Differentially private transferrable deep learning with membership-mappings

Mohit Kumar

Received: 18 June 2022 / Revised: 15 November 2022 / Accepted: 27 November 2022 / Published online: 15 December 2022
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2022

Abstract
Despite a recent surge of research interest in privacy and transferrable deep learning, optimizing the tradeoff between privacy requirements and performance of machine learning models remains a challenge. This motivates the development of an approach that optimizes both privacy-preservation mechanism and learning of the deep models for achieving a robust performance. This paper considers the problem of semi-supervised transfer and multi-task learning under differential privacy framework. An alternative conception of deep autoencoder, referred to as Conditionally Deep Membership-Mapping Autoencoder (CDMMA), is considered for transferrable deep learning. Under practice-oriented settings, an analytical solution for the learning of CDMMA can be derived by means of variational optimization. The paper proposes a transfer and multi-task learning approach that combines CDMMA with a tailored noise adding mechanism to transfer knowledge from source to target domain in a privacy-preserving manner.

Keywords Privacy · Deep learning · Fuzzy · Transfer learning

1 Introduction

The availability of high-quality labelled data is crucial for the success of machine learning methods. While a single entity may not own massive amount of data, a collaboration among data-owners regarding sharing of knowledge extracted locally from their private data can be beneficial. The data privacy concerns and the legal requirements may not allow a centralization of the data from multiple sources. Thus, an interest in privacy-preserving machine learning with distributed training datasets arises. We consider the privacy-preserving distributed machine learning problem under a scenario that the knowledge extracted from a labelled training dataset (referred to as source domain) is intended to improve the learning of a classifier trained using a dataset with both unlabelled and few labelled samples (referred to as target domain) such that source and target domains are allowed to be heterogeneous. That is, source and target data samples are allowed to differ in their dimensions and no assumptions are made regarding statistical distributions of source and target data. The problem of privacy-preserving semi-supervised transfer learning has previously been addressed in the literature from different prospects. We focus on the development of a method able to simultaneously deal with high-dimensional data and heterogeneous domains. Most important, the issue of loss of performance as a result of a differential privacy mechanism needs to be addressed.

The motivation of this study is to develop a differentially private semi-supervised transfer and multi-task learning framework that

R1: is capable of handling high-dimensional data and heterogeneity of domains;
R2: 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;
R3: allows learning of the target domain model without requiring an access to source domain private training data;
R4: ensures that a high level of privacy (i.e. sufficiently low value of privacy-loss bound) would not degrade the learning performance;
R5: 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;
R6: allows an efficient learning of deep models while addressing the issues such as determining the optimal model structure, smaller training dataset, and iterative time-consuming nature of numerical learning algorithms.

A lot of research has been carried out in the area of transfer learning. The heterogeneous data from source and target domain (i.e. source and target domains have different feature space and dimensions) can be transformed to a common subspace using two different projection matrices. Existing supervised learning methods (e.g., SVM) can be then employed to learn the projection matrices and the target domain classifier (Li et al. 2014). It is possible to learn a transformation that maps feature points from one domain to another using cross-domain constraints formed by requiring that the transformation maps points from the same category (but different domain) near each other (Hoffman et al. 2014). A study (Herath et al. 2017) learns projections from each domain to a latent space via simultaneously minimizing a notion of domain variance while maximizing a measure of discriminatory power where Riemannian optimization techniques are used to match statistical properties between samples projected into the latent space from different domains. Another study (Courty et al. 2017) proposes a regularized unsupervised optimal transportation model to perform the alignment of the representations in the source and target domains. The method in Gong et al. (2012) uses geodesic flow to construct an infinite-dimensional feature space that assembles information on the source domain, on the target domain, and on phantom domains interpolating between source and target domains. Inner products in infinite-dimensional feature space give rise to a kernel function facilitating the construction of any kernelized classifiers. Another approach is of an adaptation of source model to the target domain via iteratively deleting source-domain samples and adapting the model gradually to the target-domain instances (Bruzzone and Marconcini 2010). Boosting-based learning algorithms can be also used to adaptively assign the training weights to source and target samples based on their relevance in the training of the classifier (Dai et al. 2007). Bayesian learning can be a framework to study transfer learning through modeling of a joint prior probability density function for feature-label distributions of the source and target domains (Karbalayghareh et al. 2018). Deep learning framework is another promising research direction explored for transfer learning (Long et al. 2015, 2016; Ganin et al. 2016).

The datasets may contain sensitive information that need to be protected from model inversion attack (Fredriksson et al. 2015) and from adversaries with an access to model parameters and knowledge of the training procedure. This goal has been addressed within the framework of differential privacy (Abadi et al. 2016; Phan et al. 2016). Differential Privacy (Dwork et al. 2006; Dwork and Roth 2014) is a formalism to quantify the degree to which the privacy for each individual in the dataset is preserved while releasing the output of a data analysis algorithm. Differential privacy provides a guarantee that an adversary, by virtue of presence or absence of an individual’s data in the dataset, would not be able to draw any conclusions about an individual from the released output of the analysis algorithm. This guarantee is achieved by means of a randomization of the data analysis process. In the context of machine learning, randomization is carried out via either adding random noise to the input or output of the machine learning algorithm or modifying the machine learning algorithm itself. A limited number of studies exist on differentially private semi-supervised transfer learning. The authors in Ji and Elkan (2013) suggest an importance weighting mechanism to preserve the differential privacy of a private dataset via computing and releasing a weight for each record in an existing public dataset such that computations on public dataset with weights is approximately equivalent to computations on private dataset. The importance weighting mechanism is adapted in Wang et al. (2018) to determine the weight of a source hypothesis in the process of constructing informative Bayesian prior for logistic regression-based target model. Papernot et al. (2017) introduces private aggregation of teacher ensembles approach where an ensemble of teacher models is trained on disjoint subsets of the sensitive data and a student model learns to predict an output chosen by noisy voting among all of the teachers. Another approach (Acs et al. 2017; Xie et al. 2018; Zhang et al. 2017) is to construct a differentially private unsupervised generative model for generating a synthetic version of the private data, and then releases the synthetic data for a non-private learning. This technique is capable of effectively handling high-dimensional data in differential privacy setting. The study in Niinimäki et al. (2019) uses a large public dataset to learn a dimension-reducing representation mapping which is then applied on private data to obtain a low-dimensional representation of the private data followed by the learning of a differentially private predictor. However, these methods do not consider the heterogeneous domains. Differential privacy preserves the privacy of the training dataset via adding random noise to ensure that an adversary can not infer any single data instance by observing model parameters or model outputs. We follow the input perturbation method where noise is added to original data to achieve $(\epsilon, \delta)$—differential privacy of any subsequent computational algorithm processing the perturbed data. However,
the injection of noise into data would in general result in a loss of algorithm’s accuracy. Therefore, design of a noise injection mechanism achieving a good trade-off between privacy and accuracy is a topic of interest (Geng et al. 2018; Balle and Wang 2018; Ghosh et al. 2012; Gupte and Sundararajan 2010; Geng and Viswanath 2016a; Geng et al. 2015; Geng and Viswanath 2016b). The authors in Kumar et al. (2019) derive the probability density function of noise that minimizes the expected noise magnitude together with satisfying the sufficient conditions for \((\epsilon, \delta)\)-differential privacy. This noise adding mechanism was applied for differentially private distributed deep learning in Kumar et al. (2020) and Kumar et al. (2021c).

Deep neural networks outperform classical machine learning techniques in a wide range of applications but their training requires a large amount of data. The issues, such as determining the optimal model structure, smaller training dataset, and iterative time-consuming nature of numerical learning algorithms, are inherent to the neural network-based parametric deep models. The nonparametric approach on the other hand can be promising to address the issue of optimal choice of model structure. However, an analytical solution instead of iterative gradient-based numerical algorithms will be still desired for the learning of deep models. These motivations have led to the development of a nonparametric deep model (Kumar and Freudenthaler 2020; Kumar et al. 2021e, d; Zhang et al. 2022) that is learned analytically for representing data points. The study in Kumar and Freudenthaler (2020) and Kumar et al. (2021e) introduces the concept of \textit{Student-t fuzzy-mapping} which is about representing mappings through a fuzzy set with Student-t type membership function such that the dimension of membership function increases with an increasing data size. A relevant result is that a deep autoencoder model formed via a composition of finite number of nonparametric fuzzy-mappings can be learned analytically. However, (Kumar and Freudenthaler 2020; Kumar et al. 2021e) did not provide a formal mathematical framework for the conceptualization of so-called fuzzy-mapping. The study in Kumar et al. (2021b) provides to fuzzy-mapping a \textit{measure-theoretic} conceptualization and refers it to as \textit{membership-mapping}. The interpolation property of the membership-mapping was exploited for data representation learning for developing an analytical approach to the variational learning of a membership-mappings based data representation model. Further, (Kumar et al. 2021a) uses membership-mapping as the building block of deep models. An alternative idea of deep autoencoder, referred to as Bregman Divergence Based Conditionally Deep Autoencoder (CDMA) 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 CDMA.

This study introduces a novel approach to differentially private transfer and multi-task learning problem. The basic idea of the proposed approach is stated as in Fig. 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 are 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 \textit{Conditionally Deep Membership-Mapping Autoencoder (CDMA)} 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 CDMA.
- 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.
- 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 \textit{“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.

The source and target models in our methodology employ CDMMAs for data representation learning. The CDMA considered in this study is composed of layers such that each layer learns data representation at a certain abstraction level through a membership-mapping autoencoder (Kumar et al. 2021a). The CDMA offers the advantage that an analytical learning solution could be derived via variational optimization methodology to address the issues typically associated with parametric deep models (such as determining the optimal model structure, smaller training dataset, and iterative time-consuming nature of numerical learning algorithms).

The proposed approach and the method for differentially private transfer and multi-task learning is novel and independent of existing methods. To the best knowledge of authors, this is the first study to address sufficiently simultaneously all of the identified requirements (i.e. R1–R6). The core idea of the proposed methodology (that an analytical approach to the learning of CDMA, facilitating high-dimensional data representation learning at varying abstraction levels across CDMA’s different layers, will be applied for transferring knowledge from source to target domain in a privacy-preserving manner) is original. Since the membership-mapping serves as the building block of the CDMA, there is an interest in the development of an efficient algorithm for their learning. Although (Kumar et al. 2021b) provides an algorithm for the variational learning of membership-mappings. An analytical solution to the variational learning of CDMMAs is presented in Sect. 3. The analytical solution derived in Sect. 3 facilitates in Sect. 4 a differentially private semi-supervised transfer and multi-task learning method implementing an optimal noise adding mechanism. The experiments in Sect. 5 provide demonstrative examples (on MNIST and USPS datasets) and the comparison with standard techniques (on Office and Caltech256 datasets). Finally, concluding remarks are given in Sect. 6.

2 Background

2.1 Notations

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

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

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

\[
\mathcal{F}(X) := \{ f : X \rightarrow \mathbb{R} \mid f \text{ is } \mathcal{A} \text{-} B(\mathbb{R}) \text{ measurable} \}. \tag{2}
\]

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

\[
T_x(A) := \{ f \in \mathcal{F}(X) \mid f(x) \in A \}. \tag{3}
\]

- Let \( T \) be the family of cylinder sets defined as

\[
T := \left \{ T_x(A) \mid A \in B(\mathbb{R}^{|x|}), \ x \in S \right \}. \tag{4}
\]

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

\[
\langle g \rangle_\mu := \frac{1}{\int_{\mathbb{R}^N} \mu(y) \, d\lambda_N(y)} \int_{\mathbb{R}^N} g(y) \mu(y) \, d\lambda_N(y). \tag{5}
\]

Table 1 summarizes the introduced notations.

