Parallel and Distributed Processing for Unsupervised Patient Phenotype Representation

GARCÍA H. John A.[1], PRECIOSO Frédéric[1], STACCINI Pascal[2], and RIVEILL Michel[1]

[1] Université Côte d'Azur, CNRS, Laboratoire I3S, Sophia Antipolis, France
henao@i3s.unice.fr, frederic.precioso@unice.fr, michel.riveill@unice.fr
[2] Université Côte d'Azur, CHU Nice, Nice
pascal.staccini@unice.fr

Abstract. The value of data-driven healthcare is the possibility to detect new patterns for inpatient care, treatment, prevention, and comprehension of disease or to predict the duration of hospitalization, its cost or whether death is likely to occur during the hospital stay.

Modeling precise patients phenotype representation from clinical data is challenging over its high-dimensionality, noisy and missing data to be processed into a new low-dimensionality space. Likewise, processing unsupervised learning models into a growing clinical data raises many issues, in terms of algorithmic complexity, such as time to model convergence and memory capacity.

This paper presents DiagnoseNET framework to automate patient phenotype extractions and apply them to predict different medical targets. It provides three high-level features: a full-workflow orchestration into stage pipelining for mining clinical data and using unsupervised feature representations to initialize supervised models; a data resource management for training parallel and distributed deep neural networks.

As a case of study, we have used a clinical dataset from admission and hospital services to build a general purpose inpatient phenotype representation to be used in different medical targets, the first target is to classify the main purpose of inpatient care.

The research focuses on managing the data according to its dimensions, the model complexity, the workers number selected and the memory capacity, for training unsupervised staked denoising auto-encoders over a Mini-Cluster Jetson TX2.

Therefore, mapping tasks that fit over computational resources is a key factor to minimize the number of epochs necessary to model converge, reducing the execution time and maximizing the energy efficiency.

Keywords: Health Care Decision-Making · Unsupervised Representation Learning · Distributed Deep Neural Networks

1 Introduction

A critical step of personalized medicine is to develop accurate and fast artificial intelligence systems with lower rates of energy used for tailoring medical care (e.g. treatment, therapy and usual doses) to the individual patient and predict the length and cost of the hospital stay. In this context, inferring common patient phenotype patterns that could depict disease variations, disease classification and patient stratification, requires massive clinical datasets and computationally intensive models [1, 2]. Thus, the complex structure, noisy and missing data from large Electronic Health Records (EHR) data became a core computational task to automated phenotype extractions [3].

In this paper, we describe the unsupervised learning method for mining her data and build low-dimensional phenotype representations using a mini-cluster with 14 Jetson TX2 in order to distribute training and to obtain a patient phenotype representation. This representation could be used as an input of supervised learning algorithms to predict the main purpose of inpatient care.

We present an application-framework called DiagnoseNET that provides three high-level features: The first allows the construction of a processing workflow to select and extract the data to be processed, to construct a binary representation, to reduce its dimensions through unsupervised learning and to process the data through supervised learning; the second is a data resource management to feeding the clinical dataset into the Jetson TX2 according to their memory capacity, while multiple replicas of a model are used for minimizing the loss function and third, an energy-monitoring tool for scalability analyses impact of using different batch size factor to minimize the number of epochs needed to converge and projected the energy efficiency measures.

2 Related work

In the past century, health research models were traditionally designed to identify patient patterns given a single target disease, where domain experts supervised definitions of the feature scales for that particular target and usually worked with small sample size, which was collected for research purpose [4, 5]. Nevertheless, in general, clinical data are noisy, irregular and unlabeled to directly discover the underlying phenotypes. This is supposed to be a limitation for this approach. Nowadays, computer science has facilitated the design and the implementation of emerging frameworks and practical approaches, offering different ways to extract valuable information as phenotypes [6].

To derive patients' phenotypes, it is necessary to extract the occurrence of their medical data (demographics, medical diagnoses, procedures performed, cognitive status, etc.). Although possibly the evolution of this information over time must be able to 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 [7], The generation of the different vectors

generally takes an important time. A couple of other possibilities are *nonnegative matrix factorisation* and *nonnegative tensor factorisation* for extracting phenotypes as a set of matrices, tensor candidates that show patients clusters linked on specific medical features and their date [8–10]. Other approaches use nonnegative vectors for embedding the clinical codes and use word representations as (skip-gram or Glove) to generate the corresponding visit representation [11].

