Project no. 826278

SERUMS

Research & Innovation Action (RIA)
SECURING MEDICAL DATA IN SMART PATIENT-CENTRIC HEALTHCARE SYSTEMS

Final Report on Distributed Differential Privacy Transfer and Homomorphic Encryption
D3.3

Due date of deliverable: 31st March 2022

Start date of project: January 1st, 2019

Type: Deliverable
WP number: WP3

Responsible institution: Software Competence Center Hagenberg
Editor and editor's address: Michael Rossbory, Software Competence Center Hagenberg

Version 1.0

Project co-funded by the European Commission within the Horizon 2020 Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change Log

Rev.	Date	Who	Site	What
1	10/02/22	Michael Rossbory	SCCH	Document Structure
2	20/02/22	Mohit Kumar	SCCH	T3.2, T3.3
3	25/02/22	Eduard Baranov	UCL	T3.4
4	27/03/22	Michael Rossbory	SCCH	Summary, Proof Reading, Finalizing

Executive Summary

A methodology for practical secure privacy-preserving distributed machine (deep) learning is proposed via addressing the core issues of scalable transferrable deep learning, differential privacy, and fully homomorphic encryption. Considering that private data is distributed and the training data may contain directly or indirectly an information about private data, an architecture and a methodology are suggested for

1. addressing the optimal model size determination issue and computationally fast training issue of scalable and fast machine (deep) learning with an alternative approach based on variational learning;
2. addressing the privacy-accuracy tradeoff issue of differential privacy via optimizing the noise adding mechanism;
3. defining an information theoretic measure of privacy-leakage for the design and analysis of privacy-preserving schemes; and
4. mitigating the impracticality issue of fully homomorphic encryption (arising from large computational overhead) via very fast gate-by-gate bootstrapping and introducing a learning scheme that requires homomorphic computation of only efficient-to-evaluate functions.

Contents

Executive Summary . . . 2

1. Privacy-Preserving Semi-Supervised Transfer and Multi-Task Approaches (T3.2) 4
1.1 Mathematical Background . . . 5
1.1.1 Notations . . . 5
1.1.2 A Review of Measure Theoretic Conceptualization of Membership-Mappings . . . 6
1.2 Variational Conditionally Deep Membership-Mapping Autoencoders . . . 7
1.2.1 Conditionally Deep Membership-Mapping Autoencoders . . . 7
1.2.2 Variational Learning of Membership-Mappings . . . 8
1.2.3 Algorithm for Variational Learning of Conditionally Deep Membership-Mapping Autoencoders . . . 11
1.2.4 Wide Conditionally Deep Membership-Mapping Autoencoder . . . 13
1.2.5 Classification Applications . . . 14
1.3 Privacy-Preserving Transferrable Deep Learning . . . 14
1.3.1 An Optimal $(\epsilon, \delta)-$Differentially Private Noise Adding Mechanism . . . 14
1.3.2 Differentially Private Semi-Supervised Transfer and Multi-Task Learning . . . 16
1.4 Experiments . . . 17
1.4.1 Demonstrative Examples Using MNIST and USPS Datasets . . . 18
1.4.2 Comparisons Using Office and Caltech256 Datasets . . . 19
1.5 Concluding Remarks . . . 25

2. Secure Multiparty Computation and Homomorphic Encryption (T3.3) 26

3. Verification of Differential Privacy Deep Learning Models (T3.4) 28
3.1 Membership Inference Attack on SERUMS Model . . . 28

Appendices 30
4.1 Evaluation of Membership Function . . . 30
4.2 Solution of Optimization Problem . . . 30
4.3 Membership-Mapping Output Estimation . . . 31

1. Privacy-Preserving Semi-Supervised Transfer and Multi-Task Approaches (T3.2)

A novel differentially private semi-supervised transfer and multi-task learning framework is developed that

1. is capable of handling high-dimensional data and heterogeneity of domains;
2. optimizes the differential private noise adding mechanism such that for a given level of privacy, the perturbation in the data is as small as possible;
3. allows learning of the target domain model without requiring an access to source domain private training data;
4. ensures that a high level of privacy (i.e. sufficiently low value of privacy-loss bound) would not degrade the learning performance;
5. allows employing deep models in source and target domains so that data features at different abstraction levels can be used to transfer knowledge across domains;
6. follows the analytical approach [1–5] to the learning of deep models while addressing the issues related to optimal choice of model structure.

The basic idea of the proposed approach is stated as in Fig. 1.1. The method is as follows:

- An optimal differentially private noise adding mechanism is used to perturb the source dataset for preserving its privacy. The perturbed source data is used for the learning of classifier and for the computation of other parameters required for transferring knowledge from source to target domain.
- The classifiers consist of *Conditionally Deep Membership-Mapping Autoencoder (CDMMA)* based compositions. A multi-class classifier is presented that employs a parallel composition of CDMMAs to learn data representation for each class. An analytical approach is presented for the learning of the CDMMA.
- Since differential privacy will remain immune to any post-processing of noise added data samples, the perturbed source dataset is used to
  - build a differentially private source domain classifier,
  - compute a differentially private source domain latent subspace transformation-matrix.

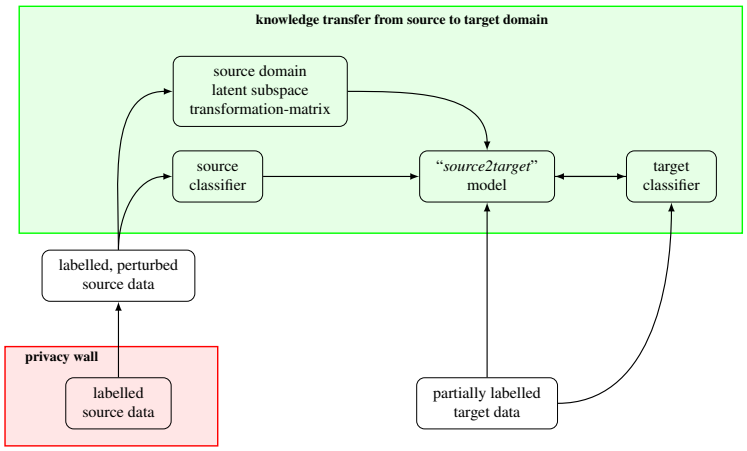

Figure 1.1: The proposed approach to privacy-preserving semi-supervised transfer and multi-task learning.

- The target domain classifier is learned adaptively in a manner that higher-level data features are used during initial iterations for updating the classifier parameters and as the number of iterations increases more and more lower-level data features are intended to be included in the process of updating the classifier parameters.
- The knowledge from source to target domain is transferred via
  1. building a "*source2target*" model that uses variational membership-mappings to define a transformation from source domain data space to the target domain data space,
  2. combining both source and target domain classifiers with source2target model for a transfer and *multi-task* learning scenario.
- Since no flow of data/information occurs from target to source domain, no privacy-preserving mechanism is used to protect the target data.

1.1 Mathematical Background

1.1.1 Notations

- Let $n, N, p, M \in \mathbb{N}$.
- Let $\mathcal{B}(\mathbb{R}^N)$ denote the *Borel* $\sigma-$*algebra* on $\mathbb{R}^N$, and let λ^N denote the *Lebesgue measure* on $\mathcal{B}(\mathbb{R}^N)$.
- Let $(\mathcal{X}, \mathcal{A}, \rho)$ be a probability space with unknown probability measure ρ.
- Let us denote by $\mathcal{S}$ the set of finite samples of data points drawn i.i.d. from ρ, i.e.,

$$\mathcal{S} := \{(x^i \sim \rho)_{i=1}^N \mid N \in \mathbb{N}\}. \tag{1.1}$$

- For a sequence $\mathrm{x} = (x^1, \cdots, x^N) \in \mathcal{S}$, let $|\mathrm{x}|$ denote the cardinality i.e. $|\mathrm{x}| = N$.
- If $\mathrm{x} = (x^1, \cdots, x^N),\ \mathrm{a} = (a^1, \cdots, a^M) \in \mathcal{S}$, then $\mathrm{x} \wedge \mathrm{a}$ denotes the concatenation of the sequences x and a, i.e., $\mathrm{x} \wedge \mathrm{a} = (x^1, \ldots, x^N, a^1, \ldots, a^M)$.
- Let us denote by $\mathbb{F}(\mathcal{X})$ the set of $\mathcal{A}$-$\mathcal{B}(\mathbb{R})$ measurable functions $f : \mathcal{X} \to \mathbb{R}$, i.e.,

$$\mathbb{F}(\mathcal{X}) := \{f : \mathcal{X} \to \mathbb{R} \mid f \text{ is } \mathcal{A}\text{-}\mathcal{B}(\mathbb{R}) \text{ measurable}\}. \tag{1.2}$$

- For convenience, the values of a function $f \in \mathbb{F}(\mathcal{X})$ at points in the collection $\mathrm{x} = (x^1, \cdots, x^N)$ are represented as $f(\mathrm{x}) = (f(x^1), \cdots, f(x^N))$.
- For a given $\mathrm{x} \in \mathcal{S}$ and $A \in \mathcal{B}(\mathbb{R}^{|\mathrm{x}|})$, the cylinder set $\mathcal{T}_{\mathrm{x}}(A)$ in $\mathbb{F}(\mathcal{X})$ is defined as

$$\mathcal{T}_{\mathrm{x}}(A) := \{f \in \mathbb{F}(\mathcal{X}) \mid f(\mathrm{x}) \in A\}. \tag{1.3}$$

- Let $\mathcal{T}$ be the family of cylinder sets defined as

$$\mathcal{T} := \left\{\mathcal{T}_{\mathrm{x}}(A) \mid A \in \mathcal{B}(\mathbb{R}^{|\mathrm{x}|}),\ \mathrm{x} \in \mathcal{S}\right\}. \tag{1.4}$$

- Let $\sigma(\mathcal{T})$ be the σ-algebra generated by $\mathcal{T}$.
- Given two $\mathcal{B}(\mathbb{R}^N) - \mathcal{B}(\mathbb{R})$ measurable mappings, $g : \mathbb{R}^N \to \mathbb{R}$ and $\mu : \mathbb{R}^N \to \mathbb{R}$, the weighted average of $g(\mathrm{y})$ over all $\mathrm{y} \in \mathbb{R}^N$, with $\mu(\mathrm{y})$ as the weighting function, is computed as

$$\langle g \rangle_\mu := \frac{1}{\int_{\mathbb{R}^N} \mu(\mathrm{y})\ \mathrm{d}\lambda^N(\mathrm{y})} \int_{\mathbb{R}^N} g(\mathrm{y})\mu(\mathrm{y})\ \mathrm{d}\lambda^N(\mathrm{y}). \tag{1.5}$$

1.1.2 A Review of Measure Theoretic Conceptualization of Membership-Mappings

1.1.2.1 Representation of Samples via Attribute Values

Consider a given observation $x \in \mathcal{X}$, a data point $\tilde{x} \in \mathcal{X}$, and a mapping $\mathbf{A}_{x,f}(\tilde{x}) = (\zeta_x \circ f)(\tilde{x})$ composed of two mappings $f : \mathcal{X} \to \mathbb{R}$ and $\zeta_x : \mathbb{R} \to [0,1]$. $f \in \mathbb{F}(\mathcal{X})$ can be interpreted as physical measurement (e.g., temperature), and $\zeta_x(f(\tilde{x}))$ as degree to which $\tilde{x}$ matches the attribute under consideration, e.g. "*hot*" where e.g. x is a representative sample of "*hot*". This concept is extended to sequences of data points in order to evaluate how much a sequence $\tilde{\mathrm{x}} = (\tilde{x}^1, \ldots, \tilde{x}^N) \in \mathcal{S}$ matches to the attribute induced by observed sequence $\mathrm{x} = (x^1, \ldots, x^N) \in \mathcal{S}$ w.r.t. the feature f via defining

$$\mathbf{A}_{\mathrm{x},f}(\tilde{\mathrm{x}}) = (\zeta_{\mathrm{x}} \circ f)(\tilde{\mathrm{x}}) \tag{1.6}$$
$$= \zeta_{\mathrm{x}}(f(\tilde{x}^1), \ldots, f(\tilde{x}^N)), \tag{1.7}$$

where the membership functions $\zeta_{\mathrm{x}} : \mathbb{R}^{|\mathrm{x}|} \to [0,1]$, $\mathrm{x} \in \mathcal{S}$, satisfy the following properties:

Nowhere Vanishing: $\zeta_{\mathrm{x}}(\mathrm{y}) > 0$ for all $\mathrm{y} \in \mathbb{R}^{|\mathrm{x}|}$, i.e.,

$$\mathrm{supp}[\zeta_{\mathrm{x}}] = \mathbb{R}^{|\mathrm{x}|}. \tag{1.8}$$

Positive and Bounded Integrals: the functions ζ_{x} are absolutely continuous and Lebesgue integrable over the whole domain such that for all $\mathrm{x} \in \mathcal{S}$ we have

$$0 < \int_{\mathbb{R}^{|\mathrm{x}|}} \zeta_{\mathrm{x}} \, \mathrm{d}\lambda^{|\mathrm{x}|} < \infty. \tag{1.9}$$

Consistency of Induced Probability Measure: the membership function induced probability measures $\mathbb{P}_{\zeta_{\mathrm{x}}}$, defined on any $A \in \mathcal{B}(\mathbb{R}^{|\mathrm{x}|})$, as

$$\mathbb{P}_{\zeta_{\mathrm{x}}}(A) := \frac{1}{\int_{\mathbb{R}^{|\mathrm{x}|}} \zeta_{\mathrm{x}} \, \mathrm{d}\lambda^{|\mathrm{x}|}} \int_A \zeta_{\mathrm{x}} \, \mathrm{d}\lambda^{|\mathrm{x}|} \tag{1.10}$$

are consistent in the sense that for all $\mathrm{x}, \ \mathrm{a} \in \mathcal{S}$:

$$\mathbb{P}_{\zeta_{\mathrm{x}\wedge\mathrm{a}}}(A \times \mathbb{R}^{|\mathrm{a}|}) = \mathbb{P}_{\zeta_{\mathrm{x}}}(A). \tag{1.11}$$

The collection of membership functions satisfying aforementioned assumptions is denoted by

$$\Theta := \{\zeta_{\mathrm{x}} : \mathbb{R}^{|\mathrm{x}|} \to [0,1] \mid (1.8), (1.9), (1.11), \ \mathrm{x} \in \mathcal{S}\}. \tag{1.12}$$

1.1.2.2 Measure Space

It is shown in [4] that $(\mathbb{F}(\mathcal{X}), \sigma(\mathcal{T}), \mathbf{p})$ is a measure space and the probability measure $\mathbf{p}$ is defined as

$$\mathbf{p}(\mathcal{T}_{\mathrm{x}}(A)) := \mathbb{P}_{\zeta_{\mathrm{x}}}(A) \tag{1.13}$$

where $\zeta_{\mathrm{x}} \in \Theta$, $\mathrm{x} \in \mathcal{S}$, $A \in \mathcal{B}(\mathbb{R}^{|\mathrm{x}|})$, and $\mathcal{T}_{\mathrm{x}}(A) \in \mathcal{T}$. It follows from [4] that for a given $\mathcal{B}(\mathbb{R}^{|\mathrm{x}|}) - \mathcal{B}(\mathbb{R})$ measurable mapping $g : \mathbb{R}^{|\mathrm{x}|} \to \mathbb{R}$, expectation of $(g \circ f)(\mathrm{x})$ over $f \in \mathbb{F}(\mathcal{X})$ w.r.t. probability measure $\mathbf{p}$ is given as

$$\mathbb{E}_{\mathbf{p}}[(g \circ \cdot)(\mathrm{x})] = \langle g \rangle_{\zeta_{\mathrm{x}}}. \tag{1.14}$$

The significance of equality (1.14) is to allow calculating averages over all real valued functions belonging to $\mathbb{F}(\mathcal{X})$ via simply computing a weighted average.

