Deep learning methods for predicting flows in power grids : novel architectures and algorithms

Benjamin Donnot

NNT : 2019SACLS060

Thèse de doctorat

Deep Learning Methods for Predicting Flows in Power Grids: Novel Architectures and Algorithms

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

Ecole doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Orsay, le 13 Février 2019, par

BENJAMIN DONNOT

Composition du Jury :

M. Alexandre Gramfort Professeur, Inria (Parietal Team)	Président
M. Liva Ralaivola Professeur, Aix Marseille Université (LIS)	Rapporteur
M. Louis Wehenkel Professeur, University of Liège (Montefiore institut)	Rapporteur
M. Pierre Pinson Professeur, Technical University of Denmark (ELMA)	Examinateur
Mme Isabelle Guyon Professeur, Université Paris Saclay (LRI)	Directeur de thèse
M. Marc Schoenauer Professeur, INRIA (LRI)	Co-directeur de thèse
M. Antoine Marot Chercheur, RTE France	Co-directeur de thèse

Abstract

This thesis addresses problems of security in the French grid operated by RTE, the French “Transmission System Operator” (TSO). In this context, security entails keeping every piece of equipment in the grid within its defined operational limits. For instance, all transmission lines must transport less than a certain power to avoid the expansion of electrical conductors due to overheating. Several factors, including progress in sustainable energy, electricity market efficiency, and novel consumption patterns (e.g. electric cars, solar panels), push TSO's to operate the grid closer to its security limits, without (obviously) sacrificing a high quality of service. To this end, it is essential to make the grid “smarter”, or in other words propose more advanced algorithms than the currently used optimization methods, which rely on computationally inefficient simulators of the physical grid.

To tackle this issue, this manuscript explores the benefits of applying artificial neural networks, recently re-renamed “deep learning”, to this problematic. Such methods and other machine learning algorithms have seen a recent surge of interest given their capacity of learning from examples. In many areas of application that previously required a lot of human expertise and engineering, such as computer vision and natural language processing, these methods delivered performant solutions, which are distributed widely in the consumer market and have greatly improved upon previous state-of-the-art. This mini-revolution has happened very rapidly over the past few years, largely due to the increase in computing power (particularly the availability of relatively affordable graphics cards), allowing researchers to train in a matter of days these "data hungry" algorithms. After training, the resulting systems are generally more effective and efficient than other baselines handcrafted by human engineers.

In this context, following the footsteps of other pioneers in this field who have explored applications of neural networks to power systems, we propose **novel deep learning algorithms and architectures** to assist the decisions of human operators (TSO dispatchers). Our goal is to offer, in real time, a reasonable set of actions that dispatchers can choose from, to resolve or prevent problems from occurring on the grid. One of the first alleys to be explored is to imitate past actions of dispatchers. To

that end, RTE has a large collection of historical grid states that we have been fortunate enough to have access to for our experimental work. Our main contributions are:

- **Data preparation:** Preparing data for training is a daunting task. RTE stores complete snapshots of the grid, including when and where power is produced and consumed, as well as grid *topology* (pattern of line interconnections). Since topology changes are the preferred flexibility used by dispatchers for cost-effectiveness reasons, our first contribution has been to extract from real historical data relevant impactful actions of dispatchers (as opposed to maintenance maneuvers). In [10] we introduced a novel counterfactual method: "What if a change didn't occur?"

- **Guided Dropout:** The main algorithm we developed, "*guided dropout*", allows us to emulate physical simulators of power flows, providing fast and accurate approximations [14, 13, 12]. The inputs to the system are "injections" (productions and consumptions) and a topology indicator (not the full pattern of inter-connections). Our system generalizes not only with respect to new injections (generalization) but also with respect to new topologies it has never encountered during training (super-generalization). This can be interpreted as an instance of zero-shot transfer learning (generalization to new unseen situations).

- **"At scale" applications:** We have explored various impactful applications of our algorithm in the context of power transmission. We established the viability of the approach with numerical experiments demonstrating computational feasibility on grids of realistic sizes.

In a nutshell, the "guided dropout" algorithm allows us to predict the consequences on power flows following a grid willful or accidental modification (i.e. action or contingency). We tackle this issue by separately encoding the two types of inputs we have, **injections** and **topologies**, via two separate channels: (1) Continuous data (injections = productions and consumptions) are introduced in a standard way, via a neural network input layer; (2) Discrete data (grid topologies) are encoded directly **in** the neural network architecture. The neural network architecture is then dynamically modified based on the power grid topology, i.e. each topology corresponds to a pattern of connections in the neural network. This is achieved by switching on or off the activation of hidden units depending on the state of the power grid. Compared to other approaches, the main advantage of this technique lies in its ability to predict the flows even for previously unseen grid topologies. It is able to extrapolate the results of unary actions seen during training when combining actions at test time. We explored two types of scenarios in our applications: predictive mode and preventive mode.

- **Predictive mode:** The "guided dropout" algorithm, introduced in [14], predicts power flows in high voltage power grids after contingencies (accidental changes in topology, e.g. due to line rupture in a storm). Power flow could be predicted up to 99% precision on standard study cases (using the MAPE - Mean Absolute percentage error) with a 300 times speedup compared to physical grid simulators based on Kirchoff's law. The neural network could also accurately predict power flows for unseen contingencies, without detailed knowledge of the grid structure. In this sense, it is a big step towards emulating the physical simulator for a given power grid without its full description, which can be seen as another advantage over the simulator since it requires this complete information to compute the flows.

- **Preventive mode:** In [13] we showed that *guided dropout* can be used to rank contingencies that might occur in the order of severity. This may be used by operators to prioritize pro-active studies and concentrate on the most critical eventualities. In this application, we demonstrated that our algorithm obtains the same risk as currently implemented policies while requiring only 2% of today's computational budget. We train on only 0.5% of the total grid configurations, thus showing that the algorithm can handle grid cases never seen before. This last framework was further refined in [12] by complementing our machine learning approach with targeted simulations carried out by a physical simulator (such as those currently used in operational processes). In this last paper, we demonstrated that we can accurately predict the risk [1] taken when operating a grid state in a given topology with a precision higher than 95% in cases for which this overall computation was previously infeasible.

In conclusion, we have demonstrated that machine learning, and in particular deep learning, could be used to accurately predict power flows given productions and consumptions as well as a grid topology and contingencies on the designated grid. This fast approximation can be used to replace or to complement physical simulators when computational efficiency is a concern and loss of precision is not critical given uncertainties in the data. The fact that the proposed neural network architecture does not need a full description of the grid elements (physical properties as well as observed connectivity) is a desirable property since it is hard to know it exactly in advance before real-time operations.

[1]See *E. Karangelos and L. Wehenkel, "Probabilistic reliability management approach and criteria for power system real-time operation," in Power Systems Computation Conference (PSCC),2016, IEEE, 2016, pp. 1–9.* Equation 3 for a formal definition of this risk.

Résumé en français

Cette thèse porte sur les problèmes de sécurité sur le réseau électrique français exploité par RTE, le gestionnaire de réseau de transport électrique (GRT). Dans ce contexte, la sécurité consiste à maintenir chaque équipement du réseau dans ses limites opérationnelles. Par exemple, toutes les lignes de transport doivent transporter moins qu'un certains courant pour éviter la dilatation des conducteurs électriques due à surchauffe. Plusieurs facteurs, notamment les progrès en matière d'énergie renouvelable, l'efficacité du marché de l'électricité ainsi que les nouveaux modes de consommation (voitures électriques, panneaux solaires), poussent les GRT à exploiter le réseau électrique plus proche de ses limites de sécurité, sans sacrifier une qualité élevée du service. Pour ce faire, il est essentiel de rendre leréseau "plus intelligent". Ceci peut être effectué via l'utilisation d'algorithmes plus avancés que les méthodes d'optimisation actuelles s'appuyant principalement sur des simulations physiques, inefficaces sur le plan informatique.

Ce manuscrit explore les avantages de l'application des réseaux de neurones artificiels, récemment rebaptisés "*deep learning*", à cette problématique. De telles méthodes ont récemment connu un regain d'intérêt en raison de leur capacité d'apprentissage à partir d'exemples: dans de nombreux domaines d'application qui nécessitaient auparavant beaucoup d'expertise humaine et d'ingénierie, comme la vision par ordinateur et le traitement du langage naturel. Ces méthodes ont fourni des solutions performantes qui sont largement diffusées et qui se sont considérablement améliorées par rapport à l'état actuel des connaissances. Cette révolution s'est produite très rapidement au cours des dernières années, en grande partie grâce à l'augmentation de la puissance de calcul (en particulier la disponibilité de cartes graphiques relativement abordables), permettant aux chercheurs de former en quelques jours ces algorithmes "avides de données". Après la formation, les systèmes qui en résultent sont généralement plus efficaces et efficients que d'autres systèmes de référence fabriqués à la main par des ingénieurs humains.

Dans ce contexte, à l'instar d'autres pionniers dans ce domaine qui ont exploré les applications des réseaux de neurones artificiels à d'autres aplications (vision par ordinateur, compréhension du langage naturel), nous proposons des algorithmes et des architectures d'apprentissage profond novateurs pour aider les opérateurs humains

(dispatcheurs) à prendre des meilleures décisions. Notre objectif est d'offrir, en temps réel, un ensemble raisonnable d'actions parmi lesquelles les dispatcheurs peuvent choisir, pour résoudre ou prévenir les problèmes sur le réseau électrique. Pour ce faire, RTE dispose d'une importante collection d'états du réseau électrique auxquels nous avons eu la chance d'avoir accès pour nos travaux expérimentaux. Nos principales contributions sont :

- **Préparation des données:** La préparation des données pour l'apprentissage est une tâche ardue. RTE stocke des instantanés complets du réseau, y compris quand et où l'électricité est produite et consommée, ainsi que la *topologie* du réseau (schéma d'interconnexions de lignes). Puisque les changements de topologie sont la flexibilité préférée des répartiteurs pour des raisons de coût, notre première contribution a été d'extraire des données historiques réelles les actions pertinentes des dispatcheurs. Dans [10] nous avons introduit une nouvelle méthode contrefactuelle : "Et si un changement ne s'était pas produit ?"

- **Guided Dropout:** L'algorithme principal que nous avons développé, "'*guided dropout*"', nous permet d'émuler des simulateurs physiques de flux d'énergie, fournissant des approximations rapides et précises [14, 13, 12]. Les entrées dans le système sont des "injections " (productions et consommations) et un indicateur topologique (et non le schéma complet des interconnexions). Notre système généralise non seulement en ce qui concerne les nouvelles injections (généralisation) mais aussi en ce qui concerne les nouvelles topologies qu'il n'a jamais rencontrées pendant la formation (super-généralisation). Cela peut être interprété comme un exemple de "0 shot learning" (généralisation à de nouvelles situations jamais vues).

- **Passage à l'échelle:** Nous avons exploré diverses applications industrielles de notre algorithme dans le contexte des réseaux électriques. Nous avons établi la viabilité de l'approche à l'aide d'expériences numériques démontrant l'applicabilité des méthodes développées sur des réseaux électriques de taille réalistes.

En résumé, l'algorithme de "guided dropout" permet de prédire les conséquences sur les flux d'électricité à la suite d'une modification volontaire ou accidentelle du réseau. Nous abordons ce problème en codant séparément les deux types d'entrées que nous avons, les **injections** et la **topologie**, via deux canaux séparés : (1) Les données continues (injections = productions et consommations) sont introduites de manière standard, via une couche d'entrée réseau neuronal ; (2) Les données discrètes (topologies du réseau électrique) sont codées directement **dans** l'architecture réseau de neurones. L'architecture du réseau de neurones est ensuite modifiée dynamiquement en fonction de la topologie du réseau électrique. Chaque topologie correspond à un schéma de connexions dans le réseau de neurones. Ceci est obtenu en activant ou désactivant des unités cachées en fonction de l'état du réseau électrique. Par rapport à d'autres approches, l'avantage principal de cette technique réside dans sa capacité à prédire les flux, même pour des topologies de réseau jamais observées auparavant. Le réseau de neurones est capable d'extrapoler les résultats des actions unitaires observées pendant la formation en combinant les actions au moment du test. Nous avons exploré deux types de scénarios dans nos applications:

- **Mode prédictif:** L'algorithme "guided dropout", introduit dans [14], prédit les flux d'énergie dans les réseaux électriques haute tension après des imprévus (changements accidentels de topologie, par exemple suite à une rupture de ligne dans une tempête). Le flux de puissance pourrait être prédit jusqu'à 99 % de précision sur des cas d'étude standard (à l'aide du MAPE - Erreur absolue moyenne en pourcentage) avec un temps de calcul 300 fois plus faible que celui des simulateurs de grille physique basés sur les lois de Kirchoff. Le réseau neuronal pourrait également prédire avec précision les flux de courant en cas d'imprévus non observés durant l'entraînement, sans connaissance détaillée de la structure du réseau électrique. En ce sens, c'est un grand pas vers l'émulation du simulateur physique sans la description complète du réseau électrique. Ceci peut être considéré comme un avantage par rapport au simulateur puisque celui-ci nécessite cette information complète afin de calculer les flux.

- **Mode préventif:** Dans [13] nous avons montré que le "*guided dropout*" peut être utilisé pour classer les contingences qui peuvent survenir par ordre de gravité. Ceci peut être utilisé par les opérateurs pour prioriser les études de manière proactive et se concentrer sur les plus critiques. Dans cette application, nous avons démontré que notre algorithme permet d'obtenir le même risque que les politiques actuellement mises en place tout en n'exigeant qu'environ 2% du budget informatique actuel. Nous nous entraînons sur seulement 0.5% du total des configurations du réseau électrique. Ceci montre ainsi que l'algorithmedu guided dropout peut traiter des cas de grille jamais vus auparavant. Ce dernier cadre a été affiné dans [12] en complétant notre approche d'apprentissage machine par des simulations ciblées réalisées par un simulateur physique (comme celles actuellement utilisées dans les processus opérationnels). Dans ce dernier article, nous avons démontré que nous pouvons prédire avec précision le risque

[2] pris lors de l'exploitation d'un état de grille dans une topologie donnée avec une précision supérieure à 95% alors même que ce calcul global était auparavant irréalisable.

En conclusion, nous avons démontré que l'apprentissage automatique, et en particulier l'apprentissage en profondeur (deep learning), pouvait être utilisé pour prédire avec précision les flux de courant en fonction des productions et des consommations ainsi que de la topologie du réseau électrique. Cette approximation rapide peut être utilisée pour remplacer ou compléter les simulateurs physiques lorsque l'efficacité des calculs est préoccupante et que la perte de précision n'est pas critique étant donné les incertitudes dans les données. Le fait que l'architecture de réseau neuronal proposée ne nécessite pas une description complète des éléments de la grille (propriétés physiques ainsi que la connectivité observée) est une propriété souhaitable puisqu'il est difficile de la connaître exactement à l'avance avant les opérations temps réel.

[2]Voir *E. Karangelos et L. Wehenkel, " Probabilistic reliability management approach and criteria for power system real-time operation ", Power Systems Computation Conference (PSCC),2016, IEEE, 2016, pp. 1-9.* Equation 3 pour une définition formelle de ce risque.

Table of contents

List of figures

List of tables

Glossary

"Direct Current" (DC) approximation is a linearization of the power flow equations. Typically, this model is used as an approximation of an "alternating current" more realistic solution of the same power flow equations. 14

activation function In the neural network literature, activation functions are appliedelement-wise to the hidden units of a neural network. This allows a neural network to represent some non linear mappings between inputs and its outputs. Most frequent activations are the **sigmoid**, the **linear activation function**, or the **ReLU**.. 33

architecture (when referring to neural network): The architecture of a neural network represents how the computation are performed in this neural network. For example, it can be the number of layers,the number of units per layer, but can also include which layer is connected to which other oneetc. 59

bus In a "power grid", a bus is where all objects ("power lines", injections", etc.) are connected together. If two objects are connected together, they are on the same bus.. 4, 5, 45

dispatcher In a power system context, a dispatcher is a person in charge of maintaining, in real time, 24h a day, 7 days a week the power grid in a secure state. His job implies performing topological changes, anticipating risk that may cause problem in the future, or even make sure that a "planned **outage**" can be operated in real time (*i.e.* that it will not cause any issues). 22

dropout it is a technique commonly used in the neural network community since its introduction in Srivastava et al. [46]. The main objective is to improve the performance of neural networks on unseen example by **randomly** killing units of a neural network at training time.. 61

generalization In the machine learning literature, this terminology is used to denote the capacity for a learned model (for example a neural network) to make good prediction on unseen data, even if these data have neverbeen observed during training. The assumption behind the generalization is that these new data must come from the same distribution as the training set.. xxiv, 64, 84

guided dropout it's a neural network architecture developed during this PhD that allows to predict power flows rapidly given only partial topological information about the topology. It is presented in detail in chapter 5.. 2, 61, 62, 64, 65, 68, 71, 73, 74

Hades2 is the load-flow solver currently in use at RTE. This solver is being used as the ground truth when generating the power system dataset in this manuscript.. 12, 52, 78, 81

injection is an object that inject (or extract) power from a "power grid". Most often, if the value of power injected is positive it can be called a "production" or "generator". If this value is negative, it can be denoted as a "load" or "consumption". xv, 4, 30, 31, 34, 56, 59, 60, 62

linear activation function Is the most simple activation function, that doesn't change its input, often use at the last layer, when making predictions for a regression problem:

$$\text{Linear}(t) \overset{\text{def}}{=} t$$

. xxi

load (also called consumption) represents the consumer that withdraw power from the grid. In this manuscript this can either be aggregation of citizens (city) or big industrial consumers. Consumptions are a subset of the more generic terminology "injection". xv, 3, 4, 31, 62

MAPE (**M**ean **A**bsolute **P**ercentage **E**rror) is a way to measure the error made by predicting values $\hat{\boldsymbol{y}}$ when the true values are $\boldsymbol{y}$ in percentage. It's formal definition is :

$$\text{MAPE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N} \sum_{1 \leq k \leq N} \left| \frac{\hat{\boldsymbol{y}}_k - \boldsymbol{y}_k}{\boldsymbol{y}_k} \right|$$

. xx, 38, 97, 105, 106

neural network is a machine learning model described in Chapter 3. In this manuscript, to avoid confusion, the terminology "network" will always refer to "artificial neural network" and not to a "graph" nor a "(power) grid". xvi, 29, 32, 34, 35, 64, 90

outage In a power system context, a contingency is a disconnection of one or multiple lines of a power grid due external event. We speak of "unplanned outage" if this external event has not been anticipated, such as the fall of a tree or a windstorm. On the contrary, "planned outage" are operated most of the time for maintenance reason: repainting a tower for example. xxi, 20, 78

plain architecture In the "guided dropout" setting, the plain architecture is a neural network architecture with all units connected. It does not necessarily corresponds to a possible encoding of actions $\boldsymbol{\tau}$. xvi, 62, 64, 67

power flow is a mathematical problem representing the Kirchhoff's equations that a grid in a quasi static regime must verify. Most of the time, it takes the form of a non convexproblem, with non convex constraints. It is solved using dedicating solver, such as "Hades2", the one currently in use at RTE. 12, 52

power grid aims at transmitting the power from the productions to the consumers. In all this manuscript, a grid will be the power system infrastructures managed by TSO. In this manuscript, to avoid confusion, the terminology "grid" denotes exclusively a power grid, and wont' be used as meaning "graph" nor a "(neural) network". Gridsare composed of "power lines" interconnecting "buses" . 4, 30, 59–61

production (also called generator or power plant) is an entity that injection power in a transmission power grid. It isalso the main object that can maintain a certain voltage amplitude at a given bus. Productions are a subset of the more generic terminology "injection". xv, 3, 26, 31, 62

ReLU (**Re**ctifier **L**inear **U**nit) is an activation function widely used by the deep neural network community. It is defined as:

$$\text{Relu}(t) \overset{\text{def}}{=} \max(0, t)$$

. xxi, xxiv

Réseau de Transport dÉlectricité (RTE) is the French TSO. It is also the company in which the PhD took place. 1, 12

RMSE **R**oot **M**ean **S**quared **E**rror is a way to measure the error made by predicting values $\hat{\boldsymbol{y}}$ when the true values are $\boldsymbol{y}$. It represents the variance of the error. It's formal definition is :

$$\mathrm{RMSE}(\hat{\boldsymbol{y}},\boldsymbol{y}) = \frac{1}{N}\sqrt{\sum_{1\leq k\leq N}(\hat{\boldsymbol{y}}_k-\boldsymbol{y}_k)^2}$$

. 38, 99

sigmoid (or hyperbolic tangent) was the most common non linearity for hidden units of neural network, now replaced by the **ReLU**. It's formula is either:

$$\mathrm{sigmoid}(t) \overset{\mathrm{def}}{=} \frac{1}{1+e^{-t}} \tag{1}$$

$$\tanh(t) \overset{\mathrm{def}}{=} \frac{2}{1+e^{-2.t}} - 1 \tag{2}$$

$$= 2.\mathrm{sigmoid}(2t) - 1 \tag{3}$$

. xxi

substation is a physical place where object are interconnected. Typically, one of the end of a line arrives at a substation, and then is routed to a specific "bus" in this substation. 4, 103

super generalization This is a terminology we adapted from the **generalization**. For the problem of predicting flows based on injections and partial topological information, the "super generalization" property is the ability to predict accurately the flows even when the topology has not been observed during training. This deviatesfrom the standard "**generalization** mainly because the test distribution (never observed topology) is different from the training distribution.. 61, 64, 84

superposition theorem It is a theorem used in the power system community to compute the flows or the voltage in some circuit. It states that "for a linear system the response (voltage or current) in any branch of a bilateral linear circuit having more than one independent source equals the algebraic sum of the responses caused by each independent source acting alone, where all the other independent sources are replaced by their internal impedances" (source: wikipedia).. 70

test set In machine learning, a test set is a dataset use to report final error of machine learning algorithms. Data of this test set are never seen at training time: this dataset is not used to find the optimal parameters, nor involved in the selection of meta parameters.. 35

topology In a power system, a topology represents how the objects (productions, loads and power lines) are interconnected with one another. This terminology is used both do denote this interconnection at a grid level, in this case we refer it as "grid topology" and at a substation level, in this case called "substation topology". The "grid topology" is exactly the list of the "substation topologies" for each substation of the power grid.. 5, 49

training set In machine learning, a training set is a dataset used to find the optimal parameters of a model. Generally, only data of this training set are used to compute gradient and adjust the model parameter. 60, 84

transmission line is a generic term used to denote everything that allows power to flow in a "power grid". More specifically, in this manuscript a line will allowthe current to flows between 2 "buses" . xv, 4, 30, 31, 59, 88

Transmission System Operator (TSO) The entity responsible for managing the "transmission power grid". 1, 4, 60

Chapter 1

Background and motivations

This work has been motivated by an industrial problem: how to manage as efficiently as possible a power grid in a changing environment given the occurring Energy Transition. This work has been carried out at (Réseau de Transport d'Électricité) **Réseau de Transport dÉlectricité (RTE)** the French Transmission System Operator (**Transmission System Operator (TSO)**) in charge of managing the security of the power grid. In Europe, RTE is one of the largest TSO. It is responsible for managing a powergrid counting approximately 10 000 powerline representing more than 100 000 km covering the entire French territory. While electricity is today a common good that people take for granted, the complexity of managing the powergrid is rising and becoming even more challenging for TSO's: the average annual total power demand has stopped increasing, new behaviors are emerging, and the public acceptance of new infrastructures has become very low.

TSO's, and RTE, in particular, have decided to investigate new methods in order to help them tackle these new challenges. Among the methods explored, RTE chose to focus especially on "machine learning" methods to handle recurrent real-time situations by leveraging the use of historical data, leaving time to operators to work on atypical cases. This work took place at a time when a lot of enthusiasm followed the renaissance of artificial neural networks (re-baptized "deep learning" methods (*e.g.* [23], [28],[3]), leading to a profusion of new inspiring publications. Deep learning techniques have become pervasive in computer vision and natural language processing applications [54]. That is the main reason why "deep learning" techniques have been the main focus of this work.

To help tackle this problem, RTE has a lot of data at its disposal. Every 5 mins, from 2011 to the present time, a full snapshot of the grid is stored. The first part of this work has been dedicated to understanding this dataset: which information can be extracted from these data? How to use them? During this first phase, we also acquire insights on how a powergrid is operated, and what are the issues "operators" (people in charge of the security of the power grid, also called "dispatchers") are coping with today. After understanding what properties are required for a learning machine that could possibly be used by RTE, we started to devise a method to generated realistic semi-synthetic datasets. Using synthetic dataset has multiple advantages. First, the quantity of data generated can be adapted to the difficulty of the task, as well as the size of the powergrid studied. Most importantly, we have some control over the dataset. Operators perform actions on the power grid to solve different problems not necessarily related to the issues treated in this work. More details about the real data are provided in chapter 4. Finally, once the developed method obtains good results on a synthetic dataset at a relevant scale, the developed algorithm has been tested on the French dataset.

This work is organized as followed:

- In Chapters 2 and 3 we present the basics of the powersystem and of "machine learning". In these two first chapters, we also introduce some notations and try to expose the model of a "perfect operator" and expose how we could use this modeling to suggest relevant actions to real-time operators.
- Chapter 4 introduces the dataset we are using in this work. This includes descriptions of the French dataset available in this work and the method to generate synthetic, but realistic data.
- In Chapter 5 we introduce the main method developed in this work, that we called **guided dropout**. This method is also detailed in this chapter, and some interpretations of how it works are also developed.
- In Chapter 6 we show the main results of this work.
- Finally, Chapter 7 concludes with a discussion on the results obtained and on further work that is being pursued at the moment.

Chapter 2

Power system description

2.1 Power grid description

In this chapter, we introduce the objects and concepts that we use for the modeling and the simulation of power grids, focusing on the ones required for the operation of such a system. First, we introduce what a power grid is. Then, we explain what are the equations we are using for explaining the behavior of a power grid (we are aware these equations are approximations, we will elaborate on these approximations in the dedicated sections). Finally, we present a solver for these equations, as well as a method to approximate these flows: the DC approximation, an approximation widely used in the power system community, that will serve as a baseline for our model.

2.1.1 Utility of a power grid

The electrical system on the French territory is divided into different parts, operated by different actors: production (supply), transmission, distribution (consumption, load).

The first sector is **production**: The main sources of power generation are thermal power plants (nuclear, fuel, gas, or coal power plants), hydropower plants (dam, run-of-the-river) or coming from other renewable sources (mostly wind or solar panels). Each of these power sources has their specificities but in our case, they will be collectively referred to as **production**.

The second sector is **consumption**. In this manuscript we adopt the TSO standpoint: consumer is not a single house, but rather an aggregate of customers, *e.g.* a small town, or a big industrial firm. Such consumers are connected to the high voltage power-grid through a low voltage grid (last mile grid) called “Distribution Grids” and operated by “Distribution System Operators” (DSO). This entire last mile grid is called a **load**. In our modeling, this distribution grid is not included into the “power grid”.

The last sector of the electricity landscape is the **power grid**. It is operated by **Transmission System Operator (TSO)**. This will be the main focus of this thesis. One role commonly assigned to TSOs is to make sure every consumer can withdraw the amount of power they need when they want, wherever they are. TSOs are in charge to ensure the security of supply for the consumers, maintaining the system reliability at a defined level. This aspect of their work is what we are interested in. It is further described in Section 2.2 of this manuscript.

In most European countries, producers/suppliers, distribution system operators (DSO), and transmission system operators (TSO) are three distinct entities since the European Energy Market was designed and launched in 2000. Additionally, in most cases, the power grid (transmission part) is a State monopoly. This impacts our work in the following way: the producers and consumers have specific behaviors and TSOs have to provide them a fair, transparent and efficient access to the grid. This drives our modeling choice, separating the infrastructure of electricity transportation (the **power grid**) from producers (modeled as productions) and consumers (including the distribution grids, modeled as load). TSOs have the power grid under their control, but but they must have when operating this transmission powergrid, a minimal impact on the behavior of the consumers and/or producers.

In order to introduce and illustrate vocabulary and notations, we show in figure 2.1a a schematic representation of a power grid. This power grid is composed of several types of entities: eight **transmission lines** ($n = 8$), denoted by $l_1, \dots l_8$, three **consumptions** or **loads** c_1, c_2, c_3, and two **productions** or power plants p_1 and p_2. The main focus of this work will be the configuration of the grid, defined by the connectivity of the power lines, and not so much the power injected in terminal nodes (productions and consumptions).

A **"substation"** is a physical location in which the different components are connected, switching devices allow to change the connectivity between components and to create electrical **buses**[1] At substations, dedicated transformers interconnect distribution grids and the transmission grid (for simplicity these transformers are modeled as **load**), and switching devices interconnect **transmission line** to ensure the long distance transmission of power. It is also at substations that power plants are connected to the rest of the grid. For brevity, in our manuscript, **productions** and **consumptions** will be both referred to as **"injection"** and power lines, also

[1]When modeling a powergrid as a graph, a bus denotes the nodes of a graph, and all objects interconnecting two buses (power lines, transformers, etc.) are the edges of this graph. The terminology “bus” is then equivalent to a “node” in this context. However, in the power system community, “node” sometimes refers to the concept of “busbar” illustrated in Figure 2.1b.

called **"transmission lines"** will refer to entities connecting (at least) two substations to allow power to flow between substations of the power grid. Other devices are connected to the powergrid, such as transformers between different voltage levels, phase shifter transformers (that play important roles in managing the flows), reactor and capacitors banks (mostly used to control the voltages) and HVDC (High Voltage Direct Current) coupled with AC/DC converters. These devices (and others), as well as their associated properties and controls, are not directly addressed in this work. They are however modeled by the power flow solver we are using, even though we don't explicitly describe their behaviors in this work.

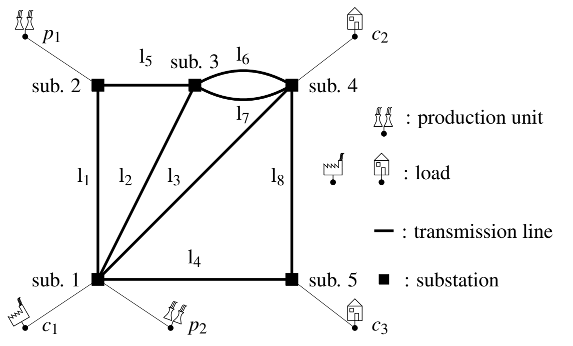
For most substations, TSOs can decide how to route power. To that end, substations may include one or several “busbar” to which objects (injections or transmission lines) are connected. Thanks to these “busbar” it is possible to have multiple **buses** in the same substation. A bus ensures the electrical interconnection via circuit breakers (See Figure 2.1b). This implies that **two objects can enter the same substation, and not be interconnected**. Buses allow the TSO to disconnect a line, *e.g.* for maintenance reason, or to control the current flow by diverting it. An example of a substation is provided in Figure 2.1b. For example, in Figure 2.2, we show how a TSO could divide the first substation into two separate buses. Note that in the displayed schema, there is no *direct* connection between bus 1 and bus 2: the only way to go from the first bus to the second one is through another substation. The way the injections and lines are interconnected with each other is denoted by **topology**. There are 2 concepts behind this term. When referring to a whole power grid, we will speak of "grid topology". Interconnection in a specific substation is denoted by "substation topology".

2.1.2 Injections and flows

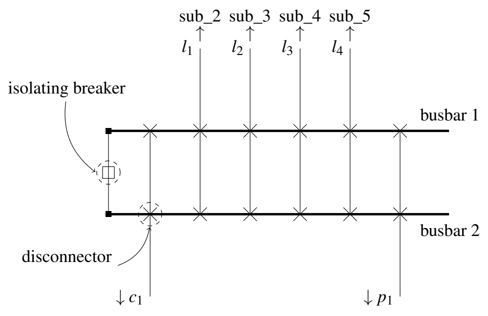
As explained in the previous section, from the TSO point of view, injections are imposed by external actors. In this subsection, we expose how given injections result in "steady state" power flows.

Since today most parts of the power grid are operated in alternating current (AC) – as opposed to direct current (DC)–, we focus exclusively on describing AC power flows.

As is known [30], AC power transmission systems are described by a number of variables introduced in this section: voltage and current intensity. Both voltage and current are described by magnitude and phase (also referred to as "angle"). We hereby explain how to compute these magnitudes and angles at each bus of the grid from available data, and how to use them for determining flows on power lines. The process of computing flows given injections is called a "power flow". The equations we



(a) Schematic representation of a power grid with 3 loads (two residential, one industrial) denoted by c_1, c_2 and c_3, 2 productions units p_1 and p_2, 8 power lines, and 5 substations denoted by "sub. 1" to "sub. 5"



(b) Zoom in the structure of the first substation (sub. 1). Disconnectors allow connecting objects (lines or injections) to buses; a coupling breaker allows to connect the two busbars together, or contrarily to isolate them from each other thus forming two independent buses.

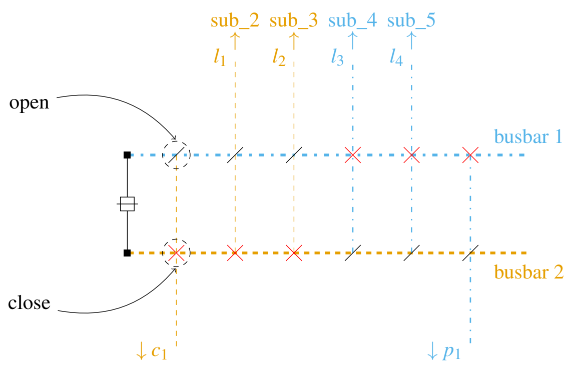
Fig. 2.1 Schematic representation of a small power grid (2.1a) and zoom in the structure of the subtation 1 (bottom left substation - 2.1b).

describe are valid for a **fixed voltage frequency**, and in a **quasi-stationary regime** where productions and loads are balanced (*e.g.* the sum of all the power produced by all productions is equal to the sum of all power consumed and the sum of the losses). Typically, if a power line is disconnected, the resulting flows computed with this method reach a steady state within a few minutes (1 or 2 typically) after the disconnection. We are not interested in this work in transient phenomena occurring over shorter periods of time, however, this phenomenon can be a dramatic importance for powergrid operations in some cases. Finally, as we already motivated, the equations will be shown only for the three main types of objects we are interested in: productions, loads and powerline.

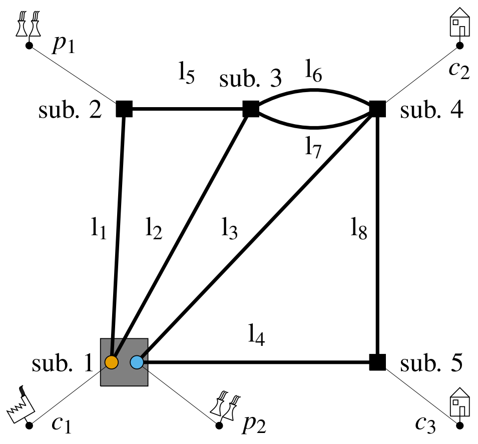
In this subsection, we denote by G a grid with m buses and n power lines. In the examples of Figure 2.1, we have $m = 5$ and $n = 8$, but in the example of Figure 2.2 $n = 8$ still, but $m = 6$. In practice, the number of electrical buses on a power grid is not constant: it changes depending on how the TSO operates the grid and the chosen topology at each substation.

For each bus $k \in \{1, \dots, m\}$, let θ_k be the voltage angle and $|v_k|$ the voltage magnitude of this bus. If a power line l[2] connects bus j and bus k, we will denote by $z_{j,k}$ its "impedance". The impedance represents the physical properties of this powerline, in most modeling of a transmission power grid, it is a complex number

[2] Or any other objects: HVDC, transformers etc.



(a) Representation of a possible way to split the substation 1 in two distinct electric bus.



(b) Representation of the power grid shown in 2.1a after splitting the substation 1 in two buses.

Fig. 2.2 Schematic representation of the power grid shown in 2.1a, for which substation 1 has been split into two independent electrical buses. Bus 1 is colored in blue / dashed-dotted and regroup load c_1 and powerlines l_1 and l_2. Bus 2 is represented in orange / dashed and is made of production p_1 and powerlines l_3 and l_4.

