Morpheo
traceable machine learning on hidden data

Mathieu Galtier[1] mathieu@rythm.co

Camille Marini[2] camille.marini@polytechnique.edu

[1]Rythm, Paris, France
[2]CMAP, École Polytechnique, Palaiseau, France

March 30, 2017

Abstract

Morpheo is a machine learning platform designed to attract data and algorithms on a large scale. It is designed to handle multiple data sources in a transfer learning approach. It aims at building state-of-the art prediction models in various fields where data are sensitive. Indeed, it offers strong privacy of data and algorithm, by preventing anyone to read the data, apart from the owner and the chosen algorithms. Computations in Morpheo are orchestrated by a blockchain infrastructure, thus offering total traceability of operations. Eventually, Morpheo aims at building an economic ecosystem around data prediction by channeling money from prediction requests to useful data and algorithms providers.

What is this document?
It is a white-paper: an introduction to the technology powering Morpheo.
It will evolve will the development of Morpheo and corresponds to our current vision of the project. This is v0.
Although Morpheo is developed for with a medical application in mind (sleep data analysis), both the white-paper and the technology are data agnostic.

Agenda:

- April 2017: open sourcing of all Morpheo components.
- Fall 2017: intermediary functional release of Morpheo with classical backend.
- Spring 2018: first stable release of Morpheo with blockchain backend.

1 Introduction

We live in a time of convergence: the progress in artificial intelligence is meeting an ever increasing computing power, and the collection of unprecedented amount of quality data. It is clear that this wave of digital intelligence will overwhelm many fields and in particular Healthcare. Not only is traditional medicine undergoing massive digitalization,

but the development of medical-grade sensing products used everyday by the consumers is fuelling the emergence of predictive precision medicine. Sleep is particularly rich and under-explored source of information. Indeed, analysing physiological signals such as brain and heart activity during sleep will lead to the definition of relevant biomarkers to characterize many pathologies from insomnia, sleep apnea to depression and neurodegenerative diseases. These accurate and rich data will provide a strong opportunity to better understand the physiology of each individual. In few years, we will be able to detect pathologies before it even happens, to treat earlier, faster and cheaper.

At the heart of this revolution lies Artificial Intelligence and Machine Learning. The central paradigm is to design models which learn by example: training algorithms on large datasets of examples builds models which can extrapolate to new data. With the appropriate algorithms [1], the quality of the models prediction is directly linked to the quality and quantity of data used for training. Thus, most applications require large datasets as a requirement for performance. When there is sufficient data the performance of such machine learning algorithm is often at the state-of-the-art and in medicine it is increasingly becoming better than human accuracy [2, 3].

However, these huge quantities of collected data often need harmonization: they come from multiple sources with different quality and sometimes they may be incomplete. Another strong constraint is that lots of data are not associated with a specific target. For instance, if we consider the problem of sleep stages classification, various devices may record different data types (electroencephalogram, electromyogram, accelerometer etc). Some of this data may have been manually annotated, and may not. To cope with this data heterogeneity a new promising methodology is emerging: transfer learning [4, 5]. With this approach, it is possible to merge datasets from different sources in a unique machine learning model which benefits from all data. In fact, several field are seing the emergence of a unique deep neural network as a basis for other learning architectures, notably imagenet is the international standard for image processing [6]. To our knowledge, there is no such basis model in sleep data analysis.

A key topic for collecting so many data is privacy, even more with sensitive data as in Healthcare. There are severe ethical issues regarding the massive collection of private data which can be used to characterize and maybe manipulate citizens. More precisely, the collection of identifying data is strictly regulated and there are many methods to properly anonymize a database [7]. Beyond ethics, there are countless situations where having a macroscopic information is useful, yet the different actors do not necessarily want to make their data accessible to others. Following the example of sleep data, machine learning algorithms require a big dataset to be efficiently trained, which require using data from different sleepers, which are usually recorded within a sleep laboratory and saved in an authorized system hosting personal data. Pooling these data would significantly increase the algorithms efficiency for various pathology detection, but there is a lot of meaningful friction due to the reluctance of the actors to share openly their data, even more in a world with non-trusted actors which could reuse data for a different purpose. Thus, it appears there is a strong need for a system putting sensitive data together, while guaranteeing their privacy and ownership.