Nevertheless, 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 [12], the application of employ it to produce patient phenotype representations can significantly improve predictive clinical model for a diverse array of clinical conditions as it was shown in deep patient approach [13].

Other derivative approaches use a record into a sequence of discrete elements separated by coded time, in which uses the unsupervised embedding Word2Vec to pre-detected the continuous vector space, them uses a convolution operation which detects local co-occurrence and pooled to build a global feature vector, which is passed into a classifier [14].

Another approach train a recurrent neural network with 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 [15].

However, these approaches to derive patients' phenotypes algorithms demand considerable effort in deploying preprocessing pipelines and data transformation, in which are built without taking into account the response time.

In this perspective, a large number of authors have explored scaling up deep learning networks, training well-known datasets focused on the impact of synchronization protocol and state gradient updates [16–18]. At the same time, other groups have been working on high-level frameworks to easily scale out to multiple machines to extend libraries for parameter management to allow more agile development, faster and fine tuning hyper-parameter exploration [19, 20]. All these developments are not applied to medical care and do not consider energy consumption.

Our aim is to construct a completed framework for scaling deep learning techniques in direction of extracting effective patient phenotype representations on low-power platforms for empowering the hospitals and medical centers their ability to monitor health, to early disease detection and manage to personalize treatments to specific patient profiles.

3 Material and Methods

Figure 1 shows the workflow implemented in the DiagnoseNET application. It highlights the different steps needed to build the phenotype whose goal is to create an equivalent but smaller representation for more effective clinical or medico-

administrative prediction. The first stage is to focus *mining EHR data* to drive a binary matrix of patient term documents from the clinical document architecture. The second stage *unsupervised representation* maps the patient's binary representation via an unsupervised stacked denoising auto-encoder to obtain a new latent space and identify the patient's phenotypic representations. And the third stage focuses on *supervised learning*, we use the latent representation of patients as an input for a random forest classifier, and as an initialiser for deep neural networks. The different results are compared to the binary representation of the patient.

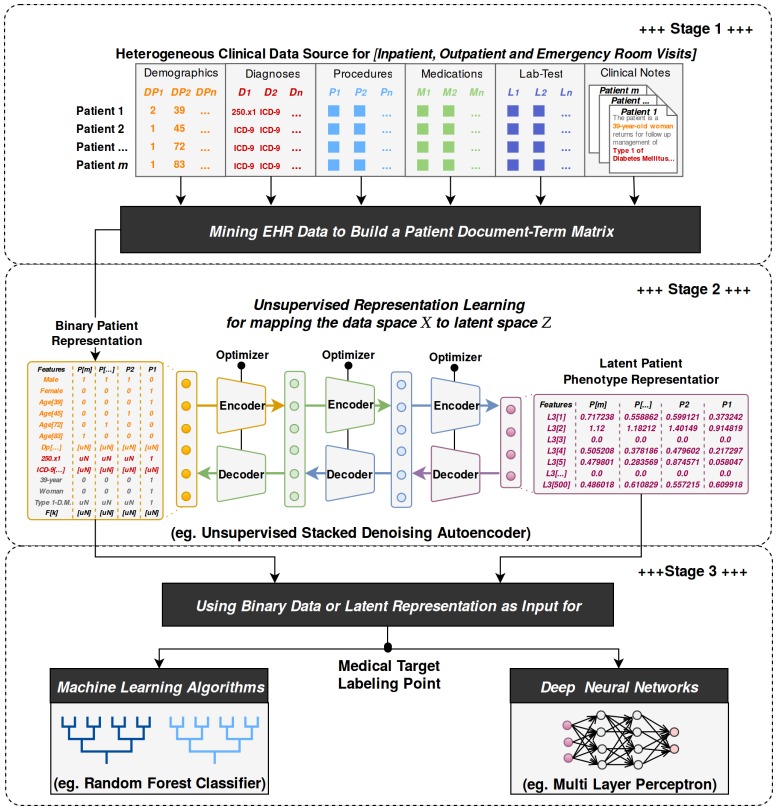


Fig. 1: Workflow scheme to automate patient phenotype extractions and apply them to predict different medical targets.