(*i.e.* includes both a resistance and a reactance - more information can be found in [30]). It happens that the equations are better yet written when taking into account the admittance. The admittance is defined as $y_{j,k} \overset{\text{def}}{=} \frac{1}{z_{j,k}}$. If no power line connects bus j to bus k, it is equivalent to having a power line with an admittance of 0.

As shown in [31], if we denote by $i_{k \to j}$ the current (complex number) flowing between bus k and bus j, we have:

$$i_{k \to j} = y_{k,j}(v_j - v_k) \tag{2.1}$$

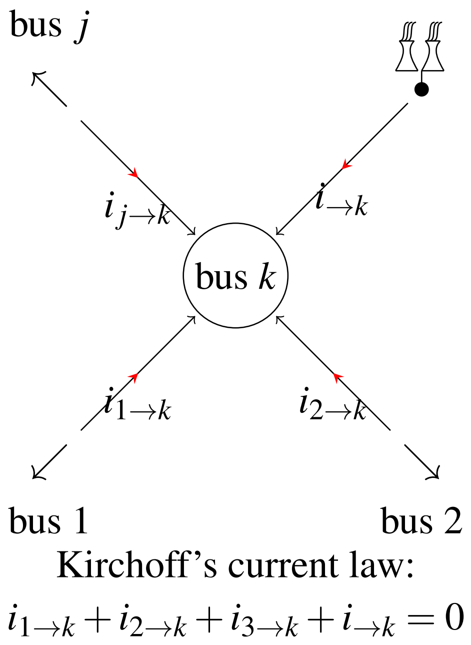
This equation is illustrated in figure 2.3b. Note that in case there are other elements[3] connected to one end of a power line, the admittance $y_{k,j}$ considered in Equation 2.1 takes into account these other object. So the current flow $i_{k \to j}$ will not be, in that case, the current flow on the power line. We prefer not entering in this detailed modeling. For more information on this issue, see [30].

Kirchhoff's law (illustrated in Figure 2.3a) state that everything entering in a bus must go out. This a special case of the conservation of energy law . If we denote by $i_{\to k}$ the current injected at bus k (typically by a productions or by a loads) this entails

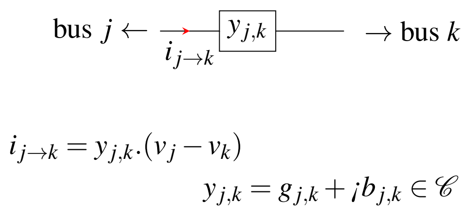
[3]We made the choice, for clarity, not to model all the objects present in a power grid. Such object connected to one end of a powergrid can include shunts of selfs. More about their modeling can be found in [30] for example.

that:

$$i_{\to k} + \sum_{j=1, j\neq k}^{m} i_{k\to j} = 0 \tag{2.2}$$



(a) Illustration of Kirchoff's bus law, which is a particular case of the conservation of energy. Nothing is created (nor destroyed) in a bus.


bus $j \leftarrow$
$y_{j,k}$
$\rightarrow$ bus k
$i_{j\to k}$
$i_{j\to k} = y_{j,k}.(v_j - v_k)$
$y_{j,k} = g_{j,k} + \jmath b_{j,k} \in \mathscr{C}$

(b) Definition of the admittance y (and the conductance g and susceptance b). We denoted by $\jmath$ a complex number such that $\jmath^2 = -1$, often denoted by j in the power system literature.

Fig. 2.3 **Representation of Kirchhoff's equations**. Left (Figure 2.3a) is the Kirchhoff's bus law. Right (Figure 2.3b) illustrates the link between current and i, admittance y and voltages v.

By substituting 2.1 in 2.2 we get:

$$i_{\to k} + \sum_{j=1, j\neq k}^{m} y_{k,j}(v_j - v_k) = 0 \tag{2.3}$$

The equation can be written for the full powergrid using matrix notations. Let Y (commonly called the admittance matrix) be the $m \times m$ matrix :

$$Y \overset{\text{def}}{=} \begin{bmatrix} -\sum_{j\neq 1} y_{1,j} & y_{1,2} & \dots & y_{1,m} \\ y_{1,2} & -\sum_{j\neq 2} y_{2,j} & y_{2,3} & \dots \\ \vdots & & & \\ y_{1,m} & y_{2,m} & \dots & -\sum_{j\neq m} y_{2,j} \end{bmatrix} \tag{2.4}$$

Then is that case, the equations showed in 2.3 can be written as:

$$\begin{bmatrix} i_{\to 1} \\ \vdots \\ i_{\to m} \end{bmatrix} = Y \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \tag{2.5}$$

Let's now define p_k (called active power) and q_k (called reactive power) such that, by denoting by $s_k = i_{\to\ k}\bar{v_k}$, we have, for the power injected at bus k:

$$s_k = \underbrace{p_k}_{\text{active power injected}} + \jmath \underbrace{q_k}_{\text{reactive power injected}} \tag{2.6}$$

And in the same manner, for the active (resp. reactive) power $p_{k\to j}$ (resp. $q_{k\to j}$) flowing from bus k to bus j:

$$s_{k\to j} = i_{k\to j} * \bar{v_k} \tag{2.7}$$

$$s_{k\to j} = \underbrace{p_{k\to j}}_{\text{active power flow}} + \jmath \underbrace{q_{k\to j}}_{\text{reactive power flow}} \tag{2.8}$$

In these notations, $\jmath$ is the complex number such that $\jmath^2 = -1$. It is often called i by the mathematical community (this can be confused with the electrical current in our case) or by j by the power system community. We prefer to reserve the letter j for free variables. We will also define : $g_{j,k}$ and $b_{j,k}$ such that $y_{j,k} = g_{j,k} + \jmath b_{i,k}$. Then, from equation 2.5, if we split real part and imaginary part, with our notations, we get, for each bus k of our power grid:

$$\begin{cases} 0 = -p_k + \sum_{j=1}^{m} |v_k||v_j| \left(g_{k,j}.\cos(\theta_k - \theta_j) + b_{k,j}\sin(\theta_k - \theta_j)\right) & \text{for the active power} \\ 0 = q_k + \sum_{j=1}^{m} |v_k||v_j| \left(g_{k,j}.\sin(\theta_k - \theta_j) - b_{k,j}\cos(\theta_i - \theta_k)\right) & \text{for the reactive power} \end{cases} \tag{2.9}$$

The equations 2.9 describe the relations between the different quantities, especially between the voltage angle and magnitude (θ and $|v|$) and the active and reactive power flow (p and q). A power flow solver, such as Hades2 computes some of these quantities from the others. Let's clarify what are the unknowns and what are inputs (that will be provided to the simulators) in the equations 2.9.

The $g_{k,j}$ and $b_{k,j}$ are physical properties of the power lines, they need to be provided to the power flow solver.

We also know how much active power (p_k) is being supplied or withdrawn at bus k: this is an input of our problem. For each bus (except for a particular bus, chosen as "slack bus"[4] where we impose $\theta = 0$), θ_k are unknown and must be computed by the power flow.

Depending on whether or not a generation unit able to do voltage control is connected to the node, either q_k is the input and $|v_k|$ is computed, or the other way round In this explanation, we will adopt a simplified but unrealistic setting, where there are no limits on the amount of reactive power a production can absorb or produce[5]. This implies that, if a production unit is connected to a specific bus, it will maintain a given voltage magnitude setpoint, in that case, the unknown is q_k. If no production units are connected[6] to a bus k, then the unknown is $|v_k|$. A summary of the variables provided as input and the unknowns can be found in table 2.1.

Table 2.1 **Summary of the variables computed by a power flow solver**. There is at least one slack bus per power grid (where $\theta = 0$ is ensured), every other bus is either "PV" (meaning values of P - active value - and V - voltage magnitude - are given, which is often the case for buses where productions are connected), or "PQ" (meaning values of P and Q - reactive value - are provided).

Bus type	Description	Is the variable provided in inputs ?			
		θ_k	p_k	$\|v_k\|$	q_k
"**Slack**" bus	at least one per grid	**yes**	no	yes	no
"**P**,**V**" bus	a production is connected	no	**yes**	**yes**	no
"**P**,**Q**" bus	no production$^+$ is connected	no	**yes**	no	**yes**

$^+$ Remember that sometimes, due to physical properties, some productions behave like a load. Rigorously, a bus can be "P,Q" even if a production unit is connected to it.

As we stated in the previous section, we are mostly interested in this work in two types of variables. We explained in the previous paragraph (in a simplified setting) how the injections (productions and consumptions) impact the flows. Now let's detail the impact of the topology. It has an impact on the admittance matrix Y, as it can change its size, and change the value of its coefficients. The topology has no impact on the physical relationships exposed in 2.9: these remain correct[7] even if the topology is

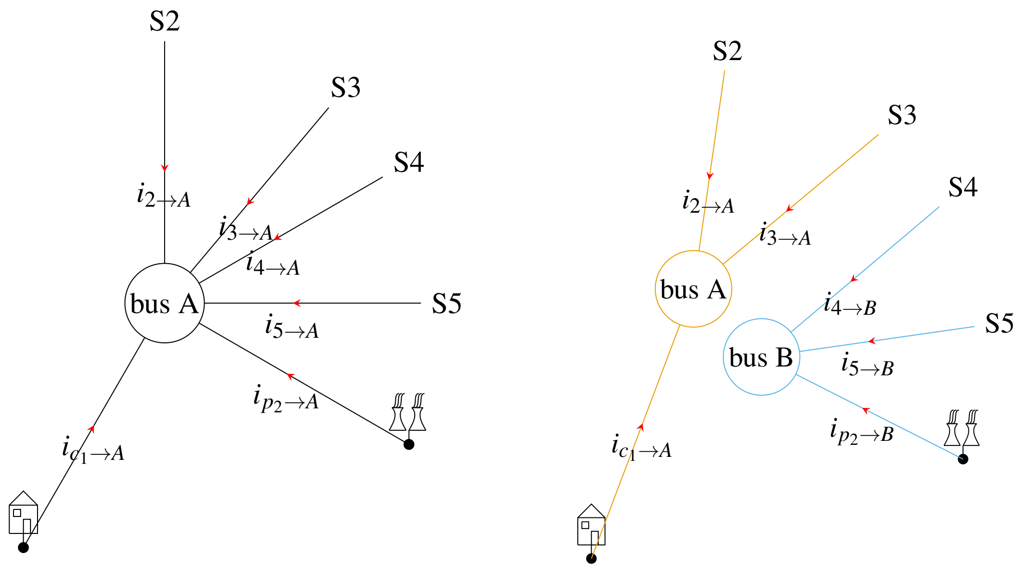
[4]For some more advanced solvers, there are multiple slack buses. But as this work does not focus on power flow solvers, we detail here only the case where one single slack bus is present.

[5]Sometimes, due to physical properties of the production unit, a production will behave like a load: instead of providing a fixed voltage magnitude, it will produce (or absorb) a given amount of reactive power q. A more detailed description of this phenomenon is presented in [30].

[6]Or if a generation unit is connected, by certain physical limits prevents it to fix such voltage level.

[7]Under our assumptions.

evolving. An example of such changes is illustrated in Figure 2.4, where the topology of substation 1 of the powergrid showed in figures 2.1a and 2.2 is changed.



$$\begin{cases} i_{2\to A} + i_{3\to A} + i_{4\to A} + i_{5\to A} + \\ i_{p_2\to A} + i_{c_1\to A} = 0 \end{cases}$$

$$\{i_{p_2\to A} + i_{2\to A} + i_{3\to A} = 0 \quad \text{bus A}$$
$$\{i_{c_1\to B} + i_{4\to B} + i_{5\to B} = 0 \quad \text{bus B}$$

(a) Illustration of the Kirchhoff's current flow equation in the substation 1 of the figure 2.1, where everything is connected to a single bus.

(b) Illustration of the Kirchhoff's current flow equations in the substation 1 of the figure 2.2, where two independent buses exist.

Fig. 2.4 Impact of the topology on the powerflow equations. Here only the current flow equation is written for 2 topologies of the same substation (sub. 1) of the grid showed in Figure 2.1 (left) and Figure 2.2 (right).

The third and last variable of interest for our problem is the flows. Here we detailed how to obtain the power flow from the θ_k, $|v_k|$ and the physical parameters $g_{k,j}$ and $b_{k,j}$. We can deduce the flows on each power lines using the equations 2.1, and 2.8 to compute the active, reactive and current flow flowing from bus k to bus j. The relations are:

$$p_{k\to j} = |v_k|\,|v_j|\,g_{k,j}.\sin(\theta_k - \theta_j) \tag{2.10}$$

$$q_{k\to j} = -|v_k|\,|v_j|\,b_{k,j}.\cos(\theta_k - \theta_j) \tag{2.11}$$

$$i_{k\to j} = \frac{\sqrt{p^2_{k\to j} + q^2_{k\to j}}}{\sqrt{3}.|v_k|} \tag{2.12}$$

A closer study of what these equations entail can be found in [31], and is developed in section 2.2.1 of this manuscript.

The process to compute θ_k and $|v_k|$, as well as the power line flows is called a power flow computation[8] It requires a simulation engine which solves a sequence of non linear equalities The simulation engine used for generating the synthetic dataset in this work is called "**Hades2**". It is a proprietary software owned by **Réseau de Transport dÉlectricité (RTE)**. Its functioning is described in the subsection 2.1.3. A freeware version can be found at http://www.rte.itesla-pst.org/.

2.1.3 Powerflow solver

The problem 2.9, with variables $|v|$, q and θ is a non linear non convex problem. Multiple methods are available in the literature to solve it. A first historical solution to solve such a problem is the "Fast Decoupled Load Flow" method [47]. The main idea behind it is to take advantage of the weak coupling between active (and angle) on one side and reactive (and voltage magnitude) on the other side to reduce the need of expensive matrix factorization, and sparing computation time. One of the main issues of such methods is that they are not able to provide results sufficiently accurate, especially in the most stressed conditions. This is not the method adopted in **Hades2**. Among many others, we can mention a relatively new method called the "Holomorphic Embedding Load-Flow Method" [50] It has been recently developed and transforms the load-flow problem into a holomorphic function (*i.e.* a function that output values in the complex number), and to use some "complex analysis" methods[9] to approximate the load flow solution as a series of polynomials. The **power flow** "Hades2" solves a sequence of non-linear problems using for each of them an iterative process, derived from the Newton-Raphson method.

The first step consist in assigning initial values to all the variables. This is usually done using some heuristic. Then an iterative process, described in Algorithm 1 is used, sometimes multiple times.

There are two main stopping criteria for this algorithm:

- Either $\sum_k \Delta p_k \approx 0$ and $\sum_k \Delta q_i \approx 0$, in this case the load-flow "converged" : it has successfully found a suitable solution
- Or the number of iteration reaches a pre-determined maximum number, in which case, the solver is said to have "diverged". No suitable solution has been found.

The divergence can occur for different reasons, the two most important ones being:

[8]Sometimes “power flow computation” is abbreviate more simply by “power flow”

[9]Here "complex" stands for "complex number" and is not related to "complexity" or "difficulty".

```
Input: Admittance matrix $Y$, Slack node ID, $p_{\rightarrow k} \forall k$, $q_{\rightarrow k} \forall k \in$ {P,Q buses},
    $|v_k| \forall k \in$ {P,V buses}
Output: $|v_j|$ and $\theta_j$ for all buses of the power grid
    Initialization :
 1: $|v_j| \leftarrow 104\% v_{nom}$ and $\theta_j \leftarrow 0$
    Main loop :
 2: while Stopping criteria not met do
 3:     $\Delta p_j = -p_j + \sum_{k=1}^{m} |v_j||v_k| \left( g_{j,k}.\cos(\theta_j - \theta_k) + b_{j,k} \sin(\theta_j - \theta_k) \right)$
 4:     $\Delta q_j = -q_j + \sum_{k=1}^{m} |v_j||v_k| \left( g_{j,k}.\sin(\theta_j - \theta_k) - b_{j,k} \cos(\theta_j - \theta_k) \right)$
        Compute the "Jacobian" matrix :
 5:     $$J = \begin{bmatrix} \frac{\partial \Delta p}{\partial \Delta \theta} & \frac{\partial \Delta p}{\partial \Delta |v|} \\ \frac{\partial \Delta q}{\partial \Delta \theta} & \frac{\partial \Delta q}{\partial \Delta |v|} \end{bmatrix}$$
 6:     invert it, ie compute $J^{-1}$
        Compute missmatch for $|v|$ and $\theta$:
 7:     $$\begin{bmatrix} \Delta \theta \\ \Delta |v| \end{bmatrix} = -J^{-1}. \begin{bmatrix} \Delta p \\ \Delta q \end{bmatrix}$$
        Update the new values
 8:     $|v_j| = |v_j| + \Delta |v_j|$
 9:     $\theta_j = \theta_j + \Delta \theta_j$
10: end while
11: return $|v_j|$ and $\theta_j$
```

Algorithm 1: Generic (simplified) power-flow solver based on the Newton-Raphson method. This algorithm shows the algorithm used in power flow solver. Such solvers, such as Hades2 uses this routine more than once in the process of computing flows, and this routine is optimized in multiple fashion that we don't detailed here.

- the instance of the problem shown in equation 2.9 has no solution for the specific input data. In the case of synthetic data, this can happen if a sufficiently high-quality standard is not met. Alternatively, in the case of actual data (coming from a powergrid snapshot) this probably mean the system is not stable. This lack of stability may be, for example, an indicator of a potential black-out.

- The starting point used is too far from the solution: a Newton-Raphson scheme is used to solve a non-convex problem. There is no theoretical guarantee it will converge to a desirable solution.

2.1.4 The DC approximation

In the power system community, a common approximation is used. This approximation is called the **"Direct Current" (DC) approximation**, and as such suffers some drawbacks. It is sometimes poor, especially when the power grid is prone to voltage problems and should avoid being used in these cases. This approximation is scarcely used for real-time operation at RTE, though it can be used in other areas such as grid development planning for example.

Despite these drawbacks, it has two main advantages: it cannot diverge (always gives a solution, provided that all information to compute it are available), and it is fast to compute. This is used as a baseline method in the experiments shown in this work when it makes sense to use it (*e.g.* when all the information are available regarding the line characteristics, the production and loads as well as a full description of the topology of the powergrid).

In comparison with the equations shown in 2.9, the DC modeling make three important simplifications, each one being the topic of the following paragraphs of this subsection:

1. the resistance r of a line is much less than its reactance x
2. for two connected buses (let's say j and k) the difference $\theta_j - \theta_k$ is small
3. the voltage amplitude $\left|v_j\right|$ at each bus j is very close to its setpoint value.

The impact of each of these simplifications on the simplified version of the power-flow equations explained in 2.9 will be discussed in a separated paragraph.

***$r << x$*:** This assumption entails that there are no losses on a system solved using the DC approximations. This makes particular sense for high voltage power grids which have a high conductance x through power lines compared to their resistance r. This assumption has an effect on the admittance $y_{j\to k}$ of the powerline between bus j and bus k. By omitting the subscript $j \to k$ this gives us:

$$\begin{aligned} y_{j\to k} &= \frac{1}{z_{j\to k}} = \frac{1}{r + \jmath x} \\ &= \frac{r^2}{r^2+x^2} - \jmath \frac{x}{r^2+x^2} \end{aligned}$$

And by definition of g and b, we have :

$$y = g + \jmath b$$

thus :

$$g = \frac{r^2}{r^2+x^2} \underset{r\to 0}{\longrightarrow} 0 \tag{2.13}$$

and

$$b = \frac{-x}{r^2+x^2} \underset{r\to 0}{\longrightarrow} \frac{-1}{x} \tag{2.14}$$

So the power flow equations become:

$$\begin{aligned} 0 &= -p_j + \sum_{k=1}^{m} |v_j||v_k| \left(b_{j,k} \sin(\theta_j - \theta_k)\right) \\ 0 &= q_j + \sum_{k=1}^{m} |v_j||v_j| \left(b_{j,k} \cos(\theta_j - \theta_k)\right) \end{aligned}$$

$\theta_i - \theta_k \approx 0$: This will allow us to linearize the trigonometric functions sin and cos that will be approximated by the identity and the constant 1, giving:

$$\begin{aligned} 0 &= -p_j + \sum_{k=1}^{m} |v_j||v_k| b_{j,k}(\theta_j - \theta_k) \\ 0 &= q_j + \sum_{k=1}^{m} |v_j||v_k| b_{j,k} \end{aligned}$$

$\left|v_j\right| \approx |v|_{\text{nom}}$: This last hypothesis is made to make the problem completely linear, as $|v|$ will be constant. In practice, this is justified by the fact that voltages are always operated around a reference value. This assumption results in:

$$p_j = \sum_{k=1, k\neq j}^{m} b_{j,k}(\theta_j - \theta_k) \tag{2.15}$$

$$0 = q_j + \sum_{k=1}^{m} b_{j,k} \tag{2.16}$$

Recall that from the way the admittance matrix Y is defined (see equation 2.4) $0 = q_j + \sum_{k=1}^{m} b_{j,k}$. The second equation can then be ignored. Thus, the only remaining unknowns are the θs. And, by combining the definition of the admittance matrix Y

and the assumption made in 2.13 and 2.14, the equation 2.15 can be written:

$$\begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} = Y. \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_m \end{pmatrix} \tag{2.17}$$

The DC equations can then be solved by any method allowing to solve a linear system, for example by inverting the matrix Y.

2.2 Security criteria

In the previous section, we saw how the productions and loads induce some flows on lines. In this section, we will explain why TSO have to manage the current on the power lines to maintain the system in security at all times. We know from the previous sections that injections condition flows on power lines. In the first subsection, we explain why large flows are dangerous for both the powergrid infrastructures and its surrounding. Then we explain what is the actual security protocol implemented by most TSO today. Lastly, we emphasize some of its limits and expose new security criteria.

2.2.1 Thermal limit and "N" security

The injections induce flows, as we explained in the previous section. These flows induce themselves some current, that will heat the power line. This heating phenomenon is called "Joule's Effect" and is the main cause for losses on the transmission network. In fact, the losses on a powerline $l_{k\to j}$ that connects bus k to bus j, and with resistance $r_{k\to j}$ are equal to $r_{k\to j}.\left|i_{k\to j}\right|^2$.

Joule's effect has another impact on the power grid. It heats the conductor the powerline is made of. Because of this metal thermal expansion, the lines get closer to the ground, a phenomenon called "sag". The more current, the more heat hence the more sag. If it gets too close to the ground, a house, or even a pedestrian, the air insulating layer becomes thinner and beyond a certain distance there exists a risk for a short circuit, damaging the infrastructure but most importantly this can be lethal for surrounding population.

To avoid these possibly dramatic consequences, each power line $l_{k\to j}$ has a "thermal limit", that we denote by $\bar{l}_{k\to j}$. It can be further understood as the line capacity to transmit power. If the current flow $\left|i_{k\to j}\right|$ on this line is above this threshold for some time (for example 5 minutes), a protection will automatically act on the powerline and disconnect it from the powergrid (the effect of these protections can be understood as the opening of the disconnectors at both ends of the power grid).

These protections have drawbacks. When a line is disconnected, it often increases the flow in some other power lines as the flow on this opened line gets redispatched on the grid. These other powerlines could themselves overheat, and be automatically disconnected one after the other. This phenomenon is called a "cascading failure", and can lead to a black-out: *i.e.* a deficient grid state where no line is in service in a huge area, preventing customers to use electricity, and producers to sell their production. Additionally, putting the lines back in service and restoring the grid is a delicate operation that can take time, as it can require the visual inspection of a maintenance team. Hence outages of power may last several hours.

In order to avoid such dramatic events, the TSO ensures that the flow in each line is below the thermal limit at all times. This is often called "N" security. This "N" meaning "when the powergrid is complete[10], when there are still n power lines available". This criterion is not the one currently used by most TSO. In the next subsection, we will detail the "N-k" security criteria

2.2.2 "N-k" security

The "N" security is the first step, but it is not enough. Transmission grids often cover vast areas and are composed of thousands of lines. The French power grid, operated by RTE (short for "Réseau de Transport d'Électricité", the French TSO) counts approximately 10.000 powerlines, covering all the French territory with a total of more than 100.000 km of powerlines.

Considering such large grid, the probability that a "bad" event (such as a birds nest, a thunderstorm, too much wind or snow) causes an equipment failure (also called short circuit) cannot be neglected. For example, for the entire French power system, there have been 261 600 short circuits from 1994 to 2017[11]. This corresponds on average to 10 900 short-circuit a year, most of them being of very short duration (a few milliseconds). The Figure 2.5 shows the histogram of the duration (in hour) of unplanned line disconnection after short circuits. As we can see, most of these lines are repaired on the day they occur. We also observed a multi-modal distribution, with 2 other modes centered around 24h and 48h: this is due to the difficulty of repairing the power lines during night hours (an inspection in the field must be performed before any other actions and requires to send a crew or a helicopter to do it and if the weather

[10]Complete here refer to the state where each powerline is connected except the one disconnected for maintenance operation (planned outage).

[11]Removing the most dramatic events due to the "Lothar" and "Martin" windstorms from December 26th, 27th, and 28th, and the Klaus windstorms of January 23rd, 24th and 25th 2009.

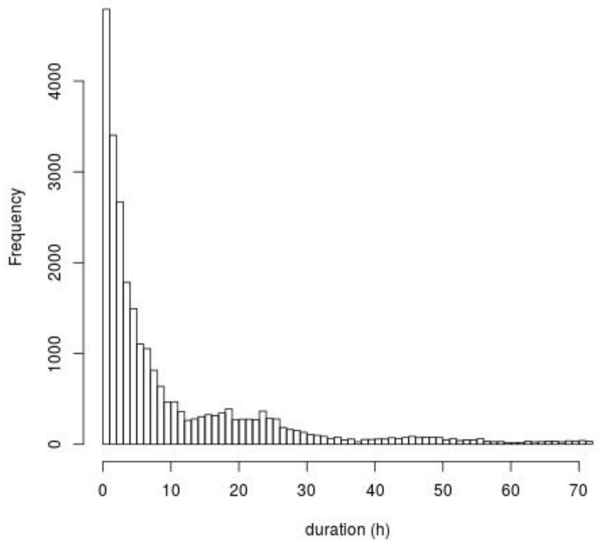


Fig. 2.5 Histogram of the duration of line disconnections for those lasting more than 1 minutes and having been repaired in less than 72 hours.

conditions are unfit to such operations). We can also observe that most of the power lines are repaired less than 10h after the short circuit happens. This repairing process is not an exogenous random process but the results of a risk assessment policy. If the risk assessment policy is modified, for example by introducing a new security criterion, this time could be impacted and should not be taken as constant. This issue will not be addressed in this work.

TSO then often operate their powergrid using the "N-1" security criterion. This criterion states that, if any *single* element[12] in the powergrid were to be damaged, the flow in every power lines should still remain below its thermal limit. If this is not the case, a "curative action" must be implemented to set the power grid back in security before the protection disconnects the concerned lines. A more formal definition of a "curative action" will be detailed in the section 2.3 of this chapter. The "N-1" means here "if the powergrid loses one component, then there would still be $n-1$ power lines available (below their thermal limit)".

Assessing the "N-1" security of a powergrid today requires a lot of computational resources. It requires a first load-flow to compute whether or not the grid is in "N" security, and n more load-flows to simulate the impact of the failure of each powerline,

[12]We remain loose about the terminology equipment here on purpose. This rule has some differences depending on the TSO's. In its most strict version an equipment can be anything on the grid: from a tower to a powerline, to a busbar for example.

i.e. compute the resulting flows on each line of the grid, after the possible failure of this element.

If TSOs were to compute the security in "N-2", *i.e.* taking into account external causes that could lead to the disconnection of two power lines, it would require $\frac{n(n+1)}{2} + 1$ load flows computations. With $n = 10^4$, the approximate number of power lines in the French power grid, this would lead to evaluate $\approx 5.10^7$ load flows. The current software takes of the order of 100ms to compute one load flow. Assessing the "N-2" security would require the equivalent of 5.10^6s of sequential computation (approximately 2 months).

Another possibility would be to parallelize this computation: in that case, TSO would need approximately 16 667 cores to perform these 5.10^7 load-flows in less than 5 minutes. This explains why TSOs limit themselves today to the "N-1" security assessment, that takes "only" a few minutes of sequential computation per grid state considered[13]. On the contrary, assessing higher order security, such as considering the failure of all possible triplets of elements, is today impossible for TSOs. Enforcing these higher order security measure has not been a priority in the past: the probability of suffering two lines disconnection is almost negligible compared to the probability of suffering one unplanned outage. This was true when the powergrid was operated in real time. The introduction of more and more complex equipment, the switching from centralized highly controlled generation units towards intermittent renewable energy sources and the difficulty to build new heavy infrastructures is calling to study the powergrid on an even longer time horizon. If this temporal window is longer than 10h, the probability to have two lines disconnected becomes much higher: one line can disconnect at a given date, and another a few hours later, while the first one has not yet been repaired.

2.2.3 New conceptual security frameworks

Nowadays, many changes are occurring. The loads are more and more complex: some new behaviors are emerging. Disruptive technologies, such as electric cars, might have a big impact on the power grid, as well as "prosumers" *i.e.* people that possess batteries, solar panel or wind turbine at home, and depend less on the power grid for their power supply.

The environment is also changing on the production side. An increasing amount of energy is now produced via renewable sources, wind turbines, and solar panels. This has two main impacts. First, they are weather dependent. As the weather is difficult to

[13]We note that in both cases, RTE is effectively using HPC (High-Performance Computing) and parallel processing to reduce the computation times.

forecast days in advance, it is even harder to predict how much power will be produced. Even worse, they can only produce when the weather conditions are suited: you cannot control them, while traditional sources of energy are more easily controllable.

Moreover, the current energy market (where traders can sell and buy electricity almost in real time) has an increasing role in power demand. What used to be done over the counter, between big producers, long time in advance, is now happening at a much quicker rate (every 15 minutes in some cases), between a greater number of actors: producers, traders etc.

The power grid is, therefore, facing new challenges. But in the meantime, in most industrial countries, such as France, the average annual power demand has stopped increasing. Building new heavy infrastructures (*e.g.* new power lines) is becoming hard to justify in this context, moreover, the public acceptance of such infrastructure impacting the landscape becomes very low. This means that new flexibilities have to be found on the actual power grid to tackle the challenges addressed by this new, difficult to predict and to control environment. This also implies that the current aging infrastructures must be operated closer to their limits while keeping a really high quality of service, including few blackouts, and no electricity shortage.

To tackle these new challenges, RTE has recently participated in two European pojects. The first one, **ITESLA ("Innovative Tools for Electrical System Security within Large Areas"** of which a full description is available at http://www.itesla-project.eu/) focused on large pan European systems. It also aimed at simulating the impact of **contingencies** on the system taking into account how TSOs operate (*i.e.* by trying to find corrective remedial actions that bring back the system states inside their nominal ranges, should an outage occur.) This project counted many aspects, addressing many issues faced by TSO today (handling power grid at a pan European level, taking into account dynamical phenomena ignored by steady-state modeling exposed here etc.). The most relevant aspects of this project for this work is a proposed workflow for classifying contingencies in multiple categories:

1. **Learning "security rules".** This training will not be performed in real time and consider historical records of past power grid states, as well as new, but realistic, grid states. These security rules aim at classifying the contingencies (failure of equipment) into 4 parts:

    - "good": the grid is still in security after the outage;
    - "curable": there exist at least one curative action that can be implemented, and that allows the power grid to get back in security after the outage;

- "preventive needed": there is no corrective action that can, alone, be implemented in real time and totally cure the grid. But if a specific action had been implemented a few hours before, some curative actions would set back the grid in a secure state
- "not handled": no known action can cure the power grid if this outage were to occur.

2. **Applying "security rules" in real time** to assess the dangerousness of contingencies on a grid state. A careful simulation would only be carried out for contingencies that are not in the "good" category. This would allow reducing the computational cost (provided that the "security rules" are fast to assess). This reduction in the computational cost could allow us to evaluate the security of a higher number of grid states, in a Monte Carlo like framework for example, provided that we are able to generate realistic and plausible future scenarios.

The second project is more theory-oriented. It is the **GARPUR** European project (**"Generally Accepted Reliability Principle with Uncertainty modelling and through probabilistic Risk assessment"** see https://www.sintef.no/projectweb/garpur/ for complementary informations). It evaluates the flaws of today's security policy (the "N-1" criterion) and gives some recommendations on how to better take into account the new environment in which the power grid is operated.

One of the outcomes of this project was the definition of a probabilistic reliability criterion that should replace the N-1 rule, and which is expected to lead to a better social-welfare. The definition of such a reliability criterion can be found in [26] for example.

In this chapter and the rest of the manuscript, z will denote an outage (disconnection of a powerline due to external causes). For example, z_1 corresponds to "the power line 1 is disconnected". Each outage z is associated to a cost $c(z)$. This cost should take into account multiple components, such as the cost of the corrective action that taken after the occurrence of outage z (if any), as well as the cost of service interruptions (blackout) implied by outage z. The outage z also has a probability of occurrence, that we will denote by $p(z)$ that depends, among other factors, on the weather conditions[14] , on the aging of the infrastructure and on the way the maintenance has been performed on this specific equipment. In this framework, the risk of a grid state $s = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\tau})$[15]

[14]There is a higher chance that a line trips (*i.e.* that an automaton detects an anomaly and disconnect this powerline) during a thunderstorm for example.

[15]For a sake of simplicity, we detail the work performed on this project supposing that this grid state s is deterministic. The GARPUR methodology also takes into account possible stochastic powergrid states. More details about the GARPUR project can be found at https://www.sintef.no/projectweb/garpur/.

where $\boldsymbol{x}$ are the injections, $\boldsymbol{y}$ the flows and $\boldsymbol{\tau}$ represents the topology is then:

$$\underbrace{R(s)}_{\text{total risk}} = \sum_{z \in \mathscr{A}} \underbrace{p(z)}_{\text{Probability of occurrence}} \cdot \underbrace{c(z;s)}_{\text{Associated cost}} \tag{2.18}$$

where $\mathscr{A}$ denotes the set of all possible contingencies that the system can endure.

The "N-1" strategy can be interpreted as a particular case of the more generic GARPUR reliability criterion, where heavy assumptions are made. It can be seen as neglecting all contingencies resulting in more than two lines disconnections (*i.e.* restrain $\mathscr{A}$ to single contingencies), associating the same cost $c(z)$ for all single contingencies and supposing that each disconnection has the same probability of occurring $p(z)$. As explained before, new flexibilities in the operations of the power grid must be found, and even if the "N-1" security policy has proven to be successful in the past, it must be refined in the future.

However, and despite its flaws, the “N-1” approach presented the great benefit of simplicity, while there are multiple difficulties in the computation of the proper risk defined in equation 2.18. First, $\mathscr{A}$ is an extremely large set of contingencies. For a power grid counting n power lines, card$(\mathscr{A}) > 2^n$ (recall that n is the number of powerlines of the system). A power grid can count $n = 10.000$ powerlines, making the computation impossible to carry out ($2^{10\ 000}$ is a number with 3.011 digits). The estimated number of atoms in the universe "only" has approximately 82 digits). Hence, at the moment the GARPUR framework corresponds to a sound approach but is not tractable with the current tools available to the TSO's. Secondly, there is no easy way to compute the cost associated with an outage $c(z;s)$ in most of the cases. The simulation of a loss of a power line must be carried out using really complex simulators, that can take into account multiple phenomena that are most of the time ignored by power flow solvers. For example, power flow solvers make the hypothesis the power grid is in "steady state", which does not hold when the power grid is heavily stressed for example.

2.3 Modeling a perfect operator

In the previous sections, we discussed the definition of a power grid and the kind of problem it could face. In this section, we explain how a power grid is operated *i.e.* what is a "perfect **dispatcher**". This section is organized as followed. First, we explain what actions the operator can perform to maintain a given reliability level. Then we expose a model of what a perfect operator could do. This model has been introduced in the GARPUR methodology.