1.1.2.3 Student-t Membership-Mapping

Definition 1 (Student-t Membership-Mapping) *A Student-t membership-mapping, $\mathcal{F} \in \mathbb{F}(\mathcal{X})$, is a mapping with input space $\mathcal{X} = \mathbb{R}^n$ and a membership function $\zeta_{\mathrm{x}} \in \Theta$ that is Student-t like:*

$$\zeta_{\mathrm{x}}(\mathrm{y}) = \left(1 + 1/(\nu - 2)\,(\mathrm{y} - \mathrm{m}_{\mathrm{y}})^T K_{\mathrm{xx}}^{-1} (\mathrm{y} - \mathrm{m}_{\mathrm{y}})\right)^{-\frac{\nu + |\mathrm{x}|}{2}} \tag{1.15}$$

where $\mathrm{x} \in \mathcal{S}$, $\mathrm{y} \in \mathbb{R}^{|\mathrm{x}|}$, $\nu \in \mathbb{R}_+ \setminus [0,2]$ is the degrees of freedom, $\mathrm{m}_{\mathrm{y}} \in \mathbb{R}^{|\mathrm{x}|}$ is the mean vector, and $K_{\mathrm{xx}} \in \mathbb{R}^{|\mathrm{x}| \times |\mathrm{x}|}$ is the covariance matrix with its $(i,j)-$th element given as

$$(K_{\mathrm{xx}})_{i,j} = kr(x^i, x^j) \tag{1.16}$$

where $kr : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a positive definite kernel function defined as

$$kr(x^i, x^j) = \sigma^2 \exp\left(-0.5 \sum_{k=1}^{n} w_k \left| x_k^i - x_k^j \right|^2\right) \tag{1.17}$$

where x_k^i is the $k-$th element of x^i, σ^2 is the variance parameter, and $w_k \geq 0$ (for $k \in \{1, \cdots, n\}$).

It is shown in [4] that membership function as defined in (1.15) satisfies the consistency condition (1.11).

1.1.2.4 Interpolation

For a zero-mean Student-t membership-mapping $\mathcal{F} \in \mathbb{F}(\mathbb{R}^n)$, let $\mathrm{x} = \{x^i \in \mathbb{R}^n \mid i \in \{1, \cdots, N\}\}$ be a given set of input points and the corresponding mapping outputs are represented by the vector $\mathrm{f} := (\mathcal{F}(x^1), \cdots, \mathcal{F}(x^N))$. Let $\mathrm{a} = \{a^m \mid a^m \in \mathbb{R}^n,\ m \in \{1, \cdots, M\}\}$ be the set of auxiliary inducing points and the mapping outputs corresponding to auxiliary inducing inputs are represented by the vector $\mathrm{u} := (\mathcal{F}(a^1), \cdots, \mathcal{F}(a^M))$. It follows from [4] that f, based upon the interpolation on elements of u, could be represented by means of a membership function, $\mu_{\mathrm{f;u}} : \mathbb{R}^N \to [0, 1]$, defined as

$$\mu_{\mathrm{f;u}}(\tilde{\mathrm{f}}) := \left(1 + \frac{1}{\nu + M - 2}(\tilde{\mathrm{f}} - \bar{m}_{\mathrm{f}})^T \left(\frac{\nu + (\mathrm{u})^T (K_{\mathrm{aa}})^{-1}\mathrm{u} - 2}{\nu + M - 2}\bar{K}_{\mathrm{xx}}\right)^{-1} (\tilde{\mathrm{f}} - \bar{m}_{\mathrm{f}})\right)^{-\frac{\nu+M+N}{2}} \tag{1.18}$$

$$\bar{m}_{\mathrm{f}} = K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u} \tag{1.19}$$

$$\bar{K}_{\mathrm{xx}} = K_{\mathrm{xx}} - K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T, \tag{1.20}$$

where $K_{\mathrm{aa}} \in \mathbb{R}^{M \times M}$ and $K_{\mathrm{xa}} \in \mathbb{R}^{N \times M}$ are matrices with their $(i, j)-$th elements given as

$$(K_{\mathrm{aa}})_{i,j} = kr(a^i, a^j) \tag{1.21}$$

$$(K_{\mathrm{xa}})_{i,j} = kr(x^i, a^j) \tag{1.22}$$

where $kr : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a positive definite kernel function defined as in (1.17).

Here, the pair $(\mathbb{R}^N, \mu_{\mathrm{f;u}})$ constitutes a fuzzy set and $\mu_{\mathrm{f;u}}(\tilde{\mathrm{f}})$ is interpreted as the degree to which $\tilde{\mathrm{f}}$ matches an attribute induced by f for a given u.

1.2 Variational Conditionally Deep Membership-Mapping Autoencoders

Bregman divergence based conditionally deep autoencoders were introduced in [5]. Here, a special case of Bregman divergence corresponding to the squared Euclidean norm is considered for the conditionally deep autoencoders.

1.2.1 Conditionally Deep Membership-Mapping Autoencoders

Definition 2 (Membership-Mapping Autoencoder [5]) *A membership-mapping autoencoder, $\mathcal{G} : \mathbb{R}^p \to \mathbb{R}^p$, maps an input vector $y \in \mathbb{R}^p$ to $\mathcal{G}(y) \in \mathbb{R}^p$ such that*

$$\mathcal{G}(y) := [\,\mathcal{F}_1(Py)\ \cdots\ \mathcal{F}_p(Py)\,]^T, \tag{1.23}$$

where $\mathcal{F}_j$ ($j \in \{1, 2, \cdots, p\}$) is a Student-t membership-mapping (Definition 1), $P \in \mathbb{R}^{n \times p} (n \leq p)$ is a matrix such that the product Py is a lower-dimensional encoding for y. That is, membership-mapping autoencoder first projects the input vector onto a lower dimensional subspace and then constructs the output vector through Student-t membership-mappings.

Definition 3 (Conditionally Deep Membership-Mapping Autoencoder (CDMMA) [5]) *A conditionally deep membership-mapping autoencoder, $\mathcal{D} : \mathbb{R}^p \to \mathbb{R}^p$, maps a vector $y \in \mathbb{R}^p$ to $\mathcal{D}(y) \in \mathbb{R}^p$ through a nested composition of finite number of membership-mapping autoencoders such that*

$$y^l = (\mathcal{G}_l \circ \cdots \circ \mathcal{G}_2 \circ \mathcal{G}_1)(y),\ \forall l \in \{1, 2, \cdots, L\} \tag{1.24}$$

$$l^* = \arg \min_{l \in \{1,2,\cdots,L\}} \|y - y^l\|^2 \tag{1.25}$$

$$\mathcal{D}(y) = y^{l^*}, \tag{1.26}$$

where $\mathcal{G}_l(\cdot)$ is a membership-mapping autoencoder (Definition 2); y^l is the output of $l-$th layer representing input vector y at certain abstraction level such that y^1 is least abstract representation and y^L is most abstract representation of the input vector; and the autoencoder output $\mathcal{D}(y)$ is equal to the output of the layer re-constructing the given input vector as good as possible where re-construction error is measured in-terms of squared Euclidean distance. The structure of deep autoencoder (as displayed in Fig. 1.2) is such that

$$\begin{aligned} y^l &= \mathcal{G}_l(y^{l-1}), \\ &= \left[\,\mathcal{F}_1^l(P^l y^{l-1})\ \cdots\ \mathcal{F}_p^l(P^l y^{l-1})\,\right]^T \end{aligned}$$

where $y^0 = y$, $P^l \in \mathbb{R}^{n_l \times p}$ is a matrix with $n_l \in \{1, \cdots, p\}$ such that $n_1 \geq n_2 \geq \cdots \geq n_L$, and $\mathcal{F}_j^l(\cdot)$ is a Student-t membership-mapping.

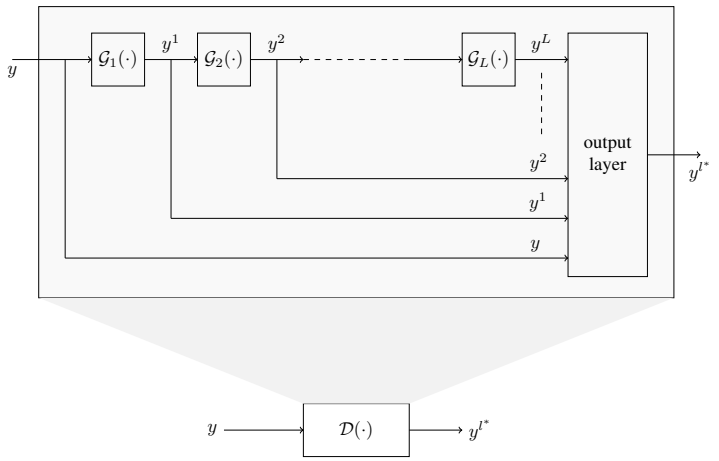

Figure 1.2: The structure of an $L-$layered conditionally deep autoencoder consisting of a nested compositions of membership-mapping autoencoders.

1.2.2 Variational Learning of Membership-Mappings

1.2.2.1 A Modeling Scenario

Given a dataset $\{(x^i, y^i) \mid x^i \in \mathbb{R}^n,\ y^i \in \mathbb{R}^p,\ i \in \{1, \cdots, N\}\}$, it is assumed that there exist zero-mean Student-t membership-mappings $\mathcal{F}_1, \cdots, \mathcal{F}_p \in \mathbb{F}(\mathbb{R}^n)$ such that

$$y^i \approx \left[\, \mathcal{F}_1(x^i) \ \cdots \ \mathcal{F}_p(x^i) \,\right]^T. \tag{1.27}$$

1.2.2.2 Disturbances and Auxiliary Inducing Points

For $j \in \{1, 2, \cdots, p\}$, define

$$\mathrm{y}_j = \left[\, y_j^1 \ \cdots \ y_j^N \,\right]^T \in \mathbb{R}^N \tag{1.28}$$

$$\mathrm{f}_j = \left[\, \mathcal{F}_j(x^1) \ \cdots \ \mathcal{F}_j(x^N) \,\right]^T \in \mathbb{R}^N \tag{1.29}$$

where y_j^i denotes the $j-$th element of y^i. The vectors y_j and f_j will be subsequently referred to as *data* and *output* of membership-mappings, respectively. The difference between data and membership-mappings' outputs will be referred to as *disturbance* and denoted by v_j, i.e.,

$$\mathrm{v}_j = \mathrm{y}_j - \mathrm{f}_j. \tag{1.30}$$

A set of auxiliary inducing points, $\mathrm{a} = \{a^m \in \mathbb{R}^n \mid m \in \{1, \cdots, M\}\}$, is introduced. The membership-mappings' output values at auxiliary inducing input points are collected in a vector defined as

$$\mathrm{u}_j = \left[\, \mathcal{F}_j(a^1) \ \cdots \ \mathcal{F}_j(a^M) \,\right]^T \in \mathbb{R}^M. \tag{1.31}$$

1.2.2.3 Membership Functional Representation Approach

Definition 4 (Membership Functional Representation of Variables) *A variable $y \in \mathbf{Y}$ is represented by means of a membership function $\mu_y : \mathbf{Y} \to [0, 1]$, where the pair $(\mathbf{Y}, \mu_y)$ constitutes a fuzzy set and $\mu_y(\tilde{y})$ is interpreted as the degree to which a point $\tilde{y} \in \mathbf{Y}$ matches an attribute induced by $y \in \mathbf{Y}$.*

Definition 5 (Disturbance-Model) *Disturbance vector v_j is represented by means of a zero-mean Gaussian membership function as*

$$\mu_{\mathrm{v}_j}(\tilde{\mathrm{v}}_j) = \exp\left(-0.5\beta \|\tilde{\mathrm{v}}_j\|^2\right) \tag{1.32}$$

where $\beta > 0$ is the precision.