Mining EHR Data

The growing health-wide research is largely due to the clinical dataset is composed of a secondary usage of patient records collected in admission and hospital process [21]. Therefore the EHR is not a direct reflection of patient and their physiology but is a reflection of recording process inherent in healthcare with noise and feedback loops [22]. A data mining library has been built as a collection of functions to feature extraction and to build a patient document-term matrix from a clinical dataset composed of discrete objects as diagnosis in ICD-10 codes, medical acts in French CCAM codes and other derived objects as admission hospital details represented in codes established by the French agency ATIH and generated by the PMSI system to standardize the information contained in the patient's medical record. The collection functions are:

1. Clinical Document Architecture (*CDA*): identifies the syntax for clinical records exchange between the system PMSI and DiagnoseNET, through the new versions generate by the agency ATIH. The *CDA* schema basically 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.
2. Features Composition: Serialize each patient record and get the CDA object for processing all patient attributes in a record object.
3. Vocabulary Composition: Enables dynamic or custom vocabulary for selecting and crafting the right set of corresponding patient attributes by medical entities.
4. Label Composition: This function gets the medical target selected from the CDA schema to build a one-hot or vector representation.
5. Binary Record: Mapping the features values from record object with the corresponding terms in each feature vocabulary, to generate a binary corpus using Term-document Matrix.

Unsupervised Representation Learning

After the significant success of representation learning to encode audio, images and text with rich, high-dimensional datasets [**?**, 23–25]. In this work we extend the deep patient approach [13], in which all the clinical descriptors are grouped in patient vectors and each patient can be described by a high-dimensional vector or by a sequence of vectors computed in a predefined temporal window.

The vector collection is used to derive the latent representation of the patient via an unsupervised encoder network. At each step of the network, the coding matrix and the associated latent representation of a smaller dimension are obtained simultaneously as shown in Figure 2.

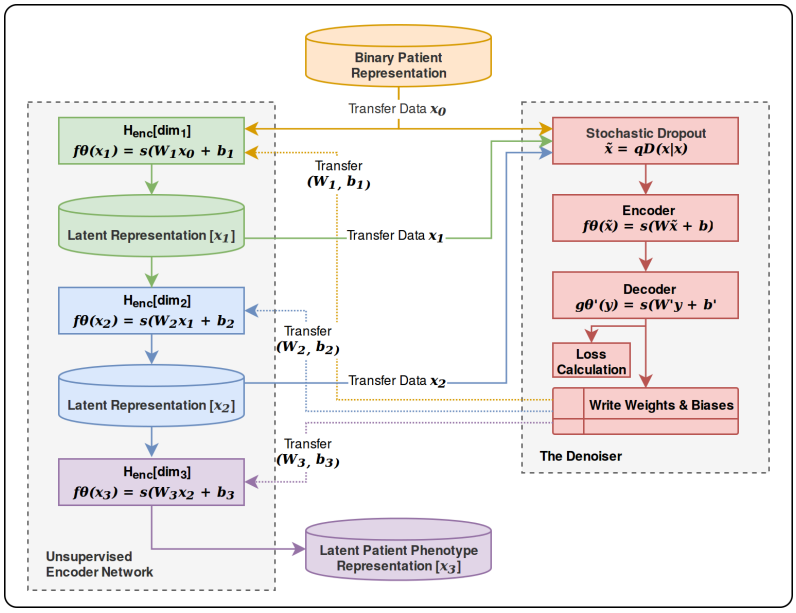


Fig. 2: Unsupervised encoder network for mapping binary patient representation x to latent patient phenotype representation z.

This unsupervised encoder network is composed of an *Unsupervised Stacked Denoising Autoencoders*: a deterministic mapping from cleaning 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 [26].

