoding network trained with a discriminator. The results are assessed based on the capacity of a single binary discriminator which seems not as strong as our method of assessment based on a multiple output classifier. The produced encoded images are anonymized in a really strong manner that removes any useful information most of the time.

The work of [Zhu et al., 2017] is not initially applied to anonymization but is a famous paper about domain adaptation and transferring image from one domain to another without matching pair samples from the two domains. They present a method based on adversarial network to shift images from one domain to the other with a pair of GAN. Their visual results are quite neat. Our method also use adversarial network to produce neat results but it differentiates from this work because instead of translating images into many domains, we focus on providing a single 'identity domain' representation where we make as hard as possible to track the initial identity.

The work of [Moyer et al., 2018] is about learning invariant representations without adversarial training. It uses a mutual information derived loss and a VAE network. Their mutual information bound is based on a divergence as in traditional VAE and they choose to model the latent space empirical distribution through a gaussian mixture to compute its divergence. The model is tested on database where the data they want to hide is binary for supervised invariance.

The work of [Liu, Breuel, and Kautz, 2017] is also about image to image translation. Their work is not oriented toward anonymization. They focus on learning a joint distribution of images from different domains using a shared space between the two domains and two GANs and two VAEs. Therefore one of their trained model can translate images from one domain into another domain. They do not test their method on several domains at the same time as we are doing while anonymizing several identities with the same model.

Another keyword which sound similar to but is different from anonymization is de-identification. Definitions given by [Ribaric, Ariyaeeinia, and Pavesic, 2016] explain clearly the difference between the two. De-identification refers to the reversible process of removing or obscuring any personally identifiable information from individual records in a way that minimizes the risk of unintended disclosure of the identity of individuals and information about them. It involves the provision of additional information to enable the extraction of the original identifiers by, for instance, an authorized body. Anonymization refers to the process of data de-identification that produces data where individual records cannot be linked back to an original as they do not include the required translation variables to do so. In other words, if you recall the earlier definition of anonymization given by european authority: anonymization irreversibly

prevents identification. If the data processing is done in a way to prevent any direct identification, but indirect identification is still possible (using additional data not contained in the database for example), then this process is called De-identification. The main difference between this concept and data anonymization is that some identifying features can be preserved, not leading to a direct re-identification, but in order to be relinked only by a trusted party or by the original data operator, whereas in the case of anonymization, no re-identification should be possible by anyone.

We introduce here the two de-identification papers most related to our work. The first paper is the work of [Meden et al., 2017]. They focus on faces de-identification with generative deep neural network. They present an extensive complex model to perform feature extraction from the images, then compare the feature to the k-closest faces and generate a new image combining the k-closest images. The model they used is called a pipeline and produces good results but the corresponding architecture is really different from our because of its complexity. Any comparison with our method is difficult, because, even if they mention the goal of preserving useful non-private information they never perform any test related to it.
The second paper is the work of [Li and Lyu, 2019]. It focuses on de-identification while keeping as much as possible of the non-private attributes which is exactly the scope of our work. The process is to transfer facial attribute from a subject to the face of a 'donor' whose identity will be released on purpose. The method they used is quite complex and uses different successive processing: face detector that locates and bounds in a box the face, a facial landmark extraction algorithm detects landmark face points (tips of eyes, eyebrows, nose, mouth ad contour). The landmark are then aligned with a standard frontal oriented face. This allows to transform the extracted face to a frontal oriented version of the face. They afterwards use a facial attribute transfer model that transposes the extracted face to the produce a donor version of it (This transfer model is based on [Liu, Breuel, and Kautz, 2017]). This version with the donor identity is then reoriented to fit the original picture with a lot more complication to blend it nicely. Their method produces neat results but the test they have done to check for anonymization gives disappointing results and they are puzzled about it.
The process explained in the above mentioned paper is quite close to methods used to produce deepfake video content. Deepfake videos are produced using regular video footage and a collection of a person's face pictures. The facial features of the person depicted in the video are changed into the one provided by the pictures. Face features are transferred frame by frame. The processed is assessed using the discriminator of as in the GAN framework; the discriminator network is used to enhance the resulting images into lifelike images. Images are then assembled into a video. The results are really neat and can full human eyes in some cases (see youtube deepfake video of celebrities). Nonetheless, deepfake videos can be distinguished from genuine videos as presented in the work of [Korshunov and Marcel, 2018] and [Li and Lyu, 2018]. Note that deepfake is a term that is also applied to forged sound files: the specific person's voice samples are used to produce record of sentences that were never tell by the person.

Remark: The state of the art of the chapter 5 is actually at the beginning of the above mentioned chapter because its notation is intrinsic with the presentation of the problem.

1.6 Contributions

Statistical methods protecting sensitive information or the identity of the data owner have become critical to ensure privacy of individuals as well as of organizations. This work investigates statistical anonymization methods.
Another side of my work is the study of predictor outputs to extract relevant information about the predictor's behavior.

1.6.1 Supervised anonymization (Chapter 3)

In the chapter regarding the supervised framework (private and regular labels are known) the contributions are mainly the introduction of a new loss that is designed to provide representation under a trade-off constraint: preserve as much as possible the useful information regarding a target task while anonymizing as much as possible the representation. The associated architecture was designed in order to learn a model that can process the samples into the representation. Additionally a training procedure was devised in order to train this complex architecture model. The combination of the loss with the architecture mains characteristics are: ***i*** the adversarial network tries to classify among a large number of person's identities (instead of among two domains), and ***ii*** the training objective is designed to lead to more robust training, avoiding the numerical difficulties that would be caused by maximizing unbounded cross-entropy (between the representation and the private label). This maximization would occur if a vanilla method is used to maximize cross-entropy in order to anonymize the private features[5].

1.6.2 Semi-supervised anonymization(Chapter 4)

In the chapter regarding the semi-supervised framework, the contribution is the adaptation of the previously mentioned loss. This leads to the design of a new architecture due both to the lack of regular task labels and to the change in the nature of the anonymized representation (from a simple feature vector to a coherent picture). A training procedure was also derived to fit the new framework. This new model was used on the same images dataset and provided interesting results in the context of anonymization without a target task: some quantitative and other qualitative. Indeed the notion of intertwined database (intertwined meaning that the targeted private variable features are strongly connected to the other constitutive features of the image) was a unsuspected factor that laid two main variations each being of a related yet distinct interest.
The first proposal focused on processing separable database, separable meaning that the targeted private variable features are not deeply connected with other features. This resulted on good results on the not-intertwined dataset and poor results on the intertwined dataset.
This is why the second proposal was designed to tackle the case of producing significantly better visual results for this type of dataset. Adding geometric consideration, this second variation manages to perform pertinent face substitution therefore providing stronger results on a computer tested anonymity while keeping as much as possible other features.

1.6.3 Predictor monitoring (Chapter 5)

In the last chapter we focused on monitoring a 'black box' predictor outputs behavior in order to detect possible discrepancy between test samples and training samples.
Two methods are used to detect change of behavior of the classifier statistic in the context of batches data. The first method is a coding theory induced idea to assess the behavior of the predictor with the help of Pearson's chi-squared test.
The second method compares different tools performances. These tools are motivated with mathematical considerations. The performances of these tools, despite being affected by the type and nature of the database are conclusive even for small sized batches.

[5] Vanilla method is for example applying straight domain adaptation methods to anonymize representation.

Chapter 2

Datasets presentation

This chapter presents the various datasets used in the following chapters. They are first introduced and the less famous ones are also illustrated with samples. For preprocessed database, the method of preprocessing is precisely described and illustrated by samples before and after preprocessing.

The databases chosen for the anonymization task are the ones that are easily exploitable in this context. We wanted primarily picture databases in order to focus on the core of the problem of anonymization. We avoided other forms of data. The ones that rely on the meaning of words and their relation with each other are difficult to represent in numerical values. The same goes for databases that involved qualitative words as one or several attributes. The use of embeddings that maps words to vectors is a possible solution to this problem, however the design of such embeddings adds significant complexity to the initial task. To avoid any interference between the performance of such a pre-processing and the performance of our framework we limited ourselves to the use of numerical value database such as digital data (pictures here). Traditional datasets such as list of features or tabular sometimes used in privacy articles are not used here. Databases composed of digits attributes or composed of unequivocal digits translatable attributes would then have to meet the requirement of our framework: Having at least several samples for each pair of labels (the pair of labels stands for both the private task label and the regular task label). Table datasets that are used in privacy learning are often composed of a unique sample per pair of labels and even a unique sample per private label. This is not the kind of problem our framework is build to tackle.

2.1 MNIST

MNIST (Modified National Institute of Standards and Technology database) database is for sure one of the most famous datasets used in several machine learning fields such as handwriting recognition.

The MNIST database is composed of handwritten digits and is derived from NIST database and introduced initially by [Lecun et al., 1998] (today's reference is [LeCun and Cortes, 2010]). This database consists in black digits from 0 to 9 on a white background. These images have been normalized with respect to their size and centered to fit into a 28-by-28 pixels and anti-aliased which introduce shades of grey in the image. The database is split into two sets: a 60 000 samples training set and a 10 000 samples test set. Note that each set is matched by the corresponding label file.

2.2 Fashion-MNIST

The Fashion-MNIST database introduced by [Xiao, Rasul, and Vollgraf, 2017] is an MNIST-like database, meaning it is similar to the above presented MNIST database: centered object on a white background in a square image. This database has been made by Zalando research team and consists in 28-by-28 pixels grey scale images of clothes. The ten classes of this database are: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. The size of the sets is the same as MNIST: 60 000

images in the training set and 10 000 images in the test set with the corresponding label file.

2.3 CIFAR-10

CIFAR-10 (Canadian Institute For Advanced Research) database was introduced by [Krizhevsky, 2009]. This database is composed of small square (32-by-32 pixels) color images each picturing an object or a living creature in their usual background environment. There are exactly 10 classes in this database (CIFAR-10): airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The train set and the test set consist of respectively 50 000 and 10 000 images. Note that the classes are mutually exclusive, meaning there is no overlap: one picture displays a subject that belongs to one and only one class.

2.4 SVHN

SVHN (Street View House Number) database was introduced by [Netzer et al., 2011]. Note that the format 2 has been used for this work. This database is composed of house number pictures extracted from the street view's feature of Google Maps. The format 2 means that the picture have been pre-processed in the following manner: digits have been cropped and resized to a 32-by-32 pixels resolution color image. This cropping and resizing is done without any distortion. The fact that most of the time house numbers are made of *numbers* and not digits means that this cropped digit images may have some residuals parts of other digits on sides (most of the time left and right side, but with variability due to numbers being tilted due to numerous reasons).

2.5 Pen-digits

2.5.1 Presentation

We selected a convenient enough dataset, named Pen-digits from Alpaydin [Alimoglu and Alpaydin, 1996]. This dataset is interesting to study anonymization because it has double labels (the user IDs of writers and the digit categories) and it has many examples of each writer. The dataset provides the coordinates of digitally acquired pen movements of 44 persons (30 are involved in the training set and 14 in the test-set) writing digits from 0 to 9. We only used the training set (to use only 30 identities) which was randomly split into training, validation and test data sets (size 5494, 1000 and 1000, respectively), sharing images of the same 30 persons. At the time of collecting this dataset, inconclusive digits were removed. This dataset contains 25 times each digits for each person minus the few discarded digits. The raw data is a set of pen trajectories. It is preprocessed in several steps. The coordinates of all the curves corresponding to a single sample were normalized in order to center the image and reduce variability by making it fit a 80x80 image. After being drawn, each image was then down-sampled into a 20x20 image. Resulting images were transformed into grey scale images with a black background.

2.5.2 Samples

A table overview of the samples is shown Figure 2.1: digits in the same lines come from the same writer. Variability between written samples is exposed on Figure 2.2 for the digit 9.

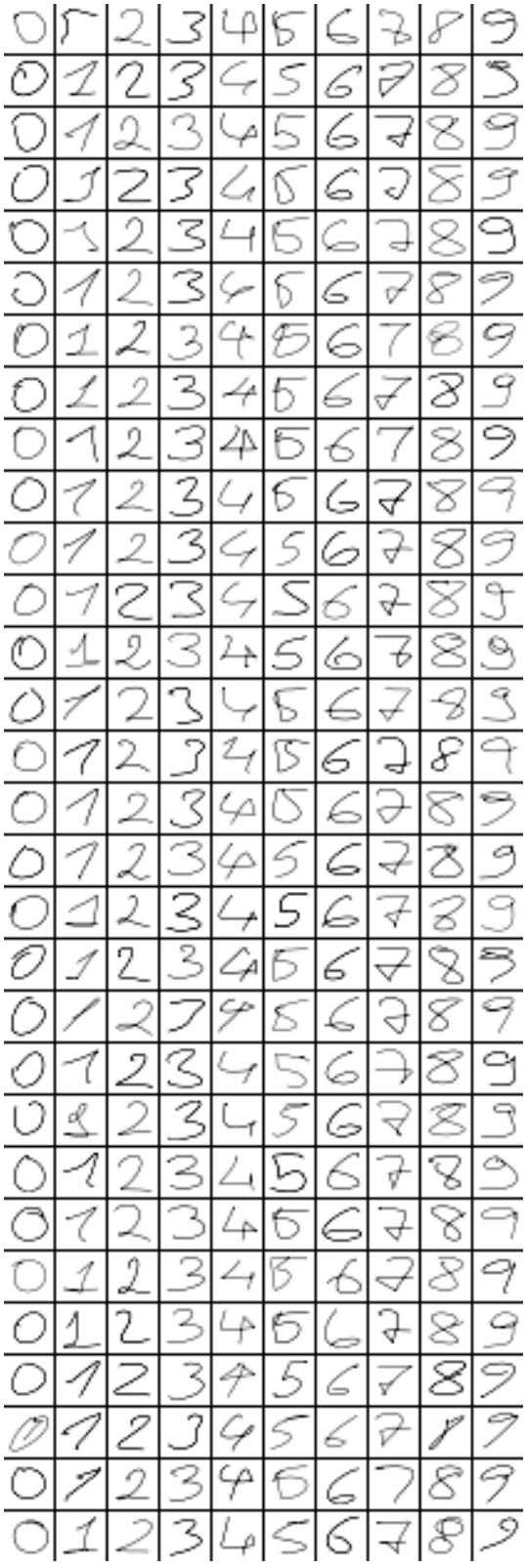


FIGURE 2.1: Samples from the Pen-digits database. Each line corresponds to one writer and shows samples of different digits from the training set.

2.6 JAFFE

2.6.1 Presentation

The JAFFE (Japanese Female Facial Expression) database [Lyons et al., 1998] and [Dailey et al., 2010] contains 213 pictures of Japanese women's faces composed of 10 different persons as shown Figure 2.3. Each person has between 2 and 4 pictures per facial

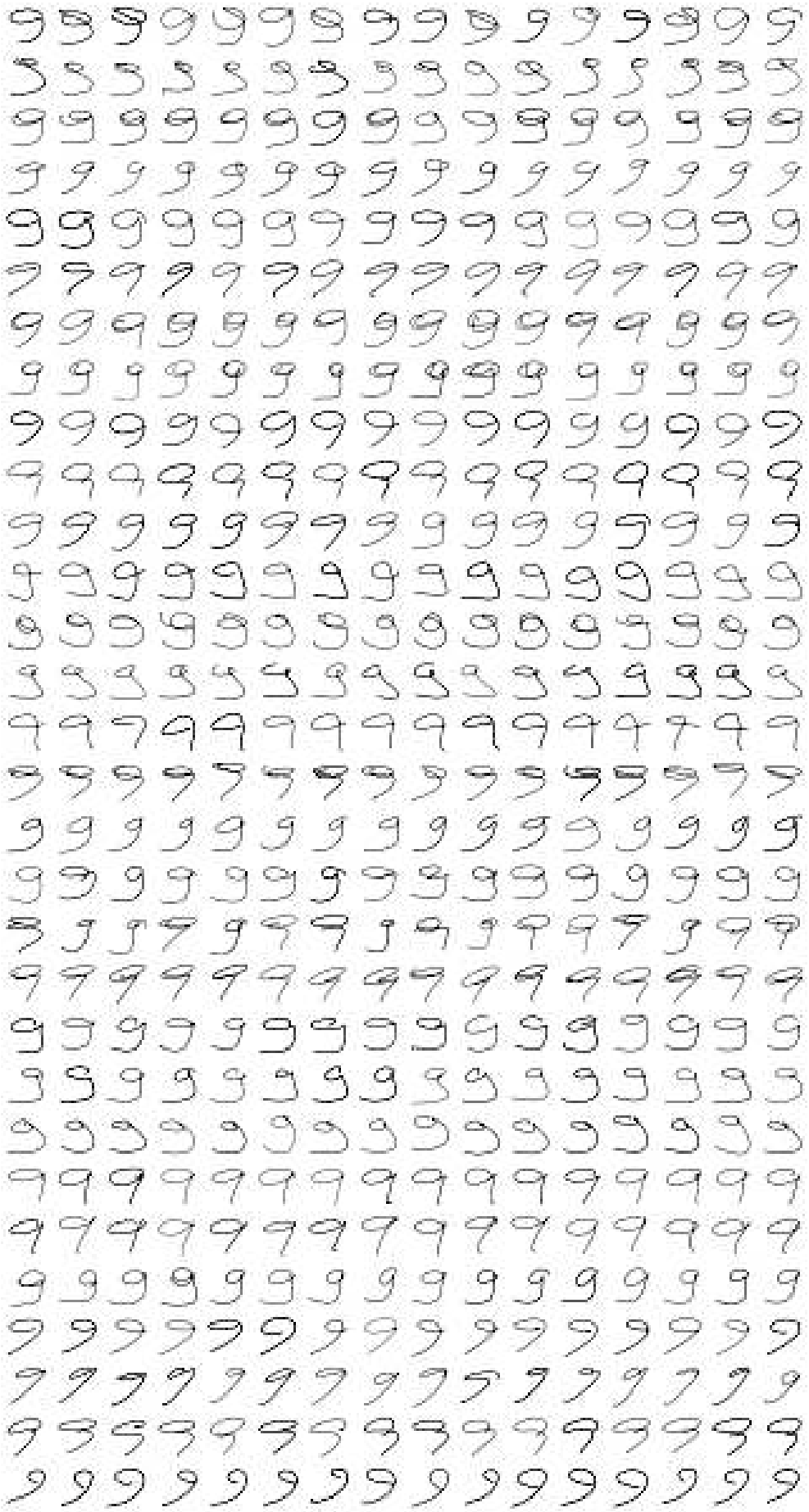


FIGURE 2.2: Samples from the Pen-digits database. Each line shows the same writer variability of the 9s. Samples are from the training set.

expressions. The seven facial expressions are associated with the following feelings: “neutral”, “anger”, “fear”, “surprise”, “sadness”, “joy” and “disgust”. The pictures were processed to remove irrelevant background pixels. Pictures have been cut in order to have the bottom of the chin as the bottom pixels line, the frontier between hair and forehead as the top pixels line, the frontier between hair and temple as the far right and far left pixels columns. To do so the cumulative brigthness (either vertically or horizontally) variation are plotted and minimum value among a specified window are chosen as frontier, as shown on the Figure 2.4. The remaining pixels in the corner that does not belong to the face were set to black. The original pictures are 256x256 pixels and the resulting images are 29x37 pixels and they are shown on Figure 2.5. The dataset is divided into a 139 pictures training set and a 74 pictures test set. There are barely enough data to perform the training properly so the training set is used as the validation set as well. This decision may be considered as fallacious but a validation set is needed because several steps of the algorithm are optimized with the loss value or the accuracy value on the validation set.

2.6.2 Samples

Figure 2.3 shows the unprocessed images, Figure 2.4 shows the preprocessing and Figure 2.5 shows the processed images.

2.7 FERG

2.7.1 Presentation

The FERG (Facial Expression Research Group) database [Aneja et al., 2016] contains 55767 annotated face synthetic images of six stylized characters modeled using the MAYA software (a 3D computer animation, modelling, simulation and rendering software). This database has 256x256 pixels images depicting the seven following facial expressions (or feelings): “neutral”, “anger”, “fear”, “surprise”, “sadness”, “joy” and “disgust”. The main advantage of this database is the number of samples which is huge. It contains a double labeling and the synthetic face have human-like emotion expressions. For each expression and character, there are between 911 and 2088 images. Original 256x256 pixels colour images have been pre-processed into re-sized 8-bit grey-scale 50x50 pixels images.

2.7.2 Samples

Figures 2.6 and 2.7 display samples from the database respectively before and after the pre-processing.

FIGURE 2.3: JAFFE database initial samples without any pre-processing

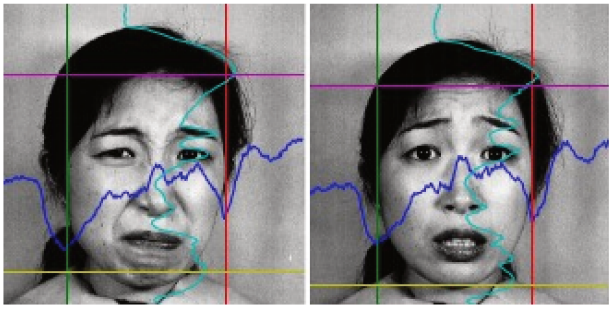


FIGURE 2.4: Pre-processing JAFFE samples. The navy blue curve shows the evolution of the vertical axis cumulative brightness. The azure curve shows the evolution of the horizontal axis cumulative brightness. Using the mimimum of these in a selected region allows to find the subject face frontier (purple, red, yellow and green lines)

FIGURE 2.5: Jaffe database samples after pre-processing

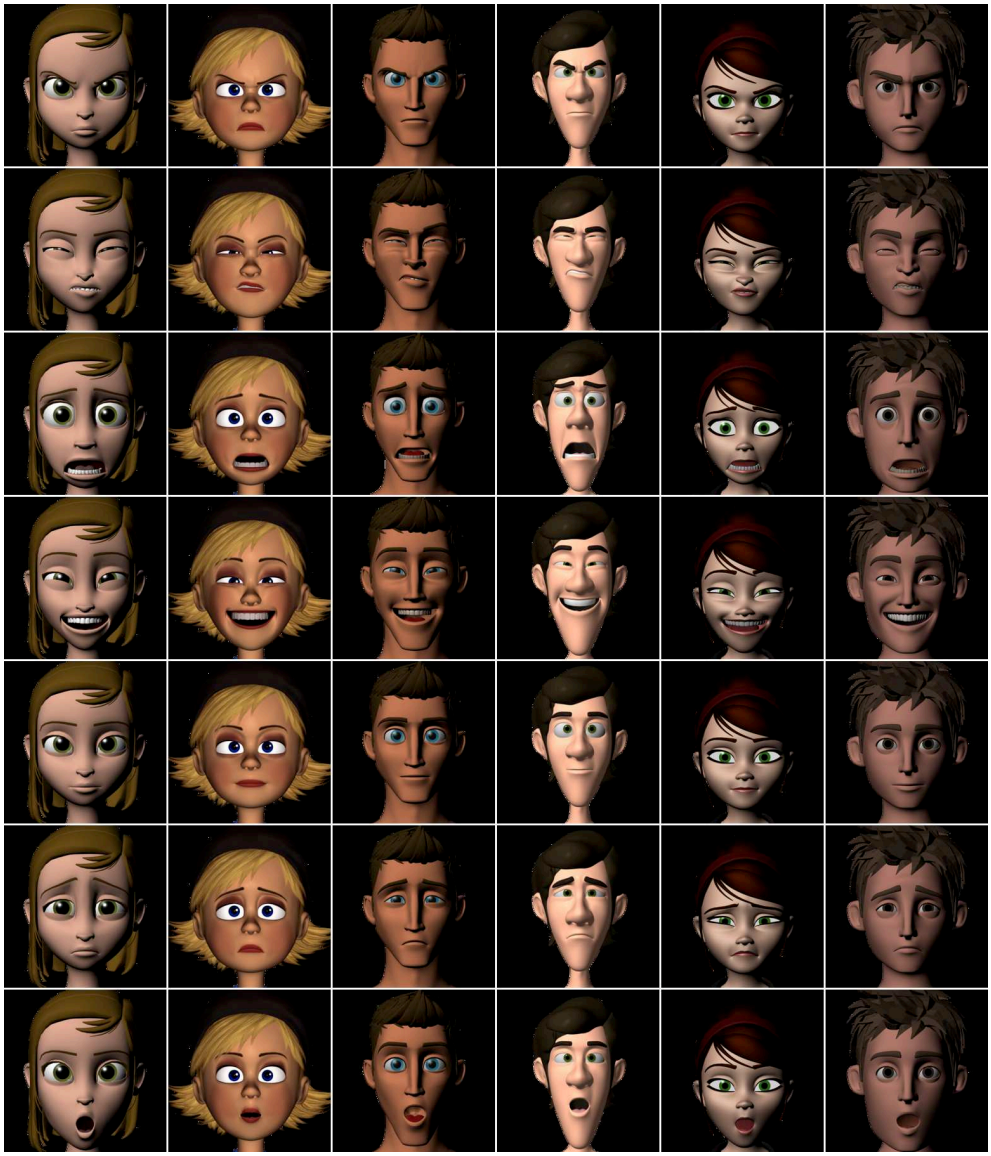


FIGURE 2.6: Initial samples from the FERG database.

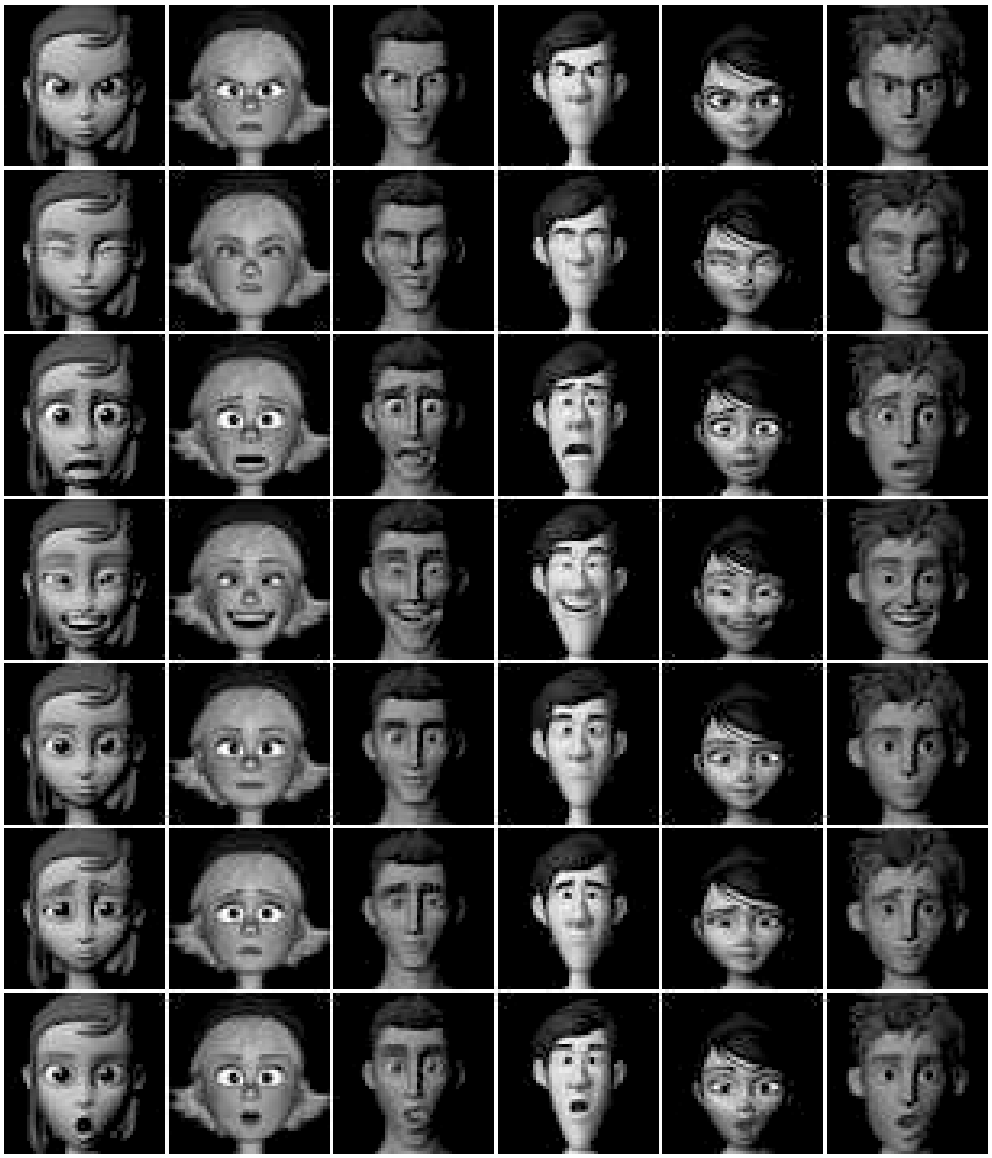


FIGURE 2.7: Pre-processed samples from the FERG database.