### 2.2 A review of membership-mappings

#### 2.2.1 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 \( A_{\mathcal{X},f}(\tilde{x}) = (\zeta_x \circ f)(\tilde{x}) \) composed of two mappings \( f : \mathcal{X} \to \mathbb{R} \) and \( \zeta_x : \mathbb{R} \to [0, 1] \). \( f \in \mathcal{F}(\mathcal{X}) \) 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 to evaluate how much a sequence \( \tilde{x} = (\tilde{x}^1, \ldots, \tilde{x}^N) \in \mathcal{S} \) matches the attribute induced by observed sequence \( x = (x^1, \ldots, x^N) \in \mathcal{S} \) w.r.t. the feature \( f \) via defining

\[
A_{\mathcal{X},f}(\tilde{x}) = (\zeta_x \circ f)(\tilde{x})
\]
\[
= \zeta_x(f(\tilde{x}^1), \ldots, f(\tilde{x}^N)), \tag{6}
\]

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

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

\[
\text{supp}[\zeta_x] = \mathbb{R}^{|x|}. \tag{8}
\]

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

\[
0 < \int_{\mathbb{R}^{|x|}} \zeta_x \, d\lambda^{|x|} < \infty. \tag{9}
\]

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

\[
\mathbb{P}_{\zeta_x}(A) := \int_{\mathbb{R}^{|x|}} \zeta_x \, d\lambda^{|x|} \int_A \zeta_x \, d\lambda^{|x|} \tag{10}
\]

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

\[
\mathbb{P}_{\zeta_x,a}(A \times [a]) = \mathbb{P}_{\zeta_x}(A). \tag{11}
\]

The collection of membership functions satisfying aforementioned assumptions is denoted by

\[
\Theta := \{ \zeta_x : [0, 1] \to [0, 1] \mid (8), (9), (11), \ x \in \mathcal{S} \}. \tag{12}
\]

#### 2.2.2 Measure space

It is shown in Kumar et al. (2021b) that \( (\mathcal{F}(\mathcal{X}), \sigma(T), \mathbb{P}) \) is a measure space and the probability measure \( \mathbb{P} \) is defined as

\[
\mathbb{P}(T_x(A)) := \mathbb{P}_{\zeta_x}(A), \tag{13}
\]

where \( \zeta_x \in \Theta, x \in \mathcal{S}, A \in \mathcal{B}(\mathbb{R}^{|x|}) \), and \( T_x(A) \in \mathcal{T} \). It follows from Kumar et al. (2021b) that for a given \( B(\mathbb{R}^{|x|}) = B(\mathbb{R}) \) measurable mapping \( g : \mathbb{R}^{|x|} \to \mathbb{R} \), expectation of \( (g \circ f)(x) \) over \( f \in \mathcal{F}(\mathcal{X}) \) w.r.t. probability measure \( \mathbb{P} \) is given as

\[
\mathbb{E}_{\mathbb{P}}[(g \circ f)(x)] = \langle g \rangle_{\zeta_x}. \tag{14}
\]

The significance of equality (14) is to allow calculating averages over all real-valued functions belonging to \( \mathcal{F}(\mathcal{X}) \) via simply computing a weighted average.

#### 2.2.3 Student-t membership-mapping

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

\[
\zeta_x(y) = \left(1 + 1/(v - 2) (y - m_y)^T K_{xx}^{-1} (y - m_y)\right)^{-\frac{v+1}{2}}, \tag{15}
\]

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

\[
(K_{xx})_{i,j} = kr(x^i, x^j),
\]

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

\[
kr(x^i, x^j) = \sigma^2 \exp\left(-0.5 \sum_{k=1}^{n} w_k \left| x^i_k - x^j_k \right|^2 \right),
\]

where \(x^i_k\) is the \(k\)th element of \(x^i\), \(\sigma^2\) is the variance parameter, and \(w_k \geq 0\) (for \(k \in \{1, \ldots, n\}\)).

It is shown in Kumar et al. (2021b) that membership function as defined in (15) satisfies the consistency condition (11).

### 2.2.4 Interpolation

For a zero-mean Student-t membership-mapping \(F \in \mathcal{F}(\mathbb{R}^n)\), let \(x = \{x^i \in \mathbb{R}^n \mid i \in \{1, \ldots, N\}\}\) be a given set of input points and the corresponding mapping outputs are represented by the vector \(f := (F(x^1), \ldots, F(x^N))\). Let \(a = \{a^m \mid a^m \in \mathbb{R}^n, m \in \{1, \ldots, M\}\}\) be the set of auxiliary inducing points and the mapping outputs corresponding to auxiliary inducing inputs are represented by the vector \(u := (F(a^1), \ldots, F(a^M))\). It follows from Kumar et al. (2021b) that \(f\), based upon the interpolation on elements of \(u\), could be represented by means of a membership function, \(\mu_{f;u} : \mathbb{R}^N \to [0, 1]\), defined as

\[
\mu_{f;u}(\tilde{f}) := \left(1 + \frac{(\tilde{f} - \bar{m}_t)^T \left(\frac{(\mu(a^m)K_{aa})^{-1}u - 2\bar{K}_{as}}{\nu + M - 2}\right) \bar{K}_{ss}^{-1} (\tilde{f} - \bar{m}_t)}{\nu + M - 2} \right)^{-\frac{\nu + M - 2}{2}},
\]

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

\[
(K_{aa})_{i,j} = kr(a^i, a^j),
\]

\[(K_{xa})_{i,j} = kr(x^i, a^j),\]

where \(kr : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}\) is a positive definite kernel function defined as in (17).

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

### 2.2.5 Membership-mapping autoencoders

**Definition 2** (Membership-Mapping Autoencoder (Kumar et al. 2021a)) A membership-mapping autoencoder, \(G : \mathbb{R}^p \to \mathbb{R}^p\), maps an input vector \(y \in \mathbb{R}^p\) to \(G(y) \in \mathbb{R}^p\) such that

\[
G(y) := [F_1(y) \ldots F_p(y)]^T,
\]

where \(F_j (j \in \{1, 2, \ldots, p\})\) 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) (Kumar et al. 2021a)) A conditionally deep membership-mapping autoencoder, \(D : \mathbb{R}^p \to \mathbb{R}^p\), maps a vector \(y \in \mathbb{R}^p\) to \(D(y) \in \mathbb{R}^p\) through a nested composition of finite number of membership-mapping autoencoders such that

\[
y^l = (G_l \circ \ldots \circ G_2 \circ G_1)(y), \ \forall l \in \{1, 2, \ldots, L\}
\]

\[
l^* = \arg \min_{l \in \{1, 2, \ldots, L\}} \|y - y^l\|^2
\]

\[
D(y) = y^{l^*},
\]

where \(G_l(\cdot)\) 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 \(D(y)\) 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. 2) is such that
\[ y^j = G_l(y^{j-1}), \]
\[ = \left[ \mathcal{F}_1(p^1y^{j-1}) \ldots \mathcal{F}_p(p^1y^{j-1}) \right]^T, \]
where \( y^0 = y, \) \( p^i \in \mathbb{R}^{n_i \times p} \) is a matrix with \( n_i \in \{1, \ldots, p\} \) such that \( n_1 \geq n_2 \geq \cdots \geq n_L, \) and \( \mathcal{F}_j(\cdot) \) is a Student-t membership-mapping.

Definition 3 provides a special case of Bregman divergence-based conditionally deep autoencoders which were introduced in Kumar et al. (2021a).

### 3 Variational conditionally deep membership-mapping autoencoders

This section is dedicated to the variational learning of CDMMAs. Since CDMMA consists of layers of membership-mappings, a solution to the variational learning of membership-mappings is provided in Sect. 3.1. The solution leads to the design of a learning algorithm for CDMMAs in Sect. 3.2. To deal with the big data, a wide CDMM is presented in Sect. 3.3. The autoencoder-based classification is considered in Sect. 3.4.

#### 3.1 Variational learning of membership-mappings

##### 3.1.1 A modeling scenario

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

\[ y^j \approx \left[ \mathcal{F}_1(x^j) \ldots \mathcal{F}_p(x^j) \right]^T. \] (27)

##### 3.1.2 Disturbances and auxiliary inducing points

For \( j \in \{1, 2, \ldots, p\} \), define

\[ y_j = \left[ y^j_1 \ldots y^j_N \right]^T \in \mathbb{R}^N \] (28)

\[ f_j = \left[ \mathcal{F}_j(x^j) \ldots \mathcal{F}_j(x^N) \right]^T \in \mathbb{R}^N, \] (29)

where \( y^j_i \) denotes the \( j \)th element of \( y^j \). 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.,

\[ v_j = y_j - f_j. \] (30)

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

\[ u_j = \left[ \mathcal{F}_j(a^1) \ldots \mathcal{F}_j(a^M) \right]^T \in \mathbb{R}^M. \] (31)

#### 3.1.3 Membership functional representation approach

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

**Definition 4** (Disturbance model) Disturbance vector \( v_j \) is represented by means of a zero-mean Gaussian membership function as

\[ \mu_{v_j}(\tilde{v}_j) = \exp \left( -0.5\beta \|\tilde{v}_j\|^2 \right), \] (32)

where \( \beta > 0 \) is the precision.