2.3.1 Possible actions

We can distinguish two main types of actions. “Corrective remedial actions” can be implemented after a problem arises. They are often quick to implement and are mostly related to topology. On the contrary “preventive remedial actions” must be performed before the arising of outage z. Most of the preventive actions are related to “redispatching”, *i.e.* connecting or disconnecting a production for example. This can take up to a few hours. Table 2.2 shows in more detail the main actions dispatchers can perform. Some are impossible to do in real time, for example starting and ramping up quickly a nuclear power plant to a desired production level. Indeed, a power plant needs sometimes more than 8 hours in order to be connected to the power grid. Similarly, power plants need some time to cool down, and once started, they must produce electricity for at least a few hours, so switching off a power plant cannot always be performed.

Other types of action are used only in extreme cases because they can damage either the TSO infrastructure or the power quality on the customer side. We include these types of action, but they are seldom used. Such actions include reducing the voltages everywhere or performing load shedding for example.

Table 2.2 **Description of some actions implemented by operators**$^+$ when a problem appears, as well as their related cost and their applicability in a curative action setting: can they be applied rapidly or do they have important implementation time?

Name	Description	Implementation time	priority
topological action	change buses at substations	few minutes*	high
renewable curtailment	reduce wind / solar panel production to a given threshold	few minutes to few hours	average
demand side management	pay consumers for reducing their consumption at a given time	must be decided a day before usually	low
redispatching	imposing producers to produce more / less	few minutes to few hours (	low
load shedding	cut off an aggregate of consumers within an area	a few hours (with caution, under some conditions)	very low
sub-nominal voltage setting	decrease nominal voltage Vn by 5%	few seconds to few minutes†	extremely low

$^+$ This table is a brief presentation of some of the actions an operator can perform.
* Some reconfiguration needs more than a few minutes to be performed in certain cases. These cannot be used in real time.
† This depends if the production units is connected or not to the power grid, is it’s already producing or not, and of the type of the productions (thermal, hydraulic, etc.)

To cure a real problem, we will call π the corrective remedial actions action a dispatcher can perform. This can be a complex action, for example starting a production unit one day in advance, and modifying the topology when the outage occurs. This notation comes from the "reinforcement learning framework", where the "policy", *i.e.* a function that chooses the (best) action to do in a given state is denoted by π. In other fields, such as the optimal control literature, this function is often called u.

This action π can be implemented in two separated parts. The preventive part, implemented in advance, before any outage arises. This part can include starting a new power plant or paying industrials to consume less for example. This preventive part is associated with a cost of $CP(\pi, s)$ and depends only on the system state s and the chosen action π.

The action π can also have a corrective part. The corrective part will only be paid if an outage occurs. For example, we could choose to perform a topology change in substation 1, **only if** the power line 3 is disconnected due to external causes. To this corrective cost is associated with a cost of $CC(\pi, s, z)$. This cost is dependent on the outage z considered.

In any case, the power grid state after applying the action π will be denoted by $s \odot \pi$.

2.3.2 Assisting operator with machine learning

In the previous subsection, we showed that operators can take different actions in real time. In this section, we report how the GARPUR framework models the decision-making process: this is what we call the "operator's problem". We will explain what a "good" remedial action π is, and how we can quantify it. Our formulation is largely inspired by the GARPUR framework and is shown in [26] equation 1 for example. The dispatcher should take the action π that minimizes some quantity, that takes into account multiple aspects of the security criterion. The cost of an action includes its preventive cost CP, its corrective cost CC. Both these costs depend on the outage considered, *e.g.* the cost of the fall of lines 1 is not the same as the cost of fall of powerline 2. The GARPUR framework doesn't enforce that a policy π must prevent any blackout: this is taken into account by associating a cost to the power grid state $s \odot \pi$ if outage z arises. The best action π^* that an operator can choose for a grid state s is[16]:

[16]In the GARPUR framework is also presented a method to take into account uncertainties in the grid state s *e.g.* if injections are not known perfectly in case of forecasted grid state for example. As we already stated, we don't consider it in the modeling of the operator we present here.

$$\pi^* = \text{argmin}_\pi \left\{ \underbrace{CP(\pi, s)}_{\text{Preventive cost}} + \sum_{z \in \mathscr{A}} \underbrace{p(z)}_{\text{probability of occurrence of } z} \underbrace{\left(\underbrace{CC(\pi, s, z)}_{\text{Corrective cost of z}} + \underbrace{c(z; s \odot \pi)}_{\text{cost of fixing the power grid}} \right)}_{\text{cost of the grid after outage } z \text{ arises}} \right\} \tag{2.19}$$

Finding the best action π^* is a really difficult task. It has the same level of computational complexity as the one involved in computing the risk of a power grid state defined by the GARPUR methodology (see Equation 2.18). The problems come from the huge set of possible contingencies and the difficulty to accurately simulate the behavior of the powergrid for some of these contingencies and for all the credible uncertainties[17]. Thus, solving the optimization problem 2.3.3 is not tractable relying solely on the methods currently used by TSO's. As stated in the introduction, recent progress in "machine learning" and especially "deep learning" methods achieved good performances in multiple domains, such as computer vision [21], natural language processing [43], and game playing (with the Go [44] or the game of poker [5]). In this work, we show that these deep learning methods can be implemented to assist operators. The main algorithm developed, called "guided dropout" allows to learn the outputs of a simulation engine based on physical laws (in our case Hades2) and is able to rapidly compute:

- the effect of taking action π on grid state s, *i.e.* computing the state $s \odot \pi$
- the effect of a given contingency z on the grid states s or $s \odot \pi$
- the risk as defined in equation 2.18 by taking into account a more important number of contingencies compared to what operators study nowadays.

In this thesis, we do not address directly the problem of finding the optimal policy π^* explicitly. Our goal is to assist operators to make decisions as best as possible according to this criteria. The implementation of the proposed method, at least in a first version, would remain in the hands of the operators.

[17]The problems of assessing uncertainties in powergrid state will not be addressed in this work

2.3.3 Related works

In this section, we provide some insight into other methods that have been used in the power system community to tackle the operator's problem.

One of the first method that can be seen as solving the operator's problem is the OPF "**O**ptimal **P**ower **F**low", and has been first formulated by Carpentier in 1962 [7]. This problem, part of the continuous optimization, aims at finding the best dispatching strategy for productions units such as to minimize the losses for example. When coupled with the "N-1" security criteria (a simplification of the equation 2.18 presented page 22), the formulation of the "OPF" can be refined and is therefore referred at "SCOPF" for "**S**ecurity **C**onstrained **O**ptimal **P**ower **F**low". This approach has been developed for example in [34] or [6].

In these papers, the main control variables are the **productions** and the constraints of the problem are the Kirchhoff's equations and the thermal limits on the lines. Our work is more focused on the topology since topology is under TSO's control. These previous methods are not particularly fitted, even though some adaptation exists to take into account some topological actions ([41] or [55]). We decided to focus our work on assisting operators rather than optimizing the problem presented in equation . That's we decided not to pursue the work presented in these papers.

2.3.4 Summary

In this chapter, we detailed the different elements present in a power grid. In this work, the mains elements will be:

- the productions, that produces the power;
- the loads, that consumes the power;
- the transmission lines that allows the power to flows from the productions to the loads.

The way the power flows on the power lines depends on many factors, for example the physical properties of the power lines and follows Kirchhoff's laws. In this work, all the productions and loads values have been gathered in a vector denoted by $\boldsymbol{x}$. On the other hands, the flows on each powerlines of the grid are denoted by $\boldsymbol{y}$.

Some flexibility exists on a power grid. Under certain circumstances, operators can change the way the powerlines are interconnected with one another. The way these powerlines are connected is denoted by "topology", and the action to modify it is called "topological change". The topology has an impact on the power flows on the power grid. This is one of the preferred way to manage a power grid and ensure it remains in security. In this work, the topology is represented by a vector of $\{0,1\}$, called structural vector and denoted by $\boldsymbol{\tau}$.

Today, the process of computing the flows from the productions and the loads $\boldsymbol{x}$ and the topology $\boldsymbol{\tau}$ is called "a power flows computation" and carried out with slow, but accurate simulators. The main contributions of this work is to be able to approximate this computation. By using machine learning, we show that we could gain a 300 times speed up in computation time, while still being accurate. The developed learning machine is also able to generalize to new topologies, a phenomenon we call "super generalization".

The methodology adopted, as well as the data we used to train such learning machine and the results that are achieved on controlled experiments as well as on experiences carried out on real data. However, the next chapter introduces the basis of "learning machines" and some theory behind them.

Chapter 3

Artificial Neural Networks

In the previous chapter, we formalized what a would be “perfect operator” could perform. Unfortunately, this problem is nowadays intractable relying only on methods used by TSOs. Machine learning can be an alternative to assist operators and tackle some of the problems exposed in the previous chapter. In this chapter, we introduce the basics of “machine learning” and **neural network**. We also show how some of these methods compare with each other using the small test case presented in the previous chapter, as an example to show how the results on more real datasets will be presented in future chapters.

This chapter is organized as follow. The first section details first the linear regression, one the oldest statistical model to finish on a short description of the neural network. The second section detailed, on a small example, how the concepts introduced in the first section can be implemented to solve the problem of predicting power flows on a small size powergrid.

3.1 From linear regression to artificial neural networks

The original idea behind a deep artificial neural network comes from a machine introduced by Rosenblatt in [40] called the "perceptron". The first section of this chapter is devoted to understanding how this learning machine works. As a first step, we introduce linear methods, that can be seen as a particular case of a neural network. Then we detail how the linear regression has been improved in the literature. Then we provide an introduction to deep learning. Finally, the last section reports the methods we used to train our models.

3.1.1 Linear regression

Linear regression is a statistical tool used to model data since the mid-1800's. The general principle is to predict an univariate outcome $y \in \mathbb{R}$, with p input variables $\boldsymbol{x} \in \mathbb{R}^p$. For example, this model allows to predict the flow on the **transmission line** 1 given all the **injections** on the **power grid** shown in Figure 2.1a.

One could build a prediction $\hat{y}$ for the variable y by making a linear combination of variables $\boldsymbol{x}$ with some weights $\boldsymbol{w}$, with $\dim(\boldsymbol{w}) = (p,1)$. Given N data points $(\boldsymbol{x}_j, y_j)_{1 \leq j \leq N}$, such a linear combination for example for $\boldsymbol{x}_j$ $(\dim(\boldsymbol{x}_j) = (1,p))$ can be written[1] as

$$\hat{y}_j = \boldsymbol{x}_j.\boldsymbol{w} \tag{3.1}$$

Linear regression searches the linear combination of $\boldsymbol{x}$ that minimizes the distance between $\hat{y}$ and y given some distance function d. More formally, this solves the optimization problem shown in 3.2 with respect to the coefficient vector $\boldsymbol{w}$.

$$\boldsymbol{w}_{\text{opt}} = \text{argmin}_{\boldsymbol{w} \in \mathbb{R}^p} \sum_{1 \leq j \leq N} d(y_j, \boldsymbol{x}_j.\boldsymbol{w}) \tag{3.2}$$

In the case of a linear regression, the distance d is often the l_2 norm, defined as :

$$l_2(z,t) \overset{\text{def}}{=} (z-t)^2 \tag{3.3}$$

There exist multiple algorithms to find the best parameters $\boldsymbol{w}_{\text{opt}}$. For example in [16] (equation 3.6 page 45) Friedman et al. show that[2]:

$$\boldsymbol{w}_{\text{opt}} = (X'X)^{-1}X'Y \tag{3.4}$$

where X is the "design matrix" of size (N,p): the j^{nth} row of this matrix being row vector $\boldsymbol{x}_j$ (of size p). Y is the $(N,1)$ column vector such that $Y_j \overset{\text{def}}{=} y_j$. We denote by X' the transpose of matrix X. In this formulation, w_{opt} is a column vector of size $(p,1)$.

Another method to solve this optimization problem is via gradient descent. It is shown in [16], that the linear regression problem is convex, so a gradient descent algorithm can be applied to its resolution. Such an algorithm is described below in the Algorithm 2:

[1]The following formulation does not explicitly mention a bias in its formulation. In fact, the bias can be added by concatenating a constant "1" to each vector $\boldsymbol{x}_j$.

[2]Supposing that the matrix $X'.X$ is invertible, where X' denotes the transpose of matrix X.

```
Input: $(\boldsymbol{x}_j)_{1 \leq j \leq N}$ and $(y_j)_{1 \leq j \leq N}$ and learning rate $\lambda$
Output: $\boldsymbol{w}$ with $\dim(\boldsymbol{w}) = (p, 1)$
    Initialization :
 1: $\boldsymbol{w} \leftarrow [0, \dots, 0]$
    Main loop :
 2: while Stopping criteria not met do
 3:     Compute gradients:
 4:     for $k \in [1, 2, \dots, p]$ do
 5:         $\nabla w_k \leftarrow \frac{\partial \sum_{1 \leq j \leq N} ||y_j - \boldsymbol{x}_j.\boldsymbol{w}||_2}{\partial w_k}$
 6:     end for
 7:     Update the coefficients:
 8:     $w_k \leftarrow w_k - \lambda \; \nabla w_k$
 9: end while
10: return $\boldsymbol{w}$
```

Algorithm 2: Resolution of a linear regression by gradient descent

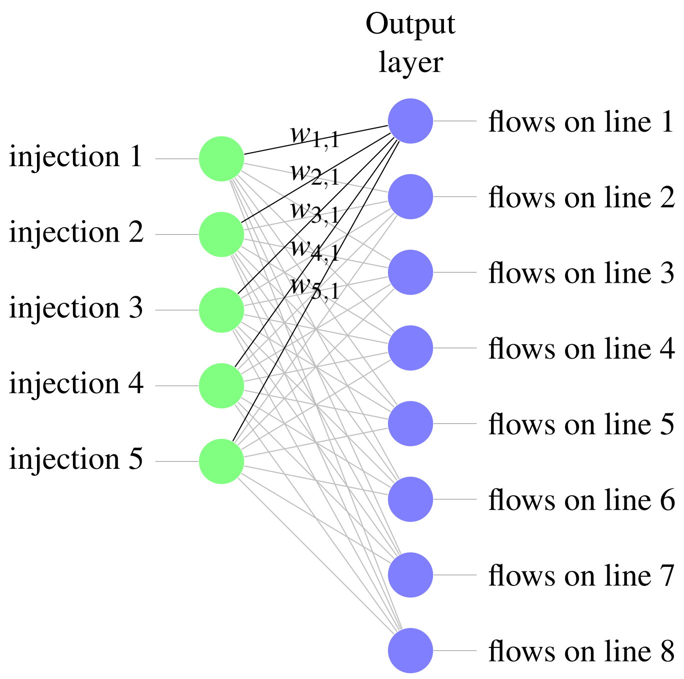


Fig. 3.1 **Representation of a linear model** that computes the flows $\boldsymbol{y}$ on each power line of the system shown in figure 2.1 from the injections $\boldsymbol{x}$. The green dots (left) represent the inputs, in this case the 5 **injections** (2 **productions** and 3 **loads**). The purple dots (right) show the output, in our example the flows on the **transmission lines**. Each straight line represents a coefficient $w_{j,k}$ of the weight matrix W. We highlighted the coefficients for predicting the flow on power line number 1 (first column of W).

3.1.2 Extensions of the linear regression

Improving performance Variants of the linear model exist, that can show better performance (on unseen data) in practice. For example, when one can be looking for sparse parameters $\boldsymbol{w}$ (in the lasso regression [49]), *i.e.* searching among vector $\boldsymbol{w}$ for which the majority of coefficients are 0. One could also penalize parameter $\boldsymbol{w}$ that have bigger norm. This leads to the ridge regression [24]. Both of these problems can be solved in the elastic-net linear regression framework ([35], [57]), where the optimization problem solved is:

$$\boldsymbol{w}_{\text{opt}} = \text{argmin}_{\boldsymbol{w}} \sum_{1 \leq j \leq N} d(y_j, \boldsymbol{x}_j.\boldsymbol{w}) + \underbrace{\lambda_1 ||\boldsymbol{w}||_1}_{\text{Lasso penalty}} + \underbrace{\lambda_2 ||\boldsymbol{w}||_2}_{\text{Ridge penalty}} \tag{3.5}$$

This model sometimes leads to better performance, *e.g.* a lower error on unseen data. And it has another advantages: it allows us to compute a linear regression, even when the design matrix XX' is not invertible.

Handling multiple outputs One of the limits of the previous model is that it can only predict a single outcome. In practice, this model can deal with cases where y counts $n > 1$ dimensions. In that case, it is called "multiple outputs linear regression". In our case, it can be used for example for predicting the flow on each line of the power grid rather than the flow on a single power line. Such a model is, in fact, equivalent to fitting n independent linear regressions and stacking the coefficients[3]. Mathematically, this takes exactly the form given in 3.2, but instead of having p coefficients $\boldsymbol{w}$, now W is matrix of shape (p,n), we speak of "weights matrix", and prefer the upper case notation W to the bold lower $\boldsymbol{w}$. All the other equations remain the same when adopting these notations.

Non-linear extensions The two previous extensions are not the only extension that can be built on the linear regression methodology. In fact, it is possible to fit non-linear least square estimators, to impose constraints on the set describing the weights etc. A more detailed view about all these methods can be found in [25] for example. In the next chapter, we detail the **neural network**, another statistical model that is part of the non-linear models.

In this section, we saw the basis of statistical learning, with linear regression. We provided some methods that generalize in some sense the linear regression. In the next section, we will expose another learning machine call the artificial **neural network**.

[3]Under the hypothesis where the linear regression is not penalized.

3.1.3 Artificial Neural Network

The major issue of linear regression lies in its inability to learn the non-linear mapping between the inputs and the outputs. In most real-world applications, as in our specific problem of flow prediction, outputs are not linear combinations of the inputs. Different kinds of models allow to tackle these non-linearities between inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. We choose to focus on the "artificial neural networks". In [40], Rosenblatt introduced a learning machine loosely inspired by the brain architecture. Today the terminology "artificial neural network" is preferred. His idea is to introduce some hidden layers of units within the network as intermediate predictors which gradually transform the information and learn the relation between inputs and outputs. It is equivalent to stacking multiple linear regressions. If only linear combinations were performed within these hidden layers, the model would remain linear. To allow non-linearities in the model, people often perform a non linear element-wise operation for each hidden unit before passing its value to the next layer. This non linear operation is called an "**activation function**".

$$\forall\, 1 \leq l \leq (\eta - 1),\ \boldsymbol{h}^{(l+1)} = \phi_l \left(\boldsymbol{h}^{(l)} . W^{(l)} \right) \tag{3.6}$$

where ϕ_l is the "**activation function**" of layer l and $W^{(l)}$ its weight matrix. In the above equation, η is the number of layer of our neural network. The representation of such a "multilayer neural network" is shown in figure 3.2. If, similarly to linear regressions, a dataset of $(x_j, y_j)_{1 \leq j \leq N}$ is available, training a multilayer neural network on this dataset means solving the following optimization problem:

$$\boldsymbol{W}_{opt} = \text{argmin}_{\boldsymbol{W}} \sum_{1 \leq j \leq N} d(y_j, h^{(\eta)}) \tag{3.7}$$

where $\boldsymbol{W}$ is the object containing all the weights matrices $W^{(l)}$ for each layer l of the neural network.

There exist various algorithms to perform this optimization as shown in [19] (chapter 4 section 3, and chapter 8 in particular). The most common approach is to use the "stochastic gradient descent" algorithm. The gradients are computed with the back-propagation algorithm described in [32], which is the application of the chain rule.

The stochastic gradient descent algorithm is really similar to the gradient descent algorithm showed in Algorithm 2. The main difference lies in the minibatch selection. Instead of computing the gradient for the entire dataset (line 4 of the algorithm 2 where the index j run through all the training set: $1 \leq j \leq N$) this algorithm selects only,

```
Input: $(x_j)_{1 \leq j \leq N}$ and $(y_j)_{1 \leq j \leq N}$ and learning rate $\lambda$ and minibatch size $b$
Output: $\boldsymbol{W}$ (stacking of the $\eta$ weights matrices) $\boldsymbol{W}^l_{j,k}$ is the coefficient at position
    $j,k$ of the weight matrix of the $l^{\text{th}}$ layer.
    Initialization :
 1: $\boldsymbol{W} \leftarrow$ random tensor
    Main loop :
 2: while Stopping criteria not met do
 3:     $(x_k, y_k)_{1 \leq k \leq b} \leftarrow$ random sampling of $b$ example of $(x_j, y_j)_{1 \leq k \leq N}$
 4:     for $l \in [1, 2, \dots, \eta]$ do
 5:         $\nabla W^l \leftarrow \frac{\partial \sum_{1 \leq k \leq b} \left|\left| y_k - h^{(\eta)} \right|\right|_2}{\partial W^{(l)}}$
 6:     end for
        Update the coefficients
 7:     $W^p \leftarrow W^p - \lambda . \nabla W^p$
 8: end while
 9: return $w$
```

Algorithm 3: Stochastic gradient descent algorithm

uniformely at random, b elements (line 5 of algorithm 3 where only k examples are seen).

A representation of an artificial neural network is shown in figure 3.2. On this figure, we can see a **neural network** with 5 inputs (the 5 **injections**) showed in green, 8 outputs (blue) each one representing the flows on a particular power-line of the power-grid. This neural network counts only 1 hidden layer of 6 units represented in red dots. The weight matrices $W^{(1)}$ (resp. $W^{(2)}$) is represented by all the coefficients linking green units to red units (resp. red units to blue units). The non-linearities ϕ_1 and ϕ_2 are not represented in this schematics.

Besides the ability to learn non-linear functions, the **neural network** offers multiple advantages compared to a linear regression. In contrast with shallow linear regressions detailed in Section 3.1.1, some parameters are "shared" for all outputs. As we explain in section 3.1.1, learning a linear regression with n outputs is equivalent to fitting n separate linear regressions. As we can see on figure 3.2, this is not the case for **neural networks**, as the weight matrix $W^{(1)}$ is used in the prediction of all flows on the grid. The role of this weight matrix is to transform the injections (green units) into different variables (red units) that can be used to predict the outputs (blue units). An internal representation is being learned.

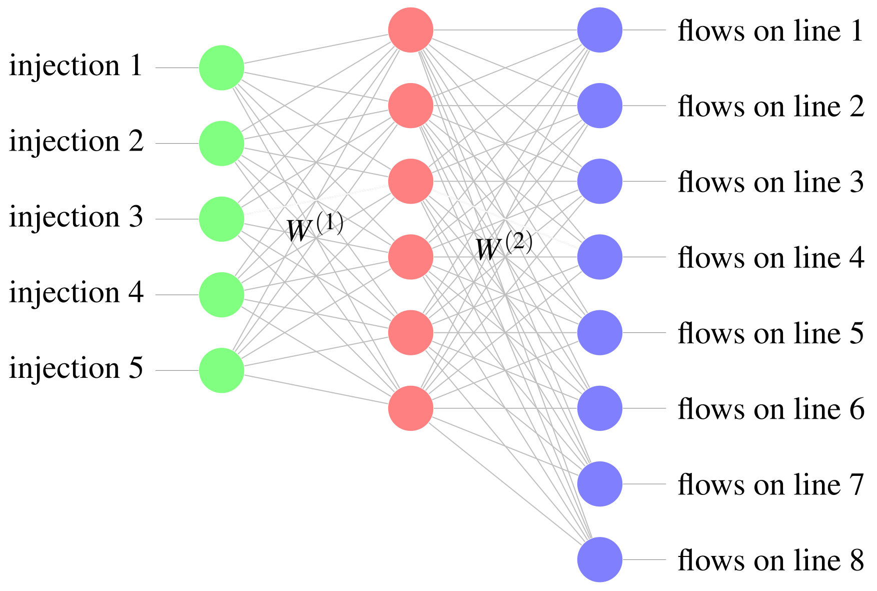


Fig. 3.2 **Representation of a *neural network***, also called "fully connected neural network" with 5 inputs (green circles - left), 8 outputs (blue circles - right). It counts one hidden layer of 6 layers, represented by the the 6 centered red circles. The straight gray lines represents the connections (weights) and the two weights matrices $W^{(1)}$ and $W^{(2)}$ have been represented.

3.1.4 Training procedure

The previous models were specific cases of a broader literature called "statistical learning" or "machine learning". In this section, we will briefly explain the precautions we used when training our models, and report the errors in this manuscript.

The goal of "machine learning" is to use a finite dataset to and learn something from it. For example, in a regression setting, the dataset often has the form $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{1 \leq i \leq N}$, where N is an integer representing the size of the dataset. The objective is to learn the best function $f_{\boldsymbol{w}_{\text{opt}}}$ that depends on the parameters $\boldsymbol{w}$ such that $f_{\boldsymbol{w}_{\text{opt}}}$ minimizes the distance between data $f_{\boldsymbol{w}_{\text{opt}}}(\boldsymbol{x})$ and the corresponding $\boldsymbol{y}$.

Because we want to estimate how well our estimator $f_{\boldsymbol{w}_{\text{opt}}}$ performs even on unseen data pairs $(\boldsymbol{x}, \boldsymbol{y})$. The error computed on unseen data is called the "generalization error". It is common to evaluate its performance on an unseen dataset. This dataset, not used to find the parameters $\boldsymbol{w}_{\text{opt}}$ is called a "**test set**". It is not used in any way, except to report the error.

Let's explain why this test set is used only once. Let's suppose we have multiple models that we want to compare. For example, in the predictions of the flows on power line 1, one could use the value of all the injections as input, or just restrain it to a subset of these. We could use the same "test set" but it will decrease the quality of approximation of this dataset. Indeed, informally the odds of finding a model that works on this specific test set increases with the number of models tested, even if these models have not learned anything. The more this test set is looked at, the less useful it will be to approximate the generalization error: the selected model will be the one that performs the best on the test set, and not the one that performs the best in all unseen dataset. To avoid this problem, people often use a set to validate which model among the tested method is the best to suit their need on a separate "validation set". The validation set is then used to select the model (for example the input vector, the learning rate etc.) that will be used.

So, to summarize what we explained in this section.

- The goal of machine learning is, given a class of functions f parametrized by weights $\boldsymbol{w}$:
  - to compute which is the best representative $f_{\boldsymbol{w}_{\text{opt}}}$,
  - best meaning that $f_{\boldsymbol{w}_{\text{opt}}}$ allows to minimize the error on a unseen test dataset
- To achieve this task, data are often split[4] into three:
  - **Training set**: is used to train the model: Given a class of functions $f_{\boldsymbol{w}}$, it will be used to select $\boldsymbol{w}_{\text{opt}}$.
  - **Validation set**: is used to choose the best class of functions among different one. The process is the following: for each class of functions, train the model coming from this class (compute $\boldsymbol{w}_{\text{opt}}$ for each class of model). Then compute the error on the validation set for each $f_{\boldsymbol{w}_{\text{opt}}}$. And select the one that has the lowest error on this set.
  - **Test set**: is used to report the error on the final model. It is not used otherwise. Error reported on this manuscript are then an approximation of the generalization error.

This whole procedure has been strictly observed throughout this manuscript. Unless explicitly indicated, errors reported are errors on the test set.

3.2 Predictions of flows and first deep neural networks

In the previous sections, we explained how to train 2 different statistical models: the linear regression, and the multilayer perceptron (also called fully connected (deep)

[4]There exist multiple method to split them. One commonly used is to randomly assign each example of a given dataset to one of the Training, Validation or Test set. This random assignment have been observed in for the error reported in this manuscript.

neural network). In this section, we will show how these models can be used to solve the problem presented in 2.3. Then we will explain how to generate synthetic yet realistic data on which we perform some first experiments as well as the experimental protocol and the quantitative metrics used to report the results. In the last part, we evaluate the performance of these two models to predict the flows of the power grid presented in 2.1a.

3.2.1 Motivations

In section 2.3, we adopt a formalism that allows us to model a “perfect operator”. We also emphasized our main contribution is to be able to assess rapidly the impact of contingencies (denoted by z) and operator's actions π on the power grid so that the power grid remains in security.

The main issue is to compute the flows given some injections and topology. Once the flows are computed, assessing whether or not the power grid is in security is a simple task, as only n comparisons must be carried out: one for each power line to assess if the flows are below their thermal limit or not.

In the machine learning literature, $\boldsymbol{x}$ often denotes the inputs of the problem. In our case, $\boldsymbol{x}$ will be the vector of all the injections. The values that we want to predict, the flows in our case, will be denoted by $\boldsymbol{y}$. To differentiate between the actual flows[5] and the one predicted by our model, the prediction will always be hatted. So $\boldsymbol{y}$ will be the ground truth, the value that we need to predict, and $\hat{\boldsymbol{y}}$ will be the value predicted by the model. Typically, $\hat{\boldsymbol{y}}$ will be obtained by a neural network, parameterized by weights $\boldsymbol{W}$: $\hat{\boldsymbol{y}} = \text{NN}(\boldsymbol{x}; \boldsymbol{W})$.

This representation, which is standard in the machine learning literature, is not sufficient for our needs as we have another type of inputs: the topology of the power grid, as well as the outage z and the operator decision π. We decided to group all this in a vector that we chose to name “structural vector” and denote by $\boldsymbol{\tau}$.

The problem we are trying to address is then to find a suitable neural network architecture NN that, once the weights $\boldsymbol{W}$ have been optimized, is able to predict accurately the flows on the power grid in different configurations $\boldsymbol{\tau}$ and different injections $\boldsymbol{x}$. The relation we are really interested in is then:

$$\hat{\boldsymbol{y}} = \text{NN}(\boldsymbol{x}, \tau; \boldsymbol{W}) \tag{3.8}$$

[5]By "actual" here, we mean the flows computed with the physical simulator Hades2

In the next subsection, we report some first results obtained by a neural network. The main focus of this section is to introduce the concepts that are used to report results for more systematic experiments, more than the problem tackled itself.

3.2.2 Flow prediction in a fixed grid state

This section is devoted to present the first results of two learning machine algorithms. Both aim at predicting flows i_j on all line j of the power system given the active values of the injections ($p^{(p)}$ and $c^{(p)}$) for a fixed power grid topology. So in this setting, we have $\dim(\boldsymbol{x}) = 5$, $\dim(\boldsymbol{y}) = 8$ and $\dim(\boldsymbol{\tau}) = 0$ (no grid topology variants). The first model is a linear regression model, and the second one is an artificial neural network with one hidden layer of 10 units. The activation function used is the hyperbolic tangent.

The Figure 3.3 shows the learning curve of our models. In the y-axis is reported the training error (the l_2 error in this experiment) as a function of the training step performed when running Algorithm 2 for the linear regression model as well as the artificial neural network.

The dashed orange line presents the learning curve of the linear model, and the plain blue line the one of the artificial neural network. As we clearly see, the artificial neural network is almost all the time able to perform better than the linear model, except maybe at the very beginning of the learning. Moreover, these curves clearly exhibit one of the limits of the linear model: it is not able to learn a good linear relationship between the input and the output.

Table 3.1 Quantitative results on the predictions of flows using a fixed grid topology by linear regression and Artificial Neural Network ("ANN"). The errors reported are the MAPE and RMSE defined respectively in Equation 3.9 and Equation 3.10

Model name	MAPE (%)	RMSE (A)
DC approximation*	8.64	75.4
Linear regression	14.7	78.7
ANN	**8.17**	**47.0**

* The DC approximation is detailed in chapter 2.

More informations about the performance of these models can be found in the Table 3.1 where the **M**ean **A**bsolute **P**ercentage **E**rror (**MAPE**), defined in Equation 3.9 and the **R**oot **M**ean **S**quared **E**rror (**RMSE**), defined in Equation 3.10 are presented. These two error measures are commonly used when facing regression tasks. The MAPE is the average error, measure in percentage, and is scale invariant. The RMSE corresponds to the square root of the l_2 norm, and can be interpreted as being the (empiric) standard deviation of the error. As we see on this table, the linear regression model performs poorly, even in this very simple setting[6]. This is the main reason why we will not use it in more challenging settings, not even as a baseline and will prefer the DC approximation. On the contrary, the artificial neural network is doing pretty well on this task, outperforming the DC approximation in terms of relative error and RMSE.

$$\text{MAPE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N} \sum_{1 \leq k \leq N} \left| \frac{\hat{\boldsymbol{y}}_k - \boldsymbol{y}_k}{\boldsymbol{y}_k} \right| \tag{3.9}$$

$$\text{RMSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N} \sqrt{\sum_{1 \leq k \leq N} (\hat{\boldsymbol{y}}_k - \boldsymbol{y}_k)^2} \tag{3.10}$$

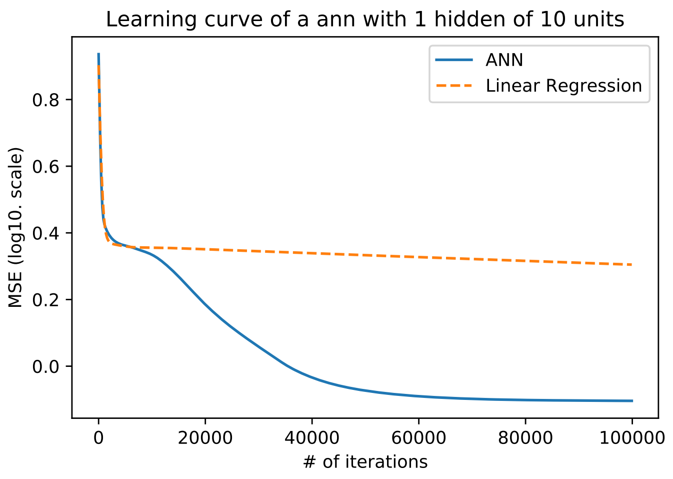


Fig. 3.3 **Learning curve** (*i.e.* the training error with respect to the number of iteration) of the linear regression model (orange dashed curve) and the artificial neural network (plain blue curve), trained on a fixed grid topology. Results presented here are computed on the training set.

In this subsection, we introduced some losses, the MAPE, and the RMSE often used in the literature when evaluating the performance of machine learning models in a context of regressions. We also explain how we could compare multiple models, motivate why we will not use simple linear regression, and introduce the "Learning curves" that can be used to decide if a model performs well in practice.

[6]Lots of data are available for predicting flows on a small grid without considering any topology variants at all.

In the next subsection, we will discuss some related work in the literature that also applies machine learning to power system problematics, and see how they are related to the problem treated in this manuscript.

3.2.3 Machine learning for power system analysis

As showed in the previous chapter, the goal of this manuscript will be to help operators to solve the problem presented in Equation 2.3.3, recalled here:

$$\pi^* = \text{argmin}_\pi \left\{ \underbrace{CP(\pi, s)}_{\text{Preventive cost}} + \sum_{z \in \mathscr{A}} \underbrace{p(z)}_{\text{probability of occurrence of } z} \underbrace{\left(\underbrace{CC(\pi, s, z)}_{\text{Corrective cost of z}} + \underbrace{c(z; s \odot \pi)}_{\text{cost of fixing the power grid}} \right)}_{\text{cost of the grid after outage } z \text{ arises}} \right\} \tag{3.11}$$

Some authors have presented a method to directly solve this problem. This is the case in [26] where Karangelos et al. directly address the problem formulated in 2.3.3. Their idea is to relax the optimization problem formulation by making some assumptions on the power grid. For example, they use of the DC approximation and linearize the formulation of the costs CP and CC etc.

All these approximations allow to cast the modeling of a perfect operator into a "Mixed Integer Linear Programming"[7] for which solvers exists. They use them to solve this problem on two medium size instances (studying the IEEE-RTS 96 grid introduced in [20]) and compare themselves to the "N-1" security criteria[8]. We did not pursue in this direction mainly because we believe this optimization problem won't scale for instances of the size of the French power grid. Also, we believe that artificial neural networks, fed with a lot of data, can be built their own representation of the power grid thus excluding the need of making such complex assumptions. One of the major drawbacks of the neural networks yet is the consistency. Being the DC approximation, or other methods, the flows $\boldsymbol{y}$ are compatible with the known structural constraints (Kirchhoff's law are respected). This property is not necessarily met if $\hat{\boldsymbol{y}}$ is obtained by machine learning algorithms.

Others have used machine learning to address power systems related issues. Most papers in this literature, address the problem of classifying grid states according to given security criteria (*e.g.* [51], [52], [42]). Other papers address the problem of

[7] A specific type of problem encountered in the optimization literature

[8] See section 2.2.2 for more information about this security policy.

building a "security index" of a power grid by taking into account both the flows and the voltages limits ([48]) using artificial neural networks or other learning machines (for example "restricted Boltzmann machines"). Compared to our problem, most of these paper assume that the topology of the power grid is known and that no corrective action, apart from re-dispatching, can be implemented. This is the main reason why we did not pursue in these directions.