Operations on sensitive data require a trusted environment. Too often, we give up our rights to our own data by tacitly agreeing on excessive 'terms of use'. Besides, we never know how our data are going to be stored, processed and transmitted. Regulators actually have no means to guarantee that our data are treated as claimed by industries. The blockchain technology provides a radical solution to this major problem: total trans-

parency and traceability of transactions. Beyond the simple exchange of crypto-currencies a new generation of blockchain protocols, such as Ethereum [8] and Hyperledger [9], is putting forward the notion of self-enforcing smart contracts: if you subscribe to a smart-contract its execution is not avoidable and rigorously in the terms defined. More precisely, a smart-contract is a standalone function living on a public ledger whose execution is entirely traceable. With these systems, one does not need to trust the actors, because the protocol is preventing them from doing things differently from what is claimed.

Morpheo is a platform living at the intersection between machine learning, transfer learning, and blockchain. It intends to build a trustful environment for building traceable machine learning models while keeping the most advanced privacy possible for data providers.

2 Design principles

Morpheo is a for-privacy platform which attracts algorithms and data in order to provide accurate and automatic detection of patterns. It is intended to be used as a software as a service (SAAS) in business models based on the valorisation of state-of-the-art prediction. The code is open source.

The basic idea of Morpheo is to gather algorithms and data, while guaranteeing total privacy and ownership, in order to learn efficient machine learning models to detect a specific pattern in new data. For now, it exclusively performs supervised learning tasks where the patterns of interest have been manually identified in many examples.

Morpheo is data and algorithm agnostic (even beyond Machine Learning), but it is clear that the scope of the data has to be limited and clearly specified for each deployment of the platform. For instance, the first deployment of Morpheo is dedicated to sleep medicine based on sleep physiological recordings such as the electroencephalogram.

The efficiency of the approach is directly linked to the quantity and quality of data and algorithms handled by Morpheo. Therefore, it is crucial that Morpheo offers an attractive environment such that data owner and algorithm developpers naturally tend to provide their data to the Morpheo platform. As detailed below, Morpheo implements strong privacy features and fair economic retribution to enforce this attractivity.

2.1 Data-centric Machine Learning

Morpheo is basically a Machine Learning backend which automates the learning and prediction of various algorithms on multiple data sets. The platform handles the algorithms source code and apply it to various hidden data remotely. At the core of the approach is the systematic update of a benchmark table, to identify which algorithm performs better on a specific prediction challenge. Even if an algorithm does not use Machine Learning, it can be compared to others in a transparent way.

Morpheo's design philosophy is data-centric: we believe it is better to invest effort on a few algorithms with lots of data, as opposed to many algorithms with a moderate amount of data. This partially mitigates the quadratic explosion of computations when there are both many algorithms and much data. Thus, Morpheo will deliberately favor the few best algorithms by feeding them with new data in priority. In the end, there may be only a single algorithm performing much better than the aggregate of the others.

Eventually, Morpheo will have the ability to handle seamlessly different sources of data for a single problem: it is designed as a tool for Transfer Learning. To aggregate as

much data as possible, Morpheo's infrastructure is adapted to manipulating various data formats, various distributions and potentially with missing data. It is up to the algorithm developers to leverage this flexibility with adapted Transfer Learning approaches.

2.2 For-privacy principles

Morpheo provides the highest degree of privacy for data and algorithm source code. Morpheo is designed specifically to handle sensitive data which must remain hidden from both the algorithm developers, but also the parties which participate to the deployment of the infrastructure. Morpheo implements the following high-level principles:

1. **Respect of ownership** of data and algorithms. No human or unwanted algorithm can ever access the data and/or algorithms of others.

2. **Transparency and traceability** of data and algorithm use. All the operations done securely on the data and algorithms, with the agreement of the owner, are logged in a public ledger.

2.3 Economic incentive

Morpheo intends to guarantee a fair retribution of data and algorithms. A Morpheo platform will generate revenue which will be split between data and algorithms providers. Indeed, requesting a prediction from the platform will be charged not only to cover the infrastructure cost but also to retribute useful data and algorithm.

This circulation of value will be backed by a blockchain infrastructure guaranteeing the consistency and immutability of transactions in time. The crypto-currency used will be exclusively backed by the specific prediction platform powered by Morpheo. Put differently, we are putting a price on automated prediction.

To make sure bad data are not blindly added to the platform, Morpheo transparently computes a global contributivity score for each data and algorithm which will be the basis for data retribution. The basic idea underlying the contributivity computation of a single datum is to compare the performance of various models having been learned with and without this datum. If adding the datum leads to a sharp increase in performance, then it will have a high contributivity.