Definition 6 (Representation of Data y_j for Given Mappings Output f_j) *Since $\mathrm{y}_j = \mathrm{f}_j + \mathrm{v}_j$, it follows from (1.32) that y_j, for given f_j, is represented by means of a membership function, $\mu_{\mathrm{y}_j;\mathrm{f}_j} : \mathbb{R}^N \to [0, 1]$, as*

$$\mu_{\mathrm{y}_j;\mathrm{f}_j}(\tilde{\mathrm{y}}_j) = \exp\left(-0.5\beta \|\tilde{\mathrm{y}}_j - \mathrm{f}_j\|^2\right). \tag{1.33}$$

Definition 7 (Representation of Mappings Output f_j Based on Interpolation) *f_j, based upon an interpolation on the auxiliary-outputs u_j, is represented by means of a membership function, $\mu_{\mathrm{f}_j;\mathrm{u}_j} : \mathbb{R}^N \rightarrow [0,1]$, as*

$$\mu_{\mathrm{f}_j;\mathrm{u}_j}(\tilde{\mathrm{f}}_j) = \left(1 + \frac{1}{\nu + M - 2}(\tilde{\mathrm{f}}_j - \bar{m}_{\mathrm{f}_j})^T \left(\frac{\nu + (\mathrm{u}_j)^T (K_{\mathrm{aa}})^{-1} \mathrm{u}_j - 2}{\nu + M - 2} \bar{K}_{\mathrm{xx}}\right)^{-1} (\tilde{\mathrm{f}}_j - \bar{m}_{\mathrm{f}_j})\right)^{-\frac{\nu+M+N}{2}} \tag{1.34}$$

$$\bar{m}_{\mathrm{f}_j} = K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j \tag{1.35}$$

$$\bar{K}_{\mathrm{xx}} = K_{\mathrm{xx}} - K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T. \tag{1.36}$$

Definition 8 (Representation of Data y_j for Fixed Auxiliary-Outputs u_j) *y_j, for given u_j, is represented by means of a membership function, $\mu_{\mathrm{y}_j;\mathrm{u}_j} : \mathbb{R}^N \rightarrow [0,1]$, as*

$$\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j) \propto \exp\left(\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{f}_j}(\tilde{\mathrm{y}}_j))\right\rangle_{\mu_{\mathrm{f}_j;\mathrm{u}_j}}\right) \tag{1.37}$$

where $\mu_{\mathrm{y}_j;\mathrm{f}_j}$ is given by (1.33), $\mu_{\mathrm{f}_j;\mathrm{u}_j}$ is defined as in (1.34), and $< \cdot >$. is the averaging operation as defined in (1.5). Thus, $\mu_{\mathrm{y}_j;\mathrm{u}_j}$ is obtained from $\log(\mu_{\mathrm{y}_j;\mathrm{f}_j})$ after averaging out the variables f_j using its membership function. It is shown in Appendix 4.1 that

$$\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j) \propto \exp\left(-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + (\mathrm{u}_j)^T \hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\tilde{\mathrm{y}}_j) - 0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j + 0.5(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j + \{/(\tilde{\mathrm{y}}_j, \mathrm{u}_j)\}\right) \tag{1.38}$$

where $\hat{K}_{\mathrm{u}_j}$, $\hat{m}_{\mathrm{u}_j}(\tilde{\mathrm{y}}_j)$ are given by (4.89), (4.90) respectively, and $\{/(\tilde{\mathrm{y}}_j, \mathrm{u}_j)\}$ represents all those terms which are independent of both $\tilde{\mathrm{y}}_j$ and u_j. The constant of proportionality in (1.38) is chosen to exclude $(\tilde{\mathrm{y}}_j, \mathrm{u}_j)-$independent terms in the expression for $\mu_{\mathrm{y}_j;\mathrm{u}_j}$, i.e.,

$$\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j) = \exp\left(-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + (\mathrm{u}_j)^T \hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\tilde{\mathrm{y}}_j) - 0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j + 0.5(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j\right). \tag{1.39}$$

Definition 9 (Data-Model) *y_j is represented by means of a membership function, $\mu_{\mathrm{y}_j} : \mathbb{R}^N \rightarrow [0,1]$, as*

$$\mu_{\mathrm{y}_j}(\tilde{\mathrm{y}}_j) \propto \exp\left(\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j))\right\rangle_{\mu_{\mathrm{u}_j}}\right) \tag{1.40}$$

where $\mu_{\mathrm{y}_j;\mathrm{u}_j}$ is given by (1.39) and $\mu_{\mathrm{u}_j} : \mathbb{R}^M \rightarrow [0,1]$ is a membership function representing u_j. Thus, μ_{y_j} is obtained from $\log(\mu_{\mathrm{y}_j;\mathrm{u}_j})$ after averaging out the auxiliary-outputs u_j using membership function μ_{u_j}.

1.2.2.4 Variational Optimization of Data-Model

The data model (1.40) involves the membership function μ_{u_j}. To determine μ_{u_j} for a given y_j, $\log(\mu_{\mathrm{y}_j}(\mathrm{y}_j))$ is maximized w.r.t. μ_{u_j} around an initial guess. The zero-mean Gaussian membership function with covariance as equal to K_{aa} is taken as the initial guess. It follows from (1.40) that maximization of $\log(\mu_{\mathrm{y}_j}(\mathrm{y}_j))$ is equivalent to the maximization of $\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\mathrm{y}_j))\right\rangle_{\mu_{\mathrm{u}_j}}$.

Result 1 *The solution of following maximization problem:*

$$\mu_{\mathrm{u}_j}^* = \arg\max_{\mu_{\mathrm{u}_j}} \left[\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\mathrm{y}_j))\right\rangle_{\mu_{\mathrm{u}_j}} - \left\langle \log(\frac{\mu_{\mathrm{u}_j}(\mathrm{u}_j)}{\exp\left(-0.5(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j\right)})\right\rangle_{\mu_{\mathrm{u}_j}}\right] \tag{1.41}$$

under the fixed integral constraint:

$$\int_{\mathbb{R}^M} \mu_{\mathrm{u}_j}\, \mathrm{d}\lambda^M = C_{\mathrm{u}_j} > 0 \tag{1.42}$$

where the value of C_{u_j} is so chosen such that the maximum possible values of $\mu_{\mathrm{u}_j}^*$ remain as equal to unity, is given as

$$\mu_{\mathrm{u}_j}^*(\mathrm{u}_j) = \exp\left(-0.5\left(\mathrm{u}_j - \hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T \hat{K}_{\mathrm{u}_j}^{-1}\left(\mathrm{u}_j - \hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)\right) \tag{1.43}$$

where $\hat{K}_{\mathrm{u}_j}$ and $\hat{m}_{\mathrm{u}_j}$ are given by (4.89) and (4.90) respectively. The solution of the optimization problem results in

$$\mu_{\mathrm{y}_j}(\tilde{\mathrm{y}}_j) \propto \exp\left(-0.5\beta\left\{\|\tilde{\mathrm{y}}_j\|^2 - 2\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j + \left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right.\right.$$
$$\left.\left. + \left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T \frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu + M - 2}(K_{\mathrm{aa}})^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right\} + \{/(\mathrm{y}_j, \tilde{\mathrm{y}}_j)\}\right) \tag{1.44}$$

where $\{/(\mathrm{y}_j, \tilde{\mathrm{y}}_j)\}$ represents all $(\mathrm{y}_j, \tilde{\mathrm{y}}_j)-$independent terms.

Proof: The proof is provided in Appendix 4.2. ■

The constant of proportionality in (1.44) is chosen to exclude $(\mathrm{y}_j, \tilde{\mathrm{y}}_j)-$independent terms resulting in

$$\mu_{\mathrm{y}_j}(\tilde{\mathrm{y}}_j) = \exp\left(-0.5\beta\left\{\|\tilde{\mathrm{y}}_j\|^2 - 2\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j + \left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right.\right.$$
$$\left.\left. + \left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T \frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu + M - 2}(K_{\mathrm{aa}})^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right\}\right). \tag{1.45}$$

1.2.2.5 Membership-Mapping Output

Definition 10 (Averaged Estimation of Membership-Mapping Output) *$\mathcal{F}_j(x^i)$ (which is the $i-$th element of vector f_j (1.29)) can be estimated as*

$$\widehat{\mathcal{F}_j(x^i)} := \left\langle \langle (\mathrm{f}_j)_i \rangle_{\mu_{\mathrm{f}_j;\mathrm{u}_j}} \right\rangle_{\mu^*_{\mathrm{u}_j}} \tag{1.46}$$

where $(\mathrm{f}_j)_i$ denotes the $i-$th element of f_j, $\mu_{\mathrm{f}_j;\mathrm{u}_j}$ is defined as in (1.34), and $\mu^*_{\mathrm{u}_j}$ is the optimal membership function (1.43) representing u_j. That is, $\mathcal{F}_j(x^i)$, being a function of u_j, is averaged over u_j for an estimation. Let $G(x) \in \mathbb{R}^{1\times M}$ be a vector-valued function defined as

$$G(x) := \left[\, kr(x,a^1) \;\cdots\; kr(x,a^M) \,\right] \tag{1.47}$$

where $kr : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as in (1.17). It is shown in Appendix 4.3 that

$$\widehat{\mathcal{F}_j(x^i)} = \big(G(x^i)\big)\left(K_{\mathrm{xa}}^T K_{\mathrm{xa}} + \frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu+M-2}K_{\mathrm{aa}} + \frac{K_{\mathrm{aa}}}{\beta}\right)^{-1}(K_{\mathrm{xa}})^T \mathrm{y}_j. \tag{1.48}$$

Let $\alpha = [\,\alpha_1 \;\cdots\; \alpha_p\,] \in \mathbb{R}^{M\times p}$ be a matrix with its $j-$th column defined as

$$\alpha_j := \left(K_{\mathrm{xa}}^T K_{\mathrm{xa}} + \frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu+M-2}K_{\mathrm{aa}} + \frac{K_{\mathrm{aa}}}{\beta}\right)^{-1}(K_{\mathrm{xa}})^T \mathrm{y}_j \tag{1.49}$$

so that $\widehat{\mathcal{F}_j(x^i)}$ could be expressed as

$$\widehat{\mathcal{F}_j(x^i)} = \big(G(x^i)\big)\,\alpha_j. \tag{1.50}$$

1.2.2.6 Choice of Parameters

The analytically derived data model (1.45) involves several parameters which are suggested to be chosen as follows:

Auxiliary inducing points: The auxiliary inducing points are suggested to be chosen as the cluster centroids:

$$\mathrm{a} = \{a^m\}_{m=1}^M = cluster\_centroid(\{x^i\}_{i=1}^N, M) \tag{1.51}$$

where $cluster\_centroid(\{x^i\}_{i=1}^N, M)$ represents the k-means clustering on $\{x^i\}_{i=1}^N$.

Degrees of freedom: The degrees of freedom associated to the Student-t membership-mapping $\nu \in \mathbb{R}_+ \setminus [0,2]$ is chosen as

$$\nu = 2.1 \tag{1.52}$$

Parameters $(w_1,\cdots,w_n)$: The parameters $(w_1,\cdots,w_n)$ for kernel function (1.17) are chosen such that w_k (for $k \in \{1,2,\cdots,n\}$) is given as

$$w_k = \left(\max_{1\le i\le N}\left(x_k^i\right) - \min_{1\le i\le N}\left(x_k^i\right)\right)^{-2} \tag{1.53}$$

where x_k^i is the $k-$th element of vector $x^i \in \mathbb{R}^n$.

Parameters M and σ^2: Define a scalar-valued function:

$$\tau(M,\sigma^2) := \frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu+M-2} \tag{1.54}$$

where a is given by (1.51), ν is given by (1.52), and parameters $(w_1,\cdots,w_n)$ (which are required to evaluate the kernel function for computing matrices K_{xx}, K_{aa}, and K_{xa}) are given by (1.53). It follows from the kernel function definition (1.17) that

$$\tau(M,\sigma^2) = \sigma^2\tau(M,1). \tag{1.55}$$

It is further observed from (1.48) that a higher value of τ corresponds to a larger level of data smoothing. Therefore, we consider a criterion for choosing M and σ^2 such that the data smoothing level (i.e. the value of τ) should match

the variance in the data. In particularly, we pose the requirement that τ should be at least as large as the data variance (averaged over dimensions), i.e.,

$$\tau(M, \sigma^2) \geq \frac{1}{p} \sum_{j=1}^{p} \text{var}\left(y_j^1, \cdots, y_j^N\right). \tag{1.56}$$

Using (1.55), the inequality (1.56) can be rewritten as

$$\sigma^2 \geq \frac{1}{\tau(M,1)} \frac{1}{p} \sum_{j=1}^{p} \text{var}\left(y_j^1, \cdots, y_j^N\right). \tag{1.57}$$

The number of auxiliary inducing points M and the parameter σ^2 for kernel function (1.17) are so determined that the inequality (1.57) holds. This can be done via

1. choosing sufficiently low value of M ensuring that $\tau(M, 1)$ remains larger than a small positive value,
2. choosing σ^2 to satisfy the inequality (1.57).

Precision of the disturbance model: The disturbance precision value β is iteratively estimated as the inverse of the mean squared error between data and membership-mappings outputs. That is,

$$\frac{1}{\beta} = \frac{1}{pN} \sum_{j=1}^{p} \sum_{i=1}^{N} \left| y_j^i - \widehat{\mathcal{F}_j(x^i)} \right|^2 \tag{1.58}$$

where $\widehat{\mathcal{F}_j(x^i)}$ is the estimated membership-mapping output given as in (1.50).

1.2.2.7 Learning Algorithm and Prediction

Algorithm 1 is suggested for the variational learning of membership-mappings. The functionality of Algorithm 1 is as follows.

1. The loop between step 4 and step 7 ensures, via gradually decreasing the number of auxiliary points by a factor of 0.9, that $\tau(M, 1)$ is positive with value larger than κ.
2. The positive value of $\tau(M, 1)$ allows steps 9 to 13 to ensure that the inequality (1.57) remains satisfied and thus the level of data smoothing by membership-mappings remains related to the data variance.
3. The loop between step 16 and step 19 iteratively estimates the parameters α and β.

Definition 11 (Membership-Mappings Prediction) *Given the parameters set $\mathbb{M} = \{\alpha, \text{a}, M, \sigma, w\}$ returned by Algorithm 1, the learned membership-mappings could be used to predict output corresponding to any arbitrary input data point $x \in \mathbb{R}^n$ as*

$$\hat{y}(x; \mathbb{M}) = \left[\widehat{\mathcal{F}_1(x)} \ \cdots \ \widehat{\mathcal{F}_p(x)} \right]^T \tag{1.59}$$

where $\widehat{\mathcal{F}_j(x)}$, defined as in (1.50), is the estimated output of $j-$th membership-mapping. It follows from (1.50) that

$$\hat{y}(x; \mathbb{M}) = \alpha^T (G(x))^T \tag{1.60}$$

where $G(\cdot) \in \mathbb{R}^{1 \times M}$ is a vector-valued function (1.47).

1.2.3 Algorithm for Variational Learning of Conditionally Deep Membership-Mapping Autoencoders

Since CDMMA consists of layers of membership-mappings, Algorithm 1 could be directly applied for the variational learning of individual layers. Given a set of N samples $\{y^1, \cdots, y^N\}$, following [5], Algorithm 2 is stated for the variational learning of CDMMA.

Algorithm 2 defines $\{n_1, \cdots, n_L\}$ to be a monotonically decreasing sequence at step 2. This results in CDMMA to discover layers of increasingly abstract data representation with lowest-level data features being modeled by first layer and the highest-level by end layer. Fig. 1.3 illustrates through an example that Algorithm 2 allows high-dimensional data representation learning at varying abstraction levels across CDMMA's different layers.

Algorithm 1 Variational learning of the membership-mappings

Require: Dataset $\left\{(x^i, y^i) \mid x^i \in \mathbb{R}^n,\ y^i \in \mathbb{R}^p,\ i \in \{1, \cdots, N\}\right\}$ and maximum possible number of auxiliary points $M_{max} \in \mathbb{Z}_+$ with $M_{max} \leq N$.