Previously each encoder was pre-trained to get a semantics representation parameters θ' by denoising autoencoder which was trained before, to obtain a robust representation $y = f\theta(\tilde{x})$ from a corrupted input $\tilde{x}$. This is represented by the next steps:

1. Applied dropout to corrupting the initial input x into $\tilde{x}$ the stochastic mapping $x \sim qD(x|x)$.
2. The corrupted input is mapped as traditional autoencoders to get a hidden representation $y = f\theta(\tilde{x}) = s(W\tilde{x} + b)$.
3. Reconstruct a schematic representation of the procedure $z = g\theta'(y) = s(W'y + b')$.
4. Where the parameters $\theta \wedge \theta'$ are trained to minimize the average reconstruction error over training set, to have z as close as possible to the uncorrupted input x.

5. And this share the new semantic representation parameters θ' 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.

Supervised Learning

It is well known that the general performance of machine learning algorithms - convergence time but also accuracy - generally depends on data representations. For this reason, the result of the unsupervised representation obtained in the previous step can be used as input of a standard supervised machine learning algorithms [12]. We therefore thought it relevant to compare the performances obtained by a random forest approach and a perceptron multi-layer approach for the different tasks allocated using either a latent representation of the phenotype or the binary representation at its origin.

Parallel and Distributed Processing for Traning DNN

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 3 describes the different hardware architectures and software stacks used in different experiments.

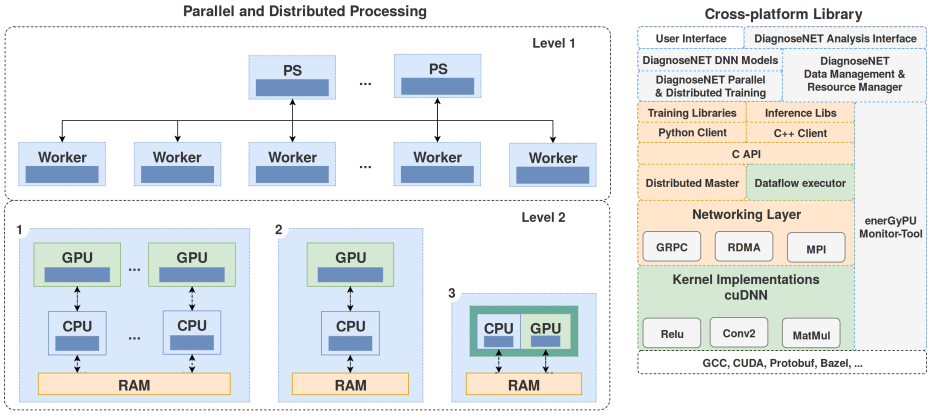


Fig. 3: Data resource management for training parallel and distributed deep neural networks and energy-monitoring tool.

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 in batch to take into account the available RAM memory 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 [27].

Dataset

The clinical dataset from admission and hospital services have an average of 85.801 inpatients records by year, with records of hospitals activities in South French Region (PACA Region: from Marseille to Nice). It contains information on morbidity, medical procedures, admission details and other variables, recorded retrospectively at the end of each week observed. This information may vary from one week to the next, depending on the evolution of the patient's clinical condition and management. As a case of study the clinical dataset taken 100.000 inpatients records by the year 2008 divided in 85.000 for training, 4.950 for validation and 10.050 for test with 11.466 features or clinical descriptors.

Medical Target: Classification of Care Inpatient Purpose

The first medical target, used for this paper, is to classify the main purpose of inpatient care, as Clinical Major Category (CMC), represented as 23 label coded in ICD-10 codes. The PMSI system can be assigned ICD-10 codes of the *Care Inpatient Purpose* as a high-level entry called Clinical Major Category used for billing procedures. Table 1 presents two examples of hospitalization that should be classified under the same CMC.

1.

Of course we are conducting further analysis of this dataset. Among these we can quote: the prediction of the duration of a hospital stay and the risk of death of the patient during this stay. The need to perform several analyses on the same dataset fully justifies the use of a latent representation provided that this allows the same analytical accuracy to be maintained. This is why, in an exploratory phase, all analyses are systematically made from the latent representation and from the binary representation. In the same way, all classifications or regressions are made using several supervised algorithms including random forest and RNN.

Table 1: Hierarchisation of diagnosis-related group to select the clinical major category as labels linked with the care inpatient purpose.

