ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Internet Of Things

VARIATIONAL AUTOENCODERS AND META-LEARNING: TRANSFORMING FEDERATED LEARNING IN IOT ENVIRONMENTS

CANDIDATE

Gianluca Di Tuccio

SUPERVISOR

Prof. Angelo Trotta

CO-SUPERVISOR

Prof. Marco Di Felice

Academic year 2022-2023

Session 4th

Abstract

This dissertation investigates the integration of Variational AutoEncoders and Meta-Learning in Federated Learning, particularly in the IoT domain, where data heterogeneity and limited samples pose significant challenges. The Class Informed-VAE (CI-VAE) is introduced, establishing a federated latent space that harmonizes data distributions of different clients into a unified distribution that also enables the generation of synthetic samples and labels. The proposed model attains performance on par with the conventional local and federated models across image datasets (FEMNIST) and sample-based datasets (HAR). Furthermore, the thesis explores the effectiveness of Meta-Learning (particularly the Reptile algorithm) in improving model performance with limited data, showcasing its potential in scenarios with few samples per class.

Contents

1 Introduction **1**

2 Theory and State of the Art **4**

2.1 AutoEncoder and VAE 4

2.2 Federated Learning 12

2.3 Meta Learning 16

2.3.1 Reptile Algorithm (Batched Version) 18

3 System Descriptions **19**

3.1 Federated ClassInformed-VAE 19

3.1.1 Linear Classifier in CI-VAE 23

3.1.2 CI-VAE in FL 24

3.2 Federated Reptile CI-VAE 26

4 Datasets **30**

4.1 FEMNIST 30

4.2 HAR 32

5 Results **34**

5.1 FEMNIST (with Rotation) 34

5.1.1 Rotation challange 43

5.2 FEMNIST (w/out Rotation) 44

5.3 HAR 46

Conclusion **49**

Bibliography **55**

Chapter 1

Introduction

Several primary limitations of today's AI in the IoT domain stem from concerns about data privacy, due to the transmission of user data from devices to a Server, and the challenge of creating a model capable of adapting to the diverse data distributions of clients. This challenge arises from data imbalance, biases, distinct features, etc., inherent in client datasets. One potential solution has been the deployment of local models trained directly on devices, thereby mitigating privacy issues and enabling distributed training. However, these models often lacked the ability to adapt to different users with distinct input distributions. **Federated Learning** has partially addressed these limitations by allowing each client to perform local training of their model, typically a classifier. The model weights from the clients are subsequently transmitted to a server, which aggregates and redistributes them for additional training iterations. This process fosters models that can adapt to a broader range of users. However, as can be understood, these models still struggle with adaptability in contexts with heterogeneous data: a classifier may fit well with certain clients but not with others, because conventional classifiers learn the mapping from input to output labels and, hence, have limited capacity to comprehend the underlying data distribution. There's no mechanism or space to associate similar inputs. A straightforward example is in the MNIST dataset, where each client in the network has differently rotated digits. Discriminative models fail to recognize that digits '4' with different rotations represent the same information. Furthermore, IoT datasets often have limited samples, necessitating the generation of new samples and, sometimes, making accurate classification challenging.

The primary motivation for this work is to address the existing limitations of Federated Learning by employing **Variational AutoEncoders** (VAEs). These VAEs facilitate a shared latent space among different clients, transforming distinct information expressed through diverse distributions into a common distribution, while simultaneously generating new samples. Specifically, this work proposes the integration of a classifier directly into the latent space of the VAE. This approach aims to establish a shared latent space for all clients, enabling classification over a uniform distribution. Additionally, the realm of **Meta-Learning** will be explored to assess how it can enhance performance through the application of **few-shot learning**.

This work is structured as follows:

i. a chapter dedicated to the theory and foundations behind the technologies used, starting with VAEs, and moving on to the concept of Federated Learning and Meta Learning;

ii. a chapter referring to the architectures proposed in this thesis, particularly focusing on the use of **ClassInformed-VAE** (CI-VAE) in a novel context like Federated Learning, showcasing all its advantages, and on the use of **Reptile**, a Meta Learning algorithm developed by OpenAI, not yet tested in a Federated Learning context;

iii. subsequent chapters will present the datasets used to test the previous architectures, such as FEMNIST and HAR;

iv. finally, the results for the previous datasets will be presented, highlighting the strengths and weaknesses of the four main models (a local classifier, a federated classifier, and the proposed Fed CI-VAE model with and without Meta Learning). Notably, these models will also be tested on a variant of FEMNIST, where each client presents an added complexity, the rotation of the images.

The corresponding code and results are also presented in the reference

GitHub repository [1]. The hardware employed for this thesis comprised a PC equipped with a 12th generation i5 CPU, a 2080 Ti GPU, and 64GB of DDR5 RAM.

Chapter 2

Theory and State of the Art

In this chapter, the technologies and frameworks used in this dissertation are presented. The focus is on the theory of Variational Autoencoders (VAE), which are used for transforming different input distributions into a same shared latent space, thereby enabling the creation of data samples that retain the unique characteristics and distributions of each client. The chapter also investigates Federated Learning (FL), an innovative approach that allows for the collaborative training of models across multiple clients while safeguarding data privacy and sustains strong performance with unseen clients with greater adaptability to diverse data distributions. Moreover, Meta Learning techniques are discussed, aiming to enhance client performance in few-shot learning environments: this aim to solve the disparities in sample sizes among users and improving model performance with a limited set of labels.

2.1 AutoEncoder and VAE

The primary architecture employed in this dissertation is the Variational AutoEncoder, an advanced variant of the classical AutoEncoder. AutoEncoders [2] belong to a category of models in the deep learning domain, which are adept at learning features; specifically, they excel at extracting unique characteristics that define a given input, compressing these features, and subsequently reconstructing the input. This process involves dimensionality reduction, where the input is mapped to a lower-dimensional space compared to the original input space. Consequently, this facilitates representing a given input with fewer pieces of information while extrapolating the variation information in the data.

As a result, the model is categorized as a lossy model due to the information loss between input and output. The primary applications of this model include input dimensionality reduction (as shown in Figure 2.1), anomaly detection, and denoising.

The architecture comprises an Encoder, a compressed space Z and a Decoder. These components can be mathematically described as $E : \mathbb{R}^n \rightarrow \mathbb{R}^m$ for the Encoder and $D : \mathbb{R}^m \rightarrow \mathbb{R}^n$ for the Decoder, where $m \leq n$. The purpose of the Encoder is to reduce the dimensionality of the input (from n dimensions to m), the intermediate layer, called the "code" or simply Z, encapsulates the compressed representation of the input in m dimensions, and finally, the Decoder is responsible for reconstructing the data, mapping it from the m dimensional space back to the n dimensional space, with an inherent loss of information.

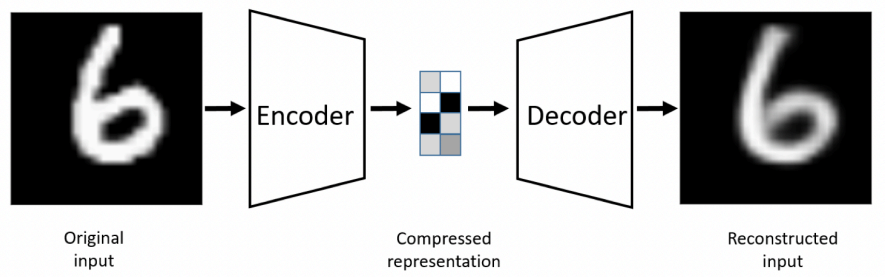

Figure 2.1: Architecture of the AutoEncoder for the digit 6, using the MNIST dataset (Image taken from [2])

Initially, AutoEncoders were also employed for data generation, however, the compressed variables Z learned during this process did not yield meaningful insights, as the model learned a mapping between the input and these Z variables without acquiring any semantic or relationships among various inputs. For example, in the MNIST dataset [3], digits 6, drawn by different users, were represented in the Z layer with distinctly different values; this was contrary to the expectation of a strong correlation among various instances of the digit 6 from different users and a weak correlation between digits representing

different numbers. This issue was subsequently addressed with the introduction of **Variational AutoEncoders** (VAE), which brought a more structured and meaningful interpretation of Z, now called **latent space**, while still retaining the structure of the encoder and decoder from classical AutoEncoders.

The purpose of VAE is to replicate the model distributions of data, occasionally complex, denoted as $p(\mathbf{x})$ with $\mathbf{x}$ a vector. This distribution can be articulated as follows:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{2.1}$$

The terms within the integral may be construed as a generative approach [4]. Indeed, there are two potential approaches to elucidate the argument of the integral (Figure 2.2): a discriminative and a generative approach. The first

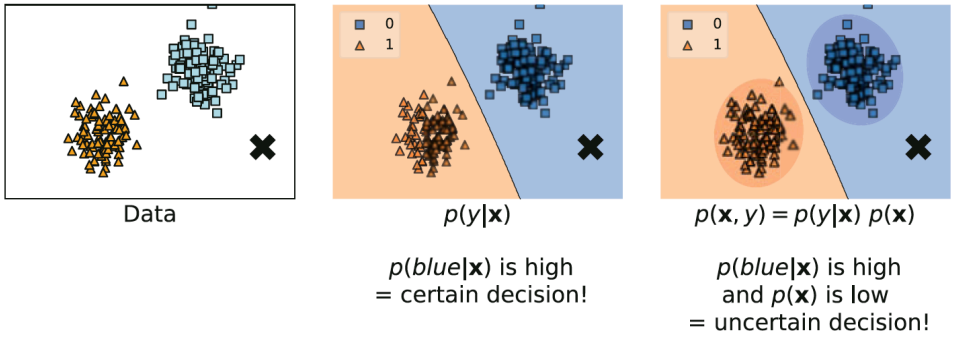

Figure 2.2: a) Description of two distinct data labels in a 2D space where X identifies the current input; b) discriminative approach, the probability of X belonging to the blue group is high, thus, according to the decision making, it is a definite and secure decision; c) generative approach, the point X is located in the blue zone, but far from the region of blue points, hence it is an uncertain decision (Image taken from [4])

approach can be articulated as $p(y|\mathbf{x})$ where $\mathbf{x}$ denotes the input and y represents the input's class (for instance, $y = 0, 1, 2, 3, ..., 9$ for the MNIST dataset). The limitation of this method is its inability to provide a level of "certainty" in the classification. Consequently, a second factor is introduced, symbolizing the class distribution: $p(y|\mathbf{x})p(\mathbf{x})$, with the latter term being referred to as the **prior probability**; if the probability of a class for a given input is high, but the

probability of the input itself is low (i.e., it is situated far from the distribution, akin to an outlier), the resulting product is a low value, thereby indicating an uncertain decision. This same concept is expressed in Equation 2.1, where the distribution $p(\mathbf{x})$ is essentially the aggregate of all possible product distributions computed among $\mathbf{z}$.

Then, the second step is how to compute Equation 2.1 and, in general, it is a difficult task. Indeed, the integral is **intractable** due to the necessity of evaluating it over the entire latent variables $\mathbf{z}$. While theoretical computation is possible, in practice, it proves to be excessively slow and costly, akin to attempting to deduce a student's University of Bologna password through mere guesswork. One simple approach would be to use the **Monte Carlo approximation**: it involves generating a large number of random samples from the distribution and then computing the average value of the function [5]. For an accurate approximation, it is necessary to sample a lot of points and with the current technology it is very easy to sample many points in reasonably short time. However, if $\mathbf{z}$ is multidimensional, the required number of samples to adequately cover the space increases exponentially (i.e. **curse of dimensionality**); if we take too few samples, then the approximation is very poor. Due to the fact that many models have a complex distribution, the Monte Carlo approximation is not an optimal method.