In other papers, people adopt a two-stage decision-making process. They consider the real-time operation to be a subproblem of one in a much longer time horizon([15], [9]). Such example of longer time horizon problem includes the maintenance performed on the grid: What is the best period of time to repair towers? In both cases, the main focus is the longer time horizons, and less care has been taken to model how the operator reacts in real time. In this manuscript, on the contrary, the focus is on daily operation.

Chapter 4

Data presentation

As presented in Chapter 2, we address the problem of assisting human operators to operate power grids, in today's changing context, using *machine learning*. We aim at increasing security and reducing costs, or maintaining the same level of quality of service even when complexity rises. This section will be devoted to giving an overview of which data are available to us at RTE, the so-called "power grid snapshots" and the limitations in using them to learn predictive models.

As we motivated in the previous chapters, machine learning can be used to tackle the problem of assisting the operators in their choices. A dataset {inputs, outputs} should be available to learn the machine learning models, as we explained in Chapter 3. In Chapter 3 we also motivate our choice to write the dataset as $\{(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})\}$, where $\boldsymbol{x}$ is the vector with all the injections, $\boldsymbol{\tau}$ is the structural vector that depends on the topology of the powergrid and $\boldsymbol{y}$ are the flows on all the powerlines. In this chapter, we expose how to built this dataset to learn our models. This chapter is divided as followed, in the first section we explain how we use the French dataset and its limits. In the second section, we detailed how we generated some synthetic but realistic dataset to properly benchmark and further select learning models and architecture in controlled experiments. Finally, we show some work to label the French powergrid snapshots with the decisions that were taken and explain how this work will be used in further studies at RTE to learn with supervision to make decisions.

In this work, in all cases, the flows $\boldsymbol{y}$ are computing the simulators based on Kirchhoff's law Hades2 from the injections $\boldsymbol{x}$ and the topology[1]. The major difference between real and simulated dataset lies in the way the injections and the topologies are obtained. For the experiments on real datasets, the injections and the topologies are retrieve from a historical dataset. We didn't perform any action on them. On the

[1]This topology is encoded by the structural vector $\boldsymbol{\tau}$ for some experiments, $\boldsymbol{\tau}$ can encode for multiple topologies.

contrary, the simulated datasets allow us to explore more systematically the impact of topologies on the neural network performances: all relevant topologies are simulated. This allows the performing of systematic yet realistic experiments in these cases. A summary of the datasets used in this work is presented in Table 4.2.

Table 4.1 **Summary of the datasets used in this work**. In this work we will use two kinds of datasets. The first one is real data, an extraction of the French powergrid snapshots described in Section 4.1. The second is used in controlled experiments and is described in Section 4.2.

	French dataset	Synthetic dataset
injections $\boldsymbol{x}$	retrieved from the snapshots	sampled
Description of $\boldsymbol{\tau}$	lines disconnection	line disconnections or buses reconfiguration
flows $\boldsymbol{y}$	computed via the simulator Hades2	
Training Set Validation Set Test Set	fraction of the snapshots[*] for given time period	injections sampled[+] controlled topology
Super Test Set	fraction of the grid snapshots taken after the end of the training set	new structural vector $\boldsymbol{\tau}$, never observed in the training / validation / test set

[*] For each time period, the sampling of which grid states goes into which dataset has been done uniformly at random. If a grid state is selected in one dataset, it is not in the others. A grid state is chosen to be at most in the Training set, the Validation set **or** the Test set.
[+] The training set, the validation and the test all contain different grid states. There are no grid state that are present in two of these dataset.

4.1 French powergrid

In this section, we describe the data at our disposal, *e.g.* that data that can be used for training machine learning models or to validate approaches developed to assist operators.

Every 5 minutes, a comprehensive snapshot of the grid state is recorded and stored. We have available data from November 1st, 2011, to present times. Let's denote by $\{g_t\}_{1 \leq t \leq N}$ this collection of N powergrids, each g_t being the powergrid description at time t.

4.1.1 Description of the French dataset

For each grid state g_t, we have access to all the injections, approximately 7 000 loads and 3 000 productions, including active power values, reactive power values, and voltages. This makes $30,000$ variables solely for the injections.

On these snapshots, the flows on each powerline are also recorded. The French High Voltage and Extra High Voltage power grid count approximately $10,000$ lines, and for each of them, information about flows in active and reactive power, and in current units is stored for both ends of the powerline, which makes 6 informations for a single powerline. Globally $60,000$ variables related to flows are stored every 5 minutes.

On a power grid, there are also **buses**, which are mathematical descriptions of how the objects (*e.g.* productions, loads, lines etc.) in the grid are interconnected. The French extra high voltage counts approximately 6 400 buses. To redirect flows, as we explained in Chapter 2, operators can act on different breakers. In the France High Voltage and Extra High Voltage power grid, we count more than 30 000 breakers each being either open or closed. So a single grid state counts 30 000 boolean variables[2].

We have at our disposal approximately $N = 600,000$ power grids snapshots, representing more than 820 GB of compressed data for grid snapshots solely.

One pitfall of these big data is the lack of annotations. Changes in grid topology are recorded: we know precisely by observations what substations have changed between two timestamps. But we cannot know what caused such changes. Are these changes routine maneuvers carried out to check whether components are properly working? Are these due to maintenance operations? Or are these preventive or curative remedial actions taken by operators to protect the grid?

We are interested in the latter category, but they count for less than 10% of the total number of actions[3]. This is not the only problem of this dataset. Sometimes, to perform a single curative actions operators must change the topology of more than one substations; in our dataset, these changes will not be linked: we will observe the two changes, but we don't know they are made for the same purpose. On the contrary, sometimes changes the configuration of substation can take more than 5 minutes[4] and a single change for an operator will, in fact, be split on multiple powergrid snapshots. Finally, the lack of annotations in the data implies an even more challenging task:

[2]To be precise, the data we are using doesn't have this precision. Only the nodal topology is available, meaning that we don't have the state of each breaker, but solely the. The powergrid is rather described as a graph with powerlines as edges of this graph, and buses as its nodes. To the node of this graph, are connected productions, loads etc.

[3]This ratio comes from expert knowledge. Most of the operator's maneuvers are in fact to ensure that all breakers can be fully maneuvered or changing the topology surrounding a maintenance operation.

[4]Recall that 5 minutes is the rate at which the data are stored.

disentangle what causes a change, and what are its consequences. For example, imagine that a line falls due to a windstorm, then the operator changes (rapidly) the topology of one substation and disconnects another line. In this (extreme) scenario, we would just see all three actions happening at once, without knowing that they are inter-related.

4.1.2 Maintenance operations

In this subsection, we explain how we could use the raw powergrid French snapshots to check whether or not the developed methodology is suitable for real dataset. First results on experiments are carried out in Chapter 7.

Dividing the grid into smaller subgrid First, to reduce the complexity of the dataset at our disposal, we decided to study independently 2 parts of the French powergrid. We focus on the Extra High Voltage (all equipment operated at more than 200kV) of two regions of France. We choose two different areas to make sure our model was able to generalize to more than part of the power grid. The "Toulouse" area, which is located at the south west of France, and the "Marseilles" area located at the southeast. We chose this two regions mainly because the Apogee project, in which this Ph.D. took place, involves operators from these two regions. This was easier to refer to them than to operators coming from other parts of France.

The “Toulouse" area counts 246 consumptions, 122 productions and 387 lines. It is divided in 192 substations often split in a variable number of buses. The “Marseilles" area is composed of 377 loads, 75 productions and 260 transmission lines. It counts 140 substations for the part we studied.

Studying the impact of planned line disconnections While operating a powergrid, operators take various decisions. Some regular maneuvers are performed each day at approximately the same time, to adapt the topology of the powergrid to the variations of the demand. Operators know when these actions are performed but it is really difficult to recover such decisions in the database available as we only observe actions without knowing their purpose. However, we know these actions are performed at approximately the same time when the power grid faced approximately the same load. A learning machine, provided with these information as input should be able to predict the flows accurately. This is supported by results in the chapter 6. This fact explains why the number of buses in the real dataset is changing a lot. To face the peak total demand and avoid overheating often operator must decide to connect the pieces of equipment in some substations to different buses and in others to merge different buses into one.

Other types of actions are the one performed after maintenance operation. When a line is disconnected for maintenance operations, operators will adjust the topology of the powergrid surrounding the powerline disconnected. This happens extremely rarely, and lost of care must be taken by modeling these phenomena. In fact, as always when dealing with the historical dataset that we used, we know that this or this powerline has been disconnected from the powergrid, but we are not able to extract easily what are the consequences (actions performed by the operators) of this planned disconnection. We hope to build a neural network that will be robust to this phenomenon, *e.g.* an architecture that given a powerline disconnected for maintenance, is able to accurately predict the flows, even when given only the (partial) information about the real topology of the grid.

This problem is however of high interest for RTE. Indeed, RTE needs to address the security of the power grid even with given partial information about the topology, on forecasted grid state. For example, suppose a power line must be disconnected tomorrow to perform some maintenance operations. Will the powergrid remain safe after the removal? Today, the only way to answer this question is to build a full grid forecast, which requires to forecast each injection of the powergrid, then to have a forecast of the full topology, to eventually perform some security assessment based on this forecasted grid state. With a statistical method that does not take into account the full description of the topology, it might be easier and cheaper to assess this day-ahead security. In the last part of Chapter 6 page 77, we show that the guided dropout model performs better than the baselines on this problem.

4.1.3 Summary of the use of French Dataset

In this section, we motivated why we used only part of the French powergrid to study our models. We also explained why we gave partial information about the topology as an encoding of $\boldsymbol{\tau}$ (different topologies can be encoded by the same $\boldsymbol{\tau}$). A summary of these dataset can be found in table 4.2.

4.1.4 Limitations in using a real dataset

This approach, however, is not perfect, and even extracting the right flows and injections from our dataset can be a tedious task (the same powerline see its name changed for example). The quality of data is also varying in time. Historical data comes from

Table 4.2 **Summary of the composition of the French dataset used in this work**. We used 2 distinct part of the French dataset for 2 periods of time with a different number of grid states in the training set and different part of the grid. Two different regions are used to test the methods in different conditions and ensure the results are not specific to one type of grid.

	Toulouse Area	Marseilles Area
Number of injections $\boldsymbol{x}$	246 loads 122 productions	377 loads 75 productions
Number of powerlines $\boldsymbol{y}$	387	260
Number of powerlines $\boldsymbol{\tau}$	387	260
Beginning	January 1st 2012	January 1st 2017
End	May 31st 2017	February 28st 2018
Training Set	80% of the dataset*	10 000 grid states*
Validation Set	10% of the dataset*	1 000 grid states*
Test Set	10% of the dataset*	1 000 grid states*
Super Test Set	All the powergrid states from June 1st 2017 to July 31st 2017	1 000 grid states* from March 1st 2018 to March 31st 2018

* Sampling has been done uniformly at random. If a grid state is selected in one dataset, it is not in the others. A grid state is chosen to be at most in the Training set, the Validation set **or** the Test set.

sensors placed at various location on the grid, event though RTE dispose of a large number of sensors, we show an increasing quality of the data stored over time[5].

We have the guarantee that the flows can be computed (by a physical solver) if we know the injection and the topology. But, because our dataset is not labeled, and because information is not available at the time the decision must be made, only partial information about the topology can be given to a learning machine. In this setting, learning with this dataset is a really challenging task. This task is made even more challenging by the fact that multiple operators are operating the grid. One operator could prefer action π_1 compared to action π_2 while another operator could chose π_2 compared to π_1.

That is why we decided to first demonstrate the performance of our models on simulated datasets. And then to assess their fitness to handle historical dataset. In the next subsection, we explain the method we used to generate some synthetic yet realistic dataset.

[5]This quality can be assessed by tools developed by RTE, for example, if, after removing the noise in the dataset, a load flow solver, such as Hades2 is not able to find a solution that satisfies the physical laws, this may indicate data of lower quality.

4.2 Generation of synthetic data, experiments on a synthetic power grid

In the previous section, we motivated how we can use the French powergrid snapshots to test our models. We also developed why it is a challenging task. In this section, we present how we generated a synthetic dataset. The advantage of a synthetic dataset is that we place ourselves in a controlled experiment, where we know everything. Especially, we can decide how the injection and the topology are varying, which, for example, allows us to benchmark our models with the DC approximation.

In this section, more precisely, we detail how we sample the injections $\boldsymbol{x}$. The topology $\boldsymbol{\tau}$ are sampled differently according to the problem studied (see Chapter 6 for more detail). And, the flows $\boldsymbol{y}$ are computed using the powerflow simulator Hades2.

We used the library [56] which provides power grid states snapshots for various grids, very often used in the literature. For each snapshot, a vast quantity of data is available, with a precise description of the **topology** (*i.e.* how the powerlines, productions and loads are interconnected with each other). The simulator also provides the physical properties of the power lines (their resistance and reactance), and of the productions (the maximum power it can produce for example). Finally, such snapshots also contain initial values for loads and sometimes for productions too. We will suppose this is the repartition of loads and productions in a "standard situation", and denote it $\boldsymbol{x} = (p_1, p_2, \dots, p_{\text{nb prod}}, c_1, c_2, \dots, c_{\text{nb load}})$. A "standard situation" being a grid state with a non-extraordinary event: the total demand is approximately the average total demand that would be observed throughout the year, there is no storm, nor heat wave, the grid is not particularly stressed by any external factor.

Care has been taken to generate injections as realistically as possible. We used our knowledge of the French Power Grid to ensure that:

1. the statistical distribution of the total demand (sum of all individual loads) is the same as the one observed for the French power grid;

2. the statistical distribution of each individual active load is the same as the one coming from the real French power system;

3. the ratio reactive value over active value is the same as the one observed in the real system for each load

4. we emulated the maintenance of the production by disconnecting them with different probability: one for winter and one for summer to mimic seasonalities in maintenance requirement or economic offer and demand adequacy[6].

To achieve such results, we split the sampling of the injections into 3 parts, each one making a specific paragraph of this manuscript:

1. Sampling active value of each load.
2. Sampling reactive value of each load.
3. Sampling active value for each production.

4.2.1 Sampling active loads

In this subsection, the active value of a load j will be called ${c^{(p)}}_j$, p being a common way to denote active value in the power system community. We choose to represent the stochasticity of the active loads with:

$${c^{(p)}}_j = \rho_j {c^{\text{ref}}}_j \tag{4.1}$$

where ρ_j is a random variable that follows a distribution that requires: to take into account spatiotemporal behavior in the observed loads of the real system[7]

To ensure that the distribution of the total demand is the same as the one observed in the French power grid, we decided to separate ρ_j into two components:

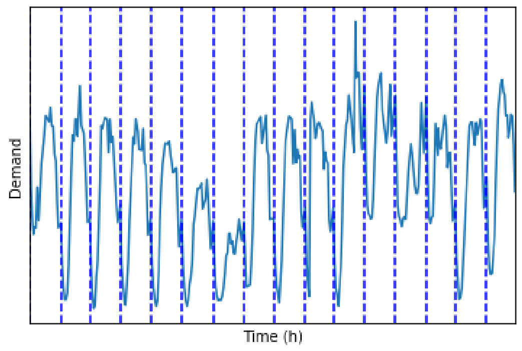
$$\rho_j = \rho^{\text{corr}} \rho_j^{\text{uncorr}} \tag{4.2}$$

where ρ^{corr} is the "correlated noise", representing the spatio temporal correlation of the loads: it is the same for all loads in the system. This correlated noise is drawn according to the French total load of 2012 that can be downloaded at https://www.rte-france.com/fr/eco2mix/eco2mix-telechargement. This parameter ρ^{corr} represents for example the time of the year (demand is on average higher in winter than in summer for example) and the time of the day (the peak demand is often at noon or at 7pm). On the contrary, ρ_j^{uncorr} represents the local variation around the tendency represented by ρ^{corr}. It is different for each load, and is drawn according to a log normal distribution, with mean 1, and variance as a free parameter. In most of our experiments,

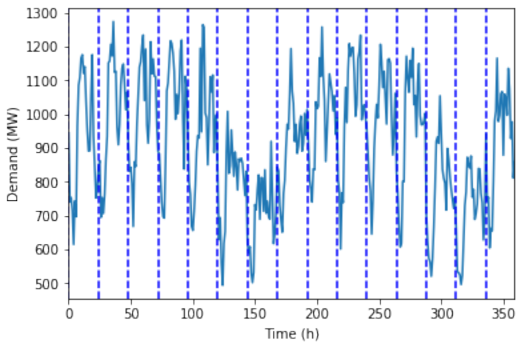
[6]some productions units are too expensive, and are disconnected in summer when the consumption is usually lower

[7]For example, people usually wake up approximately at the same time (temporal phenomenon). Some spatial correlations are also observed for geographically close loads due to weather, for example, two neighbors often have their heaters turned on at the same time if it freezes outside.

we used a variance such that 95% of the time, $\rho_j^{\text{uncorr}} \in [0.90, 1.1]$. In this setting, average $\left(\rho_j^{\text{uncorr}}\right) = 1$ and standar-deviation $\left(\rho_j^{\text{uncorr}}\right) = 0.05$.



(a) Representation of a load curve for 2 weeks for a **real data**. Vertical dotted lines represent the start of a new day



(b) Representation of a load curve for 2 weeks for a **generated data**. Vertical dotted lines represent the start of a new day

Fig. 4.1 Comparison of real data (left) and generated data (right), for a single active load.

4.2.2 Sampling reactive loads

At this point, for each load j, the active demand $c^{(p)}{}_j$ has already been sampled. In this subsection, we expose how to compute the reactive demand based on this. We denote by $c^{(q)}{}_j$ the reactive load value of load j. In the power system community, the assumption:

$$c^{(q)}{}_j = \tan(\phi).c^{(p)}{}_j \tag{4.3}$$

We used snapshots of the French powergrid, to compute the ratio $\frac{c^{(q)}}{c^{(p)}}$. We stored all the values computed, and used them to draw a $\tan(\phi)_j$ for each individual load according to their distribution in the real power system denoted $\mathscr{D}(\tan(\phi))$. Finally, the reactive load is obtained through:

$$c^{(q)}{}_j = \tan(\phi)_j.c^{(p)}{}_j, \text{ where } \tan(\phi)_j \sim \mathscr{D}(\tan(\phi)) \tag{4.4}$$

This is a rough simplification of the reality. For the true system, this distribution $\mathscr{D}(\tan(\phi))$ depends on many factors, for example the time of the year, the outside temperature[8], the presence / absence of industrial firm connected to a particular loads or the underground cables in the distribution network for example. We think that by sampling independently with a fixed empirical distribution $\mathscr{D}(\tan(\phi))$, we don't loose too much generality.

[8]Heater and air conditioner don't have the same properties with respect to the reactive value

4.2.3 Sampling productions

Productions are often disconnected for maintenance or economical reasons. We want to mimic this in our sampling strategy. That is why, for each production, we sample first if the production was disconnected or not with a Bernoulli law that has different parameters according to the time of the year. This corresponds to having: 10% of productions disconnected in winter, and 20% in summer. With these two parameters, we want to take into account two factors: productions are sometimes disconnected due to external causes (both in summer and in winter) and more productions are disconnected in summer, because traditionally the demand is lower, and producers favor this period to operate maintenance.

The voltages for the production nodes are set to their nominal values. Once the voltages have been set, the reactive power that needs to be injected by the production is computed by the **power flow** solver **Hades2**.

One of the cause leading to a divergence of a load-flow solver is an imbalance between total production and total loads. In theory, a quasi stationary power system satisfies the equation:

$$\sum_{\text{production } j} {p^{(p)}}_j = \sum_{\text{load } k} {c^{(p)}}_k + \text{ losses} \tag{4.5}$$

A solver will adapt if this equation is not perfectly satisfied by the input data, but a too high mismatch will lead to a divergence. In fact we observe on the French power grid that the total production is approximately 1.02 times the total demand[9]. We mimic this phenomenon in our simulations and at the same time, we make sure not to have a deterministic behavior for the productions. So we decided to first sample the values of each production to target the total demand increased by 2% via:

$${p^{(p)}}'_j = {p^{\text{ref}}}_j . u_j . 1.02 \frac{\sum_{\text{load } k} {c^{(p)}}_k}{\sum_{\text{prod } k} {p^{\text{ref}}}_k} \tag{4.6}$$

where $u_j \sim \mathscr{U}\,[0.8, 1.2]$, the uniform distribution between 0.8 and 1.2.

Once this has been done, we add an extra step to scale the production to ensure the relation $\sum_{\text{production } j} {p^{(p)}}_j = 1.02 \sum_{\text{load } k} {c^{(p)}}_k$ holds perfectly:

$${p^{(p)}}_j = {p^{(p)}}'_j . 1.02 * \frac{\sum_{\text{load } k} {c^{(p)}}_k}{\sum_{\text{prod } k} {p^{(p)}}'_k} \tag{4.7}$$

[9]These 2% corresponds approximately to the losses in the power grid, mostly due to Joule's effect.

4.2.4 Discussion

This sampling is, of course, a simplification of the reality. Among all the simplifications that were made, a few of them will be discussed here.

First, we don't take into account load types. In reality, behind a load, there are various different behaviors. Industrial loads will not behave the same way as residential ones: industrial loads will have a smoother pattern, whereas residential loads are heavily impacted by weather conditions (people switching on a heater or air conditioning when it is colder) or on the hour of the day (people leave for work on weekday for example). We don't take these phenomena into account. We believe this has not a strong impact on the evaluation of the performances of our models: our simulation scheme tends to simulate a wider range for every load.

Second, we also neglect the different types of productions. For example, a production injection power from renewable energy sources will have a behavior more stochastic (and more difficult to forecast), with higher variations than a nuclear power plant. This has a great impact on the grid operational processes since renewable energy sources are by nature intermittent. Our model, however, allows the prediction of flows given some injections. We believe the estimation quality of this model to be properly evaluated even if the lack of specific renewable energy sources in our dataset.

Lastly, all the loads will have the same pattern. Each individual load follows the historical French power distributions, the multivariate distribution of all the loads does not follow a realistic distribution: for example the correlation between load 1 and load 2 will have exactly the same distribution as the correlation between load 2 and load 3, and depends only on the individual noise injection when sampling active loads (denoted by ρ^{uncorr} in equation 4.1). We believe the impact of this hypothesis to be small on our simulation results for one main reason: a broader space is covered by our simulations scheme if a sufficiently high value of ρ^{uncorr} is set; the sampling strategy we adopted includes the real distribution.

4.3 French grid state snapshots: disentangling grid events to identify operator's actions and their purpose

In chapter 7, we reported how we could use the unlabeled dataset at our disposal to test the performance of our algorithms. In this section, we expose a method that allows to partially label the data at our disposal. This method is a first step that will be used in the work pursue by others Ph.D. student taking place at RTE.

This method provides a dataset $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{\tau}^*)$ whereas in most part of this manuscript $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$ denotes a powergrid state and $\boldsymbol{\tau}^*$ is an action taken by operators on powergrid described by injections $\boldsymbol{x}$, topology $\boldsymbol{\tau}$ and flows $\boldsymbol{y}$ to solve a possible security issue. This will be used as a first step to perform “imitation learning” for another Ph.D.student focusing on reinforcement learning for powergrid operations.

This methodology is also the start of a more ambitious work pursued by RTE. In this work, the objective is to allow operators to interact with machine learning techniques to relabel the actions in the powergrid snapshots. This is an ongoing work carried out by another Ph.D. student.

This section will be divided as followed. First, we expose the method developed to partially labeled the dataset. Then we report some results that were obtained by the method. Finally, we expose the limitation and discuss the further use of this method as well as the generated dataset.

The material exposed in this section comes from the published work: Donnot, B. and et al. (2017). Introducing machine learning for power system operation support. In *IREP Symposium*, Espinho, Portugal

4.3.1 Labeling the dataset

In the previous sections, we explain some of the problems we have with working with real data. We modeled in the chapter 2 how today’s operators act on the power grid, and why they prefer, in most cases, use topological actions *i.e.* reconfiguring line inter-connections, rather than cutting productions or consumptions. More precisely, in this thesis, when we refer to modifications in network topology, we mean re-configuring the way in which lines, transformers, productions and loads are connected in sub-stations. Using years of historical data collected by the French Transmission Service Operator (TSO) “Réseau de Transport d’Electricité" (RTE), we developed novel machine learning techniques to extract from this dataset, actions that could have been performed to ensure the security of the power grid.

Today, a grid is considered to be operated in “security" (i.e. in a secure state) if it is inside a zone of “constraints", which includes that power flowing in every line does not exceed given limits. The operators must avoid ever getting in a critical situation, which may lead to a cascade of failures (circuit breakers opening lines automatically to protect equipment, thus putting more and more load on fewer and fewer lines), ultimately leading to a blackout. To that end, it is standard practice to operate the grid in real time with the so-called “N-1 criterion”: this is a preventive measure requiring that at all times the network would remain in a safe state even if one component (productions, lines, transformers, etc.) would be disconnected. The idea of this labeling method is to simulate the counterfactual powergrids: “what if no topological actions had been performed”. This allows us to evaluate the grid with a "security function", namely a function of a grid state evaluating whether it is secure or not by returning a list of security criteria not met. This is performed by algorithm 4. In this algorithm, g_t denotes the powergrid at time t and $\mathbb{S}$ is a function that implements a security criteria, such as the "N-1": given a power grid states, it returns a list of possible security violations.

```
Input: $\{g_t\}_{t_{\min} \leq t \leq t_{\max}}$, $h_{\max}$, $\mathbb{S}$
Output: $\{(t,\ h,\ s,\ \tilde{g}_{t,h})\}$
     Initialisation :
 1:  res ← {}
     Main loop :
 2:  for $t \in [t_{\min}, t_{\max}]$ do
 3:     for $h \in [0, h_{\max}]$ do
 4:        create grid $\tilde{g}_{t,h}$ with the same injections than $g_{t+h}$ and the same topology
           than $g_t$
 5:        $S = \mathbb{S}(\tilde{g}_{t,h})$
 6:        if ($S \neq \emptyset$) then
 7:           for $s \in S$ do
 8:              res.append($(t,\ h,\ s,\ \tilde{g}_{t,h})$)
 9:           end for
10:        end if
11:     end for
12:  end for
13:  return res
```

Algorithm 4: Algorithm for finding unsafe networks. Observed grid states are denoted by g_t. Counterfactual grid states are denoted by $\tilde{g}_{t,h}$.

The output of Algorithm 4 is a list of security criteria not met in a stressed grid (the "counterfactual grids $\tilde{g}_{t,h}$" in which we suppress topological actions happening between t and $t+h$ from the historical dataset). At this stage, we have possibly really complex topological changes that can cure a power grid. In practice only approximately 10% of the actions performed are made for security issues (as explained in the previous subsection). So we need a way to extract the relevant actions, among these performed actions between t and $t+h$. This is done using a brute force approach, implemented in Algorithm 5. The idea is to test all combinations of these unary actions and store the results if it solves the problem.

More formally, we suppose in the historical dataset g_{t+h} is secure. So if $\tilde{g}_{t,h}$ is not, this means that at least one topological action performed between t and $t+h$ can set the powergrid back into its security domain. This property is guaranteed by our methodology: by definition, $\tilde{g}_{t,h}$ is the powergrid that has the same injections than g_{t+h} and the same topology as g_t. So the only difference between $\tilde{g}_{t,h}$ and g_{t+h} is the topology. Thus, if $\tilde{g}_{t,h}$ is not secure, but g_{t+h} is, this means a topological action can set back $\tilde{g}_{t,h}$ into a secure regime. The algorithm to extract all the topological actions that can cure the powergrid $\tilde{g}_{t,h}$ is presented in Algorithm 5.

```
**Input:** $\{g_t\}_{t_{\min} \leq t \leq t_{\max}}$, $\{(t,\ h,\ s,\ \tilde{g}_{t,h})\}$, $\mathbb{S}$
**Output:** $\{(s, \tau, \tilde{g})\}$
     Initialisation :
 1:  res $\leftarrow \{\}$
     Main loop :
 2:  for $t, h, s, \tilde{g} \in \{(t,\ h,\ s,\ \tilde{g}_{t,h})\}$ do
 3:      Compute the topological changes between $g_t$ and $g_{t+h}$, assign it to $\Gamma$
 4:      for $\tau \in subset(\Gamma)$ do
 5:          if not $s \in \mathbb{S}(\tilde{g} \odot \tau)$ then
 6:              res.append$((s, \tilde{g}, \tau))$
 7:          end if
 8:      end for
 9:  end for
10:  return res
```

Algorithm 5: Algorithm for extracting minimal remedial actions.

4.3.2 Results

We apply Algorithm 4 and Algorithm 5 to part of the French powergrid. In this work, we developed an API that allows us to modify efficiently, using a python script, the topology of the power grid, or changing the amount of power produced or consumed by **injections**. This has been built on top of RTE proprietary software Hades2, and it is also possible to launch a power flow computation and retrieve the results in approximately 300ms on the modified powergrid. We tested these algorithms on real data coming from January 1$^{\text{st}}$ 2012 and June, 30$^{\text{st}}$ 2012 (6 months, $N = 45\ 393$ grid states). For each grid state g_t we used expert knowledge from operators to determine the set of counterfactual grid states $\hat{g}_{t,h}$ that will be simulated. We chose to simulate $H = 14$ of these counterfactual grid states per time stamps (for h varying in $5, 10, 15, 30$ mins, then $1; 1,5; 2; 3; 4; 5$ hours, and 23h, 23h30 and 23h45). Running Algorithm 4 alone required the computation of $N \times H = 635\ 502$ counterfactual grids $\hat{g}_{t,h}$.

Assessing the "N-1 security" of all these grid states would require $n > 10\ 000$ load-flow computations for each of these 635 502 grids states[10], meaning approximately $N \times H \times n \approx 640\ 000\ 000$ load flows. If we suppose we can perform 3 power flow computations per sec, this is approximately 2 500 days of sequential processing. That is too much for this proof of concept. That's why we decided to assess only the security

[10]As explain in Chapter 2 assessing this security requires to run 1 load flow per powerline. The French powergrid counts 10 000 powerlines.

based on the "N security criteria"[11] that requires only the computation of 1 power flow per grid state $\hat{g}_{t,h}$, so solely 635 502 load flow computations.

Among these 635 502 grid states, 63 215(10%) are not secure according to our security criteria. These 63 215 grid states have been stored, and are accessible to other people at RTE who will use it. In particular, the Ph.D. student working on applying reinforcement learning methods to solve power system related issues.

On average there are approximately 3 topological changes on the whole French powergrid between two consecutive powergrid g_t and $g_{t+5\text{min}}$. Running Algorithm 5 for all the subset of changes would require also a lot of computations power: for example if 15 actions are performed between t and $t+2$ hours, then there would be $2^{15} = 32\ 768$ possible combinations of such actions. To simulate all these combinations would take a lot of time, and be mostly wasteful in computer resources, knowing that only 10% of actions are relevant. For this reason, in Algorithm 5 we decided to simulate only unary actions, one at a time: if we take back the example where 15 different changes occur, this lead to simulate 15 different grid state. This restriction is also motivated by experts knowledge, it is quite rare that experts need to perform 2 actions in order to solve problems happening in real time. This explains why, among the 63 215 $\hat{g}_{t,h}$ found not secure, applying Algorithm 5 allows us to find 44 041(75%) unary actions that solved the issue on this grid. To obtain such results, more than 2 000 000 load-flows were performed.

This is an encouraging result, but more needs to be done. These actions are not yet validated by expert operators, some might be real actions, but others can be caused to imperfection in the solving methods. Also, the amount of computations required is really important. We made some assumptions to reduce it, but a more generic approach, for example by approximating load-flows with neural networks (*i.e.* a machine learning emulator), can be carried out to gain computational speed. Another solution could also be to supplement the slow (but accurate) simulator with a fast machine learning emulator, the latter pruning first the most trivial cases and the former carrying out simulations to full precision for a handful of promising actions. The main methodology we present in this work will be used in this settings when it will be mature enough to handle bigger powergrids of the size of the French one.

4.3.3 Discussion and limits

In this section, we described the historical dataset and explained why it is difficult to use it (due to unlabeled data). We proposed a method to extract actions in this

[11]Recall that the "N" security criterion consists in checking that the current flow in all the powerlines is below their thermal limit. So only 1 power flow per grid state is necessary to assess the security of the powergrid based on this criteria

historical dataset that could have been taken by operators in order to maintain a power grid in “a safe state", using machine learning techniques. Finally, we reported some results about the application of this technique to real French dataset.

This re-labeling is a non-trivial problem because (1) many actions performed on the grid are not protective actions (they may include maintenance actions and miscellaneous maneuvers); (2) there is no centralized and uniform record of why given actions are performed; (3) the consequences of not performing given actions are not observed, hence it is difficult to assess how effective given protective actions may be.

We devised and implemented an algorithm based on the causal concept of counterfactuals, which allows us to identify actions that have had beneficial effects (or more precisely, without which the network would have incurred adverse effects). But these methods have some drawbacks. The main one is that it can only be used to identify topological curative actions. It also requires a lot of computational resources even in using a simplified version of the security assessment protocol. And finally even though we have the guarantee that these algorithms produce actions that effectively set back the powergrid in a safe state, we don’t know if these actions are relevant or not: some of the actions identified by this method might be due to limitations of the power flow solver for example.

This task of labeling is highly challenging and will be pursued by another Ph.D. student that will start her work in the coming weeks. All the work pursue in this section has been carefully stored on multiple servers, and can be reused by yet another Ph.D. student as part of his thesis on “reinforcement learning”. This database can be used for example as the first step to train an artificial agent by imitating the operators’ decisions for example

Chapter 5

The “Guided Dropout” methodology

In this chapter, we present the main methodology developed in this manuscript, called "Guided Dropout". This methodology contains both a dedicated neural network **architecture** and operations that give plasticity to it.

The guided dropout methodology addresses the problem of rapidly assessing the values of the flows $\boldsymbol{y}$ on all **transmission lines** of the **power grid**, knowing the **injections** values $\boldsymbol{x}$ and some actions $\boldsymbol{\tau}$ on this grid. This methodology is indeed particularly suited for problems with conditional variables, such as the grid topology, which modulates the system’s response from a reference condition or state. Guided dropout is also a way to deal with the problem having both discrete and continuous variables in input (see Section 5.2). Our main application for this architecture is the predictions of flows in a powergrid. But we believe our methodology is more generic, and able to tackle broader tasks in multiple applications domains, some of such applications are discussed in Section 5.4.2.

In Section 5.1, we present the inspiration of the proposed algorithm, as well as the vocabulary used in the rest of the chapter. In Section 5.2, we expose the algorithm in detail. Finally, in Section 5.3, we derive mathematical properties of this algorithm.

The material in this section is based on the work:
Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018d). Fast power system security analysis with guided dropout
and
Donnot, B., Guyon, I., Liu, Z., Schoenauer, M., Marot, A., and Panciatici, P. (2018a). Latent Surgical Interventions in Residual Neural Networks. working paper or preprint

5.1 Introduction

In this section, we introduce the motivation of the guided dropout algorithm. We first give the specifications of the method and then detail some related work in the literature.

5.1.1 Specifications

In the problem we are tackling, we are dealing with a specific dataset, where data can be written as $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$. In our main application, $\boldsymbol{x}$ will be the **injection** and $\boldsymbol{y}$ the powerflows. Both will be continuous variables. Our main contribution is to develop and analyze a new way of encoding the structural variables $\boldsymbol{\tau}$ that are discrete in our application domain (**power grid** topology: buses reconfiguration or power lines disconnection in our main applications).

To be usable by **Transmission System Operator (TSO)**s, a neural network must have some specific properties.

- It should be able to learn in a reasonable amount of time (a few days at most) how to compute flows from injections and topology given historical data. It needs to be generic enough so that it adapts to **multiple topology $\boldsymbol{\tau}$**.
- As explained in Chapter 4, we cannot know exactly what causes some changes in the historical dataset. So the model should be able to deal with **partial topological informations**. For example, it could still be able to predict the flows even if provided only with the information "this powerline is taken out for maintenance reasons" without the actual knowledge of what operators have changed following this maintenance operation.
- It should learn with **little data available**. The withdrawal of a powerline does not happen very often, neither does an unplanned line disconnection. That’s why the learning machine must be able to adapt its knowledge to scenarios that are rarely seen.
- And most importantly, it should be able to generalize to **unseen scenario** as simulators enable. For example, one could want to assess what happens after two lines are disconnected at the same time, even though this specific configuration has never been observed in the **training set**.