**Definition 5** (Representation of Data \( y_j \) for Given Mappings Output \( f_j \)) Since \( y_j = f_j + v_j \), it follows from (32) that \( y_j \), for given \( f_j \), is represented by means of a membership function, \( \mu_{y_j:f_j} : \mathbb{R}^N \rightarrow [0, 1] \), as

\[ \mu_{y_j:f_j}(\tilde{y}_j) = \exp \left( -0.5\beta|\tilde{y}_j - f_j|^2 \right). \] (33)

**Definition 6** (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, \( \mu_{f_j:u_j} : \mathbb{R}^N \rightarrow [0, 1] \), as

\[ \mu_{f_j:u_j}(\tilde{f}_j) = \left( 1 + \frac{\|\tilde{f}_j - \bar{u}_j\|^2}{\mathcal{K}_{xx}^{-1} + \nu + 2HM^2} \right)^{-\frac{\nu + M - 2}{2}}, \] (34)

\[ \bar{u}_j = \mathcal{K}_{xx}(\mathcal{K}_{uu})^{-1}u_j \] (35)

\[ \mathcal{K}_{xx} = \mathcal{K}_{xx} - \mathcal{K}_{uu}(\mathcal{K}_{uu})^{-1}\mathcal{K}_{xx}^T. \] (36)

**Definition 7** (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, \( \mu_{y_j:u_j} : \mathbb{R}^N \rightarrow [0, 1] \), as

\[ \mu_{y_j:u_j}(\tilde{y}_j) \propto \exp \left( \left\{ \log(\mu_{y_j:f_j}(\tilde{y}_j)) \right\}_{\mu_{f_j:u_j}} \right) \] (37)

where \( \mu_{y_j:f_j} \) is given by (33), \( \mu_{f_j:u_j} \) is defined as in (34), and \( < \cdot > \) is the averaging operation as defined in (5). Thus, \( \mu_{y_j:u_j} \) is obtained from \( \log(\mu_{y_j:f_j}) \) after averaging out the variables \( f_j \) using its membership function. It is shown in Appendix A that
\[
\mu_{y_j;u_j}(\tilde{y}_j) = \exp \left( \log(\mu_{y_j;u_j}(\tilde{y}_j)) \right)_{\mu_{u_j}},
\]

where \( \hat{K}_{u_j}, \tilde{m}_{u_j}(\tilde{y}_j) \) are given by (A2), (A3), respectively, and \( \{/(\tilde{y}_j, u_j)\} \) represents all those terms which are independent of both \( \tilde{y}_j \) and \( u_j \). The constant of proportionality in (38) is chosen to exclude \( \{/(\tilde{y}_j, y_j)\} \)---independent terms in the expression for \( \mu_{y_j;u_j}, \) i.e.,

\[
\mu_{y_j;u_j}(\tilde{y}_j) = \exp \left( -0.5\beta \|\tilde{y}_j\|^2 + (u_j)^T \hat{K}_{u_j}^{-1} \tilde{m}_{u_j}(\tilde{y}_j) - 0.5(u_j)^T \hat{K}_{u_j}^{-1} u_j + 0.5(u_j)^T (K_{aa})^{-1} u_j \right),
\]

(39)

**Definition 8** (Data-Model) \( y_j \) is represented by means of a membership function, \( \mu_{y_j} : \mathbb{R}^N \rightarrow [0, 1] \), as

\[
\mu_{y_j}(\tilde{y}_j) \propto \exp \left( \log(\mu_{y_j;u_j}(\tilde{y}_j)) \right)_{\mu_{u_j}},
\]

(40)

where \( \mu_{y_j;u_j} \) is given by (39) and \( \mu_{u_j} : \mathbb{R}^M \rightarrow [0, 1] \) is a membership function representing \( u_j \). Thus, \( \mu_{y_j} \) is obtained from \( \log(\mu_{y_j;u_j}) \) after averaging out the auxiliary-outputs \( u_j \) using membership function \( \mu_{u_j} \).

### 3.1.4 Variational optimization of data-model

The data model (40) involves the membership function \( \mu_{u_j} \). To determine \( \mu_{u_j} \) for a given \( y_j \), \( \log(\mu_{y_j}(y_j)) \) is maximized w.r.t. \( \mu_{u_j} \) 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 (40) that maximization of \( \log(\mu_{y_j}(y_j)) \) is equivalent to the maximization of \( \log(\mu_{y_j;u_j}(y_j)) \).

**Result 1** The solution of following maximization problem:

\[
\mu_{u_j}^* = \arg \max_{\mu_{u_j}} \left[ \log(\mu_{y_j;u_j}(y_j)) \right]_{\mu_{u_j}} - \left[ \log(\mu_{y_j}(y_j)) \right]_{\mu_{u_j}}
\]

under the fixed integral constraint:

\[
\int_{\mathbb{R}^M} \mu_{u_j} \lambda^M = C_{u_j} > 0,
\]

(42)

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

\[
\mu_{u_j}^*(u_j) = \exp \left( -0.5(u_j - \tilde{m}_{u_j}(y_j))^T \hat{K}_{u_j}^{-1} (u_j - \tilde{m}_{u_j}(y_j)) \right),
\]

(43)

where \( \hat{K}_{u_j} \) and \( \tilde{m}_{u_j}(y_j) \) are given by (A2) and (A3), respectively. The solution of the optimization problem results in

\[
\mu_{y_j}(\tilde{y}_j) \propto \exp \left( -0.5\beta \|\tilde{y}_j\|^2 - 2(\tilde{m}_{u_j}(y_j))^T (K_{aa})^{-1} y_j + (\tilde{m}_{u_j}(y_j))^T (K_{aa})^{-1} \hat{K}_{u_j} (K_{aa})^{-1} \tilde{m}_{u_j}(y_j)
\]

\[
+ (\tilde{m}_{u_j}(y_j))^T Tr(K_{sx}) - Tr((K_{aa})^{-1} K_{aa})^{-1} K_{aa}^{-1} \hat{K}_{u_j} (K_{aa})^{-1} \tilde{m}_{u_j}(y_j)) \right),
\]

(44)

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

**Proof** The proof is provided in Appendix B. \( \square \)

The constant of proportionality in (44) is chosen to exclude \( \{/(y_j, \tilde{y}_j)\} \)---independent terms resulting in

\[
\mu_{y_j}(\tilde{y}_j) = \exp \left( -0.5\beta \|\tilde{y}_j\|^2 - 2(\tilde{m}_{u_j}(y_j))^T (K_{aa})^{-1} y_j + (\tilde{m}_{u_j}(y_j))^T (K_{aa})^{-1} K_{aa}^{-1}\hat{K}_{u_j}(K_{aa})^{-1} \tilde{m}_{u_j}(y_j)
\]

\[
+ (\tilde{m}_{u_j}(y_j))^T Tr(K_{sx}) - Tr((K_{aa})^{-1} K_{aa})^{-1} K_{aa}^{-1}\hat{K}_{u_j}(K_{aa})^{-1} \tilde{m}_{u_j}(y_j)) \right),
\]

(45)

### 3.1.5 Membership-mapping output

**Definition 9** (Averaged Estimation of Membership-Mapping Output) \( \bar{F}_j(x') \) is the i\textsuperscript{th} element of vector \( f_j \) (29) can be estimated as

\[
\bar{F}_j(x') := \left( (f_j)_i \right)_{\mu_{f_j;u_j}}^{\mu_{u_j}}
\]

where \( (f_j)_i \) denotes the \( i \)th element of \( f_j \), \( \mu_{f_j;u_j} \) is defined as in (34), and \( \mu_{u_j}^* \) is the optimal membership function (43) representing \( u_j \). That is, \( \bar{F}_j(x') \), 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

\[
G(x) := \left[ kr(x, a_1) \ldots kr(x, a_{M}) \right],
\]

(47)

where \( kr : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \) is defined as in (17). It is shown in C that

\[
\bar{F}_j(x') = \left( G(x') \right) \left( K_{aa} K_{sx} + \frac{Tr(K_{sx}) - Tr((K_{aa})^{-1} K_{aa})^{-1} K_{aa}^{-1} K_{aa}^{-1} K_{sx})}{v + M - 2} \right) K_{aa}
\]
Let \( \alpha = [\alpha_1 \ldots \alpha_p] \in \mathbb{R}^{M \times p} \) be a matrix with its \( j \)th column defined as

\[
\alpha_j = \left( \frac{K_{aa}^{-1}y_j}{\nu + M - 2} \right) K_{xa}^{-1}y_j.
\]  

(48)

Let \( \alpha = [\alpha_1 \ldots \alpha_p] \in \mathbb{R}^{M \times p} \) be a matrix with its \( j \)th column defined as

\[
\alpha_j = \left( \frac{K_{aa}^{-1}y_j}{\nu + M - 2} \right) K_{xa}^{-1}y_j.
\]  

(49)

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

\[
\hat{F}_j(x^i) = \left( G(x^i) \right) \alpha_j.
\]  

(50)

3.1.6 Choice of parameters

The analytically derived data model (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 follows:

\[
a = \{a^m_m=1 = \text{cluster centroid}(\{x^i\}_{i=1}^N, M), \]

(51)

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

Degrees of freedom The degrees of freedom associated with the Student-t membership-mapping \( \nu \in \mathbb{R} \setminus \{0, 2\} \) is chosen as

\[
\nu = 2.1.
\]  

(52)

Parameters \( (w_1, \ldots, w_n) \) The parameters \( (w_1, \ldots, w_n) \) for kernel function (17) are chosen such that \( w_k \) (for \( k \in \{1, 2, \ldots, n\} \)) is given as

\[
w_k = \left( \max_{1 \leq i \leq N} \left( x^i_k \right) - \min_{1 \leq i \leq N} \left( x^i_k \right) \right)^{-2}
\]  

(53)

where \( x^i_k \) is the \( k \)th element of vector \( x^i \in \mathbb{R}^n \).

Parameters \( M \) and \( \sigma^2 \) Define a scalar-valued function:

\[
\tau(M, \sigma^2) := \frac{Tr(K_{xx}) - Tr((K_{aa})^{-1}K_{xa}K_{xa})}{\nu + M - 2},
\]  

(54)

where \( a \) is given by (51), \( \nu \) is given by (52), and parameters \( (w_1, \ldots, w_n) \) (which are required to evaluate the kernel function for computing matrices \( K_{xx}, K_{aa} \), and \( K_{xa} \)) are given by (53). It follows from the kernel function definition (17) that

\[
\tau(M, \sigma^2) = \sigma^2 \tau(M, 1).
\]  

(55)

It is further observed from (48) that a higher value of \( \tau \) corresponds to a larger level of data smoothing. Therefore, we consider a criterion for choosing \( M \) and \( \sigma^2 \) such that the data smoothing level (i.e. the value of \( \tau \)) should match the variance in the data. In particularly, we pose the requirement that \( \tau \) should be at least as large as the data variance (averaged over dimensions), i.e.,

\[
\tau(M, \sigma^2) \geq \frac{1}{\nu} \sum_{j=1}^{p} \text{var}(y_j^1, \ldots, y_j^N).
\]  

(56)

Using (55), the inequality (56) can be rewritten as

\[
\sigma^2 \geq \frac{1}{\tau(M, 1)} \frac{1}{\nu} \sum_{j=1}^{p} \text{var}(y_j^1, \ldots, y_j^N).
\]  

(57)

The number of auxiliary inducing points \( M \) and the parameter \( \sigma^2 \) for kernel function (17) are so determined that the inequality (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 \( \sigma^2 \) to satisfy the inequality (57).

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

\[
\frac{1}{\beta} = \frac{1}{pN} \sum_{j=1}^{p} \sum_{i=1}^{N} \left| y_j^i - \hat{F}_j(x^i) \right|^2,
\]  

(58)

where \( \hat{F}_j(x^i) \) is the estimated membership-mapping output given as in (50).

3.1.7 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 \( \kappa \).
2. The positive value of \( \tau(M, 1) \) allows steps 9 to 13 to ensure that the inequality (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 \( \alpha \) and \( \beta \).
Algorithm 1 Variational learning of the membership-mappings

Require: Dataset \(\{(x^i, y^i) \mid x^i \in \mathbb{R}^p, y^i \in \mathbb{R}^p, i \in [1, \ldots, N]\}\) and maximum possible number of auxiliary points \(M_{\text{max}} \in \mathbb{Z}_+\) with \(M_{\text{max}} \leq N\).
1: Choose \(v\) and \(w = (w_1, \ldots, w_n)\) as in (52) and (53) respectively.
2: Choose a small positive value \(\kappa = 10^{-1}\).
3: Set iteration count \(i = 0\) and \(M_{[i]} = M_{\text{max}}\).
4: while \(\tau(M_{[i]}, \kappa) < \kappa\) do
5: \(M_{[i+1]} = (0.9M_{[i]})\)
6: \(i \leftarrow i + 1\)
7: end while
8: Set \(M = M_{[i]}\).
9: if \(\tau(M, 1) \geq \frac{1}{p} \sum_{j=1}^p \text{var}(y^1_j, \ldots, y^N_j)\) then
10: \(\sigma^2 = 1\)
11: else
12: \(\sigma^2 = \frac{1}{\tau(M, 1)} \frac{1}{p} \sum_{j=1}^p \text{var}(y^1_j, \ldots, y^N_j)\)
13: end if
14: Compute \(a = (a^m)_{m=1}^M\) using (51), \(K_{xx}\) using (16), \(K_{aa}\) using (21), and \(K_{ax}\) using (22).
15: Set \(\beta = 1\).
16: repeat
17: Compute \(\alpha\) using (49).
18: Update the value of \(\beta\) using (58).
19: until \((\beta\) nearly converges)
20: Compute \(\alpha\) using (49).
21: return the parameters set \(M = \{a, a, M, \sigma, w\}\).