Chapter 3

Supervised anonymization

3.1 Introduction

This chapter addresses the precise situation where some supervised classification task has to be performed on a dataset, while keeping another information (which could also be obtained via some classifier) secret. The most simple example would be to keep secret the identity of the writer in the pen-digits database while still allowing efficient recognition of the handwritten numbers. This obviously requires to have doubly labeled data sets for training. In the next chapter, we will address the situation where we do not know exactly which kind of classification (processing) the regular task will be, but still want to maintain some information hidden.

We investigate anonymization from a perspective which is related but different from that of differential privacy. The main difference relies on the condition of the information release (sanitize) mechanism which in our case depends on the dataset itself. Additionally, differential privacy introduces randomized predictors whereas our method (after training is accomplished) induces a deterministic algorithm. As stated above, our intent is to hide information about the private labels which is implicitly present in a dataset while preserving as much information as possible about the regular relevant labels. For this purpose, we introduce a novel adversarial training objective and framework inspired by Generative Adversarial Networks (GAN) by [Goodfellow et al., 2014] and by the domain adaptation framework of [Ganin and Lempitsky, 2015].

We propose an efficient way of optimizing an information-theoretic objective by deriving backpropagation signals through a competitive process involving three networks, illustrated in Figure 3.1 for the supervised case, an encoder network which is a common trunk mapping input X to a representation U, as well as two branch networks taking U as input, i.e. a predictor for the regular labels Y and a predictor for the private labels Z. While the encoder is trained to help the predictor of Y as much as possible, it is also trained to prevent the Z predictor from extracting private information from U, leading to a trade-off between these two objectives.

3.2 Theory

We introduce our model from which sanitized representations will be learned. We develop a precise formalization of the problem and derive an information-theoretic criterion that together GAN provides a tractable supervised objective to guide the learning of constrained representations.

3.2.1 Learning model and problem definition

In this work, we are concerned with the problem of pattern classification which is about predicting the regular label (public information) of an observation based on high-dimensional representations. An observation is a sample $x \in \mathcal{X}$ presented to the learner about a target concept $y \in \mathcal{Y}$ (the regular label) and the user ID $z \in \mathcal{Z}$ (the private label). This consists of a typical supervised learning setup with a training dataset of n i.i.d. tuples: $\mathcal{T}_n := \{(x_1, y_1, z_1) \cdots (x_n, y_n, z_n)\}$, sampled according to an

unknown distribution p_{XYZ}. We consider the problem of learning a representation from examples of p_{XYZ}. We would like to find a (possibly stochastic) transformation $q_{U|X}$ that maps raw data X to a higher-dimensional (feature) space $\mathcal{U}$:

$$p_{YZ} \sim (Y,Z) \xrightarrow[\text{(unknown)}]{p_{X|YZ}} X \xrightarrow[\text{(encoder/sanitize)}]{q_{U|X}} U.$$

This representation $\mathcal{U}$ should be designed in such a way that it allows to simultaneously find a (randomized) deep encoder $q_{U|X} : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ and a soft-classifier $q_{\hat{Y}|U} : \mathcal{U} \to \mathcal{P}(\mathcal{Y})$ which maps the representation to a distribution on the label space $\mathcal{Y}$. Hence, our ultimate goal is to learn $q_{U|X}$ from a deep neural network to perform this classification task while preventing any classifier $q_{\hat{Z}|U} : \mathcal{U} \to \mathcal{P}(\mathcal{Z})$ from learning the private label Z from representation U. In other words, our representation model must learn invariant features with respect to private labels. We will formalize this problem as being equivalent to that of optimizing a trade-off between both misclassification probabilities (regular and private label). It is therefore necessary to define this notion:

Definition 1. *The probability of misclassification of the induced decision rule from an encoder $q_{U|X}$ and a classifier $q_{\hat{Y}|U}$ with respect to the distribution p_{XY} is given by*

$$P_e\big(q_{U|X}, q_{\hat{Y}|U}\big) \coloneqq 1 - \mathbb{E}_{p_{XY}q_{U|X}}\Big[q_{\hat{Y}|U}(Y|U)\Big]. \tag{3.1}$$

We can now provide an operational definition of what would make a good representation U in the anonymization problem. A representation should be useful for minimizing the misclassification probability of the public task of interest with regular labels Y while bounding from below, whatever classifier $q_{\hat{Z}|U}$ is chosen, the probability of misclassification of the identity Z, which is formally introduced in the next subsection.

3.2.2 Bounds on the probability of misclassification

Definition 2 (Learning with anonymization constraints)**.** *Consider the following constrained pattern classification problem:*

$$\min_{(q_{U|X}, q_{\hat{Y}|U}) \in \mathcal{F}} \Big\{ P_e\big(q_{U|X}, q_{\hat{Y}|U}\big) \; : \; \min_{q_{\hat{Z}|U}: \mathcal{U} \to \mathcal{P}(\mathcal{Z})} P_e\big(q_{U|X}, q_{\hat{Z}|U}\big) \quad \geq 1 - \varepsilon \Big\}, \tag{3.2}$$

for a prescribed probability $1/|\mathcal{Z}| \leq \varepsilon < 1$, where the minimization is over the set of restricted encoders and classifiers $(q_{U|X}, q_{\hat{Y}|U}) \in \mathcal{F}$ according to a model class $\mathcal{F}$.

The above expression requires representations with $(1-\varepsilon)$-approximate guarantees (over all possible classifiers) w.r.t. the misclassification probability of the private labels. ε can be replaced by a suitable positive multiplier $\lambda \equiv \lambda(\varepsilon)$ yielding a relaxed version of the objective.

$$\min \Big\{ P_e\big(q_{U|X}, q_{\hat{Y}|U}\big) - \lambda \cdot P_e\big(q_{U|X}, q^{\star}_{\hat{Z}|U}\big) \Big\}, \tag{3.3}$$

where $q^{\star}_{\hat{Z}|U}$ is the minimizer of $P_e\big(q_{U|X}, q_{\hat{Z}|U}\big)$. Expression (3.3) does not lead to a tractable objective for training $(q_{U|X}, q_{\hat{Y}|U})$. However, it suggests a competitive game between two players: an adversary trying to infer the private labels Z from our representations U, by minimizing $P_e\big(q_{U|X}, q_{\hat{Z}|U}\big)$ over all possible $q_{\hat{Z}|U}$ over a prescribed model class $\mathcal{F}$, and a legitimate learner predicting the regular labels Y, by optimizing a classifier $q_{\hat{Y}|U}$ over a prescribed model class $\mathcal{F}$. We can trade-off these two quantities via the representation (encoder) model $q_{U|X}$. This key idea will be further developed in the next section through an adversarial framework to guide learning of all involved parameters in the class $\mathcal{F}$.

In order to derive a tractable surrogate to (3.2), it is convenient to first introduce the rate-distortion function [Cover and Thomas, 2006].

Definition 3. *The rate-distortion function of a RV $Z \in \mathcal{Z}$ with distortion $d(z,u) := 1 - q_{\hat{Z}|U}(z|u)$ is defined as:*

$$\mathcal{R}_{Z,q_{\hat{Z}|U}}(D) := \min_{\substack{p_{\hat{U}|Z}:\mathcal{Z}\to\mathcal{P}(\mathcal{U}) \\ \mathbb{E}_{p_{\hat{U}Z}}[1-q_{\hat{Z}|U}(Z|U)]\leq D}} \mathcal{I}(Z;U), \tag{3.4}$$

where $p_{\hat{U}Z} = p_{\hat{U}|Z}P_Z$. Furthermore, there exists $D > 0$ s.t. $\mathcal{R}_{Z,q_{\hat{Z}|U}}(D)$ is finite [Csiszár, 1974], let the minimum be $D_{\min}$ with $R_{\max} := \mathcal{R}_{Z,q_{\hat{Z}|U}}(D)$ as $D \to D_{\min}+$.

Moreover, $\mathcal{R}_{Z,q_{\hat{Z}|U}}(D)$ is positive, monotonically decreasing and convex. Let us define:

$$\mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(I) := \inf\left\{D \in \mathbb{R}_{\geq 0} : \mathcal{R}_{Z,q_{\hat{Z}|U}}(D) \leq I\right\}, \tag{3.5}$$

which is known as the *distortion-rate* function [1]. The function $I \mapsto \mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(I)$ is positive and monotonically decreasing.

Another notion that will be necessary to obtain an upper bound of the untractable objective of (3.3) is the *cross-entropy risk* defined below:

Definition 4 (Cross-entropy loss)**.** *Given two distributions $q_{U|X} : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ and $q_{\hat{Y}|U} : \mathcal{U} \to \mathcal{P}(\mathcal{Y})$, define the average (over representations)* cross-entropy loss *as:*

$$\ell(q_{U|X}(\cdot|x), q_{\hat{Y}|U}(y|\cdot)) := \left\langle q_{U|X}(\cdot|x), -\log q_{\hat{Y}|U}(y|\cdot)\right\rangle \tag{3.6}$$

$$= \mathbb{E}_{q_{U|X=x}}\left[-\log q_{\hat{Y}|U}(y|U)\right]. \tag{3.7}$$

As usual, we shall measure the expected performance of $(q_{U|X}, q_{\hat{Y}|U})$ via the risk*:*

$$\mathcal{L}(q_{\hat{Y}|U}, q_{U|X}) := \mathbb{E}_{p_{XY}}\left[\ell(q_{U|X}(\cdot|X), q_{\hat{Y}|U}(Y|\cdot))\right]. \tag{3.8}$$

The following lemma provides bounds on the misclassification probability via mutual information and the cross-entropy loss (proof available in appendix A).

Lemma 5. *The probabilities of misclassification $P_e(q_{U|X}, q_{\hat{Y}|U})$ and $P_e(q_{q_{U|X},\hat{Z}|U})$ induced by an encoder $q_{U|X} : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ and two arbitrary classifiers $q_{\hat{Y}|U} : \mathcal{U} \to \mathcal{P}(\mathcal{Y})$ and $q_{\hat{Z}|U} : \mathcal{U} \to \mathcal{P}(\mathcal{Z})$ are bounded by*

$$P_e(q_{U|X}, q_{\hat{Z}|U}) \geq \mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}\left(\mathcal{I}(Z;U)\right), \tag{3.9}$$

$$P_e(q_{U|X}, q_{\hat{Y}|U}) \leq 1 - \exp\left(-\mathcal{L}(q_{\hat{Y}|U}, q_{U|X})\right), \tag{3.10}$$

where $q_{U|Z}(u|z) = \sum_{x\in\mathcal{X}} q_{U|X}(u|x)p_{X|Z}(x|z)$.

Observe that the lower bound in (3.9) is a monotonically decreasing function of the mutual information $\mathcal{I}(Z;U)$. This implies that any limitation of the mutual information between private labels Z and representations U will bound from below the probability of misclassification of private labels, whatever classifier $q_{\hat{Z}|U}$ is chosen. On the other hand, the upper bound in (3.10) shows that the cross-entropy loss $\mathcal{L}(q_{\hat{Y}|U}, q_{U|X})$ can be used as a surrogate to optimize the misclassification probability of regular labels, which

[1] It is worth to mention that by using $\mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(I)$ we are abusing notation. This is because in general it is not true that $\mathcal{R}_{Z,q_{\hat{Z}|U}}(D)$ is injective for every $D \geq 0$. However, when $I \in [R_{\min}, R_{\max})$ with $R_{\min} := \mathcal{R}_{Z,q_{\hat{Z}|U}}(D_{\max})$ and $D_{\max} := \min_{u\in\mathcal{U}} \mathbb{E}_{P_Z}\left[1 - q_{\hat{Z}|U}(Z|u)\right]$, under some very mild conditions on P_Z and $q_{\hat{Z}|U}(z|u)$, $\mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(I)$ is the true inverse of $\mathcal{R}_{Z,q_{\hat{Z}|U}}(D)$, which is guaranteed to be injective in the interval $D \in (D_{\min}, D_{\max}]$.

motivates the cross-entropy loss. The practical relevance of these information-theoretic bounds is to provide a mathematical objective for browsing the trade-off (3.2) between all feasible misclassification probabilities $P_e(q_{U|X}, q_{\hat{Y}|U})$ as a function of the prescribed $(1-\varepsilon)$ probability. Therefore, the learner's goal is to select an encoder $q_{U|X}$ and a classifier $q_{\hat{Y}|U}$ by minimizing jointly the risk and the mutual information, leading to tightening both bounds in Lemma 5.

However, since p_{XYZ} is unknown the learner cannot directly measure neither the risk in (3.10) nor the mutual information in (3.9). It is common to measure the agreement of a pair of candidates with a training data set based on the empirical data distribution $\hat{p}_{XYZ}$. This yields an information-theoretic objective, being a surrogate of expression of eq. (3.3):

$$\min \left\{ \mathcal{L}_{\text{emp}}(q_{\hat{Y}|U}, q_{U|X}) + \lambda \cdot \widehat{\mathcal{I}}(Z;U) \right\}, \tag{3.11}$$

for a suitable multiplier $\lambda \geq 0$, where $\mathcal{L}_{\text{emp}}(q_{\hat{Y}|U}, q_{U|X})$ denotes the *empirical risk* as in Definition 4 taking the average w.r.t. $\hat{p}_{XY}$ and the mutual information must be evaluated with empirical distribution (reminder $\widehat{\cdot}$ means empirical) using $\hat{q}_{Z|U}$ as being the posterior according to $q_{U|X}\hat{p}_{XZ}$. As a matter of fact, eq. (3.11) may be independently motivated by a rather different problem studying distortion-equivocation trade-offs [Villard and Piantanida, 2013].

3.2.3 Representation learning with anonymization

We performed initial experiments in which the training objective was similar to the one introduced by [Ganin and Lempitsky, 2015] for domain adaptation, and found that training was unstable and led to a poor trade-off between the degree of anonymity (with the classification error on private labels Z as a proxy) and the accuracy on the regular task (predicting regular labels Y). This led us to change both the training objective and the training procedure, compared to those proposed by [Ganin and Lempitsky, 2015]: the training objective in Ganin is directly the difference of the cross-entropy losses with a multiplier in front of the domain one. Our loss written further below in the current subsection is based on different approach and results in an additional term and absolute value in its expression. The training is explained in the algorithm subsection. The new adversarial training objective is presented below, starting from the information-theoretic surrogate presented above in expression (3.11).

A careful examination of expression (3.11) shows that it cannot be optimized since the posterior distribution $\hat{q}_{Z|U}$ is still not computable in high dimensions. We will further loosen this surrogate by upper bounding the empirical mutual information $\widehat{\mathcal{I}}(Z;U) = \widehat{\mathcal{H}}(Z) - \widehat{\mathcal{H}}(Z|U)$. The *empirical entropy* of Z can be upper bounded as follows:

$$\begin{aligned}
\widehat{\mathcal{H}}(Z) &\leq \mathbb{E}_{\hat{p}_Z}\left[-\log \hat{q}_{\hat{Z}}(Z)\right] && (3.12)\\
&\leq \mathbb{E}_{\hat{p}_Z}\mathbb{E}_{\hat{q}_U}\left[-\log q_{\hat{Z}|U}(Z|U)\right] && (3.13)\\
&\equiv \mathbb{E}_{\hat{p}_Z}\mathbb{E}_{\hat{p}_X}\left[\ell\big(q_{U|X}(\cdot|X), q_{\hat{Z}|U}(Z|\cdot)\big)\right] && (3.14)\\
&:= \mathcal{L}^{\text{obj}}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X}), && (3.15)
\end{aligned}$$

where (3.12) follows since the relative entropy is non-negative (cf. section 1.4); (3.13) follows by the convexity of $t \mapsto -\log(t)$ and (3.14) follows from the definition of the cross-entropy loss. We will also resort to an approximation of the conditional entropy $\widehat{\mathcal{H}}(Z|U)$ by learning an adequate empirical cross-entropy risk:

$$\begin{aligned}
&\widehat{\mathcal{H}}(Z|U) \approx \\
&\mathbb{E}_{\hat{p}_{XZ}}\left[\ell\big(q_{U|X}(\cdot|X), q_{\hat{Z}|U}(Z|\cdot)\big)\right] \equiv \mathcal{L}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X}),
\end{aligned} \tag{3.16}$$

which assumes a well-selected classifier $q_{\hat{Z}|U}$, i.e., the resulting approximation error given by $\mathcal{D}(\hat{q}_{Z|U}\|q_{\hat{Z}|U}|\hat{q}_U)$ w.r.t. the exact $q_{\hat{Z}|U}$ is small enough. By combining expressions (3.15) and (3.16), and taking the absolute value, we obtain :

$$\widehat{\mathcal{I}}(Z;U) \precsim \left|\mathcal{L}^{\text{obj}}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X}) - \mathcal{L}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X})\right|, \tag{3.17}$$

that together with eq. (3.11) leads to our tractable objective for learning, which is an approximation of expression (3.11), being the surrogate of (3.3), i.e., the objective of interest:

$$\begin{aligned}\mathcal{L}_\lambda(q_{\hat{Y}|U}, q_{\hat{Z}|U}, q_{U|X}) &:= \mathcal{L}_{\text{emp}}(q_{\hat{Y}|U}, q_{U|X}) \\ &+ \lambda \cdot \left|\mathcal{L}^{\text{obj}}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X}) - \mathcal{L}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X})\right|,\end{aligned} \tag{3.18}$$

for a suitable classifier $q_{\hat{Z}|U}$ and multiplier $\lambda \geq 0$, being a meta-parameter that controls the sensitive trade-off between data anonymity and statistical efficiency. Consequently, we can minimize and maximize the incompatible objectives of the *cross-entropy losses* in (3.18). Intuitively, the data representations we wish to achieve from $q_{U|X}$ must blur the private labels Z from the raw data X while preserving as much as possible relevant information about the regular labels Y. It is worth to mention that (3.15) corresponds to the loss of a 'random guessing' classifier in which the representations U are independent of private labels Z. As a consequence, training encoders $q_{U|X}$ to minimize (3.18) enforces the best classifier $q_{\hat{Z}|U}$ (private labels) to get closer – in terms of loss – to the random guessing classifier.

Note also that since (3.16) is only an approximation, one cannot fully ensure that the right-hand side of (3.17) is really an upper bound of the entropy. The accuracy of the approximation will be checked in the simulation section.

3.2.4 Estimation of the probability of misclassification

The following proposition provides an interesting lower bound on the estimated (e.g. over a choice of test-set) misclassification probability of any classifier attempting to learn Z from the released representations:

Proposition 1. *Let $q_{U|X}$ be a sanitize encoder and $\hat{p}_{XZ}$ be an empirical distribution over a choice of a data-set $\mathcal{T}_n := \{(x_1, z_1) \cdots (x_n, z_n)\}$. Then, the probability of misclassification of private labels satisfies:*

$$\hat{P}_e(q_{U|X}, q_{\hat{Z}|U}) \geq g^{-1}\left(\log|\mathcal{Z}| - \widehat{\mathcal{I}}(Z;U)\right), \tag{3.19}$$

uniformly over the choice of $q_{\hat{Z}|U}$, where for $0 \leq t \leq 1$:

$$g(t) := t \cdot \log\left(|\mathcal{Z}| - 1\right) + H(t), \tag{3.20}$$

with

$$H(t) := -t\log(t) - (1-t)\log(1-t), \tag{3.21}$$

and $0\log 0 := 0$

(3.22)

The function $g^{-1}(t) := 0$ for $t < 0$ and, for $0 < t < \log|\mathcal{Z}|$, $g^{-1}(t)$ is a solution of the equation $g(\varepsilon) = t$ w.r.t. $\varepsilon \in \left[0, 1 - 1/|\mathcal{Z}|\right]$; this solution exists since the function g is continuous and increasing on $\left[0, 1 - 1/|\mathcal{Z}|\right]$ and $g(0) = 0$, $g\left(1 - 1/|\mathcal{Z}|\right) = \log|\mathcal{Z}|$.

The proof of this proposition is in the appendix A.

The importance of expression (3.19) is that it provides a concrete measure for the anonymization performance of the representations. It bounds from below the

misclassification probability over the choice of the classifier $q_{\hat{Z}|U}$, using the sanitized representations. The right hand side is a quantity that involves the empirical mutual information between the representations and the private labels. It should be pointed out that since in many cases $\widehat{\mathcal{H}}(Z) \approx \mathcal{H}(Z) \equiv \log|\mathcal{Z}|$, assuming p_Z is uniformly distributed over the set $\mathcal{Z}$, then:

$$\inf_{q_{\hat{Z}|U}} \hat{P}_e(q_{U|X}, q_{\hat{Z}|U}) \gtrsim g^{-1}\left(\widehat{\mathcal{H}}(Z|U)\right), \tag{3.23}$$

and using our approximation in (3.16) the lower bound in (3.23) leads to an approximate but computable lower bound on the misclassification probability of the private labels. However, in order to provide statistical guarantees on (3.23), we need to study confidential bounds on $\mathcal{D}(\hat{q}_{Z|U}\|q_{\hat{Z}|U}|\hat{q}_U) \leq \delta$ which goes into the scope of further development of this work and is an interesting perspective.

3.3 Anonymization with Deep Neural Networks

Our ultimate goal is to learn parameters $\mathbb{R}^{d_c} \ni \boldsymbol{\theta}_c \mapsto q_{U|X}$ of a deep encoder and parameters $\mathbb{R}^{d_r} \ni \boldsymbol{\theta}_r \mapsto q_{\hat{Y}|U}$ and $\mathbb{R}^{d_p} \ni \boldsymbol{\theta}_p \mapsto q_{\hat{Z}|U}$ of the classifiers, (d_c, d_r, d_p) being the parameters' dimensions. In the following, we introduce a simplified notation to rewrite the objective (3.18) as:

$$\boldsymbol{\theta}^* \equiv \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{ \mathcal{L}_r(\boldsymbol{\theta}_c, \boldsymbol{\theta}_r) + \lambda \cdot \left| \mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) - \mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) \right| \right\}, \tag{3.24}$$

for a suitable hyperparameter $\lambda \geq 0$ to tune the trade-off between regular and private tasks, where all involved parameters are simply denoted by $\Theta \ni \boldsymbol{\theta} := (\boldsymbol{\theta}_c, \boldsymbol{\theta}_r, \boldsymbol{\theta}_p)$ with

$$\mathcal{L}_r(\boldsymbol{\theta}_c, \boldsymbol{\theta}_r) \equiv \mathcal{L}_{\text{emp}}(q_{\hat{Y}|U}, q_{U|X}), \tag{3.25}$$

$$\mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) \equiv \mathcal{L}_{\text{emp}}(q_{\hat{Z}|U}, q_{U|X}), \tag{3.26}$$

$$\mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) \equiv \mathcal{L}_{\text{emp}}^{\text{obj}}(q_{\hat{Z}|U}, q_{U|X}). \tag{3.27}$$

Assume a training set $\mathcal{T}_n$ of size n, where each element of the dataset $(\boldsymbol{x}_i, y_i, z_i)$ is composed of $\boldsymbol{x}_i \in \mathcal{X} \equiv \mathbb{R}^m$ is a real vector of size m, the regular label of the sample $y_i \in \mathcal{Y}$ and private label of the sample $z_i \in \mathcal{Z}$.

3.3.1 Adversarial training objective

Each classifier branch of the proposed architecture, i.e., $q_{\hat{Y}|U}$ and $q_{\hat{Z}|U}$, is trained to minimize the associated cross-entropy loss, whereas the encoder $q_{U|X}$ will be trained to simultaneously minimize the cross-entropy loss on the prediction of Y while maximizing an adversarial loss defined with respect to the private label predictor Z.

Each sample input $\boldsymbol{x}_i$ produces a representation $\boldsymbol{u}_i \sim \boldsymbol{q}_{U|\boldsymbol{X}=\boldsymbol{x}_i}$ and outputs two probability vectors $\boldsymbol{q}_{\hat{Y}|U}(\cdot|\boldsymbol{u}_i)$ and $\boldsymbol{q}_{\hat{Z}|U}(\cdot|\boldsymbol{u}_i)$ as soft predictions of the true labels: the regular one y_i and the private one z_i, respectively. The expressions of the losses we found in (3.25) and (3.26) are two cross-entropies computed over the whole training set:

$$\mathcal{L}_r(\boldsymbol{\theta}_c, \boldsymbol{\theta}_r) = \frac{1}{n}\sum_{i=1}^{n} \left\langle \boldsymbol{e}(y_i), -\log \boldsymbol{q}_{\hat{Y}|U}(\cdot|\boldsymbol{u}_i) \right\rangle, \tag{3.28}$$

$$\mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) = \frac{1}{n}\sum_{i=1}^{n} \left\langle \boldsymbol{e}(z_i), -\log \boldsymbol{q}_{\hat{Z}|U}(\cdot|\boldsymbol{u}_i) \right\rangle, \tag{3.29}$$

with $\boldsymbol{e}(y_i)$ and $\boldsymbol{e}(z_i)$ being "one-hot" vectors (y_i component is 1 and the others 0) of the true labels of sample $i = [1:n]$.

Let us now consider the adversarial objective. There are too many possible networks that mismatch the private labels and maximize the corresponding cross-entropy. In

particular the cross-entropy loss on the private label predictor could be increased arbitrarily by making it produce a wrong answer with high probability, which would not make much sense in our context. Hence, we want to maximize this cross-entropy but not more than that of the cross-entropy of a predictor which would be unable to distinguish among the identities, i.e., with a posterior distribution approximatly equal to $\hat{p}_Z$:

$$\mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) = \frac{1}{n} \sum_{i=1}^{n} \left\langle \hat{\boldsymbol{p}}_Z, -\log \boldsymbol{q}_{\hat{Z}|U}(\cdot|\boldsymbol{u}_i) \right\rangle, \tag{3.30}$$

which is indeed expression (3.15). This artificial loss, formally introduced by our surrogate (3.18), denotes the cross-entropy between the vector of empirical estimates of probabilities $\hat{\boldsymbol{p}}_Z$ and the predictions $\hat{\boldsymbol{z}}$. By forcing private task predictions to follow the estimated empirical probability distribution of the private labels (in many cases close to equiprobable labels) the encoder is trained to produce anonymized representations. Indeed these representations private features are expected to fool a classifier into learning a random guess model on the private task. Keep in mind that random guessing is a universal lower bound for anonymization. In fact, if the private label predictor had a cross-entropy loss higher than that of the random guessing predictor, the surrogate indicates we must *reduce* its loss. This is consistent with the adversarial training objective in (3.24). Notice that if our predictions follow the random guessing distribution then the term $\left|\mathcal{L}_p^{obj}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) - \mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p)\right|$ approaches zero.

This upper bound of the cross-entropy of the private classifier induces large improvement on the training compare to a regular cross-entropy maximization without an upper bound. The training is more robust and avoid numerical difficulties which may arise from the number of private labels while using an unbounded loss. Indeed the straight application of the loss of [Ganin and Lempitsky, 2015] for task where the number of private label is more than two (in domain adaptation there is usually only two domains) gave inconclusive simulations. These numerical difficulties arise in particular when the cross-entropy was worse than a random guess predictor on the private-label.