3 Design implementation

3.1 User interactions

The user interface will be a cross-platform software made freely available to anyone. It will necessitate an internet connection for data transfer and security protocols.
The main actions that the user will be doing are:

- Visualizing the data. The precise visualisations will depend on the nature of the data. For instance, in the case of sleep physiologic signals we will represent the filtered signal on epoch of 30 seconds as is usually done in sleep science and medicine. We will also complement this view with several intermediary metrics computed from the raw data.

- Importing and exporting data. Due to the central necessity of gathering lots of data, it will be made as simple as possible, for instance with a simple drag and drop procedure. Morpheo will provide Compatibility with common data formats. When uploading data, the precise permission regimes will be set: can the uploaded data be used for any prediction problem or is it restricted to specific problems.

- Annotating data. Data annotations are necessary inputs because they will be used as teacher for training the algorithms. Thus, Morpheo will integrate an intuitive way to provide this information to the platform.

- Inputting algorithms. A command line interface and a feature rich text field will be provided to input algorithms. A detailed benchmark of all algorithms will be accessible to compare their efficiency.

- Requesting predictions. This is the central application of the platform: users will be able to pay for automatic prediction of specific patterns. This prediction will be displayed appropriately within the visualization mentioned above, so that the user can correct the algorithm when it has made a mistake.

The main use case we consider at this stage is for an expert who wants to accelerate its data annotation process. She would upload her data and ask for and automated prediction. Then by visualizing the data she would check the quality and, sometimes, correct the annotation. Thus, we can provide a time-saving service and improve our algorithm quality at the same time. This use case is particularly appropriate for the field of sleep medicine where night analysis is particularly time consuming.

3.2 Platform architecture

To guarantee data and algorithms privacy and retribution, while combining them to create state-of-the-art prediction algorithms, we have designed an advanced infrastructure. It is made of 4 different parts: (i) a local client running on data providers machines, (ii) a cloud storage, (iii) cloud computation resources and, (iv) a private blockchain network. The functioning of this architecture is illustrated in figure 1.

In this section and for simplicity, we will refer to both raw data, algorithms source code, and machine learning models (post-learning) simply as *data*. Indeed, their processing is very similar and we will assume it is identical except when explicitly mentioned otherwise.

A core design philosophy of Morpheo's architecture is that the *data* should remain encrypted as much as possible, in particular the *data* are always stored encrypted. Decryption keys are stored locally on the *data* owner's computer. With this simple infrastructure a user can already securely store and visualize her *data*. This motivates the definition of two components of the platform:

(i) *Data client* is a fat-client installed on users computers, which can en/decrypt *data*, upload/download encrypted *data* to *Storage*, visualize *data*, and store decryption keys.

(ii) *Storage* is a simple database where *data* are always encrypted. Although there would be no fundamental problem in distributing it, there are also no risk centralizing it since the host does not have access to *data*.

Yet, to perform Machine Learning tasks, such as prediction and learning, the *data* must be decrypted somehow. The chosen solution is to create secure disposable workers for each task. The encrypted *data* are downloaded on the worker, which then requests the keys to decrypt the *data* locally. Then it performs the given task and returns the results. Finally, it self destructs, erasing all clear *data* and decryption keys. This corresponds to the following component of the architecture:

(iii) *Compute* is creating, managing and certifying a swarm of workers which are ephemeral sorts of virtual machines which aim at consuming learning or prediction tasks. Most of the computing time of the architecture is concentrated in this component. For simplicity, we use centralized cloud-based resources for now.

To complete the architecture, it is necessary to trustfully control who can access the *data* and for what purpose. We need an infrastructure-wide certifying authority which associates *data* and workers, allowing the latter to decrypt the former. This is the trust bottleneck which cannot be compromised. In a way, all the architecture is built around this component which is the traceability, and security cornerstone. Such a component, detailed here, guarantees the traceability of all operations on the platform.