Definition 10 (Membership-Mappings Prediction) Given the parameters set \(M = \{a, a, M, \sigma, w\}\) 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

\[
\hat{y}(x; M) = \left[\mathcal{F}_1(x) \ldots \mathcal{F}_p(x)\right]^T,
\]

where \(\mathcal{F}_j(x)\), defined as in (50), is the estimated output of \(j\)th membership-mapping. It follows from (50) that

\[
\hat{y}(x; M) = \alpha^T(G(x))^T
\]

where \(G(\cdot) \in \mathbb{R}^{1 \times M}\) is a vector-valued function (47).

3.2 Algorithm for variational learning of conditionally deep membership-mapping autoencoders

Since CDMA 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 \(\{y^1, \ldots, y^N\}\), following (Kumar et al. 2021a), Algorithm 2 is stated for the variational learning of CDMA.

Algorithm 2 defines \(\{n_1, \ldots, n_L\}\) to be a monotonically decreasing sequence at step 2. This results in CDMA 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.

Figure 3 illustrates through an example that Algorithm 2 allows high-dimensional data representation learning at varying abstraction levels across CDMA’s different layers.

Definition 11 (CDMA Filtering) Given a CDMA with its parameters being represented by a set \(M = \{M^1, \ldots, M^L\}, \{P^1, \ldots, P^L\}\), the autoencoder can be applied for filtering a given input vector \(y \in \mathbb{R}^p\) as follows:

\[
x^l(y; M) = \left\{ \begin{array}{ll}
P^l y^l & \text{if } l = 1, \\
P^l \hat{y}^{l-1}(x^{l-1}; M^{l-1}) & \text{if } l > 1,
\end{array} \right.
\]

Here, \(\hat{y}^{l-1}\) is the output of the \((l - 1)\)th layer estimated using (60). Finally, CDMA’s output, \(\hat{D}(y; M)\), is given as

\[
\hat{D}(y; M) = \hat{y}^t(x^t; M^t)
\]

\[
I^* = \arg \min_{l \in [1, \ldots, L]} \|y - \hat{y}^l(x^l; M^l)\|^2.
\]

3.3 Wide conditionally deep membership-mapping autoencoder

For big datasets, a wide form of conditionally deep autoencoder has been suggested (Kumar et al. 2021a) where the total data is partitioned into subsets and corresponding to each data-subset a separate CDMA is learned. The final output
Fig. 3 A CDMMA was built using Algorithm 2 (taking \( n = 20; M_{\text{max}} = 500; L = 20 \)) 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.

Fig. 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 \( r_{\text{max}} \) 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 values of \( n \)) are displayed. It is observed that as \( n \) keeps on decreasing, the autoencoder learns increasingly abstract data representation.

Algorithm 3 Variational learning of wide CDMMA

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

**1.** Apply k-means clustering to partition \( Y \) into \( S \) subsets, \( \{Y^1, \ldots, Y^S\} \), where \( S = \lfloor N/1000 \rfloor \).

**2.** for \( s = 1 \) to \( S \) do

**3.** Build a CDMMA, \( M^s \), by applying Algorithm 2 on \( Y^s \) taking \( n \) as the subspace dimension; maximum number of auxiliary points as equal to \( r_{\text{max}} \times \#Y^s \) (where \( \#Y^s \) is the number of data points in \( Y^s \)); and \( L \) as the number of layers.

**4.** end for

**5.** return the parameters set \( P = \{M^s\}_{s=1}^S \).

**Theorem**: \( s^* = \arg \min_{s \in \{1, \ldots, S\}} \|y - \hat{D}(y; M^s)\|^2 \), using (64).

Algorithm 3 requires choosing the values for subspace dimension \( n \) and ratio \( r_{\text{max}} \). Thus, we demonstrate the effect of \( n \) and \( r_{\text{max}} \) on the abstraction level of data representation through examples in Figs. 4 and 5.

### 3.4 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 Defini-
Table 2 A summary of used symbols and their interpretation

| Symbol | Interpretation |
|--------|----------------|
| \( \mathcal{F} \) | Student-t membership-mapping (Definition 1) |
| \( \mathcal{g} \) | Membership-mapping autoencoder (Definition 2) |
| \( D \) | CDMA (Definition 3) |
| \( W_D \) | Wide CDMA (Definition 12) |
| \( C \) | Classifier (Definition 14) |
| \( \mathcal{M} \) | Parameters of membership-mapping returned by Algorithm 1 |
| \( \mathcal{M} \) | Parameters of CDMA returned by Algorithm 2 |
| \( \mathcal{P} \) | Parameters of wide CDMA returned by Algorithm 3 |
| \( \{\mathcal{P}_c\}_{c=1}^C \) | Parameters of classifier returned by Algorithm 4 |

![Fig. 6](image-url) A representation of the relations between different models, parameters, and algorithms

Definition 14 (A Classifier) A classifier, \( C : \mathbb{R}^p \rightarrow \{1, 2, \ldots, C\} \), maps a vector \( y \in \mathbb{R}^p \) to \( C(y) \in \{1, 2, \ldots, C\} \) such that

\[
C(y; \{\mathcal{P}_c\}_{c=1}^C) = \arg\min_{c \in \{1, 2, \ldots, C\}} \|y - \hat{\mathcal{W}}D(y; \mathcal{P}_c)\|^2, \tag{71}
\]

where \( \hat{\mathcal{W}}D(y; \mathcal{P}_c) \), computed using (69), is the output of \( c \)th wide CDMA. 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 \( Y = \{Y_c | c \in \{1, \ldots, C\}\} \), where \( Y_c = \{y_i | y_i \in \mathbb{R}^p \mid i \in \{1, \ldots, N_c\}\} \); the subspace dimension \( n \in \{1, \ldots, p\} \); ratio \( r_{\text{max}} \in (0, 1) \); the number of layers \( L \in \mathbb{Z}_+ \).

1: for \( c = 1 \) to \( C \) do
2: Build a wide CDMA, \( \mathcal{P}_c = \{M_c^{j'}\}_{j'=1}^{S_c} \), by applying Algorithm 3 on \( Y_c \) for the given \( n \), \( r_{\text{max}} \), and \( L \).
3: end for
4: return the parameters set \( \{\mathcal{P}_c\}_{c=1}^C \).

Finally, Table 2 provides a summary and the interpretation of different symbols used for the mathematical formulations. Further, the relations between different models, parameters, and algorithms have been summarized in Fig. 6.

4 Privacy-preserving transferrable deep learning

4.1 An optimal \((\mathcal{E}, \delta)\)–differentially private noise adding mechanism

This subsection reviews an optimal noise adding mechanism that was derived using an information theoretic approach in Kumar et al. (2019). 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 \( Y \in \mathbb{R}^{p \times N} \). 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, \( \mathcal{A} : \mathbb{R}^{p \times N} \rightarrow \mathcal{M} \), where \( \mathcal{M} \) is the model space. That is, for a given dataset \( Y \), the algorithm builds a model \( \mathcal{M} = \mathcal{A}(Y) \). 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 15.

Definition 15 (A Private Algorithm on Data Matrix) Let \( \mathcal{A}^+ : \mathbb{R}^{p \times N} \rightarrow \text{Range}(\mathcal{A}^+) \) be a mapping defined as

\[
\mathcal{A}^+(Y) = \mathcal{A}(Y + V), \quad V \in \mathbb{R}^{p \times N}, \tag{72}
\]

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

\[
\text{Range}(\mathcal{A}^+) = \left\{ \mathcal{A}(Y + V) \mid Y \in \mathbb{R}^{p \times N}, V \in \mathbb{R}^{p \times N} \right\}. \tag{73}
\]
We intend to protect the algorithm \( A^+ \) 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 \( A^+ \) against the perturbation in an element of \( Y \), say \((f_0, i_0)\)th element, such that magnitude of the perturbation is upper bounded by a scalar \( d \). Following (He et al. 2020), the \( d \)-adjacency and \((\epsilon, \delta)\)-differential privacy definitions are provided in Definition 16 and Definition 17, respectively.

**Definition 16** \((d-\text{Adjacency for Data Matrices})\) Two matrices \( Y, Y' \in \mathbb{R}^{p \times N} \) are \( d \)-adjacent if for a given \( d \in \mathbb{R}_+ \), there exist \( i_0 \in [1, 2, \ldots, N] \) and \( j_0 \in [1, 2, \ldots, p] \) such that \( \forall i \in [1, 2, \ldots, N], j \in [1, 2, \ldots, p] \),

\[
|y_j^i - y_j'^i| \leq d, \text{ if } i = i_0, j = j_0
\]

\[
0, \text{ otherwise},
\]

where \( y_j^i \) and \( y_j'^i \) 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 17** \(((\epsilon, \delta)-\text{Differential Privacy for } A^+) \) The algorithm \( A^+ (Y) \) is \((\epsilon, \delta)\)-differentially private if

\[
Pr[\mathcal{A}^+ (Y) \in \mathcal{O}] \leq \exp(\epsilon)Pr[\mathcal{A}^+ (Y') \in \mathcal{O}] + \delta
\]

(74)

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

Intuitively, Definition 17 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 \( A^+ \) only by a factor of \( \exp(\epsilon) \) with probability at least \( 1 - \delta \). Thus, the lower value of \( \epsilon \) and \( \delta \) lead to a higher amount of privacy.

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

\[
f^*_ij (v; \epsilon, \delta, d) = \begin{cases} 
\frac{\delta}{d} \text{Dirac} \delta(v), & v = 0 \\
(1 - \delta) \frac{\epsilon}{2d} \exp(-\frac{\epsilon}{d}|v|), & v \in \mathbb{R} \setminus \{0\}
\end{cases}
\]

(75)

where \( \text{Dirac}\delta(v) \) is Dirac delta function satisfying \( \int_{-\infty}^{\infty} \text{Dirac}\delta(v) \, dv = 1 \).

**Proof** The proof follows from Kumar et al. (2019) and the important steps of the proof are provided in Appendix D for the sake of completeness.