An alternative approach, employed in VAE, involves **variational inference** [6, 7, 8]: an observable variable $\mathbf{x}$ is conditioned upon another hidden random variable $\mathbf{z}$, which constitutes the latent space. This latent space is typically modeled as a multivariate Gaussian distribution, characterized by its means and variances. A more precise way to express this concept would be to learn the conditional probability $p(\mathbf{z}|\mathbf{x})$. Using the chain rule of probability, the previous conditional probability could be expressed as:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \tag{2.2}$$

The issue with Equation 2.2 lies in the requirement of having a tractable $p(\mathbf{x})$ which brings us back to the original problem. The solution is manifested in the ability to find an approximation to $p(\mathbf{z}|\mathbf{x})$, such that $p(\mathbf{z}|\mathbf{x}) \approx q_\varphi(\mathbf{z}|\mathbf{x})$ with q an approximate posterior probability parameterized by φ. q can be computed as follow:

$$\log(p(\mathbf{x})) = \log(p(\mathbf{x})) \tag{2.3}$$

$$\log(p(\mathbf{x})) = \log(p(\mathbf{x})) \cdot \int q_\varphi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \tag{2.4}$$

$$\log(p(\mathbf{x})) = \int \log(p(\mathbf{x})) \cdot q_\varphi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \tag{2.5}$$

where Equation 2.3 is simply multiplied by 1, i.e. $\int q_\varphi(\mathbf{z}|\mathbf{x})d\mathbf{z}$, and in Equation 2.5 the $\log(p(\mathbf{x}))$ is brought inside the integral.

$$\log(p(\mathbf{x})) = E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log(p(\mathbf{x}))\right] \tag{2.6}$$

$$\log(p(\mathbf{x})) = E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right] \tag{2.7}$$

$$\log(p(\mathbf{x})) = E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z}) q_\varphi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x}) q_\varphi(\mathbf{z}|\mathbf{x})}\right] \tag{2.8}$$

$$\log(p(\mathbf{x})) = E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] + E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\varphi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right] \tag{2.9}$$

$$\log(p(\mathbf{x})) = E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] + D_{KL}\left(q_\varphi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{2.10}$$

where in 2.6 the definition of expectation is used, then the chain rule is applied in 2.7, while in 2.8 both numerator and denominator are multiplied by the same term, i.e. by 1, and in 2.9 the property of logarithms is used. Finally, the second term of the right part in 2.10 is, by definition, the **KL divergence** [9], that is always ≥ 0. Due this fact, the 2.10 could be written as:

$$\log(p(\mathbf{x})) \geq E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] \tag{2.11}$$

where the expectation value is called **Evidence Lower Bound** (ELBO) [10].

Using again the chain rule, the properties of logarithms and again the definition of KL divergence, the final result will be:

$$\log(p(\mathbf{x})) \geq E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] \tag{2.12}$$

$$\log(p(\mathbf{x})) \geq E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] + E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] \tag{2.13}$$

$$\log(p(\mathbf{x})) \geq E_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left(q_\varphi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) \tag{2.14}$$

In 2.14 the expectation value is the reconstruction likelihood of the decoder (where $p(\mathbf{x}|\mathbf{z})$ could be seen as a distribution parametrized modeled by θ: $p_\theta(\mathbf{x}|\mathbf{z})$, i.e. the decoder) of $\mathbf{z}$ sampled from $q_\varphi(\mathbf{z}|\mathbf{x})$, as described previously in Figure 2.2. This setup quantifies the "certainty" in reconstructing the observed data $\mathbf{x}$ from latent variables $\mathbf{z}$ under the approximate posterior $q_\varphi(\mathbf{z}|\mathbf{x})$, as shown in Figure 2.3. The KL term, instead, is the distance between the approximate posterior distribution $q_\varphi(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$, pushing $q_\varphi(\mathbf{z}|\mathbf{x})$ to become closer to $p(\mathbf{z})$, that is zero when the distributions are equal. Maximizing the ELBO means to maximize the decoder reconstruction and at the same time minimizing KL divergence (e.g., the formula could be seen as a Profit equation, where maximizing Profit means maximizing Revenues and minimizing Costs). In deep learning, maximizing the ELBO means to adjust the weights of the model along the gradient direction, with the classical Stochastic Gradient Descend [11]. However, the gradient of ELBO is a bit problematic [12]: the estimator for ELBO exhibits high variance (i.e. unstable gradient) and it can not be backpropagated in the neural network due to its sampling phase. The solution adopted is the **reparameterization trick** [12, 13], as proposed in Figure 2.4.

The final architecture of the VAE can be delineated with an Encoder, which generates $\Sigma_{\mathbf{z}|\mathbf{x}}$ and $\mu_{\mathbf{z}|\mathbf{x}}$, followed by an n-dimensional latent space, where n

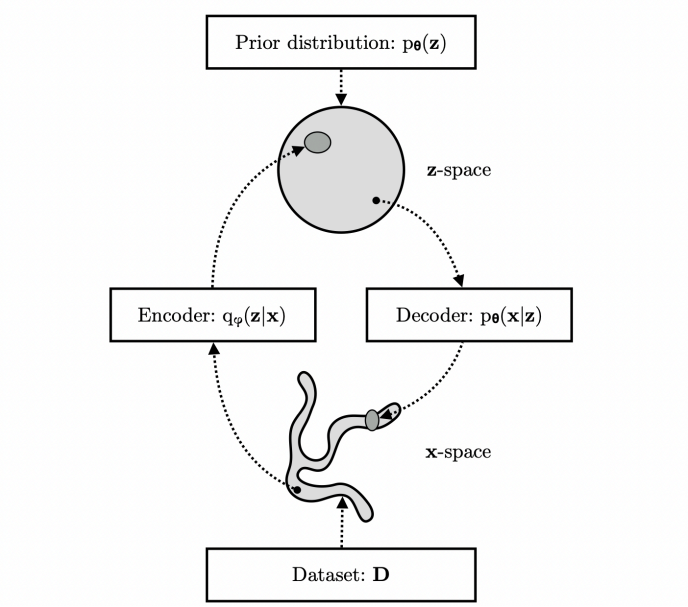

Figure 2.3: A VAE learns the map between an observed $\mathbf{x}$-space, with distribution $q_D(\mathbf{x})$, and a latent $\mathbf{z}$-space; the generative part learns $p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, while the encoder part, also called inference model, approximates the true but intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ using $q_\varphi(\mathbf{z}|\mathbf{x})$ (Image taken from [6])

trated as a hyperparameter. This is succeeded by the Decoder, as shown in Figure 2.5. As previously mentioned, the reconstructed $\hat{\mathbf{x}}$ differs from the original $\mathbf{x}$ due to the inherent lossy nature of the process. For loss functions, Mean Squared Error (MSE) or Mean Absolute Error (MAE) are typically chosen for input reconstruction, and the Kullback-Leibler (KL) divergence is employed to approximate $q_\varphi(\mathbf{z}|\mathbf{x})$ to $p(\mathbf{z})$, usually using frameworks like Keras [16] or PyTorch.

VAEs are currently employed for generative purposes, often in image creation, by sampling in the latent space. This process can involve direct sampling in the cluster of a specific input class, or by sampling in the intermediate areas between different clusters, that share common features: this technique allows for the generation of new samples that incorporate features from multiple clusters, as illustrated in Figure 2.6. This capability extends beyond the scope of what was possible with traditional AutoEncoders.

Besides, various versions of VAEs exist, such as the β-VAE, in which the

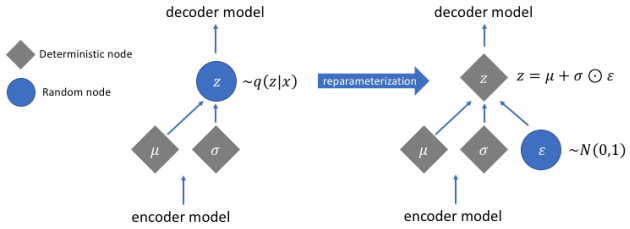

Figure 2.4: Graphical comparison of the traditional sampling in VAE (left) versus the reparameterization trick (right). On the left, $\mathbf{z}$ is directly sampled from the approximate posterior $q(\mathbf{z}|\mathbf{x})$, which is non-differentiable. On the right, the reparameterization trick is used to sample $\mathbf{z}$ by adding a deterministic transformation of two separate nodes: μ and σ, representing the mean and standard deviation of the approximate posterior, and ϵ, a random node sampled from a standard normal distribution $\mathcal{N}(0, 1)$. This allows the backpropagation of the gradient (Image taken from [13])

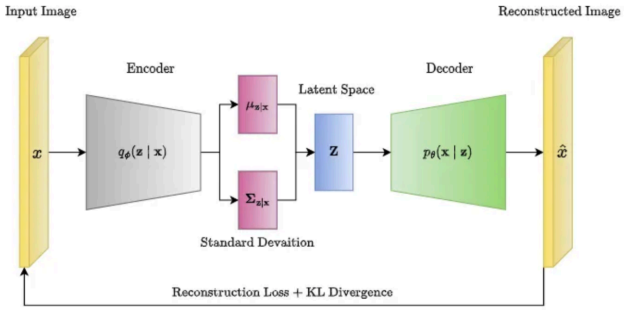

Figure 2.5: An overview of the VAE architecture, from the initial input $\mathbf{x}$ to the reconstructed, **lossy**, $\hat{\mathbf{x}}$. It incorporates the Encoder, which generates $\Sigma_{\mathbf{z}|\mathbf{x}}$ and $\mu_{\mathbf{z}|\mathbf{x}}$ based on the input or a set of inputs, followed by the application of the reparameterization trick for sampling. Then, the Decoder reconstructs the original input(s) (Image taken from [14])

KL divergence is multiplied by a constant β treated as a hyperparameter. Additionally, there are Conditional VAEs (CVAE) [17], where label information is also passed into the encoder, and MultiEncoder-VAEs (ME-VAE), which are used even in the biomedical domain [18].

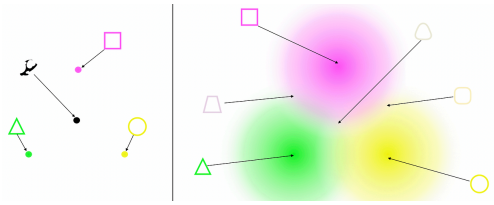

Figure 2.6: AutoEncoder (left) vs VAE (right). AutoEncoders learn the map between input and **z** without explicit consideration for the semantic distribution of the encoded features (a randomly sampled point may not correspond to a meaningful feature set, as shown in black), while VAEs encodes input data into a structured latent space in a probabilistic manner, allowing for the generation of new samples with different features from different clusters (Image taken from [15])

2.2 Federated Learning

In the realm of artificial intelligence, spatial considerations [19] play a crucial role in the structuring and implementation of AI systems. With the growing diversity and volume of data sources, as well as the increasing need for privacy and efficient data processing, the choice of AI architecture has become a significant factor in the design of intelligent systems. This chapter begins by exploring the three main spatial architectures in AI, each characterized by its approach to data processing and model training. The three main spatial options in AI are as follows:

i. **Centralized AI**: This approach involves centralized AI in a single location or Server. It typically relies on a powerful central computer or a cloud-based system to perform AI tasks and manage data. This model is efficient for handling large-scale data processing but can be limited by bandwidth and latency issues;

ii. **Distributed AI**: in this model, AI processing is distributed across multiple systems or nodes. Each node can perform tasks independently and may communicate with other nodes to share information or results. This approach is beneficial for tasks that require parallel processing and can provide resilience and scalability;

iii. **Federated AI**: federated AI involves training algorithms across multiple decentralized devices or servers holding local data samples, without exchanging them. This approach is especially useful for privacy preservation, as it allows AI models to learn from a vast amount of data without the data ever leaving its original location.

The last (i.e. Federated Learning) is an emerging paradigm in the field of AI. It distinguishes itself primarily through its focus on privacy and distributed training. Here, the model training process is decentralized, occurring across a multitude of devices or nodes: each node contributes to the learning process using its local data, without the need to share its data with a central Server or other nodes. This unique characteristic significantly enhances user privacy, because sensitive data never leaves its original location. Other remarkable advantages of Federated Learning is the ability to uncover insights and features during the federated training that a local model might fail to detect: a model trained only on local data might lack the diversity and breadth required to make accurate predictions or recognize patterns. Moreover, Federated Learning addresses a key challenge faced by local models: their limited ability to adapt to users with scarce or no labeled data. By leveraging models trained across various nodes, Federated Learning can effectively learn from a wide array of data sources, each contributing different perspectives and knowledge. This aggregated learning process results in a model that is not only more robust and generalizable but also capable of adapting to new, unseen data more effectively than a model trained in local. Besides, Federated Learning have two distinct approaches for the training: the first involves sending the model from one client to another, executing Stochastic Gradient Descent (SGD or others) on a mini-batch or an epoch. The second approach entails sending a singular model from the server to the clients, conducting local training, then returning these models to the server for the aggregation of the client models. In the first case, the model sequentially moves between clients, updating

incrementally with each client's data, fostering continuous learning but potentially leading to latency due to sequential processing. The second approach, in contrast, allows simultaneous local training across multiple clients, enhancing efficiency and speed. After training, the server aggregates these local updates into a global model, effectively combining diverse data insights. In the second federated training technique using the **FedAVG** [20] algorithm: the process begins with the server initialization with a global random model and distributing it to all participating clients. Each client then trains the model locally with their unique dataset. After local training (SGD, Adam, etc on 1 or more steps),

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figure 2.7: Algorithm of FedAVG (Image taken from [20])

clients update their models with the new insights gained. These updated models, containing the learned parameters, are sent back to the server. The server's role is to aggregate these updates to refine the global model. In FedAVG, this aggregation is typically achieved by computing the weighted average of the received model updates. The weight is often determined by the volume of data each client contributes, allowing for a proportional influence on the updated global model. This cycle of distributing, training, and aggregating continues

for multiple rounds, with each iteration improving the global model's accuracy and robustness. This method efficiently harnesses insights from varied data sources while preserving the privacy of each client's data.