3.3.2 Architecture

The architecture is similar to that of [Ganin and Lempitsky, 2015] (especially for the gradient reversal layer showed in the Fig. 3.1), initially introduced in the context of domain adaptation. The goal of domain adaptation is to train a loss on a dataset and be able to apply it efficiently on a different but related dataset. Here we have two branches on top of a common encoder. One branch is the regular task classifier and the other one is the private task classifier. The general shape of the network is shown on Fig. 3.1. The fact that the backpropagation is done while reversing the sign we coming from the private branch to the encoder is due to the usage of a layer called the gradient reversal layer, showed with pink arrow. This layer is first introduced in [Ganin and Lempitsky, 2015]. It is not actually a layer, but just a specific connection where the forward information is untouched but the backward gradient is multiplied by -1.

3.4 Algorithm

3.4.1 Training procedure

We have found best results according to the following adversarial training procedure.

1. The encoder and regular label predictor are jointly pre-trained (as a standard deep network) to minimize the regular label cross-entropy (eq. (3.28)).

2. The encoder is frozen and the private label predictor is pre-trained to minimize its cross-entropy (eq. (3.29)).

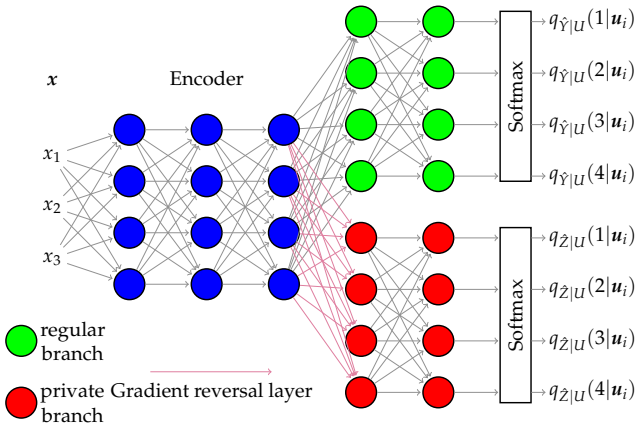


FIGURE 3.1: Architecture of the proposed deep neural network. Note that the number of layer here is just for readability of the figure and not representative of the actually used architecture

3. Adversarial training is organized by alternatively either training the branch predictors or training the encoder:

   (a) Sample N training examples and update both branch predictors with respect to their associated cross-entropies, using batch stochastic gradient descent (SGD) (i.e. the N examples are broken down into batches, with one update after each batch).

   (b) Sample N training examples and update the encoder to minimize the adversarial objective (eq. (3.24)), again using batch SGD.

In our experiments, we simply picked N as the size of the training set, so we alternated between the two kinds of updates after several epochs on each. We used batch SGD with Nesterov momentum [Nesterov, 2007].

3.4.2 Testing procedure

Once the training is done we need to test the produced results. To do so the data is processed through the encoder. The representations are then associated with both their original labels (private one and regular one). We assume the worst case scenario: an attacker have access to the private labels of the training data. The idea is to test for the presence of remaining private information inside the representations. We design an assessment network to learn to classify data from the training set according to their private labels. It is a network whose task is the same as the private branch network .i.e classifying as accurately according to the private labels. Once trained to produce the best possible results we measure the performances on the processed test set (processed through the encoder). The accuracy of this classifier on the test set representation gives us the performance of anonymization of the encoder. A lower accuracy means a better anonymization. The performances of the regular task are measured in the same fashion using a second assessment network trained on the encoded representations from training set and tested on the encoded representations from the test set. This gives the accuracy of the regular task for the encoded representation. The accuracies (regular and private) are plotted in the figure of the next section as a function of λ.

Note that this is the worst case scenario concerning the privacy. Indeed the attacker dispose of all the possible information (i.e. all the private labels) he needs to train a de-anonymizer network.

3.5 Experimental Results

The dataset used in this section are presented in the chapter 2. The ones used here all have two labels for each samples. One will be used for the private task classification and the other one will be used for the regular task classification. Information about the label is at the beginning of each dataset subsection.

We emphasize that the present method gives an anonymizer for the whole dataset, as opposed to anonymizing a query related process or a subset of the dataset. In order to tune the anonymization, we have trained a network for a wide range of values of λ. For each of them, we compute the accurate rates of both tasks: the private and the regular labels.

3.5.1 Toggle (or sequential) vs simultaneous training

The procedure we found to provide better results when training the parameters of our deep neural nets is a *toggle training*, as opposed to simultaneous training [Ganin and Lempitsky, 2015] where all updates at the encoder and at the branches occur at the same time. With toggle training the updates are performed either at the encoder or at the branches (Fig. 3.1). The purpose is to let the branches of the network to keep track of the encoder updates. This method has a key role in learning useful representations. Indeed, if classifiers are performing as efficiently as possible on their own tasks, they will feedback the most relevant information to update the encoder. In Fig. 3.2, we confronted the result of toggled training versus the simultaneous (or concurrent) training method. The regular task classification accuracies are plotted as a function of the private task anomymization accuracy. To keep this comparison fair, we found better to chose a lower learning rate on the encoder than on the branches. Note the different scale between the left axis (toggle training) and the right axis (simultaneous training). We can observe that simultaneous training enables only two regimes: either a light anonymization, with almost no available trade-off, or a strong anonymization, where a few features relevant to the regular task remain. Indeed, after training with a significant large range of λ values, we found the network to randomly converge to either of these extremes, that is why several points are not achievable and thus, missing in the plots. The toggle training makes reachable some trade-off points (high regular task accuracies while providing some degree of anonymization) that was not available in the other method.

3.5.2 Pen-digits database

The regular task consists in digit classification and the private task consists in writer identity classification. The trade-off between the regular and private tasks is presented in Fig. 3.3. The ▲-curve corresponds to the test accuracy on the private task while the $\star$-curve denotes the test accuracy on the regular task. The doted curve is the probability of error lower bound on the private task estimated with the values of cross entorpy loss function after the verification model is trained. This denotes the estimation of the private task accuracy according to (3.23) using (3.16) computed on the loss of the test-set (under the uniform private label empirical distribution hypothesis, which is true on this dataset). The rather good fitting indicates that (3.16) is a reasonable approximation.

Interestingly, the impact on the regular task stays contained inside the previously approximated baseline lower bound for the classification error probability. Some interesting conclusions can be drawn from these plots. Its ordinate reads on the right axis. The value of the accuracies of both tasks at $\lambda = 0$ is interesting. Indeed, when $\lambda = 0$ the network is updated without any concern of the private task. On the other hand, the baseline for the private task was computed separately with a dedicated network (equivalent to cascading a network similar to the encoder and the private branch). The accuracy baseline for the private task in theses conditions was found to

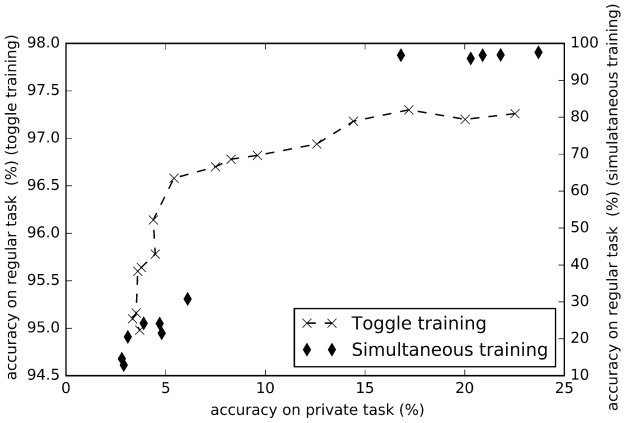


FIGURE 3.2: Comparison of the accuracy on regular task between toggle training and simultaneous training, on the Pen-digits database, as a function of the accuracy on the private task. Note that the left axis for toggle training is using a very different scale from the right axis for simultaneous training. Toggle training provides a better trade-off for the anonymization than the simultaneous training. Simultaneous training enables only two regimes: either a light anonymization, with almost no trade-off, or a strong anonymization, where a few features relevant to the regular task remain. Indeed, for a significant large range of λ values, the network randomly converges to either of these extremes, which allows only to trade-off between a few accuracies (i.e. several missing points).

be around 40%. Nonetheless, Fig. 3.3 shows a much lower accuracy because only the branch part of the network is trying to improve the classification score of the private labels, the encoder focuses in the situation of $\lambda = 0$ only on the regular part. As for the regular task, it is worth to mention that the results may vary since randomness impacts the training and thus the score as well. To average this noise, several simulations were made for the baseline obtaining scores between 97.65% and 98.45%. The impact of λ is quite important and is shown by the abrupt variation over the interval $\lambda \in [0, 1]$. After this significant decrease in the accuracy the variations are slower, even so the accuracy of this task tends to decrease a little. Interestingly, regarding the score of the regular task, variations are significantly more tenuous. Their interpretation only shows that the increase in λ does not induce any remarkable change. The impact of the private branch on the network, if such an impact exists, is rather marginal.

3.5.3 FERG database

The regular task consists in facial emotion classification and the private task consists in identity classification. The network is composed of 1200 neurons per-layer. The encoder is composed of 5 layers and each branch is formed by 3 layers, other network parameters remain the same as in our previous network configuration. The plentiful samples in the database give really strong accuracies baselines for both tasks: 100% on the private task and 98.2% on the regular task. Fig. 3.4 shows the trade-off, the $\star$-curve indicates the test accuracy on the regular task which decreases from 98.2% to 95.99%. The ▲-curve indicates the test accuracy on the private task which decreases significantly from 100% to 56.59%. Due to the non-uniform distribution of the samples among classes, the random guessing classifier over the user-ID is 19.83%. One should notice that the six characters have really different facial features, therefore they are easy to identify on the original images (private task baseline 100%). Yet, the representations learnt by the network leads to a significant anonymization with an acceptable cost on the regular task. Feeling recognition and face recognition are rather intertwined tasks. The observed degradation of performance comes from the contradictory natures of both tasks, i.e. , raising the level of anonymization comes at the cost of blurring some

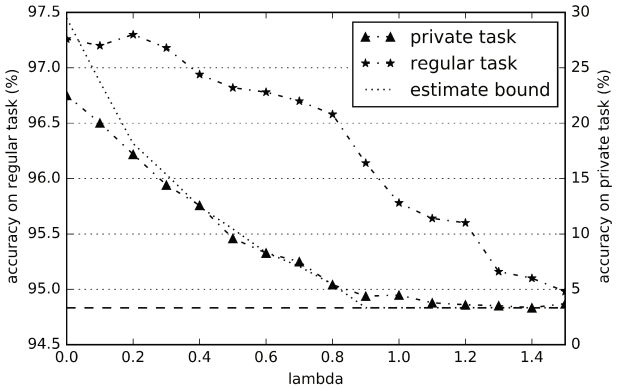


FIGURE 3.3: Accuracies as a function of $\lambda \in [0, 1.5]$ on Pen-digits database. The horizontal black dashed line is the random guessing classifier over the user-ID (3.33%). It displays the trade-off that occurs on the data set, i.e., a level of anonymization is ensured at the cost of a small performance decrease on the regular task. Dotes curve shows that eq. (3.23) with (3.16) is a reasonable estimation.

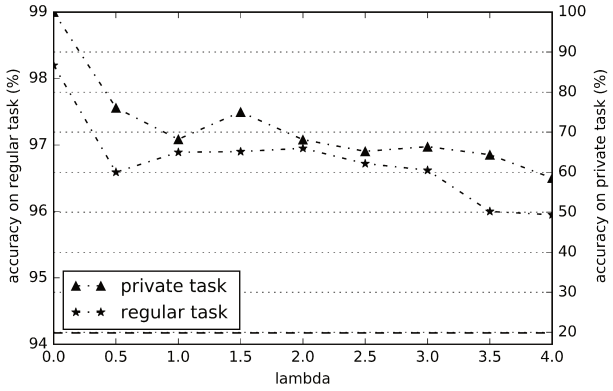


FIGURE 3.4: Accuracies as a function of $\lambda \in [0, 4.0]$ on FERG database. The horizontal black dashed line is the random guessing over the user-ID (19.83%). The available amount of samples allow the learning of anonymized still relevant representations but at a small cost on the regular task. For sake of clarity, $\lambda = 4.5$ is not plotted since both tasks decreased to random guessing accuracy.

relevant features for the regular task. The results on this database show that despite the numerous examples of the database, we cannot reach the anonymity bound, it shows that trade-offs are related to the specific nature of data and the intertwinement of their features.

3.5.4 JAFFE database

The regular task consists in facial emotion classification and the private task consists in identity classification. We note that the anonymity induced by the structure of the network itself ($\lambda = 0$) is not apparent here. The accuracies of both tasks are shown in Fig. 3.5 as a function of λ. As λ increases, the anonymization is made more apparent, i.e., the ▲-curve is decreasing from 44.59% to 21.62%. It is clear that the trade-off is stronger on this dataset than on the previous one which can be observed from the regular task ($\star$-curve), feeling recognition, that declined from 39.19% to 25.68%. This significant performance degradation is due to the contradictory natures of the tasks but also to the limited samples, which emphasizes that encoder performance is sensitive to branches training performances: with fewer samples, the branches have a lower capacity of generalization.

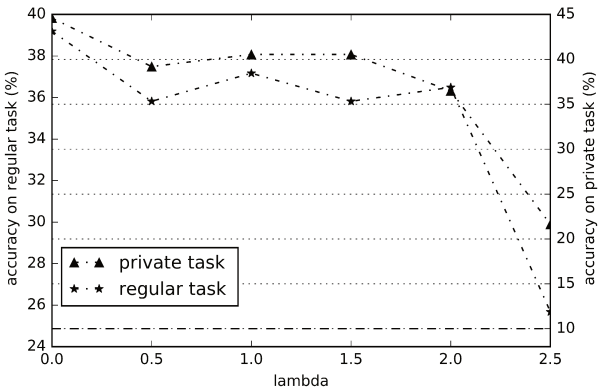


FIGURE 3.5: Accuracies as a function of $\lambda \in [0, 2.5]$ on JAFFE database. The horizontal black dashed line is the random guessing over the user-ID (10%). Despite a rather small dataset, the anonymization still occurs but at the cost of a significant (non-negligible) impact on the regular task performance.

3.6 Conclusion

This chapter provides an efficient method to obtain the available trade-off between maintaining a sufficient efficiency on a regular classification task (recovering the "useful" information) while maintaining as confidential as possible another information. Our information theoretic approach is based on the estimation and control of mutual information between a representation and a set of labels. This required the ability to both minimize and maximize the mutual information approximation applied respectively to the set of private labels and the set of regular labels. Note that maximizing an approximation gives less control on the criteria than maximizing a lower bound. The results were anyway depicting reasonable trade-offs between the two contradictory tasks. This problem will be addressed in a future work.

The current approach was obtained via the optimization of a new loss that manages through an hyper-parameter to tune the trade-off between the amount of useful information and private information that is released in the learned representation. This loss in itself was not sufficient to achieve the trade-off: we had to design an appropriate network architecture to be used in a machine learning framework. The results are quite satisfying on pictures database. Despite depending on the contradictory nature of the regular data and private data, we achieved conclusive results: we reach the anonymity lower bound at least on the pen-digits task. On the FERG database, the task was indeed much more difficult due to the fact that the features are strongly intertwined.

Chapter 4

Semi-supervised anonymization

4.1 Presentation of the problem

In chapter 3, we addressed a quite constrained problem: the regular task has to be known before undertaking the network optimization. This is a very strong constraint that we will not consider in this chapter: we are requiring that any queries possible on the original image should be possible on the processed data, while keeping the private label hidden. Allowing to have the same queries on the representations as on the original data, necessarily guided us into having representations with the same nature as the original data. It does not suppose or guarantees anything on the performances of those queries. Therefore, the process is supervised with respect to the private label, but unsupervised with respect to the original image. That is why we choose to call it a semi-supervised anonymization.

4.1.1 New objective: new architecture

There is a main difference in the goals between the previous fully supervised scheme, and this one. Obviously we still know what to hide: private labels are still available. However, we would like to be able to perform any other search in the processed data. Therefore, we cannot make any assumption about what is useful in the original data. Indeed, the method presented in the previous chapter was really different and did not preserve all the information (i.e. it was designed to keep relevant labeled information and was achieving its purpose). We need to emphasize on the limits of the previous method to motivate the new challenge.
With some craftiness, it is theoretically possible to used the supervised method to keep more than one feature at a time. Let us imagine that you want to keep information about a couple of features like for example (eyes status,mouth status) which for both could be in the set $\{closed, half\ closed, open, wide\ open\}$. It is simple to devise a single labeling method that would take into account both features label and therefore simply apply the supervised method. It could then seem less necessary to design a new method since in theory it is possible to store several features in one label. The limitation is not here indeed: what if, to keep with the same example, we want to retrieve, once the network is designed and the database anonymized, information about the position of the eyebrows. The representations were not designed to keep any eyebrow information. There is no guarantee that the inference we can do on the eyebrow position from the anonymized representations will be correct or even useful. The limitation of our supervised method presented in the previous chapter is that the features we want to keep have to be know before processing the database.
The work of this chapter focuses on providing anonymization while keeping as much possible of the original (non private) features. In other words, the anonymized representation we seek to achieve should not suppose anything about what we need to keep. A solution to this issue is to keep as much information as possible inside the obtained representation of the data, except the one that has to be kept private. This means that the outputted processed data need to have the same type as the input data. For example, the model needs to produce an image from image input. Any change in the nature of the data would compromise our hypothesis of compatibility of queries: anonymized

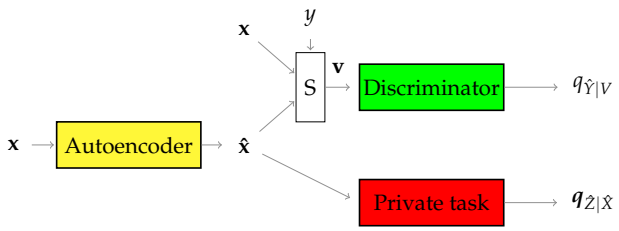


FIGURE 4.1: Semi-supervised architecture (S is the selector block)

representations have to be compatible for original data queries. Therefore the lack of knowledge about useful information yields a fundamental shift in our model: the metrics used to evaluate the output have to be a mix between a fidelity metric (in order to ensure that as much information as possible remains) and the previous private metrics. In order to do so, a similarity metric needs to be chosen. Choosing a fixed metric tailored for a specific application, or type of input looks appealing and would produce good results on specific database but it will most likely fail on a different type of input or application. Furthermore, the model we wish to design should also allow some changes, since one has to remove the information corresponding to the private label. In fact, this is not compatible with a high fidelity fixed metric that prevents any pixel to shift too much. In fact, a preliminary work has been done using cumulative Euclidean distance between pixels as a fidelity metric and the results were quite disappointing since this choice prevented the needed changes to occur. This is the motivation to use a GAN-like discriminator as the flexible metric to be trained. This has the advantage of keeping a really flexible range of applications and yet to obtain accurate results. Note that the flexibility of the metric is also useful regarding the training of the model for a specific application: this flexibility allows the model to have some degrees of freedom which can be used to shift the output of any type of data into an anonymized space.

In order to have comparable inputs and outputs, the encoder used in the previous chapter should be augmented by a decoder. The resulting architecture, displayed on Figure 4.1, is an auto-encoder followed by two branches, namely: one discriminator and one private features classifier. On the discriminator branch, one finds first a selector, denoted S, which allows a selection between $\mathbf{x}$ and $\hat{\mathbf{x}}$, followed by the discriminator. More precisely, the input $\mathbf{v}$ of the discriminator is the output of the selector block, driven by a random variable y that decides on the output of the selector S. If $y = 1$ then $\mathbf{v} = \mathbf{x}$ and if $y = 0$ then $\mathbf{v} = \hat{\mathbf{x}}$.

4.1.2 Training objective

Here our goal is to learn three sets of parameters: **i)** $\mathbb{R}^{d_c} \ni \boldsymbol{\theta}_c \mapsto q_{\hat{X}|X}$ of a deep auto-encoder; **ii)** parameters $\mathbb{R}^{d_d} \ni \boldsymbol{\theta}_d \mapsto q_{\hat{Y}|V}$ of the discriminator; and **iii)** parameters $\mathbb{R}^{d_p} \ni \boldsymbol{\theta}_p \mapsto q_{\hat{Z}|\hat{X}}$ of the private classifiers, (d_c, d_d, d_p) being the parameters' dimensions.

Variable Y is binary and gives the information whether X or $\hat{X}$ (either the original data or the autoencoded data) has gone through the selector and is inside V to enter the discriminator during its training. Respectively $q_{\hat{Y}|V}$ is the output distribution of the discriminator giving the estimated probability of input being $V = X$. In the following, we introduce a simplified notation to rewrite the objective (3.18) as:

$$\boldsymbol{\theta}^* \equiv \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{ \mathcal{L}_d(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d) + \lambda \cdot \left| \mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) - \mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) \right| \right\}, \tag{4.1}$$

for a suitable hyperparameter $\lambda \geq 0$ which tunes the trade-off between the discriminator and the private task, where all involved parameters are denoted by $\Theta \ni \boldsymbol{\theta} :=$

$(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d, \boldsymbol{\theta}_p)$ with

$$\mathcal{L}_d(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d) \equiv \mathcal{L}_{\text{emp}}(q_{\hat{Y}|V}, q_{\hat{X}|X}), \tag{4.2}$$

$$\mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d) \equiv \mathcal{L}_{\text{emp}}(q_{\hat{Z}|\hat{X}}, q_{\hat{X}|X}), \tag{4.3}$$

$$\mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) \equiv \mathcal{L}_{\text{emp}}^{\text{obj}}(q_{\hat{Z}|\hat{X}}, q_{\hat{X}|X}). \tag{4.4}$$

Assume a training set $\mathcal{T}_n$ of size n, where each element $(\boldsymbol{x}_i, z_i)$ is composed of $\boldsymbol{x}_i \in \mathcal{X} \equiv \mathbb{R}^m$, a real vector of size m and of $z_i \in \mathcal{Z}$, the private label of this sample. Additionally define the vector Y_D with components $y_i \in \{0, 1\}$ which is used during the training of the discriminator, and another vector Y_A with components $y_i = 1$, used for the training of the auto-encoder. Each branch of the proposed architecture, i.e., $q_{\hat{Y}|V}$ and $q_{\hat{Z}|\hat{X}}$, is trained to minimize the associated cross-entropy loss, whereas the auto-encoder $q_{\hat{X}|X}$ will be trained to simultaneously minimize the cross-entropy loss on the prediction $\hat{Y}$ of Y while maximizing an adversarial loss defined with respect to the private label predictor Z.

4.1.3 Loss

Compared to (3.24) (fully supervised objective), the new objective stated in (4.1) differs with the term $\mathcal{L}_d(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d)$. This term is the loss of the discriminator. During the discriminator training (while the auto-encoder parameters are fixed) minimizing this loss should improve the discriminator performance. The output $\hat{Y}$ should be as accurate as possible to predict if the data are auto-encoded ($Y = 0$) or original ones X taken from the dataset. This discriminator, like the one in GAN, measures through its output $\hat{Y}$ the fidelity the auto-encoded samples compared to the original samples. This information is then used to improve the parameters of the auto-encoder θ_d.

The purpose of this loss during the training of the auto-encoder (while the discriminator parameters are fixed) is different since it acts as a similarity metric. Each sample input $\boldsymbol{x}_i$ produces an output $\hat{\boldsymbol{x}}_i \sim q_{\hat{X}X=\boldsymbol{x}_i}$ which outputs a probability vector $\boldsymbol{q}_{\hat{Z}|\hat{X}}(\cdot|\hat{\boldsymbol{x}}_i)$ as soft predictions of the true private labels z_i. It outputs $q_{\hat{Y}|V}(1|\boldsymbol{v}_i)$ as the probability of $y_i = 1$.
The precise definitions of the losses are as follows.
The loss of the discriminator is a binary cross-entropy:

$$\begin{aligned}\mathcal{L}_d(\boldsymbol{\theta}_c, \boldsymbol{\theta}_r) = -\frac{1}{n}\sum_{i=1}^{n}\Big[&y_i \log q_{\hat{Y}|V}(1|\boldsymbol{v}_i) \\ &+(1-y_i)\log(1-q_{\hat{Y}|V}(1|\boldsymbol{v}_i))\Big].\end{aligned} \tag{4.5}$$

The private classifier loss is unchanged in its nature, just adapted to the new model (the same cross-entropy as (3.29) computed over the whole training set):

$$\mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) = \frac{1}{n}\sum_{i=1}^{n}\left\langle \boldsymbol{e}(z_i), -\log \boldsymbol{q}_{\hat{Z}|\hat{X}}(\cdot|\hat{\boldsymbol{x}}_i)\right\rangle, \tag{4.6}$$

with $\boldsymbol{e}(z_i)$ being "one-hot" vectors (z_i component is 1 and the others 0) of the true labels of sample $i = [1 : n]$. The objective cross-entropy is also the one of a predictor which would be unable to distinguish among the identities, i.e., with a posterior distribution approximately equal to $\hat{p}_Z$:

$$\mathcal{L}_p^{\text{obj}}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) = \frac{1}{n}\sum_{i=1}^{n}\left\langle \hat{\boldsymbol{p}}_Z, -\log \boldsymbol{q}_{\hat{Z}|\hat{X}}(\cdot|\hat{\boldsymbol{x}}_i)\right\rangle, \tag{4.7}$$

which is indeed the expression (3.30) with $U = \hat{X}$. This artificial loss serves the same purpose as in the fully supervised case. It denotes the cross-entropy between the vector of empirical estimates of probabilities $\hat{\boldsymbol{p}}_Z$ and the predictions $\hat{\boldsymbol{z}}$. By forcing private

task predictions to follow the estimated probability distribution of the private labels (that may or may not be close to equiprobable labels) the model output is expected to prevent any classifier from having better prediction that a random guessing. As stated before, random guessing is a universal lower bound for anonymization. Note likewise that if the predictions follow the random guessing distribution then the term $\left|\mathcal{L}_p^{obj}(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p) - \mathcal{L}_p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_p)\right|$ approaches zero.

4.1.4 Training procedure

The training procedure in the semi-supervised case is quite different from the fully supervised case.

1. The parameters of the auto-encoder are initialized using a classical pixels to pixels metric loss during the pretraining of the auto-encoder, which produces relevant image in shape but with poor frontier definition. However, this constitutes a good starting point for the next steps of the training.

2. The auto-encoder and the discriminator are trained together with alternatively:

   (a) updates on the discriminator parameters while auto-encoder parameters are frozen, with inputs being a combination of auto-encoded samples and original samples i.e. $\mathbf{x}$ and $\hat{\mathbf{x}}$ going through the selector block as illustrated in Figure 4.1. By doing so, the discriminator learns the difference between both distributions. This step is similar to the training of the discriminator in GAN.

   (b) updates on the auto-encoder while the parameters of the discriminator are frozen. The previous training of the discriminator lets him be a relevant flexible metrics to improve the distribution of the outputs $\hat{X}$

3. The next part is the alternate updates between the auto-encoder on one side and the discriminator and the private branch on the other side. This part is putting the privacy constraint on the auto-encoder. It is during this step that the privacy/fidelity trade-off occurs.

Note that among all the differences between the full supervised case and the semi supervised case an important one is the use of convolutional network. The anonymized representation changed (sample-like type representation in the semisupervised case, here image, instead of a feature vector in the fully supervised case). Therefore the network needed a change as well to adapt the new representation.

4.2 Simulation results

The dataset used in this section are presented in the chapter 2. The ones used here have two labels for each samples. One label is used for the private task classification and the other one is used for the regular task classification. Information about the label is at the beginning of each dataset subsection.

4.2.1 Pendigits database

The pen-digits database is used to validate the new architecture. The private task label are the writer identity label. The task is to transform handwritten digit pictures into other handwritten digit pictures while removing the private task related feature while at the same time having the encoded image keeping relevant information about the original image digit.

