# Green artificial intelligence to automate medical diagnosis with low energy consumption

John Anderson García Henao

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION


# L'intelligence Artificielle Verte pour Automatiser le Diagnostic Médical avec une Faible Consommation d'énergie

**John Anderson García Henao**

Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis

**Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur**
**Dirigée par** : Michel Riveill, Professeur, Université Côte d'Azur
**Co-encadrée par** : Pascal Staccini, Professeur, PU-PH Université Côte d'Azur, CHU Nice
**Soutenue le** : 6 Janvier 2021

**Devant le jury, composé de** :
Frédéric Le Mouël, Professor, INSA Lyon, Rapporteur
Yves Denneulin, Professor, INP Grenoble, Rapporteur
Fabrice Huet, Professor, Université Côte d'Azur, Président
Frédéric Precioso, Professor, Université Côte d'Azur

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION


# Green Artificial Intelligence to Automate Medical Diagnosis with Low Energy Consumption

**John Anderson García Henao**

Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis

**Presented to obtain the degree of doctor in Computer Science at Côte d'Azur University**
**Directed by** : Michel Riveill, Professor, Université Côte d'Azur
**Co-supervised by** : Pascal Staccini, Professor, PU-PH Université Côte d'Azur, CHU Nice
**Defended on**: 6 January 2021

**In front of the jury, composed of** :
Frédéric Le Mouël, Professor, INSA Lyon, Rapporteur
Yves Denneulin, Professor, INP Grenoble, Rapporteur
Fabrice Huet, Professor, Université Côte d'Azur, Président
Frédéric Precioso, Professor, Université Côte d'Azur

# Acknowledgements

I want to express my sincere thanks to my advisors, *Michel Riveill, Frédéric Precioso, and Pascal Staccini*, for their guidance and support in completing my Ph.D. thesis and developing the DianoseNet framework.

I would also like to thank all the colleagues and collaborators in the I3S lab with whom I interacted during my Ph.D. Especially to *Ziqing Du, Mohamed Younes, Arno Gobbin, and Chap Chanpiseth* for their developmental contributions to building the DiagnoseNet framework.

My thanks also go to *Carlos Jaime Barrios Hernandez* for bridging the gap between my master's degree and my PhD and for his support for research collaborations.

I am incredibly grateful to my mother *Miryam Henao Ortiz* for always giving me love, support, encouragement, and to my father *Gricelio Garcia Herrera* that strengthens my soul from a distance.

Finally, I am deeply grateful to my life partner, *Zayne Milena Roa Diaz* for her love, support, and stimulating conversations, as well as for her invaluable ability to listen to me and encourage the path of personal growth.

# Abstract

Automation of data-driven models for medical diagnosis can support the clinical decision process and increase the efficiency of healthcare delivery in clinical settings. Both clinicians and patients will benefit from personalized medicine by detecting critical events and defining new care pathways. However, although deep learning (DL) algorithms have shown high accuracy and exemplary performance in different clinical categories, very few have been integrated into intelligent medical systems. This type of system goes beyond the modeling process. It requires integrating technical and scientific innovation in a clinical setting, solving various technical and research problems to offer real solutions at the hospital level. Some of the most common barriers in the actual clinical setting are data privacy and interoperability, patient representation and multitask learning, the distributed methods to speed up the training models, and the efficient use of the computational resources available in hospitals.

This thesis proposes a modular deep learning framework optimized for medical diagnosis to create portable and scalable solutions over heterogeneous systems. The modules include data-driven representation learning and distributed deep learning methods to develop risk prediction models whit clinical outcomes such as hospital admissions, length of stay, and mortality. In this context in which the data comes from heterogeneous sources and requires intensive computing to gather and be statistically significant, the main contributions of this research thesis were divided into two components. The first component focuses on implementing unsupervised neural architectures to derive a general latent representation of the patients from electronic health records (EHRs) and apply it to different clinical tasks. The heterogeneous EHRs harmonization was based on the Fast Healthcare Interoperability Resources (FHIR) format, allowing model scalability built from EHRs of the hospital A to be replicated in hospital B with different information healthcare system formats. On the other side, the second component accelerates the training process and hyperparameter search to determine an optimal generalization model for a specific medical task using a mini-cluster of Jetson TX2 nodes. The primary approach has been a set of gradient computation modes that adapts the neural network according to the memory capacity, the number of nodes used, the coordination method between nodes, and the available inter-node communication protocol (e.g., GRPC or MPI).

We conducted different experiments using clinical descriptors collected during the first week of hospital stays of patients in the PACA region and using short ECG recordings of 30 to 60 seconds, obtained as part of the PhysioNet 2017 challenge. These experiments allowed us to evaluate the accuracy, convergence time, and scalability of our proposed

framework. This framework, available in open-source, is called DiagnoseNET for medical diagnosis. It automates the definition of the neural architecture, the search for hyper-parameters, the distribution of data on the different compute nodes, and the purpose of treatment batches in a single API. Furthermore, its execution engine orchestrates the gradient computation on the various nodes according to different cooperation strategies.

**Keywords :** Clinical Prediction Models, Representation Learning, Distributed Deep Neural Networks, Green Computing.

# Résumé

L'automatisation de la modélisation basée sur les données pour le diagnostic médical peut soutenir le processus de décision clinique et accroître l'efficacité de la prestation des soins de santé dans les établissements cliniques. Tant les médecins que les patients en tireront profit, soit par la détection d'événements critiques, soit par l'émergence d'une médecine personnalisée, soit encore par une meilleure définition des parcours de soins. Cependant, malgré le fait que les algorithmes d'apprentissage profond (DL) ont montré une grande précision et de bonnes performances dans différentes catégories cliniques, très peu ont été intégrés dans des systèmes médicaux intelligents. Ce type de systèmes va au-delà du processus de modélisation et nécessite l'intégration de l'innovation technique et scientifique dans le contexte d'un cadre clinique, en résolvant divers problèmes techniques et de recherche pour offrir des solutions réelles au niveau de l'hôpital. Certains des obstacles les plus courants dans le contexte réel sont liés à la confidentialité des données et à l'interopérabilité ; à la représentation des patients et à l'apprentissage multitâche ; aux méthodes distribuées pour accélérer les modèles de formation et à l'utilisation efficace des ressources informatiques disponibles dans les hôpitaux.

Cette thèse propose un cadre modulaire d'apprentissage approfondi optimisé pour le diagnostic médical afin de créer des solutions portables et évolutives sur des systèmes hétérogènes. Ces modules comprennent des méthodes d'apprentissage approfondi basées sur les données, l'apprentissage par représentation et l'apprentissage approfondi distribué afin de développer un modèle de prédiction des risques avec des résultats cliniques tels que les admissions à l'hôpital, la durée du séjour et la mortalité. Dans ce contexte où les données proviennent de sources hétérogènes et nécessitent un calcul intensif pour être collectées et statistiquement significatives, les principales contributions de cette thèse de recherche ont été divisées en deux volets : le premier volet vise à mettre en œuvre des architectures neuronales non supervisées pour dériver une représentation générale latente des patients à partir de dossiers de santé électroniques (EHRs) pouvant être appliqués à différentes tâches cliniques, et à harmoniser les HERs hétérogènes sur la base du format FHIR (Fast Healthcare Interoperability Resources), ce qui permet de reproduire l'extensibilité des modèles construits à partir des HERs de l'hôpital A dans un hôpital B avec différents formats de systèmes d'information sur les soins de santé. Le deuxième volet vise à accélérer le processus de formation et le réglage des hyperparamètres afin de déterminer un modèle de généralisation optimal pour une tâche médicale spécifique en utilisant un mini-groupe de nœuds Jetson TX2. La principale approche a consisté en un ensemble de modes de calcul de gradient permettant d'adapter le réseau neuronal en fonction de la capacité de mémoire, du nombre de nœuds utilisés, de la méthode de coordination entre

les nœuds et du protocole de communication inter-nœuds disponible (par exemple GRPC ou MPI).

Nous avons mené différentes expériences en utilisant des descripteurs cliniques recueillis au cours de la première semaine de séjour des patients à l'hôpital dans la région PACA ou en utilisant de courts enregistrements ECG de 30 à 60 secondes, obtenus dans le cadre du défi PhysioNet 2017. Ces expériences nous ont permis d'évaluer les performances en termes de précision, de temps de convergence et d'évolutivité du cadre que nous proposons. Ce cadre, disponible en open-source, est appelé DiagnoseNET pour le diagnostic médical. Il automatise en une seule API la définition de l'architecture neurale, la recherche d'hyperparamètres, la distribution des données sur les différents nœuds de calcul ainsi que la définition des lots de traitement. Son moteur d'exécution est chargé d'orchestrer les calculs de gradient sur les différents nœuds selon différentes stratégies de coopération.

**Mots-clés :** Modèles de prédiction clinique, apprentissage par représentation, réseaux neuronaux profonds distribués, informatique verte.

# Table des matières

# Table des figures

# Chapitre 1

# Introduction

## 1.1 Research Context

### 1.1.1 Clinical risk prediction models

Health researchers along with computer scientists are widely adopting deep learning methods and computer technologies, whose contributions make it possible to compose the next Intelligent Medical Systems (IMS) to improve the quality of care and facilitate the clinical decision support as well as to determine who will be hospitalized, what procedures they will prioritize and what results will get the best performance. The main challenges facing the IMS are generating accurate patient profiles and predictive models from large volumes of healthcare data. In which only the United States produced 153 Exabytes of health data in 2013 and 2,314 Exabytes are estimated for 2020, showing an annual growth rate of 48% [Stanford Medicine 17]

However, a 2016 systematic review by Goldstein et al. in the medical literature that evaluates clinical studies that used Electronic Healthcare Records (EHR) to develop risk prediction models, identifying that many studies did not fully utilize the depth of information on the patients available in the EHR, using a median of only 27 variables and most used extensions of traditional generalized linear models [Goldstein 16]. In contrast to recent studies that use deep neural networks and machine learning algorithms to extract more patient characteristics as input to implement risk prediction models, have reported promising results in clinical outcomes such as hospital admissions to prioritize patients for preventive care uses, length of stay, and discharges to determine the release time and mortality for high-risk warning patient of in-hospital mortality before their death.

**In hospital admission outcomes**, a recent study reports that machine learning models as random forest (RF) and gradient boosting classifier (GBC) have higher performances than the Cox proportional hazards (CPH) model in predicting the risk of

emergency hospital admission, comparing the ROC curves by the three models with different predictor sets extracted from EHRs of 4.6 million patients, including patient demographics, lifestyle factors, laboratory tests, currently prescribed medications, selected morbidities, and previous emergency admissions [Rahimian 18].

**In the length of stay outcomes**, a recent study compares the performance of Multilayer Perceptron Network (MLP) and Adaptive Neuro-Fuzzy Inference System (ANFIS) to predict patients' length of stay at intensive care units after cardiac surgery, reporting that the ANFIS resulted in the creation of a more precise model than the MLP, in which the neural network hyperparameter search for a generalized model to be compared is a limitation [Maharlou 18].

**For the risk of mortality outcomes**, a recent study compares RF, XGBoost, Support Vector Machine (SVM), LASSO, and K-nearest neighbors to calculate the risk score of mortality for each inpatient day during the in-hospital episode and use the quintiles of these calibrated risk scores to stratify risk groups, in which RF has been proven to have high accuracy as it overcomes overfitting by selecting random subsets of features to build smaller trees and can handle potential errors caused by unbalanced case-control datasets [Ye Chengyin 19].

### 1.1.2 Patient phenotype representation

To derive patients' phenotypes, it is necessary to extract their medical data (demographics, medical diagnoses, procedures performed, cognitive status, etc.). Although possible, the evolution of this information over time must be extracted. A used method is *vector based representation* in which, for each medical target is constructed a matrix correlation between patients and medical group features [Wang 14], The generation of the different vectors generally takes a critical time. A couple of other possibilities are *non-negative matrix factorization*, and *non-negative tensor factorization* for extracting phenotypes as a set of matrices, tensor candidates that show patients clusters linked on specific medical features and their date [Ho 14, Perros 17, Perros 18]. Other approaches use non-negative vectors for embedding the clinical codes and use word representations as (skip-gram or Glove) to generate the corresponding visit representation [Choi 16b].

**Unsupervised representation learning**, after the success of unsupervised feature learning for training unlabeled data to dimensionality reduction and learn good general features representations and used either as input for a supervised learning algorithm [Bengio 14], the application of employ it to produce patient phenotype representations can

significantly improve the predictive clinical model for a diverse array of clinical conditions as it was shown in deep patient approach [Miotto 16]. Other derivative approaches use a record into a sequence of discrete elements separated by coded time, which uses the unsupervised embedding Word2Vec to pre-detected the continuous vector space, them uses a convolution operation which detects local co-occurrence and pools to build a global feature vector, which is passed into a classifier [Nguyen 17]. Another approach trains a recurrent neural network with an attention mechanism to embed patients' visit vector to visit the representation, which is then fed to a neural network model to make the final prediction [Choi 16c].

However, these approaches to derive patients' phenotypes algorithms demand considerable effort in deploying preprocessing pipelines and data transformation, which are built without considering the response time. In this perspective, a large number of authors have explored scaling up deep learning networks, well-known training datasets focused on the impact of synchronization protocol, and state gradient updates [Dean 12a, Keuper 16, Suyog Gupta 17]. At the same time, other groups have been working on high-level frameworks to quickly scale out to multiple machines to extend libraries for parameter management to allow more agile development, faster and fine-tuning hyper-parameter exploration [Dünner 18]. However, not all of these developments are applied to medical care and do not consider energy consumption.

Nevertheless, most of these approaches focus on a single medical task for customizing their models and patient representations for a specific outcome. Instead, this thesis implements the unsupervised neural architecture that Miotto et al. Whose objective is to derive a general latent representation of the patients from the clinical data warehouse that can be applied to a different number of clinical tasks. [Miotto 16]. In addition, we harmonize the EHR in a single format based on the clinical document architecture (CDA) [Mandel 16], to allow the scalability of models built with EHRs from a hospital "1" to be replicated in a hospital "2" with different information healthcare system formats [Rajkomar Alvin 18].

### 1.1.3 Distributed deep neural network

On the other hand, generative models or translation models that use encoder-decoder methods to generate a latent representation that will be used as input for a second model is an expensive task that generally requires extreme hyperparameter exploration and massive parallelization to determine an optimal generalization model for a multiple medical tasks [Stanley Kenneth O. 19]. Furthermore, scale deep learning workflows over a

wide range of emerging heterogeneous system architecture increases the programming expressiveness complexity for model training and computing orchestration. In addition, with the beginning of an open era of instruction set architecture, the HPDA applications require abstracting a variety of domain-specific architectures and languages. For example, the Nvidia embedded computing family (Jetson) uses a hybrid processor on-chip, with an ARM processor coupled with one streaming multiprocessor, delivering computing capabilities similar to a GPU workstation under low power consumption [Boggs 15].

**The data-distributed methods** that use iterative-convergent Machine Learning (ML) algorithms for training, such as Bosen [Xing 15] and [Konecný 16, Bonawitz 19] who have extended these approaches to Federated Learning (FL). They can be applied generically to any ML method if data samples are independent and identically distributed. The Bosen platform provides a distributed version for some well-known ML algorithms (for example, Deep Learning, Sparse Coding, K-means clustering, Random forests or Multi-class Logistic Regression), while the FL approach is designed to be efficient in setups with many users and unreliable or slow connections. Final classification or prediction models represent a weight matrix that is stored across a large number of clients. The local weight matrix is calculated in the initial step and refined over the rounds, where updates are based on the exchange of parameters with local neighbors or a single master node.

**The model-distributed approaches** such as Strads platform [Xing 15] require ML specialized systems that perform a partition of ML algorithms into a set of parallel tasks, in general, scheduled by master node(s) and executed by a set of workers. Schedulers' task is to separate the problem into a non-overlapping set of sub-problems, divide a workload and synchronize the updates amongst the workers. This setup admits non-conflicting model updates that lead to convergence. Numerous algorithms can be deployed in this framework, such as Latent Dirichlet Allocation, Matrix Factorization, Support Vector Machine or Deep Learning algorithm based on Caffe, called Poseidon, to name a few.

**The model and data-distributed algorithms** for classification and prediction problems. In the literature, there exist only a few works. A hybrid distributed platform known as Angel [Jiang 17] appropriately combines data partitioning, scheduling, and parameter synchronization tasks and demonstrates accuracy improvement in comparison with a Petuum-based data or model distribution. In addition, there exist many parallelization methods, such as FlexFlow [Jia 18]. It is a hybrid data and model parallel (non-distributed) approach worth exploring in a distributed setup because it performs an automated search of parallelization strategies that incorporate data, attribute, parameter, and operator parallelization for DNN algorithms.

### 1.1.4 Green artificial intelligence

The remarkable precision of deep neural networks to solve general tasks in various domains, such as computer vision, speech recognition, and natural language processing, is largely due to increased complexity in the neural architecture and the number of experiments carried out to adjust its hyperparameters. Consequently, developing a generalizable model with deep neural networks is an expensive process due to the time of cloud computing or the cost of hardware and electricity required to power modern computing systems and its environmental impact to process large and multiple models. [Strubell 19]. To estimate the total cost to develop deep neural network models, [Schwartz 20] proposes a processing cost equation $Cost(R) \propto E * D * H$. In which, they illustrate three factors that linearly influence the total cost to produce a result $R$, the cost of executing a single model $E$, the size of the training dataset $D$ and the number of hyperparameter experiments $H$.

Therefore, we consider two methods to train neural networks : The first method is based on human designs to improve a neural architecture through the evolution of state of the art in a specific domain, which often involves building deeper models with more sophisticated layers, among other techniques, increasing the cost of execution $E$ per model and at the same time reduces the number of models to explore. While the second method is based on the neural architecture search to automate the design of neural architectures through search strategies such as reinforcement learning or evolutionary algorithms with policies to select a subgraph that maximizes the validation set's expected reward, increasing exponentially, the hyperparameter space $H$ to explore by each model generate $E$.

**Neural architectures by hand design,** Since AlexNet achieved a 5.3% error in the top 5 in the ImageNet challenge by using two GPUs to calculate convolution operations with 60 million parameters of the neural network [Krizhevsky 12]. This allows designing deeper neural networks, including mixed layers with convolutions, dropout, and others, which allowed achieving an error of 1% in 2017. While the amount of compute used to train deep neural learning models has increased 300.000X in 6 years.

In medical images, initializing the weights using pretrained models with ImageNET are widely adopted, facilitating the hyperparameter search space and fast convergence ; as the case of skin cancer classification from dermatology images [Esteva Andre 17] ; prediction of cardiovascular risk factors from eye images [Poplin Ryan 18] ; and among others [Raghu 19]. Nevertheless, transferring knowledge or benefit from a pretrained model to a new medical task is not always possible.

In these situations, designing a neural architecture like Inception, ResNet, and DenseNet from scratch requires exploring many candidates and finetune each one to discover a well-performing model, which is a scientific process of trial and error.

**Neural architecture search,** To overcome these limitations, automated methods and processes are increasing ; in which it is common to use a controller to generate model descriptions of neural architectures and train each model while maximizing the expected precision of generated models in a validation set [Elsken 18]. For example, in image classification with CIFAR-10, using the same number of parameters (27.6M), DenseNet achieves an error rate of 3.74%, while NASNet automated architecture achieves an error rate of 2.4%. Although this last is computationally expensive and time-consuming, it was necessary to train 20000 models with 500 GPUs (Nvidia P100) in 4 days at 2000 GPU-hours [Zoph 17], and in a previous architecture search was used 800 GPUs (Nvidia k40) in 28 days resulting 22400 GPU-hours [Zoph 16].

Consequently, designing deep neural networks with automatic methods to generate a population of models during the search space enables extreme exploration and massive parallelization, which translates into energy consumption in a long-time window [Stanley Kenneth O. 19]. Taking all these concepts into account, In this thesis, we used a green artificial intelligence term as an integrative evaluation criterion to reduce the environmental impact of developing deep neural networks, whose essential evaluation criteria comprises two optimization challenges :

1. Automating the exploration strategy to reduce the model trade-offs space ;
2. Tailoring each model generated to exploit the computing platform and minimize the energy consumption.

## 1.2 Objective and Motivations of this Thesis

This thesis aims to develop green neural networks and scale them in heterogeneous systems to accelerate the construction of clinical models for predicting multitask risks with an effective balance between accuracy and energy consumption. The main tasks that we have studied are not limiting concern ; the care purpose of hospital admissions and the prediction of the major clinical category to answer the question : in which service will the patient be rereferred after admissions ? Then the prediction of the length of stay to answer the question : How long will the bed be occupied ? Furthermore, finally, we try to predict mortality risk in the hospital. The available data relating to hospitalized patients in the Provence-Alpes-Côte d'Azur region (PACA), could make it possible to predict many other targets : the patient's destination at the end of their stay (home, care-home, hospital transfers, and death), as well as the primary medical procedures that will be performed, the risk of emergency readmission, etc.

The construction of all these models requires a high computing power to operate and train the unsupervised network to generate dense patient profiles that represent the state of health (patient phenotype representation) and then retrain the different prediction risk models. The confidentiality of the data requires data anonymization, which loses quality in the data by adding blur by grouping specific terms in more generic terms. The volume of data processed is relatively large, and the patient records are fully anonymized with an average of more than 670,000 records per year, to compose the Inpatient dataset from the PMSI PACA from patients hospitalized between 2006 to 2011.

The overriding question of this research is related to : is it possible to efficiently use a platform of GPU embedded nodes to build multitask predictive risk models with an appropriate level of performance ?. The first underlying idea is that if the answer is positive, it would then be possible to deploy this platform of embedded GPU nodes within the services-producing the data in order to be able to process them in-situ of hospitals, to free themselves from the problems of anonymization necessary to export the data outside of their place of production. The second idea is that if one wishes to process complex and voluminous problems using processors with low capacity, whether in terms of memory, execution speed, or throughput, between the different components, it is then necessary to distribute the data set as adequately as possible, to minimize memory movements and to determine precisely the best hyper-parameters to minimize the ratio : relevance of the result concerning its cost, which may be the response time or energy consumption. The two are often linked.

## 1.3 Contributions and Thesis Outline

The main contribution of this thesis is the automatization and the harmonization of distributed processing and coordination methods for training and finetuning neural networks over heterogeneous systems, avoiding to the researchers the necessitate to writing the new coordination gradient codes abstracting the complexity to data placement and memory management over several nodes for each architecture (X86 and ARM) or test new models in the hyperparameter search process. Likewise, DiagnoseNET made a workload characterization, which collects the GPU, CPU, memory tracks, and energy consumption metrics. At the same time, the DNN model is executed on the target platform, whose information is synthesized in terms of accuracy and energy ratio for the programmer or the scheduler, could be used to adjust the task granularity : model dimension and batch partition, according to the memory capacity and the number of nodes in the subsequent execution.

In the healthcare research field, preserving patient privacy is a principal requirement.

DiagnoseNET allows training a neural network using different communication protocols as MPI and GRPC. Both methods use the data and resource manager module for training the DNN model keeping a balanced portion of the dataset by each node only transferring gradients to the server. Over this methodology, DiagnoseNET can be used as a distributed platform across multiple hospitals to train DNN models without sharing patient data, using low-power consumption clusters with minimal infrastructure requirements. Overcoming several challenges in implementing DiagnoseNET as a modular framework include : Building an expression programming module to build dynamic neural networks without rewriting the code for a new platform or execution modes. Automatize the dataset splitting, placement, and balancing the batch over the nodes and their memory constraints, having input sets with different dimensions as both study cases.

### 1.3.1 Document Organization

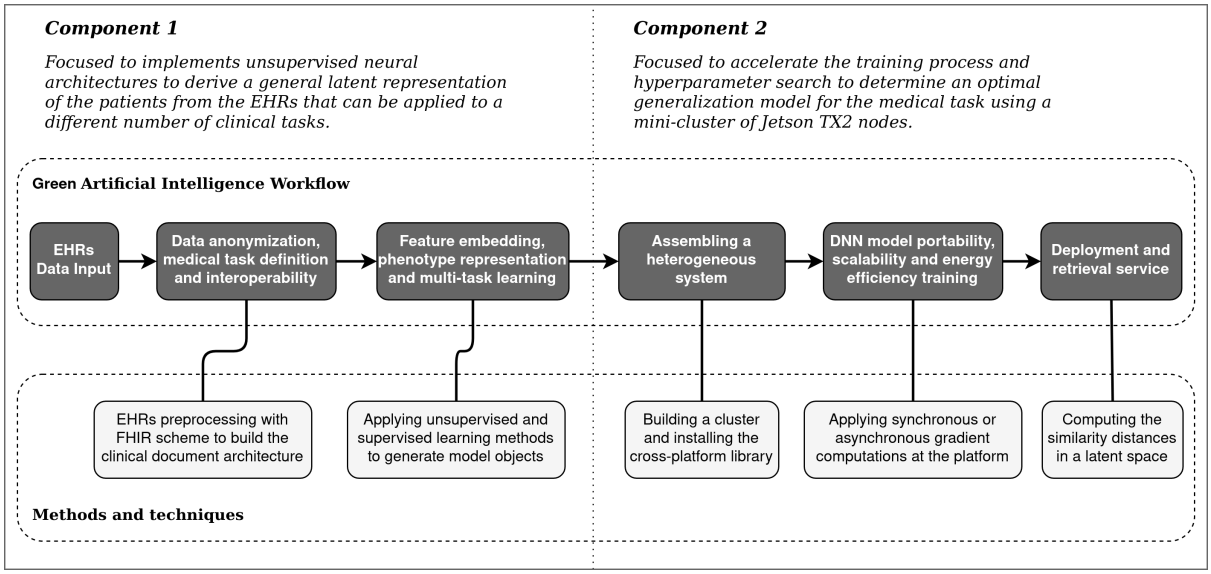



FIGURE 1.1 – Outline to highlight the artificial intelligence workflow, methods, and techniques included in the thesis manuscript.

Figure 1.1 shows the high-level contributions of this thesis, which has been organized into two components that bring together the methods and techniques required to process the complete artificial intelligence workflow proposed. The **first component** is focused on implementing unsupervised neural architectures to derive a general latent representation of EHRs patients to be applied into multitasking clinical risk models. This part consists of the following chapters. **Chapter 2** describes the learning healthcare systems and briefly describes the interoperability module for clinical data processing in the Fast Healthcare Interoperability Resource (FHIR) scheme to build the architecture of clinical documents per patient to develop predictive models, as well as presenting the patient flow

in the Côte d'Azur healthcare system and defining the case study. **Chapter 3** discuss patient phenotype extractions and apply them to predict different medical targets. It provides three high-level features : a complete deep learning workflow orchestration into stage pipelining for mining clinical data and using unsupervised feature representations to initialize supervised models and the data resource management for training parallel and distributed deep neural networks.

The **second component** is focused on speeding up the training process and fine-tune the hyperparameters to determine an optimal generalization model for the medical task using a mini-group of Jetson TX2 nodes. **Chapter 4** combines the hardware and software perspectives to build scalable deep learning models on embedded low power systems. From the hardware perspective, practices and lessons learned from assembling a cluster of embedded GPU nodes, the standard of the implemented architecture, and the general cross-platform library to integrate two levels of parallel and distributed processing deep neural networks. In contrast, the software perspective is introduced the programming framework designed for scaling deep learning models over heterogeneous systems applied to medical diagnosis. It is designed as a modular framework to enable deep learning workflow management and allow neural networks' expressiveness written in TensorFlow. At the same time, its runtime abstracts the data locality, micro batching, and distributed orchestration to scale the neural network model from a GPU workstation to multi-nodes. **Chapter 5** discusses qualitative and quantitive aspects to balance accuracy and energy-efficient to train deep neural networks on heterogeneous systems and describes the granularity of neural network tasks as a combination of data parallelism and mini-batch online learning with capabilities of platform memory as a factor for convergence model principles. **Chapter 6** summarizes our conclusions, gives an overview to automatize artificial intelligence workflow applied in medical diagnoses produced as part of this research, and outlines ideas for future work based on this thesis.

# Chapitre 2

# Towards a Learning Healthcare Systems

Clinical informatics systems were designed primarily to obtain patient information and perform administrative healthcare tasks such as billing, administering procedures, and medications, among others. However, in the last decade, several investigations have been carried out based on the secondary use of electronic health records (EHRs), which has allowed advances in clinical research based on the massive analysis of patients and their integration with learning models for the development of personalized medicine.

In particular, given the increasing volume of data in healthcare, data-driven modeling based on machine learning and deep learning methods has become a powerful approach in medical research for the risk prediction model. —Some approaches for EHRs mining and generating patient new stratification principles and revealing unknown disease correlations [Miotto 16]. Others approach combining the EHRs with genetic data to give a more refined understanding of genotype-phenotype relationships [Sánchez-Valle Jon 20]. Nevertheless, there have been few implementations of risk prediction models in real clinical settings due to the complexity of standardized units and vocabulary among healthcare systems worldwide. In addition, some health systems present challenges and standardize multiple independent sub-systems in the same institution.

## 2.1 Learning Healthcare Systems

The learning healthcare system *(LHS)* has recently emerged as a potential solution-based system to link routine healthcare systems, patient values, and the best available scientific information directly to healthcare practitioners to support the clinical decision making *(CDM)*. Consequently, the LHS has been conceptualized from the paradigms of

*evidence-based medicine (EBM)* and *translational research medicine*, taking advantage of the ubiquitous use of electronic medical record systems and technological advances in the area of artificial intelligence to develop models according with the care target [Delaney 15].

The EBM paradigma aim to incorporate knowledge generation processes based on hospital data collected from daily practice to provide feedback on their diagnoses, procedures and provide better medical care to their patients [Ethier 18]. While the translational research medicine involves fundamental scientific principles for turning biomedical research into practical applications or for expanding knowledge in the field of medicine to improve human health and well-being [Adithan 17].

Nevertheless, there is a wide variety of health systems around the world, with organizational structures designed according to their needs and resources to provide primary health care and public health. As a result, there are multiple useful frameworks for developing an LHS, but each framework has been targeted according to the requirements of your health systems [White 15, Smith 20]. For example, the *Health Innovation Program (HIP) Model* [Smith 20], proposed to build and implement effective LHS for complex case management health systems, are comprised of four process steps, including (1) identifying critical questions, (2) conducting research and evaluations, (3) sharing results, and (4) implement changes. While the *TRANSFoRm Project* [Delaney 15, Ethier 18], proposed an architecture for the LHS with respect to functional and interoperability requirements to support primary care, which describe a software ecosystem for building generic middleware components that provide essential shared functions for LHS applications, such as vocabulary services, secure data transport, authentication methods, and provenance services ; while in the second work the methods to access, process and operate the data are specified, classifying them in data warehousing and data federation. And the *Heimdall Framework* [Mclachlan 18], proposed a classification of target care systems for LHS, whose main branches are cohort identification to determine the feasibility of studies such as risk modeling and decision-making as the first operational step of LHS ; and intelligent assistance to automate routine processes, such as pathology order pre-filling and surveillance monitors for disease outbreaks or treatment problems.

Based on the steps of these models and their main requirements, we composed a high-level framework to develop a continuous LHS integrating artificial intelligence methods and computing techniques to manage access and develop the service of target care systems through a data warehouse or a data federation, for allowing the implementation of the EBM and translational research medicine paradigms, respectively, as shown in Figure 2.1.

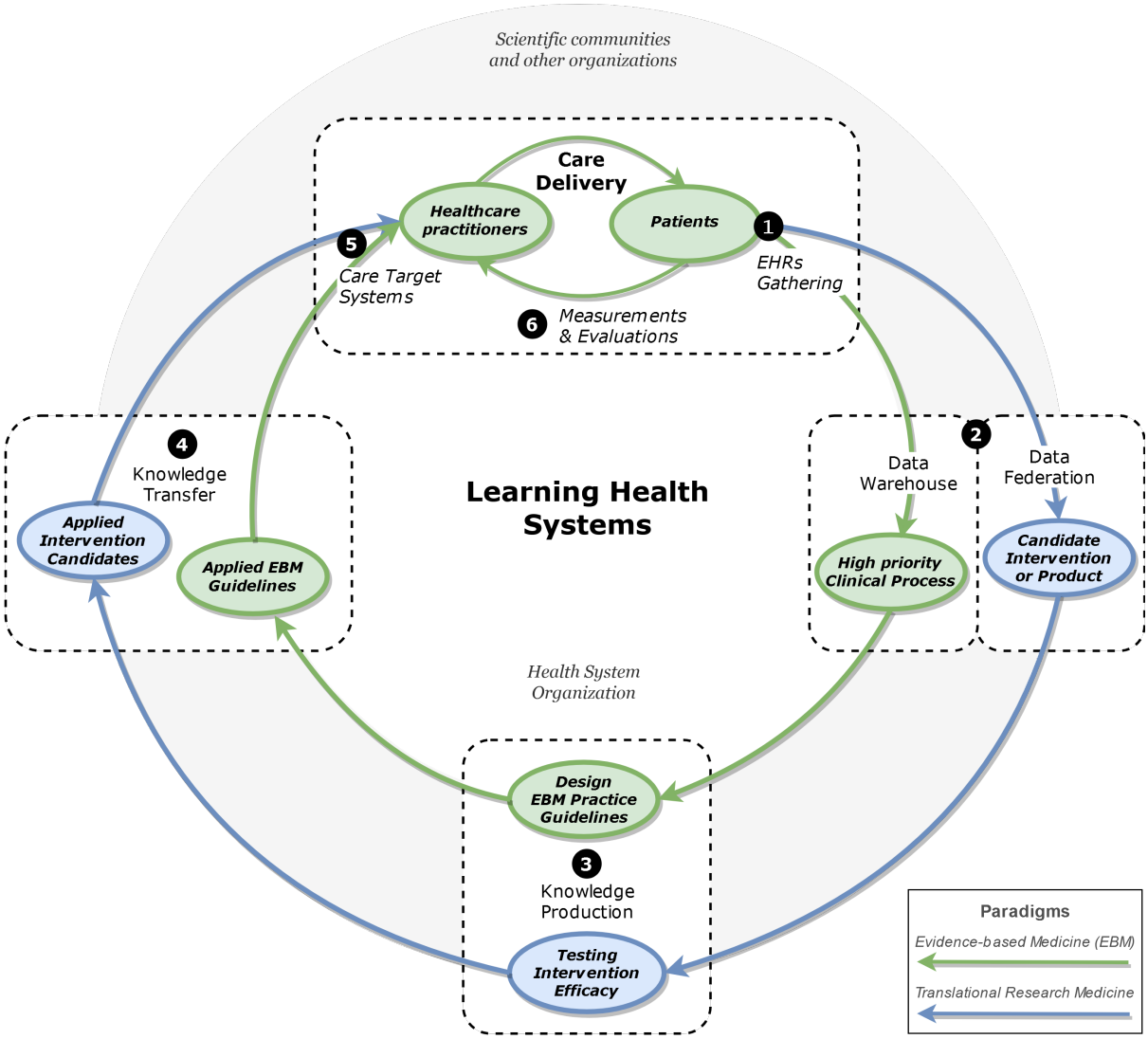



FIGURE 2.1 – The LHS framework is comprised of a continuous learning cycle that systematically collects information from the patient to incorporate knowledge generation processes, developing clinical decision support systems across the organization to produce continuous improvement in the delivery of care, based on translational research and evidence-based medicine paradigms.

1. It should allow the collection of EHRs in a standardized way to develop MBE studies that could be applied in other organizations or to implement research interventions andor products of other organizations.

2. It must have a level of data integration, which allows : storing and structuring the EHRs in a data warehouse *(local model)* in agreement with the organization to identify the high priority clinical process to develop a clinical decision support system ; while the second component, called data federation, maps the specific queries of the EHRs into a central standard *(global model)*, to validate a new cadidates intervention or product developed by the scientific community or other organizations.