(iv) *Orchestrator* is transparently organizing the computations performed by the platform by combining valuable *data* to get the best performance possible. Most importantly, it creates and maintain a ledger of all learning operations in the (simplified) form:

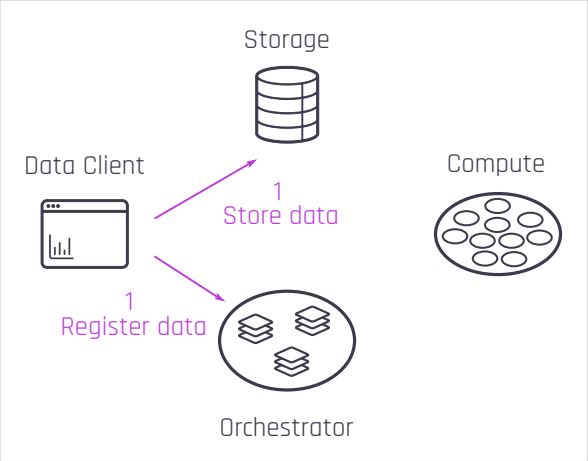
$$[\text{raw data id}, \text{model id}, \text{worker id}, \text{performance}]$$

where the 'id' are identification strings. Chronologically, it first creates such a quadruplet with an empty performance, serving as a job queue for *Compute*. Second, when the task has been consumed by *Compute*, a performance value is returned and added to the learning quadruplet. A similar ledger (without performance) is stored for prediction tasks. Third, it updates the contributivity of *data* directly from the learning ledger. *Orchestrator* runs on a private blockchain with smart contracts providing a trustful proof that the *data* have only been used for there agreed upon purpose.

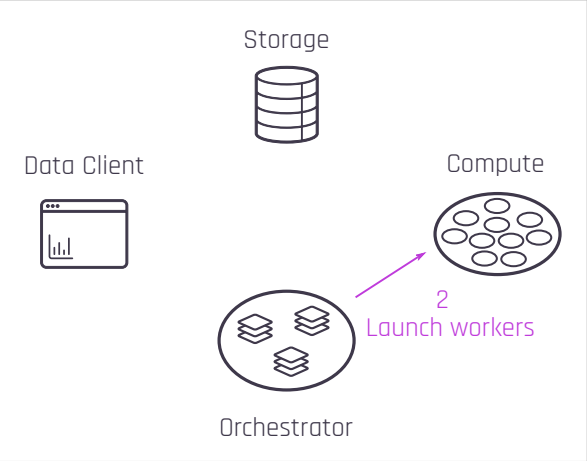
3.3 *Data* encryption

Data are always encrypted symmetrically on the *data* client before being transmitted to the rest of the platform. The symmetric key is always stored securely on the client side and will make it possible to decrypt data locally for visualization and annotation.

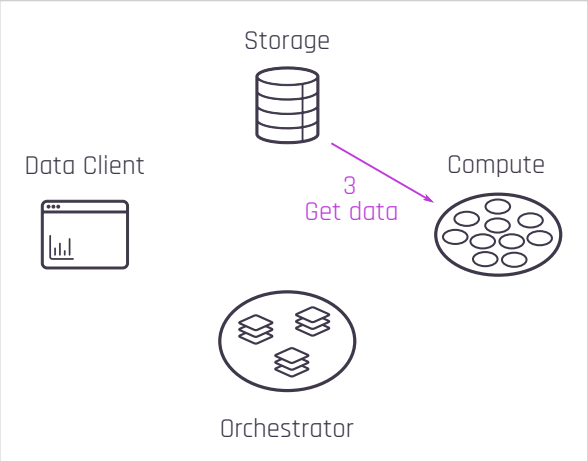
The decryption keys are then shared on the private network running the blockchain. Indeed, it cannot be assumed that the *data client* can always provide the *data* decryption keys to the other components of the platform (e.g. the *Compute* workers). Indeed, this would imply that the clients have to be always online to provide keys. To circumvent such a strong constraint, the fat-client can use a multi-party encryption scheme to spread its keys on the private network nodes so that the *data* can be decrypted if all the nodes provide partial keys to *Compute*. A simple sequential encryption of the *data* decryption key could be a valid multi-party encryption scheme. Alternatively, the symmetric decryption key could be split in several parts which are sent to different nodes in the network. In this improved version, the *data* owner can unplug its computer without disrupting



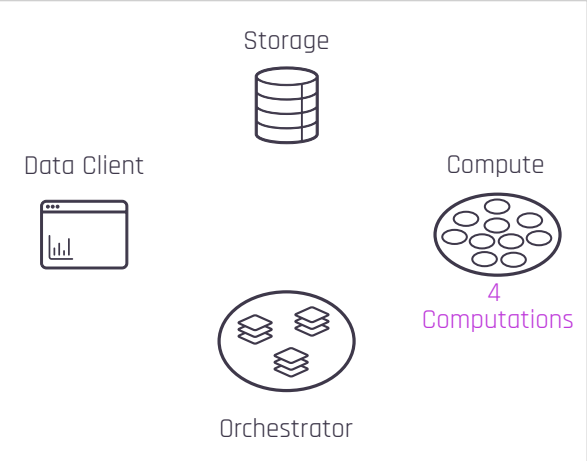
Step 1: A user uploads a new *data*. It is encrypted on the *data client* and sent to *Storage*. The data are registered on *Ochestrator* which adds several tasks to the ledgers.