	Diagnosis-related Group	ICD-10 Codes	Definition
Patient 1	Morbidity Principal	R402	Unspecified coma
	Etiology	I619	Nontraumatic intracerebral hemorrhage, unspecified
Medical Target	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
Medical Target	Care Purpose	Z518	Encounter for other specified aftercare
Label used	Clinical Major Category	60	Other disorders

4 Experiments and Results

We have carried out various experiments using different batch sizes to examine the relationship between the convergence time of a network, its energy consumption and its ability to translate a patient's phenotype into a smaller latent space.

The last experiment carried out the possible characterization of the workload during the execution of a DNN network, which runs on a variable number of Jetson TX2 according to different batch sizes.

To estimate the efficiency (in terms of accuracy and energy consumption) we measure: the loss, the accuracy, the time and the number of gradient updates by epochs, as well, is recording the power consumption, GPU SM frequency, GPU memory frequency and is stipulated the minimum loss value as convergence point to stop the training process.

According to examine how fast can be trained, the DNN network and the minimum energy consumption require to arrive at the convergence point, gives a variable computational resources. We define three factors to evaluate the execution time and their energy efficiency:

1. The number of gradient updates as a factor to early model convergence.
2. The Model Dimensionality as a factor to generate quality latent representation.
3. The number of workers and task granularity as a factor to early model convergence on synchronous distributed processing.

The number of gradient updates as a factor 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 **??**. 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.

The low power consumption presented in the largest data batch partition (20.000) is generated by idle status on the GPU with a SM frequency of 847.49 *MHz* when the large data batch is transferred from the host memory to the device memory. We do not observe this idle phenomenon for the others batch size with a GPU SM frequency of 1071.97 and 1015.49 *MHz*. 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 6a. This window is extracted of the training of the AE model when it is executed using three different batch partitions.

The Model dimensionality as a factor to generate quality latent representation

This subsection studies the relationship between model complexity, network converge time and its reliance to generate a low-dimensional space as a latent patient phenotype representation.

Specifically the experiment comparison using an established-set of hyperparameters on three model variations for each network to compare the sigmoid vs. relu as activation function to generate the latent representation, using three variations of number of neurons per layer as $[4086, 2048, 768]$, $[2000, 1000, 500]$ and $[500, 500, 500]$.

The evaluation of accuracy is measured comparing the latent representation generated by each network model and used as input for random forest classification of the clinical major category on 23 labels and their energy efficiency.

1. The first network selected was a traditional fully_connected autoencoders with three hidden layers to generate the latent representation.
2. The second is an End_to_End network using three hidden fully connected autoencoders to generate the latent representation as input for the next four hidden multilayer perceptron.

3. Encoder_network using three hidden unsupervised stacked denoising autoencoders to initialize the next three hidden layers to encode and generate the latent representation.

The Number of workers and task granularity as a factor to early model convergence

The experiment analyzes the scalability for training a traditional autoencoders using different numbers of workers using an established-set of hyperparameters in two variations of the number of neurons per layer as $[2000, 1000, 500]$ and $[2048, 1024, 768]$. The different number of Jetson-TX2 Groups are:

1. 1 P. Server and 3 workers $->$ Batch size: [768, 1024]
2. 1 P. Server and 6 workers $->$ Batch size: [1024, 1420]
3. 1 P. Server and 8 workers $->$ Batch size: [1066]

In this case is shown 1 PS and 8 workers processing in data parallelism and training the unsupervised encoder network for mapping binary patient representation x to latent patient phenotype representation z. Where shows the synchronous cooperation of going to converge points in the Figure.

Table 2: Preliminary results for processing the unsupervised patient phenotype representation on the mini-cluster Jetson TX2.

AE Network	**1 PS and 3 Workers**		**1 PS and 6 Workers**		**1 PS and 8 Workers**		**1 CPU and 1 GPU**	
	Batch Fc.	Converge Time	Batch Fc.	Converge Time	Batch Fc.	Converge Time	Batch Fc.	Converge Time
Model 1	768	13.49 *mins*	**1024**	**9.95** *mins*	1066	10.18 *mins*		
Model 1	1024	11.90 *mins*	1420	10.51 *mins*				
Model 2	768	14.50 *mins*	1024	11.40 *mins*	1066	11.76 *mins*	768	3.97 *mins*
Model 2	1024	12.50 *mins*	1420	12.48 *mins*			1420	5.96 *mins*

5 Conclusions

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 a classification, depends on the application's ability to efficiently distribute data for analysis 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.