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

\[
F_{ij} (v; \epsilon, \delta, d) = \begin{cases} 
\frac{1-\delta}{2} \exp\left(\frac{\epsilon}{d}v\right), & v < 0 \\
\frac{1-\delta}{2} \exp\left(-\frac{\epsilon}{d}v\right), & v > 0
\end{cases}
\]

(76)

The inverse cumulative distribution function is given as

\[
F^{-1}_{ij} (t; \epsilon, \delta, d) = \begin{cases} 
\frac{d}{\epsilon} \log(\frac{2t_j^\epsilon}{1-t_j^\epsilon}), & t_j^\epsilon < \frac{1-\delta}{2}, \ t_j^\epsilon \in \left[\frac{1-\delta}{2}, \frac{1+\delta}{2}\right], \ t_j^\epsilon \in (0, 1). \\
-\frac{d}{\epsilon} \log(\frac{2(1-t_j^\epsilon)}{1-t_j^\epsilon}), & t_j^\epsilon > \frac{1+\delta}{2}
\end{cases}
\]

(77)

Thus, via generating random samples from the uniform distribution on \((0, 1)\) and using (77), the noise additive mechanism can be implemented.

**Algorithm 5** \( \text{Differentially private approximation of data samples} \)

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

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

\[
y^i_j = y^i_j + F^{-1}_{ij} (t^i_j; \epsilon, \delta, d), \ t^i_j \in (0, 1).
\]

(78)

where \( F^{-1}_{ij} \) is given by (77) and \( y^i_j \) is \( j \)th element of \( y^{\tau} \in \mathbb{R}^p \).

2: return \( Y^+ = \{y^{\tau} \in \mathbb{R}^p \mid i \in [1, \ldots, N]\} \).

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

4.2 \( \text{Differentially private semi-supervised transfer and multi-task learning} \)

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 \( \{Y^s_c\}_{c=1}^C \) be the labelled source dataset, where \( Y^s_c = \{y^s_i^c \in \mathbb{R}^{p^s} \mid i \in [1, \ldots, N^s_c]\} \) represents \( c \)th labelled samples. The target
dataset consist of a few labelled samples \( \{Y^{tg}_{i,c}\}_{i=1}^{C} \) (with \( Y^{tg}_{i,c} = \{y^{tg}_{i,c} \in \mathbb{R}^{p_{tg}} \mid i \in \{1, \ldots, N^{tg}_{i,c}\}\} \)) and another set of unlabelled samples \( Y^{tg}_{*} = \{y^{tg}_{*} \in \mathbb{R}^{p_{tg}} \mid i \in \{1, \ldots, N^{tg}_{*}\}\} \). 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: \( Y^+ = \{Y^+_c \mid c \in \{1, \ldots, C\}\} \); the subspace dimension \( n \in \{1, \ldots, p\} \); ratio \( r_{max} \in (0, 1) \); the number of layers \( L \in \mathbb{Z}_+ \).

1. Build a classifier, \( \mathcal{P}^C_{c \mid k} \), by applying Algorithm 4 on \( Y^+ \) for the given \( n, r_{max}, \) and \( L \).
2. Return \( \mathcal{P}^C_{c \mid k} \).

For a given differential privacy parameters: \( d, \epsilon, \delta \); Algorithm 5 is applied on \( Y^+_c \) to obtain the differentially private approximated data samples, \( Y^{+sr}_{c} = \{y^{+sr}_{c,i} \in \mathbb{R}^{p_{sr}} \mid i \in \{1, \ldots, N^{sr}_{c}\}\} \), for all \( c \in \{1, \ldots, C\} \). Algorithm 6 is applied on \( \{Y^{+sr}_{c \mid k}\}_{k=1}^{C} \) to build a differentially private source domain classifier characterized by parameters sets \( \mathcal{P}^{+sr}_{c \mid k} \).

**Differentially private source domain latent subspace transformation-matrix** For a lower-dimensional representation of both source and target samples, a subspace dimension, \( n_{sr} \in \{1, 2, \ldots, \min(p_{sr}, p_{tg})\} \), is chosen. Let \( V^{+sr} \in \mathbb{R}^{n_{sr} \times p_{sr}} \) 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 \( V^{tg} \in \mathbb{R}^{n_{tg} \times p_{tg}} \) 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 heterogeneous 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:

\[
Y^{sr} \rightarrow Y^{tg} = \left( (V^{sr})^T V^{tg} \right) y^{tg}, \quad p_{sr} = p_{tg}
\]

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

\[
Y^{tg} \rightarrow sr := \{ y_{tg} \mid y_{tg} \in Y^{tg} \}
\]

\[
Y^{sr} \rightarrow tg := \{ y_{sr} \mid y_{sr} \in Y^{sr} \}
\]

**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 \( k \in \{1, \ldots, it_{\text{max}}\} \), consists of following updates:

\[
\{\mathcal{P}^{tg}_{c \mid k}\}_{k=1}^{C} = \text{Algorithm 4}
\]

\[
\mathcal{P}^{tg}_{c \mid k} = \left( \left( Y^{sr} \rightarrow sr \cup Y^{tg} \rightarrow sr \right)_{i=1}^{C} \right)_{c=1}^{C}, n_{k}, r_{max}, L
\]

\[
Y^{sr} \rightarrow c := \left\{ y^{sr}_{i,c} \mid y^{sr}_{i,c} \in Y^{sr} \rightarrow sr \mid C \cup Y^{tg} \rightarrow sr \rightarrow \mathcal{P}^{tg}_{c \mid k_{c=1}} \right\} = c, \quad i \in \{1, \ldots, N^{sr}_{c}\}
\]

where \{1, n, 2, \ldots \} 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. 4), \( n \) values are chosen as to form a monotonically non-decreasing sequence.

**Source2Target model** The target samples associated with a class can be filtered through the source domain autoencoder associated with the same class for defining the following dataset:

\[
\mathcal{D} := \left\{ \left( \tilde{W}\overline{D}(y; P^{+sr}_{c}), y \right) \mid y \in \left\{ Y^{sr} \rightarrow sr \cup Y^{tg} \rightarrow sr \mid it_{\text{max}} \right\}, \quad c \in \{1, \ldots, C\} \right\}
\]

where \( \tilde{W}\overline{D}(\cdot; \cdot) \) is defined as in (69), \( Y^{tg} \rightarrow sr \) is defined as in (80), and \( Y^{sr} \rightarrow tg \) is defined as in (83). Here, \( \tilde{W}\overline{D}(y; P^{+sr}_{c}) \),
where \( y \in \{ y^C_{c \rightarrow sr} |_{it_{\max}} \} \), is a representation of a \( c \)-th labelled target sample \( y \) in the source domain \( c \)-th labelled data space represented by the wide CDMMMA \( P^C_{c \rightarrow sr} \). The mapping from source to target domain can be learned via building a variational membership-mappings based model using Algorithm 1 on the dataset \( D \). That is,

\[
M^{sr \rightarrow tg} = \text{Algorithm 1} (D, M_{\max})
\]

\[
M_{\max} = \min(\lceil N^{tg}/2 \rceil, 1000),
\]

where \( N^{tg} = |D| \) 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 with a target sample \( y_{tg \rightarrow sr} \) as

\[
\hat{c}(y_{tg \rightarrow sr}; \{ P^C_{c \rightarrow sr} \}_{c=1}^C, {M^{sr \rightarrow tg}}) = \arg \min_{c \in \{1, 2, \ldots, C\}} \left\{ \left\| y_{tg \rightarrow sr} - \hat{y}(\overline{W^C_{sr \rightarrow tg}}; y_{tg \rightarrow sr}; P^C_{c \rightarrow sr}) \right\|^2, \left\| y_{tg \rightarrow sr} - \hat{y}(\overline{W^C_{sr \rightarrow tg}}; y_{tg \rightarrow sr}; P^C_{c \rightarrow sr}) \right\|^2 \right\},
\]

where \( \hat{y}(\cdot; M^{sr \rightarrow tg}) \) is the output of source2target model computed using (60). That is, \( y_{tg \rightarrow sr} \) is assigned the \( c \)-th class label, if

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

5 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 implementation details for the method are described as below:

- We study experimentally the differential privacy (Definition 17) 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 \( \epsilon \) with probability at least \( 1 - \delta \). That is, \( d \) is taken as equal to 1 for defining adjacent matrices in Definition 17.
- Algorithm 5, for a given \( d \) and \((\epsilon, \delta)\), 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 \( \min(20, \min\{n_{c}\} - 1) \) (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 (82) and (83) till a finite number of iterations with a monotonically non-decreasing subspace dimension sequence \( \{n_{1}, n_{2}, \ldots\} \) and a ratio \( r_{\max} \in (0, 1) \). For our experiments, the subspace dimension sequence is chosen as \( \{5, 10, 15, 20\} \) and \( r_{\max} = 0.5 \).
- The label associated with a target data point is predicted under transfer and multi-task learning scenarios using (87).

5.1 Demonstrative examples using MNIST and USPS datasets

5.1.1 MNIST dataset

Our first experiment is on the widely used MNIST digits dataset containing 28 \( \times \) 28 sized images divided into training set of 60,000 images and testing set of 10,000 images. The images’ pixel values were divided by 255 to normalize the values in the range from 0 to 1. The 28 \( \times \) 28 normalized values of each image were flattened to an equivalent 784-dimension data vector. The transfer learning experiment was carried in the same setting as in Papernot et al. (2017) where 60,000 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 \( \epsilon \) 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 $\epsilon$ are provided in Fig. 7.

It is observed from Fig. 7 that $\epsilon = 0.1$ is sufficiently low to preserve privacy in this case. Thus, the experiments were carried out with privacy-loss bound $\epsilon = 0.1$ while keeping failure probability fixed at $\delta = 1e^{–5}$.

**Competitive performance** Table 3 reports the experimental results. The proposed method is able to learn 95.1% accurate model together with providing (0.1, 1e$–5$)—differential privacy guarantee, which is a better result than the existing result (Abadi et al. 2016) of achieving 90% accuracy for (0.5, 1e$–5$)—differential privacy on MNIST dataset.

5.1.2 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 $16 \times 16 =$ (256) grayscale pixels.

**MNIST $\rightarrow$ 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 $\rightarrow$ USPS semi-supervised transfer learning problem was previously studied in Belhaj et al. (2018). For a comparison, MNIST $\rightarrow$ USPS problem was considered in the same experimental setting as in Belhaj et al. (2018) 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 4 reports the results of 10 independent MNIST $\rightarrow$ USPS experiments. As observed in Table 4, 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 Belhaj et al. (2018). Further, the proposed method’s consistent performance over a wide range of privacy-loss bound $\epsilon$ (from 0.1 to 1) 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$\rightarrow$MNIST problem is considered with number of labelled target samples varying from 100 to 500. Table 5 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 (87), leads to an increasing performance with increasing labelled target sample size while preserving the privacy of source domain data.

5.2 The choice of parameters

The updating of the target domain classifier using (82) and (83) requires to choose a monotonically non-decreasing subspace dimension sequence $\{n_1, n_2, \ldots\}$ and a ratio $r_{\text{max}} \in (0, 1]$. To study the effect of these parameters, the experiments were performed with the following different choices:

$$\{n_1, n_2, \ldots\} \in \{[10, 20], [5, 10, 15, 20], [4, 8, \ldots, 20], [2, 4, \ldots, 20]\}$$

$$r_{\text{max}} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}.$$  

Table 6 and Table 7 list the classification accuracies for different choices of subspace dimension sequence and ratio $r_{\text{max}}$. The bottom rows in Table 6 and Table 7 provide mean and standard deviation of accuracy over $r_{\text{max}}$ for a given subspace dimension sequence. It is observed that the sequence $\{5, 10, 15, 20\}$ results in the best performance in both MNIST dataset experiments and MNIST $\rightarrow$ USPS experiments. Thus, for all our experiments the subspace dimension sequence has been chosen as $\{5, 10, 15, 20\}$.