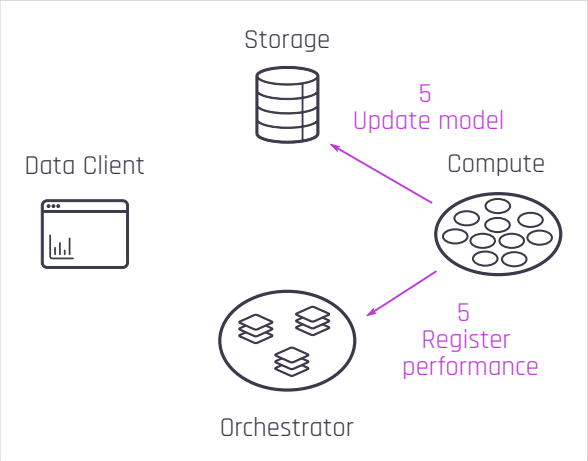
Step 2: A worker is launched to consume the task. The worker id (a public key) is written in the ledgers. The associated private is hosted on the worker to prove its identity.



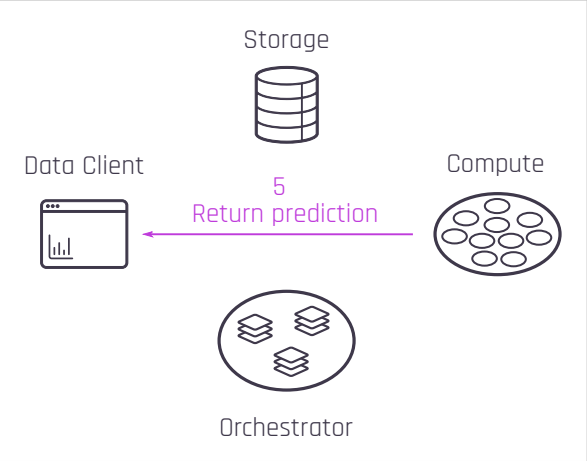
Step 3: The worker gets encrypted *data* and their decryption keys. Workers are authenticated via their id in the *Orchestrator* ledger.



Step 4: The worker decrypts the data and does learning / computes predictions with no connection to the rest of the world.



Step 5 (learn): The worker returns the performance and updates the model.



Step 5 (predict): The worker returns the prediction to the user without revealing it.

Figure 1: Workflow of a learning or prediction task. Step 1 to 4 are identical for both cases.

the platform, while being sure that the *data* cannot be retrieved by a single node. This means that the network private nodes not only run a blockchain protocol but also provide a service for securely receiving and sending the key parts. The *Orchestrator* plays a central role in the authentication of the workers to guarantee to the nodes that they are trustable: nodes only send their key part to a worker which can prove with its private key that its public key is written in one of the *Orchestrator* ledgers.

In Machine Learning, raw data and algorithms are combined into models that perform predictions. In Morpheo, it is considered that these models do not belong to anyone since they benefit from various *data*. *Data* are retributed through the contributivity mechanism, but the models themselves are simply seen as by-products of the platform, which remain encrypted and inaccessible to anyone, except the platform itself. In other words, anybody can request a prediction through the platform and use the models, but nobody can access them outside this framework. They are en/decrypted symmetrically on the *Compute* workers only. Similarly to the approach described in the previous paragraph, the encryption keys are spread on the private network to guarantee that no single node can extract the information, but that the processing has to be allowed by the *Orchestrator*.

3.4 Risk analysis

Although Morpheo is built to protect data privacy and guarantee fair retribution of *data*, zero risk does not exist. This section briefly tackles the identified risks related to the functioning of Morpheo and describes associated mitigation measures.

3.4.1 Upload of bad quality data

With good or bad intentions, users may upload data of bad quality with respect to a task which may overload the learning process. Indeed, the platform could spend too much time computing on bad data which would not improve the quality of the prediction.

To prevent such a harmful behaviour, Morpheo computes the contributivity of each data for various algorithms and uses it as a basis for planing of which data is used for which model. Therefore, bad data are quickly rated as poor and will not be used for many models.

Besides, Morpheo may ask the user to pay when she uploads lots of data. Not only would this cover the storage costs, but it would also be an incentive to prevent users from inputting bad data.