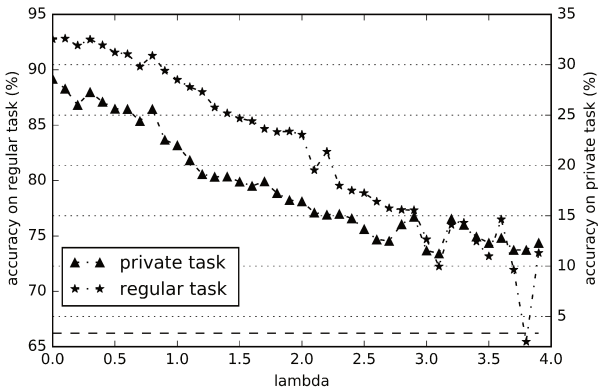


FIGURE 4.2: Accuracies as a function of $\lambda \in [0, 4]$ on Pen-digits database. The horizontal black dashed line is the random guessing classifier over the user-ID (3.33%).

Network configuration

As stated earlier a lot of things have changed for this semi-supervised application. The auto-encoder part of the network is composed of 5 convolutional layers. Between each convolutional layer a dropout layer is applied. The first layer is composed of 128 4x4 pixels filters and Relu activation function that reduces the size of the picture from 20x20 to a 5x5 matrix. After that two convolutional networks of each 64 3x3 filters and with each a Relu activation function are applied. The sample is then progressively resized to the inital picture size: first an up-sampling layer transforms the 5x5 matrix into a 10x10 matrix that is then process by a Relu 128 4x4 filters. Finally the picture is up-sampled again to 20x20 pixels and processed by a single filter 4x4 sigmoid convolutional network. The discriminator part of the network is also composed of Relu convolutional layers. There are two of them: the first one is a 64 5-by-5 filters layer and the second one is a 128 5-by-5 filters layer. After that, a simple dense one neuron with sigmoid activation is applied to produce the output of the discriminator. The private features classifier follows the same architecture. It is composed of Relu activation convolutional layers. The first one is a 64 5-by-5 filters layer and the second is a 128 5-by-5 filters layer. The final layer is a 30 neurons softmax activation layer.

Results

Here the private labels are the writer identity labels that the encoder tries to remove from the representations. Once our model is trained using the above mentioned architecture, we only use the autoencoder part. This allows to autoencode original images into transformed images. One thing left to do is to judge the results of this processing. A human assessment of the output pictures would be too tedious. Assessing the performance of our network in a fair and automated manner is not really easy because it implies comparing and judging pictures. The solution we have chosen considering the anonymization task and the information inside the database is the following: the whole dataset is processed by our trained network. The resulting images are associated with the original images labels. For each processed image we have a private label that identifies the writer identity of the original image and we have a regular label that gives the value of the number on the original image. Two assessments networks are then trained using each of theses labels. These networks, once trained as accurately as possible, are used with some test data in order to output some accuracy percentage on each task: the regular one consisting in identifying the written numbers and the private task aiming at identifying the writer identity. Both these percentages are plotted in the Figure 4.2.

This method has downsides because we assess a semi-supervised transformation in a fully supervised manner but it is a good compromise to quantify at the same

time the change regarding the private features and the change to some other features. The strong side of this assessment method is that it is very pessimistic regarding the anonymization: We use all the private information available to train the model that should extract private information from the data, this is a best case scenario for an ill intended person wanting to recover data from the processed database. Regarding the general preservation of the picture, the assessment is done using only one of potentially many regular features, yet we use to the full extent the label shipped with the database. However the information of the regular label has never been used during the training of the semi-supervised encoder. Therefore even if one regular feature may not be ideal, we still perform an assessment of fidelity on a feature that neither the decoder or the autoencoder have been trained for.

4.2.2 FERG database

Here the private labels that the encoder tries to remove from the representations are the users identity labels.

Special care for intertwined task and complex database

The standard procedure explained earlier worked well on the pen-digits database, but failed on the FERG database. On the FERG database, the failure is really insidious: the auto-encoder produce realistic images, that can easily fool human eyes but failed to fool a neural network. Using our standard architecture led to a network that can produce visually anonymized results, without computer validated anonymization. Our explanation is the following: According to the value of parameter lambda, there were 3 possible outcomes: either no change in the image (value of lambda too small), or all images changed into the same identity while transposing most facial expressions (intermediate value for lambda). The last possibility was when a too high lambda value was used and resulted in all images outputs being the same regardless of the input: either a full black image, or an artifact looking image i.e. an image without any information. With a carefully chosen value of lambda, the constraint put on the auto-encoder by the private branch and the discriminator leads to an interesting result: The images were blurred at the first epochs (privacy constraint part of the loss), the resulting blur kept a vague shape of a face. Progressively the images which first presented blurred forms converge epochs after epochs to a single identity. The resulting identity was always the same, despite having no explicit constraint on the network to influence such a choice. To our understanding, the first epochs blurred shape was some barycentric representation of a mixture of all identities. This make sense because the loss directs the network toward the empirical distribution of the training set i.e. a fixed proportions mix of each identities. It seems that the discriminator kicks in after (which seems reasonable considering the privacy part of the loss term is quite low in this configuration and around these epochs.) and it constrains the trunk in the following epochs into a shift from this barycenteric representation to the closest existing identity (closest in the sense of the discriminator similarity metrics which is hard to interpret in geometric terms). This shift is done quite progressively and uniformly on all the samples, therefore the privacy constraint part of the loss remains low during theses epochs, while the discriminator loss is decreasing on average. The resulting images are quite convincing and can definitely fool a human operator trying to classify output images according to their identity.

Despite a good anonymization on the human assessing level, the output failed to fool a neural network classifier trying to predict the identity. The failure is intuitively surprising because of the neat quality of the output images, nonetheless a neural network trained to classify the identity from the output images has an error rate between 0% and 2% leaving no doubt on the lack of anonymization of this first method. The quality of the images is not a sufficient criteria to assess anonymity. Nonetheless,

even not suited for anonymization purposes, this method can assuredly be used to satisfy aesthetic purposes.

The second method we developed is based on the same architecture. The main novelty of this method is the restriction we used on the classifier training. The core idea of this improvement is to remove useless identity from the pool used to train the discriminator. Instead of using the whole dataset to train the discriminator we restricted its training pool to only one arbitrary chosen identity. This restriction forced the discriminator to learn a special identity and its features. Such a trained discriminator mechanically impacts the autoencoder to shift all samples toward the arbitrary chosen identity. The impact of this variation is strong enough to produce qualitative results (in the sense of neat image according to the human eye) even when the private branch is not used (case $\lambda = 0$). In this configuration the tests over the value of lambda produced better results but were far from conclusive. Indeed all of the regimes decided by the lambda value are not producing satisfying results. In the case of small lambda, the autoencoder produces human level grade fooling images but a neural network testing proves that there is no anonymity at all (100% accuracy on the identity classifier).

For the optimal value of lambda, the private branch influence kicks in. Even if the difference on the produced outputs from the previous regime is not visually blatant (the image more or less looks as neat to the human eye as the previous regime) the impact on the anonymity is measurable with the neural network classifier. The accuracy for the identity classification task is 96%. This small gain in anonymity is yet interesting because it proves that the images produced by the autoencoder with this method are better than the ones produce by the previous one. The last regime is for higher value of lambda, where there is too much degradation, and the resulting images are not usable. The final method we used on strong intertwined tasks is entirely based on the usage of the previous method. The idea is to take advantage of the high visual quality of the resulting images. Our previous analysis showed that the problem of the outputed images are far from being anonymized despite a conclusive visual analysis. Not being able to directly output anonymized images, we decided to enforce in a stronger manner a guaranteed anonymity. The method is to substitute outputted images by images from the training database directly.

For the sake of clarity let consider the problem from a geometric point of view. The space we use is the one defined by a generic similarity measure such as the cumulative Euclidean distance between pixels. In the case of intertwined database you can consider the following. The dataset is composed of disjoint set or more precisely of not overlapping space regions defined by the common label of their samples (recall we only have knowledge over the private label). In order to have a meaningful purpose and at the same time to be anonymization worthy, the data need to have an (supposedly unknown) underling structure inside each of this space region but also this structure should be generalized over all space regions. In other words the dataset is composed of several disjoint clusters which labels we want to anonymized, and inside each of them there are several sub-clusters all sharing an exploitable common structure regardless of their cluster of origin. To illustrate this with the FERG database: consider that even if all images of one identity form a cluster separable from the other identity clusters, they all share underlying sub-cluster structures. For example they all have the constitutive elements of a face (eyes, eyebrows, mouth, nose etc.). Moreover they also share the emotion sub-cluster structure which is strongly linked to small hierarchical variations of the facial elements.

Our idea is to replace all sub-cluster samples by samples coming exclusively from a specified cluster. The way we choose to do it is to spatially shift clusters from their initial positions to the chosen cluster position without losing each of the underlying unknown sub-cluster structure. If the shift is done carefully and the sub-cluster does exist they will overlap one another keeping the underlying structure intact. Once the sub-clusters overlap it is easy to measure their similarity and to affect any sample to its closest neighbor from the specified cluster. Once the spatial shift is done the substitution comes easy with generic metrics, all thanks to the spatial shift. To put

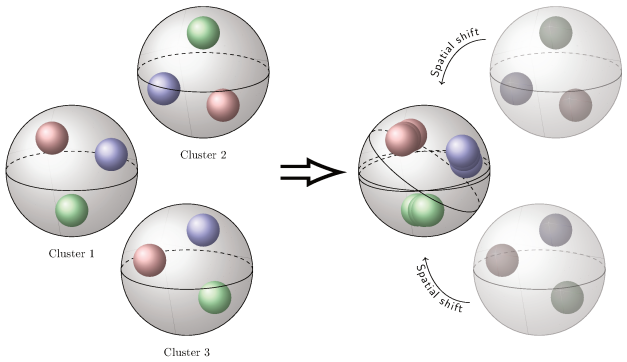


FIGURE 4.3: 3D sketch of the initial state of the dataset in the left and the processed dataset on the right with initial position of cluster 2 and 3 represented faded. The spatial shit aligned the sub-cluters. The cluster all share a similar subcluster structure (red green and blue sphere).

things into perspective let consider the substitution without the spatial shift: with all clusters disjoint the substitution of all of a cluster's samples would only concern a few samples from the specified cluster. Indeed the distance between a cluster and the specified cluster without spacial shift can induce a substitution that only takes a few of the border samples from the specified cluster. See Figure 4.3.

Note however that substitution can only be used here because the representations have been carefully spatially shifted toward a specified area in the space of the dataset. If we had considered a substitution beforehand, it would have resulted in poor results: following the example on the right of the Figure 4.3, if we choose cluster 1 as reference, all of the samples of cluster 2 are spatially close to a few samples of the blue sub-cluster of cluster 1. This will result in many missed substitutions of cluster 2 samples. In a more general view, substitution inside the raw dataset do not work. Entire clusters are replaced by few samples of the specified cluster and the sub structure is not preserved. On the opposite, once the spatial shift is well done, clusters are brought close together and sub-clusters are well aligned. It allows a meaningful substitution where data are truly replaced and the unknown underlying structure is preserved without the use of its structure-related labels.

Network configuration

For the FERG database the network still has 5 convolutional layers in its auto-encoder part with between each layer a dropout layer of parameter 0.1. The parameters of each layer have changed: the first layer is a 90 5-by-5 filters layer that reduces the size of the samples. The size of the pictures is initially 50x50 pixels and is decreased to a 10x10 matrix after the first layer. The second layer is a 180 3-by-3 layer that further reduces the size of the sample matrix to 5x5. The third layer is a 180 3-by-3 filters layer that increases the size of the sample to 10x10 matrix. The fourth layer is a 90 3-by-3 filters layer that restores data to the original image size, that is 50x50 pixels. The fifth layer is a 3-by-3 single filter layer. The first four layers use a ReLU activation function whereas the last layer uses a sigmoid activation function. This way, the last layer output values are between 0 and 1 for each pixel. These values can be then converted into grey scale values in order to plot the image.

The discriminator is composed of two convolutional layers. The fist one is a 90 5-by-5 filters layer, and the second one is a 180 3-by-3 filters layer. These two layers have Relu activation function too. On top of that there is a single neuron sigmoid activation layer that outputs the estimated probability of the discriminator. The private task branch is composed of 64 5-by-5 filters convolutional layer followed by a 128 5-by-5 filters convolutional layer. Both of them use ReLU activation function. The last layer is the size of the private task cardinal: It is a six neurons softmax activation layer.

targeted identity	0	2	3	4	5
identity task accuracy (%)	42.47	38.25	36.21	32.77	31.62
emotions task accuracy (%)	72.45	77.46	72.32	66.79	72.95

TABLE 4.1: Accuracies for the anonymization (supervised during training) task with different targeted identities and with the accuracies of emotions recognition (unsupervised during training) to measure conservation of unlabeled intertwined features. These accuracies are obtained by training a classifier on the processed database.

It outputs the identification probability of each sample. Dropout layers with the same dropout parameters as in the autoencoder are used in both branches.

Results

As stated above, two methods are associated with this particular framework. The first one begins by applying the classical method, and the second one is using the spatial shift substitution method (for intertwined database). The first method produce qualitative results, whereas the second one produce more measurable results regarding the anonymization performance.

Qualitative approach: The first method (the one that does not restrict the identity of original samples used to train the discriminator) produce interesting results on the visual quality side but poor results on the quantified performance side. It was really interesting to observe the network making the output converge to a specific identity each time whereas there was no explicit constraint on it. The second method for intertwined data base (with restriction of identity for the discriminator's training set) results are shown on the Figures 4.4, 4.5, 4.6, 4.7. The Figures 4.4 and 4.5 show the evolution, over the training epochs, of the reconstructed image for respectively a training sample and a validation sample. These pictures display the successive epochs from left to right and top to bottom. The first epochs is the end of the pretraining phase: the autoencoder is just trained to reproduce exactly (pixel to pixel) its entry. Therefore and according to the performance of the pre-training the first image is the exact copy of original sample. The first two rows are where the most drastic changes occurs: it is the shifting from one identity into the chosen anonymized identity. The following epochs only refine details in output to make it more genuine Figure 4.6 shows couple of input images and their associated outputs for random emotions and random identities. The first column images are from the training set, the other columns are from the validation set. Figure 4.7 show in an organized manner all the different identities/emotions couples. The identity is the same in each column, the emotion is the same in each row. Figure 4.8 shows some of the few images where artifact remains.

Anonymity results The anonymity was provided by the substitution method. It used the same network as before with the discriminator training pool restriction variation (only the target identity samples in this pool). The results are in the table 4.1. It shows that some differences appear in the results of the anonymization task but the general trend is consistent.

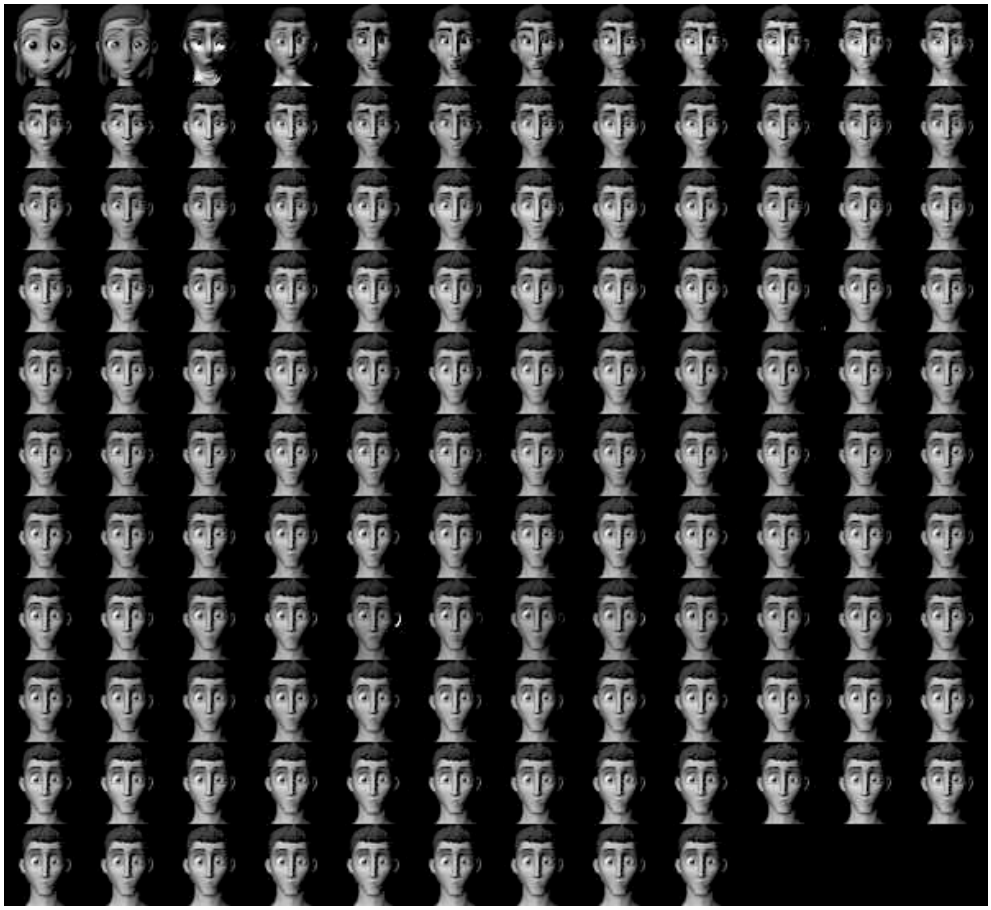


FIGURE 4.4: Evolution of the model produced image for each epoch for a training set sample. The input image is top left and going to the right and top down follows the increase of the training epochs by one unit increment. The resulting identity is fixed by the operator. The most impressive change occurs over the first epoch (firsts lines) and the other epochs refine the image.

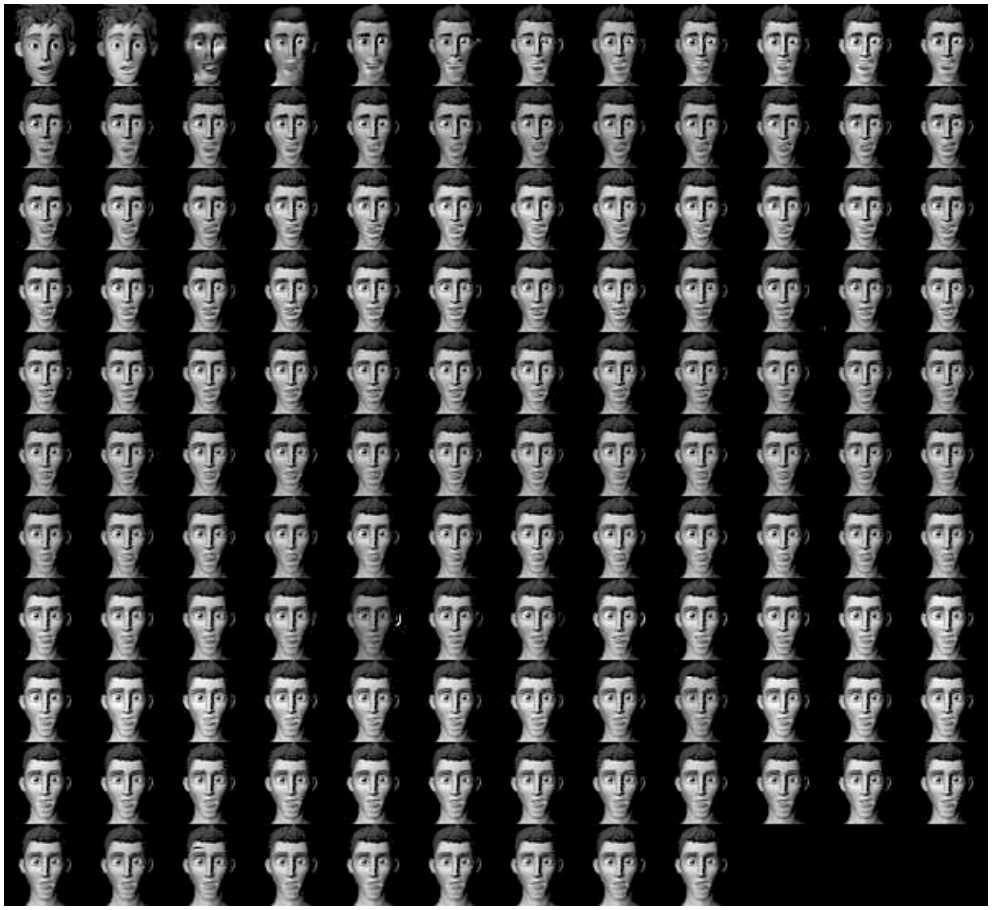


FIGURE 4.5: Example of the evolution of the model produced image for each epoch for a validation set sample. The input image is top left and going to the right and top down follows the increase of the training epochs by one unit increment. The resulting identity is fixed by the operator. The most impressive change occurs over the first epoch (firsts lines) and the other epochs refine the image.

FIGURE 4.6: Couple of images composed of the sample and the model output. The first columns are examples from the training set. The other columns are samples from the validation set.

FIGURE 4.7: Couple of images composed of the sample and the corresponding model output. All examples are from the test set. All identities and emotions are represented here. Same column couple represents the same identity with from top to bottom the following emotions: anger, disgust, fear, joy, neutral, sadness and surprise. Therefore for each line only one emotion is displayed for all identities 4.6

Remarks : One could argue that even if the database is anonymized, the chosen identity is still disclose because every other sample wear the face of one preexisting identity. This is not a problem at all considering one can artificially add a chosen virtual identity to be the template of the anonymized faces, while still using our framework. Indeed the fact that the second method allow to chose the final identity as one from the training set make possible the addition of a virtual identity that would be chosen as a destination for all the other identities to be transformed into.

4.3 Conclusion

We succeed in providing a more general usage of the previous framework by shifting from a fully supervised training where regular labels and private labels were used to a semi-supervised training where only the private labels are used. This shift was done while keeping the same spirits of the loss, but at the cost of a transformation of the architecture of the network and the training procedure.
We took advantage and reused widely known and efficient methods such as GAN discriminator and auto-encoder, and we managed to perform a smart combination of both to design a working architecture that was suited to our loss. The results of this chapter are made even more impressive because of the visual representation that we had, compared to the previous chapter representation that does not allow any direct visual representation. Indeed the vector $\boldsymbol{u}$ of the previous chapter has no visual structure constraint. It would be nonetheless possible to train a network to produce an image from the encoded representation of the previous chapter (See 6 in perspective work). The network we designed manages to affect the hidden representation of the data in the auto-encoder to shift private bit of information toward a database universal template with minimum cost to the other part of the information (this cost being the artifact shown above). The generalized information that we can deduce from this is that this method allows a spatial shift in a data structure adapted learnt space: shifting from identity to another without major loss of sub-cluster structure. The usage of the learnt discriminator allows a neat output image, pixel perfect in most cases. I have the strong

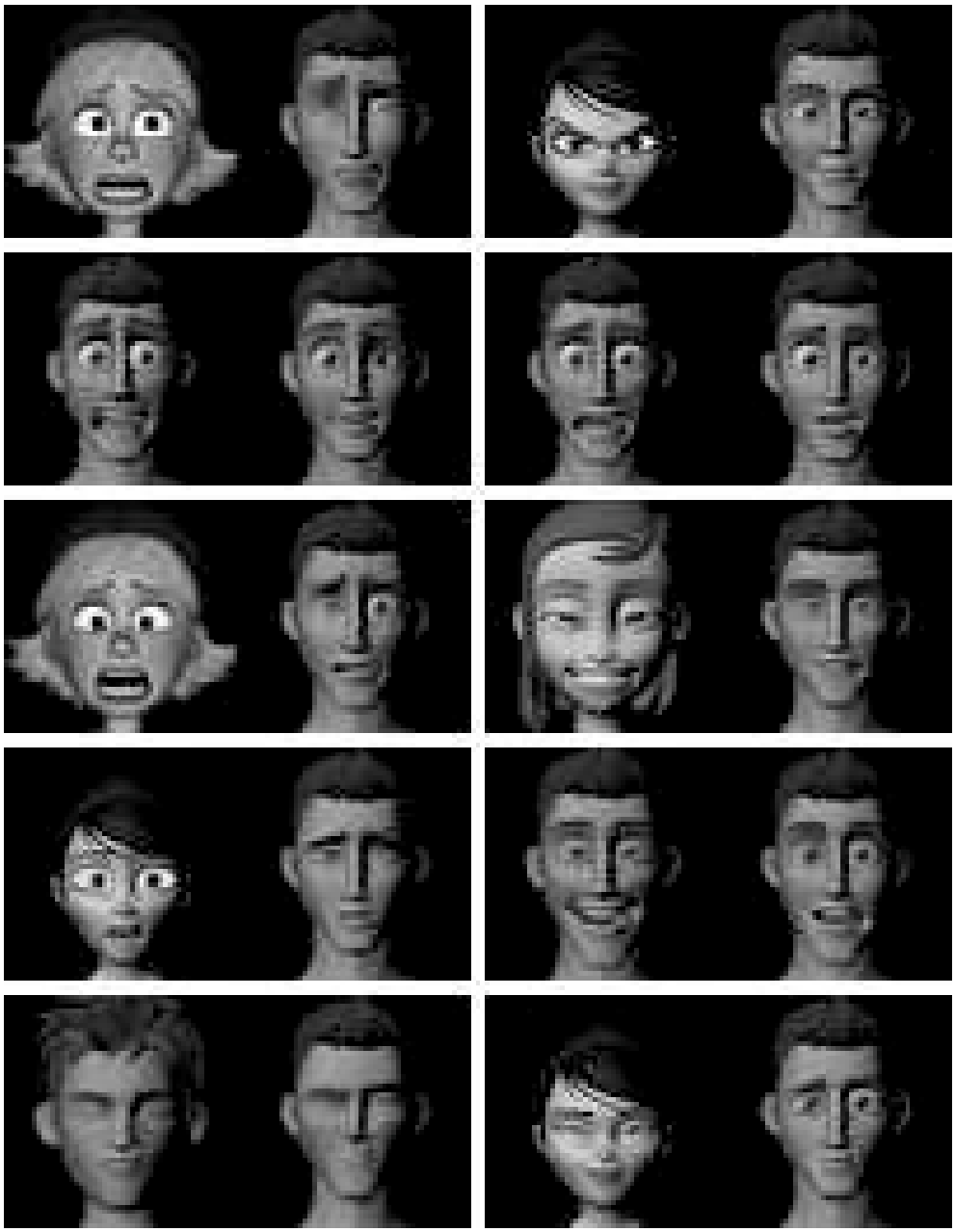


FIGURE 4.8: Couple of images composed of the sample and the corresponding model output. All examples are from the test set. We display here some selected images found while searching a hundred of randomly selected images of the test set. The first columns show pictures with artifacts: the main issue is one of the eye either not here (first line) or not fully formed (lines 2, 3 and 4). The last picture present or do not present an artifact on the right eye of the model. The input sample present closed eye (one more close that the other) and this is what the processed sample actually display, the number of pixels is not sufficient to be sure. The left column present other remarkable results selected by a human eye. The first line couple images display a mismatch of emotion and a difference of eyes symmetry on the generated model. Another difference of symmetry on the second line, but it does not affect the realism of the image. On the third line, the encoded image display almost closed eye with a small discrepancy between left and right eye. The last two lines show realistic encoded images with an human eye assessed shift of emotion.

feeling that a further work on this framework could produce a fully anonymizing model, without the need of the substitution.