3. It should minimize the resources required by an organization to develop local and central models or participate in research projects.

4. It should be flexible enough to accommodate the addition of new data sources or deletion or change of current ones.

5. It must allow the integration of data from multiple sources belonging to various fields, such as care, research and knowledge data to implement decision support systems, by being able to provide relevant pieces of information, feedback and alerts based on the data of the patient and the population.

6. It should be possible to trace both protocol variations and short- and long-term patient results, such as intermediate, final clinical, costs and satisfaction outcomes.

### 2.1.1 LHS Taxonomy of Medical Applications

Based on a qualitative meta-narrative study by [Mclachlan 18], whose study proposes a taxonomy to unify and reduce diversity and fragmentation to develop learning health systems. Therefore, here we describe six types of medical applications empowered by advances in artificial intelligence. Integrating them into a learning health system benefits both clinicians and patients by detecting critical events, such as the emergence of personalized medicine or a better definition of care pathways.

— **Discrete-Event Simulation (DES) :** Is a multi-method that discretizes the sequence of events over time to model the operation of a system. It is used in healthcare organizations to increase patient satisfaction and to reduce treatment costs. For example, DES is combined with design of experiment approach to simulate an emergency department and optimize the average waiting time of a patient to be attended [Atalan 20].

— **Predictive Patient Risk Modeling (PDRM) :** Clinicians can use patient predictive risk modeling to estimate the likelihood that an individual will experience a triple failure event in a specific future period. Predictive patient risk modeling relies on patterns discovered in inpatient data sets to identify cohorts at increased risk for future adverse events [Lewis 13].

— **Clinical Decision Support Systems (CDSS) :** The CDSS is an active healthcare system that matches patients' features against computerized algorithms to

generate patient-specific treatment patient-specific treatment recommendations. The CDSS is one of many complex and challenging decision support systems, primarily because of the various measurable and non-measurable features involved in decision making and the complex relationships between those attributes. The CDSS algorithms can be practical tools to support the decision-making process, but the patient features entered into the algorithms must be measurable and quantifiable. The clinical decision systems involve various systems, including medical devices, applications, and a variety of other types of software. Are active knowledge systems where two or more characteristics of the patient are matched to computerised knowledge bases with algorithms generating patient-specific treatment recommendations [Sandeep Kumar 20].

— **Comparative Effectiveness Research (CER) :** Compares interventions and outcome data with the current patient to determine the best outcome. CER uses an LHS to analyze outcome data from other patients with similar attributes to the current patient. Thus, they are enabling the clinician to determine the appropriate outcome for the current patient. CER can be done by using observational, quasi-experimental, and experimental study designs. These methods can use routinely collected data to fill gaps in the evidence base, ensuring more effective care in a more timely and efficient manner than would be possible with traditionally designed [Foley 17].

— **Intelligent Assistance or Computer Aims Diagnosis (CAD) :** The use of clinical data sources can be augmented with artificial intelligence (AI) to improve accuracy and efficiency while reducing unwanted variability. This can be achieved by training an AI system with annotated labels derived from high-quality reports from expert radiologists on a database of anonymized reports. The AI system can be used as a concurrent reader to aid a less experienced radiologist in making more accurate interpretations. The AI system can also be used as a pre-reader for a radiologist to improve efficiency and consistency of reports due to the anchoring of findings in the consistent results of the complementary AI [Winkel 21].

— **Health Surveillance Systems (HSS) :** HSS is intrinsically data-driven health surveillance for identifying early signals of health anomalies or disease outbreaks. The heterogeneous collection of data sources carries two technical challenges in Public Health Surveillance : the first data sourcing challenge is quickly determining operationalizable data sources that contain valuable signals, and the second analytics challenge is concerned with developing effective computational frameworks to extract such signals. AI offers solutions to help predict the course of

infectious diseases and emergencies and analyze and evaluate interventions and responses. Firstly, a data-driven method was developed to automatically identify and select keywords from social media text for disease surveillance purposes. For instance, with the rapid development of the Internet and the Internet of Things applications, ubiquitous social and device sensing capabilities are becoming a reality, presenting significant surveillance potentials. Secondly, AI provides comprehensive methods for dealing with unstructured and semistructured text, images, and videos, in addition to structured information items for automatic data-driven feature construction. Thirdly, AI offers modeling frameworks that allow the study of the evolution of epidemics under different conditions by simulating complex configurations and scenarios of infectious disease transmission and public health responses. For example, simulating the evolution of epidemics in time, space, and infection transmission dynamics using nonlinear techniques such as AI multiagent systems [Thiébaut 19].

### 2.1.2 The Interoperability Challenge in Healthcare Systems

The first difficulty for healthcare systems in transforming from data-driven healthcare to a knowledge-driven healthcare system is the health information exchange from a patient-centered perspective [Braunstein 18a]. It was moving as a scalability difficulty for deep learning-based medical applications to integrate into new clinical settings. Thus, the model that works in hospital $A$ does not work in hospital $B$ due to the structure and semantics of the patient-related information required to feed the DL medical application [Esteva 19].

The EHR interoperability challenges occur at different scales, from minimal communication and data exchange between several independent software within the same organization to the diversity of codes used to record the same characteristics of the patient between internal units and external organizations. For example, the billing system constantly integrates information when one or more records are created for each patient in one or more hospital units. As a result, the databases can have simple problems, such as gender being recorded in radiology as $M$ and $F$. In contrast, the pharmacy system uses *one* and *zero*, and in the laboratory, *zero* and *one*. [Reid 05]. The other example refers to semantic confusion, in which an $ICPC - X76$ code relates to breast cancer in the *International Classification of Primary Care*, but depending on the structure of the database, an instance of this terminology code may denote various diagnoses of breast cancer like a diagnosis of a current condition of the patient, a past state of the patient, or the current condition of a family member [Ethier 18].

Therefore, to implement clinical decision systems, healthcare information systems must exchange EHRs in any format between any healthcare system to maintain the meaning of the information being exchanged. The goal is to directly support the safe, timely, efficient, effective, and equitable delivery of patient-centered care. For this purpose, it is necessary to build healthcare information systems with the following interoperability layers :

1. **Foundational Layer :** Achieves the ability of patients to receive their personal health information electronically by enabling one system or application to securely store, transmit, and receive data from another organization's system. Those systems will include a *health information exchange* (HIE) shared electronically among providers, patients, and families. When health information is shared, patients have access to the information they need to make informed decisions regarding their health care. In addition, their providers have access to the clinical information they need to provide quality care, and healthcare organizations can provide more efficient, effective, and cost-effective care.

2. **Structured Layer :** It is the constant transfer of health data from one system to another that preserves and does not change the data's purpose and clinical or operational meaning. It, therefore, defines the format, syntax, and organization of the exchanged information, including at the level of data fields for interpretation. The intermediate form of interoperability requires the receiving system to interpret meaning at the data field level. In the most advanced form of structured interoperability, specific data fields are placed in locations that indicate their purpose [Braunstein 18b].

3. **Semantic Layer :** The semantic layer is a set of ontological relationships that operate on a set of data elements. This is the highest level of interoperability and requires sufficient shared standards to be consistent, accurate, and relevant to the data element definitions. This level of interoperability enables a receiving system to interpret the data to perform a task as if the data had been generated in the receiving system [Braunstein 18b].

4. **Organizational Layer :** This layer covers the details of implementation, management, and even legal processes to turn the concept into a tangible solution. It includes governance, policy, social, legal, and organizational considerations to facilitate secure communication and use of data both within and between organizations, entities, and individuals. For example, a Federated Learning (FL) approach ad-

dresses privacy and data governance challenges by enabling ML from non-co-located data. In an FL environment, each data controller defines its governance processes and associated privacy policies, controls access to the data, and reviews the data. In addition, the FL includes the training and validation phases. In this way, FL could create new opportunities, for example, by enabling large-scale institutional validation or enabling novel research on rare diseases [Rieke 20].

### 2.1.3 Fast Healthcare Interoperability Resources (FHIR)

Health Level Seven (HL7) [1] developed the Fast Healthcare Interoperability Resources (FHIR) as a standard for harmonizing and exchanging diverse clinical types such as patient, practitioner, diagnoses, procedures, medications, and others, between healthcare information systems. The FHIR structure the EHRs as a collection of "resources" using embedded ontology references such as LOINC, SNOMED, and others to ensure a common vocabulary among the various healthcare practitioners. [Braunstein 18b].

FHIR resources are created in JSON, XML, or RDF, with JSON being the most commonly used format as a common language between services and practitioners. The main implementations of FHIR server-structured EHRs are :

— FHIR server uses a common standard called Care Connect Reference Implementation CCRI to develop APIs on top of the existing database structure to communicate EHRs between various software. For example, build an FHIR server with data loading on the radiology, laboratory, and medical history system databases to integrate the patient features and compose clinical analysis to assist specialists in making decisions.

— Another approach is to use the FHIR profile structure to standardize clinical features used to feed medical intelligence applications such as patient risk models, clinical decision support systems, and others. Thus, allowing these artificial intelligence medical applications to scale from a hospital A to a hospital B.

1. The health level seven (HL7) is a not-for-profit, ANSI-accredited standards developing organization dedicated to providing a comprehensive framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services. HL7 is supported by more than 1,600 members from over 50 countries, including 500+ corporate members representing healthcare providers, government stakeholders, payers, pharmaceutical companies, vendors/suppliers, and consulting firms.

**FHIR Resource Structure :**

The EHR elements are structured by blocks called Resources. For example, the *DiagnosticOrder* resource contains the orders placed by clinicians for imaging studies of a specific patient, which in turn is structured in a *Patient* resource [Kamel 18]. Additionally, the structure definitions describe the data types, infrastructural types, and cardinalities defined in the FHIR resource. Therefore, FHIR structures are used as the basis for code generation, reporting, and user interface. And in turn, structure definitions are shared and published through repositories to enable interoperability between implementations.

When a structure definition contains a repeated element, you can chop the repeated element into sub-elements with different constraints on the sub-elements with the same meaning as the repeated element. In FHIR, this operation is known as "slicing" a list. It is common to split a list into sub-lists, each containing a single element, effectively placing constraints on each list element. For example, in a sliced resource structure, an *Observation* defines the *component* element containing a nested code and a value for the observations with several values. In which a resource instance of blood pressure measurement that includes one value for the systolic and one value for the diastolic, looks like this :

API Expressions 2.1 – Structure of a blood pressure monitoring resource in a JSON format.

```
{
"resourceType": "Observation",
"id": "blood-pressure",
"meta": {"profile":[
        "http://hl7.org/fhir/StructureDefinition/vitalsigns"]},
"subject.referece": "Patient/1186747",
"component":[
                {"Observation": "Systolic␣BP",
                        "name": "Systolic",
                        "coding": "LOINC␣8480-6",
                        "value.units": "mmHg",
                },
                {"Observation": "Diastolic␣BP",
                        "name": "Diastolic",
                        "coding": "LOINC␣8462-4",
                        "value.units": "mmHg",
                }]
}
```

**FHIR Operation or Qurey :**

The operation resource defines the input parameters that are passed to an executable operation or query. The FHIR servers execute the operations to determine system compatibility and thus enable dynamic generation of forms for sharing resources. Technically the operations are defined under a set of schemas like OpenAPI Specification, that formally represent the FHIR RDF compliance rules—sending a copy of the FHIR resource to another system via the FHIR API with JSON payload or other interoperability protocol [Solbrig 17].

Resource operation provides relative URLs to interact with the data. For example, blood pressure observation can be fetched by specifying the resource, observation identifier, and format, and the query returns an Atom feed, which includes all related resources. For example, in the case of a Blood Pressure Observation could be fetched as follows :

API Expressions 2.2 – Fetch a blood pressure monitoring resource in a JSON format.

```
GET https://hl7.org/Observation/blood-pressure?_format=json
```

The operation parameters are specified directly in a profile or resource structure by describing an operation's input and output parameters. In other words, the operation parameters can be defined using the *OperationDefinition.parameter* element or through a profile. If a profile is used, it must follow the information defined in the operation parameter element using *OperationDefinition.inputProfile* and *OperationDefinition.outputProfile*.

As an example, consider an operation that defines three parameters : the input that contains an integer parameter and a patient parameter ; and the output that includes the result parameter, is defined as follows :

API Expressions 2.3 – Example of defining an operation as a profile.

```
{
"meta": {
    "inputProfile":[
       "http://hl7.org/StructureDefinition/op.x.in.profile"],
    "outputProfile":[
       "http://hl7.org/StructureDefinition/op.x.out.profile"],
    }
}
```

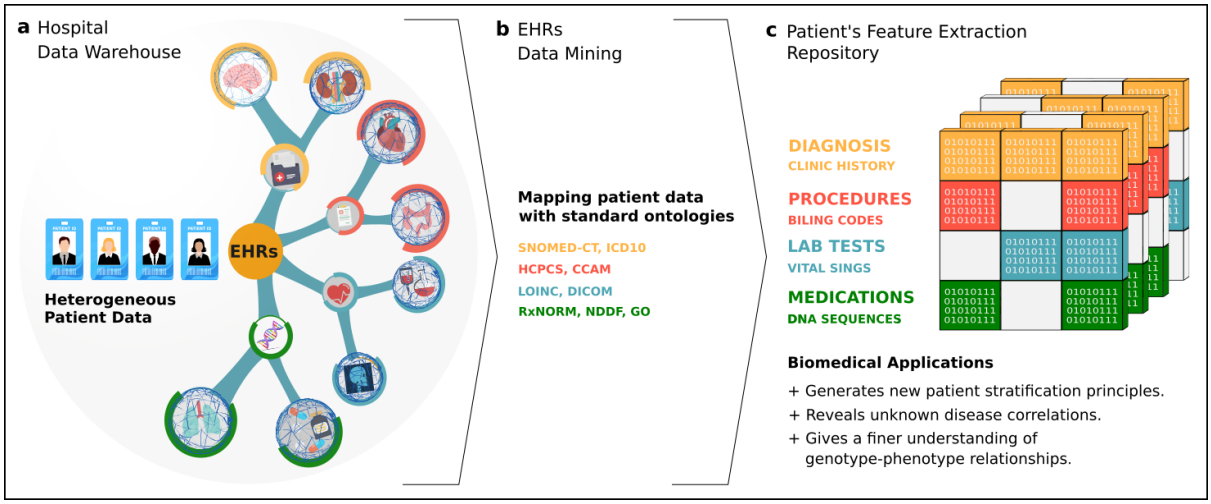



FIGURE 2.2 – EHR mining schema with the harmonization of clinical features and extracting phenotype representation of patients to feed biomedical applications.

### 2.1.4 Healthcare Data Mining

Given the increasing volume of data in healthcare systems, data mining is widely applied to extract clinical features characteristics of patients and is used in conjunction with deep learning models to develop clinical decision systems. This section synthesizes the stages for EHRs data mining, then harmonizes clinical features and extracts patient phenotype representation to feed it into biomedical applications. Figure 2.2 shows a pipeline based on three steps to harmonize heterogeneous and unstructured clinical data using standard ontologies to enable semantic interoperability of clinical features for developing scalable models between clinical centers.

**a.** The data warehouse generally includes as much information related to the patient as demographic data, medical notes, medications, vital signs, laboratory data, and others generated by one or more encounters in any care delivery setting.

**b.** The data mining process maps heterogeneous and unstructured data to standardize into a single format to represent the patient's state, including the structure and semantic levels of interoperability, such as standard ontologies like ICD10, SNOMED-CT, LOINC, RxNORM, and others.

**c.** There are several methods to derive and represent the clinical status of the patient, which is generally called the *High-throughput Phenotype (HtP)* and used deep learning models to generate biomedical applications as new principles of patient stratification, reveal correlations of unknown diseases, and provide a more accurate understanding of genotype-phenotype relationships. Using deep learning to develop risk model predictions will be described in more detail in the next chapter.

## 2.2 Case of tudy : Healthcare System in PACA

The objective of this study was to design a clinical risk prediction model pipeline based on deep learning methods to be integrated into a healthcare learning system, where the EHRs generated from patients could be continuously analyzed and modeled to create predictive medical tasks that would be transferred to patient care and optimize the functions of healthcare professionals.

In 2000, the French Ministry of Health [de la Sante 5], established the medicalization program of the information system (PMSI). They are designed primarily to know the medical activity of health facilities and to standardize the clinical documents. It houses applications from a hospital stay to follow-up medical visits, which should result in producing a standardized computerized record, called a standardized discharge summary (RSS), that includes administrative, demographic, medical, and supportive information.

The French Technical Agency for Hospitalization Information (ATIH) was created as a public administrative establishment of the State under the supervision of the ministers responsible for health, social affairs, and social security, to collect data from the health systems and manage medical classifications, as well as economic studies, costs, information of restitution and participate in the development of health nomenclatures. In 2019, the ATIH registered 12.9 million patients and 3,252 facilities from services of medicine, surgery, obstetrics, hospitalization, care, and rehabilitation follow-up that transmitted the RSS feed through PMSI. From this perspective, we collected many EHRs to build a clinical dataset, which was used during the development of this thesis and other related works. The clinical dataset was derivative from the hospital discharge data warehouse in the south region of France called *Provence-Alpes-Côte d'Azur (PACA)* through the *Medicalization Program of the Information System (PMSI)*[2] .

### 2.2.1 Medicalization Program of the Information System

The PMSI compiles all the information related to the hospital stay and condenses it to summarize standardized results (RSS). Which is fed by one or more medical units (RUM), depending on the number of medical units attended who cared for the patient during the stay in one or more hospitals. Likewise, the PSMI uses a classification called a group of homogeneous patients (GHM) to classify all the records produced in each hospital and delimits them into groups of coherent codes according to medical and cost terms [Chantry 12].

2. PMSI program of medicalization systems information in medical care and rehabilitation care operated by the French agency ATIH `https://www.atih.sante.fr/ssr/presentation`

The hospital discharge data warehouse is divided into four main areas of patient care, such as medicine, obstetrics, and odontology (MCO), follow-up and rehabilitation care (SSR), home hospitalization (HAD), and psychiatry. In 2019, they obtained a total of 1'162,900 in the flow of patients treated in 353 facilities, as shown in the following table 2.1.

TABLE 2.1 – Patient flow for PACA health system in 2019.

| | **Patients** | **Facilities** |
|---|---|---|
| **Medicine, obstetrics and odontology (MCO)**<br>This collection describes the medical activity of health facilities as well as the summary of outpatient activity, comprising administrative, demographic, medical and support information. | 1'024,700 | 123 |
| **Care and rehabilitation follow-up (SSR)**<br>This collection describes the medical activity of health facilities and quantifies hospital stays, which contains a summary of hospitalizations by sequences of weeks. | 96,100 | 153 |
| **Home hospitalization (HAD)**<br>This collection applies to all health establishments, public and private, with authorization for home hospitalization activity. | 6,800 | 23 |
| **Psychiatry**<br>This collection describes all the activity carried out for the benefit of patients by health establishments, in full or partial hospitalization as well as in outpatient. | 35,300 | 54 |

### 2.2.2 Care and Rehabilitation Follow-up Data Warehouse

The primary hospital discharge used during the development of this thesis was the PMSI care and rehabilitation follow-up data warehouse as a source to obtain the complete summary of the standardized results and derive the clinical dataset. Specifically, the SSR data warehouse is derived from the hospitalization and rehabilitation facilities. They are homogenizing in records with information related to the diagnosis, procedures, and rehabilitation of patient care treated during a week and with the possibility of tracking up to 52 records per year.

The SSR data covers 2006 to 2011, with an average of 680,626 records and 166,202 procedures per year, and the data corresponding to the MCO, has an average of 1'982,451 records and 5'113,458 procedures per year, as shown in the table 4.3. However, in this thesis, the MCO data is not used, although these have a greater flow of patients per year, it did not contain the necessary indexes to identify the follow-up visits or to link the patients that came from SSR facilities.

TABLE 2.2 – Summary of Patient Flows in PACA region.

| | In-Patients | | Out-Patients | |
|---|---|---|---|---|
| **Year** | **Records** | **Procedures** | **Records** | **Procedures** |
| 2006 | 660,743 | 109,605 | 1,864,122 | 5,606,190 |
| 2007 | 668,109 | 124,451 | 1,827,333 | 5,812,424 |
| 2008 | 673,676 | 139,234 | 1,868,364 | 6,250,494 |
| 2009 | 693,469 | 226,626 | 2,087,874 | 6,769,023 |
| 2010 | – | – | – | – |
| 2011 | 707,130 | 231,091 | 2,264,558 | 7,221,259 |

The main clinical variables were structured into ten different groups, which include clinical descriptors such as demographics, admission details, hospitalization details, physical dependence, cognitive dependence, rehabilitation time, comorbidities, morbidity, etiology, and procedures, for all patients who were treated in any of the hospitals in the PACA region or patients in this region who were treated in other hospitals from 2006 to 2011, as shown in the following Table 2.3.

TABLE 2.3 – Clinical descriptors to build the dataset.

| **Entity** | **Feature** | **Description** |
|---|---|---|
| Header | ID RSA | This is the identifier for the entire hospitalization based in standardized weekly summaries. |
| | ID hospital | Designate the number of the health facilitie in the national archive of social and health establishments. |
| Demographics | Age group | It is categorized into ranges like : $[0-6, 7-12, 13-17, 18-29, 30-59, 60-74]$ |
| | Sexe | It is coded as 1 male and 2 female. |
| Admission Details | Input mode | Mode of entry into the medical unit. Which can enter by mutation, final transfer, orovisional transfer or home. |
| | Input source | Specifics if the patient comes from a control unit, a hospitalization unit, intensive care unit or home hospitalization |
| | Previous state | The anticipation into the medical unit, during the week considered an indicator to order the stays of the same patient in an establishment. |
| | First week | The start dates of entry into the medical unit. |
| Hospitalization Details | Numdays week | This is the calendar week ID number. The first week of the year is the week that contains January 4 (ISO 8601 standard). |
| | Sequence number | Each calendar day of actual presence in hospitalization is coded "1" otherwise "0". |
| | Surgery time | For patients hospitalized after surgery, the date of the intervention is information from the standardized weekly summary. |

| Entity | Feature | Description |
|---|---|---|
| Physical dependence | Dressing | It is a quantitative measure of the ability to dress and undress above the waist, as well as to put on and take off an orthosis or prosthesis as the case may be. |
| | Displacement | This variable includes five actions : bed-chair-wheelchair transfers, transfers to the bathroom, transfers to the bathroom or shower, to locomotion and the use of stairs. |
| | Feeding | This variable includes three necessary actions : use of utensils, chew, swallow (swallow a bite or sip). |
| | Ccontinence | This variable includes two actions : control and hygiene of urination, control and hygiene of defecation. |
| | Wheelchair | A patient uses a wheelchair. He needed help getting from bed to chair (support and positioning by the caregiver). Once seated in his chair, he moves independently. |
| Cognitive dependence | Comportement | It includes a single action, social interaction, defined as the ability to get along with others and participate in social or therapeutic hospital situations to satisfy one's own needs. |
| | Communication | This variable includes two actions : understanding of verbal, visual or auditory communication ; clear expression of verbal and nonverbal language. |
| Rehabilitation time | Mechanical R. | The score for rehabilitation proccedures is calculated by adding the weightings of coded rehabilitation from the clinical proccedures, then dividing the result by the total number of days of presence during the week (Monday to Friday). This score is used quantitatively and in two classes (lower or higher than a threshold may vary according to medical-economic groups) |
| | Motorsensory R. | |
| | Neu. Psycho. R. | |
| | Cardiorespiratory R. | |
| | Nutritional R. | |
| | Uro. Sphincter R. | |
| | Kidneys R. | |
| | Electrical R. | |
| | Collective R. | |
| | Physiotherapy | |
| | Balneotherapy | |
| Associated diagnosis | $[DAs1, ..., DAs20]$ | Significant diagnostic association Is considered to be any health problem that coexists with the main morbidity that has led to effective management (research, treatment, etc.) |

| Entity | Feature | Description |
|---|---|---|
| Primary morbidity | Care purpose | The care purpose is usually an action that answers the following questions : What type of care did the patient mainly receive ? What was the essential nature of the medical and nursing care for this patient ? |
| | Morbidity | The main morbidity manifestation is the deterioration or functional or organic health problem on which the previous action is exercised, and which has mobilized most of the medical and nursing effort. |
| | Etiology | It is the etiology of the main morbidity, which describes the cause of a disease. |
| | Major Diag. cat. | It is a measure to classify each standardized weekly summaries in a major category. Dichotomous tests (yes-no) are performed successively on three main morbidity variables as care purpose, morbidity and etiological condition. |
| | Hom. Diag. Cat. | A second step consists in classifying each standardized weekly summaries in a nosological group (GN), most often describing the main pathology. |
| Clinical procedures | Procedures | Any clinical procedure performed during the week as part of the hospitalization and is coded under the CCAM codes. e.g. HHFA001 Appendectomy, of first quadrant. |
| Destination | Output mode | Exit mode of the medical unit. Which can leave due to mutation, hospital transfer, home transfer or the death of the patient. |
| | Destination | If the exit method requires it, in the event of exit by mutation or transfers |

## 2.3 Structural Equation Model from Quality Care to Risk Modeling

Since 2011, meta-analysis has demonstrated that healthcare information systems that have successfully adhered to clinical guidelines benefit the quality and efficiency in which healthcare organizations perform patient care [Buntin 11]. The mechanisms by which EHRs improve healthcare quality are reducing medication errors through a clinical decision system, improved clinical communication, improved information management, leading to better treatment decisions, and data sharing, which reduces information fragmentation.. [Atasoy 19]. Specifically, an observational study confirmed that the adoption of EHR improved the quality of care in medical centers for the treatment of critically ill patients. For this purpose, they performed a Cox proportional hazards regression analysis, where the leading indicators of quality of care were inpatient mortality, readmissions

within 14 days, and postoperative mortality at 48 hours [Lin 20].

In this sense, the standard Structural Equation Model (SEM) established to measure the quality of care in healthcare organizations is used to develop clinical risk prediction models, adding a new level of automation to empower clinicians and improve the quality of care. The SEM identifies and quantifies system and patient characteristics to timely and appropriately assign clinical procedures to obtain the best patient care outcomes. Figure 2.3 shows the SEM components for mapping the clinical structure to clinical outcomes ; in which the characteristics of health systems, patient characteristics, and social and family characteristics are standardized in CDA to build a repressive input vector to feed deep learning tasks to process clinical caregivers as diagnoses, procedures, and length of stay.

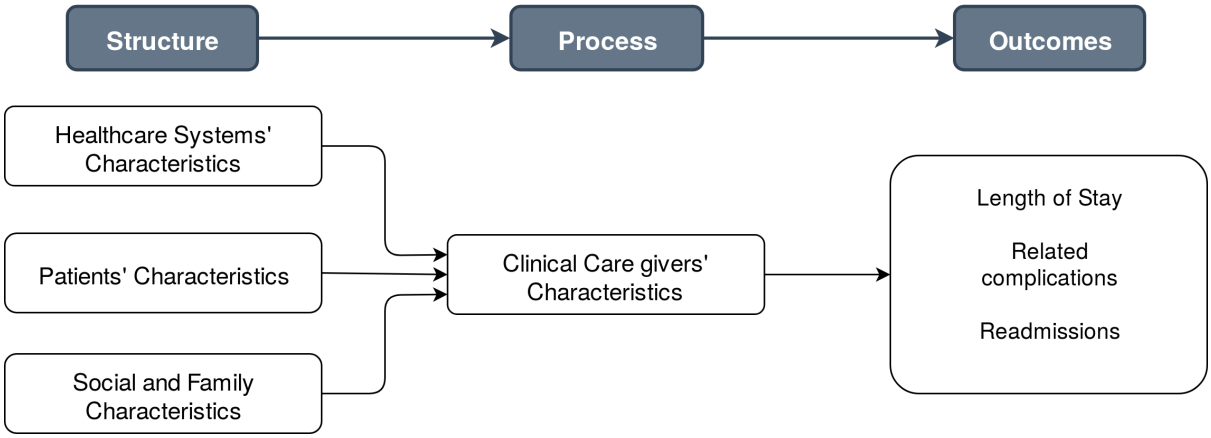



FIGURE 2.3 – Main entities to develop risk models for inpatient outcomes.

### 2.3.1 Structural Equation Model Applied to Risk Modeling in Paca Hospitals

The SEM has been shown to help clinicians better understand their patient populations' needs and improve health outcomes and reduce expenditures by targeting and tailoring care to high-need patients. In this study, we integrate the data-driven modeling with the SEM to digitize the patient information and the clinical care procedures to develop the clinical risk models.

The data-driven modeling focused on structuring the EHRs according to the clinical descriptors from the care and rehabilitation follow-up data warehouse of PACA healthcare systems. Then, a transformative library is developed to parser the various sources of EHR over the years to the clinical document architecture (FHIR) to allow semantic interoperability and allow the feeding of data into the clinical visualization API or to feed them into the clinical modeling tasks, which is described in detail in the following subsection.

The SEM approach was designed as a pipeline of deep learning tasks to map the primary care pathways according to the flow of patients from the follow-up care and rehabilitation in the PACA facilities. The conceptual scheme of the patient hospitalization process to track the medical history of individual patients and the second utilization of the EHRs to data-driven models for improving the accuracy of healthcare forecasting is presented in Figure 2.4. In which is presented the discharge flow ratio for all hospitalized patients during their first week in 2008. The first step is to extract the patient's features and embed them in a general representation of patients (phenotyping) using unsupervised learning methods. Thus, the embedded patient feature representations are fed into diverse deep learning tasks like health care purpose, medical procedures, length of stay, or destination.

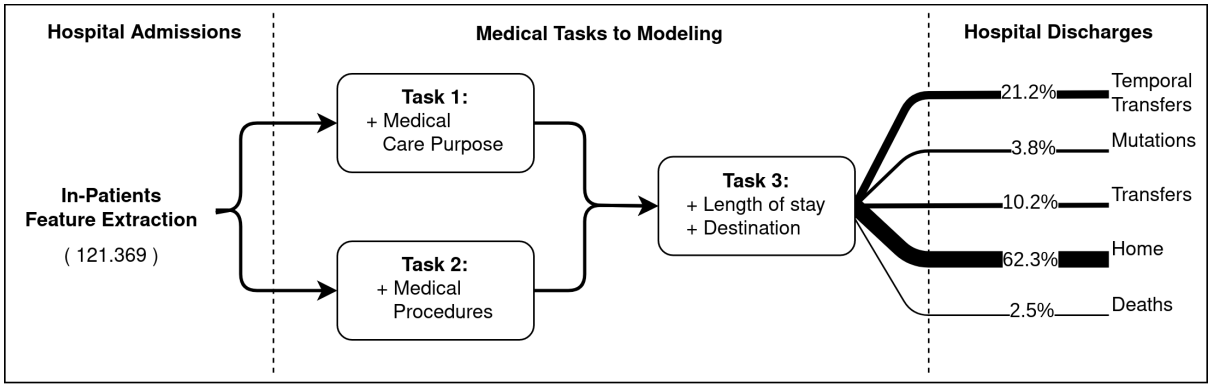



FIGURE 2.4 – Conceptual scheme of the discharge flow ratio for all hospitalized patients during their first week in 2008.

## 2.4 Development a FHIR transformer to standardize EHRs

The main objective here is to design and develop a framework that parses EHRs into the FHIR standard as the primary basis for developing medical studies using deep learning models and enabling the trained model's scalability among other institutions. The clinical document architecture is one of the leading data formats standardized in the HL7 FHIR specification and is being used by healthcare institutions showing robustness in its structure and agility to communicate information between devices and applications. However, this model is rarely used in research or studies using deep learning methods applied to medical research since this stage increases the complexity of the study.

Furthermore, MongoDB has proven to be a robust engine that offers excellent perfor-

mance, accessibility, and scalability for document non-relational database management in a wide variety of use cases and industries [Anusha 21]. MongoDB works with records as documents. The documents are stored in JSON, BSON, and XML formats, allowing data storage for various uses.

A MongoDB aggregation pipeline was developed for parsing and serializing a patient record into an FHIR document, called *ehr2fhir transformer*. The clinical document architecture was based on the clinical entities and features presented in Table 2.3. The ehr2fhir create an initial collection coming from the anonymization output summaries (RSA) obtained from the PACA PMSI data related to hospital admissions, ICD-10 diagnoses, disease grouping, procedures, and others.

The former aggregation is made to merge the RSA files with the procedures files to compose an object per each patient RSA data and concatenate all the operations performed on the same week.

The second aggregation is made to add the SSR fields. This aggregation translates the French clinical variables to English clinical variables according to the features described in the Table 2.3, as well as is performed the engineering rules to group and bucketize variables as the *age* which was used seven groups like $age\_groupe : [0-6, 7-12, 13-17, 18-29, 30-59, 60-74, 75+]$ and other variables as the rehabilitation times was a divide in buckets.

The third aggregation is made to build the clinical document architecture used during the experiments of this thesis. This aggregation defines the clinical document architecture divide in header and body according to the entities described in Table 4.3. Furthermore, it is a function to serialize each record to create a clinical document in JSON format. Finally, an example of the JSON CDA schema is presented :

API Expressions 2.4 – CDA schema of PMSI hospitalization data.

```
ssr2008_addfield.aggregate([
      {$addFields: {"x0_header": {
              "ID_RSA": "$idnum",
              "hospital": "$finess",
              "patient": "$clef",
              "patient_Rol": "Inpatient",
              "rsa_V": "$version",    }}},

      {$addFields: {"x1_demographics": {
              "age": "$age",
```

```
            "sexe": "$sexe",
            "age_group": "$age_group",
            "activity": "$type_activite",
            "postal_code": "$geograph",     }}},


    {$addFields: {"x2_admission_details": {
            "input_mode": "$mode_entree",
            "input_source": "$prov_entree",
            "previous_state": "$anteriorite",
            "first_week": "$semaine_debut",
            "month": "$Mois",
            "year": "$Annee",       }}},

    {$addFields: {"x3_hospitalization_details": {
            "numdays_week": "$Nbjour_sem",
            "numdays_weekend": "$Nbjour_we",
            "numdays_hospitalized": {
                    $toUpper: "$numdays_hosp"},
            "sequence_number": "$Num_seq",
            "surgery_time": "$date_chir_group",     }}},

    {$addFields: {"x4_physical_dependence": {
            "dressing": "$Dep_habillage",
            "displacement": "$Dep_deplacement",
            "feeding": "$Dep_alimentation",
            "continence": "$Dep_continence",
            "wheelchair": "$Fauteuil_roulant",      }}},

    {$addFields: {"x5_cognitive_dependence": {
            "comportement": "$Dep_comportement",
            "communication": "$Dep_relation",       }}},

    {$addFields: {"x6_rehabilitation_time": {
            "mechanical_rehab": "$Reeduc_meca_gr",
            "motorsensory_rehab": "$Reeduc_sensor_gr",
            "neupsycho_rehab": "$Reeduc_neuropsy_gr",
            "cardioresp_rehab": "$Reeduc_cardioresp_gr",
            "nutritional_rehab": "$Reeduc_nutri_gr",
            "urosphincter_rehab": "$Reeduc_urosph_gr",
            "kidneys_rehab": "$Readap_reins_gr",
            "electrical_equipment": "$Appareillage_gr",
            "collective-rehab": "$Reeduc_collective_gr",
            "bilans": "$Bilans_gr",
            "physiotherapy": "$PhysioT_gr",
            "balneotherapy": "$BalneoT_gr", }}},

    {$addFields: {"x7_associated_diagnosis": [
```

```
            "$DAs1", "$DAs2", "$DAs3", "$DAs4", "$DAs5",
            "$DAs6", "$DAs7", "$DAs8", "$DAs9", "$DAs10",
            "$DAs11", "$DAs12", "$DAs13", "$DAs14",
            "$DAs15", "$DAs16", "$DAs17", "$DAs18",
            "$DAs19", "$DAs20",]    }},

        {$addFields: {"x8_primary_morbidity": {
            "care_purpose": "$Fin_princ_PC",
            "morbidity": "$morbid_princ",
            "etiology": "$affect_etiol",
            "major_diagnostic_categories": "$cmc_rhs",
            "homos_diagnostic_categories": "$ghj_rhs",      }}},

        {$addFields: {"x9_clinical_procedures": {
            "procedures": "$actes.CodActe",
            "nb_actes": "$Nb_actes":,       }}},

        {$addFields: {"x10_destination": {
            "last_week": "$semaine_fin",
            "output_mode": "$mode_sortie",
            "destination": "$destination",  }}},
            ])
```

### 2.4.1 Patient Flow in PACA Hospitals

Once the clinical data coming from the hospitals of the PACA region, France, had been serialized and structured with FHIR standards. The next objective is to analyze the patient flow of hospitalizations to quantify the number of patients admitted to the hospital and map the trajectories until discharge.