3.4.2 Upload of malicious algorithms

A malicious user could upload algorithms not meant to tackle a prediction challenge. For instance, such malicious algorithm could be written to retrieve decrypted data or to bring down some part of the platform. This is one of the major risk of the platform since we will run external code on servers. In any case, a malicious algorithm cannot get retroactive access to the data which limits the harm it could do.

To mitigate this risk, algorithms are run in a totally isolated environment within *Compute*. The possibility of interaction with the rest of the world will be restricted to the bare minimum. For instance there will be no network connection. Besides, computational resources allocated to train the algorithm are limited and the worker will be killed if the behaviour is abnormal.

3.4.3 Malicious *Compute* host

To guarantee privacy, *data* are only decrypted on a totally isolated environment on *Compute*. However, a malicious *Compute* host could ultimately get access to decrypted *data*. This risk is impossible to suppress entirely and mitigation could come from organizing transparent audits of the computing platforms and promoting the trust in the host of the *Compute*. Importantly, a malicious host cannot get retroactive access to the data which significantly limits its theft capabilities. Eventually, *Compute* could be distributed among many users reducing the outcome of a single malicious host.

Concerning the potential falsification of the output of a learning task (wrong model update or fake performances) or of a prediction task (fake prediction), this is made very difficult by the blockchain structure of the network. Indeed, by organizing a certain degree of redundancy in the computation, the *Orchestrator* could easily identify a malicious host and remove its prediction from the ledgers.

3.4.4 Loss of key by user or network

If keys that can be used to decrypt *data* are lost, *data* become inaccessible for visualization and learning, which could significantly harm the system performances. This risk is mitigated, in particular concerning the learning task, by scattering the key parts on the network and ensuring the redundancy of keys on the network. If the user or a node loses the keys, the rest of the network may still have the full information. In this case there is a trade off between robust accessibility and security of data: a lot of redundancy means data could be accessed by a small number of nodes (which is dangerous), but the data will be more easily if the keys are lost by one of the actors.

Regarding the private visualization of data and the situation where a user loses his private keys. We will not be able to retrieve it because the user has lost all authentication capabilities with the network. Therefore, we have to stress on the importance of keeping the private key in a safe place.

4 Discussion

We have introduced the platform Morpheo, which is a transparent machine learning backend with a strong privacy for data and algorithms. Its main goal is to provide automated and accurate pattern predictions in a specific field. Morpheo is built to perform transfer learning: various sources of data and reuse of models from a prediction problem to another. It systematically trains the algorithms and provides transparent benchmarks of models performances.

Morpheo is open source and promotes new, open, decentralized organisations of computation. It aims at being entirely transparent and secure so as to provide a trusted environment for data providers or users.

Morpheo introduces a retribution of data and algorithms. Indeed, to provide accurate predictions, lots of quality data and well tuned algorithms are the only currency. Thus, Morpheo prices these assets and virtually puts a price on predictions relative to the specific fields. Eventually, Morpheo aims at creating an economically attractive environment for data providers and prediction requesters. Consequently, this is likely to guarantee the prediction accuracy of the Morpheo approach.

Perspectives

Morpheo is under heavy development and there are a number of topics which are not developed for the time being and considered as perspectives.

Morpheo relies on a private blockchain for the *Orchestrator*, which means that the miners of the blockchain have to be identified as a consortium of actors. This provides a lot of flexibility for a very recent technology. Eventually, Morpheo may move towards a public blockchain infrastructure, but we believe it may be dangerous for now.

Similarly, *Storage* and *Compute* could be distributed and even decentralized by allowing anyone to provide his storage and computing power. We have chosen to consider full decentralization later not only for simplicity, but also because the security risks are too high.

In the present version of Morpheo, there is no search of hyperparameters for algorithms. Indeed, the optimization of the algorithms hyperparameters has to be done by the algorithm provider herself. This automatic search of such hyperparameters may be added in the future.

To provide even more security of data, we may also decide to have network wide cleaning procedure to regularly change the encryption keys of the data and algorithms.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Maxwell W Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, 2015.

[3] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[4] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[5] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] WG G29. Opinion on anonymization techniques. `https://cnpd.public.lu/fr/publications/groupe-art29/wp216_en.pdf`, 2014.

[8] Vitalik Buterin et al. Ethereum white paper. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2017.

[9] Hyperledger. Hyperledger white paper. `https://wiki.hyperledger.org/groups/whitepaper/whitepaper-wg`, 2016.