The current work on the platform aims to reinforce these main elements by comparing the performance that can be obtained for the other tasks obtained on various platforms.

The first platform is a standard computer with a GPU, the second platform is a 24 Jetson TX cluster connected by an Ethernet switch and the third platform is an array server consisting of 24 Jetson TX all connected via a Gigabit Ethernet through a specialized managed Ethernet Switch and marketed to Connect Tech.

The performance concerns the precision that can be obtained, the execution time of the model and the energy consumption necessary to obtain it.

Acknowledgments

This work is partly funded by the French government labelled PIA program under its IDEX UCAJEDI project (ANR−15−IDEX−0001). The PhD thesis of John Anderson García Henao is funded by the French government labelled PIA program under its LABEX UCN@Sophia project (ANR−11−LABX−0031−01).

References

1. Kathrin Heinzmann, Lukas Carter, Jason S. Lewis, and Eric O. Aboagye. Multiplexed imaging for diagnosis and therapy. 1, 09 2017.
2. Cheng Yu, Wang Fei, Zhang Ping and Hu Jianying. Risk Prediction with Electronic Health Records: A Deep Learning Approach, 2016.
3. Lasko TA, Denny JC and Levy MA. Computational Phenotype Discovery Using Unsupervised Feature Learning over Noisy, Sparse, and Irregular Clinical Data, 2013.
4. Development of Inpatient Risk Stratification Models of Acute Kidney Injury for Use in Electronic Health Records. *Medical Decision Making*, 30(6):639–650, 2010. PMID: 20354229.
5. Kennedy E.H., Wiitala W.L., Hayward R.A., and Sussman J.B. Improved cardiovascular risk prediction using nonparametric regression and electronic health record data. *Medical Care*, 2013.
6. Sheng Yu, Katherine P Liao, Stanley Y Shaw, Vivian S Gainer, Susanne E Churchill, Peter Szolovits, Shawn N Murphy, Isaac S. Kohane, and Tianxi Cai. Toward high-throughput phenotyping: unbiased automated feature extraction and selection from knowledge sources. *Journal of the American Medical Informatics Association*, 22(5):993–1000, 2015.
7. Xiang Wang, Fei Wang, and 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.
8. Joyce C. Ho, Joydeep Ghosh, Steve R. Steinhubl, Walter F. Stewart, Joshua C. Denny, Bradley A. Malin, and Jimeng Sun. Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of Biomedical Informatics*, 52:199–211, 2014. Special Section: Methods in Clinical Research Informatics.

9. Ioakeim Perros, Evangelos E. Papalexakis, Fei Wang, Richard W. Vuduc, Elizabeth Searles, Michael Thompson, and 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.
10. Ioakeim Perros, Evangelos E. Papalexakis, Haesun Park, Richard W. Vuduc, Xiaowei Yan, Christopher deFilippi, Walter F. Stewart, and Jimeng Sun. SUSTain: Scalable Unsupervised Scoring for Tensors and its Application to Phenotyping. *CoRR*, abs/1803.05473, 2018.
11. Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, and Jimeng Sun. Multi-layer Representation Learning for Medical Concepts. *CoRR*, abs/1602.05568, 2016.
12. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives, April 2014.
13. Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T. Dudley. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *SCIENTIFIC REPORTS*, 2016.
14. P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh. *mathttDeepr*: A Convolutional Net for Medical Records. *IEEE Journal of Biomedical and Health Informatics*, 21(1):22–30, 2017.
15. Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. GRAM: Graph-based Attention Model for Healthcare Representation Learning. *CoRR*, abs/1611.07012, 2016.
16. 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, and Andrew Y. Ng. Large Scale Distributed Deep Networks. In *NIPS*, 2012.
17. Janis Keuper and 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.
18. 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.
19. Li Zhang and Yufei Ren Wei Zhang" "Yandong Wang. Nexus: Bringing Efficient and Scalable Training to Deep Learning Frameworks. In *25th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2017, Banff, AB, Canada, September 20-22, 2017*, 2017.
20. Celestine Dünner, Thomas P. Parnell, Dimitrios Sarigiannis, Nikolas Ioannou, and Haralampos Pozidis. Snap Machine Learning. *CoRR*, abs/1803.06333, 2018.
21. Jensen Peter B., Jensen Lars J., and Brunak Søren. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13:395, may 2012.
22. George Hripcsak and David J. Albers. Next-generation phenotyping of electronic health records. *JAMIA*, 20(1):117–121, 2013.
23. 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.

24. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and 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.
25. Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised Learning of Video Representations using LSTMs. In Francis Bach and David Blei, editors, *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.
26. Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-antoine Manzagol. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, 2010.
27. 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, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, abs/1603.04467, 2016.

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

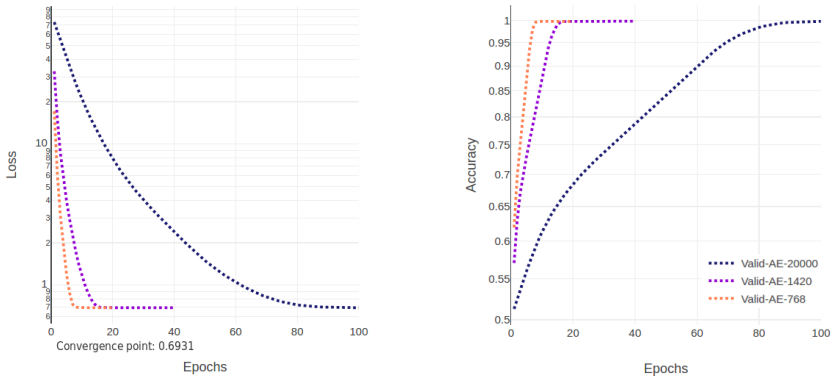
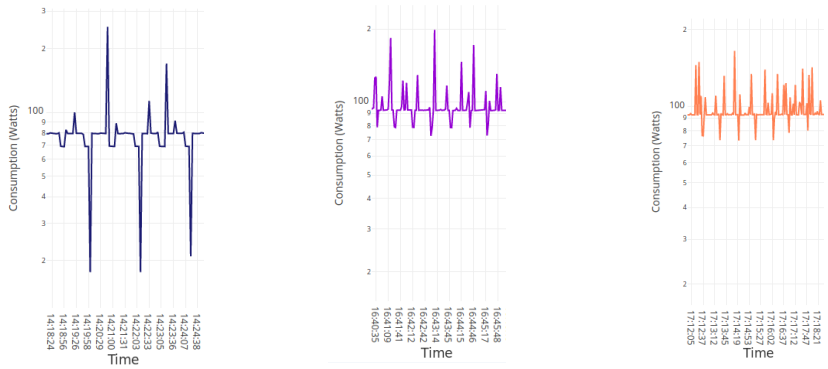


Fig. 6: Impact of GPU idle status generated by large data batch partition, consider the power consumption in a window of 6 minutes for the previous experiment.



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

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

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

Fig. 7: Comparison of different model dimensionality using sigmoid as function to generate the latent representation.

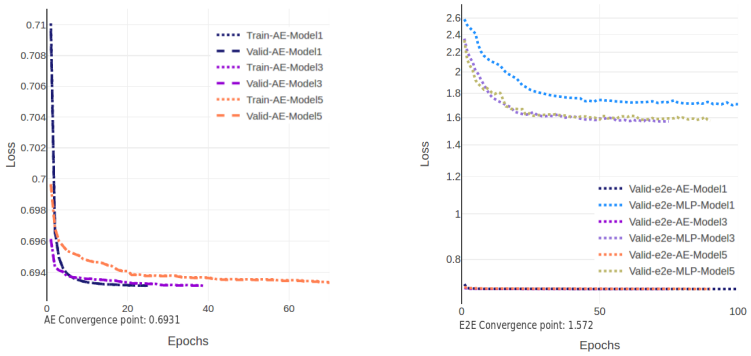


Fig. 9: Comparison of different model dimensionality using relu as function to generate the latent representation