Subsequently, it's essential to understand which architectures are used in Federated Learning. One of the most commonly used architectures in this context is the **Federated Classifier**, a deep learning network designed for local training and subsequent aggregation of model weights across multiple clients: this clients that participate in the active training are also called **activation clients**. A distinctive feature of this architecture, in addition to adhering to the fundamental principles of federated learning, is its ability to achieve good performance for clients who don't have labels, known as **unseen clients**. Furthermore, an additional approach to enhance local performance (as the model aims to adapt to all clients, local performance may be lower) is the use of fine-tuning [21]. This involves freezing all layers except the last one, which is related to classification. In this context, training the final layer, even with few labels, can improve local performance. However, the Federated Classifier faces certain limitations: often, the input distributions from different users are highly heterogeneous, which can lead to lower global performance and additionally, this model falls under the category of discriminative AI, lacking the advantages inherent to generative and probabilistic models.

Finally, for simulating Federated Learning environments, a widely used framework is **Flower** [22, 23], a Python-based framework that facilitates the construction of such systems. It is compatible with frameworks like PyTorch or TensorFlow. Flower allows for the coding of server and client components, where each client can load its dataset. The framework supports various strategies, such as the number of clients, the fit_fraction (i.e., how many clients are actually used for training), the number of training rounds, and the option to use FedAVG. However, it should be noted that this system intensively utilizes hardware resources due to the need to load (and retain) all the weights of all clients and their corresponding datasets.

2.3 Meta Learning

In deep learning, each model is typically trained on a specific task. Often, a model trained for task A may not adapt well to a similar task B, necessitating retraining on task B. However, this approach differs from human learning, where the goal is to generalize learning across multiple concepts. This is why Meta Learning [24, 25] was introduced. The aim is to create versatile AI capable of performing across various tasks using only a few samples for adaptation, known as few-shot learning (usually 1 to 10 shots, or simply sample, per class). This concept is illustrated in Figure 2.8. In this approach, the

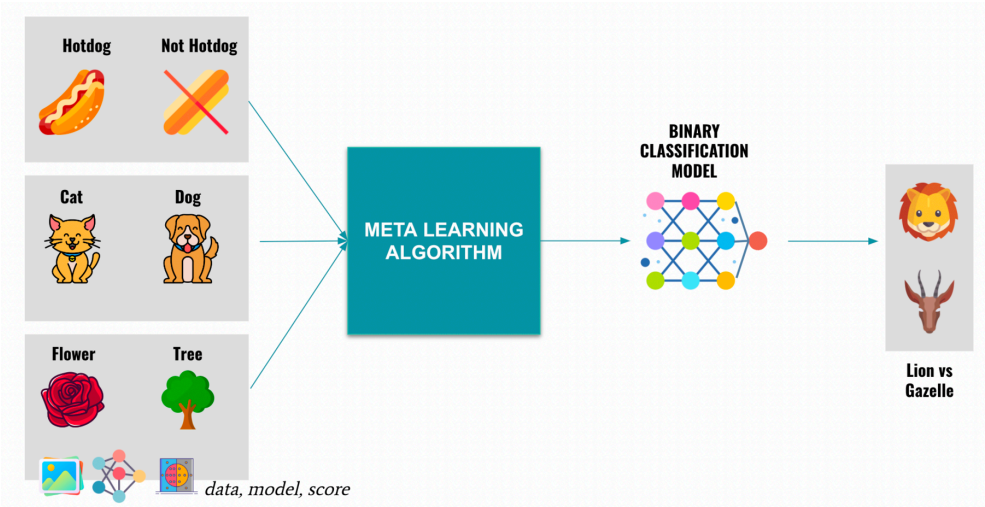

Figure 2.8: Task of learning the binary classification between multiple pairs instead of learning cat vs dog (Image taken from [26])

model is trained on a support set, which consists of samples from a dataset $\mathcal{D}$ with specific classes, and is then tested on a query set, composed of samples from different classes also taken from dataset $\mathcal{D}$. Commonly the used algorithms in this context are optimization-based, where they rely on the backpropagation of gradients. One of the most notable is MAML (Model-Agnostic Meta-Learning) [27], an algorithm that can be applied to any model that uses backpropagation. The goal of MAML is to find an optimal initial point for the model, such that it minimizes the distance between this initialization point

and the point of convergence for each task, as simply described in Figure 2.9.

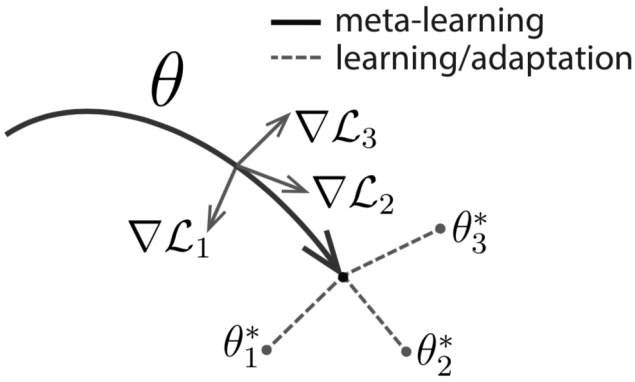

Figure 2.9: MAML algorithm, find the optimal initialization θ such that, with few-shot learning, it is possible to achieve the task specific optimal model θ_1^*, θ_2^* or θ_3^* (Image taken from [27])

Considering a model denoted as f_θ with parameters θ, for a given task $\mathcal{T}_i$ and its associated dataset ($\mathcal{D}_{train_i}$, $\mathcal{D}_{test_i}$) and the associated loss function $\mathcal{L}_{\mathcal{T}_i}$ computed on f_θ, the model parameters can be updated through one or more steps of gradient descent. These steps constitute what is referred to as the **inner loop**. The update process can be formalized as follows:

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{2.15}$$

Subsequently, in the **outer loop**, the initial parameters θ are updated in a manner that enhances performance across all tasks. This necessitates the computation of the gradient of the loss function with respect to the initial parameters:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{2.16}$$

In this step, the gradient (∇_θ) of the sum of the gradients (included in $f_{\theta_i'}$) is computed, involving the calculation of the second-order derivative.

2.3.1 Reptile Algorithm (Batched Version)

Reptile [28] is an algorithm proposed by OpenAI as an improvement to MAML. In MAML, the computation of the second derivative involves a significant computational cost; therefore, the goal was to find an algorithm that would stop at the first derivative, as proposed by Reptile. It samples a number N of tasks and subsequently computes SGD for each task, then the model is updated following the direction (gradient) common to all tasks. The analogy is to compute the gradient of gradients without actually calculating the second derivative. The algorithm that will be used in this dissertation is the batch version [29], defined as follows:

i. Randomly initialize the parameters of the model θ;

ii. Sample N tasks $\mathcal{T}_i$ from the dataset $\mathcal{D}$; in each task, the dataset is composed as $\mathcal{D}_i = (x_1, y_1), ..., (x_k, y_k)$ with k as number of samples;

iii. Inner loop: for each task, l steps of SGD are computed to obtain a new model weights θ_i (i.e. a classical training is computed for $\mathcal{T}_i$ using $\mathcal{D}_i$);

iv. Outer loop: all the previous θ_i were stored, now the weights of the model are computed using:

$$\theta \leftarrow \theta + \frac{\eta}{N} \sum_i (\theta_i - \theta) \tag{2.17}$$

with η as learning rate for the outer loop (called meta step) and N the number of tasks.

Chapter 3

System Descriptions

In this dissertation, an architecture, not yet explored in the context of Federated Learning, with two new variants will be proposed: the ClassInformed-VAE with the goal of bringing inputs from different distributions to a common distribution (i.e., the same latent space for different clients, simply termed by us as **Federated Latent Space**), and the use of Meta-Learning, specifically the new algorithm Reptile, for rapid adaptation (i.e., fine tuning using few-shot learning) of the CI-VAE to variations in client distributions.

3.1 Federated ClassInformed-VAE

Due to the limited number of samples available in common datasets, there is often a necessity to generate synthetic samples, which is not feasible with the current models in Federated Learning. Moreover, the standard Federated Classifier falls into the issue of the discriminative approach discussed in Chapter 2.1, where it does not provide a level of "certainty" in classification. Additionally, these models have a final Dense Layers that complete the classification based on the features received from the upper layers (i.e., CNN for images), that is focusing on classifying based on edges, angles, textures, and, at higher levels, parts of objects or more complex patterns in the case of images. However, the goal should be to learn the underlying distribution of the data in a continuous and regular latent space, aiming to capture not just the visible characteristics of the data but also their intrinsic variability and the probabilistic relationship between them. This concept can be addressed with the use of VAEs. Indeed, VAEs learn a probabilistic distribution in the latent space: this

implies that the model not only learns a compact representation (encoding) of the data but also how these data are distributed in the latent space. This can better capture the intrinsic variation in the data. Nevertheless, VAEs do not perform actual classification, as their architecture consists of an Encoder and Decoder, with the goal of reconstructing the data (both original and a modified version, as in denoising). For this reason, a classifier is placed near the latent space, taking as input a vector in the n-dimensional space of the latent space and providing as output the class to which the data belong: in this way, both discriminative and generative approach are used, as shown in Figure 3.1.

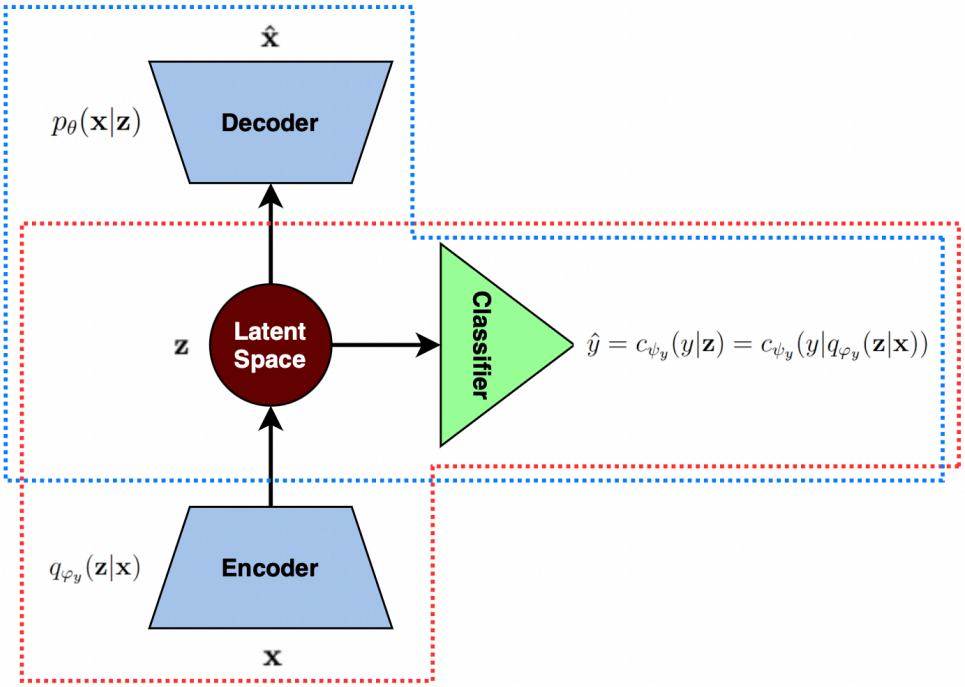

Figure 3.1: Diagram of the CI-VAE architecture. The section outlined by red dots delineates the classification part, which includes the Encoder, latent space $\mathbf{z}$ and sampling via the reparameterization trick, and the classifier. The portion outlined in blue illustrates the generative process, where a sample from the latent space is transformed into the reconstructed outputs $\hat{\mathbf{x}}$ and the synthetic $\hat{y}$