5.1.2 Background

Our new algorithm was inspired by the computer vision work of Srivastava et al. [46], who introduced a technique called "**dropout**" to regularize convolutional neural networks (*i.e.* make them more resistant to overfitting) and improve the generalization error. Previously, people where using l_1 and/or l_2 penalizations to improve generalization (see elastic net, Equation 3.5 page 32). This was sometimes called "weight decay" [29] in the neural network literature. The new idea of Srivastava et al. was to regularize by *randomly killing hidden units* in the neural network while training. However, when making predictions, all units were reactivated, and their output was scaled: activation at test time was multiplied by the probability to be connected at training time.

Gal et al. showed in [17] that dropout can be interpreted as a joint learning of multiple models in one single architecture. This is illustrated in figure 5.1. The dropout is equivalent to training, in one single architecture, multiple different neural networks. Our novel algorithm, the "**guided dropout**" proceeds in a similar way and enjoys the same superposition properties. However, rather than *randomly killing hidden units i.e.* averaging multiple random neural networks for a regularization purpose, the Guided Dropout algorithm *activates units selectively* guided by the underlying **power grid** topology, as explained in more details in section 5.2. Although Guided Dropout is very similar to classical dropout in its mathematical formulation, it has a completely different aim and effects. Dropout randomly kills units and acts as a regularizer, guided dropout doesn't kill units randomly anymore and acts as a way to encode some properties inside the architecture of a neural network.

Our method also combines ideas from residual networks [21], and conditional computation (*e.g.* [4]). The idea behind the conditional computation is to adapt the architecture of a neural network depending on its inputs. Both the weights and the architecture are trained. This conditional computation has been adapted with success by [43] for automatic translation. More recently Minhui Zou et al. [58] showed that adding some units inside an already trained neural network is enough to bias its predictions. This work is based on the same idea: the **guided dropout** consists of adding units at the heart of a residual neural network.

Guided dropout can also be seen as a transfer learning approach [38]: we train a neural network to generalize across domains (each domain is represented by a structural vector $\boldsymbol{\tau}$). We demonstrate (experimentally and theoretically under some conditions) that our predictive models are indeed capable of transfer learning and exhibit a "**super generalization**" property that can be seen as a zero shot learning: The neural network generalizes to new domains (*i.e.* new $\boldsymbol{\tau}$) for which it was never

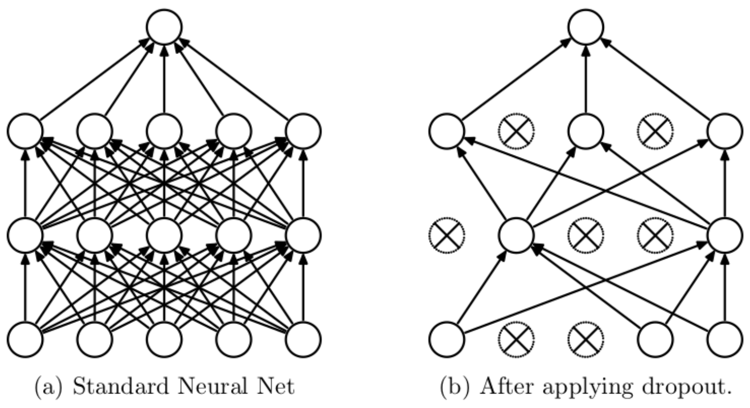


Fig. 5.1 Representation of 2 fully connected neural networks sharing the same **plain architecture** (the same pattern of connections: 5 inputs layers, 2 hidden layers each having 5 layers, and one output layers). On the left is a fully connected neural network, on the right is shown a possible network seen during training, where crossed unit have been dropped (activation forced to be 0). The image comes from [46] (Figure 1).

trained. This is made possible by encoding domains in the neural network architecture. A similar line of work has been pursued by other authors: in [27] for "one-shot learning", in [33] for "few shot learning", and in [45] for "zero shot learning", although with quite different approaches. To be fair, in our problem, the "zero shot" learning property comes from the combination of independent unary actions, which makes it fall in the "combinatorial generalization" domain as explained in [2] for example.

5.2 Description of the guided dropout methodology

In this section, we present the **guided dropout** methodology in a general context. The objective is to approximate a function $\boldsymbol{y} = S(\boldsymbol{x}; \boldsymbol{\tau})$ that maps input data $\boldsymbol{x}$ to output data $\boldsymbol{y}$. This system S is parameterized by a discrete structural vector $\boldsymbol{\tau}$, taking values in an intervention space.

In our application domain, the variable $\boldsymbol{x}$ represents **injection** (*i.e.* the power produced by **productions** and consumed in **loads**), $\boldsymbol{y}$ is the flows in all the power lines, and $\boldsymbol{\tau}$ represents the topology of the power grid[1]. In our application, the topology $\boldsymbol{\tau}$ can either represent a choice from the operators (*e.g.* when an operator chooses to modify how the powerlines are interconnected) or be external causes that impact the powergrid (*e.g.* a windstorm hits some tower and a powerline is disconnected). Both these applications are treated in this work. Most of the time we don’t distinguish if the structural vector $\boldsymbol{\tau}$ is modified by the operator, at the hands of RTE, or if it has external causes.

[1]In some applications, $\boldsymbol{\tau}$ will not represent the complete topology of the power grid.

The rest of this section will be organized as follows. First, we recall notations used. Then we develop the neural network architecture and finally, we give some intuitions on why this methodology achieves good results.

5.2.1 Notations and definitions

Let $\boldsymbol{x} \in \mathscr{X} \subset \mathbb{R}^p$ be the input vector of the system ($\dim \boldsymbol{x} = p =$ number of injections in the power system application), $\boldsymbol{y} \in \mathscr{Y} \subset \mathbb{R}^n$ the output vector ($\dim \boldsymbol{y} = n =$ number of lines in the power system application), and $\boldsymbol{\tau} \in \{0,1\}^\alpha$ the action vector ($\dim \boldsymbol{\tau} = \alpha$).

The action space is represented as a binary vector (denoted by $\boldsymbol{\tau}$ and called structural vector) without any particular encoding of this discrete topology $\boldsymbol{\tau}$. In particular, unary actions (*i.e.* single power line disconnection in the power system application) are not necessarily one-hot encoded and therefore α is not necessarily equal to the number υ of unary actions. If one-hot encoding of the topology $\boldsymbol{\tau}$ is used and unary action z_1 is encoded with $\boldsymbol{\tau}^{\{1\}} = (1,0,0,\dots)$ and unary action z_2 is encoded with $\boldsymbol{\tau}^{\{2\}} = (0,1,0,\dots)$, then double action $z_{1,2}$ is encoded with $\boldsymbol{\tau}^{\{1,2\}} = (1,1,0,\dots)$. In general, double action $z_{1,2}$ is encoded with $\boldsymbol{\tau}^{\{1,2\}} = \boldsymbol{\tau}^{\{1\}} \oplus \boldsymbol{\tau}^{\{2\}}$ (element-wise "or" operation). More generally, let $\boldsymbol{\tau}^{\mathscr{I}}$ be the overall structural vector in $\{0,1\}^\alpha$ which combines any unitary action such that $\boldsymbol{\tau}^{\mathscr{I}} = \bigoplus_{i \in \mathscr{I}} \boldsymbol{\tau}^{\{i\}}$. Here $\mathscr{I} \subset \{1,\dots,\upsilon\}$ where υ is the number of *u*nary actions. In the case of the one hot encoding we have then $\upsilon = \alpha$.

We could also have chosen to encode unary action z_1 is encoded with $\boldsymbol{\tau}^{\{1\}} = (1,1,0,\dots)$ and unary action z_2 with $\boldsymbol{\tau}^{\{2\}} = (0,0,1,0,\dots)$. For example. The combination of structural vector $\boldsymbol{\tau}$ would be performed as described in the previous paragraph. The major difference would be that in this case $\upsilon < \alpha$, as some of the action z would be encoded in two distinct dimension of $\boldsymbol{\tau}$.

For any encoding of actions, by convention, $\boldsymbol{\tau}^{\emptyset} = (0,0,\dots)$ will represent the absence of action, corresponding to the system in its reference topology.

Recall of other notations $\boldsymbol{y} = S(\boldsymbol{x};\boldsymbol{\tau})$ be the ground truth of the system's response to input $\boldsymbol{x}$ in structural vector $\boldsymbol{\tau}$. The ground truth here will be given by the physical simulator. In contrast, the approximation made by the neural network will be denoted $\hat{\boldsymbol{y}} = NN(\boldsymbol{x};\boldsymbol{\tau})$. For this work, and unless explicitly specified, hatted variables are always the results of a statistical model. On the contrary, the ground truth is denoted with non-hatted variable (*e.g.* $\boldsymbol{y}$).

Simple generalization vs. super generalization Similarly to other learning problems, for any fixed structural vector $\boldsymbol{\tau}$, training data pairs $\{\boldsymbol{x}, \boldsymbol{y}\}$ are drawn *i.i.d.* according to an unknown probability distribution[2].

We call **simple generalization** the capability of $\hat{\boldsymbol{y}} = NN(\boldsymbol{x}; \boldsymbol{\tau})$ to approximate $\boldsymbol{y} = S(\boldsymbol{x}; \boldsymbol{\tau})$ for test inputs $\boldsymbol{x}$ not pertaining to the training set. When **$\boldsymbol{\tau}$ values are drawn *i.i.d.* from training domain distribution $\mathscr{T}^{\mathbf{train}}$** that remains the same in training and test data (this covers in particular the case of a fixed $\boldsymbol{\tau}$), the term **simple generalization** is also used.

Conversely, if **values of $\boldsymbol{\tau}$ are not drawn similarly in training and test data**, *i.e.* $\boldsymbol{\tau}$ is drawn according to a certain distribution $\mathscr{T}^{\text{test}} \neq \mathscr{T}^{\text{train}}$ for the test data, we speak about **super generalization**. "Zero shot learning" is a particular case of super generalization when the domain $\boldsymbol{\tau}$ has not been seen at all during training. For our specific application, zero shot learning is important: new topological situations happen all the time when managing a power grid, and the neural network should not provide false results in these cases. Note in this work we are not interested in all possible types of super generalization. We focus on, possibly unseen, combinations of *seen* structural vectors $\boldsymbol{\tau}$.

5.2.2 Guided dropout intuitions

To illustrate the idea behind the guided dropout, we will use a system S that has 5 inputs units $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)$, 7 outputs units $\hat{\boldsymbol{y}} = (\hat{y}_1, \dots, \hat{y}_7)$ and 3 possible independent actions $\upsilon = 3$. We choose to illustrate the functioning of this algorithm on a one-hot encoding of actions, so we have $\dim(\boldsymbol{\tau}) = 3$.

We will learn this system with a **neural network** with 3 hidden layers each having 6 hidden units. This neural network will be called the **plain architecture**. It is represented in figure 5.2a.

The main idea of the proposed **guided dropout** methodology proposes to kill units depending on the structural vector $\boldsymbol{\tau}$. Hence the name "guided dropout". The *dropout* of units is *guided* by the structural vector $\boldsymbol{\tau}$, describing the topology in this setting.

This is illustrated in Figure 5.2. In Figure 5.2b, some units have been switched off in the second hidden layer. This is always the case: when no actions are performed, the neural network has fewer units. In practice then, when we choose to activate units when powerlines are disconnected, the terminology "guided drop in" would be more suited as in facts units are *added* when the corresponding unit of the structural vector is 1. We preferred, however, keeping the name "guided dropout".

[2]In our setting, $\boldsymbol{x}$ is drawn randomly, but $S(\boldsymbol{x}; \boldsymbol{\tau})$ is a deterministic function implementing Kirchhoff’s circuit laws. No noise term is involved in the calculation of $\boldsymbol{y}$ from $\boldsymbol{x}$.

In Figure 5.2c, action z_1 encoded by $\boldsymbol{\tau} = (1,0,0)$ is made, one more unit is activated. Which units are concerned by this action is chosen randomly once among the units that are in the plain architecture (Figure 5.2a) but not used in the architecture encoded for $\boldsymbol{\tau} = \tau^{\emptyset}$ (Figure 5.2b). This procedure is applied for all unary actions z_i, as can be seen in Figures 5.2c, 5.2d and 5.2e. In practice, each time a guided dropout network has been trained, this procedure has been applied. We didn't notice any particular impact on the results. We believe the random choice of masks has little impact on the final outcome of a guided dropout network.

The encoding of unary actions is the only free parameter of the guided dropout algorithm. Once set, all the actions that can be encoded by $\boldsymbol{\tau}$ can be represented by our neural network architecture by adding the units that are added component-wise. This is showed in Figure 5.2f where units activated for $\tau^{\{1\}} = 1$ and $\tau^{\{3\}} = 1$ are both activated.

The main idea of the **guided dropout** is to reflect a combination of actions by modifying the architecture of the neural network in a deterministic way. The super generalization properties of the guided dropout, *i.e.* its capacity to predict the real output $\boldsymbol{y}$ even when in unseen structure $\boldsymbol{\tau}$, comes from the combinations of unary actions in the system S: there is a relation between $S(\boldsymbol{x};(1,0,1))$ and $S(\boldsymbol{x};(1,0,0))$ and $S(\boldsymbol{x};(0,0,1))$.

The method we propose is more generic than the one described here. It can be modified in multiple fashions:

- One neuron was assigned per unary action. This can be adapted depending on the problem. For example, 2 units can be activated when $\tau_1 = 1$ (as opposed to Figure 5.2c where only 1 was). This will be the case when a non one hot encoding of the actions z is chosen.
- One layer is impacted by the guided dropout (second hidden layer in our example), but multiple such layers can be stacked at different parts of the neural network.
- We choose to activate units when components of $\boldsymbol{\tau}$ are 1, but this can be the opposite: *i.e.* deactivating units instead of activating them. This last is conflicting with the encoding of the action. For example, we could have chosen to encode the reference topology (*i.e.* the one where all lines are connected) by $\boldsymbol{\tau}^{\text{ref}} = [1,1,\dots,1]$, and to encode the action z_1 by $\boldsymbol{\tau}^{\{1\}} = [0,1,1,\dots,1]$. The impact of this change is not extensively studied in this manuscript. In early experiments, we found that using this encoding performed not as well as the proposed one for the power system application. In our domain, more data are available for the

reference topology. In this setting, it is harder for a neural network to predict with a different topology. Having more units to predict a more difficult problem may explain why the proposed method (adding units) performs better than this other methodology (removing units).

In any case, we note that at least one neuron per dimension of $\boldsymbol{\tau}$ for the layer of which the guided dropout is applied is required. Otherwise, some configurations of $\boldsymbol{\tau}$ will have the same encoding in the neural network, thus leading to the same predictions $\hat{y}$. If zero neuron is used in the structural vector $\boldsymbol{\tau}$ to encode an action, the information that this action is performed is not given to the neural network. This implies that $\dim(\boldsymbol{\tau}) = \alpha > \upsilon$, the number of actions.

5.2.3 Final Neural network architecture

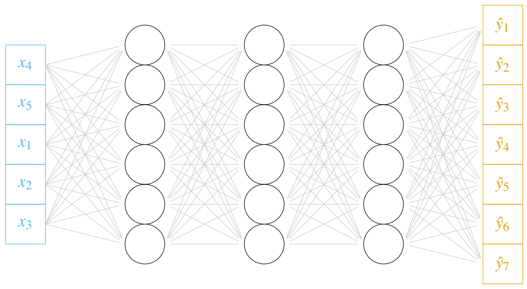
In the previous subsection, we introduced the concept behind guided dropout algorithm. In this section, we will propose a different view of this algorithm.

One of the problems of the previous methodology was that in the guided dropout layer, there are two different types of units: units that are sometimes present, sometimes absent (in dashed red in figure 5.3a), and units that are always present. This makes things more difficult for us to interpret. For example, in the reference architecture, all these units are disconnected, and there are only 3 units to propagate the information from the encoder E to the decoder layer D. In this modeling, the information on the guided dropout block is intermixed between information specific to the structural vector $\boldsymbol{\tau}$ (red dashed connections) and information that always propagates (gray connections). We decided to switch to a residual architecture to fix this issues. This new architecture is showed in Figure 5.3. Compared to the previous neural network architecture, the new one, based upon residual models reduces the number of free parameters. It is also easier to distinguish between the model in its reference structure $\boldsymbol{\tau} = [0, 0, \dots]$, and to isolate the impact of each unary action mathematically in this new architecture.

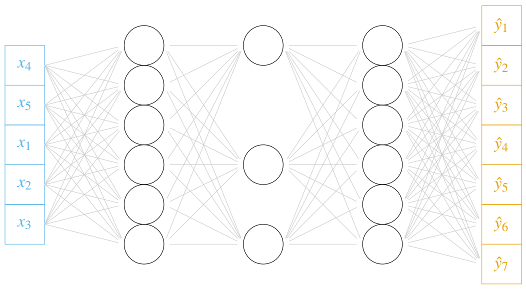
In this new architecture, let’s denote by $\boldsymbol{h_x}$ the latent representation of $\boldsymbol{x}$ after going through the encoder E and by $\boldsymbol{h_\tau}$ the transformation by the guided dropout block of $\boldsymbol{h_x}$ affected by structural vector $\boldsymbol{\tau}$. These number are shown on Figure 5.4. Formally, these numbers are defined as:

$$\boldsymbol{h_x} \overset{\text{def}}{=} \boldsymbol{E}(\boldsymbol{x}) \tag{5.1}$$

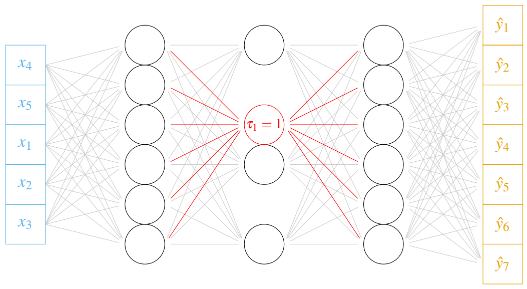
$$\begin{aligned} \boldsymbol{h_\tau} &\overset{\text{def}}{=} \boldsymbol{d}(\boldsymbol{e}(\boldsymbol{h_x}) \odot \boldsymbol{\tau}) \\ &= \boldsymbol{d}(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau}) \end{aligned} \tag{5.2}$$



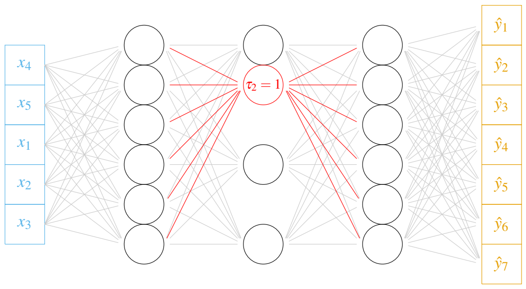
(a) **Plain architecture**, no units are disconnected



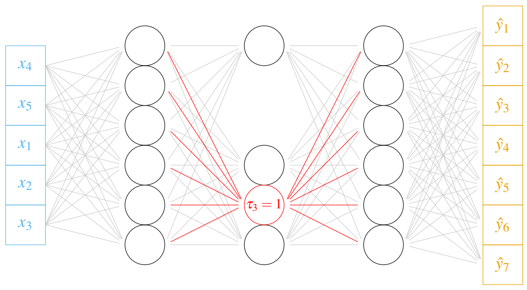
(b) Architecture for $\boldsymbol{\tau} = \boldsymbol{\tau}^{\emptyset}$. All



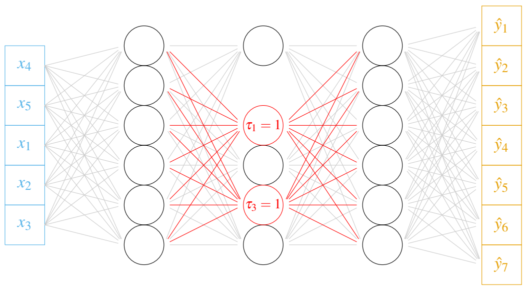
(c) Architecture for unary action encoded by $\boldsymbol{\tau} = (1,0,0)$



(d) Architecture for unary action encoded by $\boldsymbol{\tau} = (0,1,0)$

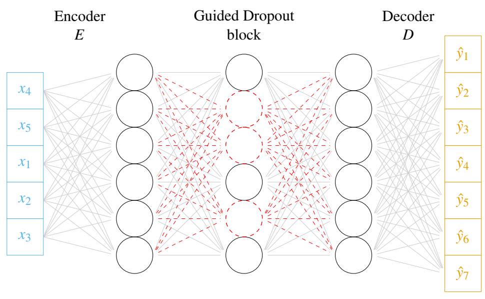


(e) Architecture for unary action encoded by $\boldsymbol{\tau} = (0,0,1)$

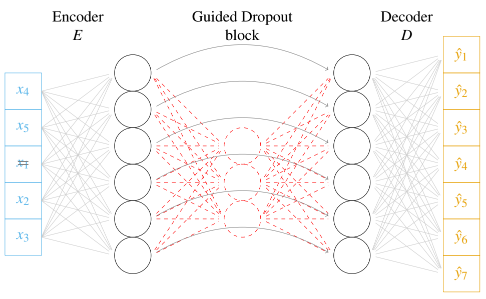


(f) Architecture for non unary actions encoded by $\boldsymbol{\tau} = (1,0,1)$

Fig. 5.2 **Schematic representation of the guided dropout methodology.** First choose a "**plain architecture**", *i.e.* the number of layer, and the number of units per layer, but also the non linearity etc. For this example, the plain architecture is composed of 3 hidden layers, of 6 units each. Then we chose which neurons will be activated / deactivated depending on the structural vector $\boldsymbol{\tau}$ that encodes the topology (showed here on Figures 5.2c, 5.2d and 5.2e. Then combinations of actions are performed by adding / removing units, as shown in Figure 5.2f.



(a) **Original** guided dropout neural network as described in Section 5.2.2. Dashed lines represent connections that can be activated / deactivated depending on the action vector $\boldsymbol{\tau}$.



(b) **ResNet** guided dropout neural architecture. Bended arrows represent direct connections.

Fig. 5.3 **Comparison of guided dropout architectures.** Original architecture (left) and ResNet architecture (right) for the guided dropout algorithm. Gray straight line represent regular connections, always present and not impacted by the structural vector $\boldsymbol{\tau}$. The red dashed line represent connections that can be activated and deactivated depending on the structural vector $\boldsymbol{\tau}$. Finally, on Figure 5.3b, black bended connections represent the "copy operator": the element on the left is copied to the right.

The proposed **guided dropout** (GD) model represented schematically in Figure 5.4 (top) is given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}) + \boldsymbol{d}(\underbrace{\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau}}_{\text{some units are masked depending on } \boldsymbol{\tau}})) \tag{5.3}$$

where $\boldsymbol{E}$ and $\boldsymbol{e}$ (encoders) and $\boldsymbol{D}$ and $\boldsymbol{d}$ (decoders) are all differentiable functions. The $\odot$ operation denotes the element-wise multiplication.

In the reference structure $\boldsymbol{\tau} = [0, 0, \dots]$ Suppose the system is in the reference topology $\boldsymbol{\tau}^{\emptyset}$, predictions are made according to $\hat{\boldsymbol{y}} = \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))$. Indeed, assuming that $d(\boldsymbol{0}) = \boldsymbol{0}$, for $\boldsymbol{\tau}^{\emptyset} = (0, 0, 0, \dots)$, we have $\boldsymbol{d}(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau}) = d(\boldsymbol{\tau}^{\emptyset}) = \boldsymbol{0}$, thus the guided dropout block lets the information flow directly, without modification. Only E and D are needed to perform a prediction in a "reference state" (encoded by $\boldsymbol{\tau} = \boldsymbol{\tau}^{\emptyset}$).

Outside the reference structure $\boldsymbol{\tau} \neq [0, 0, \dots]$ In that setting, the forward pass through this neural network is:

$$\begin{aligned} \hat{\boldsymbol{y}} &= \boldsymbol{D}(\boldsymbol{h_x} + \boldsymbol{d}(\boldsymbol{e}(\boldsymbol{h_x}) \odot \boldsymbol{\tau})) \\ \hat{\boldsymbol{y}} &= \boldsymbol{D}(\boldsymbol{h_x} + \boldsymbol{h_\tau}) \end{aligned} \tag{5.4}$$

On the contrary, if one of the components of $\boldsymbol{\tau}$ is non null ($\boldsymbol{\tau} \neq \tau^{\emptyset}$), *e.g.* if an action is performed, some components of $\boldsymbol{h_x} \odot \boldsymbol{\tau}$ will be non null. In this case $\boldsymbol{h_\tau}$ $(= \boldsymbol{E}(\boldsymbol{x}) \odot \boldsymbol{\tau})$ can be non zero because $\boldsymbol{\tau}$ will have at least a non null component. And as we can see in equation 5.4 (or in figure 5.4) , the results of this activation will be added to $h_{\boldsymbol{x}}$ before the information is processed by D. In this setting, $h_{\boldsymbol{\tau}}$ is exactly the modifications of $h_{\boldsymbol{x}}$ induced by $\boldsymbol{\tau}$ being non zero. This phenomenon will be further developed in the next subsection.

5.2.4 Interpretation and latent spaces

One possible interpretation of the guided dropout is the following. Input data $\boldsymbol{x}$, from a manifold $\mathscr{X}$, are first processed by an encoder E that transforms them into a vector $\boldsymbol{h_x}$. Then, if no action is performed, this vector $\boldsymbol{h_x}$ is decoded via D to make the predictions

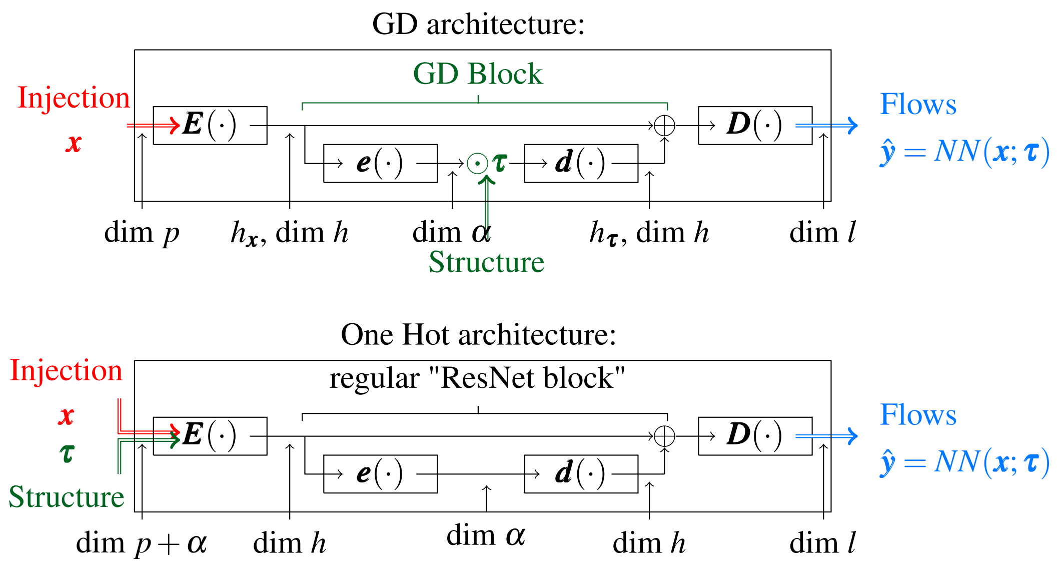


Fig. 5.4 **GD and baseline architecture**. The main novelty of the GD architecture is that interventions are introduced by performing an element-wise multiplication $\odot$ with a binary vector $\boldsymbol{\tau}$, which has the effect of selectively activating or deactivating units. The baseline architecture instead includes a standard residual network block [21]. Interventions are introduced as additional inputs to the first block. In both architectures, multiple ResNet or GD blocks may be stacked.

$\hat{\boldsymbol{y}} = \boldsymbol{D}(\boldsymbol{h_x})$. In this setting, $\boldsymbol{h_x}$ can be seen as the latent representation of the input $\boldsymbol{x}$, once encoded by the function E.

The effects of e and d are a bit different. e takes as input $\boldsymbol{h_x}$ and will encode it in $\boldsymbol{h_\tau}\left(\stackrel{\text{def}}{=}\boldsymbol{d}(\boldsymbol{e}(\boldsymbol{h_x})\odot\boldsymbol{\tau})\right)$. So their role is to encode the manifold $\mathscr{H}$ in which the data are mapped with encoder E, into a "action manifold" in which action $\boldsymbol{\tau}$ is applied (through the element wise multiplication). Then this actions space is decoded with d and the effect of action $\boldsymbol{\tau}$ is then translated into a shift of h_x by $\boldsymbol{h_\tau}$ into the manifold $\mathscr{H}$.

The guided dropout algorithm can then be interpreted as an encoder E which transforms inputs $\boldsymbol{x} \in \mathscr{X}$ into a subspace $\boldsymbol{h_x} \in \mathscr{H}$. In the standard configuration (defined by being encoded with $\boldsymbol{\tau} = \boldsymbol{\tau}^{\emptyset}$), this subspace is decoded to represent the flows $\boldsymbol{y}$ as can be showed in the figure 5.5.

A change in $\boldsymbol{\tau}$ will be propagated in this architecture by adding a perturbation that depends both on the inputs $\boldsymbol{x}$ via $\boldsymbol{h_x}$ and on the perturbation encoded by $\boldsymbol{\tau}$. So the (learned) latent representation $\boldsymbol{h} = \boldsymbol{h_x} + \boldsymbol{h_\tau}$ is, by construction, built such that additive perturbation in this latent space results in the proper output predictions.

A parallel between the guided dropout methodology and the **superposition theorem** can be made. This theorem can be used to compute some flows when a generator or a load is added in a circuit, for example, [3]. Adopting this point of view, the guided dropout will learn a representation $\mathscr{H}$ of the input manifold $\mathscr{X}$. On this manifold, the modification induced by $\boldsymbol{\tau}$ (denoted by $\boldsymbol{h_\tau}$) are added to the representation of the data $\boldsymbol{h_x}$, thus superposing modifications conditioned by $\boldsymbol{\tau}$ to the state $\boldsymbol{h_x}$.

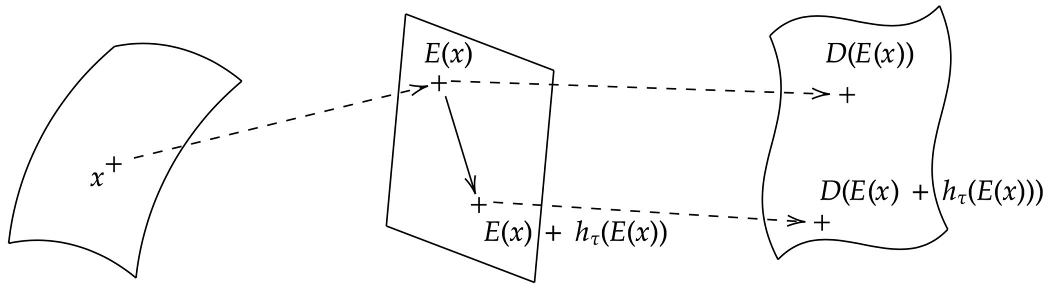


Fig. 5.5 **Schematic representation of the guided dropout algorithm, seen as an encoder / decoder.** On the left is showed the input data manifold $\mathscr{X}$. A point $\boldsymbol{x}$ of this manifold is transformed into $\boldsymbol{h_x}$. Then the action $\boldsymbol{\tau}$ will translate this value on the manifold. Finally, the output value will be computed by applying function D.

[3]We emphasize that this theorem is valid when the topology of the circuit doesn’t change, which is not our setting. This is only an analogy and we agree the guided dropout is not the encoding of the superposition theorem into a neural network.

5.3 Super-generalization of GD for a system with additive perturbations

In this section, we mathematically prove the super-generalization property of the guided dropout algorithm in some limited setting. Recall that super-generalization is the ability to predict efficiently the outcome $\boldsymbol{y}$ of a system S when tested on new structural vector $\boldsymbol{\tau}$ (while for regular generalization $\boldsymbol{\tau}$ is drawn similarly in the test set as it was during training). Although our mathematical proof rests on simplifying assumptions that do not hold in practice, we show in the experimental section (see chapter 6) that guided dropout exhibits super-generalization under a broad range of practical settings. The benefit of our limited theoretical analysis is to provide some justification and insight regarding the super-generalization. We also wanted to recall that this super generalization comes from the combination of unary actions that has been seen during training, this is effective in our setting because $S(\boldsymbol{x};(1,1,0\ldots,0))$ can be predicted using $S(\boldsymbol{x};(1,0,0\ldots,0))$ and $S(\boldsymbol{x};(0,1,0,0\ldots,0))$. Super generalization does not apply to a new unary action for example. In particular, we suppose then the guided dropout has seen every unary action during training.

We consider in this section a system S with *additive perturbation* (formally defined in Theorem 1). We begin by proving a theorem about super-generalization in the case of "perfect predictions" (meaning that $NN(\boldsymbol{x};\boldsymbol{\tau}) = S(\boldsymbol{x};\boldsymbol{\tau})$) in *standard generalization* (*i.e.* same $\boldsymbol{\tau}$ at training and test time). Then we generalize this result to the noisy case of imperfect predictions.