This first analysis allows us to see the density of the population entering the hospitalization stage, to know the mode of entry and the mode of exit. For example, if a patient comes from a hospitalization transfer from one hospital to another ; if the transfer is from one unit to another in the same hospital ; And mapping the output of these patients transferred to understand where the patient load is and identified the tasks that clinicians need the AI to assist the decision-making.

To perform this exploration, we selected data concerning the first week of hospitalization for all hospitalized patients in 2008. As a result, Figure 2.5 illustrates the flow of 121,369 patients admitted to PACA health services from the general community, transferred from other hospitals, and discharges into the health services systems for patient care, threat, and discharge. It is observed that the highest flow of hospital admissions comes from the emergency department, with a rate of 72,422 patients in the first week

of hospitalization in 2018 for the entire PACA region. 62.4% of patients in the transfer admission modality are discharged home, followed by 14% of patients in the transfer admission modality are discharged to another inpatient unit such as ICU and others. And the majority number of decedents per week comes from the transfer admission modality at 2.67%, with 1,931 decedents out of the total number of transfers processed in the first week of hospitalization in 2018 for the entire PACA region.

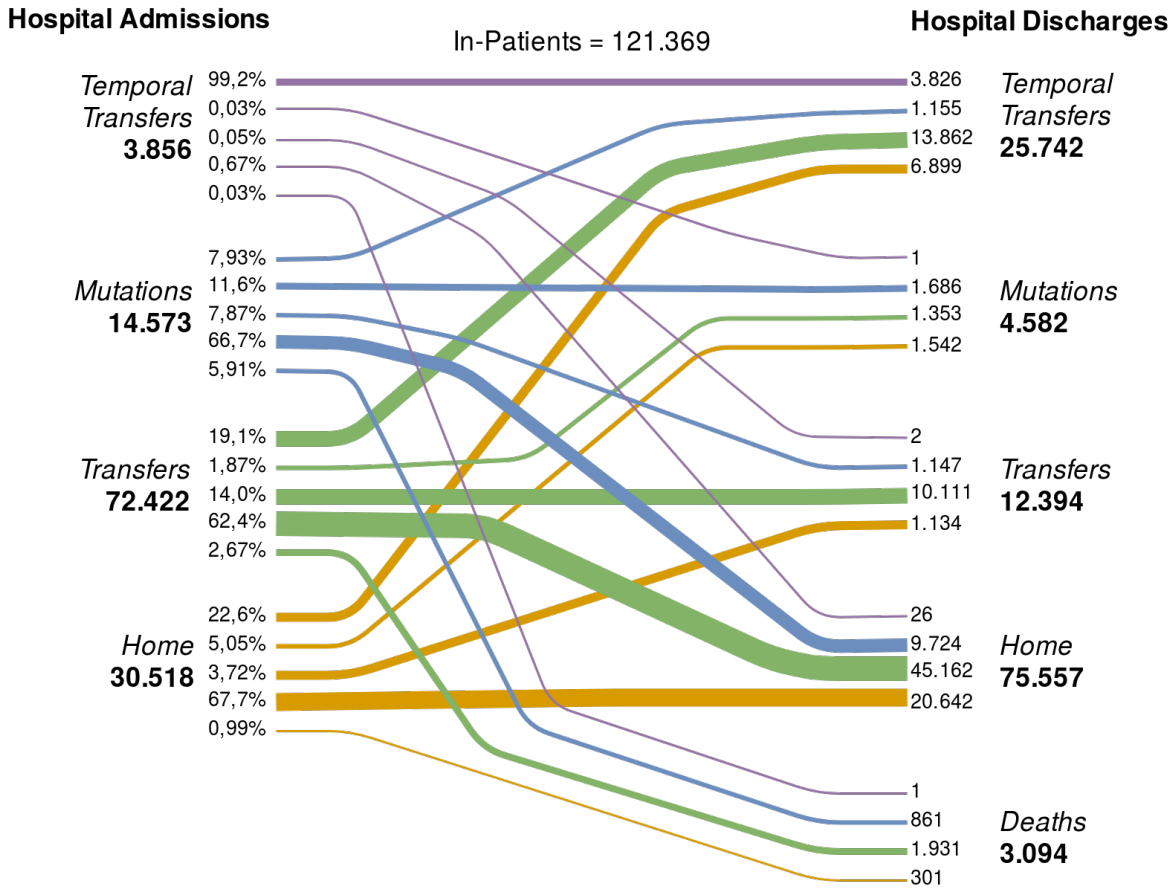



FIGURE 2.5 – The inpatients flow ratio in PACA health services for all hospitalized patients during their first week in 2008.

### 2.4.2 Medical Care Purpose Classification for Inpatients

The PMSI system has two ways to track the patient flow and correlate those with the medical units. Once using the medical diagnoses with the ICD-10 codes, the other uses the equivalent diagnosis-related groups organized in hierarchization levels encoded in a Nosological Group (GHJ).

The medical diagnosis contains the care purpose of each hospitalization—allowing the correlation between the patient flow with the GHJ and thus obtaining a better quantification and reporting of the workload and the need for IA tools. Being the nosological

group codes complementary information of the primary morbidity of the patient, which is composed of :

- **Care purpose :** Healthcare realization of the main load taking and high-level expertise to attend to the patient.
- **Primary morbidity :** Primary morbid manifestation such as illness, symptom, trauma, injury, poisoning, or situation at hospital admission.
- **Etiology :** Cause or origin to cause disease.

Therefore, to enrich the patient flow analysis is extracted the high-level care purpose group, obtaining 14 labels-categories to classify the medical care of patients hospitalized, as shown in Table 2.4.

TABLE 2.4 – Medical Target 1 : Care purpose classification task to assist the inpatient encounter.

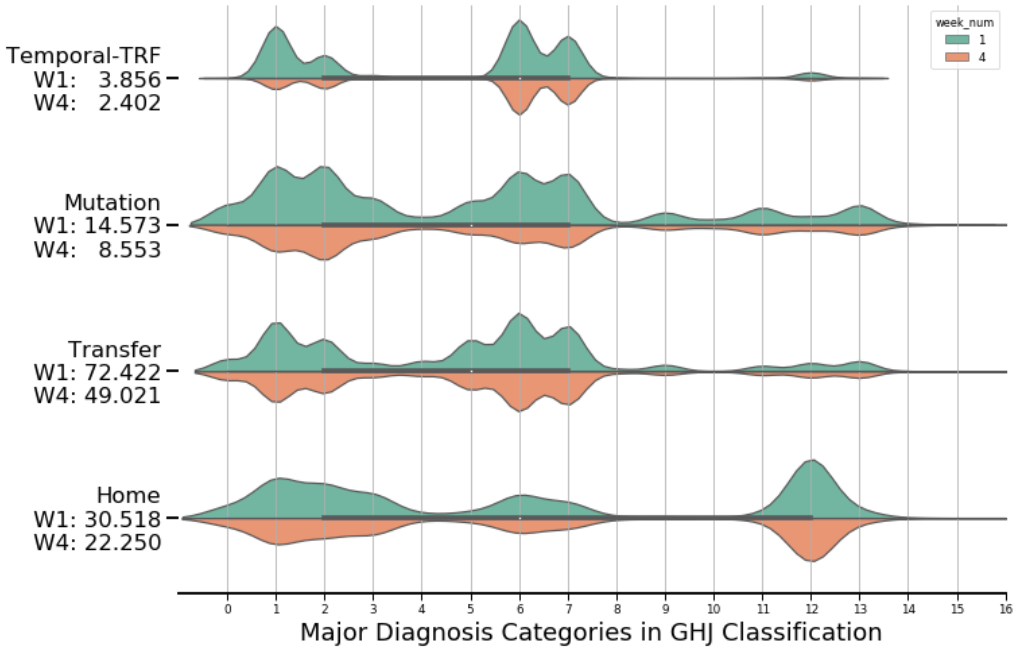
| Class ID | Labels Description | In-Patients Number |
|---|---|---|
| 0 | Other Situations | 5,271 |
| 1 | Proceedings of Medical Cardiovascular / Respiratory Care | 21,684 |
| 2 | Proceedings of Circulatory System Disorders | 15,342 |
| 3 | Proceedings of Neuro-Muscular Medical Care | 6,310 |
| 4 | Proceedings of Medical Care Mental Health | 3,448 |
| 5 | Proceedings Sensory and Skin Medical Care | 9,679 |
| 6 | Proceedings of Rheumatics / Orthopedic Medical Care | 21,247 |
| 7 | Proceedings of Post-Traumatic Medical Care | 16,594 |
| 8 | Proceedings of Medical Amputations | 875 |
| 9 | Palliative Care | 2,426 |
| 10 | Placement Expectation | 359 |
| 11 | Rehabilitation | 2,643 |
| 12 | Proceedings of Nutritional Medical Care | 10,799 |
| 13 | No grouping | 4,692 |
| | **Total** | **12,1369** |

The second analysis of the patient flow is divided by two charts, the first concerning inpatient admissions versus major diagnosis in the nosological group classification. And the other concerning the discharge from hospitalization versus major diagnosis in the nosological group classification.

The figure shows the multimodal distribution of the patient flow with admissions and discharge mapped across a high level of care purpose groups or GHJ. The green distributions represent all patients hospitalized during their first week in 2008, while the orange distributions represent all patients hospitalized during their fourth week in 2008. The categories with the highest flow of inpatient hospitalization at admission are 1 : Cardiovascular and respiratory medical care procedures, 2 : Circular system disorders

procedures, 3 : Neuro-muscular medical care procedures, 5 : Sensory and skin medical care procedures, 6 : Rheumatic and orthopedic medical care procedures, 7 : Post-traumatic medical care procedures, 12 : Nutritional, medical care procedures.

(a) Hospitalization admissions input sources.




(b) Hospitalization discharge output sources.

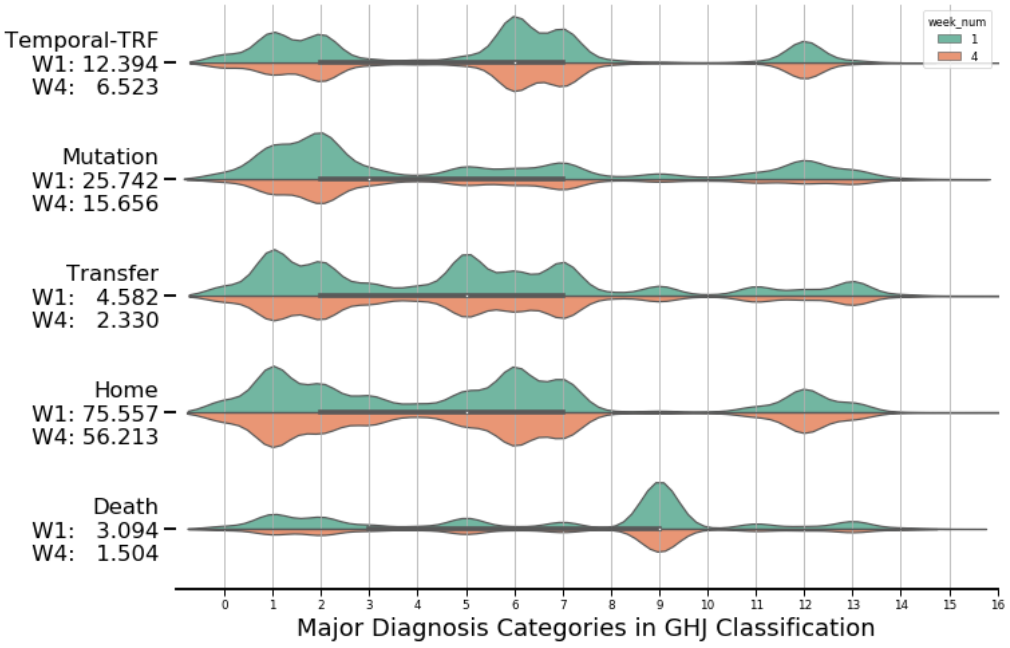



FIGURE 2.6 – Multimodal distribution chart of the patient flow correlates admission and discharging hospitalizations with the major diagnosis categories using the nosological group classification.

## 2.5 Summary

The second use of EHRs allows the development of new DL models that empower researchers and healthcare professionals with the possibility of finding new answers for the treatment, prevention, and understanding of disease and quality of life. While the interplay of big data analytics and the development of artificial intelligence applications in automating routine tasks will free up the workload of healthcare professionals, allowing them to focus on more personalized and in-depth analysis for each patient.

Despite the enormous amount of work hospitals and healthcare centers require to upgrade their health information systems from management information to new integrated medical intelligence applications and enable scalability of the model across different organizations and build learning healthcare systems.

We believe that the basis for building continuous learning healthcare systems is to add a layer of standardization of EHRs through a standard medical model, such as those provided by the FHIR standard. Applying the standardization to the EHR-driven modeling allows DL models to be scaled from hospital A to hospital B, providing an opportune scenario to evaluate the robustness and generalizability of DL models.

The next chapter focuses on machine-and-deep learning approaches to enable high-throughput phenotype discovery. It seeks to transform the general representational space of patients and group patients by groups that share similarities from vital signs, comorbidities, the purpose of care to enable the training of supervised learning algorithms that allow the medical task to begin.

# Chapitre 3

# AI-based to Automate Clinical Risk Prediction Workflows

EHRs are growing and flowing more and more through healthcare systems, collecting longitudinal data on patient and physician experience. Raising a common challenge in healthcare systems, as it contains a wealth of historical patient information, but clinicians are short on time and often do not have the right tools to synthesize this information [Choi 16a]. In response, data-driven approaches have been applied to develop clinical decision support systems based on secondary use of EHRs for modeling different clinical tasks or events. In general, the design of these systems involves integrating diverse machine learning training processes, in which input features are mapped to obtain a set of intermediate features that are then mapped to the output classes. [Shamout 20]. However, modeling an accurate representation of the patient's state in an embedded feature representation is a task that involves solving several challenges in terms of algorithmic complexity, such as the number of hours required for model convergence and the considerable memory size needed to process these models.

This chapter describes the first component of the DiagnoseNET workflow to automate extracting and representing the patient phenotype for use in mapping different medical targets. This component focuses on implementing unsupervised neural networks to derive general latent representations fed by supervised learning algorithms, while computing resources such as convergence time, GPU memory, and power consumption are monitored to improve their utilization. The data manager abstracts the entire workflow orchestration stage by stage, from the transformation of clinical features to the latent representation, dividing the data for training, validation, and testing, and using micro-batch management to train deep distributed neural networks.

DiagnoseNET workflow automates the training process of the machine learning pi-

peline, allowing each stage to have enough dynamism to configure different parameters according to the problem or to search for the best model to specify and explain the medical task. The main stages of the DiagnoseNET workflow are as follows :

1. In the first stage, it previously processes the EHRs to build a sparse binary representation of the patient and thus describe the patient's condition according to a standardized semantic structure that guarantees the interoperability of the model to be built.
2. Then, in the second stage, we seek to transform the representation space to obtain latent representations that embed the clusters of patients with similar clinical descriptors and reduce the dimensionality of the data than the original space where the data lives.
3. In the third stage, both representations could be used as input to supervised learning models to model clinical risk predictions according to the medical target.

## 3.1 EHRs derivation to represent the patient's condition

Representing patient conditions is an essential step in developing clinical decision support systems. Much of the performance of clinical predictive models depends on the representation of data features and how underlying bias is handled in the EHRs, which often have noisy data. For which a method called High-throughput Phenotyping (HtP) has recently been introduced to extract impartially and automatically select informative features, which can be comparable to those chosen by experts in terms of machine learning tasks [Yu 15]. In addition, the HtP is enhanced by the use of representation learning algorithms applying different embedding approaches, such as standard dimensionality reduction techniques, distributed representations used in language modeling, the use of embedding layers as part of a larger model, or through the latent space of autoencoders and their variants [Shamout 20].

### 3.1.1 High-throughput Phenotyping from EHRs

Patient phenotypes are the basis for clinical and genetic studies of disease risk, and outcomes [Zhang 19]. Generally, a clinical phenotype refers to the disease or condition representing an observable trait of the subject. A retrospective analysis of patient datasets is used to derive and identify correlations of *clinical features* with host-response patterns and clinical outcomes to determine whether a patient with a particular set of clinical characteristics meets that definition [Yu 15]. These clinical features are usually composed

of heterogeneous data from diagnoses, procedures, medications, vital signs, laboratory results, and clinical notes coded and recorded according to each healthcare system.

High-throughput phenotyping is a mechanism that aims to derive millions of fine-grained phenotypes, speeding up the process of trait selection for phenotyping algorithms with minimal human intervention. Algorithms that identify the desired phenotype could be constructed in quite different ways ; One of the methods used is the vector representation in which, for each medical objective, a correlation matrix is constructed between the patients and the features of the medical data [Wang 14]. The generation of the different vectors usually takes significant time. Other possibilities are nonnegative matrix factorization, and nonnegative tensor factorization to extract the phenotypes as a set of matrices, tensor candidates showing the patients clusters linked to specific medical characteristics and their date [Ho 14, Perros 17, Perros 18]. Other approaches use nonnegative vectors to embed the clinical codes and use word representations such as skip-gram or Glove to generate the corresponding visit representation [Choi 16b].

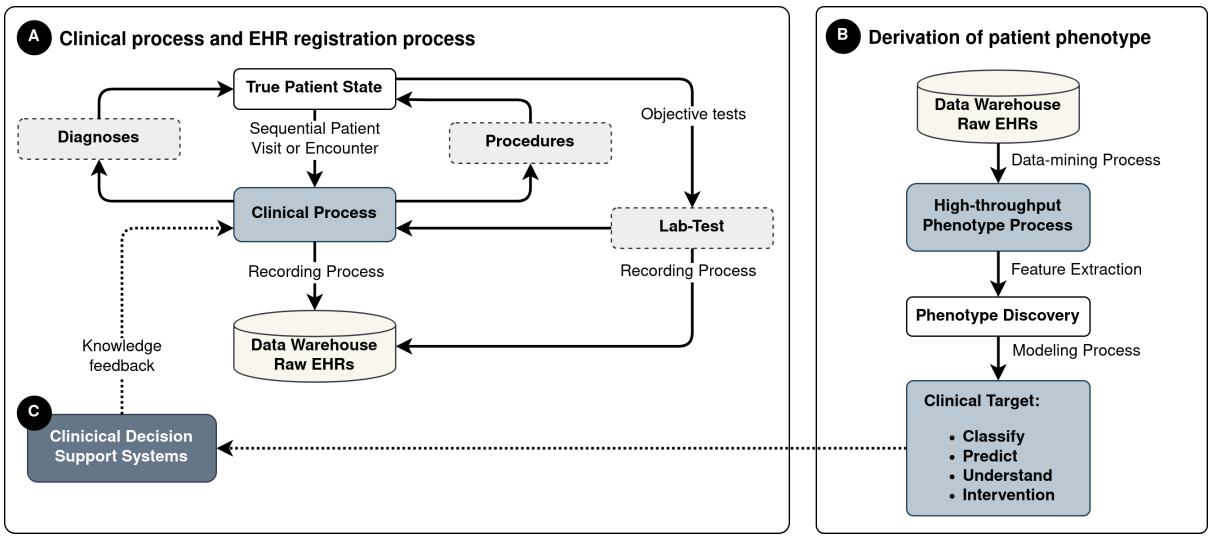



FIGURE 3.1 – A conceptual diagram to encounter the clinical process in healthcare facilities and derive patient phenotypes.

The figure 3.1 shows the conceptual diagram depicting the clinical and EHR recording process and its derivation of patient phenotype to feed this knowledge back through clinical decision support systems. The box **A** represents the feedback loops in the EHR, where the patient's condition varies and determines not only the value of the measurements in the recording but also the type and timing of the measurements. While box **B** represents the process of high-throughput phenotyping and its subsequent use for clinical modeling. And box **C** represents the system that is designed to integrate these phenotypes and clinical models to highlight individual patient features and assist clinicians in

decision making. However, EHRs are not a direct reflection of the patient's proper condition, but rather a review of the recording process inherent in the healthcare system with noise and feedback loops [Hripcsak 13]. Depending on the healthcare system, these data pose modeling challenges associated with missing data, mismatches, sparsity, and noise at the medical record level. These challenges are being solved by applying deep learning algorithms to healthcare research, described in the following sections.

### 3.1.2 Representation Learning to Derive Patient Phenotypes

In recent decades, manual phenotyping methods were traditionally designed to identify patterns in patients with a single target disease, with domain experts overseeing the definitions of trait scales for a particular medical target and typically working with well-defined. Still, minuscule sets of data [str 10, Kennedy E.H. 13]. Meanwhile, the new generation of artificial intelligence medical systems is expanding medical capabilities from tracking patients' health to predicting early risk detection and providing new patterns to help personalize treatments. Therefore, representing the state of patients through the phenotyping process is a crucial step in the development of these systems.

Following the significant success of representation learning for image, text, and audio with rich high-dimensional data, there has been increasing transfer of knowledge gained in these domains to train deep neural networks in medical settings [Bengio 11, Mikolov 13, Srivastava 15]. Healthcare researchers and computer scientists have used unsupervised methods to train deep neural networks to discover latent representations of patient phenotype.

They effectively built predictive models for a broad set of medical conditions. The strength of these unsupervised methods is that they do not require domain experts at the level of the feature extraction process and solve some issues of missing data, sparsity, and log-level noise.

In general, the derivation of a patient's phenotype involves a mathematical transformation of high-dimensional features into a lower-dimensional space. In the deep learning process, the representation is learned using two neural network models with multiple hidden units for encoding and decoding the clinical features. First, an encoder network or an embedding function $f\theta(.)$ is a generative model, which takes the essential clinical features $X$ as input data to transform them into a new latent space $Z$ as the output. Second, the representation $Z$ is fed into the decoder network $g\theta'(.)$, which is used to reconstruct the input data $X$ and also to predict the clinical label $y_n$ given the input latent representation $Z$, as shown in the Figure 3.2. Meanwhile, the choice of a model usually depends on the

performance of subsequent learning tasks.

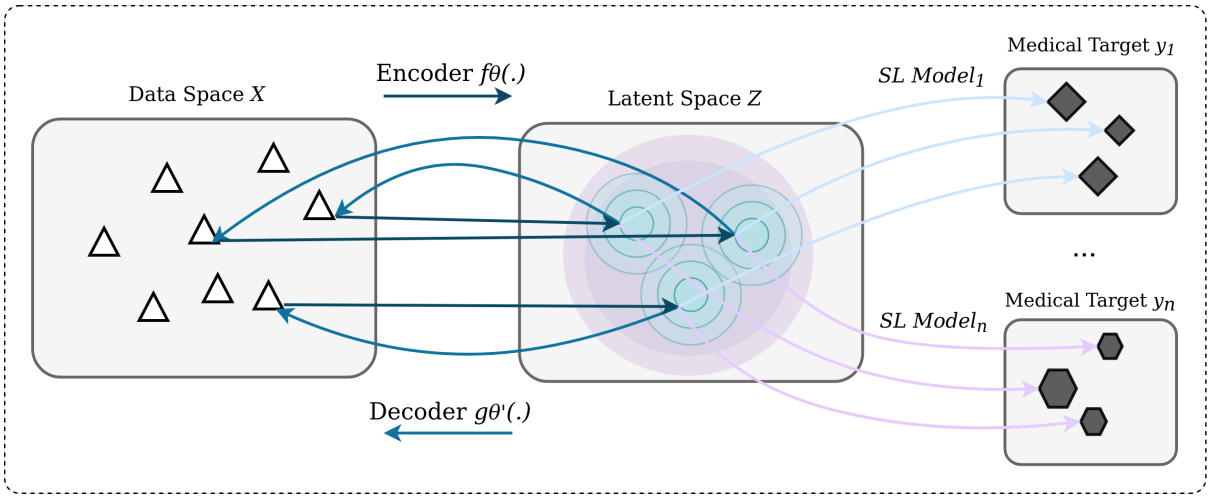



FIGURE 3.2 – An unsupervised encoder network maps clinical features into a new latent space to represent the patient's phenotype.

## 3.2 DiagnoseNET to Automate Clinical Risk Prediction Workflow

In this thesis, we extend the deep patient approach [Miotto 16], in which all clinical descriptors are grouped inpatient vectors, and each patient can be described by a high-dimensional vector or by a sequence of vectors computed in a predefined temporal window.

Figure 3.5 shows the process of using the DiagnoseNET workflow to build a clinical-specific risk prediction model from EHRs. It highlights the different steps needed to construct the phenotype, aiming to create an equivalent but more miniature representation for more effective clinical or medico-administrative prediction. The first stage focuses on mining the heterogeneous clinical data source to build a patient document-term matrix. The second stage uses unsupervised representation learning methods for mapping the patient's document-term matrix to a new latent space with a lower-dimensional data representation. In the third stage, the medical task classes are matched with the new patient representations to train a classification model using machine learning algorithms or deep neural networks. Finally, the quality of the new patient representation consists of evaluating the trained model on a new test set by comparing the accuracy performance against a second classification model trained using the patient's document-term matrix directly. A detailed description of the three stages is provided in the following paragraphs.

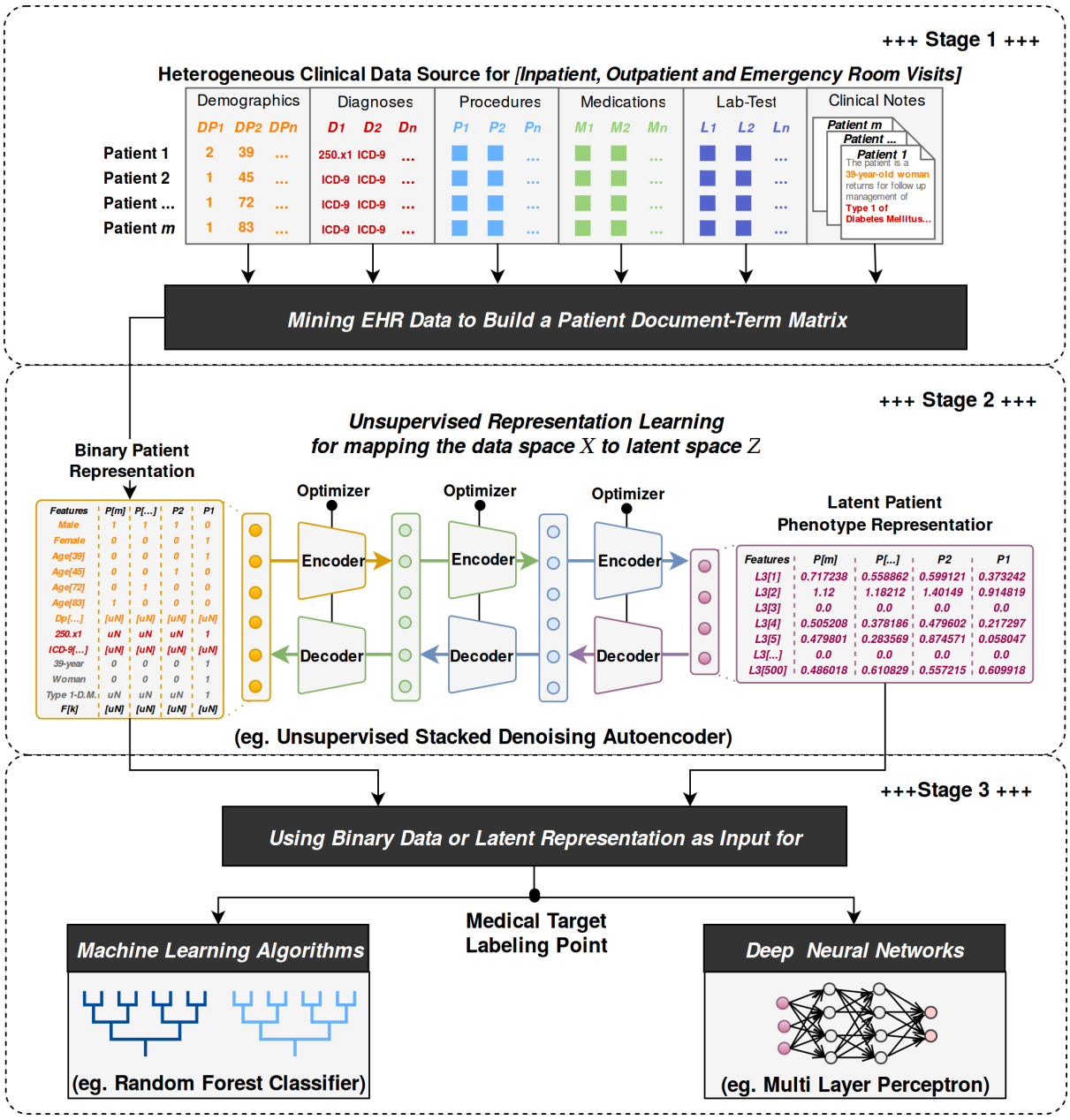



FIGURE 3.3 – Workflow scheme to automate patient phenotype extractions and apply them to predict different medical targets.

### 3.2.1 DiagnoseNET : Patient Feature Composition

The growing health-wide research is mainly due to the clinical dataset being composed of secondary usage of patient records collected in admission, and hospital process [Jensen Peter B. 12]. However, there is no standard library for mining and composing representative patient vectors [Evans 16]. The main objective of the *patient feature composition* is to develop a set of functions for high-throughput phenotyping that provides a set of data mining tasks and tools for processing patient records. To this end, a data mining library was built that transforms the EHRs into a clinical document architecture, using the standard called fast healthcare interoperability resources (FHIR) to reproduce high-

throughput phenotyping from dataset A to a dataset B.

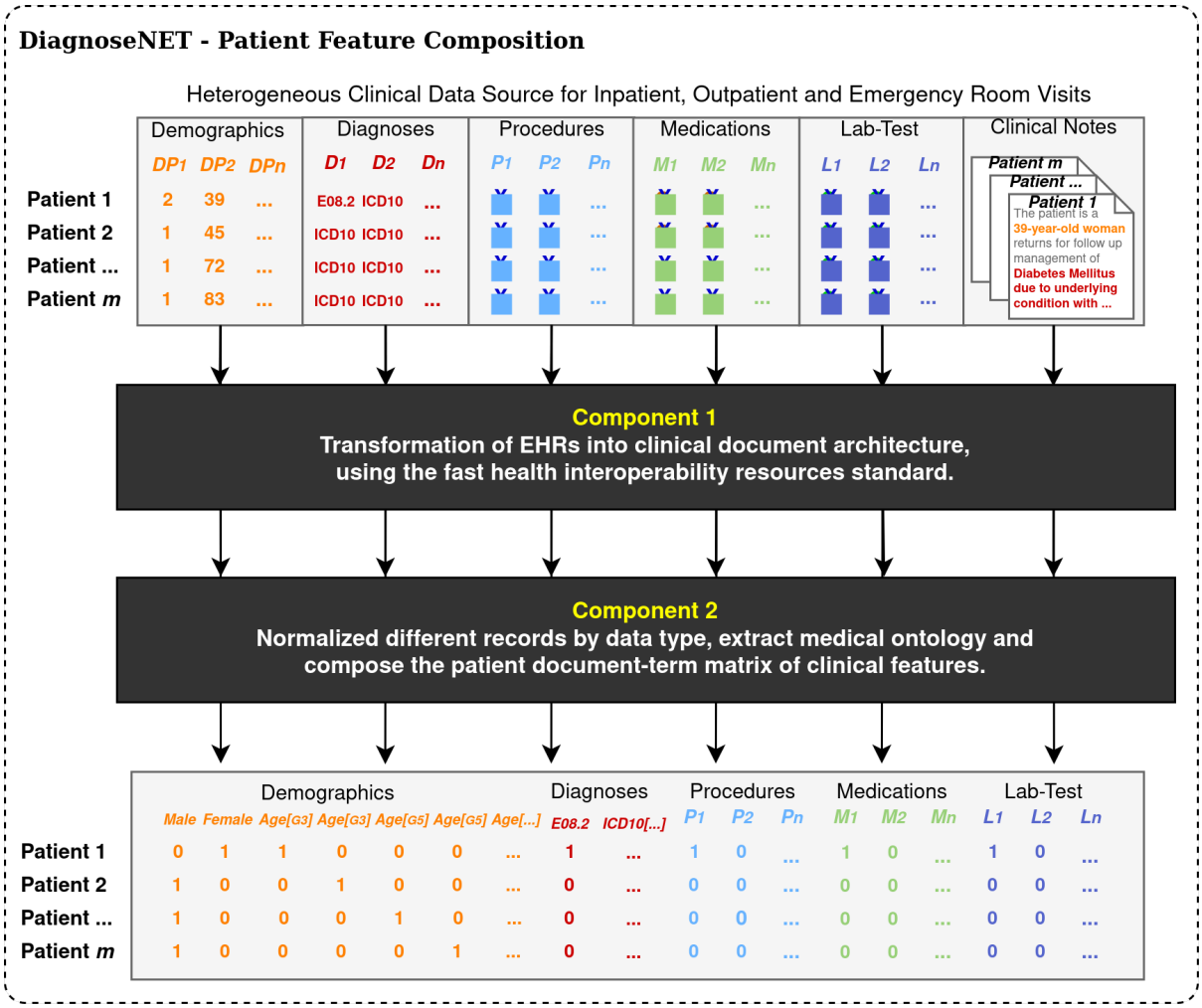



FIGURE 3.4 – DiagnoseNET data mining library transforms EHRs into a clinical document architecture according to the FHIR standard and then is derived the clinical data of the patients in a document-term matrix from clinical features.

The following two components that achieve this objective were developed, as shown in Figure 3.4. The first component, preprocess a heterogeneous clinical data source (e.g., EHR) through a chain of aggregations to structure the clinical descriptors by entities, using the fast health care interoperability FHIR standard discussed in the previous chapter, and its repository is available at https://github.com/jagh/ehr2fhir.

The second component focuses on standardizing the different records by data type to compose a matrix of patient terms whose columns are the clinical terms, i.e., a sequence of demographic information followed by all the ICD10 codes for medical diagnoses. The main step is to extract a vocabulary (medical ontology) from the EHR or import a ready-made one to store clinical terms and serve as the ontological basis for patient feature composition. Whose main functions are described in the following points, and its repository is

available at https://github.com/IADBproject/patient-feature-composition.

— **Clinical Document Architecture (CDA)** The CDA identifies the syntax for clinical records exchange between the system PMSI and DiagnoseNET through the new versions generated by the agency ATIH. The *CDA* schema consists of a header and body :
  — Header : Includes patient information, author, creation date, document type, provider, etc.
  — Body : Includes clinical details, demographic data, diagnosis, procedures, admission details, etc.