The decision to impose a classifier in the latent space rather than after the decoder is represented by the fact that in the decoder's output there is no control of the output: indeed, the classifier might learn the classification based

on what the decoder provides, perhaps even features different from those we would expect. This model of VAE with the latent space classifier is called ClassInformed-VAE (**CI-VAE**) [30]. It offers numerous advantages:

i. the classifier, **necessarily linear**, aids in the learning of features in the latent space; in fact, the information of the labels can be backpropagated into the Encoder, adjusting the weights of the latter using also the class information of the data (for example, a 4, a 9, or a 7 in MNIST dataset could have extremely similar features, causing the features to sometimes overlap with each other, as shown in Figure 3.2). Indeed, as demonstrated in [31, 32, 30], it is possible to achieve a latent space with greater differentiation of the various labels from each other. The previous consideration is explained by the following observation:

> "Once the linear discriminator is saturated and is unable to linearly separate the data in the latent space, through gradient back-propagation, it sends signals to the entire network to contribute to forming a latent space that is more linearly separable while maintaining low reconstruction quality" [30];

ii. during the generation process, the vector sampled in the latent space can also be passed through the classifier, having in output both the synthetic sample via the Decoder and the label via the classifier (blue part in Figure 3.1) using a **threshold** on the output probability of the classifier;

iii. in the case of simple classification, it is possible to use the Encoder with the classifier to find the belonging class (red part in Figure 3.1);

iv. the CI-VAE demonstrates promising results even with datasets containing a limited number of samples, such as in the case of digits drawn by a single user. As evidenced across various users, a more organized latent space can be achieved through the classifier, an outcome not feasible with standard VAEs when the number of samples is insufficient for an

optimal organization (as shown in Figure 3.3), leading to some overlap. **Therefore, CI-VAEs might be highly suitable for Federated Learning**. However, this phenomenon also strongly depends on the latent dimension size: with a larger latent size, even VAEs can better organize features. Nonetheless, an increasingly larger size may lead to the "holes" phenomenon in the latent space: during inference or the generation of new data, the model might behave unpredictably or produce unrealistic samples when exploring these "empty" or "hole" regions of the latent space. Obviously, with a small dataset, a small latent dimension means also worse reconstructed and synthetic images. Thus, the choice of the latent dimension value becomes extremely important.

For these reasons, CI-VAE could be considered a suitable solution in Federated Learning, even though it has not yet been evaluated in this context.

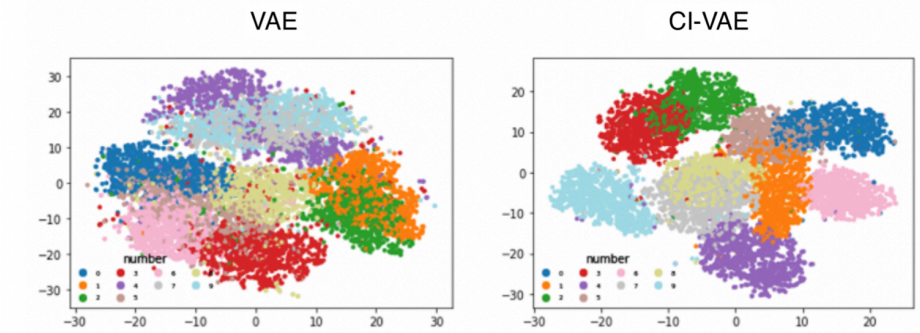

Figure 3.2: Comparative visualization of latent spaces. The figure presents a 2D t-SNE projection of the 20-dimensional latent spaces for the MNIST dataset, as modeled by a standard VAE (left) and CI-VAE (right). The CI-VAE latent space displays a more organized and distinct clustering of digits with smaller overlap, indicating enhanced class separation compared to the standard VAE (Image taken from [30])

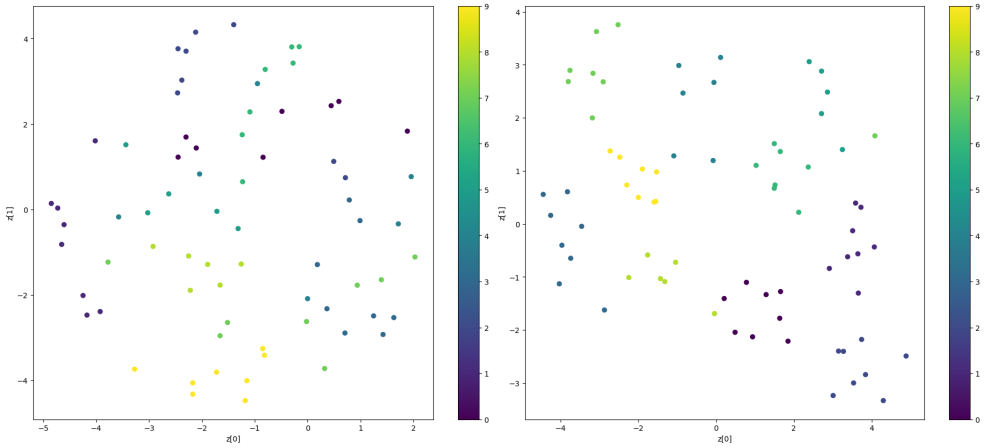

Figure 3.3: Latent space for the standard VAE (left) and CI-VAE (right) representing digits drawn by the same individual with a limited number of samples and using a 2D t-SNE projection of the 3-dimensional latent spaces. Notably, the CI-VAE benefits from the integration of a classifier, which aids in discerning distinct features and distributions, resulting in clear and discernible clusters. In contrast, the standard VAE, with its sparse sampling, fails to adequately regularize the latent space

3.1.1 Linear Classifier in CI-VAE

The classifier, applied to the latent space, can be described by $c_{\psi_y}(y|\mathbf{z})$ parameterized by ψ with weights dependent on the label value y during backpropagation, and using the latent representations z to predict y, as illustrated in Figure 3.1. However, within the CI-VAE's Encoder, z is computed as $q_{\varphi_y}(\mathbf{z}|\mathbf{x})$, thus the complete relationship of the classifier is:

$$c = c_{\psi_y}(y|q_{\varphi_y}(\mathbf{z}|\mathbf{x})) \tag{3.1}$$

In practice, this classifier consists of a single layer of linear Dense with an activation function of either softmax or sigmoid depending on the case. This also enables the backpropagation of the classifier's information into the Encoder, using the cross-entropy loss once the classifier weights have saturated; hence, the final loss for the CI-VAE will be:

$$\mathcal{L}_{total} = \mathcal{L}_{VAE} + \alpha \cdot \mathcal{L}_{classifier} \tag{3.2}$$

where $\mathcal{L}_{VAE}$ could simply be the loss for Equation 2.14 or the β-VAE loss, while $\mathcal{L}_{classifier}$ could be simply equal to $-\sum y \log c_{\psi_y}(y|\mathbf{z})$. α, considered as a hyperparameter, is the weight size for the classifier loss (privilege reconstruction and KL or the classification) and could be any value ≥ 0. Moreover, it is also important to introduce a regularization factor (L1, also called Lasso [33]) both in the classifier and in the VAE, as demonstrated in [30]. Furthermore, regularization will also be a very determining factor of the CI-VAE in Federated Learning. Indeed, L1 regularization pushes the majority of weights towards zero, aiding the model in preventing overfitting and making it more generalizable to new data, as well as identifying which inputs are relevant for a particular task, an aspect that is extremely crucial in Meta-Learning. The L1 can be simply described as:

$$\mathcal{L}_{new} = \mathcal{L} + \lambda \sum |\omega| \tag{3.3}$$

with ω being the model weights. An additional regularizer available in Keras is L2 (Ridge), which penalizes larger weights (using ω^2) by pushing them towards a distribution of smaller and more uniform weights. Through certain analyses, L1 has been found to yield better performance for the CI-VAE.

3.1.2 CI-VAE in FL

As discussed in the preceding section, CI-VAEs could offer several benefits in the Federated Learning scenario. Firstly, it is essential to demonstrate the standard federated VAE is capable of operating within an FL environment, as evidenced in [34]. This reference illustrates that an Fed-VAE can achieve similar generation performance with respect to the local VAEs.

Having established this, the next step is to verify the correct functioning of the classifier applied to the latent space. The objective of the CI-VAE is to align the same information, which may be expressed with different distributions or rotations, into a single distribution (i.e., two digit ‘4’s from different

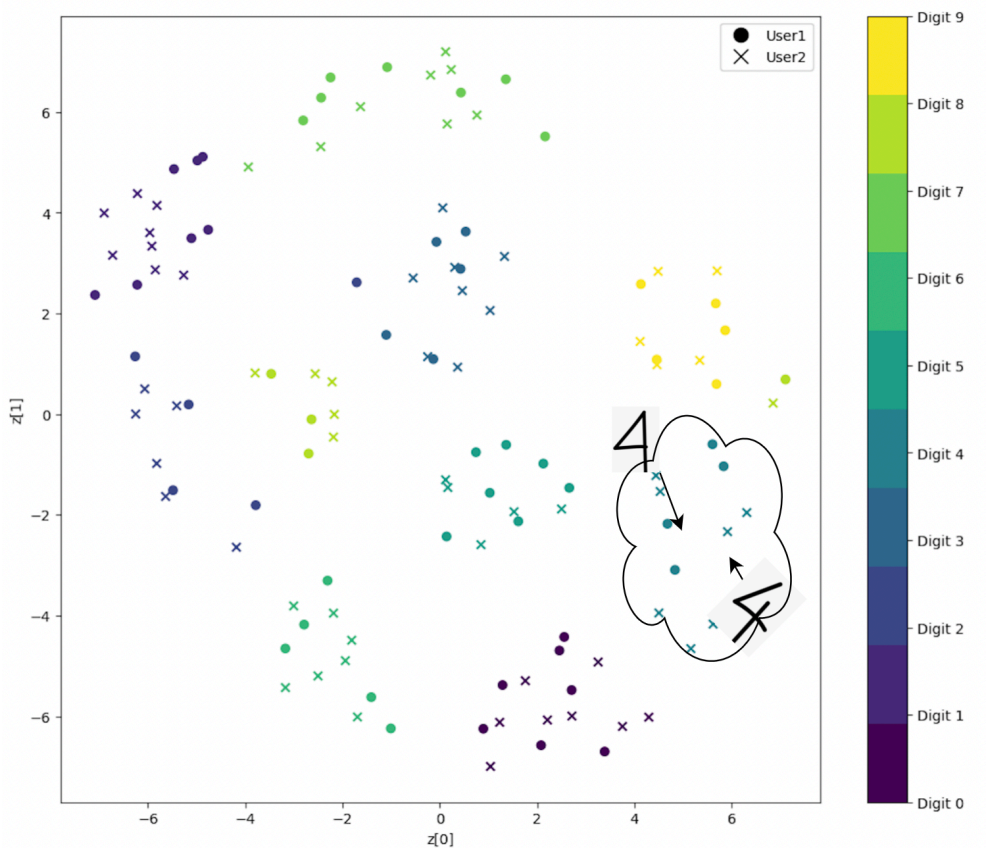

Figure 3.4: Visualization of the 2D projection with t-SNE of the shared latent space enabled by the classifier in FL for MNIST images for User1 and User2, demonstrating how different shapes and rotations for various digits lead images with the same label to the same distribution, as shown by the two digit '4's of different shapes and rotations brought into the same area of the latent space

individuals with varying rotations and shapes should be represented similarly in the latent space). With standard local VAEs, this is unachievable due to the fact that each client creates a custom latent space, precluding interoperability among different clients. Conversely, an Fed-VAE might struggle to represent the same information in the same region in the latent space, what one client perceives as the digit '9' could be interpreted as a '6' by another, simply by considering how the rotation of a camera device is kept. This is precisely where the classifier plays a crucial role: by using a shared classifier in FL and a local VAE (i.e., the VAE is updated locally without receiving weights from

the server), it is feasible to describe different distributions using the same latent variables z, as depicted in Figure 3.4, with different Encoders. Here, two clients with different rotations and features were trained in a FL environment with local VAEs and a shared classifier, proving the previous assertions. However, a significant challenge for this architecture is finding a balance between the size of the latent space and the decoder. Specifically, a very small latent dimension risks not adequately separating the different features for various clients, while an excessively large latent space could preclude the generation of samples due to the presence of "hole" in the latent space for smaller datasets.