5.3.1 Perfect prediction case

Theorem 1. *Consider a system $S(\boldsymbol{x},\boldsymbol{\tau})$ that satisfies*

$$\begin{cases} S(\boldsymbol{x};\boldsymbol{\tau}^{\emptyset}) = F(\boldsymbol{x}) \\ S(\boldsymbol{x};\boldsymbol{\tau}^{\{i\}}) = F(\boldsymbol{x}) + \varepsilon_i(\boldsymbol{x}), \quad i = 1,\ldots,c \end{cases} \tag{5.5}$$

and

$$S(\boldsymbol{x};\boldsymbol{\tau}^{\mathscr{I}}) = F(\boldsymbol{x}) + \sum_{i\in\mathscr{I}} \varepsilon_i(\boldsymbol{x}), \quad \textit{for some } \mathscr{I} \in \{0,1\}^{\alpha} \textit{ with } |\mathscr{I}| \geq 2 \tag{5.6}$$

*for some (unknown) deterministic functions $F(\boldsymbol{x}),\varepsilon_1(\boldsymbol{x}),\ldots,\varepsilon_c(\boldsymbol{x})$. Then, for a **guided dropout** architecture $NN(\boldsymbol{x};\boldsymbol{\tau})$ with* linear *submodules $\boldsymbol{d}$ and $\boldsymbol{D}$ and that can make perfect predictions on training data triplets $(\boldsymbol{x},\boldsymbol{\tau},y)$ coming only from training set* *defined by equations (5.5), this network also makes perfect predictions on data coming from a broader test domains given by equation (5.6).*

Proof. Recall that the GD architecture $NN(\boldsymbol{x};\boldsymbol{\tau})$ gives predictions

$$\hat{y} = NN(\boldsymbol{x};\boldsymbol{\tau}) = \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}) + \boldsymbol{d}(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau})) \in \mathbb{R}$$

Since we assumed $n = 1$, $\boldsymbol{D}$ is actually a scalar linear function. By writing $\boldsymbol{\tau} = (\tau_1, \dots, \tau_c) = \sum_{i=1}^{c} \tau_i \boldsymbol{\tau}^{\{i\}}$, we can use the linearity of $\boldsymbol{d}$ and $\boldsymbol{D}$ to write the output of GD as

$$\begin{aligned}\hat{y} &= \boldsymbol{D}\left(\boldsymbol{E}(\boldsymbol{x}) + \boldsymbol{d}\left(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \sum_{i=1}^{c} \tau_i \boldsymbol{\tau}^{\{i\}}\right)\right)\\ &= \boldsymbol{D}\left(\boldsymbol{E}(\boldsymbol{x}) + \sum_{i=1}^{c} \tau_i \boldsymbol{d}\left(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau}^{\{i\}}\right)\right)\\ &= f_0(\boldsymbol{x}) + \sum_{i=1}^{c} \tau_i f_i(\boldsymbol{x})\end{aligned}$$

where

$$\begin{aligned}f_0(\boldsymbol{x}) &= \boldsymbol{D}\boldsymbol{E}(\boldsymbol{x})\\ f_i(\boldsymbol{x}) &= \boldsymbol{D}\boldsymbol{d}\left(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x})) \odot \boldsymbol{\tau}^{\{i\}}\right), \quad i = 1, \dots, c.\end{aligned}$$

As $NN(\boldsymbol{x};\boldsymbol{\tau})$ makes perfect predictions for any data point $(\boldsymbol{x}, \boldsymbol{\tau}, y)$ coming from training domain, i.e. $\boldsymbol{\tau} \in \mathscr{T}^{\text{train}} = \{\boldsymbol{\tau}^{\emptyset}, \boldsymbol{\tau}^{\{1\}}, \dots, \boldsymbol{\tau}^{\{c\}}\}$, for any $\boldsymbol{x} \sim D(\mathscr{X})$ and any $\boldsymbol{\tau} \in \mathscr{T}^{\text{train}}$, we have

$$S(\boldsymbol{x};\boldsymbol{\tau}) = NN(\boldsymbol{x};\boldsymbol{\tau})$$

which means that the following equalities hold

$$\begin{aligned}F(\boldsymbol{x}) &= f_0(\boldsymbol{x})\\ F(\boldsymbol{x}) + \varepsilon_i(\boldsymbol{x}) &= f_0(\boldsymbol{x}) + f_i(\boldsymbol{x}), \quad \forall i = 1, \dots, c.\end{aligned}$$

So we must have

$$\begin{aligned}F(\boldsymbol{x}) &= f_0(\boldsymbol{x})\\ \varepsilon_i(\boldsymbol{x}) &= f_i(\boldsymbol{x}), \quad \forall i = 1, \dots, c.\end{aligned}$$

for almost all $\boldsymbol{x}$. As in our case the distribution $D(\mathscr{X})$ of $\boldsymbol{x}$ doesn't depend on $\boldsymbol{\tau}$ (i.e. the distribution of $\boldsymbol{x}$ is the same for $\boldsymbol{\tau} \in \mathscr{T}^{\text{train}}$ AND for $\boldsymbol{\tau} \in \mathscr{T}^{\text{test}}$), the above equality also holds for $\boldsymbol{x}$ in target domains. So when we use the trained $NN(\boldsymbol{x};\boldsymbol{\tau})$ to make predictions for $\boldsymbol{\tau} \in \mathscr{T}^{\text{test}} = \{\boldsymbol{\tau}^{\mathscr{I}}, |\mathscr{I}| \geq 2\}$, the equality

$$NN(\boldsymbol{x};\boldsymbol{\tau}^{\mathscr{I}}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{c} \tau_i f_i(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{i \in \mathscr{I}} f_i(\boldsymbol{x}) = F(\boldsymbol{x}) + \sum_{i \in \mathscr{I}} \varepsilon_i(\boldsymbol{x}) = S(\boldsymbol{x};\boldsymbol{\tau}^{\mathscr{I}})$$

holds with probability 1 for any $\tau^{\mathscr{I}}$, which concludes the proof. □

The above theorem shows that under some conditions, **guided dropout** is indeed capable of performing *super-generalization*: making good predictions on unseen test domains : $\boldsymbol{\tau} \in \mathscr{T}^{\text{test}} \neq \mathscr{T}^{\text{train}}$. The fact that $\boldsymbol{d}$ and $\boldsymbol{D}$ are linear is essential for capturing the additivity of perturbations. For other types of perturbations such as the multiplicative case defined as follows

$$S(\boldsymbol{x};\boldsymbol{\tau}^{\mathscr{I}}) = F(\boldsymbol{x}) \prod_{i \in \mathscr{I}} (1+\varepsilon_i(\boldsymbol{x})),$$

GD can achieve similar results if $\boldsymbol{D}$ has, for example, exponential-like behavior (i.e. it transforms additions to multiplications).

On the other hand, the fact that Theorem 1 works for any unknown functions $\varepsilon_i(\boldsymbol{x})$ makes the results very general. We can consider for example linear perturbation with $\varepsilon_i(\boldsymbol{x}) = \boldsymbol{W}_i\boldsymbol{x}$, constant additive perturbation with $\varepsilon_i(\boldsymbol{x}) = \boldsymbol{\alpha}_i$ or heteroskedastic perturbations with $\varepsilon_i(\boldsymbol{x}) = \boldsymbol{\alpha}_i G(z)$ for a fixed function G, etc. And all these types of perturbation are special cases of the above theorem.

5.3.2 Imperfect prediction case

Theorem 1 gives a strong theoretical guarantee, but the condition of making *perfect* predictions on the training domain is too strong to be satisfied in practice. So we now investigate the case with *imperfect* predictions, i.e. the case where we don't require exact learning on training domains, for $\boldsymbol{\tau} \in \mathscr{T}^{\text{train}}$. To do this, we first need to introduce a notion of distance between functions to deal with. As we used mean square error (MSE) in our paper for the regression problem, the distance we use will be defined in the same flavor. Let μ be a probability measure on $\mathscr{X} = \mathbb{R}^p$ and f,g two functions from $\mathscr{X}$ to $\mathscr{Y} = \mathbb{R}^l$. We define the distance $d_\mu(f,g)$ between f and g to be

$$d_\mu^2(f,g) = \int_{\mathscr{X}} \|f(\boldsymbol{x}) - g(\boldsymbol{x})\|^2 d\mu(\boldsymbol{x}).$$

Here $\|\cdot\|$ represents the ℓ_2-norm on $\mathbb{R}^l$. Notice that this distance depends on the probability measure μ. If we put $\mu = D(\mathscr{X})$ (the ground truth distribution of $\boldsymbol{x}$) and write $y = g(\boldsymbol{x})$, the right hand side becomes nothing but the generalization error[4] of f. If we instead put $\mu = \frac{1}{m}\sum_{i=1}^m \delta_{\boldsymbol{x}_i}$ (the empirical distribution), the RHS becomes

[4]We remind that in our problem setting, the function to be learn $S(\cdot,\boldsymbol{\tau})$ is deterministic (solutions of differential equations), the labels y are thus deterministic given $\boldsymbol{x}$ and $\boldsymbol{\tau}$. So we don't need to consider the joint distribution on the pair $(\boldsymbol{x},y)$.

MSE test error (or training error). We are now ready to formulate our theorem on the imperfect prediction case.

Theorem 2. *Let $S(\boldsymbol{x};\boldsymbol{\tau})$ be a system satisfying the same conditions (5.5)(5.6) in Theorem 1. Let $NN(\boldsymbol{x};\boldsymbol{\tau})$ be a* ***guided dropout*** *architecture with linear submodules* ***d*** *and* ***D*** *such that*

$$\begin{aligned} d_\mu\left(NN(\cdot,\boldsymbol{\tau}^{\emptyset}),S(\cdot,\boldsymbol{\tau}^{\emptyset})\right) &\leq d_0 \\ d_\mu\left(NN(\cdot,\boldsymbol{\tau}^{\{i\}}),S(\cdot,\boldsymbol{\tau}^{\{i\}})\right) &\leq d_i, \quad i=1,\dots,c. \end{aligned} \tag{5.7}$$

for some constant $d_0,d_1,\dots,d_c \in \mathbb{R}$. Then for any $\mathscr{I} \subset \{1,\dots,c\}$, we have

$$d_\mu\left(NN(\cdot,\boldsymbol{\tau}^{\mathscr{I}}),S(\cdot,\boldsymbol{\tau}^{\mathscr{I}})\right) \leq (|\mathscr{I}|+1)d_0 + \sum_{i\in\mathscr{I}} d_i. \tag{5.8}$$

Proof. According to (5.5) and (5.6) in Theorem 1, we can write $S(\cdot,\boldsymbol{\tau}^{\emptyset}) = F$ and $S(\cdot,\boldsymbol{\tau}^{\{i\}}) = F+\varepsilon_i$. And since ***d*** and ***D*** are linear, we can write $NN(\cdot,\boldsymbol{\tau}^{\emptyset}) = f_0$ and $NN(\cdot,\boldsymbol{\tau}^{\{i\}}) = f_0+f_i$ according to the same argument in the proof of Theorem 1. Then we can rewrite (5.7) as

$$\begin{aligned} d_\mu\left(f_0,F\right) &\leq d_0 \\ d_\mu\left(f_0+f_i,F+\varepsilon_i\right) &\leq d_i, \quad i=1,\dots,c. \end{aligned}$$

Because the distance d_μ defined above satisfies triangle inequality (and is translation invariant), we have

$$\begin{aligned} d_\mu\left(NN(\cdot,\boldsymbol{\tau}^{\mathscr{I}}),S(\cdot,\boldsymbol{\tau}^{\mathscr{I}})\right) &= d_\mu\left(f_0+\sum_{i\in\mathscr{I}} f_i, F+\sum_{i\in\mathscr{I}}\varepsilon_i\right) \\ &= d_\mu\left(f_0+\sum_{i\in\mathscr{I}}[(f_0+f_i)-f_0], F+\sum_{i\in\mathscr{I}}[(F+\varepsilon_i)-F]\right) \\ &\leq d_\mu(f_0,F)+\sum_{i\in\mathscr{I}} d_\mu(f_0+f_i,F+\varepsilon_i)+\sum_{i\in\mathscr{I}} d_\mu(f_0,F) \\ &\leq d_0+\sum_{i\in\mathscr{I}} d_i+\sum_{i\in\mathscr{I}} d_0 \\ &= (|\mathscr{I}|+1)d_0+\sum_{i\in\mathscr{I}} d_i, \end{aligned}$$

which concludes the proof. □

Theorem 2 shows that as long as guided dropout approximates well the ground truth functions 5.5 by fitting data from training domain, its error on any test domain is bounded by a weighted sum of the errors on training domains. This can indeed be considered as a super-generalization property. Notice that the weight of d_0 is larger than that of the others, which suggests we should make more efforts on improving the accuracy of the predictions on data coming from $\boldsymbol{\tau} = \boldsymbol{\tau}^{\emptyset}$.

Theorem 1 is a special case of Theorem 2 by putting $d_0 = d_1 = \cdots = d_c = 0$. At last, we emphasize that the distance d_μ is a very flexible notion and can be considered as generalization error or more importantly, test error in practice. We can even have different μ in different distribution (one for the training set, and one for the test set) and the proof is still valid.

In this section, we mathematically proved that the guided dropout algorithm had some super generalization properties. If actions combine linearly, then a guided dropout architecture with linear decoders d and D will be able to make predictions with bounded error even on unseen actions $\boldsymbol{\tau} \notin \mathscr{T}^{\text{train}}$, where $\mathscr{T}^{\text{train}}$ is the training distribution of actions $\boldsymbol{\tau}$. This proof has been performed in the simplify settings of additive perturbations, but we believe our method to be more generic, and be able to combine actions non linearly. This is showed in more detail in the chapter 6 page 77.

5.4 Discussion

In the previous sections, we developed the guided dropout methodology and mathematically showed the "super generalization" property observed in practice. In this section, we try to develop insights about this method by comparing it to a physical simulator or to link it to other literature where an approach close to ours has been developed.

5.4.1 Comparison with a solver

In this subsection, we explain the main differences between the use of guided dropout methodology and the use of a solver based on Kirchhoff's laws (as the one detailed in chapter 2 for example).

The first difference, if properly used, a solver will probably be more accurate than this neural network. And in such a critical environment, accuracy is important.

But, a solver is in general slower than a neural network. Early experiments report a speedup of around 1 000 could get achieved with neural networks compare to solvers currently at used at RTE for example. With fewer computations, an operator could have faster results and have more time to take critical actions for example. Or, when studying forecasts, *i.e.* possible grid states for the future, this speed up could be used to study a higher number of powergrid states in the same times for example. This approach is developed in the ITESLA and GARPUR frameworks, as we developed in chapter 2 via proxy.

One major difference also lies in the quantity of data needed to perform a computation of flow given some injections. A solver requires all physical properties of all objects (*e.g.* productions, loads, lines etc.) of the power grid, whereas our method is data-driven. This entails two major differences. On the first hand, a carefully engineered optimizer has well-known properties, and it’s possible to rely on experts on the power system community to check its result. On the other hand, if some data are wrong, misinterpreted, or unavailable at the time of the study (such as the exact topology of the power grid for example), it is not possible to use a simulator, or once have also to forecast the missing variables (*e.g.* the topology), which may be a really hard problem.

5.4.2 Related works

We have recently become aware of parallel work from[37] and [8] in which the authors also encode actions by switching off some neurons, in application to frame predictions in video games, when the player’s actions are known. Our method differs however in multiple ways. First, the problem treated and the application are different: In our application to power grids, there is no temporal dependency between the inputs $\boldsymbol{x}$ and the outputs $\boldsymbol{y}$ (steady state modeling); in theirs, temporal aspects play a key role. Also, in the proposed methodology, we are rather interested in studying combinations of unseen actions by switching on and off some units. This induces that Oh et al. have focused their work on long-term frame forecasting (recurrent use of the architecture to forecast frame after frame) while this work is more focus on combinations of actions in a single grid state. We then believe this work to be novel and interesting for both the power system community as well as the machine learning audience.

The "next frame prediction" problem could also be addressed by our method if $\boldsymbol{x}$ represents the previous frame, $\boldsymbol{y}$ would be the next frame to be predicted, and $\boldsymbol{\tau}$ the action performed between the previous frame $\boldsymbol{x}$ and the frame we want to predict $\boldsymbol{y}$. This line of work will not be addressed in the work.

Chapter 6

Guided dropout Applications

In this section we introduce the main applications of the guided dropout algorithm developed in this work. Recall from Chapter 2 that the operator must find the action π^* that minimizes a given cost[1]. If we denote by p the probability of occurrence of a outage z, by $s = (\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$ a grid state and by π an action an operator can take and $\odot$ is the binary operator representing the application of a action on a power grid: $s \odot \pi$ means "applying action π on grid state s. The optimal action is modeled as:

$$\pi^* = \text{argmin}_\pi \left\{ \underbrace{CP(\pi, s)}_{\text{Preventive cost}} + \sum_{z \in \mathscr{A}} \underbrace{p(z)}_{\text{probability of occurrence of } z} \underbrace{\left(\underbrace{CC(\pi, s, z)}_{\text{Corrective cost of z}} + \underbrace{c(z; s \odot \pi)}_{\text{cost of fixing the power grid}} \right)}_{\text{cost of the grid after outage } z \text{ arises}} \right\} \tag{6.1}$$

All the applications detailed here have the goal to help him achieve it. Two problems are addressed in the three following sections:

- **Flow prediction:** Evaluate how power flows are affected by “contingencies” occurring on the powergrid (*e.g.* a line breaking), with the objective of simulating situations faster than with the physical simulator. This allows the operator to compute rapidly the state after an outage $s \odot z$, or to compute in a short period of time if action π is good for the power grid or not through the computation of grid state $s \odot \pi$.

- **Assisting operators:** The goal is to provide an estimation, in real time, of what are the most dangerous contingencies z, *i.e.* the one that could trigger

[1] More details have been provided about this modeling in the section 2.3.2 pages 24 and following.

cascading failures for example, so that the operator can take some preventive actions. Another application is to evaluate, in real time, the overall (would be) strain put on the grid, combining all (would be) contingencies, which we call "risk". The higher it is, the more careful the operator needs to be. If the risk is too high, the operator could receive an alarm, and perform an action that could set back the power grid in a least dangerous state for example.

This chapter is divided into two parts. In the first section, we report results of the guided dropout algorithm on two problems consisting in predicting flows from injections and topology when 1) some lines are disconnected 2) the topology of the grid changes. Flow prediction is a first step in assisting operators. But it is not enough. In the second section, we study a problem more related to assisting operators. Still, on synthetic datasets, we report how guided dropout allows to rank contingencies (allowing possibly better reaction time for operators) and on risk estimation (indicating a possibility of a blackout). These two last applications are direct metrics that can allow building trust with real-time operators. More work on these metrics and how they can be used on real dataset are on the way, and first results are shown in Chapter 7.

6.1 Predictions of flows

In this section, we explain how the guided dropout algorithm can perform better than some baselines in the predictions of power flow after a **outage** occurs or after a change of reconfiguration of the buses. These results were obtained in a synthetic and controlled experimental environment.

The material in this section is based on the work:
Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018d). Fast power system security analysis with guided dropout and
Donnot, B., Guyon, I., Liu, Z., Schoenauer, M., Marot, A., and Panciatici, P. (2018a). Latent Surgical Interventions in Residual Neural Networks. working paper or preprint

In all this section, the dataset for which the models are used is $\{(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})\}$, with $\boldsymbol{x}$ being the injections, sampled with the method described in Chapter 4, $\boldsymbol{\tau}$ is topology variations detailed in subsection 6.1.2 and $\boldsymbol{y}$ is the current flows on each powerline of the power grid. These flows $\boldsymbol{y}$ are computed using the simulator **Hades2**.

In this first application, we are interested in predicting the flows $\boldsymbol{y}$ (in amps only) on the 186 powerlines of the powergrid described in section 4.2, given some injections $\boldsymbol{x}$ only. In this experiment, to be fair with the baseline coming from the power system community, $\boldsymbol{x}$ includes only the active productions and active loads. No voltage information is provided, nor reactive load values.

Controlled experiments are performed on a standard medium-size benchmark from "Matpower" [56], a library commonly used to test power system algorithms [1]. We use case118, a simplified version of the Californian power grid. This test case includes 99 consumptions, 54 productions (dim $\boldsymbol{x} = 153$), and 186 power lines (dim $\boldsymbol{y} = 186$).

This section is organized as followed. First, we explain the baseline methods. Then we detailed how the data have been generated. We then report the experiments performed that demonstrate empirically the performance of the guided dropout methodology. Lastly, we sum-up what has been demonstrated in these experiments and their limits.

6.1.1 Baseline methods

We compared the performance of our proposed **Guided Dropout (GD)** method with multiple baselines, from other classical neural network models to a widely used physical approximation in power systems. A summary of these baselines can be found in Table 6.1, and their detailed description is given in the following paragraphs.

One Model is a brute force approach that does not scale well with the size of the power grid. One neural network is trained for each $\boldsymbol{\tau}$. A power grid with n lines would require training $n(n-1)/2$ neural networks to implement all “n-2” cases. Beside, confronted to a novel situation, unseen outages in training set, for example, this approach wouldn’t allow making predictions in that case. We show this model as some kind of lower limit on the loss we can achieve with neural networks.

One Variable (OV) is our simplest encoding allowing us to train one network for all “n-1” cases. One single input variable encodes which line is disconnected (0 for no line disconnected, 1 for line 1 is disconnected, 2 for line 2, etc.). However, it can’t generalize to “n-2” cases. Again, this model has little interest, as it can’t allow making predictions on unseen outages.

One Hot (OH) is a reference architecture, which allows us to generalize to “n-2” cases. If n is the number of lines in the power grid, n extra binary input variables are added, each one coding for connection/disconnection. This encoding is widely used in the machine learning literature when dealing with discrete variables like the line disconnections vector $\boldsymbol{\tau}$ in our problem.

The *DC approximation* is an approximation of the AC (Alternative Current) non-linear powerflow equations. It neglects reactive power and permits to quickly compute an approximation of power flows using a matrix inversion, given a detailed physical model of the grid. The power equations then look similar to the classical equations of electricity. This approximation is detailed in section page 80 and bellow. It is a relevant approximation for transmission power grid as it is usually accurate within 5-10% error on power flows. However, it is not used today at RTE for daily operations. In this experiment, we used only active loads values and active productions as inputs of our neural network. This is done to ensure a fair comparison with the DC approximation.

On the contrary to the DC approximation, adding other informations to a neural network is relatively easy. We expect the informations about reactive load value and production voltage setpoint to improve the performance of our method. However, this study has not been performed.

Table 6.1 **Description of the baselines in the experiments for predicting power flows**.

	Encoding of $\boldsymbol{\tau}$	Super Gen.?*	Comments
One Model	One neural network is trained for each $\boldsymbol{\tau}$	no	Unusable in practice. Not tested on buses reconfiguration.
One Var (OV)	One single input variable encodes which line is disconnected (0 for no line disconnected, 1 for line 1 is disconnected, 2 for line 2, etc.)	no	Not tested on buses reconfiguration.
One Hot (OH)	*n* extra binary input variables are added, each one coding for connection/disconnection	yes	
DC approximation	Common baseline used in the power system community	yes	Not a neural network, doesn't need to be trained.

* "Super Gen.?" here denotes the fact, for a model learned on a dataset, its capacity to be used, without retraining, on unseen structural variables $\boldsymbol{\tau}$. It does not presume any performance on the super generalization set.

6.1.2 Dataset Generation

In this section, our goal is to demonstrate empirically that multiple grid topologies can be modeled with a single neural network. Our purpose is to provide fast **current**

flow predictions (measured in Amps), for given injections $\boldsymbol{x}$ and given topologies, to anticipate whether some lines might exceed their thermal limit should an outage occur. Recall that both of the following experiments are conducted on a medium size benchmark grid from Matpower [56] with $n = 186$ lines, 99 loads, and 54 injections. We want to assess the robustness of the guided dropout methodology in two different settings:

- When powerlines are disconnected from a power grid. Doing so, we hope the guided dropout model can be used to predict which powerline, if disconnected by windstorm, for example, causes other powerlines to exceed their thermal limits. If such a power line exists, operators should then take specific actions to cure the grid.
- With a broader class of topological changes: when buses are reconfigured. This experiment demonstrates the possibility to use the neural network to assess if a given topological action (change of topology) made by an operator is good or bad for the power grid for example.

Both these problems are useful to be able to propose, in real time, actions that would benefit the security of the power grid, or even detect possible weakness in advance. In the next paragraphs, we detail the method used to generate these datasets. Note that $\boldsymbol{x}$ are sampled with a method described in Chapter 4, and $\boldsymbol{y}$ is computed with **Hades2**. The major differences on these datasets it about what $\boldsymbol{\tau}$ represents, and how it is built. Following paragraph focus then on the description of $\boldsymbol{\tau}$.

Line disconnections In this setting we want to test whether or not, if a power line where to be disconnected, the grid would remain safe. For this experiment, $\boldsymbol{\tau}$ encodes the presence/absence of a power line. For example $\boldsymbol{\tau}_i = 1$ means that powerline i is disconnected. We won't consider any other topological changes in this experiment.

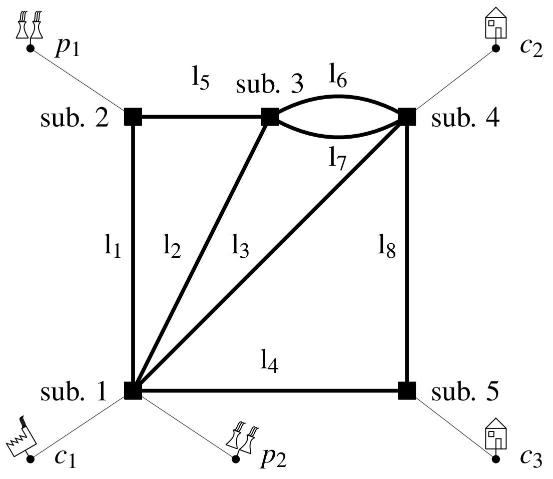
For the training, validation and test set, we use all the 187 variants of grid topologies with zero or one disconnected powerline. For each of these 187 different $\boldsymbol{\tau}$, we sample 10 000 different injections set $\boldsymbol{x}$ and use Hades2 to compute the resulting flows $\boldsymbol{y}$. This resulted in an "n-1" dataset of $1,870,000$ samples (we include in the "n-1" dataset samples for the reference topology)[2] Among this dataset, 50% is used for training, 25% for hyper-parameter selection, and 25% for testing.

We built another dataset for evaluating the performance of our model on unseen topologies $\boldsymbol{\tau}$. Uniformly at random, we sample 200 cases of pairs of disconnected

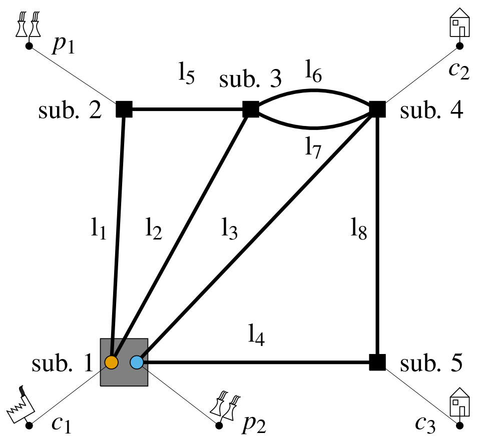
[2]Note that this "n-1" dataset include cases where one line has been disconnected and cases where no lines have been disconnected.

lines (**“n-2” dataset**), out of $186*185/2$ to assess whether or not the model could "super generalize" to unseen configurations. This model is then tested on 200 different $\boldsymbol{\tau}$ (each having exactly 2 components equal to 1) out of $186*185/2 = 17205$ possible $\boldsymbol{\tau}$. For each of these $\boldsymbol{\tau}$, as we did for the other datasets, we sampled 10 000 different injections $\boldsymbol{x}$ and run Hades2 to compute the flows. This results in a “n-2” dataset of $2,000,000$ samples.

Buses reconfiguration In this setting, we want to rapidly assess the current flows $\boldsymbol{y}$ if a given action (topological changes) is performed. Topology changes consist in reconfiguring line connections in one or more substations (see Figure 6.1). To perform this study, we adopted the following modeling. The reference topology showed for a small example in Figure 6.1a, where everything is interconnected is encoded by $\boldsymbol{\tau} = \tau^{\emptyset} = [0,0,\dots,0]$. On most powergrid, it is possible to reconfigure the buses in some substation. An example of such reconfiguration, for a small example, is showed on Figure 6.1b. This would correspond a "unary change" (as it affects only one substation) and is encoded for example by $\boldsymbol{\tau} = [1,0,\dots,0]$. There is only one 1 as the change is unary.



(a) Reference topology, is encoded by $\boldsymbol{\tau} = [0,0,\dots,0]$



(b) Reconfiguration of substation 1 in two buses. This topology is encoded by $\boldsymbol{\tau} = [1,0,\dots,0]$ for example.

Fig. 6.1 **Illustration of bus reconfiguration at substation 1**. Compare to the reference topology (everything is connected to everything) showed on the left (Figure 6.1a), the substation 1 is split into two independent buses on the right (Figure 6.1b).

This encoding is a somewhat different from others encoding. Let’s focus on some of the difference. To each possible *unary topology* action correspond a unique vector $\boldsymbol{\tau}$. We hence have the property sizeof($\boldsymbol{\tau}$) = number of unary topological changes.

Combining two or more *unary topological change* ***on different substations*** is done the usual way, by making an element wise "or" operation[3]. This encoding is *injective* (two different topologies are encoded by two different $\boldsymbol{\tau}$). The main difference compare to previous encoding of $\boldsymbol{\tau}$ is that the encoding is not *surjective* (some $\boldsymbol{\tau}$ doesn't encode for a real grid state). For example, $\boldsymbol{\tau}^{(1)} = [1,0,0,\dots,0]$ encodes for change of topology in substation 1. Now let's assume $\boldsymbol{\tau}^{(2)} = [0,1,0,\dots,0]$ encodes also for a topology in substation 1 (different from the reference and from $\boldsymbol{\tau}^{(1)}$), then the vector $\boldsymbol{\tau}^{\{1,2\}} = [1,1,0,\dots,0]$ doesn't code a valid grid state. In fact the topology of substation 1 cannot take 2 different values. This is has absolutely no impact on the task we are trying to achieve. An operator choose certain topology for certain substations, the corresponding vector $\boldsymbol{\tau}$ counts as much 1 as the number of substations where the topology is not the reference topology.

With all the precautions needed[4] we counted 11 558 possible unary actions (corresponding to single bus splitting or merging, compared to the reference topology) in the case118 powergrid of Matpower. For this experiment, we sampled randomly 100 different possible changes among the 11 558 relevant changes. This imply that, for this experiment $\dim(\boldsymbol{\tau}) = 100$.

To build the dataset, we sampled 50 000 different inputs $\boldsymbol{x}$ in the reference topology ($\tau^{\emptyset}$), and use Hades2 to compute the flows $\boldsymbol{y}$. Then for each unary action $\boldsymbol{\tau}^{(i)}$ (among the 100 that have been selected), we sampled 1000 inputs $\boldsymbol{x}$. This resulted in a dataset of 150 000 rows. 70% of this dataset has been used for training, 15% as a validation set for hyperparameters selection and 15% as a test set to report errors on this manuscript.

The super test dataset is composed of 1500 combination of two unary actions among the pairs of valid possible binary actions. Then, for each of these 1500 $\boldsymbol{\tau}^{\{i,j\}}$, we sampled 100 inputs $\boldsymbol{x}$. We used the same physical simulator (Hades2) to compute the $\boldsymbol{y}$ from the $\boldsymbol{x}$ and the $\boldsymbol{\tau}$. The super-generalization set counts then 150 000 rows, corresponding to 150 000 different triplets $(\boldsymbol{x}, \boldsymbol{\tau}^{\{i,j\}}, \boldsymbol{y})$.

A summary of the composition of each dataset is shown in Table 6.2.

6.1.3 Results

In this section, we expose the results of both of these experiments. In both cases we show a phenomenon that we call **super-generalization**: with the "guided dropout" topology encoding. A neural network, trained with unary action[5], generalizes to binary

[3]This is described in more details in chapter 5

[4]We impose that the power grid must be represented as a connected graph, or that no loads or productions get disconnected.

[5]Action being powerline disconnection or buses reconfiguration depending on the experiment

Table 6.2 **Summary of the composition of the dataset for assessing the flows on controlled experiments**.

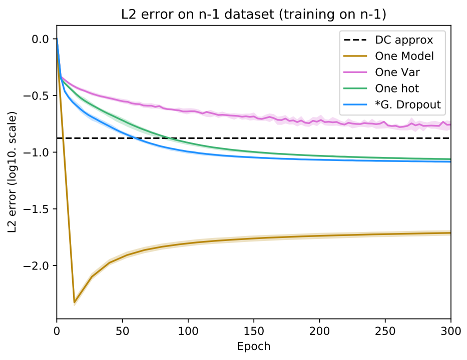
	Lines disconnections	Buses reconfiguration
injections $\boldsymbol{x}$	sampled (see Chapter 4, Section 4.2 page 49)	
Description of $\boldsymbol{\tau}$	$\boldsymbol{\tau}_i = 1$ line i is disconnected $\dim(\boldsymbol{\tau}) = 186$	$\boldsymbol{\tau}_i = 1$ i^{th} topology apply $\dim(\boldsymbol{\tau}) = 100$
flows $\boldsymbol{y}$	computed via the simulator Hades2	
Training Set^{+} Validation Set^{+} Test Set^{+}	at most one line disconnected	at most one substation with a topology different from its reference's
Super Test Set	exactly 2 lines disconnected Counting 200 different $\boldsymbol{\tau}$	exactly two substations with a different topology than their reference's Counting 1500 different $\boldsymbol{\tau}$

$^{+}$ The training set, the validation and the test all contain different grid states. There are no grid state that are present in two of these dataset.

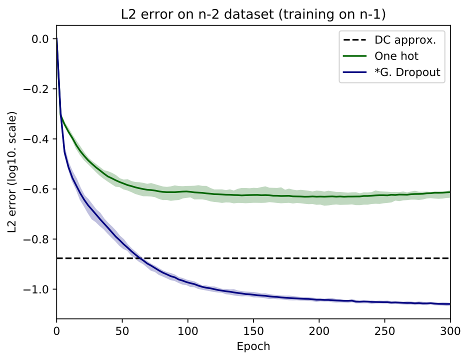
actions cases. We first report the results on the powerline disconnection dataset, then on the more challenging buses reconfiguration one.

On powerline disconnection Figure 6.3 shows **generalization** and **super generalization** learning curves for the various methods. We recall that, as explained in Chapter 3, "generalization" in this case is the property to obtain good performance on a test set similar to the training set (so in this example in a test set where at most one powerline have been disconnected at a time), and the "super generalization" is the ability to predict accurately flows in an unseen grid topology (in this case "n-2" dataset, when no double line disconnections have been observed in the **training set**). For reasons given above, super-generalization can only be achieved by One Hot and Guided Dropout, which explains that Figure 6.3-b has only two curves. The DC approximation is represented as a horizontal dashed line (since it does not involve any training). The test error is represented on a log scale. Hence, it can be seen in Figure 6.3 that neural networks are very powerful at making load flow predictions since the “One Model” approach (yellow curve) outperforms the DC approximation by an order of magnitude.[6] However the “One Model” approach is impractical for larger grid

[6]We note in the yellow curve some slight over-fitting as evidenced by the test error increase after 10 training epochs, which could be alleviated with early stopping.



(a) **Regular generalization.**

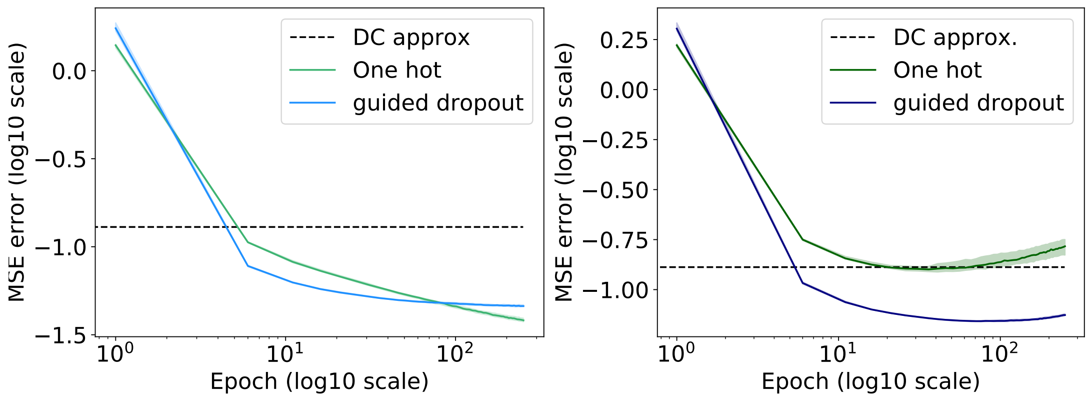


(b) **Super-generalization.**

Fig. 6.2 **Grid of 118 buses (nodes).** We show the L2 error in Amperes on a log scale as a function of training epochs. The neural network in both cases is **trained for all "n-1" cases** with multiple examples of injections. (a) **Regular generalization.** Test set made of (all) test injections for **"n-1" cases**. (b) **Super-generalization.** Test set made of a subset of test injections for **"n-2" cases**. Error bars are 25-75% quantiles over 10 runs having converged.

sizes. The "One Hot" approach is significantly worse than both "Guided Dropout" and "DC approximation". Of all neural network approaches, "Guided Dropout" gives the best results, and it beats the DC approximation for "n-2" cases (super-generalization).

The super-generalization capabilities of neural networks trained with "Guided Dropout" are obtained by combining in a single network "shared" units trained with all available data for many similar (yet different) grid topologies and specialized units activated only for specific topologies. Indeed, power grids, like many industrial systems, operate most of the time around nominal conditions. Given the nature of such operations, it is hence interesting to learn a reference model for our system in these nominal conditions for which we have a lot more observations and only dedicate few additional units to encode those occasional changes. This economy of resources, similar in spirit to weight sharing in convolutional neural networks, performs a kind of regularization. Obtaining a good performance on new "unseen" grid topologies (not available for training) is the biggest practical advantage of "Guided Dropout": Acquiring data to train a model for all "n-2" cases for the Extra High Voltage French power grid (counting $\simeq 1{,}400$ nodes, $\simeq 2{,}700$ lines) would require computing $\simeq 50$ million power flow simulations, which would take almost half a year, given that computing a full AC power flow simulation takes about 300 ms for RTE current production software. Conversely, RTE stores almost all "n-1" cases as part of "security analyses" conducted every 5 minutes, so the "n-1" dataset is readily available.