— **Vocabulary Composition** Enables dynamic or custom vocabulary for selecting and crafting the right set of corresponding patient attributes by medical entities.

— **Features Composition** Serialize each patient record and get the CDA object for processing all patient attributes in a record object.

— **Label Composition** This function gets the medical target selected from the CDA schema to build a one-hot or vector representation.

— **Binary Patient Representation** To build the initial patient representation is mapping the values of the features from the record object with the corresponding terms in each feature vocabulary to generate a binary corpus using Term-document Matrix.

### 3.2.2 DiagnoseNET : Unsupervised Learning

Once the patient document-term matrix has been constructed, it is time to perform a representation model learning to transform the input data into a new latent space with lower dimension and discover the clusters that best represent the phenotype of the patients for input into a clinical-specific risk prediction model. It is the most sensitive stage, both because of the variability of the clinical characteristics that can be selected as an input and because of the intense finetuning of the hyperparameters to choose the model that generates a quality patient phenotype to predict medical tasks.

Consequently, it was necessary to build a data manager that automatically stores the clinical features selected to compose the patient's document-term matrix in a sandbox to generate and store the subsequent transformations hierarchically according to each generated model hyperparameters applied to the selected deep neural network. As

shown in Figure 1, the data manager is incorporated in each of the stages, from the data mining process, the unsupervised learning process to the execution of different supervised algorithms to perform the medical tasks. For the first case, the data manager is in charge of structuring the storage from the data staging according to the FHIR standard, followed by the construction or import of the vocabularies (medical ontology) to build the patient document-term matrix according to the selected clinical features called *binary_representation*. Then, the data manager divides the data into training, validation, and test so that the network feeds them into the unsupervised neural network and stores the new latent representation according to the hyperparameter used. Finally, the data manager can use the binary representation or a different latent representation to perform the subsequent supervised learning tasks and store the learning metrics and computational metrics for each new training process.

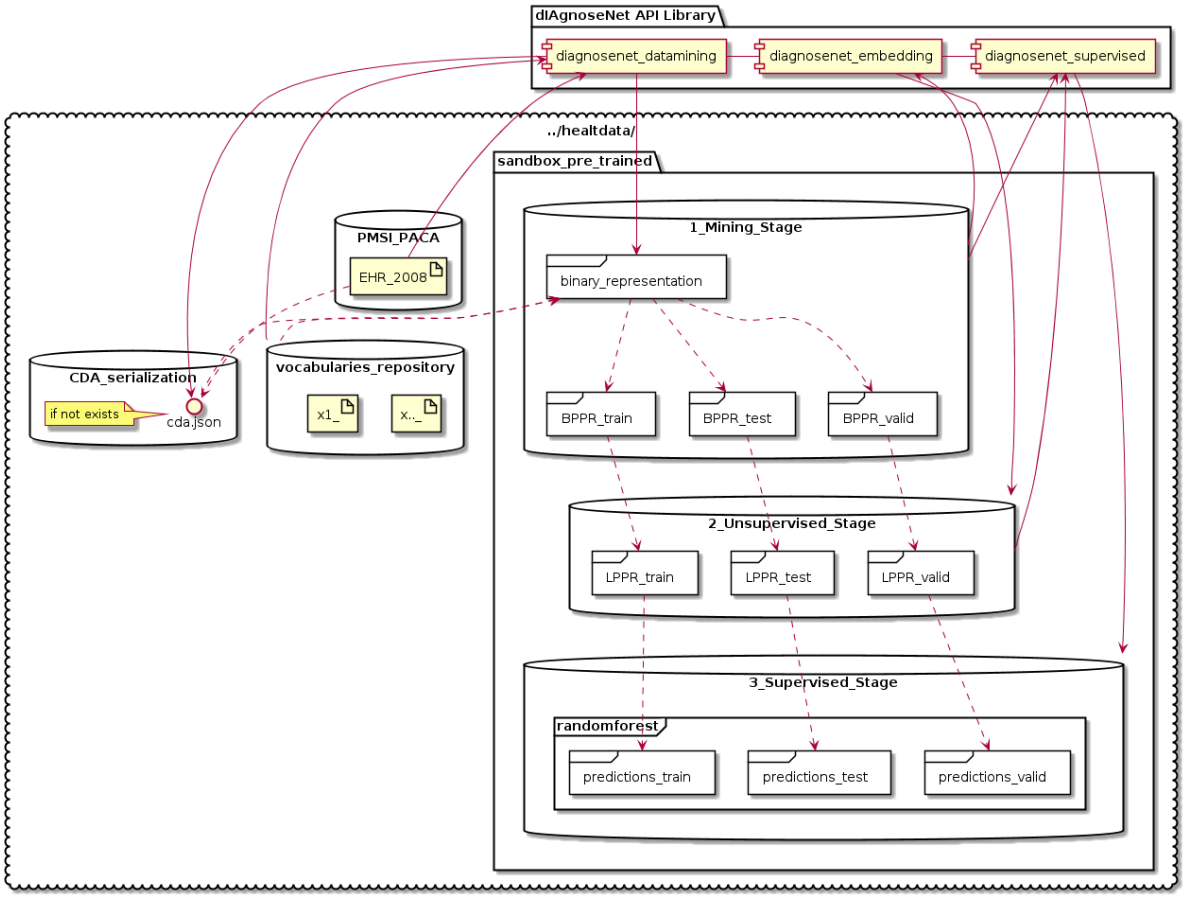



FIGURE 3.5 – DiagnoseNET datamanager to automate the s.

On the other hand, a variation of autoencoder called *Unsupervised Denoising Stacked Autoencoders* was selected as the neural network architecture to reduce the dimension of data and discover the new patient representation. Since training these neural networks is time and resource-consuming, it was necessary to develop a parallelizable GPU-based framework to train the neural networks with the adaptation into the NVIDIA Jetson

TX2 cards. Whose model and parallel transformation will be described in the following subsections.

### Unsupervised Stacked Denoising Autoencoders

The representation learning model is built from the unsupervised denoising stacked autoencoders (USDA), which combine an end-to-end generative autoencoder as a weight generator to pre-train a network of encoders layers that generate the latent representation. The objective of the USDA model is to generate representations that are sufficiently robust to the introduction of noise. For this, the noise process is introduced in the first layer employing Gaussian additive noise that allows the random assignment of the subset of inputs to 0, with a certain probability [Vincent 10].

Usually, the model is trained with unlabeled images, whose input images are used to compare the reconstruction image generated from the input image with noise. The USDA model comprises a stack generative autoencoder and an encoder network that generates the latent representation to feed the classification learning model. Each latent layer of the autoencoder has several neurons equal to that of the corresponding layer of the coding network to which the weights are transferred. At each step of the training process of the USDA network, the weights of the stacked autoencoders and the associated lower-dimensional latent representation are obtained simultaneously, as shown in Figure 3.6.

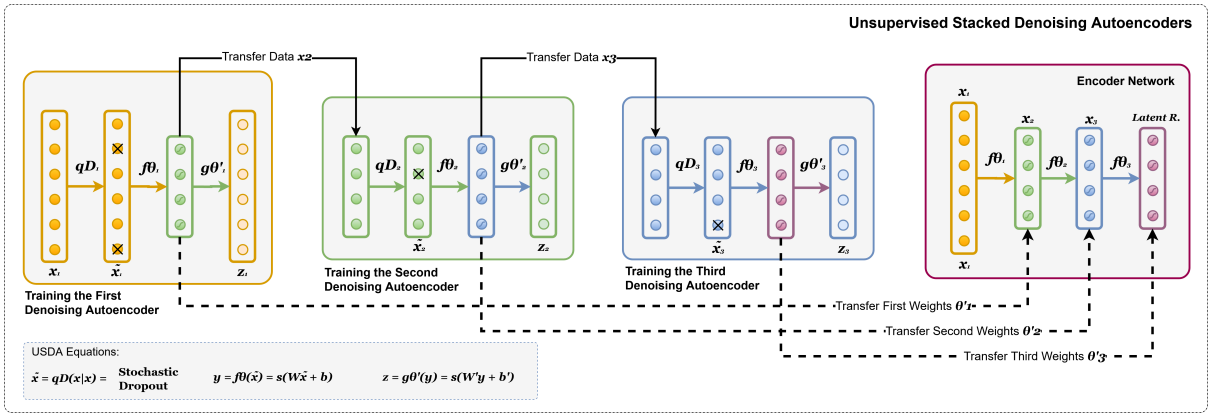



FIGURE 3.6 – An unsupervised encoder network maps the binary patient representation $x$ into a new space, obtaining the latent patient phenotype representation $z$.

The USDA network was composed as a deterministic mapping from the cleaning of the partially corrupted input $\tilde{x}$ *(denoising)* to obtain a hidden features representation $y = f\theta(\tilde{x})$ by layer. Therefore, each stacked layer is independently trained to reconstruct a clean input $x$ from a corrupted version of it $z = g\theta'(y)$, this approach was introduced by [Vincent 10]. Previously each encoder was pre-trained to get a semantics representation parameters $\theta'$ by denoising autoencoder which was trained before, to obtain a robust

representation $y = f\theta(\tilde{x})$ from a corrupted input $\tilde{x}$. The following steps described the USDA implementation :

— First is applied dropout to corrupting the initial input $x$ into $\tilde{x}$ the stochastic mapping $x \sim qD(x|x)$.

— Second, the corrupted input is mapped as traditional autoencoders to get a hidden representation $y = f\theta(\tilde{x}) = s(W\tilde{x} + b)$.

— Third, a schematic representation is reconstructed from a hidden space $z = g\theta'(y) = s(W'y + b')$.

— Four, once the parameters $\theta \wedge \theta'$ are trained to minimize the average reconstruction error over the training set, to have $z$ as close as possible to the uncorrupted input $x$.

— Finally is shared the new semantic representation parameters $\theta'$ to next layer as new initial input $x_2$ and corrupting it into $\tilde{x_2}$ by stochastic mapping $x_2 \sim qD(x_2|x_2)$ and repeat steps.

**Distributed Approach for Training UDSA Networks**

Unsupervised learning algorithms are an emerging approach in which the search for a robust and generalizable architecture is a common argument in state-of-the-art. Meanwhile, unsupervised methods have shown promising results in reducing the dimensionality of clinical features and clustering of subjects. However, unsupervised learning models such as USDA are computationally complex since this method involves one optimizer per block or stacked layer. Additionally, unsupervised neural networks require a much more extensive training dataset than supervised neural networks, and since this is the main task to derive the phenotype representation of patients to then train various clinical risk models using supervised.

Thus, a distributed approach and batch training overcomes the limitations of the training speed when the model does not fit in the GPU memory. In this section, we extend the data parallelism and distributed deep learning process proposed by [Dean 12b]. Furthermore, we developed an approach for training unsupervised denoising autoencoders using a mini-cluster of Nvidia Jetson TX2, adding a high level of task-based programming model to speed up the tasks when independent tasks are executed concurrently. The figure 3.7 sketch the distributed implementation to train the USDA network using the DiagnoseNET framework. The part **A.** shows the task parallelism for training the USDA

network and the part **B.** shows the data dependencies and the tasks that are processed concurrently for training the USDA network.

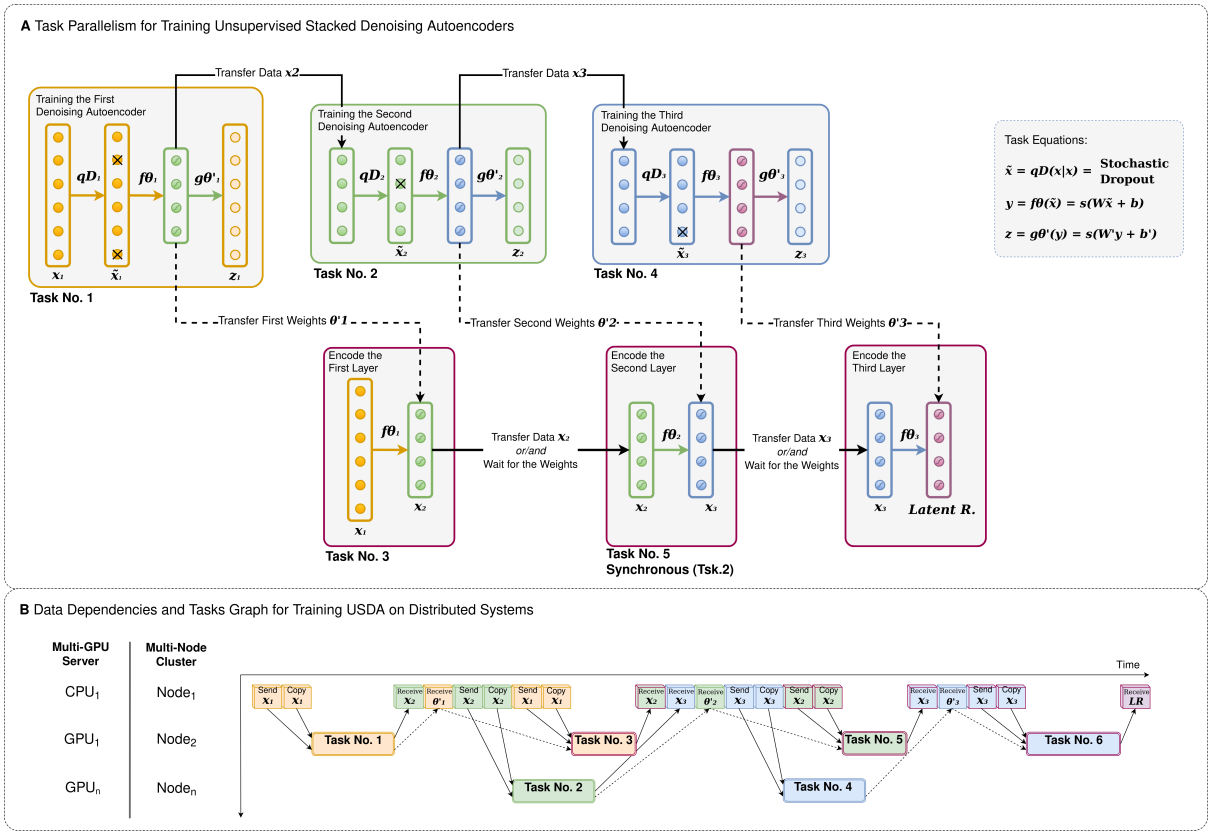



FIGURE 3.7 – Illustrative schema of the parallel and distributed processing train unsupervised stacked denoising autoencoders implemented to obtain the patient phenotype representation.

We reconstructed a three-block USDA network with TensorFlow-1.5 to extract and share the weights to initialize an encoder network to obtain the latent representation that compresses the input clinical data and represents the patients' state.

**Distributed cluster setup :** Distributed cluster setup aims to identify each node and define each node's job name and task index.

— The first step is to build a distributed session and let all the nodes be ready to train the deep neural network.

— The second step is to declare the number of layers and pre-trained the denoising autoencoder for processing the last encoder network.

— The third step specifies the host IPs and the role of each machine as a parameter server (ps) or as a worker. In which is define the machine role setting the input flags like number of ps, number of workers, job name, and task index.

**USDA training and distributed processing :** Declare the number of layers and pre-trained the denoising autoencoder for processing the last Encoder Network.

*The parameter server node* will wait until all workers are connected. Once all the workers are synchronized, build a distributed session that produces a queue and dequeue to summarize the gradients computed by each worker and send the new recapitulate gradients to each worker.

*The worker node* first connects to the distributed session using a device replication approach. It then builds the neural network graph in memory using a placeholder to store the training data set. Similarly, a placeholder is created to make the neural network graph and keep the data validation set. Then, each worker enters the neural network using a distributed session through a supervisor method. It allows to host the previously defined network graph and creates checkpoints to restore from a checkpoint when an error occurs. Additionally, in the supervisor, you can define rules to end the training, resulting from penalties for learning or epoch convergence. Finally, each worker loads and reads the mini-batch files assigned to initialize the training process, and the weights with the highest accuracy are saved for the validation set.

### 3.2.3 DiagnoseNET : Supervised Learning

It is well known that the overall performance of machine learning algorithms to convergence time and accuracy generally depends on the data representations. For this reason, the result of the unsupervised model obtained in the previous step can be used as input for a standard supervised machine learning algorithm [Bengio 14]. Thus, this approach allows using the latent or vector representations obtained from the training of unsupervised neural networks as input data in algorithms such as Random Forest and other neural networks, the multilayer perceptron, to predict the final medical task. In the following section, we describe those two approaches implemented to compare the prediction results, using the latent representation of the patients and the initial clinical data of the patients.

#### Random Forest

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for random forests converges as the number of trees becomes large, as well as the classification error depends on the strength of the individual trees in the forest and the correlation between them [Breiman 01].

The random forest algorithm is an ensemble method that grows multiple decision trees at training time. Then, at test time, a majority vote is performed on the predictions of the individual trees.

The algorithm is an extension of bagging, where each tree is grown using a bootstrap sample of the original training data. Each tree is grown using a bootstrap sample of the initial training data in bagging. In random forests, each tree is grown using a bootstrap sample of the original training data, with replacement [Breiman 96].

For example, in a forest of trees, each tree is different to classify a data set :
— Each tree is grown by randomly selecting a sample of the data and then increasing the tree using that sample.
— The trees are grown until they are as big as possible.
— The trees are allowed to vote on the best classification.
— The classification with the most votes is the final classification.

**Multilayer Perceptron Network**

The multilayer perceptron (MLP) is a feed-forward neural network. It consists of three layers : the input, output, and hidden layers. The input layer receives the input data to be processed. The output layer produced a label vector containing the probabilities corresponding to tasks such as classification and prediction. An arbitrary number of hidden layers is placed between the input and output layers to transform the data space and reduce dimensionality [Abirami 20]. Therefore, the MLP transforms the input high-dimensional patient's representations to the desired dimension of the label projections as an output of the medical task.

Every node in the multi-layer perception uses an activation function such as a sigmoid, a rectified linear unit (ReLU), and others. Therefore, each neuron takes all the input values and convert them to number between 0 and 1 using the activation function. The computation taking place at the output of each hidden layer as follows : $h_i = f(\sum_{j=1}^{n} w_{ij}x_j + b_{ij})$ ; where $x_j$ : is the output of the previous layer ; $w_{ij}$ : is the weight value associated with $x_j$,$b_{i,j}$ is the bias associated with $x_j$ ; $n$ : is the number of neurons in the previous layer and $f$ : is the activation function.

The general network architecture comprises fully-connected layers ; each neuron is connected to all previous layer neurons building a stacked neural network and then a softmax layer on top. Usually, an MLP network requires a finetuning process, where the primer hyperparameters to search are related to the number of units per layer, the num-

ber of layers, the batch size, the loss function, optimization algorithm such as stochastic gradient descent (SDG), a method for stochastic optimization (ADAM) [Kingma 15] and others, which hyperparameters affect the computational cost directly.

## 3.3 Case Study : Hospitalized Patients in PACA

A case study uses a clinical dataset sourced from the EHRs of hospitals in the PACA region, structured and serialized from the FHIR MongoDB database. The clinical data consists of high-level data entities related to the patient features like : demographic data, admission details, hospitalization details, physical dependency, cognitive dependency, rehabilitation time, and associated diagnoses to represent the patient status and predict primary morbidity as a deep learning task, whose labels are described in the next section.

Figure 3.8 shows the pipeline for extracting patient features. First, the patient records are serialized in CDA-JSON format, as described previously. Second, all features are converted into the term-document matrix to represent each patient feature in a vector representation. Finally, the dataset is divided into 85% for training, 5% for validation, and 10% for testing.

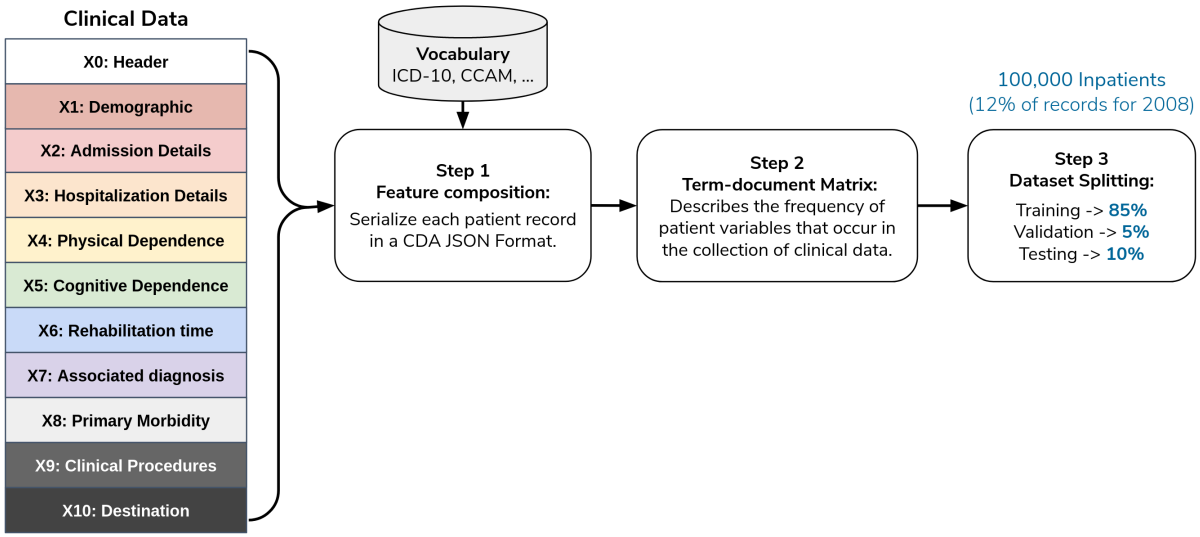



FIGURE 3.8 – PMSI clinical dataset used for patient feature-composition.

### 3.3.1 Medical Target : Medical Care Purpose Classification

The first medical objective used for this work is to classify the primary purpose of hospital care as a Major Clinical Category (MCC) as the clinical label to train the deep learning and machine learning algorithms, which groups the classification of ICD-10 codes at a high level. This clinical label allows answering questions such as What type of care

did the patient benefit from, or What was the essential nature of this patient's medical and nursing care ?

These codes are used in French hospitals and are stored throughout the PMSI system to assign ICD-10 codes to the "purpose of hospital care." Table 1 shows two examples of hospitalization. Patient one is hospitalized under an unspecified coma with an etiology of the Nontraumatic intracerebral hemorrhage ; therefore, the care purpose is an encounter for palliative care, and the clinical label will be palliative care. While patient two is hospitalized for significant morbidity related to neoplastic (malignant) related fatigue with an etiology of Maligna neoplasm of the rectum ; however, the patient was encountered for other specified aftercare purposes, and the clinical label is other disorders.

TABLE 3.1 – Example of diagnosis-related group hierarchization to select the main clinical category as clinical labels to train the algorithms and link them to patient care.

| | Diagnosis-related Group | ICD-10 | Definition |
|---|---|---|---|
| **Patient 1** | Morbidity Principal | R402 | Unspecified coma |
| | Etiology | I619 | Nontraumatic intracerebral hemorrhage, unspecified |
| | Care Purpose | Z515 | Encounter for palliative care |
| *Label used* | Clinical Major Category | 20 | Palliative care |
| **Patient 2** | Morbidity Principal | R530 | Neoplastic (malignant) relate fatigue |
| | Etiology | C20 | Malignant neoplasm of rectum |
| | Care Purpose | Z518 | Encounter for other specified aftercare |
| *Label used* | Clinical Major Category | 60 | Other disorders |

## 3.4 Experiments and Results

In this chapter, we have performed three groups of experiments. The first set of experimentation is focused on identifying the clinical features that benefit the performance accuracy to predict the medical target, comparing four clusters of patients' clinical characteristics to represent and find patients' phenotypes through unsupervised neural networks and then classifying the purpose of care as a supervised learning task and medical objective.

In the second set of experiments, we analyzed the behavior of computational resources by training USDAs with different batch sizes to examine the relationship between a network's convergence time, energy consumption, and ability to translate a patient's phenotype into a smaller latent space.

Finally, the third set of the experiment focused on characterizing the workload during the execution of a DNN network running on a variable number of Jetson TX2 according to different batch sizes. To estimate the efficiency (accuracy and energy consumption), we measure loss, accuracy, time, and number of gradient updates per epoch. Thus, we record the energy consumption, GPU SM frequency, GPU memory frequency and stipulate the minimum value of the loss as the convergence point to stop the training process.

### 3.4.1 Feature Assessment to Compose the Patient's Phenotype

Driving EHRs to build a general patient phenotype representation presents critical challenges in big data, deep learning, and parallel processing as the high energy cost for training models until finding the optimal model generalization accuracy. The general dataset is composed of the diagnosis-related group represented in object form ; such as ICD-10 codes, CCAM codes, and other codes established by the agency ATIH and generated by the system PMSI for the Intensive Care Unit (ICU) and Clinical operations (MCO) with activities of hospitals in PACA and the activities of residents PACA hospitalize in another region.

As a case of study, we have used an Intensive Care Unit (ICU) dataset with an average of 785,801 Inpatients records by year to build a general-purpose inpatient phenotype representation for available applicative medical targets, such as :

1. Predict the 'Major Clinical Category' from inpatients features recorded in the process 1, 2, and 3, presented in the scheme.
2. Predict the Clinical Procedures PMSI Data recorded in the process 1, 2, 3 and the Primary Morbidity.
3. Predict the 'Clinical Procedures' from inpatients features recorded in the process 1, 2, 3 and the Primary Morbidity, presented in the scheme.
4. Predict the 'Inpatient Destination' (home, transfer, death) and length of hospitalization stay from inpatients features recorded in the process 1, 2, 3, Primary Morbidity and Clinical Procedures, presented in the scheme.

#### DiagnoseNet Data-mining

It is a feature extraction API that uses a dynamic features composition with their respective vocabulary to build a binary patient phenotype representation in a 'document-term sparse matrix' from the ICU dataset, as shown in the figure below.

We present four ways to derive a BPPR, selecting different Inpatient features grouped by high-level entities as was shown at ICU data collection scheme (i.e., $X1\_Demographics$ : $sexe : [male, female], age : [..., 64, 65, 66, ...], ...$). As a result, we got different BPPR in the length of features and disk size. As an experiment are using 12% of ICU data by one year :

TABLE 3.2 – Feature Assessment to Compose the Patient's Phenotype.

| **Feature Composition** | **Num. Features** | **Disk (size)** | **Exe. Time** |
|---|---|---|---|
| $[X1, X2, X3, X4, X5, X6, X7]$ | 11094 | 3.2GB | 1.86 mins |
| $[X1, X2, X3, X4, X5, X6, X7, X8.3]$ | 14515 | 4.1GB | 2.63 mins |
| $[X1, X2, X3, X4, X5, X7]$ | 8041 | 2.3GB | 1.48 mins |
| $[X1, X2, X3, X4, X5, X7, X8.3]$ | 11462 | 3.3GB | 2.18 mins |

**DiagnoseNet Unsupervised Embedding :**

This API pre-training three Denoising Autoencoder (DAE) to get each latent representation and send them as input to the next DAE. Then, the Encoder Network uses a clean BPPR as input for mapping with the previously trained weights to obtain a new Latent Patient Phenotype Representation (LPPR), as shown in the figure below.

We use two groups of layer dimensions whit the same hyperparameters (i.e., batch size : 32, num. epoch : 10, . . . ) to change the space and reduce the length and disk size of the previous BPPR. As an experiment, each previous BPPR was divided into nine batch files :

TABLE 3.3 – Feature Assessment to Compose the Patient's Phenotype.

| **Num. Features** | **Disk (size)** | **USDA Dim.** | **Encoded** | **Exe. Time** |
|---|---|---|---|---|
| 11094 | 3.2GB | $[2000, 1000, 500]$ | 255MB | 39 mins |
| 11094 | 3.2GB | $[500, 200, 100]$ | 49MB | 24 mins |
| 14515 | 4.1GB | $[2000, 1000, 500]$ | 248MB | 51 mins |
| 14515 | 4.1GB | $[500, 200, 100]$ | 50MB | 41 mins |
| 8041 | 2.3GB | $[2000, 1000, 500]$ | 255MB | 26 mins |
| 8041 | 2.3GB | $[500, 200, 100]$ | 49MB | 18 mins |
| 11462 | 3.3GB | $[2000, 1000, 500]$ | 255MB | 38 mins |
| 11462 | 3.3GB | $[500, 200, 100]$ | 51GB | 30 mins |

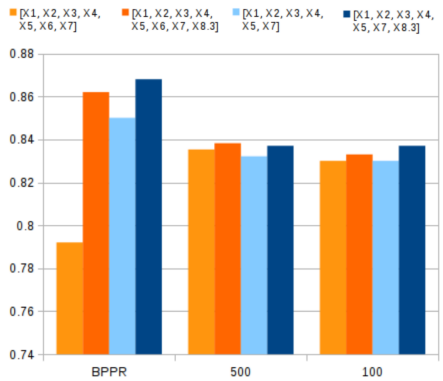
**DiagnoseNet Supervised Learning :**

We implemented Random Forest Classifier (RFC) to predict the Clinical Major Category (CMC) labels presented in the first medical target. As an experiment, we set the same RFC parameters (i.e., $estimators : 100, max\_features : sqrt, ...$) to compare the precision (AUC-ROC) between each previous BPPR and their two LPPR, as shown in following Table.

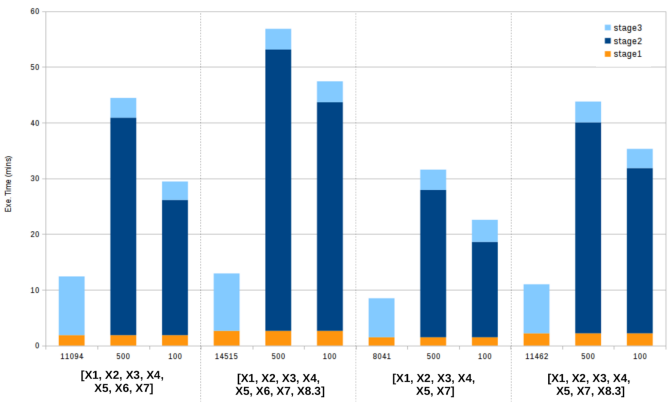
TABLE 3.4 – Feature Assessment to Compose the Patient's Phenotype.

| Feature Composition | Num. Features | AUC-ROC | Exe. Time |
|---|---|---|---|
| $[X1, X2, X3, X4, X5, X6, X7]$ | 11094 | 0.79 | 10.53 mins |
| $[X1, X2, X3, X4, X5, X6, X7]$ | 500 | 0.84 | 3.55 mins |
| $[X1, X2, X3, X4, X5, X6, X7]$ | 100 | 0.83 | 3.31 mins |
| $[X1, X2, X3, X4, X5, X6, X7, X8.3]$ | 14515 | 0.86 | 10.29 mins |
| $[X1, X2, X3, X4, X5, X6, X7, X8.3]$ | 500 | 0.84 | 3.72 mins |
| $[X1, X2, X3, X4, X5, X6, X7, X8.3]$ | 100 | 0.84 | 3.74 mins |
| $[X1, X2, X3, X4, X5, X7]$ | 8041 | 0.85 | 7.45 mins |
| $[X1, X2, X3, X4, X5, X7]$ | 500 | 0.83 | 3.66 mins |
| $[X1, X2, X3, X4, X5, X7]$ | 100 | 0.83 | 3.74 mins |
| $[X1, X2, X3, X4, X5, X7, X8.3]$ | 11462 | 0.87 | 8.80 mins |
| $[X1, X2, X3, X4, X5, X7, X8.3]$ | 500 | 0.84 | 3.73 mins |
| $[X1, X2, X3, X4, X5, X7, X8.3]$ | 100 | 0.84 | 3.44 mins |

FIGURE 3.9 – Results to claasify the medical care purpose; and execution times of the experiments.




(a) AUC-ROC to classify the medical care purpose.




(b) Execution time to perform the three stages.

As a result, the embedded representation had similar accuracy to the model with high-dimensional features, with an AUC of 0.83 versus an AUC of 0.86.4 for classifying the

first medical target. The unsupervised runtime has the most extended runtime among the three stages ; however, the classification time was shortest for phenotype group three [X1, X2, X3, X4, X5, X7], As shown in the Figure 3.9. In this sense, the following experiments will focus on reducing the execution time and analyzing the impact of unsupervised training and energy consumption.

**Energy efficiency analysis for mapping USDA on a CPU-GPU**

The USDA network's workload focuses on the latent representation data transfer and writing produced by each denoising autoencoder. Therefore to perform the first analysis of power consumption and workload characterization processing on one CPU-GPU environment, we used a submodule of DiagnoseNet called enerGyPU monitor for recording in runtime energy factors of the GPU, such as Streaming Multiprocessor Clock-frequency, Memory Clock-frequency, Memory Usage, and Power Consumption.

This DNN consists of a 3-layer denoising autoencoder with 2000, 1000, and 500 neurons in each layer, respectively, with other hyper-parameters : Relu as activation function, binary cross-entropy as loss function, and stochastic gradient descent as optimizer. The input data comprises 84.999 samples that represented 85% of the binary patient phenotype representation. we experiment on three input data mini-batch fragmentation strategies to process the DiagnoseNet Unsupervised Embedding stage.

Strategy 1 : Using all of the BPPR samples (84.999) for getting the LPPR. In this case, we can observe that the initial cost to move all the input samples from host memory to GPU memory represents 22% of the execution time. After the first 8 minutes, the first denoising autoencoder, in which the first output latent representation is encoded, is processed. The second and third autoencoder layers are even more efficient.

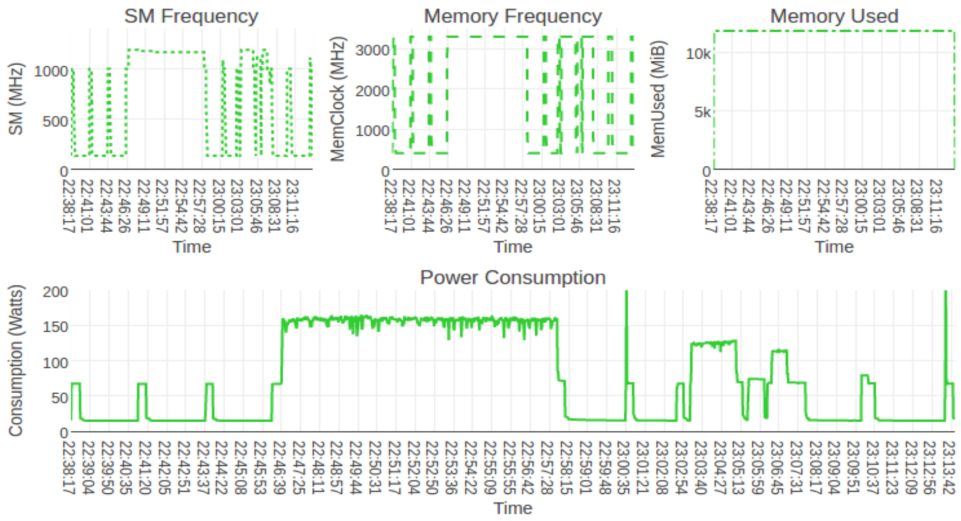



FIGURE 3.10 – Strategy 1 had 35.46 minutes, 74.75 watts and 120.94 kiloJoules.

Strategy 2 : We split the data into nine files containing up to 10.000 samples in this case. We can observe that the initial cost to transfer the input data from host to GPU memory is reduced to 6%, and the GPU governor overlaps all the transfers during processing, resulting in a 7 minutes improvement in execution time concerning case 1 while energy consumption is 1.8X lower.