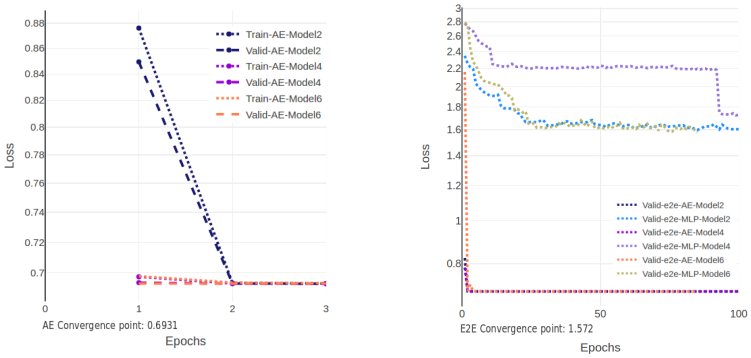


Fig. 10: Evaluation between network epoch time and their accuracy to classify 23 labels, using the latent representation as input for training random forest classifiers.

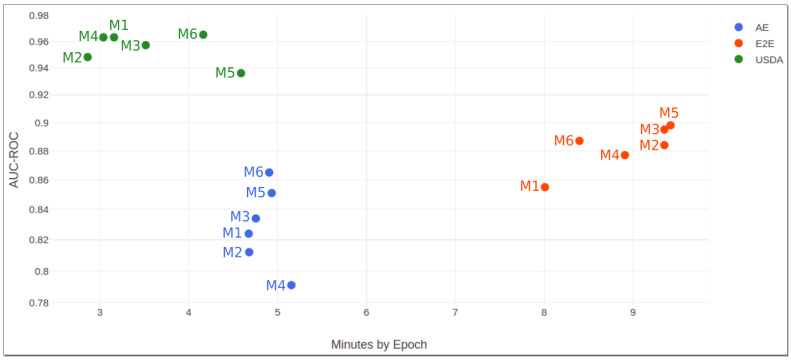
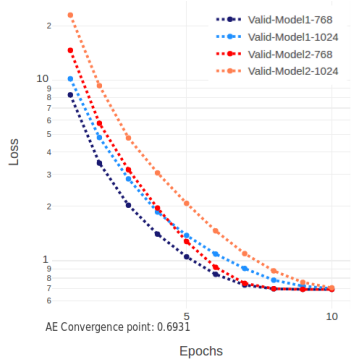
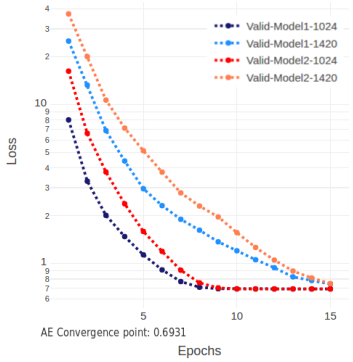


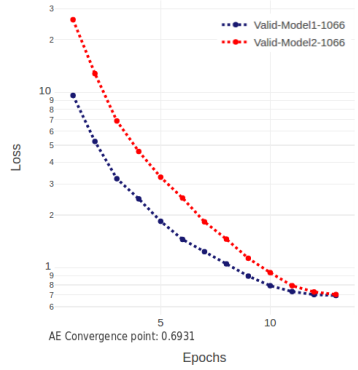
Fig. 11: Early convergence comparison between different groups of workers and task granularity for distributed training with 10.000 records and 11.466 features.



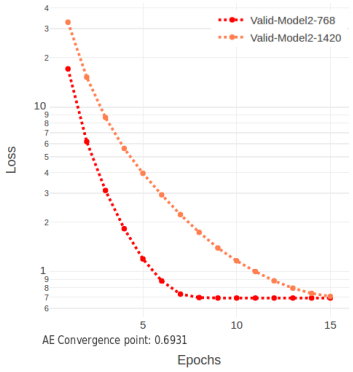
1.30 MN in average for processing one epoch on 1 PS & 3 workers.



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



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



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