Choose ν and $w = (w_1, \cdots, w_n)$ as in (1.52) and (1.53) respectively.
Choose a small positive value $\kappa = 10^{-1}$.
Set iteration count $it = 0$ and $M|_0 = M_{max}$.
while $\tau(M|_{it}, 1) < \kappa$ **do**
  $M|_{it+1} = \lceil 0.9 M|_{it} \rceil$
  $it \leftarrow it + 1$
end while
Set $M = M|_{it}$.
if $\tau(M, 1) \geq \frac{1}{p} \sum_{j=1}^{p} \text{var}\left(y_j^1, \cdots, y_j^N\right)$ **then**
  $\sigma^2 = 1$
else
  $\sigma^2 = \frac{1}{\tau(M,1)} \frac{1}{p} \sum_{j=1}^{p} \text{var}\left(y_j^1, \cdots, y_j^N\right)$
end if
Compute $\mathrm{a} = \{a^m\}_{m=1}^{M}$ using (1.51), K_{xx} using (1.16), K_{aa} using (1.21), and K_{xa} using (1.22).
Set $\beta = 1$.
repeat
  Compute α using (1.49).
  Update the value of β using (1.58).
until (β nearly converges)
Compute α using (1.49).
return the parameters set $\mathbb{M} = \{\alpha, \mathrm{a}, M, \sigma, w\}$.

Algorithm 2 Variational learning of CDMMA

Require: Data set $\mathbf{Y} = \left\{y^i \in \mathbb{R}^p \mid i \in \{1, \cdots, N\}\right\}$; the subspace dimension $n \in \{1, 2, \cdots, p\}$; maximum number of auxiliary points $M_{max} \in \mathbb{Z}_+$ with $M_{max} \leq N$; the number of layers $L \in \mathbb{Z}_+$.

for $l = 1$ to L **do**
  Set subspace dimension associated to $l-$th layer as $n_l = \max(n - l + 1, 1)$.
  Define $P^l \in \mathbb{R}^{n_l \times p}$ such that $i-$th row of P^l is equal to transpose of eigenvector corresponding to $i-$th largest eigenvalue of sample covariance matrix of data set $\mathbf{Y}$.
  Define a latent variable $x^{l,i} \in \mathbb{R}^{n_l}$, for $i \in \{1, \cdots, N\}$, as

$$x^{l,i} := \begin{cases} P^l y^i & \text{if } l = 1, \\ P^l \hat{y}^{l-1}(x^{l-1,i}; \mathbb{M}^{l-1}) & \text{if } l > 1 \end{cases} \tag{1.61}$$

  where $\hat{y}^{l-1}$ is the estimated output of the $(l-1)-$th layer computed using (1.60) for the parameters set $\mathbb{M}^{l-1} = \{\alpha^{l-1}, \mathrm{a}^{l-1}, M^{l-1}, \sigma^{l-1}, w^{l-1}\}$.
  Define M_{max}^l as

$$M_{max}^l := \begin{cases} M_{max} & \text{if } l = 1, \\ M^{l-1} & \text{if } l > 1 \end{cases} \tag{1.62}$$

  Compute parameters set $\mathbb{M}^l = \{\alpha^l, \mathrm{a}^l, M^l, \sigma^l, w^l\}$, characterizing the membership-mappings associated to $l-$th layer, using Algorithm 1 on data set $\left\{(x^{l,i}, y^i) \mid i \in \{1, \cdots, N\}\right\}$ with maximum possible number of auxiliary points M_{max}^l.
end for
return the parameters set $\mathcal{M} = \{\{\mathbb{M}^1, \cdots, \mathbb{M}^L\}, \{P^1, \cdots, P^L\}\}$.

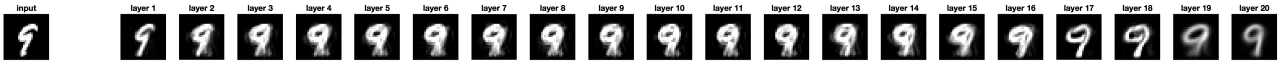

Figure 1.3: A CDMMA was built using Algorithm 2 (taking $n = 20$; $M_{max} = 500$; $L = 20$) on a dataset consisting of 1000 randomly chosen samples of digit 9 from MNIST digits dataset. Corresponding to the input sample (shown at the extreme left of the figure), the estimated outputs of different layers of CDMMA are displayed.

Definition 12 (CDMMA Filtering) *Given a CDMMA with its parameters being represented by a set,*

$$\mathcal{M} = \{\{\mathbb{M}^1, \cdots, \mathbb{M}^L\}, \{P^1, \cdots, P^L\}\},$$

the autoencoder can be applied for filtering a given input vector $y \in \mathbb{R}^p$ as follows:

$$x^l(y; \mathcal{M}) = \begin{cases} P^l y, & l = 1 \\ P^l \hat{y}^{l-1}(x^{l-1}; \mathbb{M}^{l-1}) & l \geq 2 \end{cases} \tag{1.63}$$

Here, $\hat{y}^{l-1}$ is the output of the $(l-1)-$th layer estimated using (1.60). Finally, CDMMA's output, $\mathcal{D}(y; \mathcal{M})$, is given as

$$\widehat{\mathcal{D}}(y; \mathcal{M}) = \hat{y}^{l^*}(x^{l^*}; \mathbb{M}^{l^*}) \tag{1.64}$$

$$l^* = \arg \min_{l \in \{1, \cdots, L\}} \|y - \hat{y}^l(x^l; \mathbb{M}^l)\|^2. \tag{1.65}$$

1.2.4 Wide Conditionally Deep Membership-Mapping Autoencoder

For big datasets, a wide form of conditionally deep autoencoder has been suggested [5] where the total data is partitioned into subsets and corresponding to each data-subset a separate CDMMA is learned. The final output is equal to the output of the CDMMA re-constructing the given input vector as good as possible where re-construction error is measured in-terms of squared Euclidean distance.

Definition 13 (A Wide CDMMA) *A wide CDMMA, $\mathcal{WD} : \mathbb{R}^p \to \mathbb{R}^p$, maps a vector $y \in \mathbb{R}^p$ to $\mathcal{WD}(y) \in \mathbb{R}^p$ through a parallel composition of S ($S \in \mathcal{Z}_+$) number of CDMMAs such that*

$$\mathcal{WD}(y) = \mathcal{D}_{s^*}(y) \tag{1.66}$$

$$s^* = \arg \min_{s \in \{1, 2, \cdots, S\}} \|y - \mathcal{D}_s(y)\|^2, \tag{1.67}$$

where $\mathcal{D}_s(y)$ is the output of $s-$th CDMMA.

To formally define an algorithm for the variational learning of wide CDMMA, the ratio of maximum number of auxiliary points to the number of data points is defined:

$$r_{max} = \frac{M_{max}}{N}. \tag{1.68}$$

Following [5], Algorithm 3 is suggested for the variational learning of wide CDMMA.

Algorithm 3 Variational learning of wide CDMMA

Require: Data set $\mathbf{Y} = \{y^i \in \mathbb{R}^p \mid i \in \{1, \cdots, N\}\}$; the subspace dimension $n \in \{1, 2, \cdots, p\}$; ratio $r_{max} \in (0, 1]$; the number of layers $L \in \mathbb{Z}_+$.

1: Apply k-means clustering to partition $\mathbf{Y}$ into S subsets, $\{\mathbf{Y}^1, \cdots, \mathbf{Y}^S\}$, where $S = \lceil N/1000 \rceil$.
2: **for** $s = 1$ to S **do**
3: Build a CDMMA, $\mathcal{M}^s$, by applying Algorithm 2 on $\mathbf{Y}^s$ taking n as the subspace dimension; maximum number of auxiliary points as equal to $r_{max} \times \#\mathbf{Y}^s$ (where $\#\mathbf{Y}^s$ is the number of data points in $\mathbf{Y}^s$); and L as the number of layers.
4: **end for**
5: **return** the parameters set $\mathcal{P} = \{\mathcal{M}^s\}_{s=1}^S$.

Definition 14 (Wide CDMMA Filtering) *Given a wide CDMMA with its parameters being represented by a set $\mathcal{P} = \{\mathcal{M}^s\}_{s=1}^S$, the autoencoder can be applied for filtering a given input vector $y \in \mathbb{R}^p$ as follows:*

$$\widehat{\mathcal{WD}}(y; \mathcal{P}) = \widehat{\mathcal{D}}(y; \mathcal{M}^{s^*}) \tag{1.69}$$

$$s^* = \arg \min_{s \in \{1, 2, \cdots, S\}} \|y - \widehat{\mathcal{D}}(y; \mathcal{M}^s)\|^2, \tag{1.70}$$

where $\widehat{\mathcal{D}}(y; \mathcal{M}^s)$ is the output of $s-$th CDMMA estimated using (1.64).

Algorithm 3 requires choosing the values for subspace dimension n and ratio r_{max}. Thus, we demonstrate the effect of n and r_{max} on the abstraction level of data representation through examples in Fig. 1.4 and Fig. 1.5.

Figure 1.4: On a dataset consisting of 1000 randomly chosen samples of digit 8 from MNIST digits dataset, different wide CDMMAs were built using Algorithm 3 choosing $r_{max} = 0.5$, $L = 5$, and n from $\{20, 19, 17, 15, 13, 11, 9, 7, 5, 3, 1\}$. Corresponding to the input sample (shown at the extreme left of the figure), the estimated outputs of different wide CDMMAs (built using different values of n) are displayed. It is observed that as n keeps on decreasing, the autoencoder learns increasingly abstract data representation.

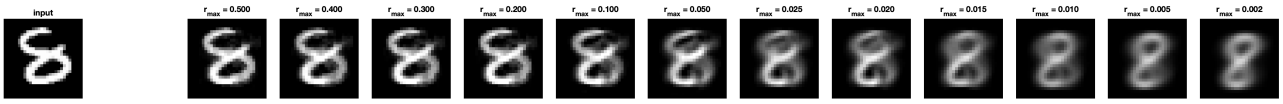

Figure 1.5: On a dataset consisting of 1000 randomly chosen samples of digit 8 from MNIST digits dataset, different wide CDMMAs were built using Algorithm 3 choosing r_{max} from $\{0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.025, 0.02, 0.015, 0.01, 0.005, 0.002\}$, $n = 20$, and $L = 5$. Corresponding to the input sample (shown at the extreme left of the figure), the estimated outputs of different wide CDMMAs (built using different r_{max} values) are displayed. It is observed that as the r_{max} value keeps on decreasing, the autoencoder learns increasingly abstract data representation.

1.2.5 Classification Applications

The autoencoders could be applied for classification via learning data representation for each class through a separate autoencoder. Formally, the classifier is defined as in Definition 15 and Algorithm 4 is stated for the variational learning of classifier.

Definition 15 (A Classifier) *A classifier, $\mathcal{C} : \mathbb{R}^p \to \{1, 2, \cdots, C\}$, maps a vector $y \in \mathbb{R}^p$ to $\mathcal{C}(y) \in \{1, 2, \cdots, C\}$ such that*

$$\mathcal{C}(y; \{\mathcal{P}_c\}_{c=1}^{C}) = \arg \min_{c \in \{1,2,\cdots,C\}} \|y - \widehat{\mathcal{WD}}(y; \mathcal{P}_c)\|^2 \tag{1.71}$$

where $\widehat{\mathcal{WD}}(y; \mathcal{P}_c)$, computed using (1.69), is the output of $c-$th wide CDMMA. The classifier assigns to an input vector the label of that class whose associated autoencoder best reconstructs the input vector.

Algorithm 4 Variational learning of the classifier

Require: Labeled data set $\mathbf{Y} = \left\{\mathbf{Y}_c \mid \mathbf{Y}_c = \left\{y^{i,c} \in \mathbb{R}^p \mid i \in \{1, \cdots, N_c\}\right\},\ c \in \{1, \cdots, C\}\right\}$; the subspace dimension $n \in \{1, \cdots, p\}$; ratio $r_{max} \in (0, 1]$; the number of layers $L \in \mathbb{Z}_+$.

1: **for** $c = 1$ to C **do**
2: Build a wide CDMMA, $\mathcal{P}_c = \{\mathcal{M}_c^s\}_{s=1}^{S_c}$, by applying Algorithm 3 on $\mathbf{Y}_c$ for the given n, r_{max}, and L.
3: **end for**
4: **return** the parameters set $\{\mathcal{P}_c\}_{c=1}^{C}$.

1.3 Privacy-Preserving Transferrable Deep Learning

1.3.1 An Optimal $(\epsilon, \delta)-$Differentially Private Noise Adding Mechanism

This subsection reviews an optimal noise adding mechanism that was derived using an information theoretic approach in [6]. We consider a training dataset consisting of N number of samples with each sample having p number of attributes. Assuming the data as numeric, the dataset can be represented by a matrix, say $\mathrm{Y} \in \mathbb{R}^{p \times N}$. The machine learning algorithms typically train a model using available dataset. A given machine learning algorithm, training a model using data matrix Y, can be represented by a mapping, $\mathcal{A} : \mathbb{R}^{p \times N} \to \mathbf{M}$, where $\mathbf{M}$ is the model space. That is, for a given dataset Y, the algorithm builds a model $\mathcal{M} \in \mathbf{M}$ such that $\mathcal{M} = \mathcal{A}(\mathrm{Y})$. The privacy of data can be preserved via adding a suitable random noise to data matrix before the application of algorithm $\mathcal{A}$ on the dataset. This will result in a private version of algorithm $\mathcal{A}$ which is formally defined by Definition 16.

Definition 16 (A Private Algorithm on Data Matrix) *Let $\mathcal{A}^+ : \mathbb{R}^{p\times N} \to Range(\mathcal{A}^+)$ be a mapping defined as*

$$\mathcal{A}^+(\mathrm{Y}) = \mathcal{A}(\mathrm{Y}+\mathrm{V}),\ \mathrm{V}\in\mathbb{R}^{p\times N} \tag{1.72}$$

where V is a random noise matrix with $f_{\mathrm{v}_j^i}(v)$ being the probability density function of its $(j,i)-$th element v_j^i; v_j^i and $\mathrm{v}_j^{i'}$ are independent from each other for $i\neq i'$; and $\mathcal{A}:\mathbb{R}^{p\times N}\to\mathbf{M}$ (where $\mathbf{M}$ is the model space) is a given mapping representing a machine learning algorithm. The range of $\mathcal{A}^+$ is as

$$Range(\mathcal{A}^+) = \left\{\mathcal{A}(\mathrm{Y}+\mathrm{V}) \mid \mathrm{Y}\in\mathbb{R}^{p\times N}, \mathrm{V}\in\mathbb{R}^{p\times N}\right\}. \tag{1.73}$$

We intend to protect the algorithm $\mathcal{A}^+$ from an adversary who seeks to gain an information about the data from algorithm's output by perturbing the values in a sample of the dataset. We seek to attain differential privacy for algorithm $\mathcal{A}^+$ against the perturbation in an element of Y, say $(j_0,i_0)-$th element, such that magnitude of the perturbation is upper bounded by a scalar d. Following [7], the $d-$adjacency and $(\epsilon,\delta)-$differential privacy definitions are provided in Definition 17 and Definition 18 respectively.