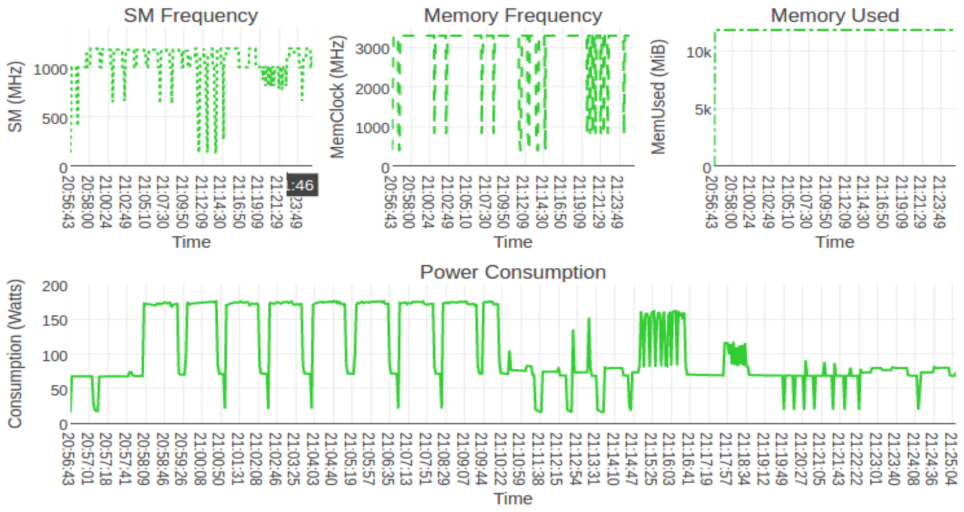



FIGURE 3.11 – Strategy 2 had 28.35 minutes, 106.05 watts and 65.11 kiloJoules.

Strategy 3 : Using a mini-batch with 1000 samples in each of the 85 files, we significantly reduce the initial data transfer cost from host to GPU memory to 3%, and the GPU governor overlaps all transfers with processing, resulting in a 2.3 mins execution time improvement concerning case 2 while being twice more energy-efficient than case 1. In this case, the layer output weights writing starts to appear.

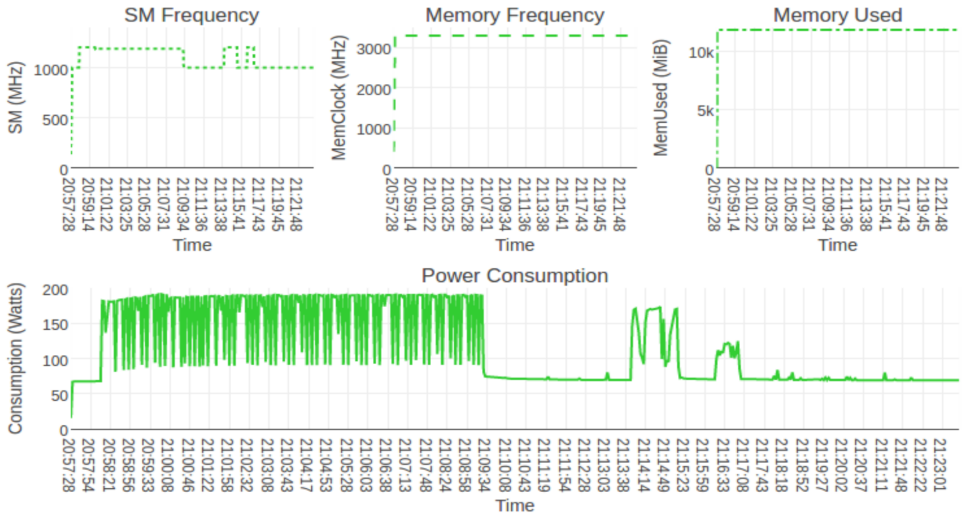


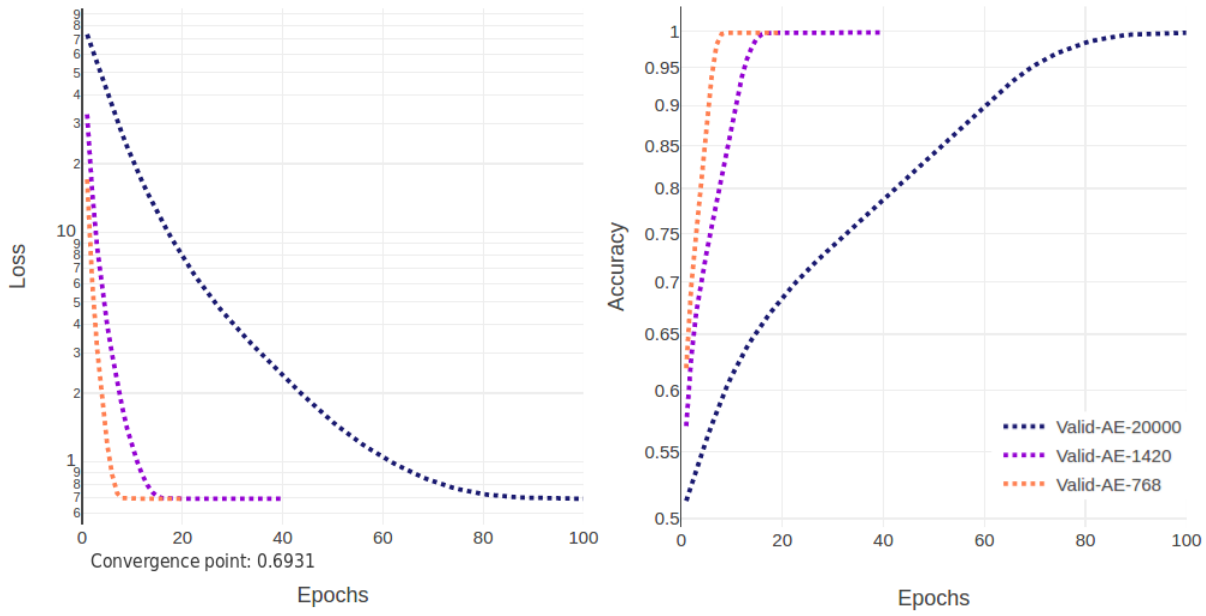

FIGURE 3.12 – Strategy 3 had 25.99 minutes, 115.19 watts and 62.20 kiloJoules.

### 3.4.2 Gradient Computations Number to Early Model Convergence

To illustrate the impact of processing more gradient updates as a factor to fast convergence, consider the traditional fully connected autoencoders (AE), parametrized with 3-hidden layers of $[2000, 1000, 500]$ neurons per layer, *relu* is used as activation function, *Adam* such as optimizer and *sigmoid_cross_entropy* as loss function. The clinical dataset uses 84.999 records for training and 4.950 records for validation.

The same AE model has been executed using three different data batch partitions of 20.000, 1.420, 768 records by batch to measure the number of epochs needed to arrive at the convergence point, characterized by the minimum loss value of 0.6931 as shown in Figure 3.13a. We can observe that the largest batch partition of data requires, to reach the convergence point, a greater number of epochs. The 20.000 batch size partition reach the convergence point in 100 epochs for 36.21 minutes for each batch, the 1.420 in 20 for 7.9 MN/batch and the last batch size (768) in 10 epochs for 4.3 MN/batch. Thus, it is possible to estimate that the consumption required to build the model has an average consumption of $[63.35, 86.61, 82.21]$ watts respectively with an energy consumption of $[137.65, 41.26, 21.87]$ kilojoules. For the dataset and model considered, a 768 item batch size is the most energy efficient for generating batch gradient updates.

FIGURE 3.13 – Network convergence using batch partitions of $[20000, 1420, 768]$ records to generate $[4, 59, 110]$ gradient updates by epoch respectively.



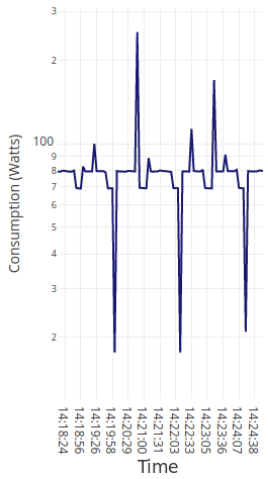

(a) Learning curves for the loss function (b) Learning curves for performance accuracy

Low power consumption was obtained when the batch of data was very large as 20.000

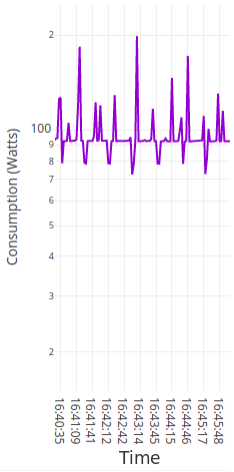
samples per batch. This amount of data plus the model exceeded the memory capacity, generating periods of idle status on the GPU with a SM frequency of 847.49 *MHz*, when the batch is transferred from main memory to the device memory. This idle phenomenon is not observed in the other batch sizes, which showed more continuous processing with a GPU SM frequency of 1071.97 and 1015.49 MHz for the 1420 and 768 records per batch.

To illustrate the impact of the idle status on the GPU, generated by large data batch partition, we can observe the same window of 6 minutes shown in the Figure 3.14a. This window is extracted of the training of the AE model when it is executed using three different batch partitions.

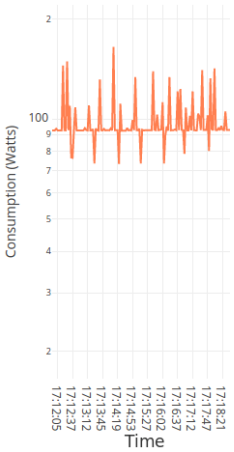
FIGURE 3.14 – Impact of GPU idle status generated by large data batch partition, consider the power consumption in a window of 6 minutes for the previous experiment.




(a) 63.35 *Watts* on average to process 68 gradient updates in 17 epochs.




(b) 86.61 *Watts* on average to process 885 gradient updates in 15 epochs.




(c) 82.21 *Watts* on average to process 1540 gradient updates in 14 epochs.

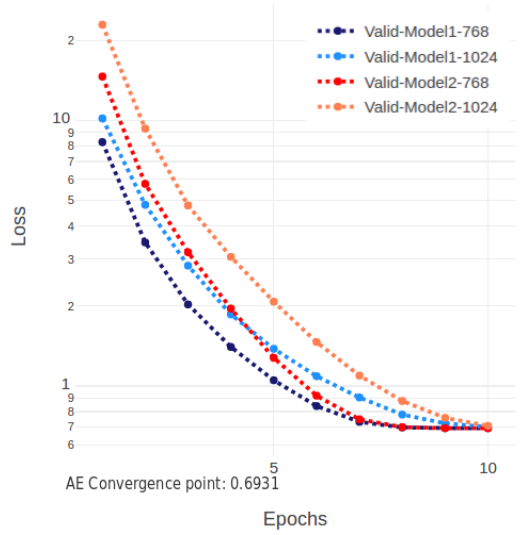
### 3.4.3 Worker Number to Early Model Convergence

The experiments use a synchronous distributed gradient for neural network training, combining computational parameters such as the number of workers with the batch size as a factor for early model convergence. The objective is to analyze the worker scalability for training the autoencoder neural networks on a mini-cluster composed by Nvidia Jetson TX2, using a fixed neural network with three layers and $[2000, 1000, 500]$ neurons per layer.

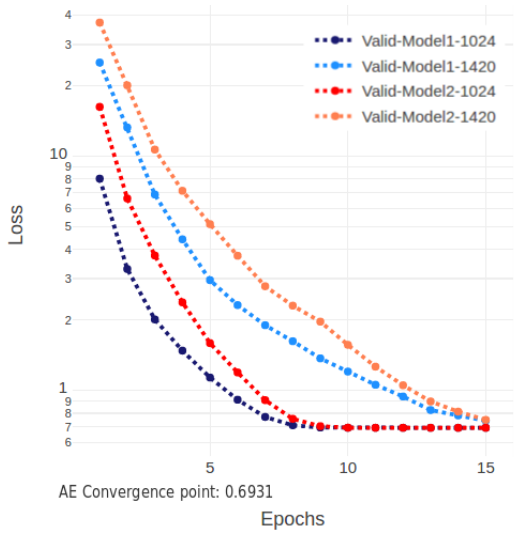
The experiments are divided into three groups of different number workers : The first group had *one Parameter Server and three workers*, using two different batch sizes like

[768, 1024] ; The second group had *one Parameter Server and six workers*, using two different batch sizes like [1024, 1420] ; And the third group had *one Parameter Server and eight workers* with a fixed batch size like [1066], and the convergence curves of the experiments are shown in Figure 3.15.

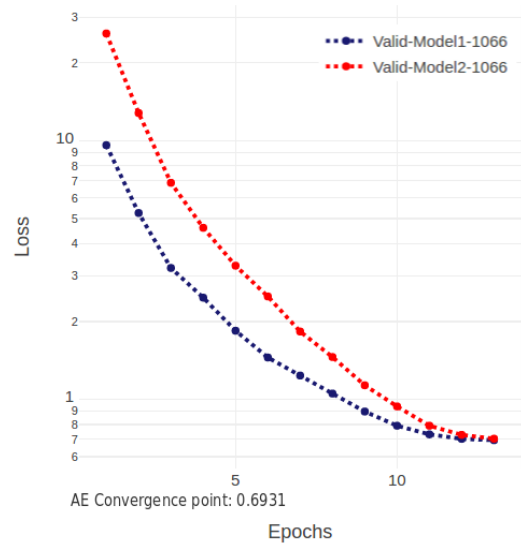
FIGURE 3.15 – Early convergence comparison between different groups of workers and task granularity for distributed training with 10.000 records and 11.466 features.



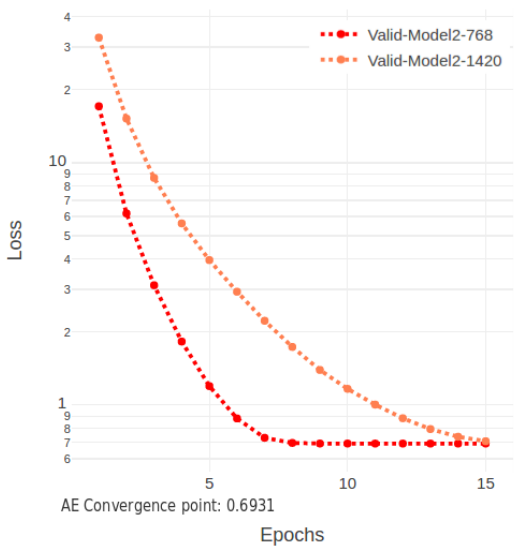

(a) 1.30 MN in average for processing one epoch on 1 PS & 3 workers.




(b) 1 MN in average for processing one epoch on 1 PS & 6 workers.




(c) 50.6 Sec in average for processing one epoch on 1 PS & 8 workers.




(d) 25.75 Sec in average for processing one epoch on 1CPU and 1GPU Titan X.

According to Figure 3.15, the third group with *one parameter server and eight workers* presented the shortest execution time per epoch, using distributed data parallelism for training the unsupervised neural network. Specifically, using eight workers executed an epoch in 50.6 seconds, while using only three workers, it took 70.6 seconds to execute

an epoch. Furthermore, the graphs show that using eight workers converges the neural network faster, as shown in the following table 3.5.

However, as the selected model was very small, the total run time was much shorter for group two with six workers, which had a total run time of 9.95 minutes with a batch partition of 1024 samples. While group three with eight works had a total run time of 10.18 minutes with a batch partition of 1066 samples. This result may be given by the synchronization latency of the workers, which is evaluated in the next chapter, using a larger workload as well as a different type of neural architecture.

TABLE 3.5 – Preliminary results for processing the unsupervised patient phenotype representation on the mini-cluster Jetson TX2.

| **DNN** | **1 PS & 3 WKR** | | **1 PS & 6 WKR** | | **1 PS & 8 WKR** | | **Workstation** | |
|---|---|---|---|---|---|---|---|---|
| | Batch Size | Conv. Time (*mins*) | Batch Size | Conv. Time (*mins*) | Batch Size | Conv. Time (*mins*) | Batch Size | Conv. Time (*mins*) |
| M-1 | 768 | 13.49 | **1024** | **9.95** | 1066 | 10.18 | | |
| M-1 | 1024 | 11.90 | 1420 | 10.51 | | | | |
| M-2 | 768 | 14.50 | 1024 | 11.40 | 1066 | 11.76 | 768 | 3.97 |
| M-2 | 1024 | 12.50 | 1420 | 12.48 | | | 1420 | 5.96 |

## 3.5 Summary

In summary, DiagnoseNET workflow automates the training process of the machine learning pipeline, allowing each stage to have enough dynamism to configure different parameters according to the problem or search for the best model to specify and explain the medical task. The work carried out so far has allowed us to highlight that the use of a well-chosen latent representation instead of the initial binary representation could make it possible to significantly improve processing times (up to 41%) while maintaining the same precision.

Minimizing the execution time of a perceptron multi-layer on a Jetson TX2 cluster, whether to perform an auto-encoder or to perform classification, depends on the application's ability to distribute data for analysis efficiently to the various Jetsons based on the available SSD memory space and then cut that data into a mini-batch based on the available memory space on the GPUs. Using hundreds of gradient updates by epochs with synchronous data parallelism offer an efficient distributed DNN training to early

convergence and minimize the bottleneck of data transfer from host memory to device memory, reducing the GPU idle status.

One technical problem in multi-platform DNN training is developing a different class by the platform for the same neural network. To avoid this problem, we have refactored the framework DiagnoseNET to provide a modular library and facilitate multi-platform experimentation and following deep neural networks implementation, described in the next chapter and explained in detail the DiagnoseNet modular framework to scale neural networks on heterogeneous systems applied to medical diagnosis.

# Chapitre 4

# Scalable Deep Learning Models in Heterogeneous Systems

Determining an optimal generalization model with deep neural networks for a medical task is an expensive process that generally requires large amounts of data and computing power. Furthermore, the complexity of the programming expressiveness increases to scale deep learning workflows over new heterogeneous system architectures for training each model and efficiently configuring the computing resources. We introduce DiagnoseNET, an automatic framework designed for scaling deep learning models over heterogeneous systems applied to medical diagnosis.

DiagnoseNET is designed as a modular framework to enable deep learning workflow management and allow neural networks' expressiveness written in TensorFlow. At the same time, the DiagnoseNET runtime abstracts the data locality, micro batching, and distributed orchestration to scale the neural network model from a GPU workstation to multi-nodes. The primary approach comprises a set of gradient computation modes to adapt the neural network according to the memory capacity, the workers' number, the coordination method, and the communication protocol (GRPC or MPI) to balance accuracy and energy consumption. The experiments allow the evaluation of the computational performance in terms of accuracy, convergence time, and worker scalability to determine an optimal neural architecture over a mini-cluster of Jetson TX2 nodes.

## 4.1 Embedded Computing Clusters

Healthcare demands new computing paradigms to meet the need for personalized medicine, next-generation clinics, enhanced quality of care, and breakthroughs in biomedical research to treat disease. Today's research requires infrastructure that can handle large

computational workloads to derive fast and accurate insights from vast amounts of data.

Heterogeneous parallel programming has two main problems on large computation systems : the first is the increase of power consumption on supercomputers in proportion to the number of computational resources used to obtain high performance. The second one is the underuse of these resources by scientific applications with the improper distribution of tasks. Therefore, selecting the optimal computational resources and making an exemplary mapping of task granularity is the fundamental challenge for building the next generation of Exascale Systems.

### 4.1.1 Rosie - Mini-Cluster of Jetson TX2 Nodes

The Rosie mini-cluster were built a mini-cluster using a desk to put together the 14-nodes Jetson TX2 interconnected by 1 $GigE$ switch Ethernet as shown in the Figure 4.5 above.

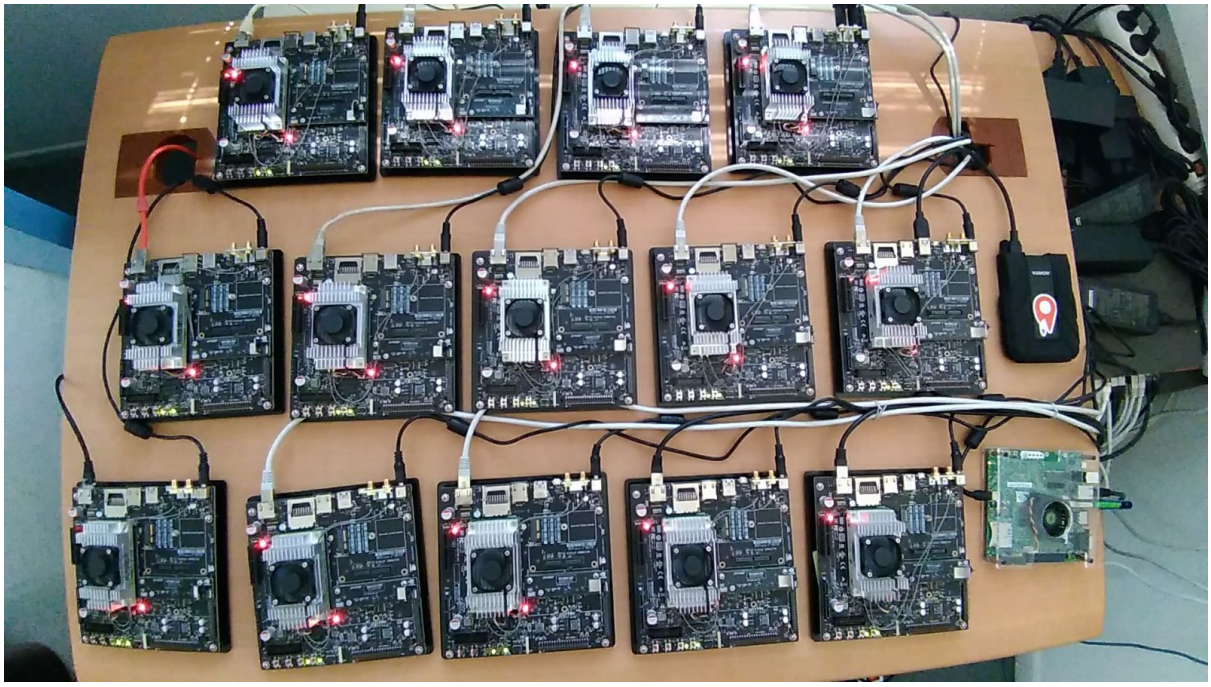


FIGURE 4.1 – The experimental Mini-Cluster Jetson TX2 for distributed training deep neural networks applied to healthcare decision-making.

The nodes are identical, independent machines, and each one runs a separate OS. Every node is composed of one developer kit Jetson TX2, which contains a hybrid processor Nvidia Denver with one ARM Cortex-A57 quad-core with a Pascal GPU 256-$CUDA$@$cores$ with a maximum. It has $8GB$ of LPDDR4 memory, $59.7GB/s$ of memory bandwidth, $32GB$ of internal storage, and one external SSD was mounted over one node

to provide a Network File System (NFS) to make that storage available to the whole cluster.

### 4.1.2 Astro - Array-Server of Jetson TX2 Nodes

The Jetson architecture-based array-server, called Astro, is a powerful, high-performance deep learning server solution with low power consumption. The Astro server has three processor module carriers that house up to eight Jetson TX2 modules each and are all connected via a Gigabit Ethernet fabric through a specialized managed Ethernet switch developed by Connect Tech with 10G uplink capability, as shown in the Figure 4.2.

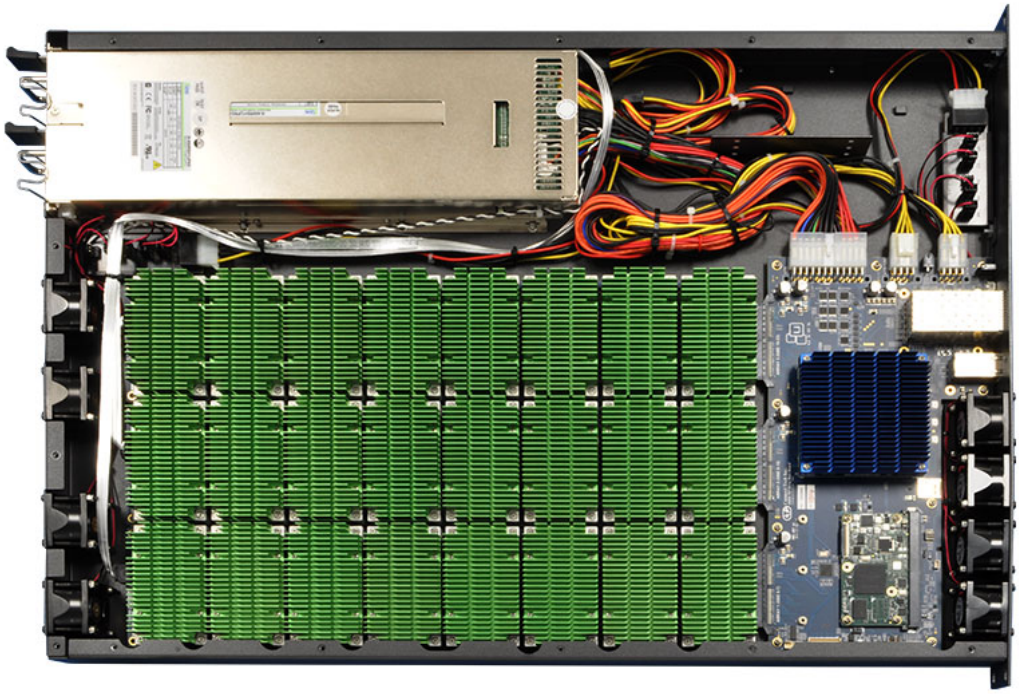


FIGURE 4.2 – The TX2 Server is an extremely low wattage, high performance deep learning server with 24 NVIDIA Jetson nodes.

Therefore, Astro contains a computing capacity of 24 NVIDIA Jetson TX2 nodes with 144 CPU cores, 6144 CUDA-enabled Pascal GPU cores, storage, and 10 Gigabit networking in a standard 1U form factor. It is structured over three arrays with 8 TX2 modules, each one with a single enclosure with access to power control, serial console, and networking. The integrated network switch further reduces the cable clutter with integrated 1 Gigabit internal network fabric and 2x 10 Gigabit uplinks located at the front of the chassis. The typical power consumption of 180W with 24 TX2 nodes running CPU and GPU loads being an edge-embedded and highly converged platform, energy-efficient

systems to be a green data center.

## 4.2 DiagnoseNET : Cross-platform Library

To implement these different algorithms and, in particular, the stage of construction of the latent representation at the heart of this paper, we used the high-level framework provided by the Tensorflow library. It enables learning algorithms to be deployed in parallel or distributed architectures, enabling the necessary computing resources to be optimized. It is necessary to adjust the granularity of the tasks according to the memory capacity of the host machine, the complexity of the model and the size of the datasets. Figure 4.3 describes the different hardware architectures and software stacks used in different experiments.

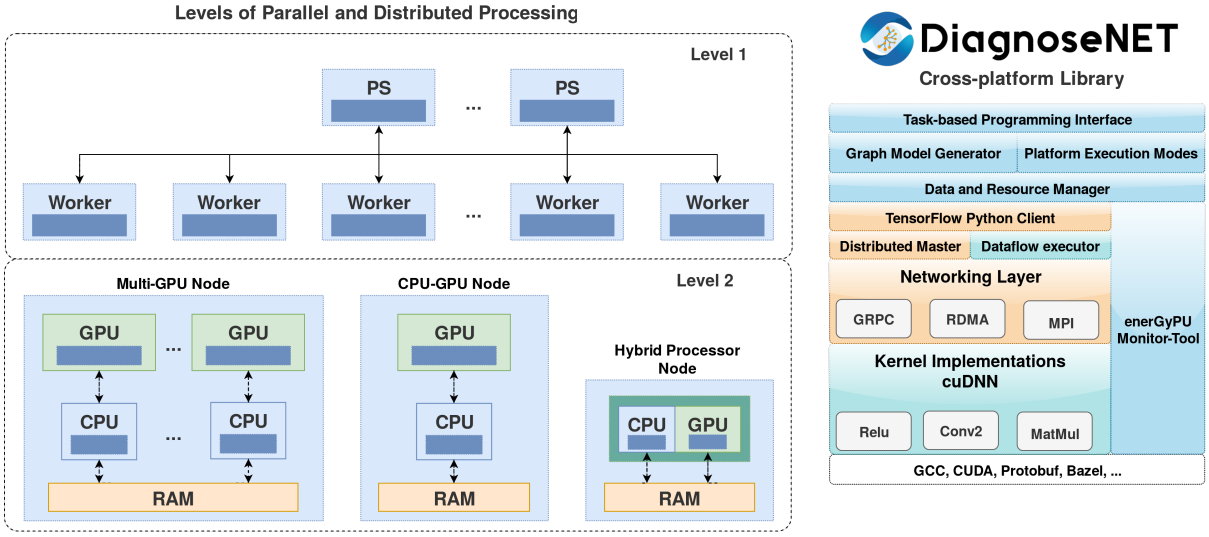



FIGURE 4.3 – Data resource management for training parallel and distributed deep neural networks and energy-monitoring tool.

To exploit the computing resources and SSD memory capacity available on Jetson TX2, the data to be processed is distributed according to the number of Jetson cards used. On each Jetson card, the data that has been assigned is also divided into the batch to take into account the available RAM space on the Jetson cards GPU.
Then, once the data is distributed on each Jetson and the mini-batch constituted on each one, the work is distributed in the form of identical tasks. In this first approach, all task replicas read the same model's values to be built from a host machine, calculate the gradients in parallel with their assigned data and return the new gradients to the host machine using the synchronous approach described in [Abadi 16a].

## 4.3 DiagnoseNET : Modular Framework Facility

DiagnoseNET was designed to harmonize the deep learning workflow and automatize the distributed orchestration to scale the neural network model from a GPU workstation to multi-nodes. Figure 4.4 shows the schematic integration of the DiagnoseNET modules with their functionalities.

The first module is the deep learning model graph generator, which has two expression languages : a *Sequential Graph* API designed to automatize the hyperparameter search and a *Custom Graph* which support the TensorFlow expression codes for sophisticated neural networks. The second module is the data manager, composed of three classes designed for splitting, batching, and multi-tasking any dataset over GPU workstations and multi-nodes computational platforms. The third module extends the enerGyPU monitor for workload characterization, constituted by a data capture in runtime to collect the convergence tracking logs and the computing factor metrics, and a dashboard for the experimental analysis results [John A. Garcia H. 16]. The fourth module is the runtime that enables the platform selection from GPU workstations to multi-nodes whit different execution modes, such as synchronous and asynchronous coordination gradient computations with gRPC or MPI communication protocols.

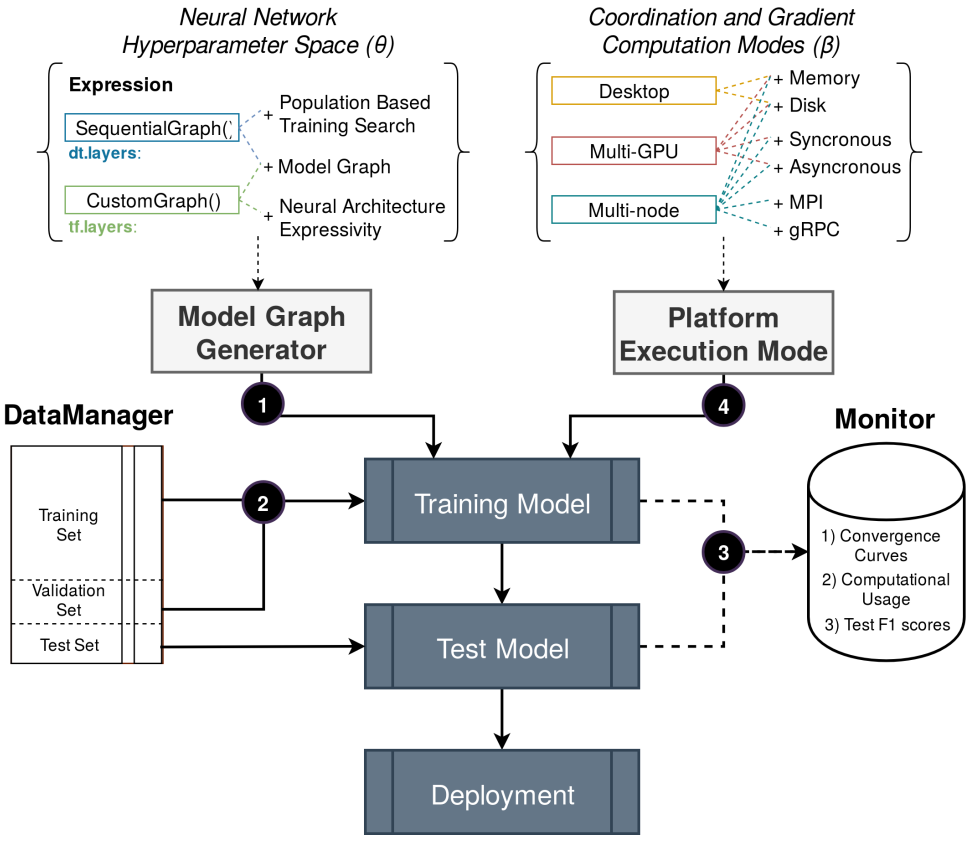



FIGURE 4.4 – DiagnoseNET framework scheme.

### Model Graph Generator

In *Sequential Graph*, the first step defines the stacked layers and sets the type of each layer, their neurons numbers, the number of layers, followed by a linear output on top since the cross-entropy will be used loss function and include the softmax function. Then the neural network hyperparameters are defined as shown in the expression 1. to generate the model graph object. Finally, *Custom Graph* uses $tf.layers$ to define the staked layers, and a similar expression as the former is used to define the optimizer and loss function for generating the model graph object.

```
import diagnosenet as dt


stacked_layer_1 =  [dt.Relu(14637, 2048),
                    dt.Relu(2048, 1024),
                    dt.Relu(1024, 1024),
                    dt.Linear(1024, 14)]


model_1 = dt.sequentialGraph(
    input_size=14637, output_size=14,
    layers=stacked_layer_1,
    loss=dt.CrossEntropy,
    optimizer=dt.Adam(lr=0.001))
```

API Expressions 4.1 – Model definition to generate several graphic-model objects.

### Data Manager

This manages the dataset according to the computational architecture, creating an isolated sandbox for each dataset and its transformations in the training process to guarantee the data location. For example, in which the dataset is splitting into well balance batches over the number of workers, and its worker-batch is micro batching according to the memory or parameter, as shown in the following code expressions :

```
data_config_1 = dt.Batching(
    dataset_name="medical_D1", valid_size=0.05,
    devices_number=4, batch_size=128)
```

API Expressions 4.2 – Dataset splitting and micro-batching over the workers.

### Monitor

The monitor collects the energy consumption metrics for *x86* and *arm* computing architectures. Additionally, it collects the bandwidth metrics when launched on a dis-

tributed platform. All the computing and DNN model metrics are written in a default directory called the testbed experiment outputs directory. For distributed environments, it requires the *machine type* specification, the DiagnoseNet workspace location and *testbed path* location as the follows.

```
monitor_config = dt.enerGyPU(
    machine_type="arm",
    file_path=diagnosenet_path,
    testbed_path=myworkspace))
```

API Expressions 4.3 – Monitor set for trancking the distributed *ARM* machines.

### Plaform Execution Mode

The last step allows the multiplatform execution, in which the model graph object is set, the dataset configuration. An example is selecting the 'DesktopExecution' for training the feed-forward neural network over a CPU-GPU machine exploiting the memory capacities.

```
# Select the computational platform:
platform = dt.DesktopExecution(
    model=mlp_model,
                datamanager=data_config,
                monitor=monitor_config,
                max_epochs=20)

# Select the training platform modes:
platform.training_memory(
    input_features=X.npy,
    target_labels=y.npy)
```

API Expressions 4.4 – Desktop execution with memory training modes.