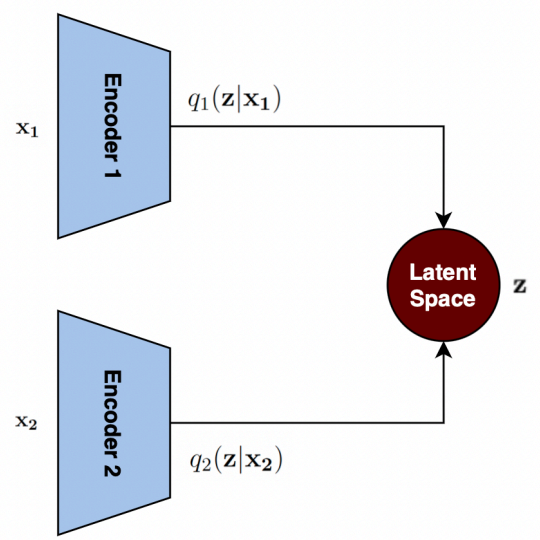

Figure 3.5: Representation of an ME-VAE [18], a concept very similar to the CI-VAE applied to Federated Learning: different clients have different datasets, x_n, and distinct encoders, E_n, with a shared latent space achieved through the classifier

3.2 Federated Reptile CI-VAE

An alternative variant of the architecture proposed in this dissertation involves the application of Meta Learning, specifically employing the batched version of the Reptile algorithm, an algorithm that has not yet been evaluated in the

context of Federated Learning. This variant leverages the same CI-VAE architecture previously described, with the addition of federated training through meta learning. The Federated Reptile CI-VAE, or **FedRep CI-VAE** has the primary objective is to facilitate the adaptation to the diverse datasets of various clients with fine-tuning, or called few-shot learning in this case. This time, the CI-VAE model is trained entirely in a federated manner, not just the classifier. Naturally, this approach might lead to lower overall performance since there isn't a custom converge point for each client. However, through few-shot learning, which is conducted after a federated training, the model should be able to adapt to the specific client. Nonetheless, it is imperative to first examine the application of Reptile within Federated Learning, a concept that has not been explored. The preference for Reptile instead of MAML stems from its utilization of first-order derivatives, unlike MAML's approach, and the straightforward manner in which Reptile can be adapted to the FL.

Algorithm 1: FedRep Algorithm

Input: Number of clients N with $N \leq$ Total Clients, Clients $c_1, c_2, \ldots, c_n$, Meta Step η, Loss $\mathcal{L}_{\mathcal{T}_i}$, k steps of SGD

Output: Updated global parameters θ

Initialize global parameters θ
for *iteration* $= 1, 2, \ldots$ **do**
    Sample N clients $c_1, c_2, \ldots, c_n$ in the pool
    for $i = 1, 2, \ldots, n$ **in parallel do**
        $\theta_i = \text{SGD}(\mathcal{L}_{\mathcal{T}_i}, \theta, k)$ (Client side)
    end
    Update $\theta \leftarrow \theta + \frac{\eta}{N} \sum_{i=1}^{N} (\theta_i - \theta)$ (Server side)
end

From Chapter 2.3.1, it is apparent that batched version of Reptile is founded on the concept of sampling various tasks with distinct classes from the same dataset. However, it is also intuitive to understand how a **task can be equated to the dataset of a specific client in Federated Learning with its unique**

features, labels, and distributions (Algorithm 1). In essence, it can be simply stated that each set of clients conducts local training with their respective datasets (i.e., client i, seen as task $\mathcal{T}_i$ with its own dataset $\mathcal{D}_i$), implementing several steps of SGD and then all models are sent back to the Server where their aggregation will take place (no longer using FedAVG). This process involves calculating the difference between the weights of the client i's trained model and the model at the initial round, and then summing the differences across all clients. With these considerations in mind, the final step of the algorithm will be (using 2.17 as reference):

$$\theta \leftarrow \theta + \frac{\eta}{N}\sum_{i=1}^{N}(\theta_i - \theta) \tag{3.4}$$

where θ_i represents the trained model with SGD for client i. Subsequently, using the dissociative property, the equation can be written as:

$$\theta \leftarrow \theta + \frac{\eta}{N}\sum_{i=1}^{N}\theta_i - \frac{\eta}{N}\sum_{i=1}^{N}\theta \tag{3.5}$$

$$\theta \leftarrow \theta + \eta\frac{\sum_{i=1}^{N}\theta_i}{N} - \eta\frac{\sum_{i=1}^{N}\theta}{N} \tag{3.6}$$

$$\theta \leftarrow \theta + \eta \cdot \theta_{avg} - \eta\frac{N \cdot \theta}{N} \tag{3.7}$$

$$\theta \leftarrow \theta + \eta(\theta_{avg} - \theta) \tag{3.8}$$

where θ_{avg} can be simply expressed as the average of the model parameters from the clients (equivalent to an unweighted FedAVG). The fundamental difference from FedAVG (remembering it is equal to $\theta \leftarrow \sum_i \frac{n_i}{N}\theta_i$ with n_i being the number of samples per client) is that FedRep updates the global parameters by moving them towards the aggregated parameters, representing an average of the parameters obtained from various clients. Ultimately, this variant offers numerous advantages due to its ability to quickly adapt to a client (i.e., task) with few-shot learning. Indeed, one can simulate a scenario where the

model is provided to a client who does not have labels, and if the classification is uncertain (i.e., the probability of the classifier's outcome is below a certain threshold), the system asks the client for label(s) and then performs fine-tuning.

Additionally, it is evident from the final formula that setting $\eta = 1$ directly yields θ_{avg}, aligning with the formula of unweighted FedAVG. However, this formula offers the advantage of transitioning from the initial point towards the midpoint in steps defined by η. In subsequent rounds, a new model θ'_{avg} will emerge, differing from the previous θ_{avg} since the starting points vary with each round. Furthermore, the Reptile algorithm entails sampling different tasks across various rounds. This concept is readily applicable to a real-world Federated Learning scenario involving thousands of devices. In this network, each round samples N clients from this pool, ensuring a consistently diverse dataset in each round (e.g., with 1000 devices and $N = 25$, the likelihood of sampling the same client in consecutive rounds is low). This approach effectively facilitates the assumption of continually changing tasks.

Chapter 4

Datasets

4.1 FEMNIST

The FEMNIST dataset [35] is a dataset used for image classification. Each image is represented as a digit, a lowercase, or an uppercase character. Indeed, the digit images can be likened to those in the MNIST dataset, with the significant difference that the images are grouped by writer. This aspect is particularly important for simulating federated learning, as it allows for the division of the dataset into several datasets that describe the images of a single writer. This dataset includes a considerable number of users ($3,550$ writers), totaling $805,263$ images. The average number of images per user is 226.83 (considering digits and letters) with a standard deviation of 88.94. Figure 4.1 provides an histogram representing the number of samples per user. For this

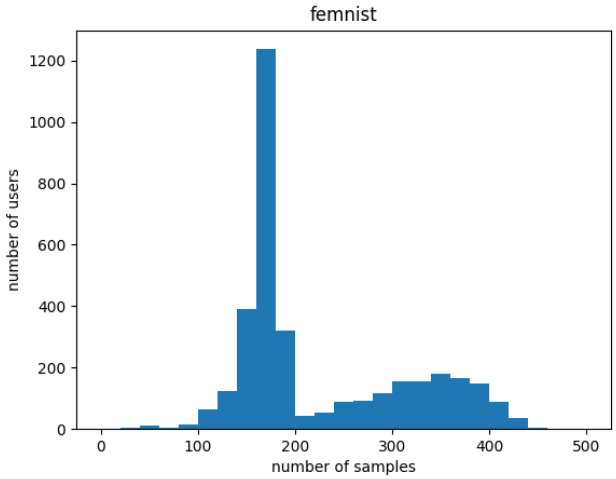

Figure 4.1: Number of images per user for FEMNIST dataset (Image taken from [35])

dataset, no preprocessing phase was applied, and the images have a resolution of $28x28$ pixels. However, the dataset contains images of varying quality and distribution, including some writers who have few or no labels for a particular class or features that are sometimes difficult to interpret. Of course, all these issues are part of the assumptions of Federated Learning, where each user often presents images of varying quality and dissimilar distributions across different users. This is precisely why the dataset was considered as is. Besides, for this dissertation, only images depicting digits were considered due to hardware limitations (i.e., to simulate a Federated Learning environment, each user must upload their own dataset and model, potentially leading to the simultaneous execution of $25 - 50$ models. Therefore, the number of labels chosen was 10 instead of 52, as would be the case with lowercase and uppercase letters).

Figure 4.2: Examples of digit images from the FEMNIST dataset illustrate how the digits exhibit various features or quality differences between different writers

4.2 HAR

Human Activity Recognition (HAR) [36] has emerged as a vital research area in the field of wearable technology and the Internet of Things (IoT). The ability to accurately recognize and classify human activities has numerous applications, including healthcare monitoring, fitness tracking, gesture control, and context-aware systems. It is extensively used in the realms of TinyML and Federated Learning.

The dataset consists of $10,299$ samples from 30 users aged between 19 and 48, divided into 21 for training and 9 for testing, with 561 attributes and across 6 activity classes: walking, walking upstairs, walking downstairs, sitting, standing, and laying. Data have been collected using an accelerometer (on X, Y, and Z axes) and a gyroscope using a Samsung SII smartphone, capturing 3-axial linear acceleration and 3-axial angular velocity at a constant rate of $50Hz$. The data was then re-processed by applying noise filters. The distribution between the training and test sets for each user is $70 - 30\%$ and the recorded values, ranging between -1 and 1, were normalized to a $0 - 1$ range. The class distribution percentage for each user is almost equal, and each client does not have missing values or users with a limited set of labels; these characteristics make it a dataset that is easy to evaluate.

However, not all of the 561 features in the HAR dataset were utilized. An increase in performance was observed through a feature selection analysis, reducing the original 561 features to 144 (which were treated as a $12x12x1$ image). The variance method was used for this purpose [37], selecting the top 144 features that exhibited the greatest compromise between variance across different classes (i.e., if an attribute has the same value for different classes, it might be considered insignificant for the classification) and the number of parameters. Indeed, some features were found to potentially worsen overall performance. A more comprehensive analysis revealed that the most important

Table 4.1: Micro F1 score among different input size

Input Size	N. Params	micro F1
50	1665	0.899
60	2391	0.918
70	3205	0.926
80	4186	0.937
90	5245	0.936
100	6481	0.944
110	7785	0.937
120	9276	0.954
130	10825	0.969
140	12571	0.972
150	14365	0.974
160	16366	0.978
170	18405	0.975

features were those related to entropy and the mean/max accelerometer readings on the X-axis. This variance method was tested for inputs ranging from 50 to 170 features, beyond which there was a decrease in performance 4.1.

Chapter 5

Results

In this chapter, the results for the two datasets introduced in the previous chapter will be explained. Each dataset will be evaluated across four different models: a locally trained classifier (Local CL), a Federated Classifier (Fed CL), and finally the new models proposed in this dissertation, Fed CI-VAE and FedRep CI-VAE. For each dataset, results will be presented for the **active clients** (those who have actively participated in the federated training) and the **unseen clients** (those who have not participated in the federated training and have few or no labels). Besides, in federated learning, the use of **batch normalization** and **dropout** can present challenges due to the non-i.i.d. nature of the data and the distributed training setting; thus, they have not been used in this dissertation.

5.1 FEMNIST (with Rotation)

For this dataset, a maximum of 25 clients were evaluated for active training due to hardware limitations. The same seed was used in the sampling of clients for each model to ensure more realistic comparisons. Moreover, in the following sections, results will be presented only for seed 42, as the same considerations and issues have been observed for other seeds used. In itself, this dataset is quite simple, not showing any real differences in performance among different users. For this reason, the dataset was further complicated by adding **rotation**: each client presents images with their own rotation, which was sampled with values in the range of $0-359$ degrees. This added complexity serves to simulate more realistic datasets where the input distributions are much more

complex (e.g. rotation of a smartphone camera can change a lots while taking the photo of the digit samples in this case) than the classic MNIST dataset, remembering always that IoT devices can't send data due to privacy.

For this dataset (and generally for the subsequent ones), four models were evaluated:

i. **Local CL**: this model served as a benchmark for the performance of each client. In each iteration, the client loaded its dataset, performed training, and then tested the model on the test set. The used architecture is depicted in Figure 5.1. The hyperparameters were set to a learning rate of 0.001 with Adam and a batch size of 4, due to the low number of images per user (ranging from 20 to 130).