Definition 17 ($d-$Adjacency for Data Matrices) *Two matrices $\mathrm{Y},\mathrm{Y}'\in\mathbb{R}^{p\times N}$ are $d-$adjacent if for a given $d\in\mathbb{R}_+$, there exist $i_0\in\{1,2,\cdots,N\}$ and $j_0\in\{1,2,\cdots,p\}$ such that $\forall i\in\{1,2,\cdots,N\}, j\in\{1,2,\cdots,p\}$,*

$$\left|\mathrm{y}_j^i - \mathrm{y}_j'^i\right| \leq \begin{cases} d, & \text{if } i=i_0, j=j_0 \\ 0, & \text{otherwise} \end{cases}$$

where y_j^i and $\mathrm{y}_j'^i$ denote the $(j,i)-$th element of Y and Y' respectively. Thus, Y and Y' differ by only one element and the magnitude of the difference is upper bounded by d.

Definition 18 ($(\epsilon,\delta)-$Differential Privacy for $\mathcal{A}^+$ [6]) *The algorithm $\mathcal{A}^+(\mathrm{Y})$ is $(\epsilon,\delta)-$differentially private if*

$$Pr\{\mathcal{A}^+(\mathrm{Y})\in\mathcal{O}\} \leq \exp(\epsilon) Pr\{\mathcal{A}^+(\mathrm{Y}'))\in\mathcal{O}\} + \delta \tag{1.74}$$

for any measurable set $\mathcal{O}\subseteq Range(\mathcal{A}^+)$ and for $d-$adjacent matrices pair (Y,Y').

Intuitively, Definition 18 means that changing the value of an element in the training data matrix by an amount upper bounded by d can change the distribution of output of the algorithm $\mathcal{A}^+$ only by a factor of $\exp(\epsilon)$ with probability at least $1-\delta$. Thus, the lower value of ϵ and δ lead to a higher amount of privacy.

Result 2 (An Optimal $(\epsilon,\delta)-$Differentially Private Noise [6]) *The probability density function of noise, that minimizes the expected noise magnitude together with satisfying the sufficient conditions for $(\epsilon,\delta)-$differential privacy for $\mathcal{A}^+$, is given as*

$$f^*_{\mathrm{v}_j^i}(v;\epsilon,\delta,d) = \begin{cases} \delta\, Dirac\delta(v), & v=0 \\ (1-\delta)\frac{\epsilon}{2d}\exp(-\frac{\epsilon}{d}|v|), & v\in\mathbb{R}\setminus\{0\} \end{cases} \tag{1.75}$$

where $Dirac\delta(v)$ is Dirac delta function satisfying $\int_{-\infty}^{\infty} Dirac\delta(v)\ \mathrm{d}v = 1$.

Proof: The proof follows from [6]. ■

Remark 1 (Generating Random Samples from $f^*_{\mathrm{v}_j^i}$) *The method of inverse transform sampling can be used to generate random samples from cumulative distribution function. The cumulative distribution function of $f^*_{\mathrm{v}_j^i}$ is given as*

$$F_{\mathrm{v}_j^i}(v;\epsilon,\delta,d) = \begin{cases} \frac{1-\delta}{2}\exp(\frac{\epsilon}{d}v), & v<0 \\ \frac{1+\delta}{2}, & v=0 \\ 1-\frac{1-\delta}{2}\exp(-\frac{\epsilon}{d}v), & v>0 \end{cases} \tag{1.76}$$

The inverse cumulative distribution function is given as

$$F^{-1}_{\mathrm{v}_j^i}(t_j^i;\epsilon,\delta,d) = \begin{cases} \frac{d}{\epsilon}\log(\frac{2t_j^i}{1-\delta}), & t_j^i < \frac{1-\delta}{2} \\ 0, & t_j^i\in[\frac{1-\delta}{2},\frac{1+\delta}{2}] \\ -\frac{d}{\epsilon}\log(\frac{2(1-t_j^i)}{1-\delta}), & t_j^i > \frac{1+\delta}{2} \end{cases},\ t_j^i\in(0,1). \tag{1.77}$$

Thus, via generating random samples from the uniform distribution on $(0,1)$ and using (1.77), the noise additive mechanism can be implemented.

For a given value of (ϵ,δ,d), Algorithm 5 is stated for a differentially private approximation of a data samples.

Algorithm 5 Differentially private approximation of data samples

Require: Data set $\mathbf{Y} = \{y^i \in \mathbb{R}^p \mid i \in \{1, \cdots, N\}\}$; differential privacy parameters: $d \in \mathbb{R}_+, \epsilon \in \mathbb{R}_+, \delta \in (0, 1)$.

1: A differentially private approximation of data samples is provided as

$$y_j^{+i} = y_j^i + F^{-1}_{\mathrm{v}_j^i}(t_j^i; \epsilon, \delta, d),\ t_j^i \in (0, 1) \tag{1.78}$$

where $F^{-1}_{\mathrm{v}_j^i}$ is given by (1.77) and y_j^{+i} is $j-$th element of $y^{+i} \in \mathbb{R}^p$.

2: **return** $\mathbf{Y}^+ = \{y^{+i} \in \mathbb{R}^p \mid i \in \{1, \cdots, N\}\}$.

1.3.2 Differentially Private Semi-Supervised Transfer and Multi-Task Learning

We consider a scenario of knowledge transfer from a dataset consisting of labeled samples from a domain (referred to as source domain) to another dataset consisting of mostly unlabelled samples and only a few labelled samples from another domain (referred to as target domain) such that both source and target datasets have been sampled from the same set of classes but in their respective domains. The aim is to transfer the knowledge extracted by a classifier trained using source dataset to the classifier of target domain such that privacy of source dataset is preserved. Let $\{\mathbf{Y}_c^{sr}\}_{c=1}^C$ be the labelled source dataset where $\mathbf{Y}_c^{sr} = \{y_{sr}^{i,c} \in \mathbb{R}^{p_{sr}} \mid i \in \{1, \cdots, N_c^{sr}\}\}$ represents $c-$th labelled samples. The target dataset consist of a few labelled samples $\{\mathbf{Y}_c^{tg}\}_{c=1}^C$ (with $\mathbf{Y}_c^{tg} = \{y_{tg}^{i,c} \in \mathbb{R}^{p_{tg}} \mid i \in \{1, \cdots, N_c^{tg}\}\}$) and another set of unlabelled samples $\mathbf{Y}_*^{tg} = \{y_{tg}^{i,*} \in \mathbb{R}^{p_{tg}} \mid i \in \{1, \cdots, N_*^{tg}\}\}$. A generalized setting is considered where source and target data dimensions could be different, i.e., $p_{sr} \neq p_{tg}$. Our approach to semi-supervised transfer and multi-task learning consists of following steps:

Differentially private source domain classifier: Since the noise adding mechanism (i.e. Result 2) is independent of the choice of algorithm operating on training data matrix, therefore any algorithm operating on noise added data samples will remain $(\epsilon, \delta)-$differentially private. That is, differential privacy remains invariant to any post-processing of noise added data samples. This allows us to build a differentially private classifier as stated in Algorithm 6.

Algorithm 6 Variational learning of a differentially private classifier

Require: Differentially private approximated dataset: $\mathbf{Y}^+ = \{\mathbf{Y}_c^+ \mid c \in \{1, \cdots, C\}\}$; the subspace dimension $n \in \{1, \cdots, p\}$; ratio $r_{max} \in (0, 1]$; the number of layers $L \in \mathbb{Z}_+$.

1: Build a classifier, $\{\mathcal{P}_c^+\}_{c=1}^C$, by applying Algorithm 4 on $\mathbf{Y}^+$ for the given n, r_{max}, and L.

2: **return** $\{\mathcal{P}_c^+\}_{c=1}^C$.

For a given differential privacy parameters: d, ϵ, δ; Algorithm 5 is applied on $\mathbf{Y}_c^{sr}$ to obtain the differentially private approximated data samples, $\mathbf{Y}_c^{+sr} = \{y_{sr}^{+i,c} \in \mathbb{R}^{p_{sr}} \mid i \in \{1, \cdots, N_c^{sr}\}\}$, for all $c \in \{1, \cdots, C\}$. Algorithm 6 is applied on $\{\mathbf{Y}_c^{+sr}\}_{c=1}^C$ to build a differentially private source domain classifier characterized by parameters sets $\{\mathcal{P}_c^{+sr}\}_{c=1}^C$.

Differentially private source domain latent subspace transformation-matrix For a lower-dimensional representation of both source and target samples, a subspace dimension, $n_{st} \in \{1, 2, \cdots, \min(p_{sr}, p_{tg})\}$, is chosen. Let $V^{+sr} \in \mathbb{R}^{n_{st} \times p_{sr}}$ be the transformation-matrix with its $i-$th row equal to transpose of eigenvector corresponding to $i-$th largest eigenvalue of sample covariance matrix computed on source samples.

Target domain latent subspace transformation-matrix Let $V^{tg} \in \mathbb{R}^{n_{st} \times p_{tg}}$ be the transformation-matrix with its $i-$th row equal to transpose of eigenvector corresponding to $i-$th largest eigenvalue of sample covariance matrix computed on target samples.

Subspace alignment for heterogenous domains For the case of heterogenous source and target domains (i.e. $p_{sr} \neq p_{tg}$), we follow *subspace alignment* approach where a target sample is first aligned to source data in subspace followed by a linear transformation to source-data-space. A target sample can be mapped to source-data-space via following transformation:

$$y_{tg \to sr} := \begin{cases} y_{tg}, & p_{sr} = p_{tg} \\ (V^{+sr})^T V^{tg} y_{tg}, & p_{sr} \neq p_{tg} \end{cases} \tag{1.79}$$

Both labelled and unlabelled target datasets are transformed to define the following sets:

$$\mathbf{Y}_c^{tg \to sr} := \{y_{tg \to sr} \mid y_{tg} \in \mathbf{Y}_c^{tg}\} \tag{1.80}$$

$$\mathbf{Y}_*^{tg \to sr} := \{y_{tg \to sr} \mid y_{tg} \in \mathbf{Y}_*^{tg}\}. \tag{1.81}$$

Building of target domain classifier Our idea is to iteratively build target domain classifier for predicting the labels of unlabelled target data samples. The $k-$th iteration, for $k \in \{1, \cdots, it\_max\}$, consists of following updates:

$$\{\mathcal{P}_c^{tg}|_k\}_{c=1}^C = \text{Algorithm 4}\left(\left\{\mathbf{Y}_c^{tg\to sr} \cup \mathbf{Y}_{*,c}^{tg\to sr}|_{k-1}\right\}_{c=1}^C, n|_k, r_{max}, L\right) \tag{1.82}$$

$$\mathbf{Y}_{*,c}^{tg\to sr}|_k = \left\{y_{tg\to sr}^{i,*} \in \mathbf{Y}_*^{tg\to sr} \mid \mathcal{C}(y_{tg\to sr}^{i,*}; \{\mathcal{P}_c^{tg}|_k\}_{c=1}^C) = c,\ i \in \{1, \cdots, N_*^{tg}\}\right\} \tag{1.83}$$

where $\{n|_1, n|_2, \cdots\}$ is a monotonically non-decreasing sequence. The reason for n to follow a monotonically non-decreasing curve during the iterations is following:

> *We intend to use higher-level data features during initial iterations for updating the predicted-labels of unlabeled target data samples and as the number of iterations increases more and more lower-level data features are intended to be included in the process of updating the predicted-labels. Since the lower values of n lead to modeling of higher-level data features and higher values lead to modeling of lower-level data features (as illustrated in Fig. 1.4), n values are chosen as to form a monotonically non-decreasing sequence.*

source2target model The target samples associated to a class can be filtered through the source domain autoencoder associated to the same class for defining the following dataset:

$$\mathcal{D} := \left\{\left(\widehat{\mathcal{WD}}(y; \mathcal{P}_c^{+sr}), y\right) \mid y \in \left\{\mathbf{Y}_c^{tg\to sr} \cup \mathbf{Y}_{*,c}^{tg\to sr}|_{it\_max}\right\},\ c \in \{1, \cdots, C\}\right\} \tag{1.84}$$

where $\widehat{\mathcal{WD}}(\cdot;\cdot)$ is defined as in (1.69), $\mathbf{Y}_c^{tg\to sr}$ is defined as in (1.80), and $\mathbf{Y}_{*,c}^{tg\to sr}$ is defined as in (1.83). Here, $\widehat{\mathcal{WD}}(y; \mathcal{P}_c^{+sr})$, where $y \in \{\mathbf{Y}_c^{tg\to sr} \cup \mathbf{Y}_{*,c}^{tg\to sr}|_{it\_max}\}$, is a representation of a $c-$th labelled traget sample y in the source domain $c-$th labelled data space represented by the wide CDMMA $\mathcal{P}_c^{+sr}$. The mapping from source to target domain can be learned via building a variational membership-mappings based model using Algorithm 1 on the dataset $\mathcal{D}$. That is,

$$\mathbb{M}^{sr\to tg} = \text{Algorithm 1}\ (\mathcal{D}, M_{max}) \tag{1.85}$$

$$M_{max} = \min(\lceil N^{tg}/2 \rceil, 1000) \tag{1.86}$$

where $N^{tg} = |\mathcal{D}|$ is the total number of target samples.