Chapter 5

Dataset shift

Advancing machine learning methods require making Artificial Intelligence (AI) systems smarter and in particular, it requires preventing accidents, that is, guaranteeing that AI systems do what users actually want them to perform on specific tasks. There has been an increasing focus on safety research from the machine learning community, such as enhancing the robustness of Deep Neural Networks (DNNs) to change in the data distribution or to build soft-classifiers that indicate appropriate uncertainty when shown novel types of images, instead of confidently trying to use its potentially non-adapted learned model. Most of the time, the classifier takes X as an input and the predictor $\boldsymbol{p}_{Y|X}$ provides a vector containing the probabilities, with respect to a number of discrete labels Y, that the processed sample belongs to each classes which are not carefully examined: the decision function directly outputs the most likely class from the predictor which implies that the end user of the model may be unaware that the DNN is inadequately confident. It may especially be the case when the classifier is used to predict the appropriate label for other instances of X that are hence assumed to be drawn from the same distribution. This is a fundamental requirement for many real-world applications, therefore it is of great importance to monitor the model behavior in order to detect prediction discrepancy between test samples predictions and training samples predictions because the classifier is blind to dataset shift. For example, introducing a dog picture in a written digit classifier results in the DNN outputting a digit, the most likely one, even if this is a complete nonsense. As a matter of fact, this situation can be further critical due to the widespread of DNNs: non-specialist users are more and more frequent since it only takes a few lines of code from a public library to train and to implement a DNN. This chapter makes a step in that direction by proposing two 'black box' methods that works as 'dataset shift detector' via a careful examination of the label predictions. 'black box' here means that the method does not require any knowledge of the predictor's architecture. Experiments demonstrate accurate shift detection on different high-dimensional datasets of natural images. Note that our method only intends to perform detection, and does not reveal specific information about the origin of the model behavior shift.

5.1 Introduction

Deep learning with large enough labeled datasets has been highly successful in several applications such as image recognition, speech recognition, recommendation, machine translation [LeCun, Bengio, and Hinton, 2015] and more recently communications [Kim et al., 2018]. These methods implicitly assume the data of interest follow the same distribution as the one underlying training sequences. However, this is a very strong assumption since in many real-world applications such as communication networks the statistics of the data evolve over time. In most cases, we have very little or even no prior knowledge about how the test distribution may shift. It is thus necessary to build from scratch, with as little hypothesis as possible, a decision rule to try to detect such changes in order to take appropriate decisions (i.e. start a new training if it seems necessary).

This work addresses the problem of detecting changes of the model output distribution, from $\boldsymbol{p}_{\widehat{Y}|X}$ estimated from the training data to $\boldsymbol{p}_{\widehat{Y}|\widetilde{X}} \neq \boldsymbol{p}_{\widehat{Y}|X}$ estimated from the test data. To this end, we propose a simple yet effective method which is applicable to any trained deep neural soft-classifier $\boldsymbol{p}_{Y|X}$ describing the probability of the label Y given the features X. This method does not require joint training nor to access the details of the classifier itself, such as the intermediate representations or parameters. Our first approach relies mainly on the inference of the cumulative probability function of the likelihood predictor[1]:

$$F(r|p_X) \equiv \mathbb{E}_Y \left[\Pr \left(-\log p_{Y|X}(Y|X) \leq r \right) \right], \tag{5.1}$$

which can be used to relate behavior shift from $P_{\widehat{Y}|X}$ to $p_{\widehat{Y}|\widetilde{X}}$ based on the model misspecification $F(r|p_X)$ and $F(r|p_{\widetilde{X}})$ induced by the predictor $p_{\widehat{Y}|X}$. A chi-squared test [Lehmann and Romano, 2005] is then implemented to determine whether there is a significant deviation between the expected frequencies from the likelihood predictor during training and the frequencies observed from test data.

The main contribution in this method is a systematic method for checking prediction shift with 'black boxes'. We demonstrate the effectiveness of the proposed method using deep feed-forward neural networks, trained for image classification tasks on various well-known datasets including: CIFAR-10 [Krizhevsky, 2009], MNIST [Lecun et al., 1998], SVHN [Netzer et al., 2011] and Fashion-MNIST [Xiao, Rasul, and Vollgraf, 2017] which are widely used in the field (see chapter 2 for their precise content).

Related work

We wish to detect when some changes occur in the distribution of the prediction. The case where "some changes" where existing between training and test data sets was studied at various places, very often with an intuitive explanation, such as:

- "Concept shift" or "concept drift" where the idea of different data distributions is associated with changes in the class definitions (i.e. the "concept" to be learned) [Widmer and Kubat, 1996]
- "Changes of classification", where it is defined as "In the change mining problem, we have an old classifier, representing some previous knowledge about classification, and a new data set that has a changed class distribution." [Wang et al., 2003]
- "Changing environments", defined as "The fundamental assumption of supervised learning is that the joint probability distribution $p_{X,Y}(x, y)$ will remain unchanged between training and testing. There are, however, some mismatches that are likely to appear in practice." [Alaiz-Rodríguez and Japkowicz, 2008]
- "Fracture points" defined as "fracture points in predictive distributions and alteration to the feature space, where a fracture is considered as the points of failure in classifiers' predictions - deviations from the expected or the norm." [Cieslak and Chawla, 2009]

We shall consider the following definition, following [Moreno-Torres et al., 2012]: Dataset shift appears when training and test joint distributions are different. That is, when $p_{X,Y} \neq p_{\widetilde{X},Y}$

Intuition: Consider the task of classifying the images ($x_i \in \mathcal{X}$) containing handwritten digits ($y_i \in \mathcal{Y}$). Assume the classifier is able to output estimates $p_{\widehat{Y}|X}(\widehat{y}_i|x_i)$ of a given label for each input vector x_i. The "learning step" consists in computing for each x_i of the training set the corresponding class probabilities, and, since the true label

[1] The probability on X is determined by the distribution p_X, which is not a function of the labels.

y is known, one is also able to empirically estimate P_Y (the true label distribution). This allows the computation of the so-called confusion matrix, which component are $a_{i,j} = p_{\widehat{Y}|Y}(\widehat{y}_i|y_j)$, which characterizes the performance of the model on the training and/or validation sets.

Counter examples: First counter example, the situation of interest is different: a pneumonia predictor is trained in August. The training set consists of chest X-rays administered in the previous year (training distribution p_X) and the labels binary indicators obtained via the diagnosis of a physician. The model f is trained to predict pneumonia given an X-ray image. Assume that in the training set 0.1% of patients have pneumonia. We deploy f in the clinic and for several months, it reliably predicts roughly 0.1% positive while used on $p_{\widetilde{X}}$ the test set. Running f on January's data, 5% of patients are predicted to have pneumonia. Because f remains fixed, the shift must owe to a change from $p_{\widetilde{X}} = p_X$ to $p_{\widetilde{X}} \neq p_X$. Several questions arise form this situation: since the statistics of X changed, is f still accurate? What is the real current rate of pneumonia? Is the classifier, trained under an obsolete prior, underestimating pneumonia? The real prevalence may be greater than 5%. The model of this situation is as follows: the conditional probability of the pneumonia $\boldsymbol{p}_{Y|X}$ certainly did not change (a reasonable approximation), but the distribution of radiologic findings has changed, $p_{\widetilde{X}} \neq p_X$.

This is denoted as covariate shift: $p_{Y|X} = p_{Y|\widetilde{X}}$ but $p_{\widetilde{X}} \neq p_X$.

Second counter example with the same setup. The physician realizes that some patients have strong cough and does not enter this information in the model. Obviously, during an epidemic, $p_{pneumonia|cough}$ may go up substantially, and you have a prior information on p_Y for the winter measurements. The (coarse) model of this situation is as follows: the manifestation of the disease of the pneumonia $p_{X|Y}$ did not change (a "reasonable" approximation), but the distribution of the decisions p_Y has changed.

This is denoted as label shift: $p_{\widetilde{X}|Y} = p_{X|Y}$ but $p_{\widetilde{Y}} \neq p_Y$.

Other possible scenario: *Sample selection bias*: the discrepancy in distribution is due to the fact that the training examples have been obtained through a biased method, and thus do not represent reliably the operating environment where the classifier is to be deployed (which, in machine learning terms, would constitute the test set).
Suppose we wish to generate a model to diagnose breast cancer. Suppose, moreover, that most women who participate in the breast screening test are middle-aged and likely to have attended the screening in the preceding 3 years. Consequently the sample includes mostly older women and those who have low risk of breast cancer because they have been tested before. This problem is referred to as sample selection bias: the examples in the training set do not reflect the general population with respect to age (which amounts to a bias in p_X) and they only contain very few diseased cases (i.e. a bias in $\boldsymbol{p}_{Y|X}$).

The last two examples are much more difficult to model and to address:

Non-stationary environments. It appears when the training environment is different from the test one, whether it is due to a temporal or a spatial change.
In real-world applications, it is often the case that the data is not (time- or space-) stationary. The signals of interest may have a drift, and it may be useful to detect the drift to launch a new training.

Spam filtering and network intrusion detection. The non-stationary environment is due to an adversary that tries to work around the existing classifier's learned concepts. In terms of the machine learning task, this adversary warps the test set so that it becomes different from the training set, thus introducing any possible kind of dataset shift. Detecting such intrusion and outliers would be very beneficial.

"Reliability" evaluation and the "black box" approaches: Many published works assume that they have a full set of training and validation samples, and also a full set of test samples, so that they can compute a statistical characterization of the test signals statistics or of the consequences of these changes on the output statistics. Having access to that much information, they also try also to compensate for the changes in the situation. We outline below two approaches, one is succeeding in working with very small test sets, but requires a re-design of the training algorithm and access to the information that lies inside the model (i.e. each layer outputs) , and another one is a full "black box" approach, but requires many test samples.

"Mahalanobis distance" for detecting out of distribution samples [Lee et al., 2018]. The target of this paper: detect Out of Distribution as well as adversarial samples. Main idea: measure the probability density of test sample on feature spaces of DNNs utilizing the concept of a "generative" (distance-based) classifier. Specifically, pre-trained features are fitted by a class-conditional Gaussian distribution and it is shown that its posterior distributions are equivalent to the softmax classifier under Gaussian discriminant analysis. Under this assumption, they define the confidence score using the Mahalanobis distance with respect to the closest class-conditional distribution, where its parameters are chosen as empirical class means and tied empirical covariance of training samples. A pre-trained softmax neural classifier is given: $p_{Y|X}(y=c|x) = \frac{\exp(W_c^T f(x) + b_c)}{\sum_{c'}(\exp(W_{c'}^T f(x) + b_{c'})}$ where W_c and b_c are the weight and the bias of the softmax classifier for class c, and $f(\cdot)$ denotes the function that maps the sample to the output of the penultimate layer of DNNs. The method heavily relies on three items:

i) derivation of generative classifiers from softmax ones. Define C class-conditional Gaussian distributions with a tied covariance Σ: $p_{\widehat{Y}|Y}(f(x)|y=c) = N(f(x)|\mu_c, \Sigma)$, where μ_c is the mean of multivariate Gaussian distribution of class $c \in \{1, ..., C\}$. The pre-trained features of the softmax neural classifier $f(x)$ have been shown to follow the class-conditional Gaussian distributions. (mean and covariance are empirical)
ii) Mahalanobis distance-based confidence score: $M(x) = \max_c -(f(x) - \mu_c)^T \Sigma (f(x) - \mu_c)$ which corresponds to the logarithm of the probability density function (PDF) of the test sample.
iii) Confidence scores computed from all layers in DNNs and integrated by weighted averaging: $\sum_l \alpha_l M_l(x)$, where $M_l(x)$ and α_l are respectively the confidence score and weight at $l-$th layer.
A threshold-based detector measures some confidence score of the test sample, and classifies it as in-distribution if the confidence score is above some threshold. The paper reports implementation with many layers (34 or 100) on various data sets (many layers are necessary because some layers may not carry a lot of information, and to avoid overconfident decisions). They tune all hyper parameters (many) on a validation set. Results are impressive for detecting out of distribution or adversarial samples as well as for class-incremental learning.

'black box' predictors for detecting and correcting label shift by [Lipton, Wang, and Smola, 2018]. Label Shift: $\boldsymbol{p}_{\widetilde{X}|Y} = \boldsymbol{p}_{X|Y}$, and $p_{\widetilde{Y}} \neq p_Y$ This implies $\boldsymbol{p}_{\widehat{Y}|\widetilde{Y}} = \boldsymbol{p}_{\widehat{Y}|Y}$ Computation based on the confusion matrix:

$$p_{\widehat{Y}} = \sum_{y \in \mathcal{Y}} \boldsymbol{p}_{\widehat{Y}|\widetilde{Y}}(\widehat{Y}|\widetilde{Y}=y) p_{\widetilde{Y}}(\widetilde{Y}=y) \tag{5.2}$$

$$= \sum_{y \in \mathcal{Y}} \boldsymbol{p}_{\widehat{Y}|Y}(\widehat{Y}|Y=y) p_{\widetilde{Y}}(\widetilde{Y}=y) \tag{5.3}$$

$$= \sum_{y \in \mathcal{Y}} \boldsymbol{p}_{\widehat{Y},Y}(\widehat{Y}, Y=y) \frac{p_{\widetilde{Y}}(\widetilde{Y}=y)}{p_Y(Y=y)}. \tag{5.4}$$

This is a linear system of equations, in which one can estimate empirically the confusion matrix on the training data, and $p_{\widehat{\widetilde{Y}}}$ is estimated by the unlabeled test data. The system

can therefore be solved in $w(y) = \frac{p_{\widetilde{Y}}(\widetilde{Y}=y)}{p_Y(Y=y)}$, providing $\widehat{w}(y)$ Then one can obtain the following estimates, based on an empirical evaluation of the corresponding quantities (confusion matrix and probabilities of the labels) in vector form: $\widehat{\boldsymbol{W}}(Y) = \boldsymbol{C}^{-1}_{\widehat{Y},\widetilde{Y}}\widehat{\boldsymbol{p}}_{\widehat{Y}}$ and $\widehat{\boldsymbol{p}}_{\widehat{Y}} = \widehat{\boldsymbol{p}}_{\widetilde{Y}}\widehat{\boldsymbol{W}}(Y)$.

In other words, the authors are correcting the empirical estimates of $p_{\widetilde{Y}}$ by the knowledge of the probabilities of error. The authors prove that these are consistent estimators, and validate the computation under various label shift experiments.

Various other approaches:

- [Shimodaira, 2000] approach, in which both training and test distributions share the same conditional distribution $p_{Y|X}$, while their marginal distributions, p_X and $p_{\widetilde{X}}$, are different. The focus was on correcting the distribution shift while here we concentrate on the detection of this shift.
  Major efforts have been dedicated to importance reweighing (see [Sugiyama and Kawanabe, 2012] and references therein). These methods have been developed for detecting either out-of-distribution or adversarial samples, or both.

- An evaluation of the confidence one can have in a classifier prediction has been studied in [Jiang et al., 2018].

- [Lee et al., 2018] (outlined above)- Detection of abnormal samples based on a Mahalanobis distance on intermediate representations. It outperforms [Hendrycks and Gimpel, 2017], [Liang, Li, and Srikant, 2018] based on the maximum probability of the softmax output. Although this method has been shown to improve significantly the detection performance, this approach comes in contradiction with the 'black box' assumption considered in this paper. Indeed the method relies on measuring and combining the final features of the neural network with low level features.

- While [Hendrycks and Gimpel, 2017; Liang, Li, and Srikant, 2018] methods are based on the sole maximum probability at the softmax output, we prove here that considering the complete softmax output is beneficial for the detection of covariate shift. [Hendrycks and Gimpel, 2017] adopts a 'black box' approach using only softmax outputs, they use labels to separate correctly and incorrectly classified *test* examples, our work only uses the softmax output without the labels.

- [Liang, Li, and Srikant, 2018] work is based on the softmax output but they used perturbations of the input as a preprocessing for their method ODIN, while the method proposed in the present work relies on a regular use of the soft probabilities: our method can be used while the model is actively used for classification. Note that our method is designed to apply to already trained 'black box' model.

- Recent work from [Lee et al., 2017] introduce a new term in the training loss so that the Kullback-Leibler divergence (KL) between the models' output for out-of-distribution samples and the uniform distribution is minimized. This method produce interesting results but it cannot be applied to a already trained 'black box' model.

- [Taigman et al., 2015, Section 4], introduces L2-norm, and links it to entropy, to reject mismatched samples from classifier softmax outputs. [Shalev, Adi, and Keshet, 2018] work rely on K-embeddings to improve quality prediction and use L2-norm over predicted embedded vector to detect out-of-distribution inputs on such embedded vectors. As opposed to this work, these two papers focus on a single sample at a time, not exploring a sequential approach.

We prove here that, despite the common belief stated at several places, considering the complete softmax output is beneficial for the detection of under-confident behavior in a sequential processing. While most of the research efforts have been towards improving classifier performance and compensating for covariate shift, detecting the presence of distribution shift using unlabeled samples in a sequence framework has received less attention.

5.2 First approach

The first approach proposed here is a first work toward the monitoring of classifier behavior.

The rest of this section is organized as follows: firstly we formulate the problem, secondly we introduce Pearson's chi-squared test which is used to detect the dataset shift. Afterward simulations results are presented. We finish with concluding remarks.

5.2.1 Problem formulation

Consider a pre-trained deep neural network with a softmax classifier denoted by $\boldsymbol{p}_{Y|X}$. Let $x \in \mathcal{X}$ denote a feature input and let $y \in \mathcal{Y} \equiv \{0, 1, \ldots, C-1\}$ denote the corresponding output or label. The softmax classifier provides as an output the probability $p_{Y|X}(c|x)$ for each class label c given the feature x. Without loss of generality, the network is assumed to perform an image classification task. Therefore, the image can be interpreted as being a redundant representation of the class and the image classification problem could be seen as a data compression process.

A binary instantaneous code of minimum expected-length could be constructed to describe the source with optimal coding length: $-\log_2 p_{Y|X}(c|x)$. Intuition behind this work is that the statistical distribution of the code lengths should reflect to some extent the statistical distribution of the inputs. In the following, the empirical cumulative distribution function $\widehat{F}_n(r|\texttt{Data})$ of $-\log_2 \boldsymbol{p}_{Y|X}$:

$$\widehat{F}_n(r|\texttt{Data}) \equiv \mathbb{E}_Y \left[\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{[0,r]}(Y|x_i) \right], \tag{5.5}$$

where $\mathbb{I}_{[0,r]}(y|x_i) = 1$ if $-\log_2 p_{Y|X}(y|x_i) \leq r$ and zero otherwise, is used in order to detect data shift. Obviously, assuming that the data samples come from distribution $\boldsymbol{p}_X$, by the Glivenko-Cantelli theorem [Billingsley, 1986], it follows that

$$\sup_r \left| \widehat{F}_n(r|\texttt{Data}) - F(r|p_X) \right| \longrightarrow 0, \tag{5.6}$$

almost surely in the limit when n goes to infinity. Kolmogorov strengthened this result, by effectively providing the rate of this convergence. However, in practice, the statistic requires a relatively large number of data points (in comparison to other goodness of fit criteria) to properly reject the null hypothesis. This will be made clear in the following example.

Assume that detection is based on batches of images, which is indeed the case for many practical situations. Consider the empirical cumulative distribution function plotted in Fig. 5.1 for the case of CIFAR-10 based on the neural network specified in Algorithm 3 relegated to appendix B.1. Two different variants are studied: $\widehat{F}_n(r|\texttt{Matched test})$ which is the restriction of the predictor's likelihood to feature inputs x belonging to the test set and $\widehat{F}_n(r|\texttt{Mismatched data})$, which is obtained by adding salt and pepper noise to `Matched test` with corruption probability 0.08. We use 10^4 input images and the CIFAR-10 dataset consists of 10 classes. The size of each set is thus equal to 10^5.

It is observed in Fig. 5.1 that the range is almost the same for both sets. Whereas an hypothesis test based on the comparison of the value of the components of $-\log_2 \boldsymbol{p}_{Y|X}$

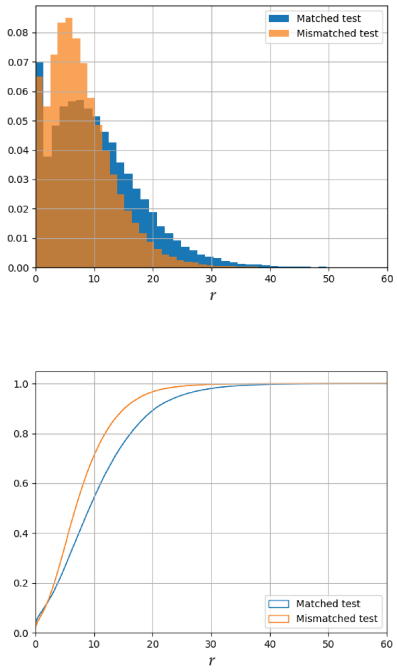


FIGURE 5.1: Empirical distribution function (top) and empirical cumulative distribution function (bottom) $\widehat{F}_n(r|\texttt{Matched test})$ and $\widehat{F}_n(r|\texttt{Mismatched test})$, where `Matched test`$\equiv$ CIFAR-10 and `Mismatched test`$\equiv$ CIFAR-10+S&P.

to a threshold would not be relevant in this case. In contrast, a test of homogeneity seems to be more appropriate here since the distributions exhibit different shapes. This test will be introduced in the next subsection.

5.2.2 Pearson's chi-squared test

The method proposed here to detect a possible evolution of the data set is based on a chi-Square test for homogeneity [Lehmann and Romano, 2005]. This test compares the empirical distribution by counting occurrences sharing a close value of a categorical variable, i.e., the variable $-\log_2 \boldsymbol{p}_{Y|X}$ is used here by making a partition of its support and the test count the empirical occurrences in each of the partition's set. The test determines whether frequency counts are distributed identically across different populations. Two groups are considered here. The first group, called `baseline (BAS)` serves as a proper benchmark. `baseline` contains realizations obtained by sampling jointly the likelihood predictor function: $-\log_2 \boldsymbol{p}_{Y|X}$, according to the label distribution p_Y and the feature samples X from input images, e.g., images from the validation set. The second group called `To-be-tested (TBT)` contains $C \times N^{\text{TBT}}$ realizations of $-\log_2 \boldsymbol{p}_{Y|X}$, where C is the number of classes and N^{TBT} is the number of sample in the TBT set. Obviously, we want an efficient test for small values of N^{TBT}. However, such minimal value will heavily depend on the statistical difference between the underlying distributions. The test procedure is described below.

Testing homogeneity

We return to the simple goodness of fit problem for categorical data that was briefly considered. Suppose that each sample $-\log_2 \boldsymbol{p}_{Y|X}$ is divided into $|K|$ categories (e.g. defining $|K|$ real valued intervals). We would like to evaluate whether the distribution of categories in each group is the same. Let us denote the empirical estimates of the probabilities as $\Pr(K = k|\texttt{BAS}) \approx \widehat{p}_k^{\text{BAS}}$ and $\Pr(K = k|\texttt{TBT}) \approx \widehat{p}_k^{\text{TBT}}$ and the corresponding probabilities in the population from which the sample groups are selected as p_k^{BAS}

and p_k^{TBT}. Then we want to test the following hypotheses:

$$\begin{cases} \text{H0} & : \quad p_k^{\text{BAS}} = p_k^{\text{TBT}} = p_k, \\ \text{H1} & : \quad \text{otherwise.} \end{cases} \tag{5.7}$$

A standard test, proposed by Pearson (1900), rejects H0 for large values of Pearson's Chi-squared statistic:

$$T \equiv \sum_{k=1}^{|K|} \frac{(N_{k,th}^{\text{BAS}} - N_k^{\text{BAS}})^2}{N_{k,th}^{\text{BAS}}} + \frac{(N_{k,th}^{\text{TBT}} - N_k^{\text{TBT}})^2}{N_{k,th}^{\text{TBT}}}, \tag{5.8}$$

where N_k^{BAS} and N_k^{TBT} stand for the number of instances in the k-th category within each sample group, i.e., the evidence $N_k^{\text{BAS}} = \widehat{p}_k^{\text{BAS}} N^{\text{TBT}}$ and $N_k^{\text{TBT}} = \widehat{p}_k^{\text{BAS}} N^{\text{BAS}}$. The theoretical numbers of instances are computed as follows: $N_{k,th}^{\text{BAS}} \equiv p_k N^{\text{BAS}}$ and $N_{k,th}^{\text{TBT}} \equiv p_k N^{\text{TBT}}$. Since the true probabilities $\{p_k\}$ are unknown, these are estimated as: $p_k \approx \frac{N_k^{\text{BAS}} + N_k^{\text{TBT}}}{C(N^{\text{BAS}} + N^{\text{SMP}})}$ which is the best estimator based on the observations and under the assumption that H0 holds. The total number of categories is chosen to be 10, with intervals of equal length. If either $N_{k,th}^{\text{BAS}} < 5$ or $N_{k,th}^{\text{TBT}} < 5$ then the k-th category is grouped with an adjacent category and thus $|K| \leq 10$.

Dataset shift detection

As previously mentioned, `Baseline (BAS)` serves as a benchmark and characterizes the behavior of the pre-trained network with respect to the likelihood's predictor when input features and random labels are used. Therefore, a simple way to construct `Baseline` is to aggregate the softmax outputs obtained with the validation set. `To-be-tested (TBT)` is obtained by randomly sampling the likelihood's predictor either from `Matched test` feature inputs, i.e., H0 holds, or from `Mismatched test` feature inputs, i.e., H1 holds and H0 should be rejected. For that purpose, T should be compared to the critical value. The critical value of the test is chosen according to the desired level of confidence and can be estimated from expression (5.5) according to the `Baseline` feature inputs. The efficiency of this test is usually measured through the two error probabilities and for several critical values:

$$\begin{cases} \alpha(T) & \equiv \quad \Pr\left(\text{reject H0} \,|\, \text{TBT} \equiv \texttt{Matched test}\right), \\ \beta(T) & \equiv \quad \Pr\left(\text{accept H0} \,|\, \text{TBT} \equiv \texttt{Mismatched test}\right). \end{cases}$$

5.2.3 Simulation results

Noisy data detection

In this subsection, the mismatched test data is modelled by an additive noise on background of the proper data [Vincent et al., 2010]. Namely, `Mismatched test` $= \{x + n, x \in$ `Matched test`$\}$ where n is a background noise with uniform distribution between 0 and $U_{max} = 30$ on a grey scale image (i.e. pixels values are in $[0, 255]$). In the following, we use two datasets presented in Chapter 2, namely MNIST dataset and Fashion-MNIST dateset. The two datasets present a uniform background that can be affected with this additive noise. Both training sets were split such that 10% of the set is used as a validation set.

Numerical results are given in Table 5.1 and 5.2. The true negative rate (TNR) stands for the proportion of mismatched sequences that are properly identified, true positive rate (TPR) stands for proportion of matched sequences that are properly identified. AUROC stands for Area under Receiver Operating Characteristic and is computed from the area under the curves plotted on Figures 5.2 and 5.3.

The main trend in the results follows the intuition: the larger N^{TBT} is, i.e. the longer sequences are, the better are the performances. The detection accuracy is over 95% with $N^{\text{TBT}} = 40$ on MNIST database. The results are even better with Fashion-MNIST

database, the main reason for the differences is the nature of the network being different. The network used for MNIST has a simple feed-forward architecture with one layer and dropout. The accuracy of the classification task is 98.02% on the test set (matched sequence) and 96.75% on the noisy version of the test set (mismatched sequence) The Fashion-MNIST network involved a deeper and more complex architecture based on convolutional layers with dropout, details are in the appendix (see appendix B.1).The accuracy on the classification task is 91.40% on the matched sequence (test set) and 90.30% on the mismatched sequence (noisy test set). The 95% detection accuracy (detecting true matched sequence and true mismatched sequence) is reached with only $N^{\text{TBT}} = 20$ on this database. The method is efficient to separate target data and mismatched data even when the difference between them is a rather small disturbance. These results prove the efficiency of our method, but also demonstrate that it can be applied to any neural network outputs regardless of their architecture complexity, truly behaving as a 'black box' dataset shift detector.