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 28, 28, 1)]	0
conv1 (Conv2D)	(None, 24, 24, 16)	416
pool1 (MaxPooling2D)	(None, 12, 12, 16)	0
conv2 (Conv2D)	(None, 10, 10, 32)	4640
maxpool2 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense1 (Dense)	(None, 100)	80100
dense2 (Dense)	(None, 10)	1010

Total params: 86,166
Trainable params: 86,166
Non-trainable params: 0

Figure 5.1: The Local Classifier model architecture for the FEMNIST dataset

ii. **Fed CL**: the previous architecture (Local CL) was then evaluated in a federated manner using FedAVG as the aggregation function on the Server. Subsequently, Fine Tuning was performed, leaving the parameters of the last Dense Layer trainable, using 2 samples per class (2-shots learning), as proposed in 2.2. The same hyperparameters as Local CL were used;

iii. **Fed CI-VAE**: Subsequently, the first variant of VAE for FL was tested, with the architecture proposed in Figure 5.2 and Figure 5.3, and a classifier consisting of a single Dense Layer with the Latent Space dimension as input and the number of classes as output. The model was trained with local Encoders and Decoders and a federated Classifier, with FedAVG as the aggregation function. Regarding hyperparameters, a latent dimension of 3 was used, learning rate of 0.001 with Adam, $\beta = 1$ for the KL loss coefficient, $\alpha = 3$ as a coefficient for the class loss, a batch size of 4 and L1 regularizer of 0.1;

iv. **FedRep CI-VAE**: Finally, the last model tested used Meta-Learning. The Encoder and Decoder (same architectures as before) in this case were trained in a federated manner, as well as the classifier, with the significant difference being that FedRep with $\eta = 0.35$ was used as the aggregation function. All hyperparameters were the same as Fed CI-VAE, with the difference of using SGD instead of Adam. Finally, 2-shots learning was tested. However, this model has some privacy limitations, as it's possible to generate a sample through the public Encoder or Decoder of client i that has a distribution similar to client i's input. Consequently, **this model requires a more thorough future analysis involving Differential Privacy**.

All the previous models were evaluated on the same 25 individuals, a limit reached due to hardware constraints, but different seeds were used for each evaluation. Regarding the number of epochs or rounds, a convergence point was reached after 100 epochs for all federated models, always considering 25 individuals (however, the number of rounds could vary significantly with an increase in clients). Fed CL experienced slight overfitting from 60 rounds onwards 5.4, while both Fed CI-VAE and FedRep CI-VAE showed slight overfit after 80 rounds 5.5. Clearly, for these last two models, a greater number of epochs is necessary, both for the stability of the VAE and for the classifier.

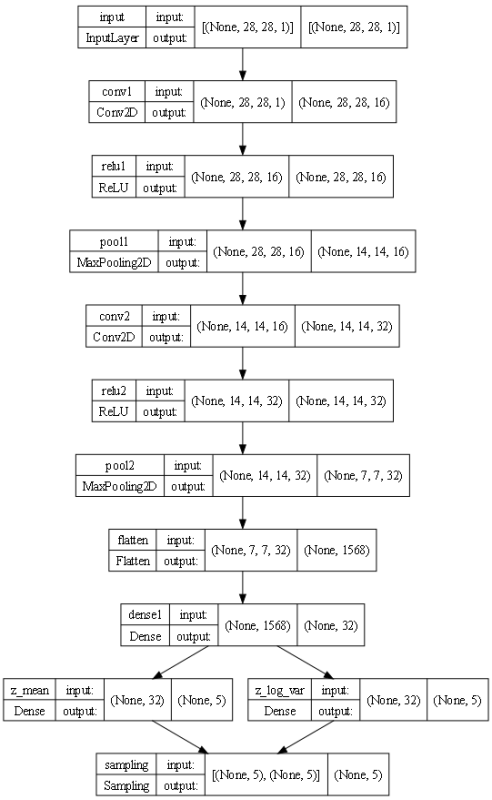

Figure 5.2: Architecture of the Encoder for CI-VAE used in FEMNIST

Besides, Local and Fed CL share the same architecture, totaling $180k$ parameters, while FedRep and Fed CI-VAE have a total of $229k$ parameters, encompassing the Encoder, Decoder, and Classifier.

The initial performance (presented in Table 5.1) were conducted on the same 25 clients, subsequently reporting the micro F1 score achieved on the test set of the active clients. After that, the models were evaluated on the unseen clients, recalling that these are the clients that do not have labels. As

Table 5.1: Micro F1 score on FEMNIST with Rotation and w/out fine tuning

Model	Test Set Active Client	Unseen Client
Local CL	**0.91**	-
Fed CL	0.83	**0.75**
Fed CI-VAE	0.87	0.70
FedRep CI-VAE	0.84	0.74

revealed by the table 5.1, local models achieved better performance for the

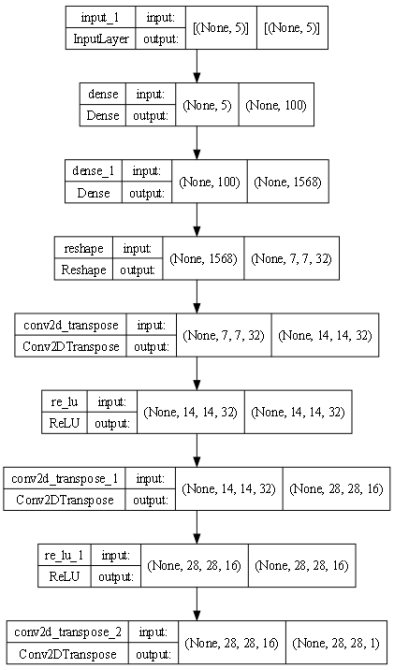

Figure 5.3: Architecture of the Decoder for CI-VAE used in FEMNIST

FEMNIST dataset with rotation, mainly because each client presents its own rotation and distribution. Thus, both the local classifier and the local CI-VAE were able to adapt to the needs of individual clients, with the CI-VAE offering the possibility of a federated and common latent space. However, these models encountered low performance with unseen clients: the Local CL could not adapt as there is no real common model, raising the issue of which model to choose from the client pool for testing the unseen clients. Regarding CI-VAE, the final model chosen took the average of all the Encoder parameters, using the federated classifier and a Decoder from the client who volunteered as a sample participant; in this case, the performance was not null but indeed very low for unseen clients (without accounting for the fact that the generated samples correspond to the data of the user who served as a volunteer), indicating struggles in situations with complex input distributions and without labels. Fed CL, however, showed comparable results between active and unseen clients, though it was the model with the lowest micro F1 in active clients (0.83), and the best in unseen clients (0.75). FedRep CI-VAE showed good

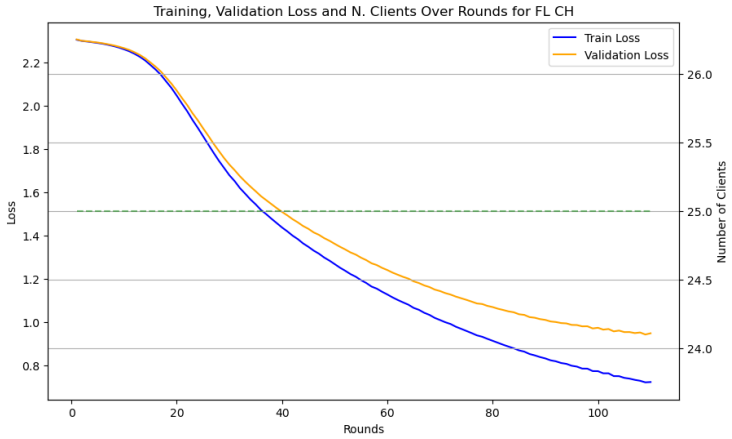

Figure 5.4: Mean Train and Val loss for Fed CL model received from the 25 clients

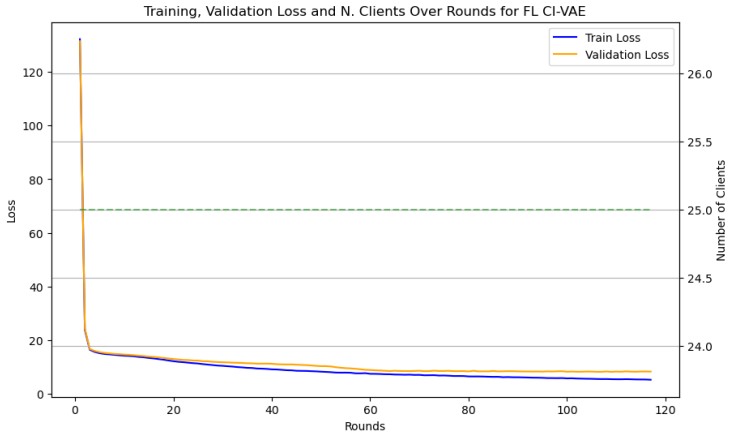

Figure 5.5: Mean Train and Val loss for Fed CI-VAE model received from the 25 clients

results for unseen clients (0.74), but it's noteworthy that this model was developed to achieve good performance through the use of Meta-Learning. For this reason, the second phase of performance evaluation involved the use of 2-shots learning for unseen clients (Table 5.2): each unseen client provided 2 random samples per class, and the remaining samples were used to test the performance. Moreover, fine tuning can also be applied to federated models for active clients: FedRep CI-VAE and Fed CL feature completely federated models, hence performance on individual clients might be lower. Therefore, fine tuning can be applied to active clients, using the train set or 2-shots learning.

It's essential to note that for FedRep CI-VAE tuning can be done by freezing the classifier. This ensures that the samples are always classified in the same area of the latent space, while enhancing the performance of the Encoder and Decoder.

Table 5.2: Micro F1 score on FEMNIST with Rotation and with fine tuning (with * no fine tuning applied)

Model	Test Set Active Client	Unseen Client
Local CL	**0.91***	0.33
Fed CL	0.86	0.86
Fed CI-VAE	0.87*	0.75
FedRep CI-VAE	0.88	**0.88**

As observed from Table 5.2, fine tuning was not performed for the Local CL and Fed CI-VAE models for active clients as they have local models trained on their datasets. Both, however, did not exhibit excellent performance on unseen clients, where Local CL was trained using only 2 samples per class. Conversely, the federated models achieved excellent performance, showing high adaptability with the use of 2-shots learning; notably, FedRep CI-VAE attained the best performance on unseen clients and showed a good margin of improvement for active clients, thanks to the concept of Meta-Learning. Hence, with 2-shots per class, it's possible to achieve about 90% of micro F1 for the FEMNIST case with a different rotation for each client, also demonstrating excellent results on the generative side, thanks to the possibility of fine-tuning the VAE while keeping the latent space unchanged. Moreover, the use of fine tuning yielded excellent results even in cases where users had only a limited set of labels, demonstrating a strong capacity for adaptation for labels not presented during the few-shot learning. This adaptability extends to situations where a user's data distribution changes over time: if the output probability of the labels falls below a certain threshold, the system can request a new set of labels (2 samples per class or more). It can then retrain the

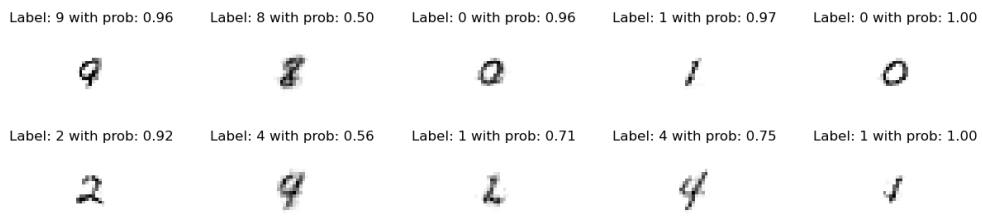

Figure 5.6: Random Samples Generated for FedRep CI-VAE with 2-shots learning using a Latent Space with 3 dimensions for Client f0068_00 (that have only 50 samples) and the output probability of the Synthetic Label generated

original model (i.e., before fine tuning) and perform a new round of fine tuning, thereby showing high adaptability. Finally, as one might easily deduce, a greater number of shots tends to yield more encouraging results. However, it's worth noting that it can sometimes be challenging to request more than 2 samples per class from a user. Lastly, it should be noted that the essential condition for Reptile tasks (i.e., the model must be trained on continuously varying tasks) was not entirely met due to the inability to load more than 25 clients. In a broader context, more users should be involved (1000 as in research papers) and a threshold T should be employed to identify the correct number of active clients (with $T \leq 1000$).