A transfer and multi-task learning scenario: Both source and target domain classifiers are combined with source2target model for predicting the label associated to a target sample $y_{tg\to sr}$ as

$$\begin{aligned}\hat{c}(y_{tg\to sr}; \{\mathcal{P}_c^{tg}\}_{c=1}^C, \{\mathcal{P}_c^{+sr}\}_{c=1}^C, \mathbb{M}^{sr\to tg}) = \arg \min_{c\,\in\,\{1,2,\cdots,C\}} \Bigg\{ \min \Bigg(&\left\| y_{tg\to sr} - \widehat{\mathcal{WD}}(y_{tg\to sr}; \mathcal{P}_c^{tg}) \right\|^2, \\ &\left\| y_{tg\to sr} - \hat{y}\left(\widehat{\mathcal{WD}}(y_{tg\to sr}; \mathcal{P}_c^{+sr}); \mathbb{M}^{sr\to tg}\right) \right\|^2, \\ &\left\| y_{tg\to sr} - \widehat{\mathcal{WD}}(y_{tg\to sr}; \mathcal{P}_c^{+sr}) \right\|^2 \Bigg) \Bigg\}. \end{aligned} \tag{1.87}$$

where $\hat{y}\left(\cdot; \mathbb{M}^{sr\to tg}\right)$ is the output of source2target model computed using (1.60). That is, $y_{tg\to sr}$ is assigned the $c-$th class label, if

- the autoencoder associated to $c-$th class of target data space (which is characterized by set of parameters $\mathcal{P}_c^{tg}$) could best reconstruct $y_{tg\to sr}$, or
- the output of the source2target model with the input as representation of $y_{tg\to sr}$ in source domain $c-$th labelled data space could best reconstruct $y_{tg\to sr}$, or
- the differentially private autoencoder associated to $c-$th class of source data space (which is characterized by set of parameters $\mathcal{P}_c^{+sr}$) could best reconstruct $y_{tg\to sr}$.

1.4 Experiments

Differentially private transferrable learning methodology was implemented using MATLAB R2017b. The experiments have been made on an iMac (M1, 2021) machine with 8 GB RAM. The implementational details for the method are described as below:

- We study experimentally the differential privacy (Definition 18) of the source domain training data such that for all $1-$adjacent training data matrices, the absolute value of privacy-loss incurred by observing the output of any computation algorithm will be bounded by ϵ with probability at least $1 - \delta$. That is, d is taken as equal to 1 for defining adjacent matrices in Definition 18.

Figure 1.6: An example of the noise added to a randomly selected sample from MNIST dataset for different values of ϵ with $\delta = 1\text{e}{-5}$.

- Algorithm 5, for a given d and (ϵ, δ), is applied to obtain a differentially private approximation of source dataset.
- Differentially private source domain classifier is built using Algorithm 6 taking subspace dimension as equal to $\min(20, p_{sr})$ (where p_{sr} is the dimension of source data samples), ratio r_{max} as equal to 0.5, and number of layers as equal to 5.
- Differentially private source domain latent subspace transformation-matrix is computed with $n_{st} = \min(\lceil p_{sr}/2 \rceil, p_{tg})$, where p_{tg} is the dimension of target data samples.
- Initial target domain classifier is built using Algorithm 4 on labelled target samples taking subspace dimension as equal to $\min(20, \min_{1 \leq c \leq C}\{N_c^{tg}\} - 1)$ (where N_c^{tg} is the number of $c-$th class labelled target samples), ratio r_{max} as equal to 1, and number of layers as equal to 1.
- The target domain classifier is updated using (1.82) and (1.83) till 4 iterations taking the monotonically non-decreasing subspace dimension n sequence as $\{5, 10, 15, 20\}$ and $r_{max=0.5}$.
- The label associated to a target data point is predicted under transfer and multi-task learning scenarios using (1.87).

1.4.1 Demonstrative Examples Using MNIST and USPS Datasets

1.4.1.1 MNIST dataset

Our first experiment is on the widely used MNIST digits dataset containing 28×28 sized images divided into training set of 60000 images and testing set of 10000 images. The images' pixel values were divided by 255 to normalize the values in the range from 0 to 1. The 28×28 normalized values of each image were flattened to an equivalent $784-$dimensional data vector. The transfer learning experiment was carried in the same setting as in [8] where 60000 training samples constituted the source dataset; a set of 9000 test samples constituted target dataset, and the performance was evaluated on the remaining 1000 test samples. Out of 9000 target samples, only 10 samples per class were labelled and rest 8900 target samples remained as unlabelled.

What is the sufficiently low value of privacy-loss bound? A lower privacy-loss bound implies a larger amount of noise being added to data samples. For an interpretation of the privacy-loss bound ϵ in terms of amount of noise required to be added to preserve data's privacy, the examples of noise added samples corresponding to different values of ϵ are provided in Fig. 1.6. It is observed from Fig. 1.6 that ϵ more than 1 is not sufficiently low to preserve privacy in this case. Thus, the experiments were carried out with privacy-loss bound $\epsilon \in \{0.1, 0.2, 0.5, 1\}$ while keeping failure probability fixed at $\delta = 1\text{e}{-5}$.

Robust performance Table 1.1 reports the values of $(\epsilon, \delta)-$differential privacy guarantees and corresponding classification accuracies. The proposed method's consistent performance over a wide range of privacy-loss bound ϵ verifies the robustness towards the perturbations in source data caused by the privacy requirements demanded by source data owner. For a comparison, the proposed method is able to learn 95.1% accurate model together with providing $(0.1, 1\text{e}{-5})-$differential privacy guarantee, which is a better result than the existing result [9] of achieving 90% accuracy for $(0.5, 1\text{e}{-5})-$differential privacy on MNIST dataset.

1.4.1.2 Learning across heterogeneous MNIST and USPS domains

We considered a problem of heterogeneous transfer learning between MNIST to USPS dataset. The USPS is another dataset that has 7291 training and 2007 test images of digits where each image has 16×16 (=256) grayscale pixels.

Table 1.1: Privacy and utility results on MNIST dataset. The second column reports the privacy-loss bound ϵ and failure probability δ of $(\epsilon, \delta)-$differential privacy guarantee.

Method	(ϵ, δ)	**classification accuracy**
proposed transfer and multi-task learning	$(0.1, 1e{-}5)$	95.1%
proposed transfer and multi-task learning	$(0.2, 1e{-}5)$	95.1%
proposed transfer and multi-task learning	$(0.5, 1e{-}5)$	95.1%
proposed transfer and multi-task learning	$(1, 1e{-}5)$	95.1%
[9]	$(0.5, 1e{-}5)$	90%

Table 1.2: Results of 10 independent MNIST→USPS experiments expressed in average accuracy $\pm$ standard deviation.

method	**accuracy (in %)**
$(0.1, 1e{-}5)-$differentially private proposed	92.23 ± 0.87
$(1, 1e{-}5)-$differentially private proposed	92.28 ± 0.62
non private proposed	92.37 ± 0.70
non private Deep Variational Transfer [10]	92.03 ± 0.38

MNIST→USPS The aim of this experiment was to study how privacy-preservation affect transferring knowledge from a higher resolution and more varied MNIST dataset to USPS dataset. The MNIST→USPS semi-supervised transfer learning problem was previously studied in [10]. For a comparison, MNIST→USPS problem was considered in the same experimental setting as in [10] where only 100 target samples were labelled and remaining 7191 samples remained as unlabelled. The experiments were carried out at privacy-loss bound $\epsilon \in \{0.1, 1\}$ while keeping failure probability fixed at $\delta = 1e{-}5$. Further, the non-private version of the proposed method corresponding to the case of $\epsilon = \infty$ was also considered. The performance was evaluated on target domain testing dataset in-terms of classification accuracy.

Table 1.2 reports the results of 10 independent MNIST→USPS experiments. As observed in Table 1.2, the proposed method, despite being privacy-preserving and having not required an access to source data samples, performs comparable to the Deep Variational Transfer (a variational autoencoder that transfers knowledge across domains using a shared latent Gaussian mixture model) proposed in [10]. Further, the proposed method's consistent performance over a wide range of privacy-loss bound ϵ verifies the robustness of the target model towards the perturbations in source data caused by the privacy requirements demanded by source data owner.

Effect of labelled target sample size To study the effect of number of labelled target samples, USPS→MNIST problem is considered with number of labelled target samples varying from 100 to 500. Table 1.3 reports the classification accuracy on target testing dataset as the number of labelled target samples is varied. It is verified that the proposed approach to combine source and target domain classifiers, as in (1.87), leads to an increasing performance with increasing labelled target sample size while preserving the privacy of source domain data.

1.4.2 Comparisons Using Office and Caltech256 Datasets

"Office+Caltech256" dataset has 10 common categories of both Office and Caltech256 datasets. This dataset has been widely used [11–14] for evaluating multi-class accuracy performance in a standard domain adaptation setting with a small number of labelled target samples. The dataset has fours domains: *amazon*, *webcam*, *dslr*, and *caltech256*. We follow the experimental setup of [11–14]:

1. the number of training samples per class in the source domain is 20 for *amazon* and is 8 for other three domains;
2. the number of labelled samples per class in the target domain is 3 for all the four domains;
3. 20 random train/test splits are created and the performance on target domain test samples is averaged over 20 experiments.

Table 1.3: Effect of labelled target sample size on performance in USPS→MNIST problem.

method	**number of labelled target samples**	**classification accuracy on target testing data**
$(0.1, 1e{-}5)-$differentially private proposed	100	92.29%
$(0.1, 1e{-}5)-$differentially private proposed	200	95.86%
$(0.1, 1e{-}5)-$differentially private proposed	300	97.31%
$(0.1, 1e{-}5)-$differentially private proposed	400	97.63%
$(0.1, 1e{-}5)-$differentially private proposed	500	97.99%

Table 1.4: Accuracy (in %, averaged over 20 experiments) obtained in *amazon→caltech256* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1e{-}5)-$differentially private proposed	VGG-FC6	<u>80.6</u>
SVM-t (without knowledge transfer)	VGG-FC6	73.4
non-private ILS (1-NN)	VGG-FC6	**<u>83.3</u>**
non-private CDLS	VGG-FC6	78.1
non-private MMDT	VGG-FC6	78.7
non-private HFA	VGG-FC6	75.5
non-private OBTL	SURF	41.5
non-private ILS (1-NN)	SURF	43.6
non-private CDLS	SURF	35.3
non-private MMDT	SURF	36.4
non-private HFA	SURF	31.0

Table 1.5: Accuracy (in %, averaged over 20 experiments) obtained in *amazon→dslr* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1e{-}5)-$differentially private proposed	VGG-FC6	**<u>91.2</u>**
SVM-t (without knowledge transfer)	VGG-FC6	<u>90.0</u>
non-private ILS (1-NN)	VGG-FC6	87.7
non-private CDLS	VGG-FC6	86.9
non-private MMDT	VGG-FC6	77.1
non-private HFA	VGG-FC6	87.1
non-private OBTL	SURF	60.2
non-private ILS (1-NN)	SURF	49.8
non-private CDLS	SURF	60.4
non-private MMDT	SURF	56.7
non-private HFA	SURF	55.1

Following [12], the deep-net VGG-FC6 features are extracted from the images and the proposed method is compared with

1. SVM-t: A base-line is created using a linear SVM classifier trained using only the labelled target samples without transfer learning.
2. ILS (1-NN) [12]: This method learns an Invariant Latent Space (ILS) to reduce the discrepancy between domains and uses Riemannian optimization techniques to match statistical properties between samples projected into the latent space from different domains.
3. CDLS [15]: The Cross-Domain Landmark Selection (CDLS) method derives a domain-invariant feature subspace for heterogeneous domain adaptation.
4. MMDT [14]: The Maximum Margin Domain Transform (MMDT) method adapts max-margin classifiers in a multi-class manner by learning a shared component of the domain shift as captured by the feature transformation.
5. HFA [16]: The Heterogeneous Feature Augmentation (HFA) method learns common latent subspace and a classifier under max-margin framework.
6. OBTL [13]: The Optimal Bayesian Transfer Learning (OBTL) method employs Bayesian framework to transfer learning through modeling of a joint prior probability density function for feature-label distributions of the source and target domains.

The "Office+Caltech256" dataset has been previously studied in [11–14] using SURF features. Therefore, the state-of-art results on this dataset using SURF features are additionally considered for a comparison. There are in total 4 domains associated to "Office+Caltech256" dataset. Taking a domain as source and other domain as target, 12 different transfer learning experiments can be performed on these 4 domains. Table 1.4, Table 1.5, Table 1.6, Table 1.7, Table 1.8, Table 1.9, Table 1.10, Table 1.11, Table 1.12, Table 1.13, Table 1.14, and Table 1.15 report the results and the first two best performances have been marked.

Table 1.6: Accuracy (in %, averaged over 20 experiments) obtained in *amazon→webcam* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	89.5
SVM-t (without knowledge transfer)	VGG-FC6	86.9
non-private ILS (1-NN)	VGG-FC6	90.7
non-private CDLS	VGG-FC6	**91.2**
non-private MMDT	VGG-FC6	82.5
non-private HFA	VGG-FC6	87.9
non-private OBTL	SURF	72.4
non-private ILS (1-NN)	SURF	59.7
non-private CDLS	SURF	68.7
non-private MMDT	SURF	64.6
non-private HFA	SURF	57.4

Table 1.7: Accuracy (in %, averaged over 20 experiments) obtained in *caltech256→amazon* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**91.5**
SVM-t (without knowledge transfer)	VGG-FC6	84.4
non-private ILS (1-NN)	VGG-FC6	89.7
non-private CDLS	VGG-FC6	88.0
non-private MMDT	VGG-FC6	85.9
non-private HFA	VGG-FC6	86.2
non-private OBTL	SURF	54.8
non-private ILS (1-NN)	SURF	55.1
non-private CDLS	SURF	50.9
non-private MMDT	SURF	49.4
non-private HFA	SURF	43.8

Table 1.8: Accuracy (in %, averaged over 20 experiments) obtained in *caltech256→dslr* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**91.6**
SVM-t (without knowledge transfer)	VGG-FC6	90.2
non-private ILS (1-NN)	VGG-FC6	86.9
non-private CDLS	VGG-FC6	86.3
non-private MMDT	VGG-FC6	77.9
non-private HFA	VGG-FC6	87.0
non-private OBTL	SURF	61.5
non-private ILS (1-NN)	SURF	56.2
non-private CDLS	SURF	59.8
non-private MMDT	SURF	56.5
non-private HFA	SURF	55.6

Table 1.9: Accuracy (in %, averaged over 20 experiments) obtained in *caltech256→webcam* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**91.6**
SVM-t (without knowledge transfer)	VGG-FC6	88.8
non-private ILS (1-NN)	VGG-FC6	91.4
non-private CDLS	VGG-FC6	89.7
non-private MMDT	VGG-FC6	82.8
non-private HFA	VGG-FC6	86.0
non-private OBTL	SURF	71.1
non-private ILS (1-NN)	SURF	62.9
non-private CDLS	SURF	66.3
non-private MMDT	SURF	63.8
non-private HFA	SURF	58.1

Table 1.10: Accuracy (in %, averaged over 20 experiments) obtained in *dslr→amazon* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**90.7**
SVM-t (without knowledge transfer)	VGG-FC6	84.4
non-private ILS (1-NN)	VGG-FC6	88.7
non-private CDLS	VGG-FC6	88.1
non-private MMDT	VGG-FC6	83.6
non-private HFA	VGG-FC6	85.9
non-private OBTL	SURF	54.4
non-private ILS (1-NN)	SURF	55.0
non-private CDLS	SURF	50.7
non-private MMDT	SURF	46.9
non-private HFA	SURF	42.9