## 4.4 DiagnoseNET : Automated Distributed Deep Learning

The data resource management is built to scale deep learning models quickly, automatize the mapping process on the computational resources, and adjust task granularity according to memory host capacity, model complexity, and data batch size to minimize the energy consumption at the training stage. Concurrently the data resource manager designates the role and transmits the DNN hyperparameters to be used on the master (or parameter-server) and each of the workers, as shown in the above Figure 4.5.

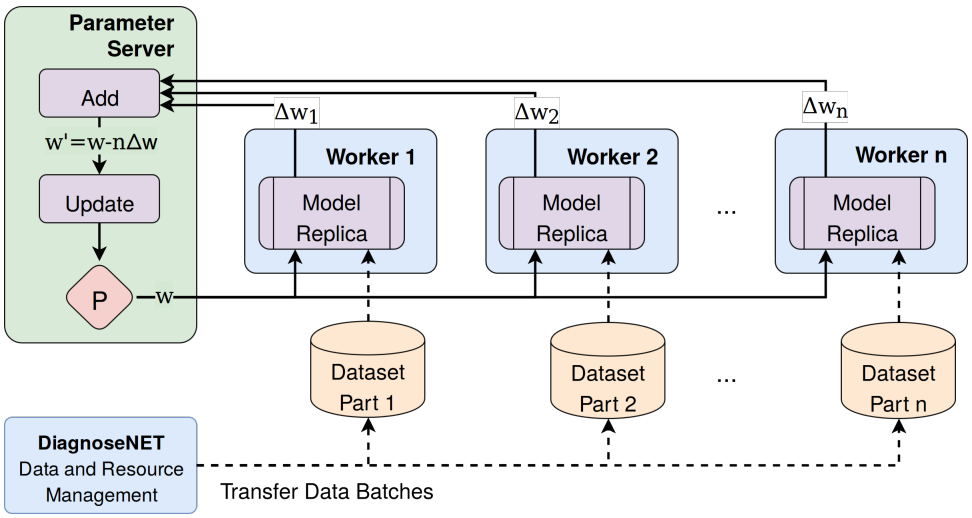



FIGURE 4.5 – The schema for the synchronous learning of mini-batches in a distributed memory platform, using the data and resources management module of DiagnoseNET.

### 4.4.1 Training Deep Neural Networks with gRPC

It harmonizes the computational resources with the dataset manager to train previously defined models over a multi-node platform, automating the gRPC communication protocol to coordinate the workers with asynchronous gradient computations. The resource manager divides the dataset equally onto the workers' nodes of the system, where each worker has a copy of the neural network (graph) and its local weights. Each worker operates on a unique subset of the dataset and updates its local set of weights. These local weights are shared across the cluster to compute a new global set of weights through an accumulation algorithm.

Compared to the Desktop implementation on DiagnoseNET API, the distributed gRPC uses a *Resource Manager* instance, which will be responsible for launching the experiment and turning on the monitor over the nodes selected. The main task is to launch the model replica on every node via the SSH protocol for the graph replication function.

```
import diagnosenet as dt
dt.between_graph_replication(
            d_replica_path=/myworkspace,
            d_replica_name="GRPC_replica.py",
            ip_ps="host1",
            ip_workers="host2,host3,host4,host5",
            num_ps=1, num_workers=4)
```

API Expressions 4.5 – Distribted orchestration with GRPC asynchronous.

On the side of the replica script, it gives the model graph object, creates the dataset batching, and passes both of these to a *Distributed_GRPC* object. This object is

responsible for launching the experiment through its function *asynchronous_training*.

```
platform = dt.Distibuted_GRPC(
            model=model_1,
            datamanager=data_config_1,
            monitor=enerGyPU(machine_type="arm"),
            max_epochs=20,
            ip_ps=argv[0], ip_workers=argv[1])

platform.asynchronous_training(
            dataset_path=/myworkspace/datasetpath,
            inputs_name="X.npy", targets_name="Y.npy",
            job_name=argv[0], task_index=argv[1])
```

API Expressions 4.6 – GRPC asynchronous replica.

### 4.4.2 Training Deep Neural Networks with MPI

DiagnoseNET implements synchronous and asynchronous MPI methods to improve performance in the communication between workers. For example, asynchronous gradient updates were optimized with a weighting parameter, which is responsible for determining the number of workers required in each step to compute the new weights and broadcast them.

```
platform = dt.Distibuted_MPI(
model=model_1, datamanager=data_config_1,
monitor=enerGyPU(machine_type="arm"),
max_epochs=20, early_stopping=3])

platform.asynchronous_training(
dataset_name="medical_D1",
dataset_path=d/myworkspace/datasetpath,
inputs_name="X.npy", targets_name="Y.npy",
weighting=1)
```

API Expressions 4.7 – MPI Platform Execution Modes.

### 4.4.3 MPI Synchronous Gradient Descent

It uses Point-to-Point communication between workers, unlike gRPC does not require a launcher orchestration, but each worker will be blocked while sending and receiving messages.

The algorithm 1 describes the MPI synchronous coordination training with parameter server. It uses the nodes' ranks to assign them the parameter server or worker role, defined the rank 0 as parameter server (PS), and the other ranks as workers. When launching the program, the PS performs pre-processing tasks, such as loading the dataset and compiling

the model. After these tasks, the PS sends the model to the workers, ready to receive it. At each training step, the PS sends a different subset of the data to every worker for loss optimization. At the end of an epoch, the PS will gather the new weights from every worker. Then, workers collect weights and compute the average weight for the global update. For the other computing parts, it works as the desktop version.

**Algorithm 1** Synchronous MPI Kernel

```
if master True then
    masterInput ← {dataset, workers}
    DistributedBatching(dataset, workers)
else
    workerInput ← {batches, hyperparameters}
    model ← GraphGenerator(hyperparameters)
while ConvergenceCondition do
    if master == True then
        for all worker ∈ workers do
            masterGrads ← received(workerGrads)
        averageGrads ← average(masterGrads)
        send(averageGrads)
    else
        workerGrads ← compute(model, batches)
        send(workerGrads)
    if master == True then
        for all worker ∈ workers do
            masterLoss ← received(workerLoss)
        averageLoss ← average(masterLoss)
        if decrease(averageLoss) True then
            send(updated(masterWeights))
        if overfitting(averageLoss) True then
            send(averageLoss, earlyStopping)
        else
            send(averageLoss, False)
    else
        workerWeights ← received(masterWeights)
        projection ← model.Apply(workerWeights)
        workerLoss ← computeLoss(projection, labels)
        send(workerLoss)
```

### 4.4.4 MPI Asynchronous Gradient Descent

The algorithm 2 describes the MPI asynchronous coordination training with parameter server. It allows training multiple model replicas in parallel on different nodes with different subsets of the data. Each model replica processes a mini-batch to compute gradients and sends them to the parameter server that applies a function (mean, weighted average) between previous and received weights, then updates the global weights accordingly and sends them back to the workers.

**Algorithm 2** Asynchronous MPI Kernel

```
if master True then
    masterInput ← {dataset, workers}
    DistributedBatching(dataset, workers)
else
    workerInput ← {batches, hyperparameters}
    model ← GraphGenerator(hyperparameters)
while ConvergenceCondition do
    if master == True then
        convergeFlag ← received(workerCond)
        masterGrads ← received(workerGrads)
        collectGrads ← collection(masterGrads)
        averageGrads ← average(collectGrads)
        send(averageGrads)
        (convergeCond, modelGrads) ← stopping(convergeFlag)
    else
        if overfitting(averageLoss) True then
            send(averageLoss, earlyStopping)
        else
            send(averageLoss, False)
        if decrease(averageLoss) True then
            send(Updated(masterWeights))
        workerGrads ← compute(model, workerInput)
        send(workerGrads)
    if master False then
        workerWeights ← received(masterWeights)
        projection ← model.Apply(workerWeights)
        workerLoss ← computeLoss(projection, labels)
```

Every worker will compute its gradients individually until convergence ; the conver-

gence occurs when we start overfitting, which means that the training loss decreases while the validation loss increases. [Chahal 18]. The master responsible for computing the weighted average of received weights and its weights will stop when all workers converge. To check the status of convergence of workers, the master has a queue that stores converged workers, and when its length is equal to the number of workers, the master knows that all workers converged and stops training. Since each node computes gradients independently and does not require interaction among each other, they can work at their own pace and have greater robustness to machine failure.

## 4.5 Case studies and Neural Architectures

### 4.5.1 Medical Care Purpose Classification for Inpatients :

The clinical dataset was derived from the medicalization of information systems (PMSI) collection of synthetic medical information in a standardized and anonymized format from hospitalizations carried out in medical care or rehabilitation settings. The patient-feature composition module was used to generate the representations of the patients' status in the first week of hospitalization using one year of the PMSI data collection. The primary clinical descriptors used were demographics, admission details, hospitalization details, physical dependence, cognitive dependence, rehabilitation time, comorbidities, morbidity, and etiology. The clinical dataset obtained has $116,831$ different inpatients and $14,637$ clinical-features embedded in a document-term sparse matrix [Garcia Henao 18]. The PMSI system has two ways to track hospitalized patients' medical care using ICD-10 codes and equivalent diagnosis-related groups organized in hierarchical levels. In this section, we worked with the high-level group called Clinical Major Category (CMC), obtaining 14 labels-categories to classify the medical care of patients hospitalized as shown in Table 2.4 in section 2 called *Healthcare Interoperability and Data Mining*.

### 4.5.2 Atrial Fibrillation Classification for Cardiac Diagnosis :

The ECG dataset was obtained from the 2017 PhysioNet Challenge [1]. The dataset was already labeled, and the four labels are : Normal, Atrial Fibrillation, Others, and Noisy. The Others label means recordings of those similar heart diseases. The total number of source datasets is $8,528$. Each sample is a single short ECG lead recording. Since the length of the sample is inequivalent, samples are transformed into structured input. The peaks $R$ of recordings are extracted to get the centered windows of 260 time steps, which are complete ECG rhythms for a cycle. Then, to better represent the behavior of the recording, each five consecutive centered windows are concatenated into a training

1. Atrial Fibrillation Dataset Classification from a short single lead ECG recorded by the physioNet computing in cardiology challenge 2017. `https://physionet.org/challenge/2017/`

sample as shown in the following table 4.1. It contains similar rhythms, which are labeled as Others. Noisy recordings are also added to decrease episodic detection. The addition of other irregular ECG recordings and noisy data samples can help detect the AF rhythms better.

TABLE 4.1 – Medical Target 2 : Cardiac Arrhythmia Labels

| Class | Label Description | Source dataset | Training Dataset | Small Samples |
|---|---|---|---|---|
| 0 | Normal | 5,050 | 34,303 | 4,241 |
| 1 | AF | 738 | 6,542 | 815 |
| 2 | Others | 2,456 | 18,986 | 2,424 |
| 3 | Noisy | 284 | 1,382 | 171 |

### 4.5.3 Implementation of the MLP in DiagnoseNET

In DiagnoseNET, the network architecture was composed dynamically through fully-connected layers ; each neuron is connected to all neurons of the previous layer building a stacked neural network and followed by a softmax layer on top $h_i = f(\sum_{j=1}^{n} w_{ij}x_j + b_{ij})$, where $x_j$ is the output of the previous layer and $w_{ij}$ is the weight value associated with $x_j$ with a bias associated $b_{i,j}$ and $n$ is the number of neurons in the previous layer, while $f$ is the as activation function. Having as a baseline the neural network used in work called *improving palliative care deep learning* [**?**] and after finetuning it to classify the medical care purpose with PACA inpatients. The model used to evaluate the scalability comprised an input (of 10,833 dimensions), four hidden layers (every 512 dimensions), and a softmax output layer as activation function was used rectified linear unit (ReLU), as loss function was used categorical cross-entropy and Adam as optimizer [Kingma 14]. The first medial task is based on a Multilayer Perceptron Network (MLP) designed to input the high-dimensional patients' representations and generate the label projections as outputs.

### 4.5.4 Implementation of the CNN in DiagnoseNET

The neural network baseline for the second medial task is based on a Convolutional Neural Network (CNN) designed to take as input the time-series of ECG signal and generates the sequence of label predictions as outputs [Rajpurkar 17]. The general neural architecture is composed using DiagnoseNET with 75 layers of convolution followed by a fully-connected layer and a softmax layer on top, as shown in the Appendix 4.6.

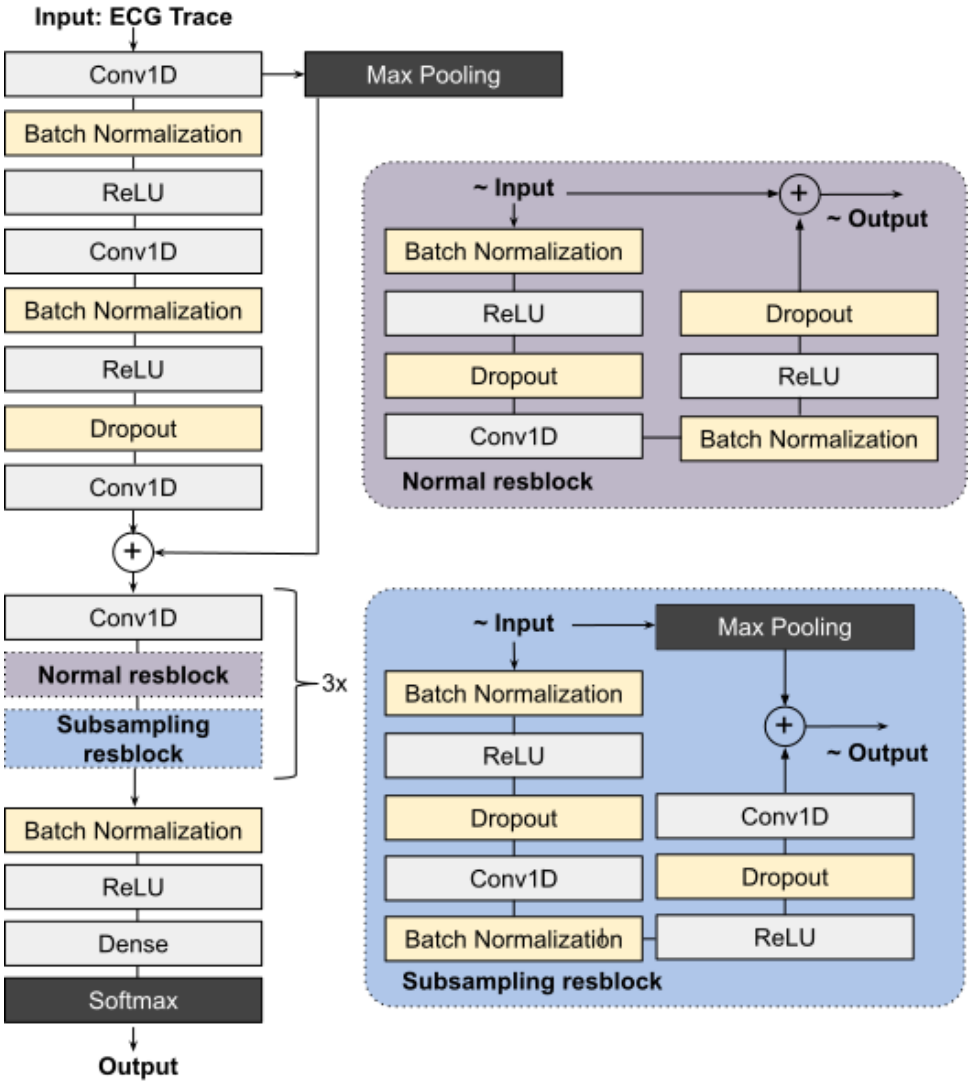



FIGURE 4.6 – ECG Convolutional Neural Architecture.

The significant elements in the CNN model are the residual network, the convolutional layers, and the regularization methods, such as batch normalization, dropout, and activation, which are used to improve the performance and regularization of the CNN model. The convolutional layers are used to extract features relative to the form of the traces wave. The pure CNN model leads to the problem that the last layer of the model may not exploit the original features or the ones extracted in the first layers

This can be solved by the residual network connections proven to solve the information loss problem. To implement this, a second information stream is added to the model. In this way, deeper layers have access to the original features, in addition to the information processed by the previous layers [He 15, Bai 18]. Two different types of a residual blocks are included to access the different states of the information. The stable residual block preserves the input size while the sub-sampling residual block lowers the size of the input

down to a half. By using max pooling, the network extracts only the high values from input so that the size of its output is halved.

## 4.6 Experiments and Results

The experimental procedures are oriented to compare the distributed training scalability using different parallel methods to classify the medical targets, while is analyzed the convergence effects between accuracy and the usage of the computational resources for training each model on different workers to compare the coordination training modes using GRPC and MPI communication protocols. Here we present the first results of a series of experiments that provide clues for efficient computing performances over heterogeneous platforms.

The experiments use the DiagnoseNET self-expression codes for training the first task called : *medical care purpose classification*, and use the Tensorflow API to describe neural network plus the DiagnoseNet runtime for training the second task called : *atrial fibrillation classification*. The core algorithms used to process the distributed gradient computation and train the two deep learning tasks are the GRPC asynchronous, MPI synchronous, and MPI asynchronous communication protocols.

### 4.6.1 HPC System and Enviroment :

The distributed experiments use a mini-cluster with 14-nodes NVIDIA Jetson TX2 interconnected by 1 $GigE$ switch Ethernet. The nodes are identical, independent machines, and each one runs a separate OS on Ubuntu 16.04, with CUDA 8.0 support, cuDNN v6 for Python 3.6. Every node is composed of one developer kit Jetson TX2, which contains a hybrid processor Nvidia Denver with one ARM Cortex-A57 quad-core with one a Pascal GPU 256-$CUDA$@$cores$ with a maximum, it has $8GB$ of LPDDR4 memory, $59.7GB/s$ of memory bandwidth, $32GB$ of internal storage, and one external SSD was mounted over one node to provide a Network File System (NFS) to make that storage available to the whole cluster.

### 4.6.2 Worker Scalability for Training the Medical Task 1 :

The baseline got 11.04 hours as convergence time for training the MLP model described in the previous session. The distributed method to communicate and synchronize the nodes' computations uses gRPC asynchronous to coordinate and compute the gradient updates between two workers and one master. In contrast, the best setting reduces

the convergence time to 1.3 hours with the MPI asynchronous method to coordinate and compute the gradient updates between 12 workers and one master, as shown in Figure 4.7,

(a) Convergence Time (HRS) (b) Speedup Scalability (Strong)

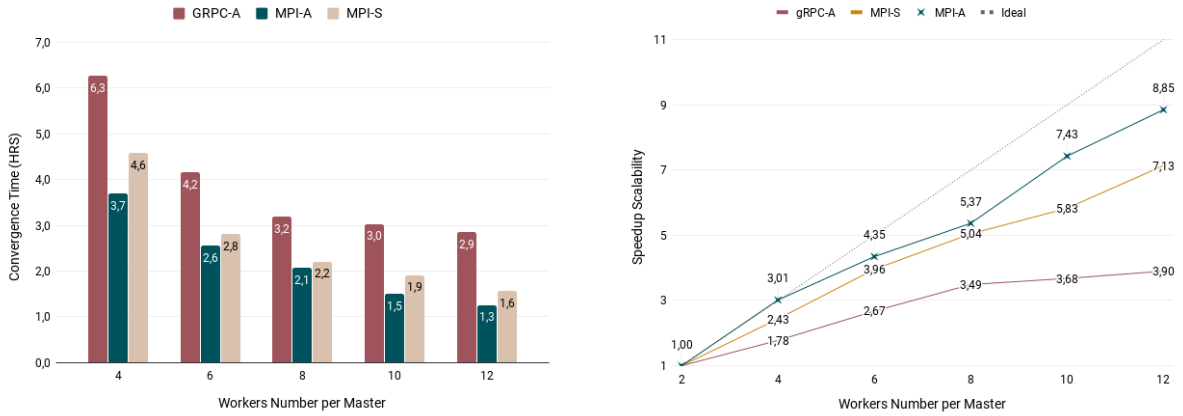



FIGURE 4.7 – Worker scalability comparison for distributed training on a mini-cluster of Jetson TX2 to classify the medical care purpose.

To predict the medical care purpose, a medium dataset (3.6 GB) with 116.831 patients was used a small model with four fully-connected layers of multilayer perceptron network that produced 35GB of host memory. Where the baseline uses a gRPC asynchronous training modes with four workers takes 7.04 hours as a time to solution achieving one accuracy of 0,87 F1-score, while the MPI asynchronous training modes with 12 workers take 2.02 hours as a time to solution achieving one accuracy of 0,61 F1-score, as shown in the Table 4.2.

TABLE 4.2 – Scalability results for training the multilayer perceptron network for the medical care purpose classification task.

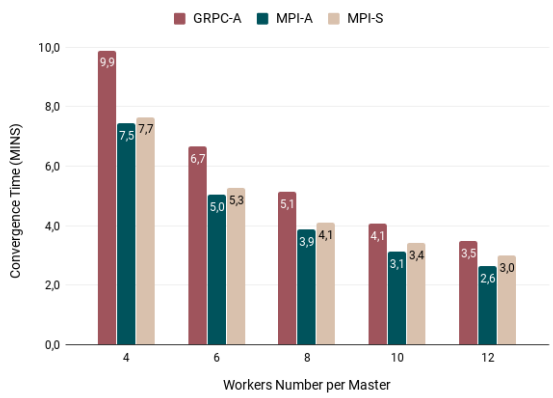
| | **F1-Score (Micro)** | | | **Time to Solution (HRS)** | | | **Latency (HRS)** | | |
|---|---|---|---|---|---|---|---|---|---|
| W. | gRPC | MPI-S | MPI-A | gRPC | MPI-S | MPI-A | gRPC | MPI-S | MPI-A |
| 4 | **0,879** | 0,645 | 0,643 | **7,04** | 6,26 | 5,01 | 0,76 | **1,69** | 1,31 |
| 6 | 0,702 | 0,637 | 0,632 | 4,87 | 3,85 | 3,32 | 0,71 | 1,04 | 0,76 |
| 8 | 0,694 | 0,629 | 0,621 | 3,83 | 3,12 | 2,86 | 0,64 | 0,91 | 0,79 |
| 10 | 0,704 | 0,620 | 0,615 | 3,55 | 2,72 | 2,29 | 0,52 | 0,81 | 0,79 |
| 12 | **0,697** | **0,631** | **0,613** | 3,30 | 2,24 | **2,02** | **0,44** | 0,68 | 0,76 |

### 4.6.3 Worker Scalability for Training the Medical Task 2 :

The atrial fibrillation classification task uses a small dataset (77MB) with 8, 528 patients and a medium model with 72 layers fully-connected to a convolutional neural net-

work with residual network connections. Where the baseline uses a gRPC asynchronous training modes with four workers take 13 minutes as a time to solution achieving one accuracy of 0.63 F1-score, while the MPI asynchronous training modes with 12 workers take 5 minutes as a time to solution achieving the same accuracy of 0.63 F1 scores, as shown in the Figure 4.8. Worker scalability comparison between the communication protocols and their coordination methods to compute the gradient updates for distributed training of each neural network by medical task on a mini-cluster of Jetson TX2.

(a) Convergence Time (MINS)




(b) Speedup Scalability (Strong)


FIGURE 4.8 – Worker scalability comparison for distributed training on a mini-cluster of Jetson TX2 to the classify atrial fibrillation.

The baseline uses a gRPC asynchronous training modes with four workers, take 13 minutes as a time to solution achieving one accuracy of 0,63 F1-score, while the MPI asynchronous training modes with 12 workers take 5 minutes as a time to solution achieving the same accuracy of 0,63 F1 score, as shown in the Table 4.3.

TABLE 4.3 – Scalability results for training the convolutional neural network for the atrial fibrillation classification task.

| | **F1-Score** (Micro) | | | **Time to Solution** (MINS) | | | **Latency** (MINS) | | |
|---|---|---|---|---|---|---|---|---|---|
| W. | gRPC | MPI-S | MPI-A | gRPC | MPI-S | MPI-A | gRPC | MPI-S | MPI-A |
| 4 | 0,63 | 0,62 | 0,64 | **12,9** | 11,3 | 10,6 | 3,0 | **3,7** | 3,1 |
| 6 | 0,65 | 0,63 | 0,60 | 9,9 | 8,8 | 8,2 | 3,2 | 3,5 | 3,2 |
| 8 | 0,66 | 0,63 | 0,62 | 9,0 | 7,8 | 6,7 | 3,9 | 3,7 | 2,8 |
| 10 | 0,66 | 0,61 | 0,63 | 7,9 | 6,9 | 5,7 | 3,8 | 3,4 | 2,5 |
| 12 | **0,65** | **0,66** | **0,63** | 6,5 | 6,2 | **5,0** | **3,0** | 3,2 | 2,4 |

Figures 4.9 and 4.10 compare the validation loss curves between the communication protocols. In the case of MPI synchronous training modes, the PS will gather the weights

from workers after an epoch to compute the average weight for a global update. It saves weight when minimizing the validation loss and loads it for testing to avoid overfitting problems. Finally, the computation for the neural network is assigned to the workers, and the rest of the jobs will be finished by the parameter master. While the MPI Asynchronous training modes graph of validation loss curves is shown in Figures 4.9 and 4.10 that using a small number of workers, we achieve convergence faster in terms of epoch (four epochs for the experiment with four workers, versus eight epochs for the experiment with 12 workers), and then we start having over-fitting ; this is due to using a low learning rate as well as having multiple workers doing calculations independently as well as the problem of stale gradients where some workers could be computing gradients using master weights that may be several gradient steps behind the current version of global weights making convergence slow and not guaranteed.

FIGURE 4.9 – Validation loss curves comparison bewteen the communication protocols (GRPC and MPI), and their methods (Synchronous and Asynchronous) to compute the gradient updates for training the medical task 1.

(a) Asynchronous gRPC. (b) Asynchronous MPI. (c) Synchronous MPI.

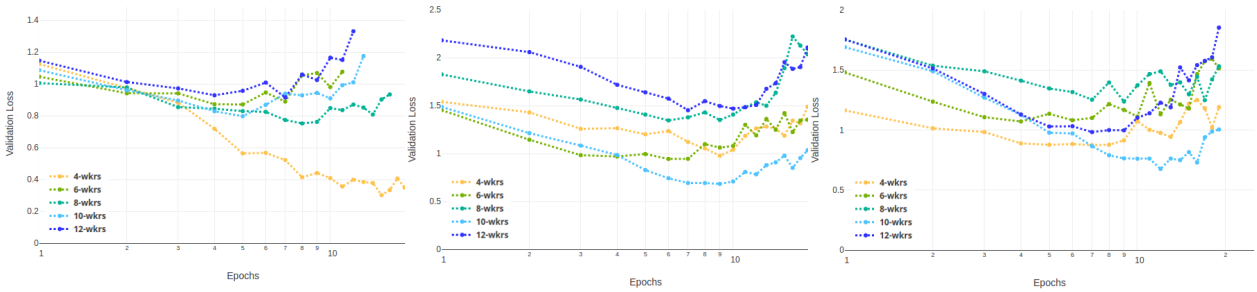
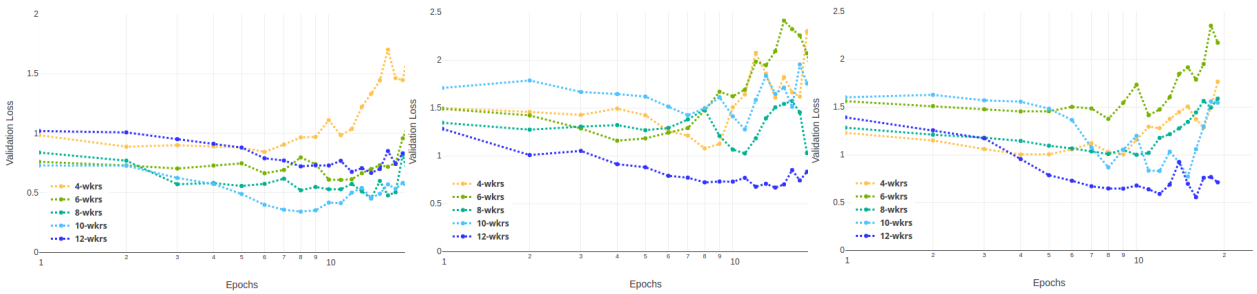



FIGURE 4.10 – Validation loss curves comparison bewteen the communication protocols (GRPC and MPI), and their methods (Synchronous and Asynchronous) to compute the gradient updates for training the medical task 2.

(a) Asynchronous gRPC. (b) Asynchronous MPI. (c) Synchronous MPI.



## 4.7 Summary

DiagnoseNET increases the developer's productivity facilitating the programming process to build and finetune Deep Learning workflows, while its runtime abstracts the data locality and the distributed orchestration to scale each model from a GPU workstation multi-nodes.

Furthermore, implementing a mini-cluster of Jetson TX2 nodes presents good scalability for distributed training of each neural network by their medical task. Therefore, clusters with embedded computation platforms can be used as a deep learning platform system with minimal infrastructure requirements and low power consumption, offering the computing capacity for processing large datasets and models in the HPDA ecosystem.

To characterize the deep learning tasks and improve the balance between accuracy, convergence time, and worker scalability, partitioning the micro-batches to use hundreds of gradient updates by epochs with MPI asynchronous gradient computations with data parallelism offer an efficient distributed neural network training for early convergence.

Likewise, adapting the number of records by batch and the model dimensionality helps minimize the bottleneck of data transfer from host memory to device memory, reducing the GPU idle status.

# Chapitre 5

# Towards Green-AI for Training Deep Neural Networks

Deep learning models have been getting increasingly extensive and computationally intensive, with the training cost for state-of-the-art models doubling every few months. The cost of training these models has been happening in several areas of artificial intelligence, including object recognition, game playing, speech recognition, and machine translation. Some researchers have argued that this trend is both environmentally unfriendly and prohibitively expensive, raising barriers to participation in artificial intelligence research.

In response to this situation, Schwartz et al. has published an article entitled Green AI in Communications of the ACM, December 2020 [Schwartz 20]. He defined Red AI as the design and training of neural networks, where they are only concerned with obtaining greater accuracy, linearly with the increment of data, the number of parameters, and the number of computational resources used without considering cost carbon footprint. In contrast, Green AI considers efficiency as a primary evaluation criterion, along with accuracy, time, and reproducibility. Therefore, the neural network design is focused on solving the domain task and performing similar or better than the state-of-the-art with a lower carbon footprint.

Identifying the optimal granularity level to train deep neural networks is necessary to efficiently use the computational resources and reduce energy consumption in execution time. Therefore, this chapter describes the importance of task granularity in deep learning algorithms and how it affects data movement, convergence time, and energy consumption. Experiments and results are measured to balance accuracy and energy consumption according to the hyperparameter settings of the neural network. For the different configurations of neural architecture, the partitioning of the task between fine-grained and coarse-grained tasks is analyzed to observe the impact of energy consumption.

## 5.1 Green AI Approaches to Reduce Carbon Footprint

Green-deep learning is a way of doing deep learning that is more environmentally friendly. Therefore, AI workflows are modified from different approaches to use less electricity and create fewer carbon emissions while automation achieves the target task. A recent systematic review on developing deep learning technologies in the environment classifies approaches to achieve novel results with lightweight and efficient technology into four categories [Xu 21]. As shown in the following items :

— **Compact Architecture Design :** This is divided into two parts, component design, and component assembling. *The component design,* is focused on building new neural architectures by introducing efficiency variants, whose components include convolution layers, attention mechanism, as well as the use of embedding to reduce the size of the representations. While, *the component assembling* ocuses on building a network efficiently by sharing resources such as shared memory, weight sharing, and weight sharing through selected convolutional blocks as used in the neural architecture search process to select the best model.

— **Energy-Efficient Training :** focuses on training neural networks that can reduce computations required during the whole training, including weight tuning and hyper-parameter tuning. This can be done by making sure the network is properly initialized, normalized, and by using progressive training. For example, EfficienNet focuses on building and searching for a CNN that can achieve higher accuracy and efficiency, using a multi-objective function to build the model that obtains good accuracy with fewer parameters [Tan 19].

— **Energy-Efficient Inference :** This is a way to reduce the amount of energy needed to make inferences. This is done by reducing the number of calculations needed, using low-rank factorization, quantization, and knowledge distillation.

— **Efficient Data Usage :** This is about using data more efficiently, not needing as much data to get good results. It includes ways to do this, like active learning and pre-training approaches such as Few-shot learners. Pre-trained models are computer models that are heavily trained on a large dataset and then used as a starting point for a model on a new task. This is done in order to save time and data on

the new task.

This chapter focuses on reducing the carbon print of deep neural networks through energy-efficient training approaches. Specifically, we minimize power consumption for the hyperparameter finetuning and model selection process. The DNN trains several models to determine the optimal generalization accuracy model, which consumes time and energy. In addition, using hundreds of gradient updates with synchronous data parallelism impacts the solution's energy. Therefore, the direct approach extends the task granularity from High Performace Computation (HPC) simulations to Deep Learning (DL) models, which combines the principles of data parallelism and mini-batch online learning with the platform memory capacities as a factor for early model convergence.

## 5.2 Task-Based Parallel Programming Model For HPC

Task-based parallelization is a way to make a computer program run faster by dividing it up into smaller tasks that can be completed at the same time. Usually, a task-based program is transformed into a direct acyclic graph (DAG) of tasks. The vertices are the computational operations, and the edges are the data needed and the dependencies between them. Usually, the task-based program uses heuristic algorithms to manage the tasks over computational resources like CPU, GPU, and memory to reduce the program's execution time [Bramas 20].

Task-based programming models for shared memory like Intel Cilk Plus [Asai 15] and OpenMP 3.0 [Liao 10] are well established and documented. Moreover, more recent frameworks designed for systems with distributed memory and using the task-based programming model are Charm++ [Kalé 11] and StarPU [Augonnet 09]. The main applications of which have been focused on solving non-linear equations through dense linear algebra algorithms with applications in different domains such as Climate/Weather prediction, computational astronomy, molecular dynamics, and others.

### 5.2.1 Task Granularity

In computing, the term "granularity" refers to the amount of data processed at once. In general, the smaller the chunks of processed data, the more granular the computing. Conversely, the larger the chunks of data, the less granular the computing. Granularity is often measured in terms of the amount of time it takes to complete a computation compared to time spent communicating with other processors. into two approaches Fine-

grain Parallelism and Coarse-grain Parallelism.

— **Fine-grain Parallelism :** It is a way of processing operations that involves breaking tasks down into small tasks and doing them very quickly, with much communication between tasks. This type of parallelism is suitable for load balancing but can be inefficient if the tasks are too small.

— **Coarse-grain Parallelism :** It is a way of processing operations that involves breaking tasks down into heavy tasks, which means that a computer can do a lot of work simultaneously as it is talking to other parts of the computer. This makes the computer faster, but it is harder to balance the work so that each computer part is doing the same amount.

## 5.3 Task-Based Programming Model For Distributed Deep Learning

Energy-efficient training to reduce the carbon footprint of green learning combines two optimization challenges. The first focuses on building a neural architecture with the necessary components to obtain higher accuracy, reliability, and interpretation. The second focuses on training and optimizing the system resources for tailoring each neural network generated to exploit the computing platform and use the computational resources efficiently to minimize the execution time and energy consumption.

Therefore, we propose to divide the training process of DNNs into tasks by transforming it into a DAG of tasks and thus control the execution of tasks among computing resources, just as task-based programming models have done for solving computing tasks in high-performance computing. As a first step, in this chapter, we do not focus on characterizing the performance of a multi-layer perceptron to evaluate the balance between accuracy and energy-efficient performance. Instead, partitioning the micro-batches to use hundreds of gradient updates by epochs with asynchronous data parallelism offers an early convergence of distributed training of deep neural networks.