(a) Generalization (b) Super-generalization

Fig. 6.3 **Synthetic data on a 118 node grid.** We show the MSE error in Amperes on a log scale as a function of training epochs. Neural networks are trained with 15000 injections, for the reference topology $\boldsymbol{\tau}^{\emptyset}$ and unary changes $\boldsymbol{\tau}^{(i)}$. (a) **Regular generalization.** Test injections for **unary changes** $\boldsymbol{\tau}^{(i)}$. (b) **Super-generalization.** Test injections for **binary changes** $\boldsymbol{\tau}^{(ij)}$. Error bars represents the [20%, 80%] intervals, computed on 30 independently trained model.

On buses reconfiguration We compared the proposed guided dropout method with two benchmarks: the DC approximation and the one hot. We optimized the L2 (mean-square) error, using the Adam optimizer from Tensorflow. To make the comparison least favorable to the guided dropout architecture, all hyper-parameters of the neural network (learning rates, number of units, etc.) were optimized by cross-validation for the baseline network architecture.

The results are shown in Figure 6.3 indicate that the guided dropout method (blue curves) performs better than the DC approximation (black dashed line) both for regular generalization and super-generalization. In contrast, the one hot neural network architecture (green curves) does not outperform the DC approximation in the super-generalization case (Figure 6.2b).

Figure **??** indicates that the guided dropout architecture may possibly be slightly under-fitting since it is outperformed by the baseline one hot encoded neural network for regular generalization. This can be explained by the fact that the baseline network has many more available connections to learn from (no unit in the inner layer being disabled). Adding more hidden units in the guided dropout might yield yet better performance.

Figure 6.2b shows that the baseline neural network architecture is not viable: not only does it perform worse than the DC approximation, but its variance is quite high. While it is improving in regular generalization with the number of training epochs, its super-generalization performances get worse.

6.1.4 Conclusion of this work

Our comparison of various approaches to approximate “load flows" (predictions of power flows in electricity transmission grids) using neural networks has revealed the superiority of “Guided Dropout”. It allows training a single neural network to predict power flows for variants of grid topology (powerline disconnection or buses reconfiguration). Specifically, when trained on all variants with one unary modification, lines disconnection or substations buses reconfiguration, the network generalizes to variants with BINARY modifications. Given the combinatorial nature of the problem, this presents a significant computational advantage. We can also note that the guided dropout performances do not decrease whether it treats a problem of lines disconnections or a more complex problem of buses reconfigurations. This result is quite promising.

We empirically demonstrated on standard benchmarks of AC power flows that our method compares favorably with several reference baseline methods including the DC approximation used by the power system community, both in terms of predictive accuracy and computational time. In daily operations, only “n-1” situations are examined by RTE because of the computational cost of AC simulations. “Guided Dropout” would allow us to rapidly pre-filter alarming “n-2” situations, and then to further investigate them with AC simulation. Preliminary computational scaling simulations performed on the Extra High Voltage French grid indicate the viability of such hybrid approach: A neural network would be $\simeq 300$ times faster than the currently deployed AC power flow simulator.

Our target application is to pre-filter serious grid contingencies such as combinations of line disconnections that might lead to equipment damage or service discontinuity. This line of work is further examined in the next sections.

6.2 Operator decisions support

In the previous section, we explained how the guided dropout methodology could be used to accurately predict flows even on some unseen grid configuration $\boldsymbol{\tau}$. In this section, we explain how this can be embedded in a tool to help operators make better decisions. We detail how a neural network trained with guided dropout on only a small subset of contingencies can be used to assist operators. For the brevity of notation, a grid state $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$, with $\boldsymbol{x}$ being the injections, $\boldsymbol{y}$ the flows and $\boldsymbol{\tau}$ a modeling of the topology is denoted by s in this section.

Material in this sections comes from the papers: Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018c). Anticipating contingengies in power grids using fast neural net screening. In *IEEE WCCI 2018*, Rio de Janeiro, Brazil and Donnot, B., Guyon, I., Marot, A., Schoenauer, M., and Panciatici, P. (2018b). Optimization of computational budget for power system risk assessment. working paper or preprint

This section is organized as followed.

First we will introduce notations and formulate the problem. The first subsection details how we could mathematically formulate "assisting" operators. We detailed two possible applications: the first one is ranking the dangerous contingencies, from the more dangerous to the least dangerous. The second one is the to provide, given a security criteria, how secure is a power grid states.

Then we present the methodology for this two applications. First the methodology adopted for ranking contingencies is explained. Then we detailed how the method to rank contingencies can be adapted to allow a real time risk estimation.

Finally we report the results of the two methodologies on a standard benchmark of the litterature. The security criteria used is the "N-2" security criteria. This is not possible to simulate it currently for TSO, yet this proposed methodology is able to perform this computation.

6.2.1 Notations and problems formulation

We always analyze a situation corresponding to fixed injection in this section and sometimes omit $\boldsymbol{x}$ for brevity of notation. We also omit to specify time ordering, although states are time ordered. What we denote by $z \in \mathscr{Z}$ are **sudden would-be (potentially disruptive) events**, corresponding to a change in grid topology, such as a **line disconnection**, assuming injections remain constant. We denote by $z^{\{i\}}$ the disconnection of line i and more generally by $z^{\mathscr{I}}$ ($\mathscr{I}$ being a subset of $\{1,2,\ldots,n\}$) the disconnection of all the powerlines in the set $\mathscr{I}$. With this notation, $z^{\{1,2\}}$ corresponds to the outage where powerlines 1 and 2 are disconnected. Contingencies are encoded with a vector $\boldsymbol{\tau}$, such that contingencies $z^{\mathscr{I}}$ is encoded with: $\boldsymbol{\tau}_i = 1 \forall i \in \mathscr{I}$ and $\boldsymbol{\tau}_i = 0 \forall i \notin \mathscr{I}$. For example $z^{\{1,2\}}$ is encoded by $\boldsymbol{\tau} = [1,1,0,0,\ldots,0]$.

An outage z might arise with probability $p(z)$[7] and is associated with a loss function $L(z;s)$. In our application context, we assume that $L(z;s)$ is the $\{0,1\}$ loss, with 0 meaning that the outage z arising in state s is innocuous and 1 that it is risky or “bad” or "dangerous" for our system (*i.e.* at least one **transmission line**, still in service after

[7]For instance, events z might be single line disconnections occurring with probability $p(z) = \zeta(1)$ or double line disconnections occurring with probability $p(z) = \zeta(2) = \zeta(1)^2$.

z arose, exceeds its thermal limit)[8]. Thus:

$$L(z;s) = \begin{cases} 0 & \text{“No current flowing on any line} \\ & \text{exceeds the line thermal limit} \\ & \text{after outage } z \\ & \text{in grid state } s\text{”} \Rightarrow \text{OK} \\ 1 & \text{“Otherwise”} \Rightarrow \text{“Bad” event} \end{cases} \tag{6.2}$$

The “severity score” $\psi(z;s)$ of a outage z on grid state s is defined as:

$$\psi(z;s) \overset{\text{def}}{=} p(z).L(z;s) \tag{6.3}$$

This represent the expected loss of this outage. This is the loss of this outage weighted by its probability of occurrence.

We are interested in evaluating the risk taking into account all single and double outage. This leads to neglect all higher order contingencies. Today, operators take into account mainly the risk a single power lines disconnection when following the "N-1" security criteria[9]. This approach is an improvement from the "N-1" security criteria towards the formulation of the risk recalled in Equation 6.1. First, it takes into accounts double outage, which the "N-1" doesn't. And secondly it does not assume all contingencies have the same probability of occurring. It also does not implies that their cost is the same. In the proposed modeling, this means that $\mathscr{Z}$ the set all considered contingencies is composed of $\mathscr{Z} = \{z, z \text{ is a single or double outage}\}$. And, as the powergrid counts $n = 186$ powerlines, $|\mathscr{Z}| = n.(n-1)/2 = 17205$. We can now define the total risk, as being the overall strain put on the grid. This is done by considering the severity score of all contingencies:

$$R_{\max}(s) = \sum_{z \in \mathscr{Z}} \psi(z;s) \tag{6.4}$$

$$= \sum_{z \in \mathscr{Z}} p(z)L(z;s) \tag{6.5}$$

[8]This is a simplification. The real damage of the grid would endure after outage z would require computing a full "cascading failure" (as presented in [22] for example), which is computationally too expensive to calculate presently, even for a small test case like ours.

[9]This criterion is more detailed in Chapter 2

Finally, we observe that, if $\mathscr{V} \subset \mathscr{Z}$ is a set of contingencies, then we have:

$$R_{\max}(s) = \sum_{z \in \mathscr{Z}} p(z)L(z;s) \tag{6.6}$$

$$= \sum_{z \in \mathscr{V}} p(z)L(z;s) + \sum_{z \notin \mathscr{V}} p(z)L(z;s) \tag{6.7}$$

This is true for any set $\mathscr{V}$. The way we define this set $\mathscr{V}$ allows us to estimate accurately this risk $R_{\max}(s)$.

6.2.2 Methodology

In this subsection, we describe the method that allows to rank contingencies in decreasing order of gravity, *i.e.* with the one having the most important severity score $\psi(z;s)$ first. Given a computation budget, this ranking allows to use the real physical simulator only on the most dangerous one, and thus to evaluate the risk $R_{\max}(s)$ accurately. The exact methods to perform these two tasks is developed in the next paragraph.

Outage ranking

Here we show how to rank contingencies z (encoded in our neural network by $\boldsymbol{\tau}$) with respect to their estimated severity score $\hat{\psi}(z;s)$ for a given system state s. In this work $\hat{L}(z;s)$ denotes the approximation of the true loss provided by an **neural network**, trained on simulated data generated using a high-end load flow simulator. The data are simulated with the method described in 4.2. Consider a fixed grid state s and a given outage z, we denote by f_i the flow, computed with the high-end simulator, on the i^{nth} line of grid s after outage z occurs, and by $\bar{f}_i$ the thermal limit for this line.

Summary of the workflow Here is a summary of workflow for the ranking process. More detailed about each of this step is provided in the next paragraph. Let s be the considered grid state.

1. For each outage z :
   (a) Assess the flows $\hat{f}_i$ on each powerline of the grid after outage z using an artificial neural network
   (b) Compute the score $\hat{L}_i$, for each line to be above its thermal limit $\bar{f}_i$.
   (c) Compute the score for the grid to be unsecure after outage z:

$$\hat{L}(z;s) \overset{\text{def}}{=} \max_{1 \leq i \leq n} \hat{L}_i(z;s)$$

   (d) Assign a score $\hat{\psi}(z;s)$ to the outage z

2. Rank the outages z according to their score $\hat{\psi}(z;s)$

Detailed workflow We propose to first train a neural network with "guided dropout", as described in [14] to approximate rapidly the power flow for the given grid state s and outage z. We denote by $\hat{f}_i$ the flow predicted by our proxy (in this case our neural network) for the i^{nth} line of the power grid.

It has been observed that neural network tend to be "overconfident" in their predictions (see for example [36]). This overconfidence could lead to a bad ranking in practice with dramatic effects. We propose to calibrate the score of our neural network to take into account a fixed (yet calibrated) uncertainty by assuming:

$$\forall i, (f_i - \hat{f}_i) \sim \mathcal{N}(0, \sigma_i) \tag{6.8}$$

where σ_i represents the model uncertainty for line i. We calibrate the vector $\boldsymbol{\sigma}$ (of dimension n) using a calibration set distinct from the training set the neural network was trained on and also distinct from the final test set we use to evaluate performance. During our experiments, this calibration set has the same distribution as the test set. It is composed of 100 different grid states. And for each grid states a all single and double contingencies have been simulated.

On this calibration set, we compute the true values f_i, using the high-end simulator, and the predictions $\hat{f}_i$ coming from our proxy (neural network). Then, σ_i is set to:

$$\sigma_i \overset{\text{def}}{=} \frac{1}{\text{number of simulations}} \cdot \sum_{\text{simulations} s} (\hat{f}_i - f_i)^2 \tag{6.9}$$

These σ_i's are then used to compute the scores $\hat{L}_i$ that a given line is above its thermal limit as:

$$\hat{L}_i \overset{\text{def}}{=} 1 - F_{\sigma_i}(\bar{f}_i - \hat{f}_i) \tag{6.10}$$

where F_{σ_i} is the cumulative density function of the Normal law with mean 0 and variance σ_i.

This gives us a score for each power line. For our problem, a grid is said to be " non-secure" after outage z, if at least one of its line is above its thermal limit. The score of the power grid, in state x after outage z, is then obtained by:

$$\hat{L}(z;s) \overset{\text{def}}{=} \max_{1 \leq i \leq n} \hat{L}_i(z;s) \tag{6.11}$$

We have now defined $\hat{L}(z;s)$. If all outages z were equiprobable ($\zeta(1) = \zeta(2)$), we could use this approximation for ranking. In practice, some events occur more often than others and therefore must be prioritized accordingly: as some event occur more often, it's natural to place them first on the sorted list.

The first thing we tried was to rank directly the outages z with respect to their relative cost $p(z)\hat{L}(z;s)$. This worked, but this order tends to be really conservative: all the single outages were ranked first. This is not surprising given the relative simplicity of the error modeling[10] To remedy this problem, we tried various weighting schemes, and found that scaling the "n-1" event relatively to the "n-2" events with the logarithm of $\zeta(1)/\zeta(2)$ leads to the best results in almost every situation $(z;s)$, and use this for the ranking. To wrap up, all the grid states and outages are sorted with respect to their "scores" $\hat{s}(z;x)$ defined as:

$$\hat{\psi}(z;x) \overset{\text{def}}{=} \begin{cases} \hat{L}(z;x) \times \log_e \left(\dfrac{\pi_{(1)}}{\pi_{(2)}} \right) \text{if } z \text{ is a single outage} \\ \hat{L}(z;x) \text{ otherwise} \end{cases} \tag{6.12}$$

Risk estimation

In this section we try to provide an accurate estimation of the total risk $R_{\max}(s)$ of the grid state s. We recall that this risk is computed on all single and double outage. Today operators do not consider this kind of risk systematically in daily operation planning. Our objective here is to use the same computational budget as what operators do today (*i.e.* computing exactly n power flow using a high-end simulator) and still be able to estimate the total risk. In order to evaluate our method, we denote by η the computational budget at our disposal, knowing that we are more interested in cases where $\eta \approx n$.

Summary of the workflow Here we summarize the workflow of this risk estimation. This workflow is really similar to the one used for ranking outages. We recall here the ranking process to present a complete workflow. The ranking is detailed in the next paragraph, however.

1. For each outage z :

[10]There is a 10^3 relative factor between $\zeta_{(1)}$ and $\zeta_{(2)}$, so for a double outage $z_{i,j}$ to be simulated before a single one z_i, this would mean that $\hat{L}(z_{i,j};s) > 10^3.\hat{L}(z_i;s)$. But, we made various hypotheses, in particular, that the error on the flow where normally distributed (see equation 6.8) which is not the case in reality. In general we have then $\hat{L}(z_i;s) - L(z_i;s) >> 10^{-3}$. Even if the outage z_i is harmless (*i.e.* $L(z_i;s) = 0$), this would imply $\hat{L}(z_i;s) > 10^{-3}$, so this single outage would be ranked before any "n-2" outage, which of course is not true in practice.

(a) *Assess the flows $\hat{f}_i$ on each powerline of the grid after outage z using an artificial neural network

(b) *Compute the score $\hat{L}_i$, for each line to be above its thermal limit $\bar{f}_i$.

(c) *Compute the score for the grid to be not secure after outage z:

$$\hat{L}(z;s) \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} \hat{L}_i(z;s)$$

(d) Assign a score $\hat{\psi}(z;s)$ to the outage z. This step is also present in the ranking section, but it is slightly modified here compared to the ranking procedure.

2. * Rank the outages z according to their score $\hat{\psi}(z;s)$

Then, once this has been performed:

1. For the η top-ranked outages (*i.e.* the most dangerous for the power grid according to the neural network):
   - Assess its risk $L(z;s)$ using the slow simulator
   - Sum the risk for each such most dangerous outages
2. For all the other outages (lowest ranked outages, the ones that are the least dangerous for the powergrid)
   - Assess its estimated risk $\hat{L}(z;s)$ using only machine learning
   - Sum the risk for each such least dangerous outages
3. Add up the two cumulated risk

* denote the steps already developed in the previous subsection to rank the outages. We do not cover their detail again in the detailed work-flow.

Detailed workflow The first steps of this application are identical to the steps performed to compute the ranking. We do not recall them here. Thanks to Equation 6.11, recalled for clarity:

$$\hat{L}^{(\text{aux})}(z;s) \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} \hat{L}_i(z;s)$$

we have an estimator $\hat{L}^{(\text{aux})}(z;s)$ of the loss of outage z on grid state s. This is a ***biased*** stochastic estimator[11] of the true risk $L(z;s)$: $\mathbb{E}\left(\hat{L}^{(\text{aux})}(z;s)\right) \neq L(z;s)$. If we take the

[11]The stochastic aspect comes from the training data. A different training dataset would lead to a different estimator.

expectation over all possible training set, our estimator will not converge to the true value of the risk. One novelty of this method is to create an unbiased estimator of this quantity. To remove this bias, we proceed in two steps. First, we estimate this bias on the calibration set:

$$b(z;s) \overset{\text{def}}{=} \mathbb{E}_{\text{calibration set}} \left(\hat{L}^{(\text{aux})}(z;s)\right) - L(z;s) \tag{6.13}$$

and we subtract this bias to the previous estimator to get an unbiased estimator of L(z; s):

$$\hat{L}(z;s) \overset{\text{def}}{=} \hat{L}^{(\text{aux})}(z;s) - b(z;s) \tag{6.14}$$

This "evaluated loss" $\hat{L}(z;x)$ is an unbiased estimator[12] of the loss of the outage z: $L(z;x)$. An unbiased estimator of the severity score of outage z on situation x is then:

$$\hat{\psi}(z;s) \overset{\text{def}}{=} \hat{L}(z;s)p(z) \tag{6.15}$$

After having ranked the outages in decreasing score, we use the slow simulator to compute the risk of the most dangerous outages z, given the available computational budget η. Formally, let $\mathscr{V}_\eta$ be the set of the η most dangerous outages according to the estimated severity score. We have estimated with the simulator based on Kirchoff's laws the risk:

$$R(\eta;s) = \sum_{z \in \mathscr{V}_\eta} p(z)L(z;s) \tag{6.16}$$

We then reuse the approximations $\hat{\psi}(z;s)$ of $\psi(z;s)$ to define the residual risk (*i.e.* the risk of not computing accurately the set of outages in $\mathscr{Z} - \mathscr{V}_\eta$):

$$\hat{R}_{\text{Residual}}(\mathscr{V}_\eta;s) = \sum_{z \notin \mathscr{V}_\eta} p(z)\hat{L}(z;s) \tag{6.17}$$

Note that this is an estimator of the residual risk. This is the reason why it is hatted.

And finally the estimation of the risk is given by:

$$\hat{R}_{\max}(\eta;s) = R(\mathscr{V}_\eta;s) + \hat{R}_{\text{Residual}}(\mathscr{V}_\eta;s) \tag{6.18}$$

This is the application of Equation 6.7 where $\mathscr{V} = \mathscr{V}_\eta$

[12]We have: $\mathbb{E}(\hat{L}(z;s)) = \mathbb{E}(\hat{L}^{(\text{aux})}(z;s)) - b$. But, by definition $b \overset{\text{def}}{=} \mathbb{E}\left(\hat{L}^{(\text{aux})}(z;s)\right) - L(z;s)$. Thus $\mathbb{E}(\hat{L}(z;s)) = \mathbb{E}(\hat{L}^{(\text{aux})}(z;s)) - (\mathbb{E}_{\text{calibration set}}\left(\hat{L}^{(\text{aux})}(z;s)\right) - L(z;s))$. If, as we supposed the calibration is good enough to do the approximation: $\mathbb{E}_{\text{calibration set}}\left(\hat{L}^{(\text{aux})}(z;s)\right) \approx \mathbb{E}\left(\hat{L}^{(\text{aux})}(z;s)\right)$. Finally $\mathbb{E}(\hat{L}(z;s)) = \mathbb{E}(\hat{L}^{(\text{aux})}(z;s)) - (\mathbb{E}\left(\hat{L}^{(\text{aux})}(z;s)\right) - L(z;s)) = L(z;s)$.

6.2.3 Results

In this subsection, we expose the results of the two methods, each in one following paragraph. But first, let's introduce the way the datasets have been generated.

Dataset generation We tested our methods on the case118 of matpower that we already use in the previous section. We use the same method (*i.e.* detailed in Chapter 4) to sample some injections $\boldsymbol{x}$ and use Hades2 to compute the flows $\boldsymbol{y}$ from the injections and the topologies. The topology of the powegrid has been kept constant, and only changes due to lines disconnection (outages z) are affecting this topology. We generate 500 different grid states changing the injections $\boldsymbol{x}$ of the initial grid provided in Matpower.

On these 500 cases, we then computed, still using the high-end simulator Hades2, the full "N-1" (making $500 \times 186 = 93\ 000$ load flow computations). Among this dataset, 75% have been used for training our model, and the rest (25%) for finding the best architecture and meta-parameters (learning rate, number of units per layer, number of layers, etc.) for the neural networks. Note that to be able to estimate the overall generalization of our method, we don't train our neural network on double outages.

For the calibration dataset, we simulate 100 different grid states x and the full "N-1" and "N-2" for all of these simulations. The test set is also composed of 100 different grid states, and their full "N-1" **and "N-2"** (*i.e.* we simulate all single and double lines disconnections). The grid states in the test set are different from the one of the calibration set and the one in the training/validation set and have never been seen during either training or meta parameters estimation. We also want to emphasize that the distributions of the test set (representing the data the network will be tested on) and the distribution of the training set (data available for training the model, corresponding to what operators do today) are different: the test set is composed of single and double outages whereas the training has only single outages. This test set is similar to the Super Test set of the previous experiments on the flows computations.

Outages ranking In this paragraph, we report the errors on the problem of ranking the outages. We compare 4 different models.

- **Operators** The first one consists in the approach taken today by operators: they evaluate first all single outage, and then all the double outages.
- **G. Dropout** $_{\textbf{trained n-1 only}}$ The second model is the guided dropout trained with the dataset described in the previous paragraph.

- **G. Dropout** We use the same architecture (with the same meta parameters) but this time we also trained them on some double **s** (5000 sampled uniformly among the 17 205).

- **Ideal** Corresponds to the ideal ranking. This is an oracle, it cannot be computed in practice and represents what we should do if we knew the exact order. Computing it requires to simulate with a real simulator all contingencies, their associated score, and to rank them according to this.

A proper ranking can be evaluated in multiple ways. We try to focus on 3 mains metrics:

- **Gini coeff.** [18] (between 0 and 1, 1 is best) is a measure commonly used in the literature to evaluate ranking and also a measure of inequalities. It evaluates globally how the ranking performs compared to an ideal ranking.

- $\frac{R(\eta=n)}{R_{\max}}$: we do the ratio between the risk we don't assess and the the total risk for the top n outages. Recall that the risk $R(\eta = n)$ is defined at equation 6.16. This error measure is bounded between its Ideal value, that uses all the budget properly, to 1 not a single outages ranked among the top n is dangerous. The lower it is, the better the model performs. It evaluates how well the ranking performs on the top n outages.

- $C(R^*)$ is defined the minimum number η, such that $R(\eta = n)$ is lower or equal to R^*. R^* being the risk taken into account by operators today. The lowest is the best. It measures how well the ranking is doing on the first few outages.

As observed on Table 6.3, the neural network method is always better that the TSO operator strategy, with respect to all metrics. The most promising results of Table 6.3 is in the 3rd row: the neural networks perform as well as the optimal strategy: the most dangerous outages observed are always ranked among the first one, and on the contrary, on the top-ranked outages, there are only the most dangerous. On average, we need to simulate only $3-4$ outages to achieve a residual risk below the risk taken by operators. This is at the sale of this problem a speedup by a factor of more than 50. Given that we scale the ratio of "bad" outages to be representative of the French power grid, we expected speed-up in the same order of magnitude (10 fold improvement) for the French Extra High Voltage powergrid. We can achieve that with both neural networks, even the network trained only on “n-1” data.

As in every ranking problem, there is a tradeoff between the hit rate (detecting true “bad” outages) and the false alarm rate (falsely diagnosing harmless outages as “bad”). From the point of view of ensuring the security of a power grid, the severity of

Table 6.3 **Comparison of methods** on the whole test set (mean over all s considered $\pm$ one standard deviation). Gini coefficient (between 0 and 1, 1 is best), the risk $R(\eta = n)$ for a maximum computational budget $\eta = n$ and computation cost $C(R^*)$ ensuring that the risk $R(\eta = n)$ remains below R^* (in test data). For both $R(\mathscr{V}^*)$ and $C(R^*)$, smallest is best.

	Operators	G. Dropout trained n-1 only	G. Dropout	Ideal
Gini coeff.	0.41 $\pm$0.04	0.52 $\pm$0.04	0.95 $\pm$0.01	1.00
$\frac{R(\mathscr{V}*)}{R_{\max}}$	0.59 $\pm$0.04	0.58 $\pm$0.04	0.46 $\pm$0.03	0.44 $\pm$0.03
$C(R^*)$	186	3 $\pm$2	3 $\pm$2	3 $\pm$2

both types of errors is not symmetric. It is far more important to have a high hit rate than a low false alarm rate. The Gini coefficient does not capture such imbalance but $C(R^*)$ does. What is practically important to ensure the adoption of the method by the power system community is that the risk curve $R(\eta)$ decreases very fast, ensuring that the "hit rate" be high initially. Remarkably, both neural networks rank first the "bad" single outages, then chose among the worst "n-2" outages. The neural networks privileged a high "hit rate" over lowering the "false alarm rate", a risk-averse behavior, which is expected to operate power systems in security.

Risk Estimation For this particular task, our baseline is to use only machine learning to provide an estimation of the total risk. This is a benchmark against our method, that uses a computational budget of $\eta = n$, the number of powerlines in the power grid.

Figure 6.4 presents the risk of the 100 grid states of the test set: the true risk [13] $R_{\max}$ is represented in blue and is computed with the "physical simulator" according to the equation 2.18. In orange, we show the estimated risk $\hat{R}_{\max}$. In operational processes, the true risk $R_{\max}$ is unknown. For clarity in the representation, the 100 test set situations have been sorted in increasing order of $R_{\max}$.

As we can see in Figure 6.4 (orange points), an estimate of the overall risk is possible. Our estimate $\hat{R}_{\max}$ is quite close on average of the total risk $R_{\max}$. The **MAPE** is 8.7%: Globally, we are also able to predict which situations will be the riskiest: the Pearson correlation coefficient between the estimate and the true values is 0.96: there exists almost a linear relationship between the proposed estimate and

[13]This is not available in practice as it would require too many calls to the physical simulator.

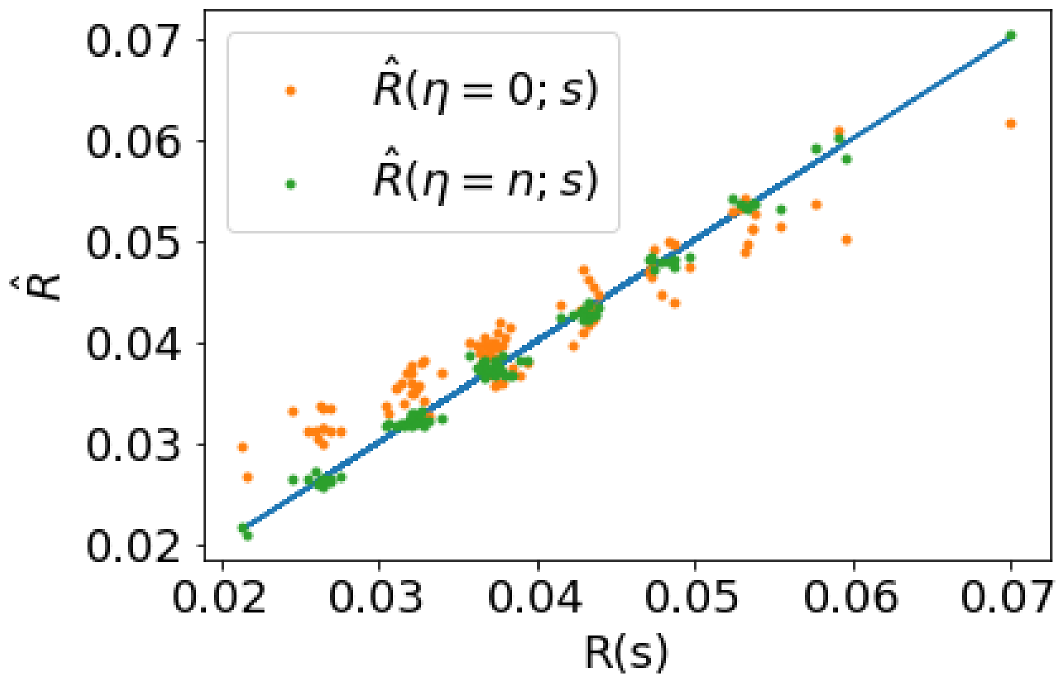


Fig. 6.4 **Comparison of two possible risk evaluation**. In this figure, we compare the true risk (in blue) and and the estimation of the risk (in orange) using : machine learning only (on the left - with the estimator $\hat{R}(0;s)$) and the proposed methodology that relies on a physical simulator (right with the estimator $\hat{R}(\eta = n;s)$ comparable computationally to what operators are doing today). Errors are reported for the 100 grid cases of the test set.

the actual true value. But, for the most interesting cases, where the true risk is the highest (rightmost part of the histogram), the performance decreases. These are the most interesting cases for the TSO but the empirical risk estimation is below the true risk, which can be misleading when making decisions based on this indicator. This estimation of the risk only relies on machine learning. This has limitations as we just exposed. In the next subsection, we will explain how a careful use of a "physical simulator", *e.g.* an algorithm that computes flows based on physical properties of the system (Kirchoff's laws for example) can increase the precision of the estimation of the risk.

Once allowed a computational budget of $\eta = n$, the results for this new estimate of the risk are presented in figure 6.4 (green points). As we can see, there is a significant improvement. Using a slow simulator can drastically help increase the precision of the risk. The MAPE between this new estimate and the real value is 2.5% compares to 8.7% with the machine learning only. The main drawback of this approach is that we only evaluate what happens for a fixed computational budget $\eta = n$. In Figure 6.5

we vary the computational budget η (in x-axis) and report the MAPE between the evaluation of the risk $\hat{R}_{\max}(\eta;s)$ (in y-axis) for the 100 grid states s of the test set.

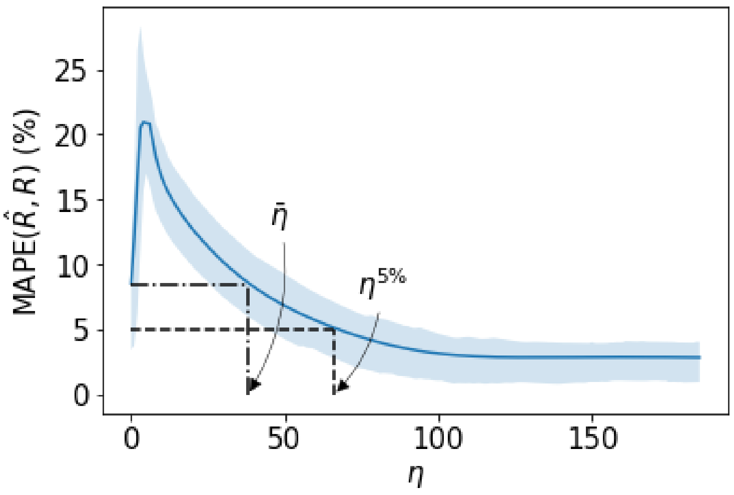


Fig. 6.5 **Precision depending on the computational budget** η. This figure represents the MAPE gap between $\hat{R}_{\max}(\eta;s)$ and $R_{\max}(s)$ for the 100 grid states s of the test set as a function of the number of calls to the physical simulator η. The error bars represent the $[25\% - 75\%]$ confidence interval. As shown in the figure, only $\eta^{5\%} \approx 60$ calls to the physical simulator are needed to have an error in the risk estimation bellow 5%. There is a sharp increase in error when the simulator is called only a few times. The detailed explanation about this phenomenon can be found in Section A.2 of appendix page 121 and following.

Figure 6.5 shows that the error on the risk $\hat{R}_{\max}(\eta;s)$ decreases after a few calls to the simulator when $\eta > 20$. This is not proper to the error[14] used, the same shape is obtained when considering other error measures (such as the **RMSE**). The error is divided by 3 if we compare the error on the $R_{\max}$ and the error on the residual risk after n calls to the physical simulator. This is not surprising: the neural network makes a good job in ranking the outages but seems to struggle in the prediction of the severity score $\psi(z,s)$ especially for the most dangerous outages z. Using the simulator on these cases to asses this score with precisions solves this problem.

6.2.4 Conclusion

In both controlled experiments analyzed in this section, we showed that the guided dropout algorithm could be used to effectively assist operators in its daily tasks.

Our proposed methods rank outages efficiently: identifying first the dangerous single outages. This quick estimation could be used by operators either to spend

[14]In this case the MAPE

more time in studying the most dangerous outages or to make better use of their computational budget by studying other types of outages not systematically considered today (*e.g.* double outages).

We showed that our method allows us to also aggregate the risk induced by all outages and compute how secure our insecure a power grid would be overall. This tools can be used to assess various “would be” scenarios, for example testing whether an action (*e.g.* change of topology) would benefit to the power grid (evaluating the risk before and after a potential application of this action). It could also be used as an alarm in real time: if the risk is higher than a threshold, an action must be performed rapidly to prevent a potential blackout, for example.

Chapter 7

Forward looking work

We presented in the previous section systematic experiments that indicate the potential of our method. To go beyond this initial proof of concept, we conducted a number of experiments on real data with two objectives:

- Demonstrating that our method scales up to grid sizes commensurate with systems of actual interest to RTE.
- Evaluating how close we are from having a solution viable for deployment.
- Identifying directions of future research.

One limitation of experiments in real data is that analyses such as those of the previous chapter relying on controlled experiments, that cannot be replicated stricto-sensu in the real French powergrid. The principal limitation we are facing is the lack of annotations in real data, as detailed in Chapter 4, and particularly annotations concerning the accidental or intentional of changes in grid topology: we only have records of the state of the grid, not what triggered modifications.

In this chapter, we nevertheless attempted to work with real data and go as far as possible to demonstrate our methodology in a realistic setting. In particular, we study two possible ways to apply our methodology directly to real powergrid snapshots:

- We present early work we carried out and aiming at labeling a posteriori the actions that have been taken by operators for security reasons. This work, based on the concept of "counterfactual" requires a lot of computational resources and inspired another Ph.D. thesis at RTE that will combine both machine learning and experts to relabel this dataset.
- We also present early results in the approximation of the flows from the French powergrid snapshots. The methodology developed in the previous chapter to

rank outages and computing is a much better approximation compared to the N-1 security criteria (detailed in Chapter 2). This is an ongoing work at this stage.

This chapter is organized as followed. First, we present work on the labeling the French powergrid. Then we present early results on the application of the guided dropout methodology on part of the French power grid. Finally, we conclude the whole work.

7.1 Real use case: application of the guided dropout on real data

In this section, we use some real data coming from the French extra high voltage power grid to solve a problem that is today interesting for RTE. We apply the guided dropout algorithm to real data coming from the French powergrid. These dataset have been described in Chapter 2 4 pages 43 and following.

We recall that for various reasons detailed in Chapter 4 we decided to study the problem of predicting the security of the powergrid after some lines have been disconnected for maintenance operations. In this section, we report errors on real datasets. These errors are not what really interests TSOs and in particular RTE, that are more interested in predicting whether or not an outage is dangerous. More suited metrics, *e.g.* similar to the one used in Chapter 6 Section 6.2 for example are under study to validate really the method on these data.

As in every application treated in this work, the dataset is composed of $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$ with $\boldsymbol{x}$ being the injections (productions and loads) of the French area studied, $\boldsymbol{y}$ the flows on the power line. $\boldsymbol{\tau}$ is the structural vector encoding whether or not a given powerline is connected or disconnected from the powergrid. On the contrary to what we have done in the controlled experiments, $\boldsymbol{\tau}$ is not the full description of the topology. The baseline is made by switching a residual block instead of a guided dropout block. We emphasize that in this setting, it would not be fair to compare ourselves with the DC approximation. The DC approximation requires the full description of the topology and in these applications, we want to evaluate the performance of a learning machine that does not require this detailed description.