Table 1.11: Accuracy (in %, averaged over 20 experiments) obtained in *dslr→caltech256* semi-supervised transfer learning experiments.

method	feature type	accuracy (%)
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**81.4**
SVM-t (without knowledge transfer)	VGG-FC6	73.6
non-private ILS (1-NN)	VGG-FC6	**81.4**
non-private CDLS	VGG-FC6	77.9
non-private MMDT	VGG-FC6	71.8
non-private HFA	VGG-FC6	74.8
non-private OBTL	SURF	40.3
non-private ILS (1-NN)	SURF	41.0
non-private CDLS	SURF	34.9
non-private MMDT	SURF	34.1
non-private HFA	SURF	30.9

Table 1.12: Accuracy (in %, averaged over 20 experiments) obtained in *dslr→webcam* semi-supervised transfer learning experiments.

method	**feature type**	**accuracy (%)**
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	88.7
SVM-t (without knowledge transfer)	VGG-FC6	86.9
non-private ILS (1-NN)	VGG-FC6	**95.5**
non-private CDLS	VGG-FC6	90.7
non-private MMDT	VGG-FC6	86.1
non-private HFA	VGG-FC6	86.9
non-private OBTL	SURF	83.2
non-private ILS (1-NN)	SURF	80.1
non-private CDLS	SURF	68.5
non-private MMDT	SURF	74.1
non-private HFA	SURF	60.5

Table 1.13: Accuracy (in %, averaged over 20 experiments) obtained in *webcam→amazon* semi-supervised transfer learning experiments.

method	**feature type**	**accuracy (%)**
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	**92.0**
SVM-t (without knowledge transfer)	VGG-FC6	84.8
non-private ILS (1-NN)	VGG-FC6	88.8
non-private CDLS	VGG-FC6	87.4
non-private MMDT	VGG-FC6	84.7
non-private HFA	VGG-FC6	85.1
non-private OBTL	SURF	55.0
non-private ILS (1-NN)	SURF	54.3
non-private CDLS	SURF	51.8
non-private MMDT	SURF	47.7
non-private HFA	SURF	56.5

Table 1.14: Accuracy (in %, averaged over 20 experiments) obtained in *webcam→caltech256* semi-supervised transfer learning experiments.

method	**feature type**	**accuracy (%)**
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	82.3
SVM-t (without knowledge transfer)	VGG-FC6	75.0
non-private ILS (1-NN)	VGG-FC6	**82.8**
non-private CDLS	VGG-FC6	78.2
non-private MMDT	VGG-FC6	73.6
non-private HFA	VGG-FC6	74.4
non-private OBTL	SURF	37.4
non-private ILS (1-NN)	SURF	38.6
non-private CDLS	SURF	33.5
non-private MMDT	SURF	32.2
non-private HFA	SURF	29.0

Table 1.15: Accuracy (in %, averaged over 20 experiments) obtained in *webcam→dslr* semi-supervised transfer learning experiments.

method	**feature type**	**accuracy (%)**
$(0.1, 1\text{e}{-5})-$differentially private proposed	VGG-FC6	89.6
SVM-t (without knowledge transfer)	VGG-FC6	88.9
non-private ILS (1-NN)	VGG-FC6	**94.5**
non-private CDLS	VGG-FC6	88.5
non-private MMDT	VGG-FC6	85.1
non-private HFA	VGG-FC6	87.3
non-private OBTL	SURF	75.0
non-private ILS (1-NN)	SURF	70.8
non-private CDLS	SURF	60.7
non-private MMDT	SURF	67.0
non-private HFA	SURF	56.5

Table 1.16: Comparison of the methods on "Office+Caltech256" dataset.

method	**number of experiments in which method performed best**	**number of experiments in which method performed 2nd best**
$(0.1, 1\text{e}{-5})-$differentially private proposed	7	3
non-private ILS (1-NN)	5	5
non-private CDLS	1	2

Finally, Table 1.16 summarizes the overall performance of top three methods. As observed in Table 1.16, the proposed method remains as best performing in maximum number of experiments. The most remarkable result observed is that the proposed transfer and multi-task learning method, despite ensuring privacy-loss bound to be as low as 0.1 and not requiring an access to source data samples, performs better than even the non-private methods.

1.5 Concluding Remarks

We presented a novel approach to differentially private semi-supervised transfer and multi-task learning that exploits the variational deep mappings and an optimal noise adding mechanism for achieving a robustness of target model towards the perturbations in source data caused by the privacy requirements demanded by source data owner. A variational membership-mapping based approach was introduced that sufficiently addresses all of the requirements identified regarding the privacy-preserving transferrable deep learning problem. Numerous experiments were carried out using MNIST, USPS, Office, and Caltech256 datasets to verify the competitive performance of the proposed method. The experimental studies further verify that our approach is capable of achieving a low privacy-loss bound without letting the accuracy be much degraded.

2. Secure Multiparty Computation and Homomorphic Encryption (T3.3)

A methodology for practical secure privacy-preserving distributed machine (deep) learning is proposed via addressing the core issues of fully homomorphic encryption. Considering that private data is distributed and the training data may contain directly or indirectly an information about private data, an architecture and a methodology are suggested for mitigating the impracticality issue of fully homomorphic encryption (arising from large computational overhead) via very fast gate-by-gate bootstrapping and introducing a learning scheme that requires homomorphic computation of only efficient-to-evaluate functions.

The emergence of cloud infrastructure not only raises the concern of protecting data in storage, but also requires an ability of performing computations on data while preserving the data privacy. Fully homomorphic encryption (FHE) being capable of directly performing an unbounded number of operations on encrypted data forms a solution to the privacy concerns in the cloud computing scenario. The first FHE scheme [17] is based on ideal lattices and the bootstrapping procedure is introduced to reduce the noise contained in a ciphertext for allowing arbitrary computations. The bootstrapping operation is performed on a ciphertext via evaluating the decryption function homomorphically using the bootstrapping key (which is the encryption of the private decryption key under the public encryption key). Bootstrapping is the computationally most expensive part of a homomorphic encryption scheme. The theoretical breakthrough of [17] was followed by several attempts to develop more practical FHE schemes. The scheme introduced in [18] uses only elementary modulo arithmetic and is homomorphic with regard to both addition and multiplication. This scheme was improved in [19] with reduced public key size, extended in [20] to support encrypting and homomorphically processing a vector of plaintexts as a single ciphertext, and generalized to non-binary messages in [21]. Schemes based on a different hard problem, referred to as Learning With Errors (LWE) problem [22], were constructed and many current schemes still rely on LWE or its variants. A FHE scheme constructed in [23] is based solely on the standard LWE assumption that is known to be at least as hard as solving hard problems in general lattices. In a variant of the LWE problem, called ring learning with errors problem (RLWE) problem, the algebraic structure of the underlying hard problem reduces the key sizes and speeds up the homomorphic operations. A leveled fully homomorphic encryption scheme based on LWE or RLWE, without bootstrapping procedure, was proposed in [24]. The ciphertexts contain a certain amount of noise for security purposes that grows with homomorphic operations. For a better management of the noise growth, [24] introduced a modulus switching technique where a complete ladder of moduli is used for scaling down the ciphertext to the next modulus after each multiplication. A tensoring technique for LWE-based FHE that reduced ciphertext noise growth after multiplication from quadratic to linear was introduced in [25]. As the scheme of [25] does no longer require the rescaling of the ciphertext, this scheme was called a scale-invariant fully homomorphic encryption scheme. An RLWE version of the scale-invariant scheme of [25] was created in [26]. A technique for building LWE based FHE scheme called as approximate eigenvector method in which homomorphic addition and multiplication are just matrix addition and multiplication was proposed in [27]. The essence of this scheme is that the secret key is an approximate eigenvector of the ciphertext matrix and the message is the corresponding eigenvalue. Several works that followed the theoretical breakthrough of [17] were aimed at improving the bootstrapping as the bootstrapping remained the bottleneck for an efficient FHE in practice. A much faster bootstrapping, based on a scheme similar to the type of [27] that allows to homomorphically compute simple bit operations and refresh (bootstrap) the resulting output in less than a second, was devised in [28]. Finally, the TFHE scheme was proposed in [29, 30] that features an improved bootstrapping procedure that is considerably more efficient than the previous state of the art. The TFHE scheme generalizes previous structures and schemes over the torus (i.e., the reals modulo 1) and improves the bootstrapping dramatically. For practical applications, TFHE is an open-source C/C++ library [31] implementing the ring-variant of [27] together with the optimizations of [28–30]. TFHE library implements a very fast gate-by-gate bootstrapping and supports the homomorphic evaluation of the binary gates. The library allows to evaluate homomorphically an arbitrary boolean circuit composed of binary gates without restriction on the number of gates or on their composition, over encrypted data, without decrypting. However, the bootstrapped bit operations are still several times slower than their plaintext equivalents. Thus for an efficient secure machine learning scenario in practice, homomorphic evaluation of the function with smallest possible number of gates is one of the optimality criteria for designing learning algorithms with distributed data.

Requirement: An efficient secure machine (deep) learning with fully homomorphic encryption in practice demands an approach ensuring that the homomorphic evaluation of functions would require evaluating the circuits with number of gates as small as possible.

The requirement, identified for a practical secure privacy-preserving learning, is fulfilled with an architecture assuming that the complete data set on which we apply machine-learning based big data analytics, is distributed across different organizational units and that learning models are constructed and applied within each unit independently, after which these local models are combined into a global model. To combine local output data from different organizational units into a global output, the data needs to be transmitted between organizations and, therefore, encryption is needed. Each local model output is homomorphically encrypted and shared in the cloud where a global model (that combines the distributed local models) is homomorphically evaluated in an efficient manner to predict the output. The issue regarding the impracticality of homomorphic computation of global model due to large computational overhead is mitigated via two ways:

1. Very fast gate-by-gate bootstrapping is implemented [31].
2. Combining local entities' models requires homomorphic computation of only simpler functions (such as computing minimum or maximum amongst scalars) that can be homomorphically evaluated in an efficient manner. For this, a rule-based fuzzy model can be used for combining the local models [32, 33].

3. Verification of Differential Privacy Deep Learning Models (T3.4)

3.1 Membership Inference Attack on SERUMS Model

One of the most known attacks on machine learning models is membership inference. Given a model and an input record, membership inference attempts to determine whether the record was used during the model training. The attack is developed from the assumption that models in general perform better on the entries from the training dataset, having more confidence in the output. A successful attack can raise a significant privacy risk, as the training datasets can contain private patients data.

The membership inference attack has been originally proposed by [34], are later received multiple modifications. [35] provides a classification and a survey on different types of membership inference attacks. Several works, for example [36, 37], show that differential privacy can provide protection against the membership inference attacks, though [38] argues that the accuracy-privacy trade-off is hard to achieve.

We investigate the effect of the $(\epsilon - \delta)$ differential privacy on the capabilities of membership inference attacks. We used models trained from a fabricated dataset provided by USTAN and attempted to perform several versions of the membership inference attack.

In particular, we consider the following capabilities of an attacker:

- An attacker can obtain a dataset similar to the training one: it has similar data distribution as a training dataset, but there are no common records in the two datasets.
- An attacker can call the trained model and can receive an output prediction vector. Note, that this assumption will not be present in the SERUMS software: SERUMS models output a predicted class only.
- An attacker knows a structure of the trained model, but not the trained coefficients.

We consider the following types of membership inference attacks:

- **Metric based attacks**. These attacks compute metrics of prediction vectors and decide the membership by comparing the metrics with a threshold. The following metrics have been used:
  - Prediction correctness: an input is a member of the training dataset if the model makes a correct prediction. This simple version of attack is often considered as a baseline.
  - Prediction loss: compares prediction loss with a threshold.
  - Prediction confidence: compares prediction confidence with a threshold.
  - Prediction entropy: compares prediction entropy with a threshold.
- **Neural Network based attack**. This attack attempts to build a binary classifier for membership prediction. In order to train the classifier, a technique called *shadow training* is used. The idea is to build a set of shadow models that mimic the behaviour of the target model. Since the attacker knows the training datasets of the shadow models, their output is used as a training dataset for the binary classifier. We tried several different classifiers: staring from support-vector classifiers, continuing with decision tree based classifiers up to a neural network. In one of the experiments, we tried a different neural network for shadow models: the model has been taken from [38]. For the shadow dataset we split the original USTAN dataset into two disjoint parts. Being the fabricated dataset, the two parts are expected to have similar distributions. An extension of the attack proposes to build a binary classifier per output class of the target model. We use the extension in our evaluation and report the average result of all classes.

We compare the results of attacks on several models. As a baseline, we take a model where without additional privacy - no noise has been added to the training dataset. For the other models we added noise with different ϵ and δ. ϵ has values in $\{0.1, 1, 5, 10\}$; δ has values in $\{0.01, 0.001, 0.0001\}$. The number of shadow models for neural network based attack has been set to 20.

Epsilon	Correctness	Loss	Confidence	Entropy
non-private	0.4906	0.4896	0.4905	0.5114
0.1	0.4905	0.4907	0.5057	0.5107
1	0.4896	0.4908	0.5057	0.5112
5	0.4897	0.4904	0.5059	0.5105
10	0.4906	0.4909	0.5067	0.5102

Figure 3.1: Metric based attack accuracy

Epsilon	Accuracy
non-private	0.4815
0.1	0.4818
1	0.4815
5	0.4823
10	0.4818

Figure 3.2: Neural network based attack accuracy

The results for the metric based attacks are shown in Figure 3.1. All metrics on all models have the results close to 0.5 showing that none of the metrics can distinguish whether entries came from the training dataset.

The results for the neural network based attack are shown in Figure 3.2. Results with other classifiers have similar values. For this attack we can see that the trained classifier cannot infer the membership on any model.

Thus we can conclude that none of the membership inference attacks succeeded. A potential reason to the low performance of the attack is low accuracy of the target models: all models (including shadow models) have accuracy around 0.62.