However, a more detailed analysis of individual clients is also necessary (Table 5.3). From the results obtained for each active client, it's observable that local models (Local CL and our, i.e., Fed CI-VAE) achieved higher performance because they are better at capturing the individual client's rotations. In contrast, federated models adapt to perform optimally across all the unseen clients, mainly because the aggregation function is based on averaging the weights of all the possible clients. However, as previously noted, local models struggle to adapt to unseen clients. Another critical observation is that the results are heavily influenced by the number of training samples available (clients f0005, f0012, f0033, and f0048) which exhibit lower performance for both local and federated models, indicating they possess extremely distinctive features. Despite this, one might expect improved performance from federated

Client	N. Samples	Angle	Local CL	F CL	our	FRep our
f0000_14	62	57	0.95	0.83	0.88	0.85
f0005_26	49	308	0.82	0.74	0.80	0.79
f0010_18	75	359	0.94	0.92	0.88	0.83
f0011_13	68	216	0.95	0.85	0.90	0.84
f0012_39	50	15	0.85	0.81	0.84	0.82
f0013_38	61	302	0.88	0.83	0.86	0.83
f0015_48	60	101	0.88	0.83	0.85	0.85
f0019_38	71	346	0.93	0.85	0.85	0.83
f0020_08	60	140	0.88	0.82	0.85	0.81
f0027_29	75	279	0.95	0.84	0.86	0.83
f0033_49	47	258	0.83	0.79	0.85	0.83
f0035_19	71	327	0.95	0.91	0.94	0.91
f0043_10	69	71	0.89	0.88	0.88	0.87
f0044_12	56	111	0.88	0.86	0.89	0.88
f0045_12	74	47	0.95	0.86	0.92	0.88
f0048_00	36	16	0.78	0.67	0.72	0.71
f0054_14	73	114	0.88	0.87	0.88	0.86
f0058_07	71	13	0.88	0.86	0.88	0.87
f0068_00	56	287	0.88	0.84	0.87	0.86
f0077_13	71	119	0.91	0.88	0.92	0.88
f0086_47	73	332	0.86	0.87	0.85	0.84
f0089_16	69	125	0.86	0.85	0.87	0.83
f0093_24	75	44	0.92	0.86	0.89	0.86
f0097_05	70	12	0.91	0.87	0.90	0.87
f0099_18	75	52	0.92	0.88	0.92	0.88
Total	1617	-	0.91	0.83	0.87	0.84

Table 5.3: Micro F1 for the four different models on active clients' test sets w/out fine tuning

models. However, even with the same rotations, other clients achieve better performance, highlighting a stark example where a local model with Transfer Learning could be a better solution in the case of few labels. Nonetheless, a more detailed discussion concerning rotations is warranted, as described in Section 5.1.1. During the 2-shots learning phase, there was an increase in client performance, as indicated in Table 5.2, although the improvement was almost consistent across all clients: those with initially low performance saw only a slight enhancement. For a more accurate analysis, it would be necessary to involve a larger number of active clients and to explore Transfer Learning

techniques from MNIST. Despite this, the CI-VAE has shown commendable results (comparable with Local and Fed CL) even with a limited number of samples available, thanks to the classifier applied to the latent space.

5.1.1 Rotation challange

Rotation represents a real challenge during federated training; in fact, from the results described in the previous sections, it is possible to notice how the i.i.d. problem (different rotations, shapes, and numbers of samples for each dataset) is partly solved. However, for a more complete analysis, it is necessary to view the performance of the models in a complex scenario, where there is a single client in the network with distributions completely different from the remaining clients. In FEMNIST dataset, the rotation will be evaluated, but the task can be extended to any dataset where the majority of the network is composed of clients with the same distributions. Rotation might pose a problem in cases where the majority of users in the network experience the same rotation (e.g., with 10 clients of which 9 have the same rotation, the tenth could have lower performance due to the fact that, in general, the Server with FedAVG simply averages the model weights). To this end, four scenarios will be tested on 25 clients with the models previously described, except for the local model for obvious considerations. In all the scenario Fine Tuning was used to restore the client-performance (Table 5.4). Clearly, for a more complete analysis, it would be necessary to test on a larger number of clients and with many more configurations, something not possible with the hardware resources used.

The first scenario (**Scenario 1**) describes a simulation in which 24 clients had a rotation between $0 - 90$ degrees, while a single client had a rotation of 250 degrees; this scenario was simulated by sampling different clients. However, it is important to note that it also depends heavily on the client selected for the 250 degrees rotation. In the case of a client with a dataset containing errors or difficult to interpret, the performance might not reflect those described in the table. In this scenario, all models demonstrated high adaptability. The second

Table 5.4: Models' adaptability to each scenario using mean micro F1 on the selected client(s); High > 0.85; $0.75 \leq$ Medium < 0.85; $0.65 \leq$ Low < 0.75; Very Low < 0.65

Scenario	Fed CL	Fed CI-VAE	FedRep CI-VAE
1	High	High	High
2	Low	**High**	Medium
3	Low	Very Low	**Medium**
4	Very Low	Very Low	**Low**

scenario (**Scenario 2**) represents a simulation where 24 clients have the same rotation (0 degrees) and one client has a different rotation (250 degrees). In this case, the Fed CI-VAE model showed high adaptability thanks to the presence of the local VAE, while the model with Reptile achieved higher performance compared to Fed CL. Finally, the last two scenarios describe a situation where unseen clients are evaluated. The first (**Scenario 3**) is a simulation where all active clients during training had rotations between $0 - 90$ degrees, while the unseen clients had a rotation of 250 degrees. The second case (**Scenario 4**) describes a simulation where all active clients had an identical rotation (0 degrees), and the unseen clients had a rotation of 250 degrees. In these last cases, it is evident that the Fed CI-VAE has low adaptability due to the presence of the local VAE, which is not able to adapt to unseen clients. As for the other two models, FedRep CI-VAE showed better adaptability compared to Fed CL. However, it can be concluded that the greater the variability of the clients (in this case, more rotations, i.e., photos taken with different angles), the better the performance of FedRep CI-VAE and to some extent also for Fed CL.

5.2 FEMNIST (w/out Rotation)

The models described in the previous section were also evaluated for the FEMNIST dataset without any image rotation. Understandably, using the same architectures and the same number of parameters, it was possible to achieve

higher performance compared to the scenario with image rotation, as can be easily inferred. Regarding the hyperparameters, all the previously described values from the preceding section were used due to hardware limitations. However, for a more thorough analysis, a complete re-tuning should be performed, possibly with the aid of a larger number of active clients, which has consistently been maintained at 25. Initially, the models were tested without the aid

Table 5.5: Micro F1 score on FEMNIST w/out rotation and w/out fine tuning

Model	Size	Test Set Active Client	Unseen Client
Local CL	180k	0.93	-
Fed CL	180k	0.95	**0.92**
Fed CI-VAE	229k	**0.96**	0.72
FedRep CI-VAE	229k	0.90	0.85

of fine tuning 5.5. In this case, high performance were achieved for the active clients with the Fed CI-VAE variant, while the local classifier exhibited lower performance (therefore, it can be concluded that the CI-VAE architecture and the federated latent space contributed to an increase in performance). Regarding the unseen clients, the only model that achieved acceptable results was Fed CL, even though the FedRep variant for CI-VAE showed a significant increase in performance compared to Fed CI-VAE. Subsequently, the four

Table 5.6: Micro F1 score on FEMNIST w/out rotation and with 2-shot learning (with * no fine tuning applied)

Model	Size	Test Set Active Client	Test Set Unseen Client
Local CL	180k	0.93*	-
Fed CL	180k	**0.97**	**0.95**
Fed CI-VAE	229k	0.96*	0.77
FedRep CI-VAE	229k	0.96	0.93

different models were evaluated using 2-shots learning 5.6. It is important to note that in the local models, fine tuning was not performed for the Test Set of the active clients. In this scenario, the best performance were achieved by

Fed CL, with a low adaptability of Fed CI-VAE (which improved from 0.72 to 0.77 with 2-shots learning, a minimal increase). This variation in the dataset demonstrates how federated knowledge can lead to better performance compared to local models: this can be attributed to the fact that some clients have features that are difficult to interpret locally, which can be leveraged through federated learning. However, it should be highlighted that FedRep CI-VAE achieved performance similar to Fed CL, with the added benefits of a federated latent space and the ability to generate samples. It should also be noted that, in this case too, a complete tuning of the hyperparameters was not performed due to hardware limitations.

5.3 HAR

In this section, the results achieved by four reference models (Local and Fed CL, Fed and FedRep CI-VAE) for the HAR dataset are presented. For this dataset, 21 clients were used as active clients (with 100% of the fit_fraction) and 9 clients as unseen clients. The primary goal for this dataset was to minimize the number of parameters while still achieving high performance. Additionally, the focus for this dataset was predominantly on classification, although the generative models also exhibited good performance, as demonstrated and shown in [1]. Different structures and architectures for Encoders and Decoders should be considered, especially given the limited memory available on IoT devices.

The models under consideration have architectures similar to those used in the FEMNIST case, with the variation of employing Dense layers. Both the local and federated classifiers feature a first Dense layer with 128 neurons and a second layer with 64 neurons. The CI-VAE, on the other hand, has a latent dimension of 3, with an Encoder composed of $64 - 32$ neurons for each layer, and a Decoder with $16 - 32 - 64$ neurons (thus an asymmetrical structure), which showed better benefits in sample generation. Lastly, there is a linear

classifier with a single layer. For all versions, a batch size of 8 was used, Adam as the optimizer with a learning rate of 0.001, a weight of 3 for the class loss, 0.1 as L1 regularizer, and finally, $\eta = 0.35$ for the FedRep approach.

Regarding performance, in this instance as well, the number of active clients used was quite low due to the small number of users in the dataset. This was the case despite the hardware allowing for a greater number of active clients for federated training, thanks to the small size of the models, as shown in Table 5.7.

Table 5.7: Sizes of the models proposed on HAR dataset

Model	N. Params
Fed CL and Local CL	$27.2k$
FedRep and Fed CI-VAE	$23.7k$

The advantage of architectures with CI-VAE is that they can achieve performance similar to other solutions while also having the ability to generate samples and distribute knowledge through the federated latent space. The results for the different models without and with the use of fine tuning (2-shots learning in this case) are shown in Tables 5.8 and 5.9. As can be inferred, for Fed CI-VAE and Local CL, fine tuning was not considered for the Test Set of Active Clients, since the training is already performed locally. From the over-

Table 5.8: Micro F1 score on HAR w/out fine tuning

Model	Test Set Active Client	Unseen Client
Local CL	**0.98**	-
Fed CL	0.92	**0.90**
Fed CI-VAE	0.97	0.75
FedRep CI-VAE	0.90	0.88

all results, it is observed that FedRep CI-VAE and Fed CL demonstrate good performance for unseen clients without fine tuning (without labels), achieving very high performance for local models, which, however, show very low

Table 5.9: Micro F1 score on HAR with 2-shots learning (with * no fine tuning applied)

Model	Test Set Active Client	Test Set Unseen Client
Local CL	**0.98***	0.23
Fed CL	0.96	**0.93**
Fed CI-VAE	0.97*	0.85
FedRep CI-VAE	0.92	0.92

adaptability to unseen clients. These, in turn, exhibited low adaptability even in the case of 2-shots learning, although Fed CI-VAE showed higher performance compared to the local classifier. Regarding the other models, Fed CL performed better than FedRep CI-VAE, although the latter offers the capability of a federated latent space and the possibility to generate samples.

Moreover, a more detailed analysis on individual clients revealed that some users have low performance [1], such as for user 14, with results shown in Table 5.10. From these performance results, it can be noted that higher micro

Table 5.10: Micro F1 score on Active Client 14 Test Set (with * no fine tuning applied)

Model	w/out fine tuning	After 2-shots
Local CL	0.88	0.88*
Fed CL	0.69	0.94
Fed CI-VAE	**0.95**	**0.95***
FedRep CI-VAE	0.72	0.90

F1 values were achieved with local models, likely because this particular user has input distributions (and in a minority compared to the network) that are very different from those of other users. In this case, fine tuning represents a very valid choice to address the issues of users who have distributions that differ significantly from all other users in the network.

In conclusion, for this dataset too, the performance should be repeated with a larger number of active clients to evaluate the limits and advantages of all the models, in addition to better parameter tuning for CI-VAE.