5.3 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 (Hoffman et al. 2013; Herath et al. 2017; Karbalayghareh et al. 2018; Hoffman et al. 2014) 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 Hoffman et al. (2013), Herath et al. (2017), Karbalayghareh et al. (2018) and Hoffman et al. (2014):

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 3 Privacy and utility results on MNIST dataset

| Method                                           | $(\epsilon, \delta)$     | Classification accuracy |
|--------------------------------------------------|---------------------------|-------------------------|
| proposed transfer and multi-task learning (Abadi et al. 2016) | (0.1, $10^{-5}$)         | 95.1%                   |
|                                                  | (0.5, $10^{-5}$)         | 90%                     |

The second column reports the privacy-loss bound $\epsilon$ and failure probability $\delta$ of $(\epsilon, \delta)$-differential privacy guarantee.

### Table 4 Results of 10 independent MNIST→USPS experiments expressed in average accuracy ± standard deviation

| Method                                           | Accuracy (in %)       |
|--------------------------------------------------|-----------------------|
| $(0.1, 10^{-5})$—differentially private proposed | 92.23 ± 0.87          |
| $(1, 10^{-5})$—differentially private proposed   | 92.28 ± 0.62          |
| Non-private proposed                             | 92.37 ± 0.70          |
| Non-private Deep Variational Transfer (Belhaj et al. 2018) | 92.03 ± 0.38          |

### Table 5 Effect of labelled target sample size on the performance of $(0.1, 10^{-5})$—differentially private proposed method in USPS→MNIST problem

| Number of labelled target samples | Classification accuracy on target testing data (%) |
|----------------------------------|---------------------------------------------------|
| 100                              | 92.29                                             |
| 200                              | 95.86                                             |
| 300                              | 97.31                                             |
| 400                              | 97.63                                             |
| 500                              | 97.99                                             |

### Table 6 The effect of target domain classifier parameters on classification accuracy during experiment on MNIST dataset

| $r_{max}$ | Subspace dimension sequence | Classification accuracy (%) |
|-----------|-----------------------------|----------------------------|
|           | $[10, 20]$                  | $[5, 10, 15, 20]$          | $[4, 8, \ldots, 20]$ | $[2, 4, \ldots, 20]$ |
| 0.1       | 0.9290                      | 0.9550                     | 0.9510               | 0.9470               |
| 0.2       | 0.9310                      | 0.9500                     | 0.9620               | 0.9590               |
| 0.3       | 0.9400                      | 0.9550                     | 0.9580               | 0.9590               |
| 0.4       | 0.9390                      | 0.9540                     | 0.9570               | 0.9500               |
| 0.5       | 0.9390                      | 0.9510                     | 0.9570               | 0.9490               |
| 0.6       | 0.9340                      | 0.9550                     | 0.9310               | 0.9540               |
| 0.7       | 0.9400                      | 0.9540                     | 0.9520               | 0.9340               |
| 0.8       | 0.9370                      | 0.9570                     | 0.8820               | 0.9430               |
| 0.9       | 0.9370                      | 0.9350                     | 0.9200               | 0.9390               |
| Mean ± std| 0.9362 ± 0.0040             | **0.9518 ± 0.0066**        | 0.9411 ± 0.0262      | 0.9482 ± 0.0086      |

### Table 7 The effect of target domain classifier parameters on classification accuracy during MNIST→USPS experiments

| $r_{max}$ | Subspace dimension sequence | Classification accuracy (%) |
|-----------|-----------------------------|----------------------------|
|           | $[10, 20]$                  | $[5, 10, 15, 20]$          | $[4, 8, \ldots, 20]$ | $[2, 4, \ldots, 20]$ |
| 0.1       | 0.8959                      | 0.9158                     | 0.9043               | 0.8919               |
| 0.2       | 0.8994                      | 0.9223                     | 0.9198               | 0.9093               |
| 0.3       | 0.9033                      | 0.9243                     | 0.9258               | 0.9108               |
| 0.4       | 0.9033                      | 0.9233                     | 0.9243               | 0.9312               |
| 0.5       | 0.8974                      | 0.9213                     | 0.9183               | 0.9043               |
| 0.6       | 0.9023                      | 0.9168                     | 0.8670               | 0.9218               |
| 0.7       | 0.8999                      | 0.9208                     | 0.8844               | 0.9043               |
| 0.8       | 0.8969                      | 0.9153                     | 0.9098               | 0.9183               |
| 0.9       | 0.9008                      | 0.9183                     | 0.8794               | 0.9188               |
| Mean ± std| 0.8999±0.0028               | **0.9198 ± 0.0033**        | 0.9037 ± 0.0216      | 0.9123 ± 0.0116      |
Fig. 7: An example of the noise added to a randomly selected sample from MNIST dataset for different values of $\epsilon$ with $\delta = 1e^{-5}$

Fig. 8: The box plots of accuracies obtained by different methods on “Office+Caltech256” dataset

Following (Herath et al. 2017), 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) (Herath et al. 2017): 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 (Tsai et al. 2016): The Cross-Domain Landmark Selection (CDLS) method derives a domain-invariant feature subspace for heterogeneous domain adaptation.
4. MMDT (Hoffman et al. 2014): 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 (Li et al. 2014): The Heterogeneous Feature Augmentation (HFA) method learns common latent subspace and a classifier under max-margin framework.
6. OBTL (Karbalayghareh et al. 2018): 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 Hoffman et al. (2013), Herath et al. (2017), Karbalayghareh et al. (2018) and Hoffman et al. (2014) 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 with “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 9, Table 10, Table 11, Table 12, Table 13, Table 14, Table 15, Table 16, Table 17, Table 18, Table 19, and Table 20 report the results and the first two best performances have been marked. For a comparison of the proposed privacy-preserving method with existing non-private methods, the results of all 12 experiments have been summarized in Fig. 8 through box plots of accuracies obtained by different methods. The best performance of the proposed $(0.1, 1e^{-5})$—differentially private method is observed in Fig. 8. Further, Table 8 reports the performance of top performing methods. As observed in Table 8, the proposed method remains as best performing in maximum number of experiments. The most remarkable result
### Table 8
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 |
|---------------------------------------|------------------------------------------------------|---------------------------------------------------------|
| Private proposed ($\epsilon = 0.1$, $\delta = 1e-5$) | 7                                                    | 3                                                       |
| Non-private ILS (1-NN)                | 5                                                    | 5                                                       |
| Non-private CDLS                      | 1                                                    | 2                                                       |

### Table 9
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      | 80.6         |
| SVM-t (without knowledge transfer)               | VGG-FC6      | 73.4         |
| Non-private ILS (1-NN)                          | VGG-FC6      | **83.3**     |
| 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 10
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      | **91.2**     |
| SVM-t (without knowledge transfer)               | VGG-FC6      | 90.0         |
| 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         |

### Table 11
Accuracy (in %, averaged over 20 experiments) obtained in *amazon*→*webcam*
semi-supervised transfer learning experiments

| Method                                           | Feature type | Accuracy (%) |
|--------------------------------------------------|--------------|--------------|
| (0.1, 1e−5)—differentially private proposed     | VGG-FC6      | 89.5         |
| SVM-t (without knowledge transfer)               | VGG-FC6      | 86.9         |
| Non-private ILS (1-NN)                          | VGG-FC6      | **91.2**     |
| Non-private CDLS                                | 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         |
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.

### 6 Concluding remarks

This study has outlined a novel approach to differentially private semi-supervised transfer and multi-task learning based on CDMMAs. The CDMMAs facilitate high-dimensional data representation learning at varying abstraction levels across its different layers and can be applied for transferring...
Table 15  Accuracy (in %, averaged over 20 experiments) obtained in dslr→amazon semi-supervised transfer learning experiments

| Method                        | Feature type | Accuracy (%) |
|-------------------------------|--------------|--------------|
| (0.1, 1e−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 16  Accuracy (in %, averaged over 20 experiments) obtained in dslr→caltech256 semi-supervised transfer learning experiments

| Method                        | Feature type | Accuracy (%) |
|-------------------------------|--------------|--------------|
| (0.1, 1e−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 17  Accuracy (in %, averaged over 20 experiments) obtained in dslr→webcam semi-supervised transfer learning experiments

| Method                        | Feature type | Accuracy (%) |
|-------------------------------|--------------|--------------|
| (0.1, 1e−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         |
knowledge from source to target domain in a privacy-preserving setting. The proposed variational membership-mapping based approach 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 verify that our approach is capable of achieving a low privacy-loss bound without letting the accuracy be much degraded.
Acknowledgements The research reported in this paper has been partly supported by EU Horizon 2020 Grant 826278 “Securing Medical Data in Smart Patient-Centric Healthcare Systems” (Serums); Austrian Research Promotion Agency (FFG) COMET-Modul S3AI (Security and Safety for Shared Artificial Intelligence); FFG Grant 873979 “Privacy Preserving Machine Learning for Industrial Applications” (PRIMAL); and the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH.

Data availability statement The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

Appendix A: Evaluation of membership function

Using (34), we have

\[
\frac{\partial}{\partial \gamma} \log(\mu_{y_j; u_j}(\tilde{y}_j)) \bigg|_{\mu_{y_j; u_j}} = -0.5\beta \left\| \tilde{y}_j - \tilde{m}_{f_j} \right\|^2 -0.5\beta \frac{v + (u_j)^T (K_{aa})^{-1} u_j - 2}{v + M - 2} Tr(\tilde{K}_{xx}),
\]

where \( Tr(\cdot) \) denotes the trace operator. Using (35) and (36),

\[
\frac{\partial}{\partial \gamma} \log(\mu_{y_j; u_j}(\tilde{y}_j)) \bigg|_{\mu_{y_j; u_j}} = -0.5\beta \left\| \tilde{y}_j \right\|^2 + \beta(\tilde{y}_j)^T K_{xa}(K_{aa})^{-1} u_j -0.5\beta (u_j)^T (K_{aa})^{-1} K_{xa}^T K_{xa}(K_{aa})^{-1} u_j -0.5\beta \frac{v + (u_j)^T (K_{aa})^{-1} u_j - 2}{v + M - 2} \left( Tr(K_{xx}) - Tr((K_{aa})^{-1} K_{xa}^T K_{xa}) \right).
\]

Using (37),

\[
\mu_{y_j; u_j}(\tilde{y}_j) \propto \exp \left( -0.5\beta \left\| \tilde{y}_j \right\|^2 + \beta(\tilde{y}_j)^T K_{xa}(K_{aa})^{-1} u_j -0.5\beta (u_j)^T (K_{aa})^{-1} K_{xa}^T K_{xa}(K_{aa})^{-1} u_j -0.5\beta \frac{v + (u_j)^T (K_{aa})^{-1} u_j - 2}{v + M - 2} \left( Tr(K_{xx}) - Tr((K_{aa})^{-1} K_{xa}^T K_{xa}) \right) + \frac{1}{(\tilde{y}_j, u_j)} \right).
\]

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

\[
\hat{K}_{uj} = \left( (K_{aa})^{-1} + \beta (K_{aa})^{-1} K_{xa}^T K_{xa} (K_{aa})^{-1} \right)^{-1} + \beta \frac{Tr(K_{xx}) - Tr((K_{aa})^{-1} K_{xa}^T K_{xa}) (K_{aa})^{-1}}{v + M - 2} \left( (K_{aa})^{-1} \right)^{-1}
\]

\[
\hat{m}_{uj}(\tilde{y}_j) = \beta \hat{K}_{uj}(K_{aa})^{-1} (K_{xa})^T \tilde{y}_j
\]

to express \( \mu_{y_j; u_j}(\tilde{y}_j) \) as (38).