4. Appendices

4.1 Evaluation of Membership Function

Using (1.34), we have

$$\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{f}_j}(\tilde{\mathrm{y}}_j)) \right\rangle_{\mu_{\mathrm{f}_j;\mathrm{u}_j}} = -0.5\beta\|\tilde{\mathrm{y}}_j - \bar{m}_{\mathrm{f}_j}\|^2 - 0.5\beta\frac{\nu + (\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j - 2}{\nu + M - 2}Tr(\bar{K}_{\mathrm{xx}})$$

where $Tr(\cdot)$ denotes the trace operator. Using (1.35) and (1.36),

$$\begin{aligned}\left\langle \log(\mu_{\mathrm{y}_j;\mathrm{f}_j}(\tilde{\mathrm{y}}_j)) \right\rangle_{\mu_{\mathrm{f}_j;\mathrm{u}_j}} &= -0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + \beta(\tilde{\mathrm{y}}_j)^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j - 0.5\beta(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j \\ &\quad -0.5\beta\frac{\nu + (\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j - 2}{\nu + M - 2}\left(Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})\right). \end{aligned} \tag{4.88}$$

Using (1.37),

$$\begin{aligned}\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j) \propto \exp\Big(&-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + \beta(\tilde{\mathrm{y}}_j)^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j - 0.5\beta(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j \\ &-0.5\beta\frac{(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j}{\nu + M - 2}\left(Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})\right) + \{/(\tilde{\mathrm{y}}_j, \mathrm{u}_j)\}\Big)\end{aligned}$$

where $\{/(\tilde{\mathrm{y}}_j, \mathrm{u}_j)\}$ represents all those terms which are independent of both $\tilde{\mathrm{y}}_j$ and u_j. Define

$$\hat{K}_{\mathrm{u}_j} = \left((K_{\mathrm{aa}})^{-1} + \beta(K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1} + \beta\frac{Tr(K_{\mathrm{xx}}) - Tr((K_{\mathrm{aa}})^{-1}K_{\mathrm{xa}}^T K_{\mathrm{xa}})}{\nu + M - 2}(K_{\mathrm{aa}})^{-1}\right)^{-1} \tag{4.89}$$

$$\hat{m}_{\mathrm{u}_j}(\tilde{\mathrm{y}}_j) = \beta\hat{K}_{\mathrm{u}_j}(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j \tag{4.90}$$

to express $\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j)$ as (1.38).

4.2 Solution of Optimization Problem

A new objective functional is defined after excluding u_j-independent terms and taking into account the integral constraint through a Lagrange multiplier γ:

$$\begin{aligned}\mathcal{J} = &\left\langle (\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j) - 0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j + 0.5(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j - \log(\mu_{\mathrm{u}_j}(\mathrm{u}_j)) - 0.5(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}\mathrm{u}_j \right\rangle_{\mu_{\mathrm{u}_j}} \\ &+\gamma\left\{\int_{\mathbb{R}^M}\mu_{\mathrm{u}_j}(\mathrm{u}_j)\,\mathrm{d}\lambda^M(\mathrm{u}_j) - C_{\mathrm{u}_j}\right\} \end{aligned} \tag{4.91}$$

$$\begin{aligned}= &\frac{1}{C_{\mathrm{u}_j}}\int_{\mathbb{R}^M}\mathrm{d}\lambda^M(\mathrm{u}_j)\,\mu_{\mathrm{u}_j}(\mathrm{u}_j)\left\{(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j) - 0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j - \log(\mu_{\mathrm{u}_j}(\mathrm{u}_j))\right\} \\ &+\gamma\left\{\int_{\mathbb{R}^M}\mu_{\mathrm{u}_j}(\mathrm{u}_j)\,\mathrm{d}\lambda^M(\mathrm{u}_j) - C_{\mathrm{u}_j}\right\} \end{aligned} \tag{4.92}$$

Setting the functional derivative of $\mathcal{J}$ w.r.t. μ_{u_j} equal to zero,

$$0 = \gamma + (1/C_{\mathrm{u}_j})\left\{-1 - 0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j + (\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j) - \log(\mu_{\mathrm{u}_j}(\mathrm{u}_j))\right\}. \tag{4.93}$$

That is,

$$\mu_{\mathrm{u}_j}(\mathrm{u}_j) = \exp(\gamma C_{\mathrm{u}_j} - 1)\exp\left(-0.5(\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\mathrm{u}_j + (\mathrm{u}_j)^T\hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right). \tag{4.94}$$

The optimal value of γ is obtained by solving $\int_{\mathbb{R}^M}\mu_{\mathrm{u}_j}\,\mathrm{d}\lambda^M = C_{\mathrm{u}_j}$. This leads to

$$\exp(\gamma C_{\mathrm{u}_j} - 1)\sqrt{(2\pi)^M/|\hat{K}_{\mathrm{u}_j}^{-1}|}\exp\left(0.5\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T\hat{K}_{\mathrm{u}_j}^{-1}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right) = C_{\mathrm{u}_j}. \tag{4.95}$$

Thus, the optimal expression for μ_{u_j} is given as

$$\mu^*_{\mathrm{u}_j}(\mathrm{u}_j) = C_{\mathrm{u}_j}\sqrt{|\hat{K}^{-1}_{\mathrm{u}_j}|/(2\pi)^M}\exp\left(-0.5(\mathrm{u}_j-\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j))^T\hat{K}^{-1}_{\mathrm{u}_j}(\mathrm{u}_j-\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j))\right). \tag{4.96}$$

Finally, C_{u_j} is chosen such that $\max_{\mathrm{u}_j}\mu^*_{\mathrm{u}_j}(\mathrm{u}_j)=1$. This results in

$$\mu^*_{\mathrm{u}_j}(\mathrm{u}_j) = \exp\left(-0.5(\mathrm{u}_j-\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j))^T\hat{K}^{-1}_{\mathrm{u}_j}(\mathrm{u}_j-\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j))\right). \tag{4.97}$$

Thus, $\langle \mathrm{u}_j\rangle_{\mu^*_{\mathrm{u}_j}} = \hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)$, and using (4.90), we get

$$\langle \mathrm{u}_j\rangle_{\mu^*_{\mathrm{u}_j}} = \beta\hat{K}_{\mathrm{u}_j}(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\mathrm{y}_j. \tag{4.98}$$

It follows from (4.89) and (4.90) that

$$\hat{K}^{-1}_{\mathrm{u}_j} - (K_{\mathrm{aa}})^{-1} = \beta(K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1} + \beta\frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1} \tag{4.99}$$

$$\hat{K}^{-1}_{\mathrm{u}_j}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j) = \beta(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\mathrm{y}_j. \tag{4.100}$$

Using (4.99) and (4.100) in (1.39), we have

$$\begin{aligned}\log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j)) = &-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + \beta(\mathrm{u}_j)^T(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j\\ &-0.5\beta(\mathrm{u}_j)^T\left\{(K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1} + \frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1}\right\}\mathrm{u}_j.\end{aligned}$$

Thus, $\left\langle\log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j))\right\rangle_{\mu^*_{\mathrm{u}_j}}$ is given as

$$\begin{aligned}\left\langle\log(\mu_{\mathrm{y}_j;\mathrm{u}_j}(\tilde{\mathrm{y}}_j))\right\rangle_{\mu^*_{\mathrm{u}_j}} = &-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + \beta\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j\\ &-0.5\beta\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T\left\{(K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1} + \frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1}\right\}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\\ &-0.5\beta Tr\left((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\hat{K}_{\mathrm{u}_j} + \frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1}\hat{K}_{\mathrm{u}_j}\right).\end{aligned} \tag{4.101}$$

The data-model (1.40) using (4.101) becomes as

$$\begin{aligned}\mu_{\mathrm{y}_j}(\tilde{\mathrm{y}}_j) \propto \exp\Big(&-0.5\beta\|\tilde{\mathrm{y}}_j\|^2 + \beta\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\tilde{\mathrm{y}}_j\\ &-0.5\beta\left(\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\right)^T\left\{(K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1} + \frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1}\right\}\hat{m}_{\mathrm{u}_j}(\mathrm{y}_j)\\ &-0.5\beta Tr\left((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\hat{K}_{\mathrm{u}_j} + \frac{Tr(K_{\mathrm{xx}})-Tr((K_{\mathrm{aa}})^{-1}K^T_{\mathrm{xa}}K_{\mathrm{xa}})}{\nu+M-2}(K_{\mathrm{aa}})^{-1}\hat{K}_{\mathrm{u}_j}\right)\Big).\end{aligned} \tag{4.102}$$

Thus, (1.44) follows.

4.3 Membership-Mapping Output Estimation

Using (1.34) and (1.35), we have

$$\langle(\mathrm{f}_j)_i\rangle_{\mu_{\mathrm{f}_j;\mathrm{u}_j}} = (K_{\mathrm{xa}}(K_{\mathrm{aa}})^{-1}\mathrm{u}_j)_i \tag{4.103}$$

$$= G(x^i)(K_{\mathrm{aa}})^{-1}\mathrm{u}_j. \tag{4.104}$$

Thus,

$$\widehat{\mathcal{F}_j(x^i)} = G(x^i)(K_{\mathrm{aa}})^{-1}\langle\mathrm{u}_j\rangle_{\mu^*_{\mathrm{u}_j}}. \tag{4.105}$$

Using (4.98) in (4.105), we have

$$\widehat{\mathcal{F}_j(x^i)} = \beta\left(G(x^i)\right)(K_{\mathrm{aa}})^{-1}\hat{K}_{\mathrm{u}_j}(K_{\mathrm{aa}})^{-1}(K_{\mathrm{xa}})^T\mathrm{y}_j. \tag{4.106}$$

Substituting $\hat{K}_{\mathrm{u}_j}$ from (4.89) in (4.106), we get (1.48).

Bibliography

[1] M. Kumar and B. Freudenthaler, "Fuzzy membership functional analysis for nonparametric deep models of image features," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 12, pp. 3345–3359, 2020.

[2] M. Kumar, W. Zhang, M. Weippert, and B. Freudenthaler, "An explainable fuzzy theoretic nonparametric deep model for stress assessment using heartbeat intervals analysis," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020.

[3] M. Kumar, S. Singh, and B. Freudenthaler, "Gaussian fuzzy theoretic analysis for variational learning of nested compositions," *International Journal of Approximate Reasoning*, vol. 131, pp. 1–29, 2021.

[4] M. Kumar, B. Moser, L. Fischer, and B. Freudenthaler, "Membership-mappings for data representation learning: Measure theoretic conceptualization," in *Database and Expert Systems Applications - DEXA 2021 Workshops*, G. Kotsis, A. M. Tjoa, I. Khalil, B. Moser, A. Mashkoor, J. Sametinger, A. Fensel, J. Martinez-Gil, L. Fischer, G. Czech, F. Sobieczky, and S. Khan, Eds. Cham: Springer International Publishing, 2021, pp. 127–137.

[5] ——, "Membership-mappings for data representation learning: A bregman divergence based conditionally deep autoencoder," in *Database and Expert Systems Applications - DEXA 2021 Workshops*, G. Kotsis, A. M. Tjoa, I. Khalil, B. Moser, A. Mashkoor, J. Sametinger, A. Fensel, J. Martinez-Gil, L. Fischer, G. Czech, F. Sobieczky, and S. Khan, Eds. Cham: Springer International Publishing, 2021, pp. 138–147.

[6] M. Kumar, M. Rossbory, B. A. Moser, and B. Freudenthaler, "Deriving an optimal noise adding mechanism for privacy-preserving machine learning," in *Proceedings of the 3rd International Workshop on Cyber-Security and Functional Safety in Cyber-Physical (IWCFS 2019), August 26-29, 2019, Linz, Austria*, G. Anderst-Kotsis, A. M. Tjoa, I. Khalil, M. Elloumi, A. Mashkoor, J. Sametinger, X. Larrucea, A. Fensel, J. Martinez-Gil, B. Moser, C. Seifert, B. Stein, and M. Granitzer, Eds. Cham: Springer International Publishing, 2019, pp. 108–118.

[7] J. He, L. Cai, and X. Guan, "Differential private noise adding mechanism and its application on consensus algorithm," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4069–4082, 2020.

[8] N. Papernot, M. Abadi, Ã. Erlingsson, I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data." in *ICLR*. OpenReview.net, 2017. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2017.html#PapernotAEGT17

[9] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2016, p. 308âĂŞ318.

[10] M. Belhaj, P. Protopapas, and W. Pan, "Deep variational transfer: Transfer learning through semi-supervised deep generative models," *ArXiv*, vol. abs/1812.03123, 2018.

[11] J. Hoffman, E. Rodner, J. Donahue, K. Saenko, and T. Darrell, "Efficient learning of domain-invariant image representations," *CoRR*, vol. abs/1301.3224, 2013.

[12] S. Herath, M. Harandi, and F. Porikli, "Learning an invariant hilbert space for domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] A. Karbalayghareh, X. Qian, and E. R. Dougherty, "Optimal bayesian transfer learning," *IEEE Transactions on Signal Processing*, vol. 66, no. 14, pp. 3724–3739, 2018.

[14] J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko, "Asymmetric and category invariant feature transformations for domain adaptation," *International Journal of Computer Vision*, vol. 109, no. 1, pp. 28–41, 2014.

[15] Y. H. Tsai, Y. Yeh, and Y. F. Wang, "Learning cross-domain landmarks for heterogeneous domain adaptation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5081–5090.

[16] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1134–1148, 2014.

[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 169–178.

[18] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology – EUROCRYPT 2010*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–43.

[19] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology – CRYPTO 2011*, P. Rogaway, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 487–504.

[20] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully homomorphic encryption over the integers," in *Advances in Cryptology – EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 315–335.

[21] K. Nuida and K. Kurosawa, "(batch) fully homomorphic encryption over integers for non-binary message spaces," in *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 537–555.

[22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 84–93.

[23] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011, pp. 97–106.

[24] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 309–325.

[25] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886.

[26] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012. [Online]. Available: http://eprint.iacr.org/2012/144

[27] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 75–92.

[28] L. Ducas and D. Micciancio, "Fhew: Bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 617–640.

[29] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Advances in Cryptology – ASIACRYPT 2016*, J. H. Cheon and T. Takagi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3–33.

[30] ——, "Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 377–408.

[31] ——, "TFHE: Fast fully homomorphic encryption library," August 2016, https://tfhe.github.io/tfhe/.

[32] M. Kumar, M. Rossbory, B. A. Moser, and B. Freudenthaler, "An optimal $(\epsilon, \delta)-$differentially private learning of distributed deep fuzzy models," *Information Sciences*, vol. 546, pp. 87–120, 2021.

[33] ——, "Differentially private learning of distributed deep models," in *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '20 Adjunct. New York, NY, USA: Association for Computing Machinery, 2020, pp. 193–200.

[34] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[35] H. Hu, Z. Salcic, G. Dobbie, and X. Zhang, "Membership inference attacks on machine learning: A survey," *arXiv preprint arXiv:2103.07853*, 2021.

[36] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model." *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.

[37] Z. Ying, Y. Zhang, and X. Liu, "Privacy-preserving in defending against membership inference attacks," in *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, 2020, pp. 61–63.

[38] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1895–1912.