N^{TBT}	TNR at TPR 95%	AUROC	Detection accuracy
10	48.82	83.93	80.24
20	75.96	94.40	89.05
30	90.88	97.82	93.80
40	95.76	99.04	95.77
50	98.6	99.62	97.81
60	99.58	99.88	98.79
70	99.88	99.97	99.34

TABLE 5.1: Results on the MNIST dataset for several values of N^{TBT} when mismatched is additive background noise

N^{TBT}	TNR at TPR 95%	AUROC	Detection accuracy
10	52.76	88.74	84.87
20	92.72	98.40	95.09
30	97.62	99.40	96.87
40	99.6	99.80	98.45
50	99.78	99.93	99.01
60	99.88	99.95	99.20
70	> 99.96	99.996	99.71

TABLE 5.2: Results on the fashion-MNIST dataset for several values of N^{TBT} when mismatched data is additive background noise

Plots of Figures 5.2 and 5.3 show the accuracy trade-off for different values of N^{TBT} respectively for MNIST and Fashion-MNIST. The performances shown here are quite good even with a small sequence ($N^{\text{TBT}} \leq 20$). For the higher value of N^{TBT} the detection of mismatched data can be made with high confidence (the probability is over 0.9995). It seems that for a wide range of applications one can still find the proper N^{TBT} value to satisfy its needs.

Out-of-distribution samples detection

We turn now to the detection of out-of-distribution samples. These abnormal samples can be due to an adversarial attack or to an error of manipulation. In this context, the images in `Mismatched test` and in `Matched test` are significantly different therefore the test of homogeneity is expected to be efficient for smaller values of N^{TBT} than in the previous context. The dataset we are using in this sub-subsection are CIFAR-10 and SVHN, both presented in chapter 2. The network used is a deep convolutional network

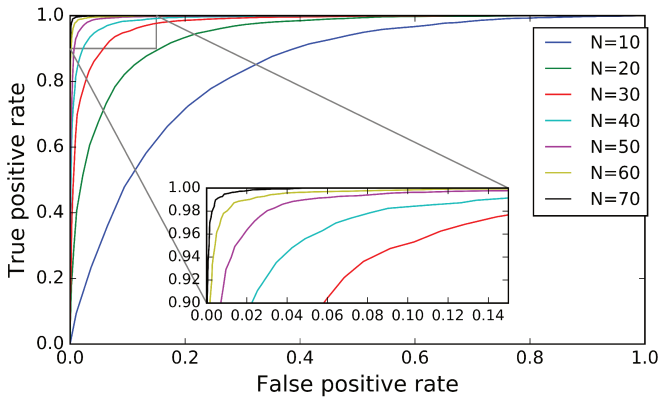


FIGURE 5.2: Accuracy trade-off for different length of sequence N^{TBT} on MNIST: true positive rate (correctly identifying normal samples) as a function of false positive rate (missing the detection of abnormal samples).

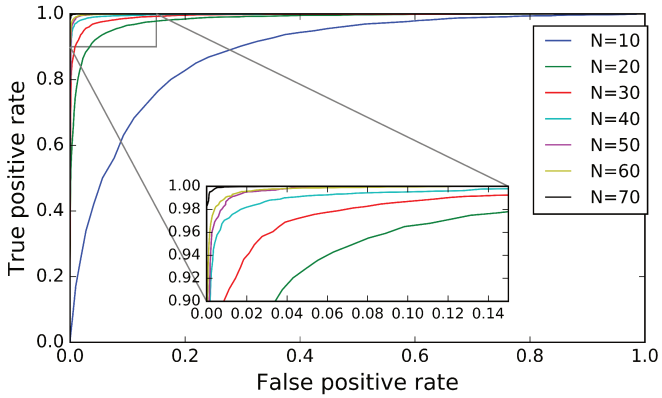


FIGURE 5.3: Accuracy trade-off for different length of sequence N^{TBT} on Fashion-MNIST: true positive rate (correctly identifying normal samples) as a function of false positive rate (missing the detection of abnormal samples).

(detailed in appendix B.1) trained on samples from one database and the mismatched test are performed with sample from the other database. The classification accuracies are 73.60% and 91.64% on respectively the CIFAR-10 dataset and the SVHN dataset.

N^{TBT}	TNR at TPR 95%	AUROC	Detection accuracy
5	86.22	95.68	91.34
7	95.88	98.48	95.88
9	98.28	99.39	97.71
11	99.34	99.74	98.47
13	99.64	99.80	99.52
15	99.98	99.99	99.81
17	>99.99	>99.99	99.87

TABLE 5.3: Results on a SVHN trained network tested on CIFAR-10 samples for several values of N^{TBT}

The error probabilities are given in Tables 5.3 and 5.4. In all studied cases, the method proves its efficiency. Indeed, with 7 or 15 images (respectively when CIFAR-10 samples and SVHN samples are the mismatched data) it is possible to take apart matched from mismatched samples with an accuracy over 95%. One can observe that the performance is quite different in both situations of a network trained on SVHN, tested on CIFAR-10, and the converse situation. This has a simple explanation: If a

N^{TBT}	TNR at TPR 95%	AUROC	Detection accuracy
5	27.14	86.69	82.47
7	55.26	92.96	88.19
9	84.8	96.80	92.72
11	88.28	97.49	93.75
13	91.26	97.84	94.62
15	98.58	99.19	96.96
17	98.72	99.28	97.06
19	99.44	99.45	97.55

TABLE 5.4: Results on a CIFAR-10 trained network tested on SVHN samples for several values of N^{TBT}

network has poor performance on the training set, it will be difficult to distinguish 'legitimate' signals from 'illegitimate ones'. And this is the case here: the network trained on SVHN has a performance of 91 %, while this performance decreases to 73 % when the network is trained on CIFAR-10. The curves on the Figures 5.4 and 5.5 look less smooth because N^{TBT} is small compared to the value used for the Figures 5.2 and 5.3. They may overlap because they are plotted with only a step of two between values of N^{TBT}. Note that the hypothesis are only valid if $N^{\text{TBT}} \geq 5$. As shown on Figure 5.4, the case $N^{\text{TBT}} = 5$ can produce surprising shaped curve but it still brings relevant information. The method is efficient to track out-of-distribution sample sequences quickly and with a strong confidence even if the network does not perform perfectly on the regular task: the CIFAR-10 trained network output statistics, despite not coming from top of the art classifier, still enable an accurate detection with as few as $N^{\text{TBT}} = 15$. Finally as shown on the last two lines of table 5.3 top performances are achieved for quite small value of N^{TBT}: the difference between $N^{\text{TBT}} = 15$ and $N^{\text{TBB}} = 17$ (reinforced by their respective curves in Figure 5.4) is faint which makes sense because the performances are close to 100%

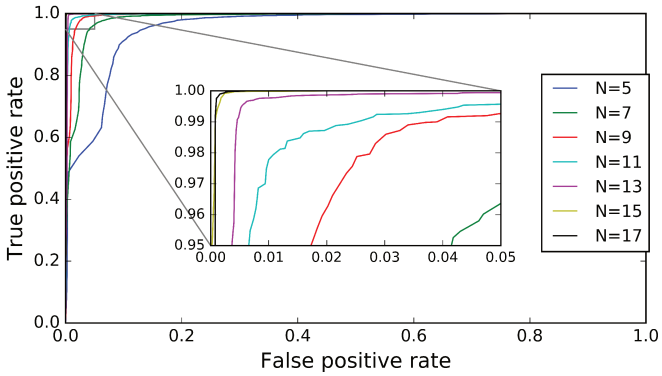


FIGURE 5.4: Accuracy trade-off for different length of sequence N on a network trained on SVHN and tested with CIFAR-10 images: true positive rate (correctly identifying normal samples) as a function of false positive rate (missing the detection of abnormal samples).

5.2.4 Conclusion

A test of homogeneity has been proposed for the detection of either data evolution or of abnormal samples in a classification context. The method is based on the assumption that the output distribution of the neural network's softmax classifier reflects the statistical properties of the input, and can serve as a basis for evaluating some "distance" between the statistical properties of the signals/images under test compared to the reference situation, given by the training sequences. This approach has been validated

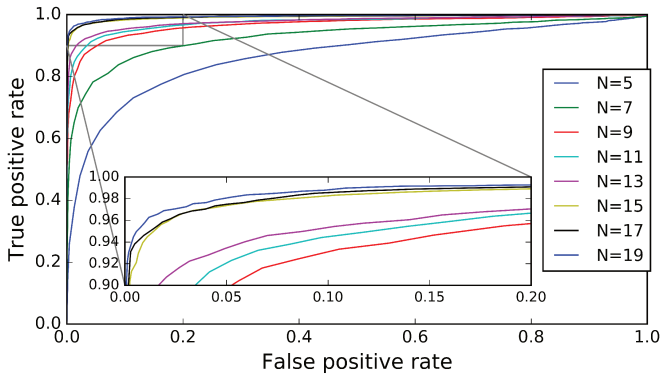


FIGURE 5.5: Accuracy trade-off for different length of sequence N on a network trained on CIFAR-10 and tested with SVHN images: true positive rate (correctly identifying normal samples) as a function of false positive rate (missing the detection of abnormal samples).

through simulations, demonstrating that good homogeneity checks can be obtained even for small number of observations. The method is very simple and is applied independently of the intrinsic neural network architecture. In fact, the neural network is considered as a 'black box' since the method necessitates the sole knowledge of the outputs. This comes in contrast with most solutions in the literature.

5.3 Second approach

The second approach uses a different framework that results into simple tools used to extract model output information (i.e. model output being the soft probabilities). This straightforward method has the advantages of producing better results.

This section is organized as follows. Firstly we introduce the statistical tools and the testing setup, then we present the simulations results and at last we provide a summary with concluding remarks, additional results are presented in the appendix B.

Learning with dataset shift is a major challenge in real-world non-stationary environments where the input data distribution may vary over time. Moreover, with the recent booming of ML applications, more and more non specialist people are using these algorithms as 'black boxes'. Therefore, there is a need for continuous monitoring of the behaviour of those systems and for tracking the "amount" of shift in order to be able to decide about retraining and/or adaptation in a timely manner. This paper introduces a novel 'black box' method for detecting dataset shift based only on the (soft) output of the classifier. The method does not require any prior assumption on the label or the structure of the network to be able to inform the user quickly about a mis-adaptation of the network to the current data. In fact, our method measures in an appropriate way the confidence of the classifier and detects under-confident behaviour of the network. Empirical results show that it reacts well to various dataset shifts of natural images.

5.3.1 Introduction

In this section, we investigate simple but still effective 'black box' methods (i.e., based only on the soft output probabilities) for detecting and distribution shift. This is done in such a way that the end user can have information about the reliability of the obtained result and thus, he takes appropriate actions depending on its target. Our contributions are as follows:

- We propose a 'black box' method for sequence detection of under-confident behavior in a sequential processing, using only the soft-classifier without any other information on the DNN than the probability vector for each feature input.

- We derive a formal relationship between the uncertainty of the classifier and two relevant statistics: the "standard deviation" (SD) and "geometric mean" (GM) methods which motivate the use of new statistics to measure classifier uncertainty.
- We demonstrate the effectiveness of our approach using Deep Convolutional Neural Networks (DCNNs), trained for image classification tasks on well-known datasets.

The derived tool is intended to be added on top of a working in use classifier which has already been trained and without altering inputs or outputs. Faced with distribution shift, our goals are as follows: (i) detect when distribution shift occurs from as a few examples as possible and without labels; (ii) quantify the amount of uncertainty. The proposed method applies to any arbitrary classifier.

5.3.2 Uncertainty Based-Methods for Detecting Confidence

The key idea is that we are not interested in the network classification rate, we only assume that the network is fairly trained on dataset's classification task. What we are interested in is the confidence of the predictions expressed by the network on the data we feed it with.

Definitions and presentation of tool

Given a set of M classes $Y \in \mathcal{Y}$ and a feature vector $X \in \mathcal{X}$, DNNs usually output soft-probabilities $\boldsymbol{p}_{\widehat{Y}|X}$. We assume that we do not have access to the DNN but only to its output soft probability, i.e., a 'black box' approach relying solely on $P_{\widehat{Y}|X}$. Most of the time, the DNN directly outputs the most likely class estimator $f(x) := \arg\max_{y\in\mathcal{Y}} p_{\widehat{Y}|X}(y|x)$. In this framework, a change in the distribution of X from p_X to $p_{\widetilde{X}}$ should result in a change of the conditional distribution provided by the DNN from $\boldsymbol{p}_{\widehat{Y}|X}$ to $\boldsymbol{p}_{\widehat{Y}|\widetilde{X}}$, except for adversarial attacks which is out of the scope of this paper.

We assume that samples from p_{XY} are available for training (e.g., training and validation sets). However, we do not have any information about how the distribution p_X will evolve or shift to $p_{\widetilde{X}}$, neither the true nor samples from $p_{\widetilde{X}}$. Our main goal is to develop a method that can provide a meaningful statistic for detecting the shift of the dataset. To this end, the samples are grouped into batches of size N. The decision rule should use only the soft probabilities output by the DNN: $\boldsymbol{p}_{\widehat{Y}|X}(\cdot|x_i)$. The test statistics – to be defined in the next paragraph – are computed based on the components of $\boldsymbol{p}_{\widehat{Y}|X}(\widehat{y}|x_i)$ for x_i such that $i \in [0, N-1]$ within the considered batch. Accepting the null hypothesis (H_0) implies that the batch is considered as in-distribution p_X. The type I error indicates that the hypothesis H_0 is rejected whereas it was true. On the other hand, the error of type II implies that the hypothesis H_1 is rejected while it was true.

While it is very intuitive that the conditional probabilities should contain information about the reliability of the result, the precise test can take various forms, and we provide below two approaches that will be compared in the experimental section. A first approach is based on a precise definition of the uncertainty of the test, also linked to conditional Shannon entropy (see [Cover and Thomas, 2006]) of the sequence. Entropy can easily be interpreted as a measure of confidence: if one of the labels is highly probable, its probability will be dominant, and the entropy small. Conversely, if the network is mis-adapted, the entropy will become larger. Beside the Shannon entropy, another efficient characterization of the behavior of a test can be obtained via the so-called confusion matrix $\boldsymbol{p}_{\widehat{Y}|Y}$, measuring the probabilities that a given label is predicted as any other one. Given an output of the classification algorithm, the comparison with the corresponding column of the confusion matrix should provide useful information. The sub-subsection formalize these connections between the confidence of a classifier and two relevant statistics related to the soft-probability.

Uncertainty of the classifier: the standard deviation (SD method) The classical expression for the error probability associated with the Bayes decision rule is $P_e := \mathbb{P}(Y \neq f(X))$, computed over the underlying probability distribution p_{XY}. The expression of this error probability is:

$$P_e(p_X) := \mathbb{P}(Y \neq f(X)) = 1 - \mathbb{E}_{X\sim p_X}\Big[\max_{y\in\mathcal{Y}} p_{Y|X}(y|X)\Big]. \tag{5.9}$$

We also define the uncertainty of the classifier as:

$$P_u(p_X) := \mathbb{P}(\widehat{Y} \neq f(X)) = 1 - \mathbb{E}_{X\sim p_X}\Big[\max_{y\in\mathcal{Y}} p_{\widehat{Y}|X}(y|X)\Big]. \tag{5.10}$$

Note that both $P_u(p_X)$ and $P_e(p_X)$ are fundamentally different quantities. One measures the uncertainty of the classifier and the other one measures the classification error. In the following, we propose an uncertainty based-method for detecting dataset shift. We investigate shift detection through the lens of statistical two-sample testing between $P_u(p_X)$ and $P_u(p_{\widetilde{X}})$, according to samples from the source distribution p_X (from which training data is sampled) and a few sample batches from the target distribution $p_{\widetilde{X}}$ (from which real-world data is sampled). However, estimating high-quality confidence scores of the probabilities (eq.(5.10)) remains difficult and unstable, as will be observed from numerical results. To overcome this difficulty, we will show that a relationship exists between the uncertainty of the classifier and a "standard deviation" which is defined below and the following quantities:

$$\Delta_\alpha(\widehat{Y}|X=x) := \sum_{y\in\mathcal{Y}} \big(p_{\widehat{Y}|X}(y|x) - \underbrace{\mathbb{E}_{X\sim p_X}\Big[p_{\widehat{Y}|X}(y|X)\Big]}_{:=p_{\widehat{Y}(y)}}\big)^\alpha, \tag{5.11}$$

and

$$\overline{\Delta}_\alpha(\widehat{Y}|X) := \mathbb{E}_{X\sim p_X}\Big[\Delta_\alpha(\widehat{Y}|X=x)\Big], \tag{5.12}$$

for $\alpha > 0$, where $p_{\widehat{Y}}$ is a given prior to model the distribution of labels, e.g., $p_{\widehat{Y}}(y) = 1/M$ or $p_{\widehat{Y}}(y) = \mathbb{E}_{X\sim p_X}[p_{\widehat{Y}|X}(y|X)]$. Interestingly , this statistics without the prior is related to Rényi entropy [Erven and Harremos, 2014]: $H_\alpha(\widehat{Y}|X=x)$ of order $\alpha \in (0,1)\cup(1,\infty)$ by noticing that $(1-\alpha)H_\alpha(\widehat{Y}|X=x) = \log \|p_{\widehat{Y}|X}(\cdot|x)\|_\alpha^\alpha$. In particular, we will show that $P_u(p_X)$ and the useful statistics in eq. (5.11) with the particular choice of $\alpha = 2$ share some interesting connections that will be exploited to motivate the empirical estimate of $\overline{\Delta}_2(\widehat{Y}|X) = \mathbb{E}_{X\sim p_X}[\Delta_2(\widehat{Y}|X)]$ as a measure of eq. (5.10), i.e., the confidence (or uncertainty) of the classifier. The formal relationship is presented in the following proposition.

Proposition 2. *Let $\boldsymbol{p}_{\widehat{Y}|X}$ be a soft classifier and $p_{\widehat{Y}}$ a probability modeling the probability of labels, the following inequalities hold:*

$$1 - \sqrt{\overline{\Delta}_2(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p_{\widehat{Y}}^2} \leq P_u(p_X) \leq 1 - \overline{\Delta}_2(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p_{\widehat{Y}}^2(y). \tag{5.13}$$

This result suggests the use of $\overline{\Delta}_2(\widehat{Y}|X)$ (since $\sum_{y\in\mathcal{Y}} p_{\widehat{Y}}^2(y)$ is constant relative to the feature inputs) to measure deviations from the confidence of the classifier on the test samples (from distribution $p_{\widetilde{X}}$) w.r.t. that on the training samples (from distribution p_X). Obviously, it is not possible to train a binary discriminator since there are no available samples from the testing distribution. Instead of this, we shall characterize the cumulative probability distribution on the training samples of the standard deviation.

Proof. The upper bound is given by:

$$\Delta_2(\widehat{Y}|X) = \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X) - 2\sum_{y\in\mathcal{Y}} p_{\widehat{Y}}(y)p_{\widehat{Y}|X}(y|X) + \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y) \tag{5.14}$$

$$\mathbb{E}_{X\sim p_X}\left[\Delta_2(\widehat{Y}|X)\right] = \mathbb{E}_{X\sim p_X}\left[\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X)\right] - \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y) \tag{5.15}$$

$$\mathbb{E}_{X\sim p_X}\left[\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X)\right] \leq \underbrace{\mathbb{E}_{X\sim p_X}\left[\max_{y\in\mathcal{Y}} p_{\widehat{Y}|X}(y|X)\right]}_{=1-P_u(p_X)}. \tag{5.16}$$

Combining the previous inequality with eq. (5.9) and (5.15) leads to:

$$1 - P_u(p_X) \geq \overline{\Delta}_2(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y) \tag{5.17}$$

$$P_u(p_X) \leq 1 - \overline{\Delta}_2(\widehat{Y}|X) - \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y). \tag{5.18}$$

The lower bound is derived from this inequality:

$$\max_{y\in\mathcal{Y}} p_{\widehat{Y}|X}(y|X) \leq \sqrt{\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X)}. \tag{5.19}$$

We take the expectation over the two different terms and we apply Jensen's inequality to the right term:

$$\mathbb{E}_{X\sim p_X}\left[\max_{y\in\mathcal{Y}} p_{\widehat{Y}|X}(y|X)\right] \leq \sqrt{\mathbb{E}_{X\sim p_X}\left[\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X)\right]}. \tag{5.20}$$

Combining the previous inequality with eq. (5.9) and (5.15) leads to:

$$1 - P_u(p_X) \leq \sqrt{\overline{\Delta}_2(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}} \tag{5.21}$$

$$P_u(p_X) \geq 1 - \sqrt{\overline{\Delta}(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}}. \tag{5.22}$$

Note that for the uniform case:

$$\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y) = \frac{1}{|\mathcal{Y}|} \tag{5.23}$$

$$P_u(p_X) \geq 1 - \sqrt{\overline{\Delta}(\widehat{Y}|X) + \frac{1}{|\mathcal{Y}|}}. \tag{5.24}$$

□

The following proposition shows that the standard deviation is also related to the conditional Shannon entropy of the label $\widehat{Y}$ given the input feature X, which represents another natural possibility to measure the uncertainty of a classifier.

Proposition 3. *Let $\boldsymbol{p}_{\widehat{Y}|X}$ be a soft classifier and $P_{\widehat{Y}}$ a probability modeling the probability of labels, the following inequalities hold for the conditional entropy:*

$$1 + (1 - \overline{\Delta}_2(\widehat{Y}|X))\log(M-1) \geq H(\widehat{Y}|X) \geq \log M - \log\left(1 + M\overline{\Delta}_2(\widehat{Y}|X)\right), \tag{5.25}$$

Proof. Using the concavity of the function $x \mapsto \log(x)$, we can lower bound the entropy by

$$H(\widehat{Y}|X) = -\mathbb{E}_{X\widehat{Y}}[\log p_{\widehat{Y}|X}(\widehat{Y}|X)], \tag{5.26}$$

using Jensen's inequality:

$$\geq -\log \mathbb{E}_{X\widehat{Y}}[p_{\widehat{Y}|X}(\widehat{Y}|X)] \tag{5.27}$$

$$= -\log \mathbb{E}_X\Big[\sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}|X}(y|X)\Big]. \tag{5.28}$$

The above expression can be rewritten using the definition of $\overline{\Delta}_2$:

$$H(\widehat{Y}|X) \geq -\log\Big(\overline{\Delta}_2(\widehat{Y}|X) + \sum_{y\in\mathcal{Y}} p^2_{\widehat{Y}}(y)\Big). \tag{5.29}$$

Assuming the distribution of $\widehat{Y}$ is uniform and $|\mathcal{Y}| = M$

$$= -\log\Big(\overline{\Delta}_2(\widehat{Y}|X) + \frac{1}{M}\Big) \tag{5.30}$$

$$= \log M - \log\Big(M\overline{\Delta}_2(\widehat{Y}|X) + 1\Big), \tag{5.31}$$

which is the lower bound of the conditional entropy in eq. (5.28).

The upper bound follows from using Fano's inequality on the RV $\widehat{Y}$ and X with the uncertainty probability $P_u(p_X) := \mathbb{P}(\widehat{Y} \neq f(X))$:

$$H(\widehat{Y}|X) \leq 1 + P_u \log(M-1), \tag{5.32}$$

using the hypothesis that $\widehat{Y}$ is uniform and using the upper bound in Proposition 2:

$$\leq 1 + \Big[1 - \frac{1}{M} - \overline{\Delta}_2(\widehat{Y}|X)\Big]\log(M-1) \tag{5.33}$$

$$\leq 1 + \Big[1 - \overline{\Delta}_2(\widehat{Y}|X)\Big]\log(M-1), \tag{5.34}$$

which concludes the proof of the proposition.

□

It is worth mentioning that the initial definition of uncertainty in eq. (5.10) could be used to estimate uncertainty. Whereas we will illustrate in the numerical results (Fig. 5.9) that the corresponding estimates are highly unstable to be trusted. This is not the case of the SD method and GM method below.

Confusion matrix: geometric mean (GM method) The "learning step" consists in computing for each x_i of the training set the corresponding probabilities $\boldsymbol{p}_{\widehat{Y}|X}$ of a given label. Using the training set is possible to further estimate the confusion matrix, noted $\boldsymbol{p}_{\widehat{Y}||Y}$, on the source domain. The 'black box' depends implicitly on stable properties of the classifier when applied to the sequence of features, corresponding to a stable matrix, i.e., closed to a diagonal matrix. Now assume the properties of the test sequence change while that of the 'black box' does not. Tracking the confusion matrix would potentially allow to monitor those changes. Two cases are identified: If $\boldsymbol{p}_{\widehat{Y}|Y}$ does not change, then the test seems as valid as before; On the other hand, if $\boldsymbol{p}_{\widehat{Y}|Y}$ changes, then the "distance" between this new matrix and the previous one is an indicator of the amount of change. We suggest to detect the change based on KL divergence. More specifically we study the case where we assume that a single estimate $\boldsymbol{p}_{\widehat{Y}|X}(\cdot|x_i)$ is available, one can build a rough estimate of a confusion matrix based on this sample, by introducing a prior

$P(Y)$:

$$\boldsymbol{q}(\widehat{Y}|x_i, Y) = \boldsymbol{q}(\widehat{Y}|x_i)P(Y). \tag{5.35}$$

The KL divergence between the confusion matrix of the training leads to the following decomposition:

$$D\left(\boldsymbol{p}_{\widehat{Y}||Y}(\cdot||Y)\|\boldsymbol{p}_{\widehat{Y}|X||Y}(\cdot|x_i||Y)\right) = \sum_{y\in\mathcal{Y}}\sum_{\widehat{y}\in\mathcal{Y}} p_{\widehat{Y}||Y}(\widehat{Y}=\widehat{y}||Y=y)\log\frac{p_{\widehat{Y}|Y}(\widehat{Y}=\widehat{y}|Y=y)}{q_{\widehat{Y}|X}(\widehat{Y}=\widehat{y}|X=x_i)}$$

(5.36)

$$= \underbrace{\sum_{y\in\mathcal{Y}}\sum_{\widehat{y}\in\mathcal{Y}} p_{\widehat{Y}||Y}(\widehat{Y}=\widehat{y}||Y=y)\log p_{\widehat{Y}|Y}(\widehat{Y}=\widehat{y}|Y=y)}_{=h \text{ constant on } x_i}$$

$$- \sum_{y\in\mathcal{Y}}\sum_{\widehat{y}\in\mathcal{Y}} p_{\widehat{Y}||Y}(\widehat{Y}=\widehat{y}||Y=y)\log q_{\widehat{Y}|X}(\widehat{Y}=\widehat{y}|X=x_i)$$

(5.37)

$$= h - \underbrace{\sum_{\widehat{y}\in\mathcal{Y}} p_{\widehat{Y}}(\widehat{Y}=\widehat{y})\log q_{\widehat{Y}|X}(\widehat{Y}=\widehat{y}|X=x_i)}_{f(x_i)}, \tag{5.38}$$

under the assumption $\widehat{Y}$ is uniform

$$f(x_i) = \frac{1}{|\mathcal{Y}|}\sum_{\widehat{y}\in\mathcal{Y}} -\log q_{\widehat{Y}|X}(\widehat{Y}=\widehat{y}|X=x_i). \tag{5.39}$$

The f function is another expression of the geometric mean:

$$GM(x_i) = \exp\left(-f(x_i)\right) = \prod_{\widehat{y}\in\mathcal{Y}} \sqrt[|\mathcal{Y}|]{q_{\widehat{Y}|X}(\widehat{Y}=\widehat{y}|X=x_i)}. \tag{5.40}$$

The resulting expression eq. (5.40) corresponds to the geometric mean (GM) of the output probabilities.

Implementation of the statistical tests

Our approach relies mainly on the inference of the cumulative probability function of the involved statistical distance, i.e., SD or GM, with respect to the training samples:

$$F_{\Delta}(r|p_X) := \widehat{\mathbb{P}}\left(\Delta_2(\widehat{Y}|X) \leq r\right), \tag{5.41}$$

$$F_{\mathrm{GM}}(r|p_X) := \widehat{\mathbb{P}}\left(\mathrm{GM}(X) \leq r\right). \tag{5.42}$$

These functions will be used to relate distribution shift from p_X to $p_{\widetilde{X}}$, by detecting the shift from $F(r|p_X)$ to $F(r|p_{\widetilde{X}})$ based on the model mis-specification induced by the appropriate quantity (either the SD or the GM). After a threshold is computed on the validation data, the observed sequence of test (or mismatched) samples: $(\widetilde{x}_1, \ldots, \widetilde{x}_N)$ is used to compute the empirical average $\widehat{\mathbb{E}}_{\widetilde{X}}[\Delta_2(\widehat{Y}|\widetilde{X})]$ or $\widehat{\mathbb{E}}_{\widetilde{X}}[\mathrm{GM}(\widetilde{X})]$ and then the decision follows by comparing to a fix threshold $r > 0$.