Although the task-based programming model divides the processing tasks for training neural networks, it also allows scheduling the computation task, the communication task, and the synchronization task. Therefore, the critical problem is mapping the mini-batch and the neural network graph to avoid communication bottleneck, maximize the computation process, and update the scheduling plan to fit the dynamic workloads. Therefore,

the optimization problem is finding the best mini-batch partition size according to the DNN model given and selecting the computational resources for deep neural networks, facing the following issues are related to the problem structure, problem dimension, and platform selection to map the task dependency graph.

### 5.3.1 Defining the task granularity for deep learning

The efficacy for training deep neural networks on a distributed platform is to divide the tasks into two distinct levels of granularity. Which division depends on the ability of the runtime system to profile the DNN model dimension, the dataset size and characterize the computational resources (memory capacity, number of CPUs, GPUs, the GPU micro-architecture, clocks frequency, and among others) as well the number of workers with the interconnection limitations.

**Course-grain tasks :** Figure 5.1 shows two samples of coarse-grain tasks, which are defined by the model dimension of, in this case, the number of parameters plus the size of the mini-batch set the task size, and the number of mini-batches defines the number of tasks. In the first example, the dataset is divided into two large batches, and the model is of medium size, while in the second example, the dataset is divided into four batches, and the model is of considerable size, with the granularity of these tasks classified as Course-Grain. Whose performance characterization relates to it : of few tasks, and each task has a considerable size. Which composition carries an intensive computation process, low-bottleneck tasks for managing CPU-GPU data transfers, and a few gradient-updates computation tasks.

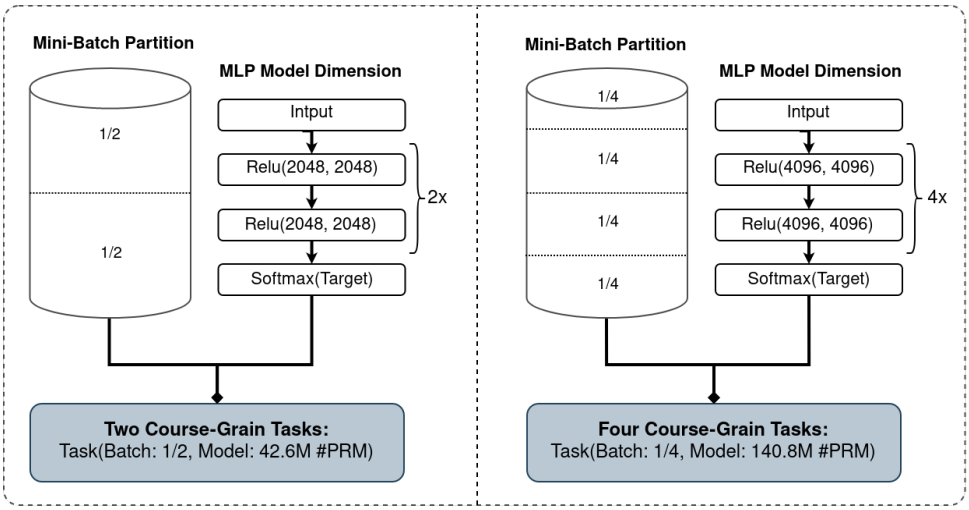



FIGURE 5.1 – Schematic sample of task granularity for fully connected neural networks.

**Fine-grain tasks :** Figure 5.2 shows two samples of fine-grain tasks, which mainly the

high partition of batches defined plus the model dimension. In the first example, the dataset is divided into 32 medium batches, and the model is of small size, while in the second example, the dataset is divided into 128 batches, and the model is of small-medium size with the granularity of these tasks classified as Fine-Grain. Whose performance characterization relates to it : of many tasks, and each task has a small size. Which composition is less intensive computation process, high-bottleneck tasks for managing CPU-GPU data transfers, and required multiple gradient-updates computation tasks.

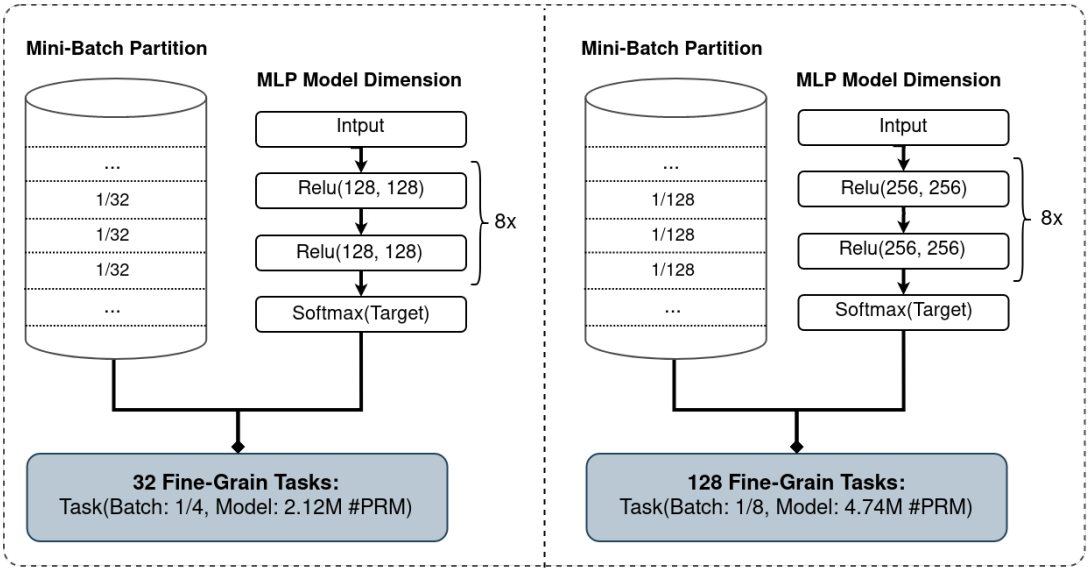



FIGURE 5.2 – Schematic sample of task granularity for fully connected neural networks.

### 5.3.2 Mapping the Dependency Graph on multi-GPUs and multi-Nodes

Regardless of the algorithm itself, the parallel section of an algorithm determines the granularity. Consequently, task granularity is characterized by three main components : algorithm structure, problem dimension, and platform selection, as shown in Figure 5.3. Commonly neural networks use data parallelism to train large datasets on multi-GPUs nodes. These algorithms add an essential factor from traditional parallel computing granularity : that seeks an appropriate rate between computation and communication task to include the synchronization task ; that in addition to minimizing the bottlenecks, has the responsibility for updating the weights and biases to decrease the loss function, which will determine a long or short convergence. The input data, model dimension, and the batch size determine the number of synchronization tasks per weight updated step, which is categorized as significant batch size increase the computational speed, and smaller batch size empirically improve better generalization.

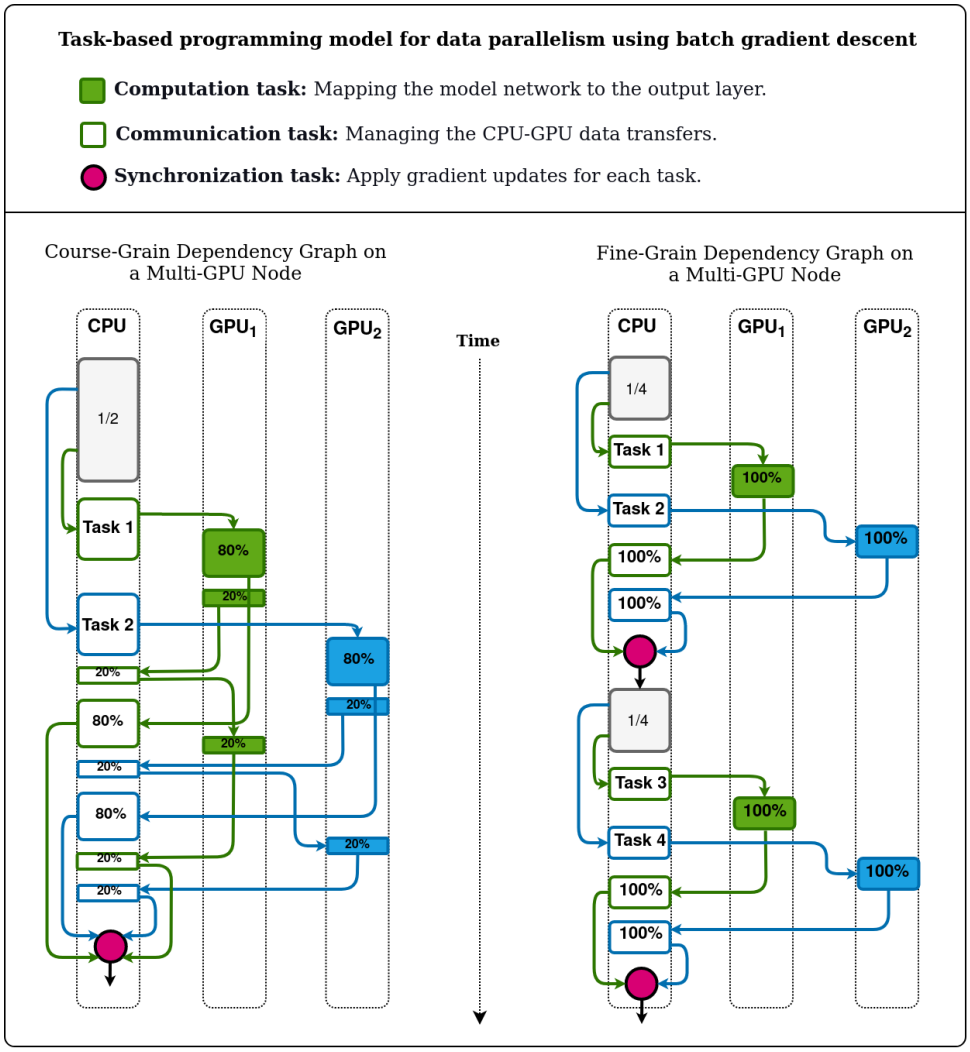



FIGURE 5.3 – Two scenarios where a bad task-partition could slow the training of neural networks.

### 5.3.3 Tasks-Granularity Problem Definition

The Task-Granularity optimization problem has the *batch size* as a crucial factor of the optimization space due that it affects both the statistical accuracy (generalization) and hardware efficiency (utilization). The objective function is designed to determine a good batch granularity, given the model and the worker numbers to minimize the convergence time and energy consumption for training a distributed neural network on a heterogeneous system.

For example, a typical supercomputer of the TOP500 [1] has 700 nodes with a hybrid architecture, and each node contains 2 CPUs Intel Xeon and 4 GPUs Nvidia Tesla V100 all connected with PCIe network cards. Each node has about 128GB DDR4 of volatile memory, four solid-state devices of 2TB, and the nodes are connected in a non-blocking

1. The TOP500 table shows the 500 most powerful commercially available computer systems known.

fat-tree using a dual-rail Mellanox EDR InfiniBand interconnect.

$$P_{h\_idle} = 2 * 16.2 + 4 * 20.0 + 3.8 + 7.3 + 2 * 9.2 \approx 141.9W \tag{5.1}$$

$$P_{h\_max} = 2 * 205 + 4 * 250 + 7 + 7.8 + 2 * 12.3 \approx 1449.4W \tag{5.2}$$

On the other hand, in this thesis, we execute the experiments on a mini-cluster of Nvidia Jetson-TX2 nodes since the Nvidia Jetson family does not have an integrated power sensor for obtaining the watts measurements for the Tegra processor. We could compute the energy consumption metric through the interpolation of the Jetson Tegra processor to obtain their watts measurements. The computing factors that determine the power consumption in watts are the memory uses, the cores' clock frequencies, the 'streaming multiprocessor' (GPU), and the bandwidth transfers. Therefore, DiagnoseNET integrates the enerGyPU monitor to enable the energy modeling and automatizes the power recording and storage of the registers ; in parallel, the deep neural networks are trained on the selected platform.

## 5.4 Measuring and Modeling Energy Efficiency at Runtime

A standard accepted metric used to help improve the energy efficiency of a supercomputer is called *Power Usage Effectiveness* (PUE) [Belady 08]. It measures the relationship between the *Total Facility Energy* which includes everything that supports the IT equipment as the energy associated with cooling, air movement, electricity transformers, lighting, and IT equipment ; divided by the amount of *IT equipment Energy* includes the energy associated with the computing system, like servers and switches.

Nevertheless, has emerged the metric called energy-to-solution $E_s$ [Hater 16, Gustavo Rostirolla 15]. It is used to estimate the energy required by a computing system for processing applications algorithms and training neural networks (computational tasks). $E_s$ is given by the integral of its instantaneous power draw, represented in the next equation :

$$E_s = \int_{t_0}^{t_0+\Delta t} P_s(\tau)\ d\tau, \tag{5.3}$$

Where $t_0$ is the time when the computational task is started, and $t_0 + \Delta t$ is the execution time spent for processing the computational task. $P_s(\tau)$ is the sum of the instantaneous power consumed by each of the computational resources of the system at

each time step $\Delta t(i)$, defined in follows the equation :

$$P_s(\tau) = \sum_{i=t_0}^{N} P_{\mathscr{H}}(i) * \Delta t(i), \tag{5.4}$$

Where $\mathscr{H}$ represents the computing system constituted by a set of physical machines $\vec{h}$ and a set of network switches $\vec{n}$, defined as $\mathscr{H} = \vec{h} + \vec{n}$. Each node $h$ can have diverse computational resources (components), like CPU, GPU, memory, storage, and network. Therefore, the power consumed by each node $P_h \in P_{\mathscr{H}}$, can be seen as a resultant vector of the power draw at each time step $\Delta t(i)$ of its computational resources, started by [Oleksiak 19] and can be expressed as follows :

$$P_h = \vec{P_{CPU}} + \vec{P_{GPU}} + \vec{P_{RAM}} + \vec{P_{io}} + \vec{P_{net}}, \tag{5.5}$$

However, the instantaneous power consumed per node $P_h(i)$, is dynamic during runtime and changes according to the computational use of its components, while each component is conditioned by two based states, shown in the equation 5.6. The idle power state $P_{h\_idle}$, represents the sum of the minimum power required by each component of the node to be on. While the active power state $P_{h\_active}$, corresponding to the sum of the intermediate power applied by each component to execute the computational task, This occurs between the component's highest possible power and the idle power.

$$P_h(i) = \begin{cases} t_0, & P_{h\_idle} \\ \Delta t(i), & P_{h\_idle} \leq P_{h\_active} \leq P_{h\_max} \end{cases} \tag{5.6}$$

Therefore, the energy efficiency goal is to determine and apply the optimal frequency at the *job level* the optimal frequency to all cores and nodes running the job to match the selected energy policy. This frequency is set by setting the appropriate policy, composed by the Dynamic voltage and frequency. According to the $Unified\ Extensible\ Firmware\ Interface$ (UEFI), the policies are :

— Efficiency-favor power mode maximizes the performance per watt efficient with a bias toward performance.

— Efficiency-favor performance mode optimizes the performance per watt efficiency with a bias toward performance.

By Ohm's law, the dynamic power consumed by a processor is given by $P_{CPU} = C * V^2 * f$, where $C$ is capacitance, $F$ is the frequency, and $V$ is voltage, which means the dynamic power increases quadratically with voltage and linearly with the frequency.

While the dynamic power consumption is dependent on the clock frequency, the leakage power is dependent on the CPU supply voltage. We will come back to the power leakage.

If we look at how voltage and frequency vary with ACPI $P_{states}$, we see that between $P_0$ state and the $P_{states}$ corresponding to the minimum voltage ($p_m$ with $m < n$ where $n$ is the highest possible $P_{state}$. Therefore, between $P_0$ and $P_m$, which is the range where a processor is executing workloads, can be approximated with : $P \approx C * F^3$. It shows dynamic power increases as the cube of frequency and how reducing the frequency when an application is running can significantly reduce the power consumption of a server.

Likewise, the theoretical peak performance per node is determined by the frequency, as : $CPU_{peak} = N_{core} * F_{CPU} * FLOPS$, where $N_{core}$ denotes the number of cores and $F_{CPU}$ is the core clock frequency. On the other hand, the GPU processor could represent, similar as the processor $GPU_{peak} = N_{SM} * F_{GPU} * FLOPS$. Therefore ;

1. To measure the energy consumed by all the workloads that have been executed.
2. One trivial way to minimize the power of a workload while running on a system is to reduce the processor's frequency.

## 5.5 Case of studies and Neural Architectures

### 5.5.1 Hyperparameter Search to Classify the Medical Task 1

A model space contains ($d$) hyperparameters and ($n$) hyperparameters configurations defined in Table 5.1 and the Table 5.2 shows the models by number of parameters. We have established some fixed hyperparameters and decided to tune the number of units per layer, the number of layers, and batch size, which are the hyperparameters that directly affect the computational cost. Each model was trained using *Adam* as an optimizer with a maximum of 40 epochs, and as a loss function is used the *Cross Entropy*.

According to the model dimension shown in the Table 5.2, we are found that it is possible to divide the models by fine, medium, and coarse grain. Which, the Figure 5.6 shows that middle-grain models from 1.99 to 8.29 million of parameters have a fast convergence in validation loss, and high accuracy levels for the majority of the 14 care purpose labels, in comparison with the other models who present a significant variation in accuracy and spent more epochs to convergence.

TABLE 5.1 – Search Space Model Descriptors.

| Hyperparameters ($d$) | Hyper. Configurations ($n$) | State |
|---|---|---|
| Learning rate | Adaptive L.R. starting | From : 0.001 |
| Activation function | relu, tanh, linear | Fixed : relu |
| Num. Units per layer | 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 | Search |
| Num. hidden layers | 2, 4, 8, 16 | Search |
| Regularization | Dropout : 0.6, 0.7, 0.8 | Fixed : 0.8 |
| Batch size | 24.576, 12.288, 6.144, 3.072, 1.536, 768 | Search |
| Num. of workers | 4, 6, 8, 10, 12 | Search |

TABLE 5.2 – Model Dimension Space in Number of Parameters (millions).

| | | Numbers of layers | | | |
|---|---|---|---|---|---|
| | | 2 | 4 | 8 | 16 |
| Neurons by Layer | 16 | | | | 0.24 |
| | 32 | | | 0.47 | 0.48 |
| | 64 | | 0.95 | 0.97 | 1.0 |
| | 128 | 1.89 | 1.93 | 1.99 | 2.12 |
| | 256 | 3.82 | 3.95 | 4.21 | 4.74 |
| | 512 | 7.76 | 8.29 | 9.34 | 11.44 |
| | 1024 | 16.05 | 18.15 | 22.35 | |
| | 2048 | 34.2 | 42.6 | | |
| | 4096 | 76.8 | | | |

## 5.6 Experiments and Results

The experimentation examines the relationship between a neural network task-granularity, mini-batch size, computational resources usage, and its reliance to predict the significant clinical categories by examining the balance between accuracy and energy consumption.

The CPU-GPU experiments used a server machine with a GPU Nvidia GTX-Titan X with 3072 CUDACores ; the device memory has $16GB$ GDDR5 $336.5GB/sec$ the memory bandwidth. In addition, we built a mini-cluster of 14-nodes Jetson TX2 interconnected by switch Ethernet for the distributed processing experiments. As a case of study, we are select the medical task to predict the medical care purpose from inpatients features recorded in hospital admission and clinical attention. We defined a feed-forward neural network hyperparameter space to generate a population of models and analyze what computational resources present an efficient balance between accuracy and energy consumption. The experimentation and results are divided into three stages :

1. Hyperparameter space to determine the best batch splitting,
2. Hyperparameter space to determine the depth and width of the neural networks,

3. Architectural parameter space to determine the worker's scalability.

### 5.6.1 Hyperparameter Space : Batch Splitting

To illustrate the impact of processing more gradient updates as a factor to fast convergence, consider the traditional feedforward neural network, parametrized with 2-hidden layers of $[4096, 4096]$ neurons per layer, *relu* is used as activation function, *Adam* such as optimizer and *_cross_entropy* as loss function. The clinical dataset uses 84.999 records for training and 4.950 records for validation.

The same DNN model has been executed using three different data batch partitions of $[24.576, 3072, 768]$ records by batch to measure the number of epochs needed to arrive at the convergence point, characterized by the minimum loss value, as shown in Figure 4. The largest batch partition of data requires more epochs to reach the convergence point since it generates data transfer latencies from host memory to GPU memory.

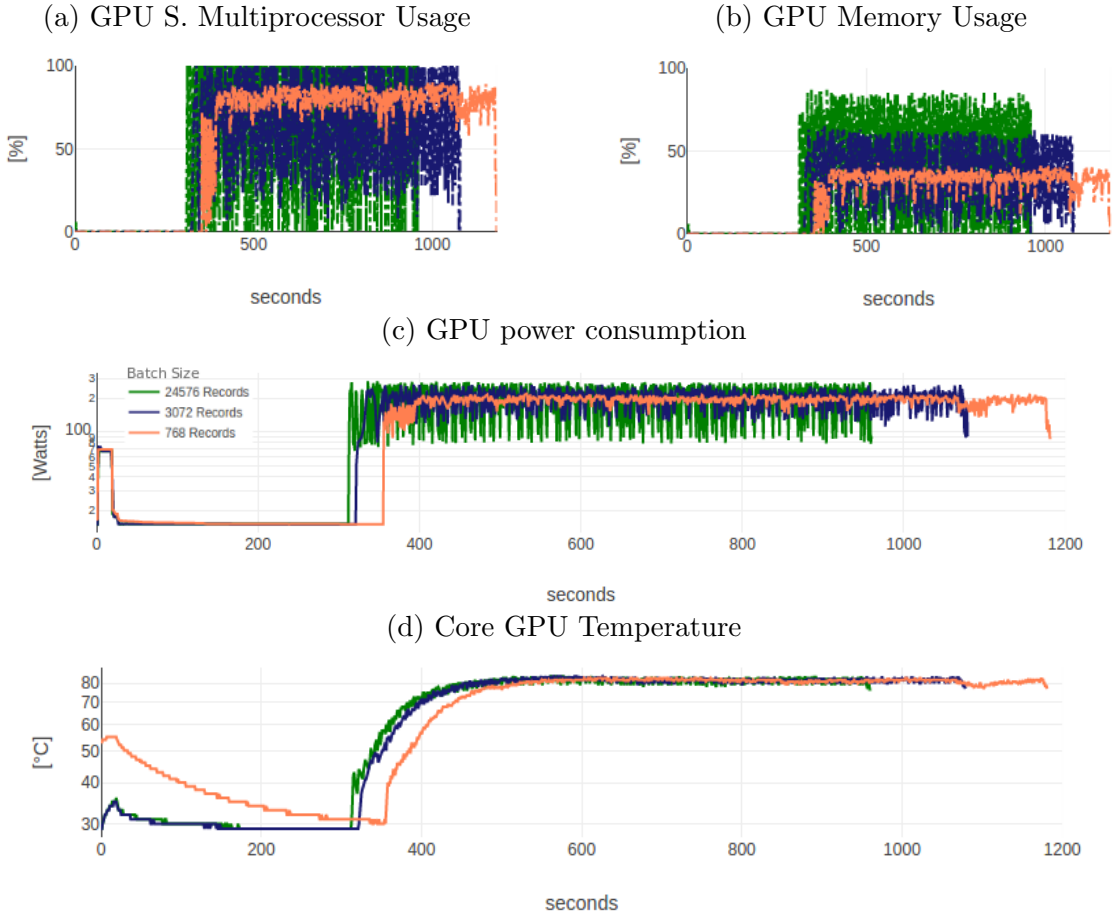


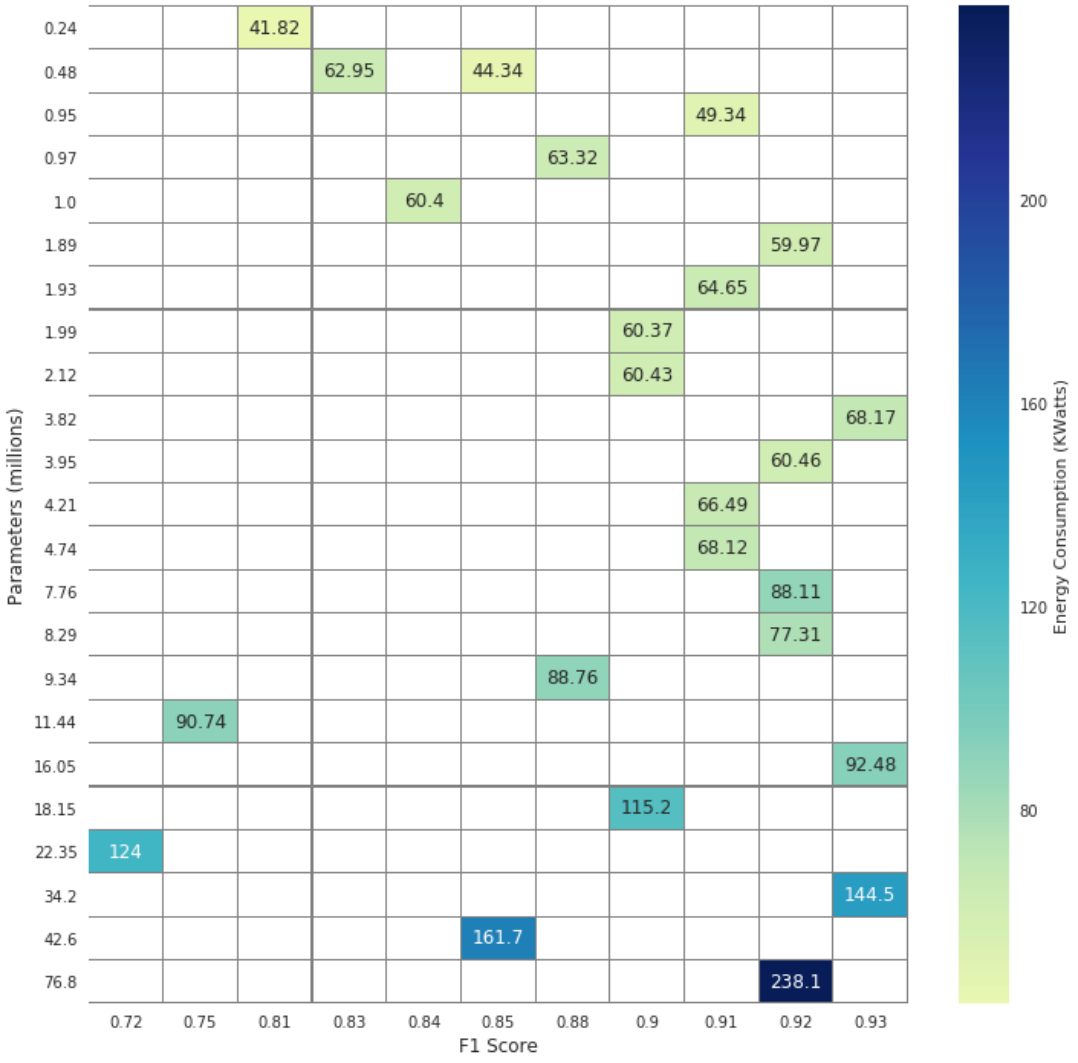

FIGURE 5.4 – GPU traces for training a MLP with different mini-batch size.

### 5.6.2 Hyperparameter Space : Determine the Depth and Width

The model selection is crucial in reducing energy consumption for deep learning networks. This analyses aim to characterize the computational performance of a feed-forward neural network in order to minimize the energy consumption and maintain a good accuracy of the model.

One of the parameters that affect the energy consumption and the execution time is the tensor dimension in the network (i.e., the number of units for each layer). For this reason, the experiments stretch the same number of neurons over multiple layers to generate different models. The hyperparameter configuration space uses stretch exploration over a feed-forward neural network. Each box represents a specific hyperparameter configuration (model) expressed into parameter numbers, and the warmer color corresponds to better performance between accuracy and energy consumption.

FIGURE 5.5 – Accuracy vs Energy Consumption

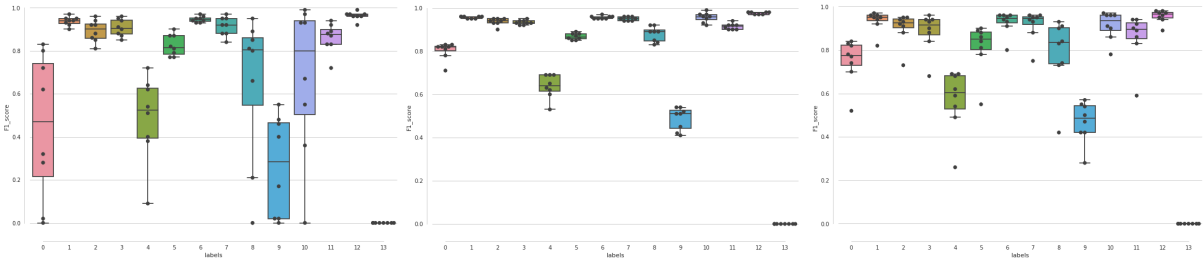


Stretching the same number of neurons affects the network's performance in terms of F1 score, execution time, and energy consumption, as shown in (Figure 6. Summary results). When we decrease the number of neurons per layer, the F1 score decreases, but the energy consumption decreases also. The aim is to find the number of neurons for which stretching over multiple layers will decrease energy consumption and time and affect the F1 score.

(a) Fine-grain test prediction by class.

(b) Middle-grain test prediction by class.

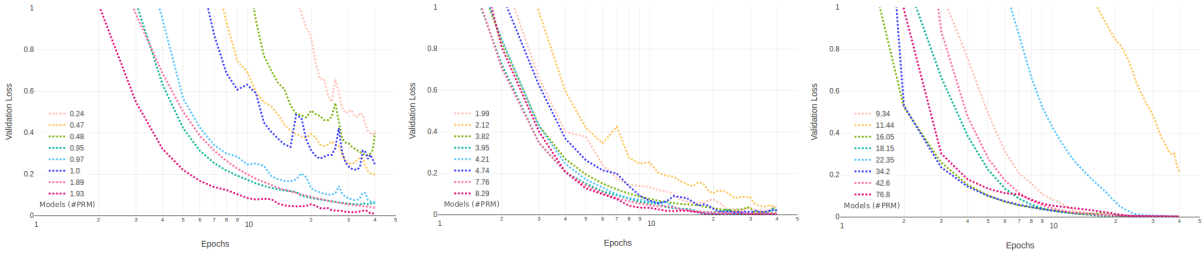
(c) Course-grain test prediction by class.




(d) Fine-grain convergence validation.

(e) Middle-grain convergence validation.

(f) Course-grain convergence validation.




(g) Fine-grain GPU P.C. distribution.

(h) Middle-grain GPU P.C.. distribution.

(i) Course-grain GPU P.C. distribution.

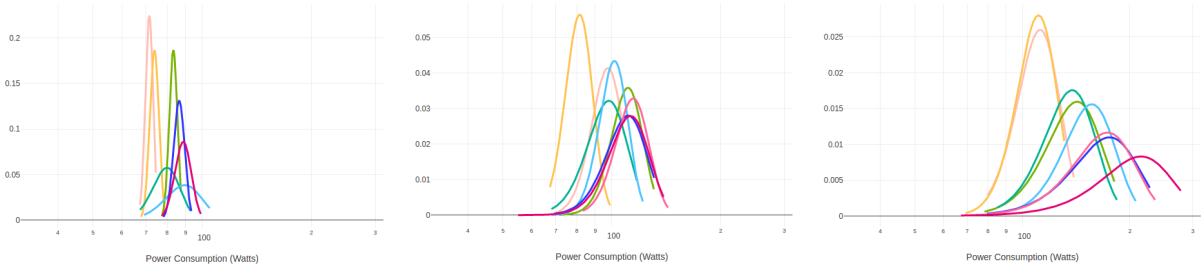



FIGURE 5.6 – Experiment results for training a feed-forward neural network, using the hyperparameter model-dimension space.

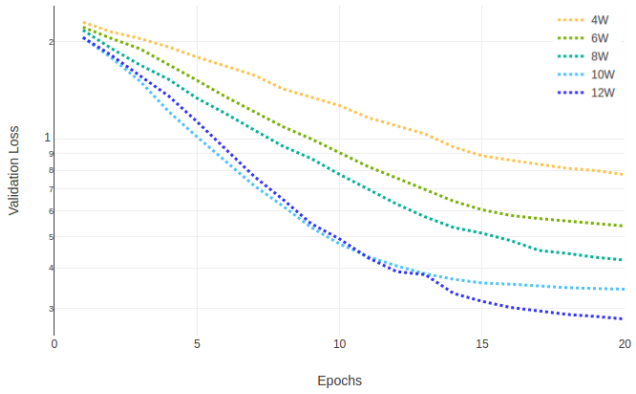
### 5.6.3 Architectural Parameter Space : Workers Scalability

The first approach uses data parallelism to training deep neural networks on a mini-cluster Jetson TX2. Where all replicas task read the same values for the current DNN

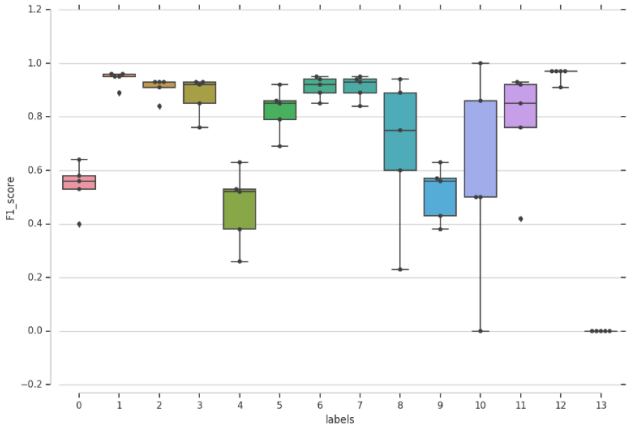
parameters, compute gradients in parallel, and theirs apply together [Abadi 16b]. To exploit the computational resources and the memory capacity of the Jetson TX2, the data is split acording with the number of Jetson nodes and each part of the assigned data is, again divided in micro-batches inside the each node to avoid memory constrains.

The distributed experiment results shows, when adding more workers for training a DNN model this generates more number of gradient updates and accelerate the convergence validation reducing the number epochs necessary. Since the performance has a great impact when the neural network has trained on memory or disk for each platform, for example : when we select a few workers number isn't possible to train the model exploits the memory, we need to use synchronous training on disk, due to the size of the batch. However, when we are select a huge number of workers the dataset batching is well balanced over the worker and is possible to train the model on synchronous training memory and improve the balance between communication and processing.

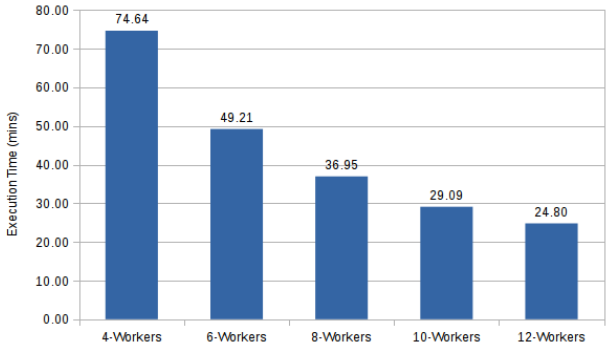
(a) Workers scalability convergence validation.




(b) Workers scalability test prediction by class.




(c) Execution time spent to training each model using different groups of workers.




(d) Cumulative comunnication banwidth between workers and the parameter server.


FIGURE 5.7 – Distributed experiment results for training a feed-forwared neural network on mini-cluster of Jetson TX2 nodes interconnected by switch Ethernet.