Appendix B: 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 \( \gamma \):

\[
\mathcal{J} = \left\{ (u_j)^T \hat{K}_{uj}^{-1} \hat{m}_{uj}(y_j) - 0.5(u_j)^T \hat{K}_{uj}^{-1} u_j + 0.5(u_j)^T (K_{aa})^{-1} u_j - \log(\mu_{uj}(u_j)) \right\}_{\mu_{uj}} + \gamma \left\{ \int_{\mathbb{R}^M} \mu_{uj}(u_j) \ d\lambda^M(u_j) - C_{uj} \right\}
\]

\[
= \frac{1}{C_{uj}} \int_{\mathbb{R}^M} \mu_{uj}(u_j) \ d\lambda^M(u_j) \left\{ (u_j)^T \hat{K}_{uj}^{-1} \hat{m}_{uj}(y_j) - 0.5(u_j)^T \hat{K}_{uj}^{-1} u_j - \log(\mu_{uj}(u_j)) \right\} + \gamma \left\{ \int_{\mathbb{R}^M} \mu_{uj}(u_j) \ d\lambda^M(u_j) - C_{uj} \right\}.
\]

Setting the functional derivative of \( \mathcal{J} \) w.r.t. \( \mu_{uj} \) equal to zero,

\[
0 = \gamma + \frac{1}{C_{uj}} \left\{ -1 - 0.5(u_j)^T \hat{K}_{uj}^{-1} u_j + (u_j)^T \hat{K}_{uj}^{-1} \hat{m}_{uj}(y_j) - \log(\mu_{uj}(u_j)) \right\}.
\]

That is,

\[
\mu_{uj}(u_j) = \exp(\gamma C_{uj} - 1) \exp \left( -0.5(u_j)^T \hat{K}_{uj}^{-1} u_j + (u_j)^T \hat{K}_{uj}^{-1} \hat{m}_{uj}(y_j) \right).
\]

The optimal value of \( \gamma \) is obtained by solving \( \int_{\mathbb{R}^M} \mu_{uj} \ d\lambda^M \)

\[
= C_{uj}.
\]

\[
\text{exp}(\gamma C_{uj} - 1) \sqrt{(2\pi)^M / |\hat{K}_{uj}^{-1}|} \exp \left( 0.5 \left( \hat{m}_{uj}(y_j) \right)^T \hat{K}_{uj}^{-1} \hat{m}_{uj}(y_j) \right)
\]

\[
= C_{uj}.
\]
Thus, the optimal expression for $\mu_{u_j}$ is given as

$$
\mu_{u_j}^*(u_j) = C_u \sqrt{\frac{\hat{K}_{u_j}}{2\pi(M)}} \exp \left( -0.5(u_j - \hat{\mu}_{u_j}(y_j))^T \hat{K}_{u_j}^{-1} (u_j - \hat{\mu}_{u_j}(y_j)) \right).
$$

Finally, $C_u$ is chosen such that $\max_{u_j} \mu_{u_j}^*(u_j) = 1$. This results in

$$
\mu_{u_j}^*(u_j) = \exp \left( -0.5(u_j - \hat{\mu}_{u_j}(y_j))^T \hat{K}_{u_j}^{-1} (u_j - \hat{\mu}_{u_j}(y_j)) \right).
$$

Thus, $\langle u_j \rangle_{\mu_{u_j}^*} = \hat{\mu}_{u_j}(y_j)$, and using (A3), we get

$$
\langle u_j \rangle_{\mu_{u_j}^*} = \beta \hat{K}_{u_j}(K_{aa})^{-1}(K_{xa})^T y_j.
$$

It follows from (A2) and (A3) that

$$
\hat{K}_{u_j}^{-1} - (K_{aa})^{-1} = \beta (K_{aa})^{-1} K_{xa}^T K_{xa} (K_{aa})^{-1} \\
+ \beta \frac{Tr(K_{xx}) - Tr(((K_{aa})^{-1} K_{xa})^T K_{xa})}{\nu + M - 2} (K_{aa})^{-1} \\
\hat{K}_{u_j}^{-1} \hat{\mu}_{u_j}(y_j) = \beta (K_{aa})^{-1} (K_{xa})^T y_j.
$$

Using (B12) and (B13) in (39), we have

$$
\log(\mu_{y_j;u_j}(\hat{y}_j)) =
-0.5\beta \|\hat{y}_j\|^2 + \beta \langle u_j \rangle^T (K_{aa})^{-1} K_{xa} (K_{aa})^{-1} \\
+ \frac{Tr(K_{xx}) - Tr(((K_{aa})^{-1} K_{xa})^T K_{xa})}{\nu + M - 2} (K_{aa})^{-1} \langle u_j \rangle.
$$

Thus, $\log(\mu_{y_j;u_j}(\hat{y}_j))_{\mu_{u_j}^*}$ is given as

$$
\log(\mu_{y_j;u_j}(\hat{y}_j))_{\mu_{u_j}^*} = -0.5\beta \|\hat{y}_j\|^2 + \beta \langle \hat{\mu}_{u_j}(y_j) \rangle^T (K_{aa})^{-1} K_{xa} (K_{aa})^{-1} \\
+ \frac{Tr(K_{xx}) - Tr(((K_{aa})^{-1} K_{xa})^T K_{xa})}{\nu + M - 2} (K_{aa})^{-1} \hat{\mu}_{u_j}(y_j).
$$

The data-model (40) using (B15) becomes as

$$
\mu_{y_j}(\hat{y}_j) \propto \exp \left( -0.5\beta \|\hat{y}_j\|^2 + \beta \langle \hat{\mu}_{u_j}(y_j) \rangle^T (K_{aa})^{-1} K_{xa} (K_{aa})^{-1} \\
- 0.5\beta \left( \hat{\mu}_{u_j}(y_j) \right)^T \frac{Tr(K_{xx}) - Tr(((K_{aa})^{-1} K_{xa})^T K_{xa})}{\nu + M - 2} (K_{aa})^{-1} \hat{\mu}_{u_j}(y_j) \\
+ \frac{Tr(K_{xx}) - Tr(((K_{aa})^{-1} K_{xa})^T K_{xa})}{\nu + M - 2} (K_{aa})^{-1} \hat{\mu}_{u_j}(y_j))
$$

Thus, (44) follows.

### Appendix C: Membership-mapping output estimation

Using (34) and (35), we have

$$
\{f_{\mu_{u_j}^*}\} = (K_{aa})^{-1} u_j.
$$

Thus,

$$
\bar{F}(x_i) = G(x_i) (K_{aa})^{-1} \{u_j \}_{\mu_{u_j}^*}.
$$

Using (B11) in (C20), we have

$$
\bar{F}(x_i) = \beta \left( G(x_i) \right) (K_{aa})^{-1} \hat{K}_{u_j} (K_{aa})^{-1} (K_{xa})^T y_j.
$$

Substituting $\hat{K}_{u_j}$ from (A2) in (C21), we get (48).

### Appendix D: Proof of result 2

**Result 3** (Sufficient Conditions for $(\epsilon, \delta)$—Differential Privacy (Kumar et al. 2019)) The following conditions on the probability density function of noise $v_j \in \mathbb{R}$ are sufficient to attain $(\epsilon, \delta)$—differential privacy by algorithm $A^+$ (Definition 15):

$$
\int_{\Theta} f_{v_j'}(v) \, dv \geq 1 - \delta, \quad \text{where}
$$

$$
\Theta \triangleq \left\{ v \mid \sup_{d \in [-d,d]} \frac{f_{v_j'}(v)}{f_{v_j}(v)} \leq \exp(\epsilon), \quad f_{v_j'}(v) \neq 0, \quad v_j' \in \mathbb{R} \right\}.
$$
The sufficient conditions for \( \epsilon \)–differential privacy follow from (D22) with \( \delta = 0 \) as

\[
\int_{\Theta} f_{\nu_j}^* (v) \, dv = 1,
\]

(D24)

\[
\Theta = \left\{ v \mid \sup_{d \in [-d, d]} \frac{f_{\nu_j^-d}^* (v)}{f_{\nu_j}^* (v)} \leq \exp(\epsilon), \quad f_{\nu_j}^* (v) \neq 0, \, v_j \in \mathbb{R} \right\},
\]

(D25)

where \( \Theta \) is defined as in (D23). The equality in (D24) is due to the fact that the integral of any probability density function over a subset cannot exceed unity.

**Result 4** (Minimum Magnitude for a Given Entropy Level (Kumar et al. 2019)) The probability density function of noise that, for a given level of entropy, minimizes the expected noise magnitude is given as

\[
f_{\nu_j}^* (v; h) = \frac{1}{\exp(h-1)} \exp \left( - \frac{2|v|}{\exp(h-1)} \right),
\]

(D26)

where \( h \) is the given entropy level. The expected noise magnitude is given as

\[
E_{f_{\nu_j}^*} [|v|] (h) = \frac{1}{2} \exp(h-1).
\]

(D27)

**Result 5** (An Optimal \( \epsilon \)–Differentially Private Noise (Kumar et al. 2019)) The probability density function of noise that minimizes the expected noise magnitude together with satisfying the sufficient conditions for \( \epsilon \)–differential privacy is given as

\[
f_{\nu_j}^* (v) = \frac{\epsilon}{2d} \exp(-\frac{\epsilon}{d}|v|).
\]

(D28)

It is obvious that the optimal noise density function (D28) satisfies the sufficient conditions (D22–D23) with \( \Theta = \mathbb{R} \) for any \( \delta \in [0, 1] \) and thus attain \( (\epsilon, \delta) \)–differential privacy for any \( \delta \in [0, 1] \). However, in this case (i.e. when \( \Theta = \mathbb{R} \) and \( \delta > 0 \)), the lower bound on \( \int_{\Theta} f_{\nu_j}^* (v) \, dv \) in (D22) is not tight. Therefore, we need to derive an optimal density function for \( (\epsilon, \delta) \)–differential privacy taking \( \Theta \subset \mathbb{R} \). Let \( \nu_0 \in \mathbb{R} \) be a point which is excluded from \( \mathbb{R} \) to define \( \Theta \), i.e.,

\[
\Theta = \mathbb{R} \setminus \{ \nu_0 \}.
\]

(D29)

We extend the solution space for optimization by considering the discontinuous distributions having an arbitrary probability mass \( r \) at an arbitrary point \( \nu_0 \). Let \( f_{\nu_j}^* (v; \nu_0, r, q_{\nu_j} (v)) \) be an arbitrary density function defined as 

\[
f_{\nu_j}^* (v; \nu_0, r, q_{\nu_j} (v)) = \begin{cases} r \text{Dirac}(v - \nu_0), & v = \nu_0 \\ (1-r)q_{\nu_j} (v), & v \in \Theta \end{cases}
\]