Testing setup

Testing protocol. The protocol aims at evaluating the capacity of detection of the mismatched data sequences while keeping the False Positive Rate (FPR) of true matched samples under control. Clearly, if the detection rate on the mismatched data is of the same order of magnitude as the FPR, the method is not reliable. The efficiency of the algorithm is measured by the increase of detection rate compared to the FPR. The

classifier is based on vanilla deep convolutional networks. The DCNN architecture is two convolutional layers (respectively 64 and 32 (3x3) filters), each followed by a 2x2 max pooling layer and on top a fully connected layer of 512 neurons and a softmax activation output layer. We use dropout with parameter 0.35. Training is performed over 20 epochs with batch size 128 and Adam Optimizer [Kingma and Ba, 2015] with learning rate 0.001.

Testing model specs. The dataset used to train the network and the detector is divided into training and validation sets. The training set is dedicated to learn the network to perform the desired classification task. Once the training is accomplished, the validation set is used to determine a threshold corresponding to a pre-determined FPR given by the degree of confidence the user wants to have. The validation samples are only used once, and they are grouped into sequences of size N. Therefore, as N grows, the number of sequences decreases. The value of the threshold is numerically evaluated: once the validation sequences are processed, we search for the value that splits validation sequences in two parts, one of which having a size congruent with the required FPR. The data from the test set is then processed in sequences of same size N. Each sequence outputs a value that is compared to the threshold, deciding on matched or mismatched samples. The results are averaged over 10 different realizations of the DCNN. Even if the performance of the model is similar from one training to the other, this gives a more reliable input less dependant of the precise network weights configuration (leading to slight performance variations).

TABLE 5.5: False positive rate (FPR) threshold consistency between validation and training sets with the SD method for SVHN trained model.

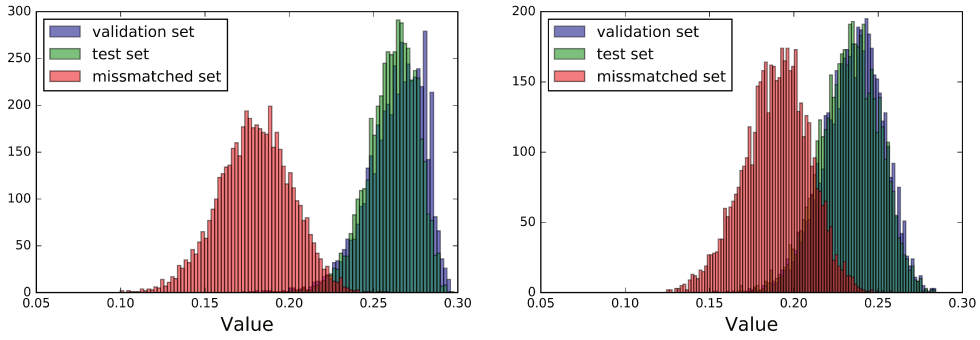
Target set FPR (in %)	Obtained test set FPR (in %) for sequence of size N								
N	2	3	4	5	6	7	8	9	10
5	4.98	4.26	4.28	3.3	4.16	3.94	3.8	4.24	3.58
0.5	0.10	0.54	0.52	0.42	0.36	0.36	0.38	0.28	0.22

Threshold specificity. Obviously, both methods of interest do not react exactly the same way to specific mismatches (this will be clear from experimental results). The mismatched samples are above the threshold for the GM method while they are below it for the SD method. In all experiments described below, performance is evaluated for FPR at 5% (square curves) and 0.5% (triangle curves), the black dashed curves corresponding to the SD method and the blue dotted curve to the GM method.

5.3.3 Experimental results

A first verification to be performed is to check whether the shape of the statistics of the validation set matches that of the test set without modification or not. This is checked on one example in table 5.5 where it is clear that the threshold value used for the validation set with the FPR gives really consistent values when applied to the test set. This is confirmed by the histograms in Figure 5.6. In this Figure, for both (a) and (b), the overlapping of the two empirical distributions of the corresponding statistics (SD value on the validation set and SD values on test set) for $N = 9$ is an additional evidence that evaluating the FPR threshold from validation set distribution is appropriate.

Direct performance without the use of validation (Mismatched data against test set) are summarized in table 5.6 which presents: (i) False Positive Rate (FPR) at 95%; True Positive Rate (TPR): probability of a mismatched sample classified as regular sample when the TPR is at 95%; (ii) Detection error: the misclassification probability when both error are equiprobable; (iii) Area Under the Receiver Operating Characteristic curve (AUROC); (iv) Area Under the Precision-Recall curve (AUPR) for both regular and mismatched samples. This table shows the raw performances of the investigated



(A) DCNN trained on SVHN and mismatched data from CIFAR-10.

(B) DCNN trained on CIFAR-10 and mismatched data from SVHN.

FIGURE 5.6: Histogram with 100 bins of the SD values for training, validation and mismatched set for $N = 9$.

methods compared to the L2-norm baseline on softmax, without using validation samples. Results show that the SD methods outperforms other methods except for the two following cases: SVHN against CIFAR-10 where the L2-norm is slightly better than GM method but overall performances are quite impressive; and the CIFAR-10 against CIFAR-100 simulation where again the GM method performances are better but still close to the other methods. Global performances vary for dataset to another. The simulations with low N show that the SD method rely on a minimal size of sequence.

TABLE 5.6: Scores of detection in various settings for the baseline (BL), the SD and the GM methods. All values are percentages, ↑ and ↓ indicate respectively larger and lower values are better.

Training data	Mismatched data	N	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
CIFAR-10	SVHN	2	BL	31.60	**16.27**	**91.00**	**92.33**	86.57
			GM	51.82	16.89	90.63	75.00	**92.16**
			SD	**29.32**	17.91	88.52	91.38	51.38
		10	BL	0.46	1.59	99.85	99.88	99.86
			GM	0.14	1.21	99.87	99.87	99.91
			SD	**0.02**	**0.33**	**99.97**	**99.99**	**99.99**
SVHN	CIFAR-10	2	BL	**30.60**	14.73	**93.57**	**93.95**	**93.15**
			GM	30.92	**13.97**	92.23	92.76	92.64
			SD	36.94	19.95	85.75	88.54	50.61
		10	BL	0.24	**1.21**	**99.91**	**99.94**	**99.93**
			GM	**0.20**	1.27	99.83	99.89	99.84
			SD	0.56	1.65	99.85	99.88	99.87
CIFAR-10	CIFAR-100	2	BL	**67.28**	28.55	78.21	**78.78**	76.00
			GM	72.68	**27.17**	**79.64**	77.72	**80.63**
			SD	69.60	31.47	73.09	75.85	51.79
		10	BL	**18.64**	10.65	95.91	96.17	95.68
			GM	19.78	**9.99**	**96.41**	**96.35**	**96.61**
			SD	21.10	11.05	95.65	95.90	83.20
F-MNIST	MNIST	2	BL	74.18	33.37	73.61	73.51	74.96
			GM	69.08	27.51	64.09	78.12	78.49
			SD	**53.46**	**23.31**	**82.62**	**85.19**	**79.87**
		10	BL	47.32	17.73	90.24	89.47	90.99
			GM	99.98	7.37	50.01	77.48	76.31
			SD	**0.48**	**1.67**	**99.85**	**99.88**	**99.87**
CIFAR-9	CIFAR-1	2	BL	95.48	48.41	52.11	46.10	52.85
			GM	87.90	45.05	58.08	59.19	61.91
			SD	**77.98**	**29.67**	**76.26**	**75.27**	**72.92**
		10	BL	89.42	41.49	62.27	60.28	58.96
			GM	69.82	31.09	76.27	76.36	75.77
			SD	**8.96**	**6.89**	**98.12**	**98.23**	**98.11**

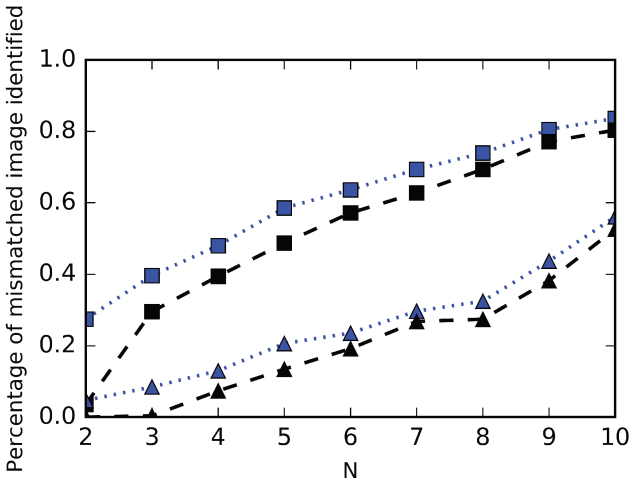


FIGURE 5.7: Mismatched data detection rate as a function of N. Model trained on CIFAR-10 and Mismatched data from CIFAR-100 database. Squares: 5%; Triangles: 0.5% false alarm rates; geometric mean (blue curve); empirical SD (black curve).

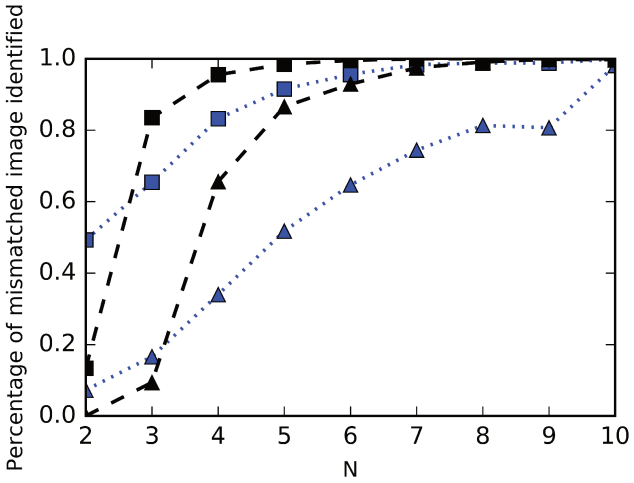


FIGURE 5.8: Mismatched data detection rate as a function of N. Model trained on CIFAR-10. Mismatched data from SVHN database. Squares: 5%; Triangles: 0.5% FPR. empirical GM (blue dotted); empirical SD (black dashed).

CIFAR-10 trained model. The model with two convolutional layers is trained over 20 epochs on CIFAR-10 dataset [Krizhevsky, 2009], resulting in correct classification rates between 74% and 75.5%. The validation data represent 10% of the training set (5 000 images), never used for training. This model has been tested with mismatched data from CIFAR-100 and SVHN databases.

CIFAR-100 mismatched data. Figure 5.7 shows the percentage of mismatched sequences identified over 5 000 sequences from CIFAR-100 [Krizhevsky, 2009] dataset for both methods. As intuition would suggest, the percentage of correct mismatch detection grows as the number of samples N in the sequence increases. In this context, the GM method clearly outperforms the SD method, especially for small N. A possible explanation is that the empirical mean value –inside the SD method formula– introduces another level of randomness when computed over 3 or less values. Indeed, the longer the sequence the closer the empirical value comes to an accurate value. For higher values of N the SD tool catches up the performance of the other tool.

SVHN mismatched data. Figure 5.8 shows the percentage of mismatched sequences correctly identified over 5 000 sequences from the SVHN [Netzer et al., 2011]

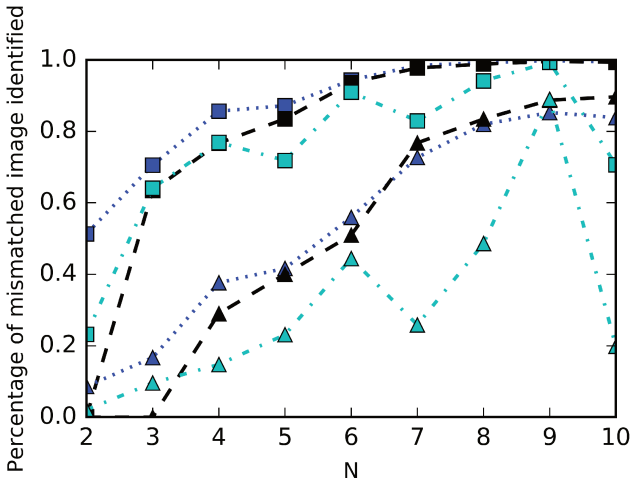


FIGURE 5.9: Mismatched data detection rate as a function of N. Model trained on SVHN. Mismatched data from CIFAR-10 database. Squares: 5%; Triangles: 0.5% FPR. empirical GM (blue dotted); empirical SD (black dashed); empirical uncertainty from definition eq. (5.10) in the paper (cyan dash-dot).

test set for both methods. The results seem to be quite different here: the SD method clearly outperforms the GM method while remaining less efficient for very small N. The efficiency of the SD tool is such that for $N = 6$ and higher, the 0.5% FPR (black triangle curve) has a mismatch detection rate similar to the one from GM method with a 5% FPR. The SD method reaches close to 100% detection rate for smaller values of N compared to the GM method. As a conclusion, in this context, the SD can be amazingly efficient, even with small (but not too small) N.

SVHN trained model with CIFAR-10 mismatched data. This simulation is the converse of the previous simulation. The database used to train the model is SVHN database. The model is composed of two convolutional layers and is trained over 20 epochs giving accuracies between 90.5% and 91.5% on the test set. This model has been tested with mismatched data from CIFAR-10 database. The results are shown on Figure 5.9. For this specific simulation, the GM method performs better than SD method for $N \leq 5$, while performance is similar for $N \geq 6$ for 5% FPR and $N = 7$ for 0.5%. We compare the results on this test set with this method from the results on the test set with the previous method. We use values from the appendix B. For N=5, the AUROC and the detection error provide better results with the current method: The detection rate goes up from 92.34% with first method to 94.65% with the second method. The AUROC value goes up from 95.68% with the first method to 98.86% with the second method. The previous method's results for $N = 5$ are quite good but not as good as the second method's results.

Fashion-MNIST trained model with MNIST mismatched data. Here, the model is trained on fashion-MNIST [Xiao, Rasul, and Vollgraf, 2017] database. The accuracy of the classification task on the test set was between 92% and 93%. The mismatched data come from MNIST database [LeCun and Cortes, 2010]. Note that, in contrast with the previous simulations both datasets share a uniform background, this implies that the statistics of numerous components will be identical (pixels of the background). The results are shown in Figure 5.10. While the SD method still performs poorly for very small N, it tremendously outperforms the other method for $N \geq 5$ (more specifically even with $N = 3$ for 5% FPR). Note that the performance under the 0.5% FPR are better than the performances of the methods GM and QM under the 5% FPR with $N \geq 5$.

CIFAR-10 trained model with addition of a class. In this simulation, we do not use mismatched data from another dataset. We remove a class (the truck class) from the CIFAR-10 dataset. The model is trained and tested on the 9 other classes. The performance on the classification of these 9 classes is between 74.2% and 75.2%. The

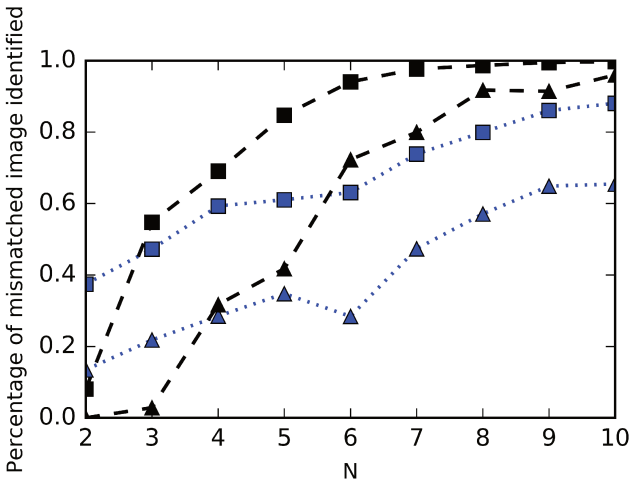


FIGURE 5.10: Mismatched data detection rate as a function of N. Model trained on fashion-MNIST and mismatched data from MNIST database. Squares: 5%; Triangles: 0.5% false alarm rates; geometric mean (blue curve); empirical SD (black curve).

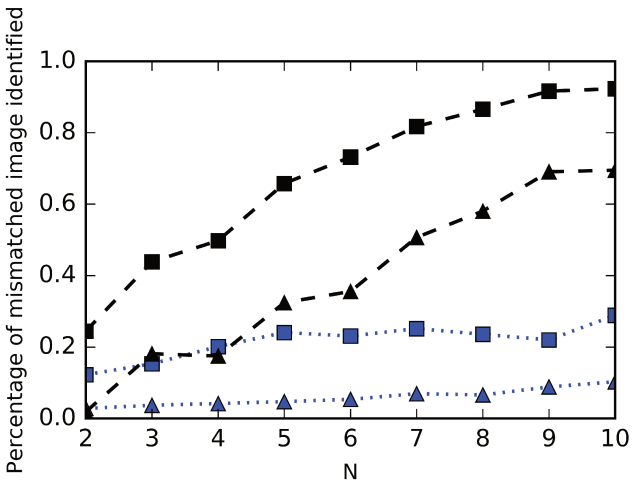


FIGURE 5.11: Mismatched data detection rate as a function of N. Model trained on CIFAR-10 without a class and mismatched data from the removed CIFAR-10 class. Squares: 5%; Triangles: 0.5% false alarm rates; geometric mean (blue curve); empirical SD (black curve).

determination of the threshold is still done from the validation set (all trucks images were removed form the validation set also). The mismatched data set is composed of the 1000 images of trucks removed from the test set. Therefore the computed values here use a lower number of sequences compared to the previous simulations. Nonetheless, as shown in Figure 5.11 the SD method clearly outperforms the GM method.

5.3.4 Concluding Remarks

We investigated the problem of detecting a mismatch between the trained deep neural network and the statistical properties of the sequence samples using this network, which may have useful applications, such as (i) deciding when the network should be re-trained, due to a progressive shift in the statistical properties of the sequence; (ii) when several users are using the same network, warn some users that they seem to use the network out of its nominal behaviour; and (iii) if the mismatch detection is very efficient, remove outliers during the training periods. The 'black box' scenario in which one cannot observe internal states of the network would seem to be a challenging one, but the methods proposed in this chapter result in amazing performance. However, performances seem to be of varying efficiency depending on the target context, which is not a surprise since the amount of mismatch should clearly impact the performance, i.e., small mismatch will require more samples to be detected than very strong ones. Clearly, the best criterion depends on the target scenario since the best algorithm based on SD and GM methods was not always the same among all experiments.

Chapter 6

Conclusion and perspectives

6.1 Conclusion

This manuscript presents the two main directions of my research: ***i*** data anonymization using deep learning tools, ***ii*** monitoring predictor behavior.

Concerning the anonymized representation, the first contribution is a loss involving some approximation, and it produced quite meaningful results on image type datasets. The private features were identified and accurately removed from the anonymized representation (to a certain degree that is controlled by a trade-off parameter) while the relevant features are kept as much as possible in the anonymized representation. The loss was based on an approximation of an information theoretic bound. Some limits of the methods where showed and leaded to a quite global understanding of this custom designed algorithm.

One of the limit of the supervised method (only features specified at the processing time are kept in the representation) lead us to think about a more general problem, the semi-supervised one. This method addresses the case where we do not have prior information about the features we want to keep. This second problem was addressed using a semi-supervised anonymized representation. In this case the representation shares the same type as the initial sample (here an image was anonymized into another image). The main purpose was to propose a method that took care of making as hard as possible the identification of private features while at the same time providing a partial integrity of the reconstructed representation. Partial integrity means here that the processed data keep as much coherence as possible under the anonymization constrain. The problem was solved using two methods according to the type of task: one for simple tasks and the other one for more intertwined tasks. The quality check of the reconstruction showed conclusive results.

Concerning the predictor monitoring in a 'black box' context, two approaches were presented.

The first one is based on a direct processing of the soft outputs provided by the predictor. The method is very simple, and is quite efficient for detecting a drift of the dataset statistics. Even if the detection requires a certain number of samples, its construction certainly allows to detect small changes, since it involves the full shape of the empirical PDF of the soft outputs, instead of the most significant values. The second approach intends to design predictors with a quick alarm in the case of unusual behavior.

From a more wider and technical standpoint, this work also contained a strong practical part. It means that I learnt to use various tools, some of which are already obsolete, and some other are cutting edge programming language for deep learning. I also learnt to work on huge GPU cluster as well as on small server that I had full responsibility of (i.e. installing my own tools and software to run simulation, and also tackling bugs when they arose).

6.2 Future work

The perspectives are numerous for the different parts of my work.

- A general remark is that it would present some interest to test the anonymization methods presented here on another type of data, for example voice recording.
- A special study of the statistical guarantees about the probabilities of error regarding the private task in the case of anonymized representation would be a nice development.
- The presented supervised method could be used in the converse way, using the regular label in the anonymization task branch and using the private label in the regular task branch. It would for sure produce surprising results.
- The practical case of crowd monitoring is a fitted application for the semi-supervised method on video data.
- A novel idea would be to produce anonymized representations that are of the same type as the original data, using the outputted representations of the supervised method. These new anonymized representations would be comparable to the representations produced by the semi-supervised method, and an assessment of the performance of each method would be interesting.
- An other idea is to test the behavior of the anonymizing encoder on data samples from another identity (another identity meaning that it is an identity that was not in the initial dataset). It would be an interesting way to test the methods for an other kind of generalization.
- Concerning the predictor monitoring work, the perspective lies into trying to make the tools available to people using machine learning without being expert. Such a goal could be achieve by developing an open access python library compatible with tensorflow for example.
- The presented methods could be tested for other types of shift.
- On a more practical note, a lot of work could be done to optimize the code of the algorithms developed here.
- Several smart grid data applications have been suggested and need to be investigated

Appendix A

Developpement of chapter 3

A.1 Proof of Lemma 5

The upper bound simply follows by using Jensen-Inequality [Cover and Thomas, 2006] while the lower bound is a consequence of the definition of the rate-distortion and distortion-rate functions. The probability of misclassification corresponding to the classifier can be expressed in terms of the expected distortion:

$$P_e(q_{U|X}, q_{\hat{Z}|U}) = \mathbb{E}_{p_{XZ}q_{U|X}}\left[d(Z,U)\right], \tag{A.1}$$

based on the distorsion measure $d(z,u) := 1 - q_{\hat{Z}|U}(z|u)$. Because of the Markov chain $Z \multimap X \multimap U$, we can use the data processing inequality [Cover and Thomas, 2006] and the definition of the rate-distortion function, obtaining the following bound for the classification error:

$$\begin{aligned}\mathcal{I}(Z;U) \geq & \min_{\substack{p_{\hat{U}|Z}:\mathcal{Z}\to\mathcal{P}(\mathcal{U}) \\ \mathbb{E}_{p_{\hat{U}Z}}[d(Z,\hat{U})] \leq \mathbb{E}_{p_{XZ}q_{U|X}}[d(Z,U)]}} \mathcal{I}(Z;U) && \text{(A.2)} \\ = & \, \mathcal{R}_{Z,q_{\hat{Z}|U}}\left(P_e(q_{U|X}, q_{\hat{Z}|U})\right). && \text{(A.3)}\end{aligned}$$

For $\mathbb{E}_{p_{XZ}q_{U|X}}\left[d(Z,U)\right]$, we can use the definition of $\mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(\cdot)$ (positive and monotonically decreasing) to obtain from (A.2), the desired inequality:

$$\mathcal{R}^{-1}_{Z,q_{\hat{Z}|U}}(\mathcal{I}(Z;U)) \leq P_{\mathcal{E}}(Q_{\hat{Z}|U}, Q_{U|Z}). \tag{A.4}$$

A.2 Proof of proposition 1

Here is the lemma 2.10 (Fano's lemma) in [Tsybakov, 2008]:

$$g\left(\hat{P}_e(q_{\hat{Z}|U}, q_{U|X})\right) \geq \log(|\mathcal{Z}|) - \frac{1}{|\mathcal{Z}|}\sum_{j=1}^{|\mathcal{Z}|} D(p_{\hat{Z}|U}||p_Z), \tag{A.5}$$

where for $0 \leq t \leq 1$:

$$g(t) := t \cdot \log\left(|\mathcal{Z}| - 1\right) + H(t), \tag{A.6}$$

with

$$H(t) := -t\log(t) - (1-t)\log(1-t), \tag{A.7}$$

and $0 \log 0 := 0$.

(A.8)

Let's remark that :

$$\frac{1}{|\mathcal{Z}|}\sum_{j=1}^{|\mathcal{Z}|} D(p_{\hat{Z}|U}||p_Z) = \mathbb{E}_{p_U}[D(p_{\hat{Z}|U}||p_Z)] \tag{A.9}$$

$$= D(p_{\hat{Z}U}||p_Z p_U) \tag{A.10}$$

$$= \hat{\mathcal{I}}(Z;U), \tag{A.11}$$

the lemma expression is :

$$g\left(\hat{P}_e(q_{\hat{Z}|U}, q_{U|X})\right) \geq \log(|\mathcal{Z}|) - \widehat{\mathcal{I}}(Z;U). \tag{A.12}$$

The function g is continuous and increasing on $[0, 1-1/|\mathcal{Z}|]$, g is $\log|\mathcal{Z}|$ above the interval and g is 0 below the interval. Therefore g^{-1} exists and can be applied on both term of the lemma which gives:

$$\hat{P}_e(q_{\hat{Z}|U}, q_{U|X}) \geq g^{-1}\left(\log(|\mathcal{Z}|) - \widehat{\mathcal{I}}(Z;U)\right), \tag{A.13}$$

which conclude the proof.

Appendix B

Developpement of chapter 5

B.1 First method

The Neural Network Architectures used in the paper are given below. Dense(n) denotes a fully-connected layer with n output units. Con2D(n, $w \times h$) denotes a convolutional layer with n output features and filter size of $w \times h$. ReLU is the rectified linear unit activation. The implementation is based on Keras.