## 5.7 Summary

The task-based programming model for distributed training of deep neural networks allows more control for processing tasks such as the model components the gradient updates. At the same time, the mini-batch is tuned to adjust the size of the task according to the GPU memory size and minimize the communication bottleneck to allow efficient scalability between nodes. Likewise, adapting the number of records by batch or the model dimensionality to minimize the bottleneck of data transfer from host memory to device memory reduces the GPU idle status.

As a theoretical conclusion for tailoring the DNN model on distributed platforms, we can answer the main question ; How many workers are necessary to train a model to get the best performance in terms of accuracy and energy efficiency ratio ? It is necessary to split the dataset so that each portion delivered to each worker occupies 90% of the main memory. In order to do this, it is necessary to determine the amount of space required to allocate the one record or the full-dataset into the host memory ($dataset_{size}$), likewise we compute the DNN model size in host memory ($model_{size}$) according to with the hyperparameters and get the worker host-memory capacity ($worker_{MEM}$). According to the previous variables, we can calculate the micro-batch partition size such as : $batch_{size} = dataset_{size}/(worker_{MEM} - model_{size})$ and the number of workers as : $workers_{number} = dataset_{size}/batch_{size}$. After all, could be used synchronous data parallelism on distributed memory for training the DNN model using the *training_memory* modes to exploit the workers' capacities to early convergence.

# Chapitre 6

# Concluding Remarks

## 6.1 Summary of Main Results

The use of modern HPC systems to determine an optimal generalization model with Deep Neural Networks DNN in healthcare research is an expensive process for developing, training and financially, due to the cost of hardware and electricity or the cloud compute time and its carbon footprint required to fuel HPC systems. Health researchers and developers are looking for alternative options that allow health institutions to exploit locally the benefits of AI in healthcare, while the data privacy of each patient is preserved and the hardware and electricity cost is affordable. In this context, the motivation of this research is to develop a programming framework to improve the usability, portability and scalability of deep learning workflows over heterogeneous systems and, evaluate low-consumption computing architecture with minimal infrastructure requirements, to accelerate clinical risk-predictive models with an efficient balance between accuracy and energy consumption.

The main contributions of this thesis are the automatization and harmonization of communication paradigms and coordination methods for processing the gradient updates over heterogeneous systems into a framework called DiagnoseNET. In which, DiagnoseNET increase developer's productivity, facilitating the programming process to build and finetune a DNN, while its run-time abstracts the data locality, the micro batching and the distributed orchestration to scale the DNN model from a GPU workstation to multi-nodes. Likewise, DiagnoseNET made a workload characterization, collecting the computational platform characteristics memory usage, CPUs, GPUs clocks frequency and the energy consumption metrics while the DNN is executed on the target platform. This information is stored in a testbed for the programmer or the scheduler could be used to adjust the task granularity : model dimension and batch partition, according to the memory capacity and the number of nodes in the next execution.

Using a cluster of embedding nodes as the Jetson TX2 gives an intra-node advantage in data movements from host memory to GPU memory, due that the CPU and the Streaming Multiprocessors (GPU) are inside on-chip. However, the critical point for the acceleration and the scalability DNN models are concentrated in the communication internodes and the coordination method to compute the gradient updates. Consequently, this thesis evaluates the most commonly used communication paradigms (MPI and gRPC) and their coordination methods for training two different neural architectures : multilayer perceptron MLP and Convolutional Neural Networks CNN applied in two medical tasks. In which, the experimental result has shown strong scalability using the MPI asynchronous method to coordinate the gradient computations in both medical cases of study, accelerating the convergence time in 1,3 hours with 12 Jetson TX2 nodes from 11,1 hours with 2 Jetson TX2 nodes, given a reasonable time to explore a medium hyperparameter space and determine a good regularization-model.

We overcome several challenges in the implementation of DIagnoseNET as a modular framework, here some of the most representative :

1. Build an expression programming module to build dynamic neural networks, without rewriting the code for a new platform or execution modes.
2. Automatize the dataset splitting, placement and balancing the batch over the nodes and their memory constrains, having input sets with different dimensions as both cases of study.

As the scientific contribution non-obvious in this thesis, we had observed in the hyperparameter search process for the multilayer perceptron, when is found a model with n number of neurons and l number of layers, we could minimize the energy consumption and keeping the same accuracy stretching (esto deberia tener mayor relevancia) the same number of neurons over more layers.

## 6.2 Released Software

In the development of this thesis, several software modules were developed as well as the DiagnoseNET framework that allowed automating the workflow of medical artificial intelligence proposed and described in detail in the previous chapters. This section summarizes the additional modules developed from the medical domain to configure a minicluster to train a distributed DNN. All the software was released to the repository called IADBproject and licenced under the GNU General Public License v3.0.

1. **Patient Feature Composition :** Is a data-mining module for driving electronic health records to compose a digital patient representation. The objective of this module is to build a dynamic feature composition from a custom vocabulary to build a patient vector representation using a *document-term sparse matrix*. This module was used to extract the clinical dataset and develop risk prediction models of the health systems of the PACA region.
   It is available at https://github.com/IADBproject/patient-feature-composition.

2. **Build Embedded Clusters :** Is a cross-platform repository to configure an Nvidia Jetson mini-cluster to train distributed deep learning with TensorFlow and DiagnoseNET. The objective of this module is to automate and develop manuals to compile the necessary libraries and allow the reproducibility of the experiments.
   It is available at https://github.com/IADBproject/buildEmbeddedClusters and was developed in collaboration with Arno.

3. **Preprocesing MIMIC-III :** Is a preprocessing component for extracting and compose a dataset with patient's hospital admissions and their sequence of events (medical procedures) from the MIMIC-III Database, which comprises 61,532 intensive care unit stays.
   It is available at https://github.com/IADBproject/prepro-mimiciii and was developed in collaboration with Chanpiseth.

4. **ECG Convoluional Network :** Is a set of parallelism programs for training an ECG classification, comparing the same convolutional neural network to train the atrial fibrillation classification task using different libraries like TensorFlow, Keras and DiagnoseNET as well as the comunication protocols as gRPC and MPI.
   It is available at https://github.com/IADBproject/ECG and was developed in collaboration with Ziqing Du.

5. **DiagnoseNET :** Is an open-source framework designed into independent and interchangeable modules for scaling deep learning models over heterogeneous system architecture applied to medical risk prediction models. It automatizes in one expression API the neural architecture definition, the hyperparameter search, the data locality and batching, while the runtime coordinates the workers according to the execution modes through synchronous or asynchronous coordination gradient computations with communication protocols as MPI or gRPC, available for x86 and arm architectures. It is available at https://github.com/IADBproject/diagnosenet.

## 6.3 Research Perspectives

The vision of this thesis was to establish the main foundations for developing a learning healthcare system framework that systematically collects patient information to train deep learning models and implement clinical decision support systems throughout the organization to produce continuous improvement in care delivery.

In this direction, several opportunities we identified to extend the research from the patient from algorithm automation to patient modeling :

1. **Neural Architecture Search :** This thesis focused on the automation and scalability of deep learning workflows in heterogeneous systems to accelerate clinical risk prediction models with an efficient balance between precision and power consumption. However, developing neural networks for medical risk classification models often requires significant architecture engineering [Zoph 17]. For which the DiagnoseNET framework could be extended with population-based training search methods, such as the multiobjective function to scale the model along the dimension of the neural network, while minimizing the number of model parameters as was introduce in the compound scling method by [Tan 19].

2. **Federated Learning :** Another method that can be applied to extend this thesis is to improve asynchronous computing modes with the distributed coordination of gradients over a federated learning environment to allow collaborative and decentralized training of neural networks without sharing patient data. Whose models can benefit from data from different centers, regions, and patient phenotypes to generalize deep learning medical models that achieve human-level performanceon in decentralize clinical organizations. As recent contribution in federated deep learning to detect COVID-19 lung abnormalities from seven different multinational centers, using three centers to train the model and data from the other center to test and evaluate generalization performance [Dou 21].

## 6.4 Publications and Presentations

The following papers and conference presentations were published during the development of this thesis :

1. John A. García H. **Good Practices on Parallel and Distributed Programming for Training Neural Networks.** SC-CAMP, Supercomputing and Distributed Camp. 2020.

2. John A. García H., Frédéric Precioso, Pascal Staccini and Michel Riveill **DiagnoseNET : Automatic Framework to Scale Neural Networks on Heterogeneous Systems Applied to Medical Diagnosis.** ICITCS2020, International Conference on IT Convergence and Security. 2020.
3. Felix Mejia, John A. García H., Carlos Barrios, Michel Riveill **Analysis of Regularization in Deep Learning Models on Testbed Architectures.** Latin American High Performance Computing Conference, CARLA. 2020.
4. John A. García H., Frédéric Precioso, Pascal Staccini and Michel Riveill **Scalability Analysis of Mini-Cluster Jetson TX2 for Training DNN Applied to Healthcare.** High Performance Computing, NVIDIA GTC Europe Conference. 2018
5. John A. García H.., Frédéric Precioso, Pascal Staccini and Michel Riveill **Parallel and Distributed Processing for Unsupervised Patient Phenotype Representation.** Latin American High Performance Computing Conference, CARLA. 2018
6. John A. García H., Frédéric Precioso, Pascal Staccini and Michel Riveill **DiagnoseNET : Framework to Build Full Deep Neural Networks Workflow From Data-mining, Unsupervised Embedding and Supervised Learning.** User-Centric Network Workshop. 2017.

# Bibliographie


[Abadi 16a] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu & Xiaoqiang Zheng. *TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems.* CoRR, vol. abs/1603.04467, 2016.

[Abadi 16b] Martín et al Abadi. *TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems.* CoRR, vol. abs/1603.04467, 2016.

[Abirami 20] S. Abirami & P. Chitra. *Chapter Fourteen - Energy-efficient edge based real-time healthcare support system.* In Pethuru Raj & Preetha Evangeline, editeurs, The Digital Twin Paradigm for Smarter Systems and Environments : The Industry Use Cases, volume 117 of *Advances in Computers*, pages 339–368. Elsevier, 2020.

[Adithan 17] C. Adithan. *Principles of translational science in medicine : From bench to bedside, 2(nd) edition.*, Mar 2017.

[Anusha 21] K. Anusha, Nichenametla Rajesh, M. Kavitha & N. Ravinder. *Comparative Study of MongoDB vs Cassandra in big data analytics.* In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pages 1831–1835, 2021.

[Asai 15] Ryo Asai & Andrey Vladimirov. *Intel Cilk Plus for complex parallel algorithms : "Enormous Fast Fourier Transforms" (EFFT) library.* Parallel Computing, vol. 48, page 125–142, Oct 2015.

[Atalan 20] Abdulkadir Atalan & Cem Donmez. *Optimizing experimental simulation design for the emergency departments.* International Journal of Operations & Production Management, vol. 17, pages 1–13, 06 2020.

[Atasoy 19] Hilal Atasoy, Brad N. Greenwood & Jeffrey Scott McCullough. *The Digitization of Patient Care : A Review of the Effects of Electronic Health Records on Health Care Quality and Utilization.* Annual review of public health, vol. 40, pages 487–500, Apr 2019.

[Augonnet 09] Cédric Augonnet, Samuel Thibault, Raymond Namyst & Pierre andré Wacrenier. *STARPU : A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures*, 2009.

[Bai 18] Shaojie Bai, J Zico Kolter & Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*, 03 2018.

[Belady 08] Christian Belady, ANDY RAWSON, AMD PFLEUGER & DELL CADER. *Green grid data center power efficiency metric : PUE and DCIE.* The Green Grid, 01 2008.

[Bengio 11] Yoshua Bengio. *Deep Learning of Representations for Unsupervised and Transfer Learning.* In Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27, UTLW'11, pages 17–37. JMLR.org, 2011.

[Bengio 14] Yoshua Bengio, Aaron Courville & Pascal Vincent. *Representation Learning : A Review and New Perspectives*, April 2014.

[Boggs 15] D. Boggs, G. Brown, N. Tuck & K. S. Venkatraman. *Denver : Nvidia's First 64-bit ARM Processor.* IEEE Micro, vol. 35, no. 2, pages 46–55, Mar 2015.

[Bonawitz 19] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage & Jason Roselander. *Towards Federated Learning at Scale : System Design.* CoRR, vol. abs/1902.01046, 2019.

[Bramas 20] Bérenger Bramas & Alain Ketterlin. *Improving parallel executions by increasing task granularity in task-based runtime systems using acyclic DAG clustering.* PeerJ Computer Science, vol. 6, 2020.

[Braunstein 18a] Mark L. Braunstein. Health information exchange, pages 79–112. Springer International Publishing, Cham, 2018.

[Braunstein 18b] Mark L. Braunstein. *Healthcare in the Age of Interoperability : The Promise of Fast Healthcare Interoperability Resources.* IEEE pulse, vol. 9, pages 24–27, Nov-Dec 2018.

[Breiman 96] Leo Breiman. *Bagging Predictors.* Machine Learning, vol. 24, no. 2, pages 123–140, August 1996.

[Breiman 01] Leo Breiman. *Random Forests.* Machine Learning, vol. 45, no. 1, pages 5–32, October 2001.

[Buntin 11] Melinda Beeuwkes Buntin, Matthew F. Burke, Michael C. Hoaglin & David Blumenthal. *The benefits of health information technology : a review of the recent literature shows predominantly positive results.* Health affairs (Project Hope), vol. 30, pages 464–71, Mar 2011.

[Chahal 18] Karanbir Singh Chahal, Manraj Singh Grover & Kuntal Dey. *A Hitchhiker's Guide On Distributed Training of Deep Neural Networks.* CoRR, vol. abs/1810.11787, 2018.

[Chantry 12] A.A. Chantry, C. Deneux-Tharaux, G. Bal, J. Zeitlin, C. Quantin & M.-H. Bouvier-Colle. *Le programme de médicalisation du système d'information (PMSI) – processus de production des données, validité et sources d'erreurs dans le domaine de la morbidité maternelle sévère.* Revue d'Épidémiologie et de Santé Publique, vol. 60, no. 3, pages 177–188, 2012.

[Choi 16a] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart & Jimeng Sun. *Doctor AI : Predicting Clinical Events via Recurrent Neural Networks*, 2016.

[Choi 16b] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey & Jimeng Sun. *Multi-layer Representation Learning for Medical Concepts.* CoRR, vol. abs/1602.05568, 2016.

[Choi 16c] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart & Jimeng Sun. *GRAM : Graph-based Attention Model for Healthcare Representation Learning.* CoRR, vol. abs/1611.07012, 2016.

[de la Sante 5] Ministere de la Sante. *Loi N. 94-748 du 31 juillet 1991 portant reforme hospitaliere.*, Paris ; 1991 :10255.

[Dean 12a] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang & Andrew Y. Ng. *Large Scale Distributed Deep Networks.* In NIPS, 2012.

[Dean 12b] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang & Andrew Y. Ng. *Large Scale Distributed Deep Networks.* In NIPS, 2012.

[Delaney 15] Brendan C. Delaney, Vasa Curcin, Anna Andreasson, Theodoros N. Arvanitis, Hilde Bastiaens, Derek Corrigan, Jean-Francois Ethier, Olga Kostopoulou, Wolfgang Kuchinke, Mark McGilchrist, Paul van Royen & Peter Wagner. *Translational Medicine and Patient Safety in Europe : TRANSFoRm–Architecture for the Learning Health System in Europe.* BioMed research international, vol. 2015, page 961526, 2015.

[Dou 21] Qi Dou, Tiffany Y. So, Meirui Jiang, Quande Liu, Varut Vardhanabhuti, Georgios Kaissis, Zeju Li, Weixin Si, Heather H. C. Lee, Kevin Yu, Zuxin Feng, Li Dong, Egon Burian, Friederike Jungmann, Rickmer Braren, Marcus Makowski, Bernhard Kainz, Daniel Rueckert, Ben Glocker, Simon C. H. Yu & Pheng Ann Heng. *Federated deep learning for detecting COVID-19 lung abnormalities in CT : a privacy-preserving multinational validation study.* npj Digital Medicine, vol. 4, no. 1, page 60, 2021.

[Dünner 18] Celestine Dünner, Thomas P. Parnell, Dimitrios Sarigiannis, Nikolas Ioannou & Haralampos Pozidis. *Snap Machine Learning.* CoRR, vol. abs/1803.06333, 2018.

[Elsken 18] Thomas Elsken, Jan Hendrik Metzen & Frank Hutter. *Neural Architecture Search : A Survey.* arXiv e-prints, 2018.

[Esteva Andre 17] Esteva Andre, Kuprel Brett, Novoa Roberto A., Ko Justin, Swetter Susan M., Blau Helen M. & Thrun Sebastian. *Dermatologist-level classification of skin cancer with deep neural networks.* Nature, vol. 542, page 115, jan 2017.

[Esteva 19] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun & Jeff Dean. *A guide to deep learning in healthcare.* Nature Medicine, vol. 25, no. 1, pages 24–29, 2019.

[Ethier 18] Jean-François Ethier, Mark McGilchrist, Adrien Barton, Anne-Marie Cloutier, Vasa Curcin, Brendan C. Delaney & Anita Burgun. *The TRANSFoRm project : Experience and lessons learned regarding functional and interoperability requirements to support*

*primary care.* Learning health systems, vol. 2, page e10037, Apr 2018.

[Evans 16] R. S. Evans. *Electronic Health Records : Then, Now, and in the Future.* Yearbook of medical informatics, vol. Suppl 1, no. Suppl 1, pages S48–S61, May 2016.

[Foley 17] Thomas John Foley & Luke Vale. *What role for learning health systems in quality improvement within healthcare providers ?* Learning health systems, vol. 1, page e10025, Oct 2017.

[Garcia Henao 18] John Anderson Garcia Henao, Frédéric Precioso, Pascal Staccini & Michel Riveill. *Parallel and Distributed Processing for Unsupervised Patient Phenotype Representation.* In Latin America High Performance Computing Conference, September 2018.

[Goldstein 16] Benjamin A Goldstein, Ann Marie Navar, Michael J Pencina & John P A Ioannidis. *Opportunities and challenges in developing risk prediction models with electronic health records data : a systematic review.* Journal of the American Medical Informatics Association, vol. 24, no. 1, pages 198–208, 05 2016.

[Gustavo Rostirolla 15] Rodrigo da Rosa Righi Vinicius Facco Rodrigues Pedro Velho Edson Luiz Padoin Gustavo Rostirolla. *GreenHPC : a novel framework to measure energy consumption on HPC applications.* In 2015 Sustainable Internet and ICT for Sustainability, Madrid, Spain, April 14-15, 2015, pages 1–8, 2015.

[Hater 16] Thorsten Hater, Benedikt Anlauf, Paul Baumeister, Markus Bühler, Jiri Kraus & Dirk Pleiter. *Exploring Energy Efficiency for GPU-Accelerated POWER Servers.* In Michela Taufer, Bernd Mohr & Julian M. Kunkel, editeurs, High Performance Computing, pages 207–227, Cham, 2016. Springer International Publishing.

[He 15] Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun. *Deep Residual Learning for Image Recognition*, 12 2015.

[Ho 14] Joyce C. Ho, Joydeep Ghosh, Steve R. Steinhubl, Walter F. Stewart, Joshua C. Denny, Bradley A. Malin & Jimeng Sun. *Limestone : High-throughput candidate phenotype generation via tensor factorization.* Journal of Biomedical Informatics, vol. 52, pages 199–211, 2014. Special Section : Methods in Clinical Research Informatics.

[Hripcsak 13] George Hripcsak & David J. Albers. *Next-generation phenotyping of electronic health records.* JAMIA, vol. 20, no. 1, pages 117–121, 2013.

[Jensen Peter B. 12] Jensen Peter B., Jensen Lars J. & Brunak Søren. *Mining electronic health records : towards better research applications and clinical care.* Nature Reviews Genetics, vol. 13, page 395, may 2012.

[Jia 18] Zhihao Jia, Matei Zaharia & Alex Aiken. *Beyond Data and Model Parallelism for Deep Neural Networks.* CoRR, vol. abs/1807.05358, 2018.

[Jiang 17] Jie Jiang, Lele Yu, Jiawei Jiang, Yuhong Liu & Bin Cui. *Angel : a new large-scale machine learning system.* National Science Review, vol. 5, no. 2, pages 216–236, 02 2017.

[John A. Garcia H. 16] Esteban Hernandez B. Carlos E. Montenegro Philippe O. Navaux Carlos J. Barrios H. John A. Garcia H. *enerGyPU and enerGyPhi Monitor for Power Consumption and Performance Evaluation on Nvidia Tesla GPU and Intel Xeon Phi.* 2016.

[Kalé 11] Laxmikant V. Kalé. Charm++, pages 256–264. Springer US, Boston, MA, 2011.

[Kamel 18] Peter I Kamel & Paul G Nagy. *Patient-Centered Radiology with FHIR : an Introduction to the Use of FHIR to Offer Radiology a Clinically Integrated Platform.* Journal of digital imaging, vol. 31, no. 3, page 327—333, June 2018.

[Kennedy E.H. 13] Kennedy E.H., Wiitala W.L., Hayward R.A. & Sussman J.B. *Improved cardiovascular risk prediction using nonparametric regression and electronic health record data.* Medical Care, 2013.

[Keuper 16] Janis Keuper & Franz-Josef Preundt. *Distributed Training of Deep Neural Networks : Theoretical and Practical Limits of Parallel Scalability.* In Proceedings of the Workshop on Machine Learning in High Performance Computing Environments, MLHPC '16, pages 19–26, Piscataway, NJ, USA, 2016. IEEE Press.

[Kingma 14] Diederik P. Kingma & Jimmy Ba. *Adam : A Method for Stochastic Optimization.* arXiv e-prints, page arXiv :1412.6980, Dec 2014.

[Kingma 15] Diederik P. Kingma & Jimmy Ba. *Adam : A Method for Stochastic Optimization.* CoRR, vol. abs/1412.6980, 2015.

[Konecný 16] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh & Dave Bacon. *Federated*

*Learning : Strategies for Improving Communication Efficiency.* CoRR, vol. abs/1610.05492, 2016.

[Krizhevsky 12] Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* In NIPS, 2012.

[Lewis 13] Geraint Lewis, Heather Kirkham, Ian Duncan & Rhema Vaithianathan. *How Health Systems Could Avert 'Triple Fail' Events That Are Harmful, Are Costly, And Result In Poor Patient Satisfaction.* Health Affairs, vol. 32, no. 4, pages 669–676, 2013. PMID : 23569046.

[Liao 10] Chunhua Liao, Daniel J. Quinlan, Thomas Panas & Bronis R. de Supinski. *A ROSE-Based OpenMP 3.0 Research Compiler Supporting Multiple Runtime Libraries.* In Mitsuhisa Sato, Toshihiro Hanawa, Matthias S. Müller, Barbara M. Chapman & Bronis R. de Supinski, editeurs, Beyond Loop Level Parallelism in OpenMP : Accelerators, Tasking and More, pages 15–28, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Lin 20] Hong-Ling Lin, Ding-Chung Wu, Shu-Meng Cheng, Cheng-Jueng Chen, Mei-Chuen Wang & Chun-An Cheng. *Association between Electronic Medical Records and Healthcare Quality.* Medicine, vol. 99, page e21182, Jul 2020.

[Maharlou 18] Hamidreza Maharlou, Sharareh R Niakan Kalhori, Shahrbanoo Shahbazi & Ramin Ravangard. *Predicting Length of Stay in Intensive Care Units after Cardiac Surgery : Comparison of Artificial Neural Networks and Adaptive Neuro-fuzzy System.* Healthcare informatics research, vol. 24, no. 2, pages 109—117, April 2018.

[Mandel 16] Joshua C Mandel, David A Kreda, Kenneth D Mandl, Isaac S Kohane & Rachel B Ramoni. *SMART on FHIR : a standards-based, interoperable apps platform for electronic health records.* Journal of the American Medical Informatics Association, vol. 23, no. 5, pages 899–908, 02 2016.

[Mclachlan 18] Scott Mclachlan, Henry W. W. Potts, Kudakwashe Dube, Derek Buchanan, Stephen Lean, Thomas Gallagher, Owen Johnson, Bridget Daley, William Marsh & Norman Fenton. *The Heimdall framework for supporting characterisation of learning health systems.* Journal of Innovation in Health Informatics, vol. 25, 2018.

[Mikolov 13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado & Jeffrey Dean. *Distributed Representations of Words and Phrases and*

*Their Compositionality*. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[Miotto 16] Riccardo Miotto, Li Li, Brian A. Kidd & Joel T. Dudley. *Deep Patient : An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records*. SCIENTIFIC REPORTS, 2016.

[Nguyen 17] P. Nguyen, T. Tran, N. Wickramasinghe & S. Venkatesh. *mathttDeepr : A Convolutional Net for Medical Records*. IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 1, pages 22–30, 2017.

[Oleksiak 19] Ariel Oleksiak, Laurent Lefèvre, Pedro Alonso, Georges Da Costa, Vincenzo De Maio, Neki Frasheri, Victor Garcia, Joel Guerrero, Sébastien Lafond, Alexey Lastovetsky, Ravi Reddy Manumachu, Benson Muite, Anne-Cécile Orgerie, Wojciech Piatek, Jean-Marc Pierson, Radu Prodan, Patricia Stolf, Enida Sheme & Sébastien Varrette. *Energy aware ultrascale systems*. In Ultrascale Computing Systems, pages 127–188. Institution of Engineering and Technology, January 2019.

[Perros 17] Ioakeim Perros, Evangelos E. Papalexakis, Fei Wang, Richard W. Vuduc, Elizabeth Searles, Michael Thompson & Jimeng Sun. *SPARTan : Scalable PARAFAC2 for Large & Sparse Data*. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017, pages 375–384, 2017.

[Perros 18] Ioakeim Perros, Evangelos E. Papalexakis, Haesun Park, Richard W. Vuduc, Xiaowei Yan, Christopher deFilippi, Walter F. Stewart & Jimeng Sun. *SUSTain : Scalable Unsupervised Scoring for Tensors and its Application to Phenotyping*. CoRR, vol. abs/1803.05473, 2018.

[Poplin Ryan 18] Poplin Ryan, Varadarajan Avinash V., Blumer Katy, Liu Yun, McConnell Michael V., Corrado Greg S., Peng Lily & Webster Dale R. *Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning*. Nature Biomedical Engineering, vol. 2, no. 3, pages 158–164, 2018.

[Raghu 19] Maithra Raghu, Chiyuan Zhang, Jon M. Kleinberg & Samy Bengio. *Transfusion : Understanding Transfer Learning with Applications to Medical Imaging*. CoRR, vol. abs/1902.07208, 2019.

[Rahimian 18] Fatemeh Rahimian, Salimi-Khorshidi Gholamreza, Payberah Amir H., Tran Jenny, Ayala Solares Roberto, Raimondi Francesca, Nazarzadeh Milad, Canoy Dexter & Rahimi Kazem. *Predicting the risk of emergency admission with machine learning : Development and validation using linked electronic health records.* PLOS Medicine, vol. 15, no. 11, pages 1–18, 11 2018.

[Rajkomar Alvin 18] Rajkomar Alvin, Oren Eyal, Chen Kai, Dai Andrew M., Hajaj Nissan, Hardt Michaela, Liu Peter J., Liu Xiaobing, Marcus Jake, Sun Mimi, Sundberg Patrik, Yee Hector, Zhang Kun, Zhang Yi, Flores Gerardo, Duggan Gavin E., Irvine Jamie, Le Quoc, Litsch Kurt, Mossin Alexander, Tansuwan Justin, Wang De, Wexler James, Wilson Jimbo, Ludwig Dana, Volchenboum Samuel L., Chou Katherine, Pearson Michael, Madabushi Srinivasan, Shah Nigam H., Butte Atul J., Howell Michael D., Cui Claire, Corrado Greg S. & Dean Jeffrey. *Scalable and accurate deep learning with electronic health records.* npj Digital Medicine, vol. 1, no. 1, page 18, 2018.

[Rajpurkar 17] Pranav Rajpurkar, Awni Hannun, Masoumeh Haghpanahi, Codie Bourn & Andrew Y. Ng. *Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks.* 07 2017.

[Reid 05] Proctor P. Reid, W. Dale Compton, Jerome H. Grossman & Gary Fanjiang, editeurs. Washington (DC), 2005.

[Rieke 20] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust & M. Jorge Cardoso. *The future of digital health with federated learning.* npj Digital Medicine, vol. 3, no. 1, page 119, September 2020.

[Sánchez-Valle Jon 20] Sánchez-Valle Jon, Tejero Héctor, Fernández José María, Juan David, Urda-García Beatriz, Capella-Gutiérrez Salvador, Al-Shahrour Fátima, Tabarés-Seisdedos Rafael, Baudot Anaïs, Pancaldi Vera & Valencia Alfonso. *Interpreting molecular similarity between patients as a determinant of disease comorbidity relationships.* Nature Communications, vol. 11, no. 1, page 2854, 2020.

[Sandeep Kumar 20] E. Sandeep Kumar & Pappu Satya Jayadev. Deep learning for clinical decision support systems : A review from the panorama of

smart healthcare, pages 79–99. Springer International Publishing, Cham, 2020.

[Schwartz 20] Roy Schwartz, Jesse Dodge, Noah A. Smith & Oren Etzioni. *Green AI*, 2020.

[Shamout 20] Farah Shamout, Tingting Zhu & David Clifton. *Machine Learning for Clinical Outcome Prediction.* IEEE Reviews in Biomedical Engineering, vol. PP, pages 1–1, 07 2020.

[Smith 20] Maureen A. Smith, Peter A. Nordby, Menggang Yu & Jonathan Jaffery. *A practical model for research with learning health systems : Building and implementing effective complex case management.* Applied ergonomics, vol. 84, page 103023, Apr 2020.

[Solbrig 17] Harold R. Solbrig, Eric Prud'hommeaux & Guoqian Jiang. *Blending FHIR RDF and OWL.* CEUR Workshop Proceedings, vol. 2042, 2017. Funding Information : This study is supported in part by NIH grants U01 HG009450 and U01 CA18094. This work was conducted using the Prot g resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health. ; 10th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences, SWAT4LS 2017 ; Conference date : 04-12-2017 Through 07-12-2017.

[Srivastava 15] Nitish Srivastava, Elman Mansimov & Ruslan Salakhudinov. *Unsupervised Learning of Video Representations using LSTMs.* In Francis Bach & David Blei, editeurs, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of *Proceedings of Machine Learning Research*, pages 843–852, Lille, France, 07–09 Jul 2015. PMLR.

[Stanford Medicine 17] Stanford Medicine. *Harnessing the power of data in health*, 2017.

[Stanley Kenneth O. 19] Stanley Kenneth O., Clune Jeff, Lehman Joel & Miikkulainen Risto. *Designing neural networks through neuroevolution.* Nature Machine Intelligence, vol. 1, no. 1, pages 24–35, 2019.

[str 10] *Development of Inpatient Risk Stratification Models of Acute Kidney Injury for Use in Electronic Health Records.* Medical Decision Making, vol. 30, no. 6, pages 639–650, 2010. PMID : 20354229.

[Strubell 19] Emma Strubell, Ananya Ganesh & Andrew McCallum. *Energy and Policy Considerations for Deep Learning in NLP.* CoRR, vol. abs/1906.02243, 2019.

[Suyog Gupta 17] Wei Zhang Fei Wang Suyog Gupta. *Model Accuracy and Runtime Tradeoff in Distributed Deep Learning : A Systematic Study.* In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 4854–4858, 2017.

[Tan 19] Mingxing Tan & Quoc V. Le. *EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks.* CoRR, vol. abs/1905.11946, 2019.

[Thiébaut 19] Rodolphe Thiébaut & Sébastien Cossin. *Artificial Intelligence for Surveillance in Public Health.* Yearbook of medical informatics, vol. 28, pages 232–234, Aug 2019.

[Vincent 10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio & Pierre-antoine Manzagol. *Stacked denoising autoencoders : learning useful representations in a deep network with a local denoising criterion*, 2010.

[Wang 14] Xiang Wang, Fei Wang & Jianying Hu. *A Multi-task Learning Framework for Joint Disease Risk Prediction and Comorbidity Discovery.* In Proceedings of the 2014 22Nd International Conference on Pattern Recognition, ICPR '14, pages 220–225, Washington, DC, USA, 2014. IEEE Computer Society.

[White 15] Franklin White. *Primary health care and public health : foundations of universal health systems.* Medical principles and practice : international journal of the Kuwait University, Health Science Centre, vol. 24, pages 103–16, 2015.

[Winkel 21] David J. Winkel, Angela Tong, Bin Lou, Ali Kamen, Dorin Comaniciu, Jonathan A. Disselhorst, Alejandro Rodríguez-Ruiz, Henkjan Huisman, Dieter Szolar, Ivan Shabunin, Moon Hyung Choi, Pengyi Xing, Tobias Penzkofer, Robert Grimm, Heinrich von Busch & Daniel T. Boll. *A Novel Deep Learning Based Computer-Aided Diagnosis System Improves the Accuracy and Efficiency of Radiologists in Reading Biparametric Magnetic Resonance Images of the Prostate : Results of a Multireader, Multicase Study.* Investigative Radiology, vol. 56, no. 10, 2021.

[Xing 15] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar & Y. Yu. *Petuum : A New Platform for Distributed Machine Learning on Big Data.* IEEE Transactions on Big Data, vol. 1, no. 2, pages 49–67, 2015.

[Xu 21] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou & Lei Li. *A Survey on Green Deep Learning*, 2021.

[Ye Chengyin 19] Ye Chengyin, Wang Oliver, Liu Modi, Zheng Le, Xia Minjie, Hao Shiying, Jin Bo, Jin Hua, Zhu Chunqing, Huang Chao Jung, Gao Peng, Ellrodt Gray, Brennan Denny, Stearns Frank, Sylvester Karl G, Widen Eric, McElhinney Doff B & Ling Xuefeng. *A Real-Time Early Warning System for Monitoring Inpatient Mortality Risk : Prospective Study Using Electronic Medical Record Data*. Journal of medical Internet research, vol. 21, no. 7, pages e13719–e13719, jul 2019.

[Yu 15] Sheng Yu, Katherine P Liao, Stanley Y Shaw, Vivian S Gainer, Susanne E Churchill, Peter Szolovits, Shawn N Murphy, Isaac S. Kohane & Tianxi Cai. *Toward high-throughput phenotyping : unbiased automated feature extraction and selection from knowledge sources*. Journal of the American Medical Informatics Association, vol. 22, no. 5, pages 993–1000, 2015.

[Zhang 19] Yichi Zhang, Tianrun Cai, Sheng Yu, Kelly Cho, Chuan Hong, Jiehuan Sun, Jie Huang, Yuk-Lam Ho, Ashwin N. Ananthakrishnan, Zongqi Xia, Stanley Y. Shaw, Vivian Gainer, Victor Castro, Nicholas Link, Jacqueline Honerlaw, Sicong Huang, David Gagnon, Elizabeth W. Karlson, Robert M. Plenge, Peter Szolovits, Guergana Savova, Susanne Churchill, Christopher O'Donnell, Shawn N. Murphy, J. Michael Gaziano, Isaac Kohane, Tianxi Cai & Katherine P. Liao. *High-throughput phenotyping with electronic medical record data using a common semi-supervised approach (PheCAP)*. Nature Protocols, vol. 14, no. 12, pages 3426–3444, December 2019.

[Zoph 16] Barret Zoph & Quoc V. Le. *Neural Architecture Search with Reinforcement Learning*. CoRR, vol. abs/1611.01578, 2016.

[Zoph 17] Barret Zoph, Vijay Vasudevan, Jonathon Shlens & Quoc V. Le. *Learning Transferable Architectures for Scalable Image Recognition*. CoRR, vol. abs/1707.07012, 2017.