(D30)

Here, \( q_{\nu_j} (v) \) is an arbitrary density function with a continuous cumulative distribution function and satisfying the sufficient conditions (D24–D25) for \( \epsilon \)–differential privacy. As \( q_{\nu_j} (v) \) is an arbitrary density function, the expected noise magnitude for \( q_{\nu_j} (v) \) must be greater than or equal to the optimal value \( d/\epsilon \), i.e.,

\[
\int_{\mathbb{R}} |v| q_{\nu_j} (v) \, dv \geq \frac{d}{\epsilon}
\]

(D31)

\[
\int_{\Theta} |v| q_{\nu_j} (v) \, dv + \int_{\{\nu_0\}} |v| q_{\nu_j} (v) \, dv \geq \frac{d}{\epsilon}.
\]

(D32)

Here, the integral over a single point is equal to zero because of a continuous cumulative distribution function associated with \( q_{\nu_j} (v) \). Thus,

\[
\int_{\Theta} |v| q_{\nu_j} (v) \, dv \geq \frac{d}{\epsilon}
\]

(D33)

where equality occurs if \( q_{\nu_j} (v) \) is equal to (D28). Also,

\[
\int_{\Theta} q_{\nu_j} (v) \, dv = \int_{\mathbb{R}} q_{\nu_j} (v) \, dv - \int_{\{\nu_0\}} q_{\nu_j} (v) \, dv = 1.
\]

(D34)

(D35)

Thus,

\[
\int_{\Theta} f_{\nu_j}^* (v; \nu_0, r, q_{\nu_j} (v)) \, dv = 1 - r.
\]

(D36)

For the density function (D30) to satisfy condition (D22), we must have

\[
1 - r \geq 1 - \delta.
\]

(D37)

The expected noise magnitude for the density function (D30) is given as

\[
E_{f_{\nu_j}^*} [|v|] (\nu_0, r, q_{\nu_j} (v)) = r |\nu_0| + (1-r) \int_{\Theta} \frac{|v| q_{\nu_j} (v) \, dv}{\geq 0 \geq 1-\delta \geq d/\epsilon}.
\]

(D38)

It follows immediately that \( E_{f_{\nu_j}^*} [|v|] \) is minimized together with satisfying the sufficient conditions (D22–D23) with the
following optimal choices for \((\delta, \epsilon, q^*_j(v))\) with \(r^* = \delta\), and \(q^*_j(v) = \frac{\epsilon}{\delta^2} \exp(-\frac{\epsilon}{\delta^2} |v|)\). The result is proved after putting the optimal values into (D30).

References

Abadi M, Chu A, Goodfellow I, et al (2016) Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, New York, USA, pp 308–318. https://doi.org/10.1145/2976749.2978318

Acs G, Melis L, Castelluccia C, et al (2017) Differentially private mixture of generative neural networks. In: 2017 IEEE International Conference on Data Mining (ICDM), pp 715–720

Balle B, Wang Y (2018) Improving the Gaussian mechanism for differential privacy: analytical calibration and optimal denoising. CoRR abs/1805.06530

Belhaj M, Protopapas P, Pan W (2018) Deep variational transfer: transfer learning through semi-supervised deep generative models. arXiv:1812.03123

Bruzzone L, Marconcini M (2010) Domain adaptation problems: a dasvm classification technique and a circular validation strategy. IEEE Trans Pattern Anal Mach Intell 32(5):770–787

Courty N, Flamary R, Tuia D et al (2017) Optimal transport for domain adaptation. IEEE Trans Pattern Anal Mach Intell 39(9):1853–1865

Dai W, Wang Q, Xie GR, et al (2007) Boosting for transfer learning. In: Proceedings of the 24th international conference on machine learning. Association for Computing Machinery, New York, USA, ICML ’07, pp 193–200

Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407. https://doi.org/10.1561/0400000042

Dwork C, Kenthapadi K, McSherry F et al (2006) Our data, ourselves: deidentifying MacCoy’s health records. In: 2006 ACM International Conference on Privacy, Security and Trust, pp 25–34

Dwork C, Roth A, Tygar JD et al (2006) Fair information exchange with intractable utility. In: 2006 International Conference on Financial Cryptography and Data Security, pp 36–52

Geng Q, Viswanath P (2016) The optimal noise-adding mechanism in differential privacy. IEEE Trans Inf Theory 62(2):925–951. https://doi.org/10.1109/TIT.2015.2504967

Geng Q, Viswanath P (2016) Optimal noise adding mechanisms for approximate differential privacy. IEEE Trans Inf Theory 62(2):952–969. https://doi.org/10.1109/TIT.2015.2504972

Geng Q, Kairouz P, Oh S et al (2015) The staircase mechanism in differential privacy. IEEE J Sel Top Signal Process 9(7):1176–1184. https://doi.org/10.1109/JSTSP.2015.2425831

Geng Q, Ding W, Guo R et al (2018) Optimal noise-adding mechanism in additive differential privacy. CoRR abs/1809.10224

Ghosh A, Roughgarden T, Sundararajan M (2012) Universally utility-maximizing privacy mechanisms. SIAM J Comput 41(6):1673–1693. https://doi.org/10.1137/09076828X

Gong B, Shi Y, Sha F et al (2012) Geodesic flow kernel for unsupervised domain adaptation. In: 2012 IEEE conference on computer vision and pattern recognition, pp 2066–2073

Gupta M, Sundararajan M (2010) Universally optimal privacy mechanisms for minimax agents. In: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. ACM, New York, USA, PODS ’10, pp 135–146. https://doi.org/10.1145/1807085.1807105

He J, Cai L, Guan X (2020) Differential private noise adding mechanism and its application on consensus algorithm. IEEE Trans Signal Process 68:4069–4082. https://doi.org/10.1109/TSP.2020.3006760

Herath S, Harandi M, Porikli F (2017) Learning an invariant Hilbert space for domain adaptation. In: The IEEE conference on computer vision and pattern recognition (CVPR)

Hoffman J, Rodner E, Donahue J et al (2013) Efficient learning of domain-invariant image representations. CoRR abs/1301.3224

Hoffman J, Rodner E, Donahue J et al (2014) Asymmetric and category invariant feature transformations for domain adaptation. Int J Comput Vis 109(1):28–41. https://doi.org/10.1007/s11263-014-0719-3

Ji Z, Elkan C (2013) Differential privacy based on importance weighting. Mach Learn 93(1):163–183

Karbalayghareh A, Qian X, Dougherty ER (2018) Optimal Bayesian transfer learning. IEEE Trans Signal Process 66(14):3724–3739

Kumar M, Freudenthaler B (2020) Fuzzy membership functional analysis for nonparametric deep models of image features. IEEE Trans Fuzzy Syst 28(12):3345–3359. https://doi.org/10.1109/TFUZZ.2019.2950636

Kumar M, Rossbory M, Moser BA et al (2019) Deriving an optimal noise adding mechanism for privacy-preserving machine learning. In: Anderst-Kotsis G, Tjoa AM, Khalil I et al (eds) Proceedings of the 3rd international workshop on cyber-security and functional safety in cyber-physical (IWCFS 2019), August 26–29, 2019, Linz, Austria. Springer, Cham, pp 108–118

Kumar M, Rossbory M, Moser BA et al (2020) Differentially private learning of distributed deep models. In: Adjunct Publication of the 28th ACM conference on user modeling, adaptation and personalization. Association for Computing Machinery, New York, USA, UMAP ’20 Adjunct, pp 193–200. https://doi.org/10.1145/3386392.3399562

Kumar M, Moser B, Fischer L et al (2021) Membership-mappings for data representation learning: a Bregman divergence based conditionally deep autoencoder. In: Kotsis G, Tjoa AM, Khalil I et al (eds) Database and expert systems applications—DEXA 2021 workshops. Springer, Cham, pp 138–147

Kumar M, Moser B, Fischer L et al (2021) Membership-mappings for data representation learning: measure theoretic conceptualization. In: Kotsis G, Tjoa AM, Khalil I et al (eds) Database and expert systems applications—DEXA 2021 workshops. Springer, Cham, pp 127–137

Kumar M, Rossbory M, Moser BA et al (2021) An optimal \((\epsilon, \delta)\)—differentially private learning of distributed deep fuzzy models. Inf Sci 546:87–120. https://doi.org/10.1016/j.ins.2020.07.044

Kumar M, Singh S, Freudenthaler B (2021) Gaussian fuzzy theoretic analysis for variational learning of nested compositions. Int J Approx Reason 131:1–29. https://doi.org/10.1016/j.ijar.2020.12.021

Kumar M, Zhang W, Weippe D et al (2021) An explainable fuzzy theoretic nonparametric deep model for stress assessment using heartbeat intervals analysis. IEEE Trans Fuzzy Syst 29(12):3873–3886. https://doi.org/10.1109/TFUZZ.2020.3029284

Li W, Duan L, Xu D et al (2014) Learning with augmented features for nonparametric deep neural networks. J Mach Learn Res 17(1):2096–2030

Long M, Cao Y, Wang J et al (2015) Learning transferable features with deep adaptation networks. In: Bach F, Blei D (eds) Proceedings of the 28th conference on neural information processing systems applications—DEXA 2015 workshops. Springer, Cham, pp 308–318.https://doi.org/10.1145/146.https://doi.org/10.1145/1807085.1807105
Long M, Zhu H, Wang J, et al (2016) Unsupervised domain adaptation with residual transfer networks. In: Proceedings of the 30th international conference on neural information processing systems. Curran Associates Inc., Red Hook, USA, NIPS’16, pp 136–144
Niinimäki T, Heikkilä MA, Honkela A et al (2019) Representation transfer for differentially private drug sensitivity prediction. Bioinformatics 35(14):i218–i224
Papernot N, Abadi M, Erlingsson U, et al (2017) Semi-supervised knowledge transfer for deep learning from private training data. In: ICLR. OpenReview.net. http://dblp.uni-trier.de/db/conf/iclr/iclr2017.html#PapernotAEGT17
Phan N, Wang Y, Wu X, et al (2016) Differential privacy preservation for deep auto-encoders: An application of human behavior prediction. In: Proceedings of the thirtieth AAAI conference on artificial intelligence. AAAI Press, AAAI’16, pp 1309–1316. http://dl.acm.org/citation.cfm?id=3015812.3016005
Tsai YH, Yeh Y, Wang YF (2016) Learning cross-domain landmarks for heterogeneous domain adaptation. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), pp 5081–5090
Wang Y, Gu Q, Brown DE (2018) Differentially private hypothesis transfer learning. In: Berlingerio M, Bonchi F, Gärtner T et al (eds) Machine learning and knowledge discovery in databases—European conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II, vol 11052. Lecture Notes in Computer Science. Springer, pp 811–826
Xie L, Lin K, Wang S, et al (2018) Differentially private generative adversarial network. arXiv:1802.06739
Zhang J, Cormode G, Procopiuc CM, et al (2017) Privbayes: Private data release via Bayesian networks. ACM Trans Database Syst. https://doi.org/10.1145/3134428
Zhang W, Kumar M, Ding W et al (2022) Variational learning of deep fuzzy theoretic nonparametric model. Neurocomputing 506:128–145

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.