Algorithm 1 Building MNIST model

```
model = models.Sequential()
model.add(Dense(512),activation='relu')
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Algorithm 2 Building Fashion-MNIST model

```
model = models.Sequential()
model.add(Conv2D(64, (2, 2), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D())
model.add(Dropout(0.3))
model.add(Conv2D(32, (2, 2), activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(256),activation='relu')
model.add(Dropout(0.5))
model.add(Dense(10))
model.add(Activation('softmax'))
```

B.2 Second method

B.2.1 Detailed description of our numerical results and simulations

Model training. The models employed to produce the figures are using the architecture presented below. The training procedure was simple and quick: We used an Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and no decay. The training set was split into 128 samples batches. The model was trained for 20 epochs. For each trained model, the whole validation is processed, and the softmax outputs are stored into N (size of sequence) labelled file. After this, for each trained model, 500 randomly chosen sequences of size N from the test set and from the missmatched data test set are processed through the model, and the softmax outputs are stored sequentially. The model is then reset and retrained. Our results use 10 model training realizations. For each new training, the performances are shown on the console.

Algorithm 3 Building CIFAR-10 or SVHN model

```
model = models.Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPool2D())
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Dropout(0.2))
model.add(Activation('softmax'))
```

From outputs to our curves. Once the python script has ended, one can load the data and process them one fixed sequence size at a time. After computing the each sequence value according to the chosen method, we advice to define a fixed false alarm rate threshold and process the validation given values accordingly to have a threshold: split in two parts the validation values such as smallest part compared to the biggest part give the same ratio as the false alarm rate. The value achieving this split is the threshold, process the test set sequences and the mismatched data sequences to obtain corresponding values. Each value is compared to the threshold: depending of the chosen method and of the position of the value to the threshold, the sample is assessed either compatible with the dataset distribution or not compatible, i.e., coming from the mismatched dataset (out of distribution sample). Note that the whole process does not include any use of labels, any modification of the initial classifier, any modification of the input of the model or any training of a learning based criteria. We emphasize that this method is therefore suited to an online usage on classifier, to monitor any under-confident behavior.

B.2.2 Architectures

Dense(n) denotes a fully-connected layer with n output units. Con2D(n, $w \times h$) denotes a convolutional layer with n output features and filter size of $w \times h$. ReLU is the rectified linear unit activation. The implementation is based on Keras.

Algorithm 4 Building CIFAR-10 or SVHN model

```
model = models.Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.35))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.35))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
```

B.2.3 Additional simulation results

Direct performances without the use of validation (Mismatched data against test set) are summarized in the following tables which presents: (i) False Positive Rate (FPR) at 95%; True Positive Rate (TPR): probability of a mismatched sample classified as regular sample when the TPR is at 95%; (ii) Detection error: the misclassification probability when both error are equiprobable; (iii) Area Under the Receiver Operating Characteristic Curve (AUROC); (iv) Area Under the Precision-Recall Curve (AUPR) for both regular and mismatched samples. This table shows the raw performances of the investigated methods compared to the L2-norm baseline on softmax, without using validation samples.

Result of the detection in various settings for the Baseline Method (BL), i.e., non-centered L2 norm of the sotf-probabilities, the SD method and the GM method. All values are percentages, ↑ indicates larger value is better, ↓ indicates lower value is better.

CIFAR-10 shifted to SVHN

Results show that the SD method clearly outperforms other methods except for the lower value of N i.e. $N = 2$. Yet there is a definitive gap between SD method and other methods for $N \geq 5$

TABLE B.1: CIFAR-10 shifted to SVHN

Training data	Mismatched data	N-	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
CIFAR-10	SVHN	2	BL	31.60	**16.27**	**91.00**	**92.33**	86.57
			GM	51.82	16.89	90.63	75.00	**92.16**
			SD	**29.32**	17.91	88.52	91.38	51.38
		3	BL	21.62	12.23	94.52	95.36	90.69
			GM	35.82	13.01	94.04	92.38	**95.04**
			SD	**13.84**	**8.83**	**96.37**	**97.02**	62.5
		4	BL	11.34	8.35	97.44	97.77	97.15
			GM	14.26	8.39	97.38	96.95	**97.81**
			SD	**5.00**	**5.01**	**98.68**	**98.92**	83.33
		5	BL	9.90	7.43	97.87	98.14	97.59
			GM	8.66	6.31	98.24	97.89	98.57
			SD	**2.34**	**3.23**	**99.48**	**99.56**	**99.26**
		6	BL	6.70	5.83	98.53	98.69	98.46
			GM	4.88	5.05	98.91	98.81	99.06
			SD	**1.10**	**2.31**	**99.74**	**99.79**	**99.75**
		7	BL	3.06	3.89	99.28	99.37	99.16
			GM	2.02	3.39	99.31	96.96	99.44
			SD	**0.44**	**1.39**	**99.87**	**99.91**	**99.88**
		8	BL	1.82	2.79	99.60	99.65	99.60
			GM	0.94	2.77	99.63	99.61	99.68
			SD	**0.14**	**0.65**	**99.95**	**99.98**	**99.97**
		9	BL	1.72	2.87	99.62	99.67	99.63
			GM	1.38	3.11	99.56	99.57	99.62
			SD	**0.06**	**0.43**	**99.97**	**99.99**	**99.99**
		10	BL	0.46	1.59	99.85	99.88	99.86
			GM	0.14	1.21	99.87	99.87	99.91
			SD	**0.02**	**0.33**	**99.97**	**99.99**	**99.99**

SVHN shifted to CIFAR-10

Results show that the baseline method and the GM method give close results, baseline method is often slighty better. SD method is a little less competitive on this particular configuration of simulation while staying close to the top results.

TABLE B.2: SVHN shifted to CIFAR-10

Training data	Mismatched data	N	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
SVHN	CIFAR-10	2	BL	**30.60**	14.73	**93.57**	**93.95**	**93.15**
			GM	30.92	**13.97**	92.23	92.76	92.64
			SD	36.94	19.95	85.75	88.54	50.61
		3	BL	**14.70**	8.85	**96.98**	**97.13**	**97.00**
			GM	14.88	**8.47**	96.43	96.73	96.59
			SD	40.18	12.85	93.65	93.43	51.07
		4	BL	7.52	6.21	**98.49**	**98.58**	**98.50**
			GM	**6.82**	**5.75**	98.16	98.47	98.20
			SD	15.96	10.51	96.19	96.41	56.25
		5	BL	5.76	**5.35**	**98.86**	**98.91**	**98.91**
			GM	**5.08**	**5.35**	98.33	98.73	98.35
			SD	11.06	7.71	97.53	97.78	62.50
		6	BL	2.04	**3.03**	**99.59**	**99.62**	**99.63**
			GM	**1.42**	3.11	99.34	99.49	99.36
			SD	4.90	4.93	98.82	98.95	75.00
		7	BL	0.76	1.97	**99.76**	**99.79**	**99.79**
			GM	**0.40**	**1.71**	99.66	99.77	99.68
			SD	1.90	2.93	99.51	99.57	99.52
		8	BL	0.32	**1.19**	**99.90**	**99.92**	**99.92**
			GM	**0.12**	1.39	99.84	99.88	99.86
			SD	1.26	2.49	99.66	99.72	99.63
		9	BL	**0.14**	0.99	99.91	99.93	99.94
			GM	0.22	**0.95**	**99.93**	**99.95**	**99.94**
			SD	0.60	1.69	99.80	99.83	99.81
		10	BL	0.24	**1.21**	**99.91**	**99.94**	**99.93**
			GM	**0.20**	1.27	99.83	99.89	99.84
			SD	0.56	1.65	99.85	99.88	99.87

CIFAR-10 shifted to CIFAR-100

Results show that for lower value of N GM method and baseling method are close. For higher value of N GM method seems to lay slightly better results than the baseline.

TABLE B.3: CIFAR-10 shifted to CIFAR-100

Training data	Mismatched data	N	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
CIFAR-10	CIFAR-100	2	BL	**67.28**	28.55	78.21	**78.78**	76.00
			GM	72.68	**27.17**	**79.64**	77.72	**80.63**
			SD	69.60	31.47	73.09	75.85	51.79
		3	BL	**57.36**	24.81	83.42	83.93	82.00
			GM	61.08	**22.73**	**85.04**	**83.95**	**85.70**
			SD	65.24	27.25	79.97	80.56	56.25
		4	BL	**51.08**	21.07	86.99	**87.44**	86.34
			GM	52.50	**19.71**	**88.27**	87.31	**88.88**
			SD	53.84	23.19	84.98	85.54	66.67
		5	BL	**43.90**	18.67	89.24	**89.69**	88.22
			GM	46.72	**17.61**	**90.04**	89.36	**90.43**
			SD	47.38	20.35	87.89	88.30	86.42
		6	BL	**33.50**	16.73	91.44	**92.20**	90.19
			GM	37.58	**15.41**	**92.42**	91.82	**92.96**
			SD	37.20	17.47	90.57	91.10	75.00
		7	BL	32.56	15.45	92.68	93.07	92.50
			GM	**32.36**	**13.79**	**93.58**	**93.27**	**93.91**
			SD	35.88	15.71	92.00	92.40	91.59
		8	BL	27.96	13.57	93.96	94.23	93.77
			GM	**23.96**	**11.81**	**95.14**	**94.93**	**95.45**
			SD	27.66	14.17	93.48	93.88	93.04
		9	BL	21.74	11.73	95.30	95.60	95.11
			GM	**21.36**	**11.23**	**95.93**	**95.88**	**96.20**
			SD	22.28	11.79	95.08	95.40	94.82
		10	BL	**18.64**	10.65	95.91	96.17	95.68
			GM	19.78	**9.99**	**96.41**	**96.35**	**96.61**
			SD	21.10	11.05	95.65	95.90	83.20

F-MNIST shifted to MNIST

Results show that the SD method outperforms other methods even for the lowest value of N. The other methods results are nowhere close. As N increase the gap is more and more tremendous.

TABLE B.4: F-MNIST shifted to MNIST

Training data	Mismatched data	N	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
F-MNIST	MNIST	2	BL	74.18	33.37	73.61	73.51	74.96
			GM	69.08	27.51	64.09	78.12	78.49
			SD	**53.46**	**23.31**	**82.62**	**85.19**	**79.87**
		3	BL	74.48	32.05	76.30	75.61	76.91
			GM	76.12	22.39	61.64	79.69	78.15
			SD	**38.66**	**17.45**	**90.53**	**91.11**	**85.95**
		4	BL	69.56	28.65	79.79	78.52	81.92
			GM	78.26	19.13	60.73	79.79	77.94
			SD	**20.62**	**11.13**	**95.38**	**95.84**	**94.83**
		5	BL	68.88	24.89	82.38	80.61	83.59
			GM	77.84	17.14	61.01	80.24	78.04
			SD	**11.88**	**7.73**	**97.51**	**97.66**	**97.30**
		6	BL	64.84	24.65	82.95	81.83	83.61
			GM	86.26	14.89	56.83	78.15	76.81
			SD	**5.68**	**5.45**	**98.82**	**98.89**	**98.84**
		7	BL	67.88	22.69	85.28	82.01	86.94
			GM	99.98	13.83	50.01	79.55	77.50
			SD	**2.38**	**3.41**	**99.44**	**99.49**	**99.46**
		8	BL	60.76	21.23	86.82	85.03	88.08
			GM	99.98	12.27	50.01	79.46	77.45
			SD	**2.22**	**3.41**	**99.51**	**99.57**	**99.53**
		9	BL	41.32	16.47	91.44	91.12	91.91
			GM	99.98	8.38	50.01	78.79	77.05
			SD	**0.90**	**1.91**	**99.80**	**99.83**	**99.82**
		10	BL	47.32	17.73	90.24	89.47	90.99
			GM	99.98	7.37	50.01	77.48	76.31
			SD	**0.48**	**1.67**	**99.85**	**99.88**	**99.87**

CIFAR-9 (CIFAR-10 without a label) shifted to CIFAR-1 (set of the removed label)

Results show that the SD method outperforms tremendously other methods on this additional class task. This method lays impressive results as N grows larger.

TABLE B.5: CIFAR-9 (CIFAR-10 without a label) shifted to CIFAR-1 (set of the removed label)

Training data	Mismatched data	N	Method	FPR (95% TPR)↓	Detection error ↓	AUROC ↑	AUPR test data ↑	AUPR mismatched ↑
CIFAR-9	CIFAR-1	2	BL	95.48	48.41	52.11	46.10	52.85
			GM	87.90	45.05	58.08	59.19	61.91
			SD	**77.98**	**29.67**	**76.26**	**75.27**	**72.92**
		3	BL	93.66	46.53	54.50	55.46	54.08
			GM	85.32	41.47	61.97	62.87	64.57
			SD	**52.58**	**23.35**	**85.22**	**85.71**	**84.55**
		4	BL	93.14	43.79	58.00	36.68	57.81
			GM	79.88	36.21	69.55	69.76	70.04
			SD	**45.56**	**19.51**	**88.51**	**88.94**	**88.05**
		5	BL	92.72	44.51	57.36	62.50	57.27
			GM	78.52	35.49	70.38	70.89	70.94
			SD	**34.98**	**16.07**	**92.07**	**92.30**	**92.11**
		6	BL	91.88	43.73	59.36	56.74	58.14
			GM	76.92	35.43	70.67	71.52	70.61
			SD	**25.04**	**12.29**	**94.74**	**94.98**	**94.71**
		7	BL	90.26	41.93	61.21	59.51	60.37
			GM	77.28	33.05	72.84	72.70	72.94
			SD	**20.20**	**10.67**	**96.06**	**96.08**	**96.15**
		8	BL	92.12	42.91	59.84	57.40	58.41
			GM	76.70	34.17	71.74	72.29	71.84
			SD	**13.10**	**8.59**	**97.21**	**97.39**	**97.16**
		9	BL	93.04	44.17	58.01	75.01	58.31
			GM	77.36	35.17	70.76	71.98	70.12
			SD	**8.90**	**7.07**	**98.20**	**98.34**	**98.18**
		10	BL	89.42	41.49	62.27	60.28	58.96
			GM	69.82	31.09	76.27	76.36	75.77
			SD	**8.96**	**6.89**	**98.12**	**98.23**	**98.11**

Bibliography

Abadi, Martin et al. (2016). "Deep Learning with Differential Privacy". In: *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, pp. 308–318. URL: https://arxiv.org/abs/1607.00133.

Alaiz-Rodríguez, Rocío and Nathalie Japkowicz (2008). "Assessing the Impact of Changing Environments on Classifier Performance". In: *Advances in Artificial Intelligence*. Ed. by Sabine Bergler. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 13–24. ISBN: 978-3-540-68825-9.

Alimoglu, F and E Alpaydin (1996). "Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition". In: *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96)*.

Aneja, Deepali et al. (2016). "Modeling Stylized Character Expressions via Deep Learning". In: *Asian Conference on Computer Vision*. Springer, pp. 136–153.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875*.

Bassily, R., A. Smith, and A. Thakurta (2014). "Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds". In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473. DOI: 10.1109/FOCS.2014.56.

Belghazi, Mohamed Ishmael et al. (2018). "Mutual Information Neural Estimation". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 530–539. URL: http://proceedings.mlr.press/v80/belghazi18a.html.

Billingsley, Patrick (1986). *Probability and Measure*. Second. John Wiley and Sons.

Bourlard, Hervé and Yves Kamp (1988). "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5, pp. 291–294.

Chaudhuri, Kamalika, Claire Monteleoni, and Anand D. Sarwate (2011). "Differentially Private Empirical Risk Minimization". In: *J. Mach. Learn. Res.* 12, pp. 1069–1109. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2021036.

Chen, Bee-Chung et al. (2009). "Privacy-Preserving Data Publishing". In: *Foundations and Trends® in Databases* 2.1–2, pp. 1–167. ISSN: 1931-7883. DOI: 10.1561/1900000008. URL: http://dx.doi.org/10.1561/1900000008.

Chen, Jiawei, Janusz Konrad, and Prakash Ishwar (2018). "VGAN-Based Image Representation Learning for Privacy-Preserving Facial Expression Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Chen, Tingting and Sheng Zhong (2009). "Privacy-preserving backpropagation neural network learning". In: *IEEE Transactions on Neural Networks* 20.10, pp. 1554–1564.

Cieslak, David A and Nitesh V Chawla (2009). "A framework for monitoring classifiers' performance: when and why failure occurs?" In: *Knowledge and Information Systems* 18.1, pp. 83–108.

Cover, T. M. and J. A. Thomas (2006). *Elements of Information Theory*. 2nd. New York, NY: Wiley.

Csiszár, I (1974). "On an extremum problem of information theory". In: *Studia Scientiarum Mathematicarum Hungarica* 9.1, pp. 57–71.

Csiszar, Imre and Janos Korner (1982). *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Orlando, FL, USA: Academic Press, Inc. ISBN: 0121984508.

Dailey, Matthew N et al. (2010). "Evidence and a computational explanation of cultural differences in facial expression recognition." In: *Emotion* 10.6, p. 874.

Duchi, John C., Michael I. Jordan, and Martin J. Wainwright (2014). "Privacy Aware Learning". In: *J. ACM* 61.6, 38:1–38:57. ISSN: 0004-5411. DOI: 10.1145/2666468. URL: http://doi.acm.org/10.1145/2666468.

Dwork, Cynthia (2006). "Differential Privacy". In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–12. ISBN: 978-3-540-35908-1.

— (2008). "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*. Ed. by Manindra Agrawal et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–19. ISBN: 978-3-540-79228-4. DOI: 10.1007/978-3-540-79228-4_1. URL: http://dx.doi.org/10.1007/978-3-540-79228-4_1.

Edwards, Harrison and Amos Storkey (2015). "Censoring representations with an adversary". In: *arXiv preprint arXiv:1511.05897*.

Erven, T. van and P. Harremos (2014). "Rényi Divergence and Kullback-Leibler Divergence". In: *IEEE Transactions on Information Theory* 60.7, pp. 3797–3820. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2320500.

Ganin, Yaroslav and Victor S. Lempitsky (2015). "Unsupervised Domain Adaptation by Backpropagation". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1180–1189. URL: http://jmlr.org/proceedings/papers/v37/ganin15.html.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Book in preparation for MIT Press. MIT Press. URL: http://www.deeplearningbook.org.

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Hamm, Jihun (2015). "Preserving privacy of continuous high-dimensional data with minimax filters". In: *Artificial Intelligence and Statistics*, pp. 324–332.

Hendrycks, Dan and Kevin Gimpel (2017). "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *ICLR*.

Hinton, Geoffrey E and Richard S Zemel (1994). "Autoencoders, minimum description length and Helmholtz free energy". In: *Advances in neural information processing systems*, pp. 3–10.

Hinton, Geoffrey E. et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580. arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

Jiang, Heinrich et al. (2018). "To Trust Or Not To Trust A Classifier". In: *NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* Pp. 5546–5557.

Kim, Hyeji et al. (2018). "Communication Algorithms via Deep Learning". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ryazCMbR-.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980.

Korshunov, Pavel and Sébastien Marcel (2018). "DeepFakes: a New Threat to Face Recognition? Assessment and Detection". In: *CoRR* abs/1812.08685. arXiv: 1812.08685. URL: http://arxiv.org/abs/1812.08685.

Krizhevsky, Alex (2009). "Learning multiple layers of features from tiny images". In:

Lecun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

LeCun, Yann, Yoshua Bengio, and Geoffrey E. Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database". In: URL: http://yann.lecun.com/exdb/mnist/.

Lecun, Yann and F. Fogelman Soulie (1987). "Modeles connexionnistes de l'apprentissage". English (US). In: *Intellectica* special issue apprentissage et machine.

Lee, Kimin et al. (2017). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *arXiv e-prints*, arXiv:1711.09325, arXiv:1711.09325. arXiv: 1711.09325 [stat.ML].

Lee, Kimin et al. (2018). "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *NeurIPS*. Montreal, Canada, pp. 7167–7177.

Lehmann, E. L. and Joseph P. Romano (2005). *Testing statistical hypotheses*. Third. Springer Texts in Statistics. New York: Springer, pp. xiv+784. ISBN: 0-387-98864-5.

Li, Yuezun and Siwei Lyu (2018). "Exposing deepfake videos by detecting face warping artifacts". In: *arXiv preprint arXiv:1811.00656* 2.

— (2019). "De-identification without losing faces". In: *CoRR* abs/1902.04202. arXiv: 1902.04202. URL: http://arxiv.org/abs/1902.04202.

Liang, Shiyu, Yixuan Li, and R Srikant (2018). "Principled detection of out-of-distribution examples in neural networks". In: *ICRL*.

Lipton, Zachary C, Yu-Xiang Wang, and Alex Smola (2018). "Detecting and correcting for label shift with black box predictors". In: *arXiv preprint arXiv:1802.03916*.

Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017). "Unsupervised Image-to-Image Translation Networks". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 700–708. URL: http://papers.nips.cc/paper/6672-unsupervised-image-to-image-translation-networks.pdf.

Lyons, M. et al. (1998). "Coding Facial Expressions with Gabor Wavelets". In: *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*. FG '98. Washington, DC, USA: IEEE Computer Society, pp. 200–. ISBN: 0-8186-8344-9. URL: http://dl.acm.org/citation.cfm?id=520809.796143.

Machanavajjhala, Ashwin et al. (2006). "l-diversity: Privacy beyond k-anonymity". In: *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, pp. 24–24.

McCulloch, Warren S. and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

Meden, Blaz et al. (2017). "Face Deidentification with Generative Deep Neural Networks". In: *CoRR* abs/1707.09376. arXiv: 1707.09376. URL: http://arxiv.org/abs/1707.09376.

Moreno-Torres, Jose G et al. (2012). "A unifying view on dataset shift in classification". In: *Pattern Recognition* 45.1, pp. 521–530.

Moyer, Daniel et al. (2018). "Invariant representations without adversarial training". In: *Advances in Neural Information Processing Systems*, pp. 9084–9093.

Nesterov, Yu. (2007). *Gradient methods for minimizing composite objective function*. CORE Discussion Papers 2007076. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE). URL: https://EconPapers.repec.org/RePEc:cor:louvco:2007076.

Netzer, Yuval et al. (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Pittaluga, Francesco, Sanjeev J. Koppal, and Ayan Chakrabarti (2018). "Learning Privacy Preserving Encodings through Adversarial Training". In: *CoRR* abs/1802.05214. arXiv: 1802.05214. URL: http://arxiv.org/abs/1802.05214.

Raval, N., A. Machanavajjhala, and L. P. Cox (2017). "Protecting Visual Secrets Using Adversarial Nets". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1329–1332. DOI: 10.1109/CVPRW.2017.174.

Ribaric, Slobodan, Aladdin Ariyaeeinia, and Nikola Pavesic (2016). *De-identification for privacy protection in multimedia content: A survey*. DOI: 10.1016/j.image.2016.05.020. URL: https://doi.org/10.1016/j.image.2016.05.020.

Rosenblatt, Frank (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY.

Shalev, Gabi, Yossi Adi, and Joseph Keshet (2018). "Out-of-distribution detection using multiple semantic label representations". In: *Advances in Neural Information Processing Systems*, pp. 7375–7385.

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, pp. 379–423.

Shimodaira, Hidetoshi (2000). "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *Journal of Statistical Planning and Inference* 90.2, pp. 227–244.

Smith, Adam D. (2008). "Efficient, Differentially Private Point Estimators". In: *CoRR* abs/0809.4794. arXiv: 0809.4794. URL: http://arxiv.org/abs/0809.4794.

Sugiyama, Masashi and Motoaki Kawanabe (2012). *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press. ISBN: 0262017091, 9780262017091.

Sweeney, Latanya (2002). "k-anonymity: A model for protecting privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05, pp. 557–570.

Taigman, Yaniv et al. (2015). "Web-scale training for face identification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2746–2754.

Tsybakov, Alexandre B. (2008). *Introduction to Nonparametric Estimation*. 1st. Springer Publishing Company, Incorporated. ISBN: 0387790519, 9780387790510.

Villard, Joffrey and Pablo Piantanida (2013). "Secure Multiterminal Source Coding With Side Information at the Eavesdropper". In: *IEEE Trans. Inf. Theor.* 59.6, pp. 3668–3692. ISSN: 0018-9448. DOI: 10.1109/TIT.2013.2245394. URL: http://dx.doi.org/10.1109/TIT.2013.2245394.

Vincent, Pascal et al. (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion." In: *Journal of Machine Learning Research* 11, pp. 3371–3408.

Wang, Ke et al. (2003). "Mining changes of classification by correspondence tracing". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, pp. 95–106.

Wang, Yu-Xiang, Jing Lei, and Stephen E. Fienberg (2016). "Learning with Differential Privacy: Stability, Learnability and the Sufficiency and Necessity of ERM Principle". In: *Journal of Machine Learning Research* 17.183, pp. 1–40. URL: http://jmlr.org/papers/v17/15-313.html.

Wasserman, Larry and Shuheng Zhou (2010). "A Statistical Framework for Differential Privacy". In: *Journal of the American Statistical Association* 105.489, pp. 375–389. DOI: 10.1198/jasa.2009.tm08651. eprint: https://doi.org/10.1198/jasa.2009.tm08651. URL: https://doi.org/10.1198/jasa.2009.tm08651.

Werbos, P.J. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University. URL: https://books.google.fr/books?id=z81XmgEACAAJ.

Widmer, Gerhard and Miroslav Kubat (1996). "Learning in the presence of concept drift and hidden contexts". In: *Machine Learning* 23.1, pp. 69–101. ISSN: 1573-0565. DOI: 10.1007/BF00116900. URL: https://doi.org/10.1007/BF00116900.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747. arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

Yang, Tsung-Yen et al. (2018). "Learning Informative and Private Representations via Generative Adversarial Networks". In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 1534–1543.

Yuan, Jiawei and Shucheng Yu (2014). "Privacy preserving back-propagation neural network learning made practical with cloud computing". In: *IEEE Transactions on Parallel and Distributed Systems* 25.1, pp. 212–221.

Zhu, Jun-Yan et al. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *CoRR* abs/1703.10593. arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Deux aspects de l'information utile : représentation anonymisée par l'apprentissage profond et surveillance de prédicteur

Mots clés : Apprentissage profond, Théorie de l'information, Anonymisation

Résumé : Le travail présenté ici est pour une première partie à l'intersection de l'apprentissage profond et anonymisation. Un cadre de travail complet est développé dans le but d'identifier et de retirer, dans une certaine mesure et de manière automatique, les caractéristiques privées d'une identité pour des données de type image. Deux méthodes différentes de traitement des données sont étudiées. Ces deux méthodes partagent une même architecture de réseau en forme de Y et cela malgré des différences concernant les types de couches de neurones utilisés conséquemment à leur objectif d'utilisation. La première méthode de traitement des données concerne la création *ex nihilo* de représentations anonymisées permettant un compromis entre la conservation des caractéristiques pertinentes et l'altération des caractéristiques privées. Ce cadre de travail a abouti à une nouvelle fonction de perte. Le deuxième type de traitement des données ne fait usage d'aucune information pertinente sur ces données et utilise uniquement des informations privées; ceci signifie que tout ce qui n'est pas une caractéristiques privées est supposé pertinent. Par conséquent les représentations anonymisées sont de même nature que les données intiales (une image est transformée en une image anonymisée). Cette tâche a conduit à un autre type d'architecture (toujours en forme de Y) et a fourni des résultats fortement sensibles au type des données.
La seconde partie de mon travail concerne une autre sorte d'information utile : cette partie se concentre sur la surveillance du comportement des prédicteurs. Dans le cadre de l'analyse de "modèle boîte noire", on a uniquement accès aux probabilités que le prédicteur fournit (sans aucune connaissance du type de structure/architecture qui produit ces probabilités). Cette surveillance est effectuée pour détecter des comportements anormaux. L'étude de ces probabilités peut servir d'indicateur d'inadéquation potentiel entre les statistiques des données et les statistiques du modèle. Deux méthodes utilisant différents outils sont présentées. La première compare la fonction de répartition des statistiques de sortie d'un ensemble connu et d'un ensemble de données à tester. La seconde fait intervenir deux outils : un outil reposant sur l'incertitude du classifieur et un autre outil reposant sur la matrice de confusion. Ces méthodes produisent des résultats concluants.

Title : Two sides of relevant information: anonymized representation through deep learning and predictor monitoring

Keywords : Deep learning, Information theory, Anonymization

Abstract : The work presented here is for a first part at the cross section of deep learning and anonymization. A full framework was developed in order to identify and remove to a certain extant, in an automated manner, the features linked to an identity in the context of image data. Two different kinds of processing data were explored. They both share the same Y-shaped network architecture despite components of this network varying according to the final purpose. The first one was about building from the ground an anonymized representation that allowed a trade-off between keeping relevant features and tampering private features. This framework has led to a new loss. The second kind of data processing specified no relevant information about the data, only private information, meaning that everything that was not related to private features is assumed relevant. Therefore the anonymized representation shares the same nature as the initial data (e.g. an image is transformed into an anonymized image). This task led to another type of architecture (still in a Y-shape) and provided results strongly dependent on the type of data.
The second part of the work is relative to another kind of relevant information: it focuses on the monitoring of predictor behavior. In the context of black box analysis, we only have access to the probabilities outputted by the predictor (without any knowledge of the type of structure/architecture producing these probabilities). This monitoring is done in order to detect abnormal behavior that is an indicator of a potential mismatch between the data statistics and the model statistics. Two methods are presented using different tools. The first one is based on comparing the empirical cumulative distribution of known data and to be tested data. The second one introduces two tools: one relying on the classifier uncertainty and the other relying on the confusion matrix. These methods produce concluding results.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France