Conclusion

This dissertation explored the potential of VAE and Meta-Learning on Federated Learning scenarios. One of the primary challenges in Federated Learning, especially in IoT scenarios where user data distributions are highly heterogeneous and the number of samples in a dataset is often limited, is the lack of shared knowledge. This thesis revealed that the CI-VAE architecture is capable of not only establishing federated knowledge through a federated latent space, where diverse input distributions converge into a unified distribution, but also of generating synthetic samples and labels through its classifier. It achieves performance comparable to existing local and federated models in the literature while using the same number of parameters, applicable to both image datasets (FEMNIST) and sample-based datasets (HAR). Furthermore, Meta-Learning has proven to be an effective approach for enhancing performance with very few samples per class, even within a Federated Learning scenario. Additional advantages include the ability to adapt to changes in user input distributions or users with a limited set of labels. Potential future directions based on this work include evaluating CI-VAE and Meta-Learning with quantization to reduce model sizes for IoT systems, exploring the potential of Knowledge Distillation (employing the Server as teacher and clients as students), model pruning, differential privacy, and considering the application of Transfer Learning where feasible. However, it's important to highlight that these results should be re-evaluated in a broader context, with a larger number of clients, additional datasets, and comprehensive hyperparameter tuning, which was not possible due to hardware limitations.

List of Figures

2.1 Architecture of the AutoEncoder for the digit 6, using the MNIST dataset (Image taken from [2]) 5

2.2 a) Description of two distinct data labels in a 2D space where X identifies the current input; b) discriminative approach, the probability of X belonging to the blue group is high, thus, according to the decision making, it is a definite and secure decision; c) generative approach, the point X is located in the blue zone, but far from the region of blue points, hence it is an uncertain decision (Image taken from [4]) 6

2.3 A VAE learns the map between an observed $\mathbf{x}$-space, with distribution $q_D(\mathbf{x})$, and a latent $\mathbf{z}$-space; the generative part learns $p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, while the encoder part, also called inference model, approximates the true but intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ using $q_\varphi(\mathbf{z}|\mathbf{x})$ (Image taken from [6]) 10

2.4 Graphical comparison of the traditional sampling in VAE (left) versus the reparameterization trick (right). On the left, $\mathbf{z}$ is directly sampled from the approximate posterior $q(\mathbf{z}|\mathbf{x})$, which is non-differentiable. On the right, the reparameterization trick is used to sample $\mathbf{z}$ by adding a deterministic transformation of two separate nodes: μ and σ, representing the mean and standard deviation of the approximate posterior, and ϵ, a random node sampled from a standard normal distribution $\mathcal{N}(0, 1)$. This allows the backpropagation of the gradient (Image taken from [13]) 11

2.5 An overview of the VAE architecture, from the initial input $\mathbf{x}$ to the reconstructed, **lossy**, $\hat{\mathbf{x}}$. It incorporates the Encoder, which generates $\Sigma_{\mathbf{z}|\mathbf{x}}$ and $\mu_{\mathbf{z}|\mathbf{x}}$ based on the input or a set of inputs, followed by the application of the reparameterization trick for sampling. Then, the Decoder reconstructs the original input(s) (Image taken from [14]) 11

2.6 AutoEncoder (left) vs VAE (right). AutoEncoders learn the map between input and $\mathbf{z}$ without explicit consideration for the semantic distribution of the encoded features (a randomly sampled point may not correspond to a meaningful feature set, as shown in black), while VAEs encodes input data into a structured latent space in a probabilistic manner, allowing for the generation of new samples with different features from different clusters (Image taken from [15]) 12

2.7 Algorithm of FedAVG (Image taken from [20]) 14

2.8 Task of learning the binary classification between multiple pairs instead of learning cat vs dog (Image taken from [26]) . . 16

2.9 MAML algorithm, find the optimal initialization θ such that, with few-shot learning, it is possible to achieve the task specific optimal model θ_1^*, θ_2^* or θ_3^* (Image taken from [27]) . . . 17

3.1 Diagram of the CI-VAE architecture. The section outlined by red dots delineates the classification part, which includes the Encoder, latent space $\mathbf{z}$ and sampling via the reparameterization trick, and the classifier. The portion outlined in blue illustrates the generative process, where a sample from the latent space is transformed into the reconstructed outputs $\hat{\mathbf{x}}$ and the synthetic $\hat{y}$ 20

3.2 Comparative visualization of latent spaces. The figure presents a 2D t-SNE projection of the 20-dimensional latent spaces for the MNIST dataset, as modeled by a standard VAE (left) and CI-VAE (right). The CI-VAE latent space displays a more organized and distinct clustering of digits with smaller overlap, indicating enhanced class separation compared to the standard VAE (Image taken from [30]) 22

3.3 Latent space for the standard VAE (left) and CI-VAE (right) representing digits drawn by the same individual with a limited number of samples and using a 2D t-SNE projection of the 3-dimensional latent spaces. Notably, the CI-VAE benefits from the integration of a classifier, which aids in discerning distinct features and distributions, resulting in clear and discernible clusters. In contrast, the standard VAE, with its sparse sampling, fails to adequately regularize the latent space 23

3.4 Visualization of the 2D projection with t-SNE of the shared latent space enabled by the classifier in FL for MNIST images for User1 and User2, demonstrating how different shapes and rotations for various digits lead images with the same label to the same distribution, as shown by the two digit '4's of different shapes and rotations brought into the same area of the latent space 25

3.5 Representation of an ME-VAE [18], a concept very similar to the CI-VAE applied to Federated Learning: different clients have different datasets, x_n, and distinct encoders, E_n, with a shared latent space achieved through the classifier 26

4.1 Number of images per user for FEMNIST dataset (Image taken from [35]) 30

4.2 Examples of digit images from the FEMNIST dataset illustrate how the digits exhibit various features or quality differences between different writers 31

5.1 The Local Classifier model architecture for the FEMNIST dataset 35
5.2 Architecture of the Encoder for CI-VAE used in FEMNIST . . 37
5.3 Architecture of the Decoder for CI-VAE used in FEMNIST . . 38
5.4 Mean Train and Val loss for Fed CL model received from the 25 clients 39
5.5 Mean Train and Val loss for Fed CI-VAE model received from the 25 clients 39
5.6 Random Samples Generated for FedRep CI-VAE with 2-shots learning using a Latent Space with 3 dimensions for Client f0068_00 (that have only 50 samples) and the output probability of the Synthetic Label generated 41

List of Tables

4.1 Micro F1 score among different input size 33

5.1 Micro F1 score on FEMNIST with Rotation and w/out fine tuning 37

5.2 Micro F1 score on FEMNIST with Rotation and with fine tuning (with * no fine tuning applied) 40

5.3 Micro F1 for the four different models on active clients' test sets w/out fine tuning 42

5.4 Models' adaptability to each scenario using mean micro F1 on the selected client(s); High > 0.85; $0.75 \leq$ Medium < 0.85; $0.65 \leq$ Low < 0.75; Very Low < 0.65 44

5.5 Micro F1 score on FEMNIST w/out rotation and w/out fine tuning 45

5.6 Micro F1 score on FEMNIST w/out rotation and with 2-shot learning (with * no fine tuning applied) 45

5.7 Sizes of the models proposed on HAR dataset 47

5.8 Micro F1 score on HAR w/out fine tuning 47

5.9 Micro F1 score on HAR with 2-shots learning (with * no fine tuning applied) 48

5.10 Micro F1 score on Active Client 14 Test Set (with * no fine tuning applied) 48

Bibliography

[1] FederatedRep CI-VAE. URL: https://github.com/DitucSpa/FederatedRep_CI-VAE.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*:353–374, 2023.

[3] MNIST dataset. URL: https://keras.io/api/datasets/mnist/.

[4] Jakub M. Tomczak. *Deep Generative Modeling*. Springer, 2022.

[5] Estimating PI | monte carlo animation | python simulation. URL: https://www.youtube.com/watch?v=jnNBM62jT5E.

[6] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[7] Saul Dobilas. Tutorial 5: variational autoencoders. URL: https://www.borealisai.com/research-blogs/tutorial-5-variational-auto-encoders/.

[8] Variational autoencoder - model, ELBO, loss function and maths explained easily! URL: https://www.youtube.com/watch?v=iwEzwTTalbg&list=LL&index=1.

[9] Yufeng Zhang, Wanwei Liu, Zhenbang Chen, Ji Wang, and Kenli Li. On the properties of kullback-leibler divergence between multivariate gaussian distributions. *arXiv preprint arXiv:2102.05485*, 2021.

[10] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[11] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*:462–466, 1952.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Zhiyuan Chen, Waleed Mahmoud Soliman, Amril Nazir, and Mohammad Shorfuzzaman. Variational autoencoders and wasserstein generative adversarial networks for improving the anti-money laundering process. *IEEE Access*, 9:83762–83785, 2021.

[14] Rushikesh Shende. Autoencoders, variational autoencoders (VAE) and β-VAE. *Medium*. URL: `https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-$%5Cbeta$-vae-ceba9998773d`.

[15] One minute overview of variational autoencoders (VAE). URL: `https://www.linkedin.com/pulse/one-minute-overview-variational-autoencoders-vae-saulius-dobilas`.

[16] Variational autoencoder. URL: `https://keras.io/examples/generative/vae/`.

[17] Artidoro Pagnoni, Kevin Liu, and Shangyan Li. Conditional variational autoencoder for neural machine translation. *arXiv preprint arXiv:1812.04405*, 2018.

[18] Luke Ternes, Mark Dane, Sean Gross, Marilyne Labrie, Gordon Mills, Joe Gray, Laura Heiser, and Young Hwan Chang. A multi-encoder variational autoencoder controls multiple transformational features in single-cell image analysis. *Communications biology*, 5(1):255, 2022.

[19] Dinesh C Verma. *Federated AI for Real-World Business Scenarios*. CRC Press, 2021.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data:1273–1282, 2017.

[21] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning:2089–2099, 2021.

[22] Daniel J Beutel and Topal et al. Flower: a friendly federated learning framework, 2022.

[23] Flower documentation. URL: `https://flower.dev`.

[24] Sudharsan Ravichandiran. *Hands-on meta learning with Python: meta learning using one-shot learning, MAML, Reptile, and Meta-SGD with TensorFlow*. Packt Publishing Ltd, 2018.

[25] Lan Zou. *Meta-learning: Theory, algorithms and applications*. Elsevier, 2022.

[26] Introduction to meta learning and neural architecture search. URL: `https://www.thinkautonomous.ai/blog/meta-learning/`.

[27] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks:1126–1135, 2017.

[28] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[29] Meta-learning: learning to learn fast. URL: `https://lilianweng.github.io/posts/2018-11-30-meta-learning/`.

[30] Mohsen Nabian, Zahra Eftekhari, and Alec Wong. CI-VAE: a class-informed deep variational autoencoder for enhanced class-specific data interpolation, 2022.

[31] Michel Barlaud and Frederic Guyard. A non-parametric supervised autoencoder for discriminative and generative modeling, 2022.

[32] Milad Memarzadeh, Bryan Matthews, and Thomas Templin. Multiclass anomaly detection in flight data using semi-supervised explainable deep learning model. *Journal of Aerospace Information Systems*, 19(2):83–97, 2022.

[33] Layer weight regularizers. URL: `https://keras.io/api/layers/regularizers/`.

[34] Hugo Dugdale. Federated learning with variational autoencoders, 2023.

[35] Leaf: a benchmark for federated settings. URL: `https://github.com/TalwalkarLab/leaf`.

[36] Reyes-Ortiz et al. Human Activity Recognition Using Smartphones, 2012. DOI: https://doi.org/10.24432/C54S4K.

[37] Tiny machine learning for human activity recognition. URL: `https://github.com/lorenzo-orsini/Tiny-Machine-Learning-for-Human-Activity-Recognition`.

[38] URL: `https://www.youtube.com/watch?v=aE-hvefJI_0`.

Acknowledgements

I would like to thank Prof. Angelo Trotta for assisting (and the patience) on this journey and for co-developing the thesis with me, I hope there can be other opportunities for collaborations.
I am grateful to Prof. Marco Di Felice for introducing me to the world of AI in IoT systems, and especially to the realm of Federated Learning.
I thank Unibo and all the faculty members of this course for their help and instruction.
Furthermore, I want to express my gratitude to Lorenzo for assisting me with exams over the years and for collaborating on some truly wonderful projects.
I also thank Alessandro, Marzio, and Luca for the growth opportunities they provided, which allowed me to continually learn more in the world of embedded systems and microcontrollers.
I am indebted to my parents for their unwavering support and also Ennio, Nonna, Paola and Rosa for always supporting me.
Lastly, I want to thank the most important person of all, Sara, who has been by my side throughout these years. You have helped, tolerated, and supported me in every possible way. You have guided and assisted me in all my academic endeavors, and I have reached the conclusion of this journey partly because of you [38].