More generally, using real dataset implies some major differences compared to the studies performed in the previous subsections.

- Data is much noisier. This is true whenever using data coming from industrial processes. The powergrid we are studying is evolving, some lines change of ID in the dataset, some physical properties are changing too (for example RTE might decide to add new lines between 2 **substations** or to increase the maximum capacity of some other lines on the grid).

- Because we are studying steady-state powergrids, it is possible to obtain the powerflows given the injections and the topology. This was what we did in the controlled experiments of the previous subsections. Here the problem is more complicated, as we provide only partial information about topology to the neural network. The information is partial because the neural network has no information about which lines is connected to which others.

- The neural network architecture is "blind" from a lot of topological changes. The only topological information provided to the neural network is whether a powerline is absent or present. It has no knowledge about topological changes that not induced by the removal from the grid of a powerline. We hope the neural network can adapt to these changes, that are made by operators either regularly (adapting the topology of the powergrid is a cheap way of handling variations of total demand) or should have low impact on the flows (a test if a breaker is still functioning normally is not performed if it has a risk of endangering the powergrid).

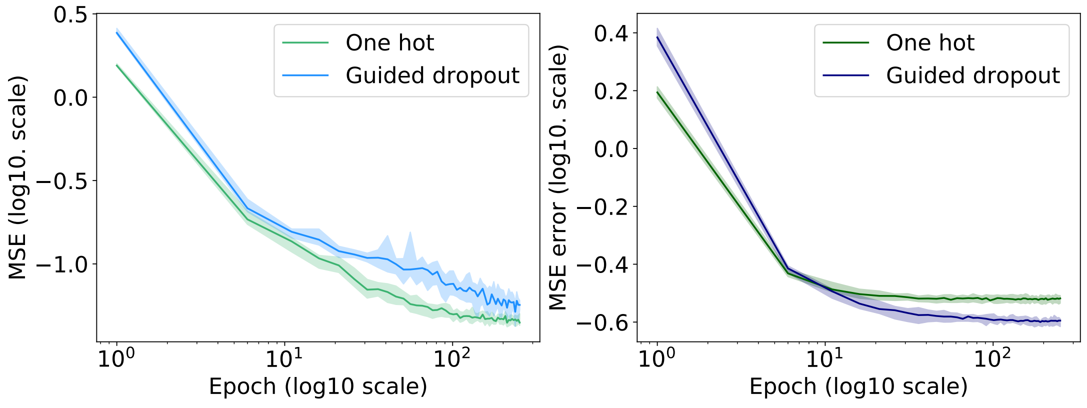
This challenge is assessed using two different experiments. In the first one, we used data coming from the Toulouse area to assess the capability of the guided dropout algorithm to predict flows on historical data, if the training data comes from historical data only. We see that even if the guided dropout performs better than the baseline, the results are not good enough to be used in real time operational processes. On the "Marseilles" area, we augmented the French dataset by disconnecting randomly some powerlines also during training, with the hope of improving the performance of our models.

Toulouse Area

We first start by reporting some error experiments on the Toulouse area, located at the south west of France. For this experiment, we trained a model with raw French powergrid snapshots from January 1st 2012 to May 31st 2017.

We evaluate the performance of our model in two different settings:

- **Test set**: when the test dataset is drawn from a similar distribution as the training dataset. This is done by evaluating error on data withdrawn uniformly at random from the training period. This corresponds to the error where the model can be



(a) Generalization (b) Super-generalization

Fig. 7.1 **Real data from the ultra high voltage power grid.** We show the MSE error in Amperes on a log scale as a function of training epochs. The neural network in both cases is **trained from data until May 2017** with real data. (a) **Regular generalization.** Test set made of randomly sampled data in the same time period as training data. (b) **Super-generalization.** Test set made of the months of June and July 2017.

often retrained, even in real time. Indeed, in a real time process, if a model can be often retrained, the distribution of the data it will be tested on (the few next time steps) is close to the one it is trained on.

- **Super test set**: this dataset is composed of the two following months of data. In this case, the model is asked to predict the flows for the months of June and July 2017. This dataset is a super test as we defined it in this manuscript. In other frameworks,*e.g.* time dependent statistical estimations such as time series forecasting, what we refer to as "super test" in denoted by "test set" more simply. We kept the "super test set" denomination because the topologies seen in this dataset, for the structural variables τ are not the same in training set and in this dataset.

Figure 7.1 displays the learning curves obtained on these real data. We arrive at the same conclusions as in the previous subsection: the guided dropout model is able to learn a similar distribution than the one it is trained on (figure 7.1a).

The guided dropout architecture can also generalize to unseen grid states better than the reference architecture (figure 7.1b), which is a critical property for the application. The baseline architecture performs well on data distributed similarly to training data (figure 7.1a) but does not super-generalize as well as the guided dropout. On this dataset, the MAPE90 error for the guided dropout is on the order of $17-18\%$ (approximately 25% for the baseline one hot architecture), which is higher than what operators would consider as acceptable. This poor performance can be explained: in the test dataset, some lines that are connected in the test set have never been observed disconnected in the training set. The guided dropout algorithm doesn't know how to handle it.

In the next subsection, we report errors for another area of France where this problem have been tackled by randomly disconnecting lines in the training also. This process adding simulated data in the training is similar to what people are doing in other fields (such as natural language processing or computer vision) and is often called data augmentation, as detailed in [39] in the application context of image classification for example.

Marseilles area

This real power grid is composed of 377 loads, 75 productions and 260 transmission lines. In this experiment, we aim at assessing the ability of the guided dropout to have sufficient performance on a more complex industrial problem. Moreover, $\boldsymbol{\tau}$ is here a very high dimensional binary vector (of dimension 260), and there is very little control over the values taken by τ in the Train and Test sets.

The data generating process is the following. In the training set, $10,000$ sets of power production, consumption (x) and partial grid topology (τ) are taken uniformly at random among the powergrid snapshots of 2017 and January and February 2018. We generated 40 more tuples by randomly disconnecting one line in τ and this 40 times. We then have 410,000 tuples (x,τ). This is done so as to increase the probability for each line to be disconnected at least one time in the Train set. Otherwise, many lines would always be observed as closed during training.

The test set is composed of 1000 drawn also uniformly at random in these same periods of time. No extra powerlines are disconnected in the test set. Grid states on the test set have never been observed in the training set.

In the Supertest set, the tuples (x,τ) are sampled from the month of March 2018 (which is never observed in the training set). But then the same procedure is applied, except that there is no forced line disconnection.

Every configuration of τ present in the Test set were previously observed in the Train set. In the meantime, we know that there are configurations of τ present in the Supertest set that were never observed in the Train set: data in the test dataset are not drawn with the same distribution as the one in the training dataset.

The metric used to assess the performance of our models is the **MAPE** for the 90% (denoted by MAPE90) of largest flow in Amps (A) for each line in absolute value. This metric insists on the error for the largest values for each line, which reflects the will

	Test (MAPE90)	Supertest (MAPE90)
One hot	$4.8\% \pm 0.1\%$	$17.3\% \pm 0.5\%$
Guided dropout	$\mathbf{4.2\% \pm 0.1\%}$	$\mathbf{14.2\% \pm 0.4\%}$

Table 7.1 **Results of experience on the Marseilles area**. We present the average (±standard deviation) of the MAPE90 (**MAPE** compute on the 10% highest true value for each power line) for two models: the baseline and the guided dropout. Both model are trained on 10 000 grid states, uniformly at random sampled among year 2017 and the 2 first months of 2018. A process of data augmentation has been used (random line disconnection in this training set). Error are reported for the Test set (1 000 grid cases drawn uniformly at random for the same period as the training set) and Supertest (1 000 drawn uniformely at random for the month of March 2018).

to assess the security of the system in extreme conditions. It is a lot more important to be accurate for flows that are large, than for flows that are almost 0. Moreover, high power flows also mean a more important non-linearity of the system. For this application, the MAPE90 is below than 5% for all the learned models (whether guided dropout or the one hot architecture). On this dataset the performance of the guided dropout and the baseline are similar. These models have been fully trained (the error is still decreasing) this can be seen as a requirement for a real application derived from these experiments: each of this models has been trained for less than 24 hours.

Table 7.1 shows the average performance for both the Test and Supertest and the standard deviation. One can observe that the guided dropout net consistently has a better accuracy for both cases. The difference is flagrant on the Supertest, as the MAPE90 is approximately 3 points lower for the guided dropout net than for the baselines, which is approximately a 17% gain in accuracy.

Compare to the experiment without data augmentation of the first part, this error is lower (drop from 17% to 14% here). But data augmentation alone is not enough to use such a method for real-time operations support: the error remains too high.

Conclusion on this preliminary work

In previous experiments, we studied how the developed methodology was working on a real dataset. Experiments lead to two major conclusions:

- The guided dropout performs better than the baseline in most interesting use case
- Errors on the Test set are reasonable[1], but the errors on the Super Test set are too high.

[1]Today, operators take most of the time some margin, and these margins are approximately 5% of the thermal limits. Having 5% error could then be considered as a "reasonable" error.

These preliminary results are encouraging in many different ways.

First, in the controlled experiments the gap between test error and super test error did not increase too much. We hope that with a better understanding of the guided dropout methodology, we could get the same property when using real dataset. Work is being pursued at this time to better understand the properties of the guided dropout, and we hope, with a new Ph.D. student to be able to publish a journal paper on the guided dropout.

Secondly, the experimental setup was not really representative of a possible use in real time. Training neural network from scratch, in this case, took approximately a few hours (12 to 24 hours), and the Super Test period last for 1 or 2 months. This might seems a bit extreme, and the neural network could be re-learn every week for example. Also, neural networks have the interesting property: they can be "fine-tuned" [53]. This process consists in reusing previously trained neural network weights as a starting point to train another one on a different dataset. This is widely used in the neural network community for image processing and could be used in this setting as well. First, a neural network could be trained using the whole historical dataset, then, each night, the model would be retrained using the most recently acquired data. That way, the distribution of the data the neural network is tested will be closer to the distribution it has been trained on. We could expect to get closer to the error on the Test set that way.

Finally, a framework closer to the one developed in Section 6.2 of Chapter 6 could be adopted. In fact, for TSOs the most important property is to be able to predict accurately whether or not an outage will push the grid outside of its security domain. The fact that flows are not predicted accurately enough doesn't necessarily mean the method will perform poorly on this detection. Assessing whether or not all double outages (two powerlines disconnecting from the grid) is not possible for tools used by TSOs as it would require too much computation. If the developed method could allow to do it, even partially, that could still be used by TSOs. Work in this direction is being performed at the moment.

7.2 Conclusion

In this section, we detailed how the guided dropout methodology can be used to predict flows from the French powergrid snapshots. Compare to the systematic experiments on simulated datasets, this problem is much harder, especially because only partial information is provided to the neural network: the neural network is blind of some topological changes, *e.g.* changes that are independent of the presence / absence of a power line.

On these more complicated cases, the guided dropout model performs better than the neural network baseline. Compared to most models in use today, this allows TSO to approximate flows without a complete information about the powergrid topology. This model achieves relatively good performances on data similar to which they have been trained on. This is a major result: guided dropout can be used for grid operations.

It has still struggle to approximate the flows on completely unseen topologies[2]. This denotes more a problem on the evaluation metric. In fact, TSOs are not really interested in approximating flows, they want to know if the powergrid will be in security or not. More work on this aspect could be carried out. Defining more precisely and computing this evaluation metric will be the focus of future studies.

Another way to improve the performance on these new topologies would be to retrain the neural network more often. A period of more than a month has been used to test the algorithm, algorithm that can be learned in a few hours, and that has the property of being “fine-tuned”. Another method to improve the results would be to learn not only from realized snapshots but on powergrid studied by the operators for example.

[2]The topologies at the grid level in the test set are not observed in the training set. On the contrary, most of the topologies at each substations are not observed in the training set, but not all necessarily.

Chapter 8

Conclusion

The main objective of this doctoral thesis was to develop and study a method to help TSOs take better decisions in a changing environment. Power grids are evolving, in an environment facing unprecedented transformations with the introduction of new market rules, new ways in which power is consumed (through the introduction of self-consumption and the emergence of new behaviors, such as electric cars) and produced (power is increasingly produced using renewable intermittent energy sources such as solar and wind), as well as the diminishing public acceptance for new heavy infrastructures. All these factors combined, make power system operations more complex than ever before. Methods currently in TSO's hands might reach their limits with these new challenges. The "renaissance" of deep learning methods, - including breakthroughs in a number of diverse areas from computer vision, to natural language processing or in some games such as Go or poker - offer an opportunity to adapt these methods to power system issues.

In this report, we begin by modeling the ideal operator and demonstrating how machine learning could lead to better decision-making in Chapter 2. In Chapter 3, we explore briefly some methods used within the machine learning community, principally neural networks.

Chapter 4 details the data available at RTE for this work. We also discussed the fact that this dataset was not labeled (only the actions are recorded while their causes and consequences remain unknown) and explained what it was necessary to build an artificial dataset to perform some controlled experiments that allow a validation of the developed methodology. Compare to what people generally do in machine learning, our dataset can be formalized as $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{y})$ with $\boldsymbol{x}$ being the value of the injections, $\boldsymbol{y}$ a vector representing the flows on the power line. Both these vectors are continuous variables. The specificity of our problem is that the mapping between $\boldsymbol{x}$ and $\boldsymbol{y}$ is parametrized by the vector $\boldsymbol{\tau}$. This structural vector $\boldsymbol{\tau}$ represents the information available for the topology of the powergrid.

In Chapter 5 the main model developed in this work is described. It consists in adapting the architecture of a neural network depending on the structural vector $\boldsymbol{\tau}$ rather than to adding this vector as an input which is the case in model present in the literature. This way of encoding the topology forces the neural network to learn pertinent representations allowing it to generalize to unseen structural vector $\boldsymbol{\tau}$. In this chapter, we also mathematically show, in a simplified setting, that this method was indeed capable of extrapolating knowledge to cases where $\boldsymbol{\tau}$ is not observed.

Finally in Chapter 6, we report some results of the proposed methodology and compare it to different baselines. The first consists of the "DC approximation" (a linearization of the power flow equations used in the power system community) and a neural network with topology added as an input. In all the settings studied, we showed that the guided dropout methodology outperformed both these baselines. This chapter also elaborates on two possible uses of machine learning to help operators: by ranking outages by severity and by assessing in real time how sensitive a powergrid is with respect to some probabilities of power line failure.

In this work, we eventually opened the path to a potential widespread use of deep learning methods in power system applications. This work has perspectives in multiple areas, as detailed in Chapter 7. First, some work can be carried out to label the French dataset and allow its use to check in more detail the validity of any machine learning methodology. We proposed a first possible approach in this labeling task relying solely on machine learning. This project of labeling will be pursued by another Ph.D. candidates at RTE, as her work focuses on using hybrid methods for this task: using both machine learning and expert operators for this difficult task.

Another way to continue this work would be to evaluate our models based on metrics that takes into account in a more precise fashion the properties sought by TSO's today. and to validate in a more systematic way the use of machine learning methods on real datasets coming from the French powergrid snapshots. This work is an ongoing process, and with another Ph.D. candidate recently hired by RTE, we hope to be able to publish some results on this method in a journal paper. This paper will also be the occasion to test the robustness of our methods in problems not necessarily related to power systems.

In this work, we proved, at least in controlled experiments, that machine learning could allow TSOs to predict flows accurately on snapshots, even in unseen cases. We focused on assessing whether an outage would endanger a powergrid, or whether an action proposed by operators could "cure" a power grid. One of the natural extensions of this assessment would be to propose relevant actions to operators in real time along with a time horizon. This work will be pursued by a Ph.D. candidate at RTE whose thesis will focus on “reinforcement learning”.

We envision several possible directions for future research. First, within the realm of power system applications, our proposed architecture could be used to learn to directly propose curative or preventive actions, trained from dispatcher historical actions. Having a neural network capable of mimicking dispatchers is a first step towards fully automating the control of grid operations, possibly refined by reinforcement learning. Secondly, outside of the field of power systems, our architecture may find applications for problems having a mix of continuous and discrete input variables.

References

[1] Alsac, O. and Stott, B. (1974). Optimal load flow with steady-state security. *IEEE transactions on power apparatus and systems*, (3):745–751.

[2] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[3] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[4] Bengio, Y. and et al. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*.

[5] Brown, N. and Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424.

[6] Capitanescu, F., Ramos, J. M., Panciatici, P., Kirschen, D., Marcolini, A. M., Platbrood, L., and Wehenkel, L. (2011). State-of-the-art, challenges, and future trends in security constrained optimal power flow. *Electric Power Systems Research*, 81(8):1731–1741.

[7] Carpentier, J. (1962). Contribution a l'etude du dispatching economique. *Bulletin de la Societe Francaise des Electriciens*, 3(1):431–447.

[8] Chiappa, S., Racaniere, S., Wierstra, D., and Mohamed, S. (2017). Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*.

[9] Dalal, G., Gilboa, E., and Mannor, S. (2016). Hierarchical decision making in electricity grid management. In *International Conference on Machine Learning*, pages 2197–2206.

[10] Donnot, B. and et al. (2017). Introducing machine learning for power system operation support. In *IREP Symposium*, Espinho, Portugal.

[11] Donnot, B., Guyon, I., Liu, Z., Schoenauer, M., Marot, A., and Panciatici, P. (2018a). Latent Surgical Interventions in Residual Neural Networks. working paper or preprint.

[12] Donnot, B., Guyon, I., Marot, A., Schoenauer, M., and Panciatici, P. (2018b). Optimization of computational budget for power system risk assessment. working paper or preprint.

[13] Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018c). Anticipating contingengies in power grids using fast neural net screening. In *IEEE WCCI 2018*, Rio de Janeiro, Brazil.

[14] Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018d). Fast power system security analysis with guided dropout.

[15] Duchesne, L., Karangelos, E., and Wehenkel, L. (2017). Machine learning of real-time power systems reliability management response. In *2017 IEEE Manchester PowerTech*. IEEE.

[16] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:.

[17] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[18] Gini, C. (1921). Measurement of inequality of incomes. *The Economic Journal*, 31(121):124–126.

[19] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[20] Grigg, C., Wong, P., Albrecht, P., Allan, R., Bhavaraju, M., Billinton, R., Chen, Q., Fong, C., Haddad, S., Kuruganty, S., et al. (1999). The ieee reliability test system-1996. a report prepared by the reliability test system task force of the application of probability methods subcommittee. *IEEE Transactions on power systems*, 14(3):1010–1020.

[21] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[22] Hines, P. D. H., Dobson, I., and Rezaei, P. (2015). Cascading power outages propagate locally in an influence graph that is not the actual grid topology.

[23] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[24] Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

[25] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

[26] Karangelos, E. and Wehenkel, L. (2016). Probabilistic reliability management approach and criteria for power system real-time operation. In *Power Systems Computation Conference (PSCC), 2016*, pages 1–9. IEEE.

[27] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

[28] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[29] Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

[30] Kundur, P., Balu, N. J., and Lauby, M. G. (1994). *Power system stability and control*, volume 7. McGraw-hill New York.

[31] Kundur, P. S. (2012). Power system stability. In *Power System Stability and Control, Third Edition*, pages 1–12. CRC Press.

[32] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[33] Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*.

[34] Milano, F., Cañizares, C. A., and Conejo, A. J. (2005). Sensitivity-based security-constrained opf market clearing model. *IEEE Transactions on power systems*, 20(4):2051–2060.

[35] Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

[36] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

[37] Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871.

[38] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

[39] Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

[40] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[41] Ruiz, P. A., Rudkevich, A., Caramanis, M. C., Goldis, E., Ntakou, E., and Philbrick, C. R. (2012). Reduced mip formulation for transmission topology control. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1073–1079. IEEE.

[42] Saeh, I. and Khairuddin, A. (2008). Static security assessment using artificial neural network. In *Power and Energy Conference, 2008. PECon 2008. IEEE 2nd International*, pages 1172–1178. IEEE.

[43] Shazeer, N. and et al. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*.

[44] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[45] Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. Y. (2013). Zero-shot learning through cross-modal transfer. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 935–943, USA. Curran Associates Inc.

[46] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[47] Stott, B. and Alsac, O. (1974). Fast decoupled load flow. *IEEE transactions on power apparatus and systems*, (3):859–869.

[48] Sunitha, R., Kumar, S. K., and Mathew, A. T. (2013). Online static security assessment module using artificial neural networks. *IEEE Transactions on Power Systems*, 28(4):4328–4335.

[49] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

[50] Trias, A. (2012). The holomorphic embedding load flow method. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE.

[51] Wehenkel, L. (1997). Machine learning approaches to power-system security assessment. *IEEE Expert*, 12(5):60–72.

[52] Wehenkel, L. A. (2012). *Automatic learning techniques in power systems*. Springer Science & Business Media.

[53] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

[54] Young, T., Hazarika, D., Poria, S., and Cambria, E. (2017). Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*.

[55] Zaoui, F., Fliscounakis, S., and Gonzalez, R. (2005). Coupling opf and topology optimization for security purposes. In *15th Power Systems Computation Conference*, pages 22–26.

[56] Zimmerman, R. D. and et al. (2011). Matpower. *IEEE Trans. on Power Systems*, pages 12–19.

[57] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

[58] Zou, M., Shi, Y., Wang, C., Li, F., Song, W., and Wang, Y. (2018). Potrojan: powerful neural-level trojan designs in deep learning models. *arXiv preprint arXiv:1802.03043*.

Appendix A

Appendix

A.1 Bus splitting, impact of topology on power flow equations

We will illustrate this changes through an example. Let's consider the figures 2.1a and 2.2b. Both powergrids are identical, except that in the second one, compared to the first 1, the substation 1 is split into 2 buses. For clarity, we report only the active part of the equations 2.9, only for the buses in the first substation and suppose that for every power lines in this network $b_{k\to j} \equiv b$ and $g_{k\to j} \equiv g$, b and g being constants. Let's also suppose that, in both cases c_1 withdraw a amount of active (resp. reactive) power $-P$ (resp. $-Q$) from the power grid, and that the production p_2 inject an active power P' and has a voltage magnitude setpoint $|V|$. Lastly, all the substations that are not the substation 1 are operated with one single bus. We will name these buses accordingly: the bus number 2 will be the bus in the second substation, the bus number 3 the bus in the third substation etc.

In Figure 2.1a, everything is interconnected at substation 1: it counts only 1 bus, that we will name C. It has a production connected to it, os it is a "PV" node according to the nomenclature of Table 2.1, and the unknowns are q_C and θ_C. For this bus C, the equations are:

$$\begin{cases} 0 & = -p_C \sum_{j=1}^{m} |v_C||v_j| \left(g.\cos(\theta_C - \theta_j) + b\sin(\theta_C - \theta_j)\right) \end{cases} \tag{A.1}$$

But at this buses, the power injected p_C is $p_C = P' - P$. Finally, we now that $|v_C| = |V|$, the setpoint voltage magnitude of the production p_2 connected to it. Moreover, we know, from Figure 2.1a, that this bus C is connected to bus 2 through line 2, to bus 3 through line 2, to bus 4 through line 3 and to bus 5 through line 4. This gives:

$$\begin{cases} (P'-P) &= |v||v_2|\left(g.\cos(\theta_C-\theta_2)+b\sin(\theta_C-\theta_2)\right) \quad [\rightarrow \text{bus 2 through power line 1}] \\ & \quad +|v||v_3|\left(g.\cos(\theta_C-\theta_3)+b\sin(\theta_C-\theta_3)\right) \quad [\rightarrow \text{bus 3 through power line 2}] \\ & \quad +|v||v_4|\left(g.\cos(\theta_C-\theta_4)+b\sin(\theta_C-\theta_4)\right) \quad [\rightarrow \text{bus 4 through power line 3}] \\ & \quad +|v||v_5|\left(g.\cos(\theta_C-\theta_4)+b\sin(\theta_C-\theta_4)\right) \quad [\rightarrow \text{bus 5 through power line 4}] \end{cases} \tag{A.2}$$

Now let's derive the equations for the configuration in Figure 2.2b. In this figure, we can see there are now 2 buses in this substation. Let's note them C' and D'. C' denotes the bus 1 where production p_1 and power lines l_3 and l_4 are connected (blue bus) and D' represents the bus where c_1 and power lines l_1 and l_2 are connected (orange bus). The equations become:

$$\begin{cases} 0 &= -p_{C'} \sum_{j=1}^{m} |v_{C'}||v_j|\left(g.\cos(\theta_{C'}-\theta_j)+b\sin(\theta_{C'}-\theta_j)\right) \quad \text{for bus } C' \end{cases} \tag{A.3}$$

$$\begin{cases} 0 &= -p_{D'} \sum_{j=1}^{m} |v_{D'}||v_j|\left(g.\cos(\theta_{D'}-\theta_j)+b\sin(\theta_{D'}-\theta_j)\right) \quad \text{for bus } D' \end{cases} \tag{A.4}$$

We now a production is connected to bus C'. This implies that $|v_{C'}| = |V|$ the setpoint value of the production, and that $p_{c'} = P'$, since the only object connected to this node is the production p_2. We also know that bus C' is connected to bus 2 via line l_1 and bus 3 via line l_2. In the same manner, we have $p_{D'} = P$ and C' is connected to bus 4 via line l_3 and to bus 5 via line l_4. The two equations above then becomes:

$$\begin{cases} P' &= |v||v_2|\left(g.\cos(\theta_{C'}-\theta_2)+b\sin(\theta_{C'}-\theta_2)\right) \quad [\rightarrow \text{bus 2 through power line 1}] \\ & \quad +|v||v_3|\left(g.\cos(\theta_{C'}-\theta_3)+b\sin(\theta_{C'}-\theta_3)\right) \quad [\rightarrow \text{bus 3 through power line 2}] \end{cases} \tag{A.5}$$

$$\begin{cases} -P &= |v_{D'}||v_4|\left(g.\cos(\theta_{D'}-\theta_4)+b\sin(\theta_{D'}-\theta_4)\right) \quad (\rightarrow \text{bus 4 through power line 3}] \\ & \quad +|v_{D'}||v_5|\left(g.\cos(\theta_{D'}-\theta_4)+b\sin(\theta_{D'}-\theta_4)\right) \quad [\rightarrow \text{bus 5 through power line 4}] \end{cases} \tag{A.6}$$

The table A.1 sum up all these equations, obtained for substation 1, and for the active part only[1]. As we can, this change in topology of substation 1 has multiple impact. Firstly there are more variables in the second case (when there are 2 nodes in this substation) because there are more buses in the power grid. Secondly there is only 1 equation that describe the kirchoff's law in this substation in the first case (left part of the Table A.1) and 2 equations when there are 2 buses. The main thing to remember

[1]The reactive part

is that a change in a substation topology deeply changes the equations describing the grid and can lead to very different solutions.

Table A.1 Transformation of the load flow equations related to the first substation of the power grid showed in Figure 2.1a when the topology changed in the one described in Figure. 2.2b. Only the active part are showed here, and only for the buses of the first substation.

Power grid configuration of				
Figure 2.1a		Figure 2.2b		
$(P'-P)$	$= \lvert v\rvert\lvert v_2\rvert(g.\cos(\theta_C-\theta_2)+b\sin(\theta_C-\theta_2))$	P'	$= \lvert v\rvert\lvert v_2\rvert(g.\cos(\theta_{C'}-\theta_2)+b\sin(\theta_{C'}-\theta_2))$	$\rightarrow$ bus 2 through line 1
	$+\lvert v\rvert\lvert v_3\rvert(g.\cos(\theta_C-\theta_3)+b\sin(\theta_C-\theta_3))$		$+\lvert v\rvert\lvert v_3\rvert(g.\cos(\theta_{C'}-\theta_3)+b\sin(\theta_{C'}-\theta_3))$	$\rightarrow$ bus 3 through line 2
	$+\lvert v\rvert\lvert v_4\rvert(g.\cos(\theta_C-\theta_4)+b\sin(\theta_C-\theta_4))$	$-P$	$= \lvert v_{D'}\rvert\lvert v_4\rvert(g.\cos(\theta_{D'}-\theta_4)+b\sin(\theta_{D'}-\theta_4))$	$\rightarrow$ bus 4 through line 3
	$+\lvert v\rvert\lvert v_5\rvert(g.\cos(\theta_C-\theta_4)+b\sin(\theta_C-\theta_4))$		$+\lvert v_{D'}\rvert\lvert v_5\rvert(g.\cos(\theta_{D'}-\theta_4)+b\sin(\theta_{D'}-\theta_4))$	$\rightarrow$ bus 5 through line 4

A.2 Machine learning and risk estimation

In the related section of the manuscript, showed in Chapter 6, Figure 6.5 (recalled hereafter for clarity), there were a phenomenon that has not been explained in the main manuscript: before decreasing the error on the estimation of the risk $\hat{R}$ and the true risk R increases.

In this section, we give an explanation about this phenomenon. In fact, this is due to the structure of the risk considered. We recall the risk of a powegrid, and the estimator of the risk used in the section:

$$R(s) = \sum_{z\in\mathscr{A}} p(z).L(z;s) \tag{A.7}$$

$$\hat{R}(\eta;s) = \underbrace{\sum_{z\in\mathscr{V}_\eta} p(z).L(z;s)}_{\text{Evaluate the most dangerous with the simulator}} + \underbrace{\sum_{z\notin\mathscr{V}_\eta} p(z).\hat{L}(z;s)}_{\text{Rely on machine learning only for the least dangerous one}} \tag{A.8}$$

where $\mathscr{A}$ is the set of all considered outages, in this case all single and double outages, $p(z)$ is the probability of occurrence of outage z, and $L(z;s)$ is the cost associated to this contingency when power grid is in state s. Recall that the methodology for this estimation was to first rank the contingencies, and then group the η first contingencies in a set denoted by $\mathscr{V}_\eta$.

We can explain this phenomenon of increasing error in two possible ways. The first one is that the estimated loss is ill predicted with the neural network. The estimator

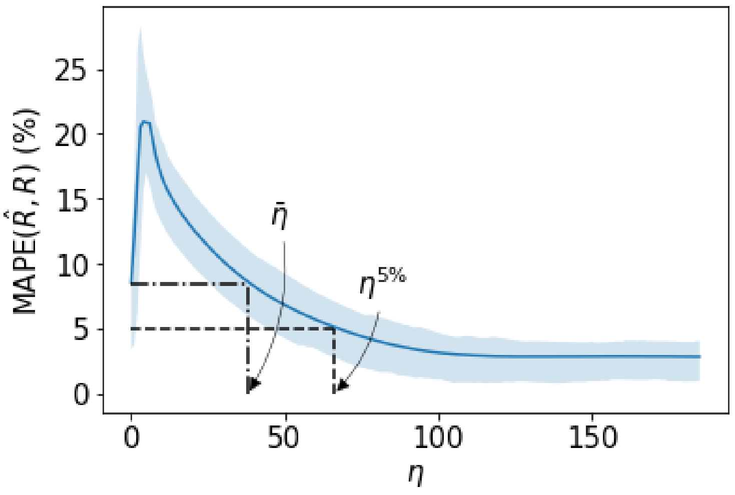


Fig. A.1 **Precision depending on the computational budget** η This figures represents the MAPE between $\hat{R}_{\max}(\eta;s)$ and $R_{\max}(s)$ for the 100 grid states s of the test set as function of the number of calls to the physical simulator η. The error bars represent the $[25\% - 75\%]$ confidence interval. As showed in the figure, only $\eta^{5\%} \approx 60$ calls to the physical simulator are needed to have an error in the risk estimation bellow 5%. We show an increasing of the error when the simulator is called only a few times. The detailed explanation about this phenomenon can be found in Section A.2 of appendix page 121 and following.

$\hat{R}(\eta = 0;s)$ is however unbiased. So replacing ill predicting values with correct values introduces a bias, that leads to an higher error. An example that illustrates this phenomenon is detailed in Table A.2.

Table A.2 **Illustration of the bias introduced in the estimation of the risk** by simulating only the most dangerous outages. In this small example, $R = 1+2+1+5+3+4+6+7+6 = 35$, $\hat{R}(\eta = 0) = 1.2+2.3+1.2+4.7+3.7+4.2+\mathbf{5.5}+\mathbf{6.5}+5.3 = 34.6$, and if the top 2 contingencies, bold faced in the table bellow, that are evaluated with the true simulator this gives $\hat{R}(\eta = 2;s) = 0.8+2.3+1.2+4.7+3.7+4.2+\mathbf{6}+\mathbf{7}+5.3 = 35.6$. So we in fact have $\left|R - \hat{R}(\eta = 0)\right| < \left|R - \hat{R}(\eta = 2)\right|$ which explains the phenomenon observed.

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9
True loss $p(z).L(z;s)$	1	2	1	5	3	4	**6**	**7**	6
Estimated loss $p(z).\hat{L}(z;s)$	1.2	2.3	1.2	4.7	3.7	4.2	**5.5**	**6.5**	5.3

Titre: Méthode d'apprentissage profond (deep learning) pour prévoir les flux dans les réseaux de transports d'électricité: nouvelles architectures et algorithmes.

Mots clés: apprentissage, super grid, optimisation

Résumé: Cette thèse porte sur les problèmes de sécurité sur le réseau électrique français exploité par RTE, le Gestionnaire de Réseau de Transport (GRT). Les progrès en matière d'énergie durable, d'efficacité du marché de l'électricité ou de nouveaux modes de consommation poussent les GRT à exploiter le réseau plus près de ses limites de sécurité. Pour ce faire, il est essentiel de rendre le réseau plus "intelligent". Pour s'attaquer à ce problème, ce travail explore les avantages des réseaux neuronaux artificiels.

Nous proposons de nouveaux algorithmes et architectures d'apprentissage profond pour aider les opérateurs humains (dispatcheurs) à prendre des décisions que nous appelons " guided dropout ". Ceci permet de prévoir les flux électriques consécutifs à une modification volontaire ou accidentelle du réseau. Pour se faire, les données continues (productions et consommations) sont introduites de manière standard, via une couche d'entrée au réseau neuronal, tandis que les données discrètes (topologies du réseau électrique) sont encodées directement dans l'architecture réseau neuronal. L'architecture est modifiée dynamiquement en fonction de la topologie du réseau électrique en activant ou désactivant des unités cachées. Le principal avantage de cette technique réside dans sa capacité à prédire les flux même pour des topologies de réseau inédites. Le "guided dropout" atteint une précision élevée (jusqu'à 99% de précision pour les prévisions de débit) tout en allant 300 fois plus vite que des simulateurs de grille physiques basés sur les lois de Kirchoff, même pour des topologies jamais vues, sans connaissance détaillée de la structure de la grille. Nous avons également montré que le "guided dropout" peut être utilisé pour classer par ordre de gravité des évènements pouvant survenir. Dans cette application, nous avons démontré que notre algorithme permet d'obtenir le même risque que les politiques actuellement mises en œuvre tout en n'exigeant que 2% du budget informatique. Le classement reste pertinent, même pour des cas de réseau jamais vus auparavant, et peut être utilisé pour avoir une estimation globale de la sécurité globale du réseau électrique.

Title: Deep Learning Methods for Predicting Flows in Power Grids: Novel Architectures and Algorithms

Keywords: deep learning, super grid, optimization

Abstract: This thesis addresses problems of security in the French grid operated by RTE, the French “Transmission System Operator” (TSO). Progress in sustainable energy, electricity market efficiency, or novel consumption patterns push TSO's to operate the grid closer to its security limits. To this end, it is essential to make the grid “smarter”. To tackle this issue, this work explores the benefits of artificial neural networks.

We propose novel deep learning algorithms and architectures to assist the decisions of human operators (TSO dispatchers) that we called “guided dropout”. This allows the predictions on power flows following of a grid willful or accidental modification. This is tackled by separating the different inputs: continuous data (productions and consumptions) are introduced in a standard way, via a neural network input layer while discrete data (grid topologies) are encoded directly in the neural network architecture. This architecture is dynamically modified based on the power grid topology by switching on or off hidden units activation's.

The main advantage of this technique lies in its ability to predict the flows even for previously unseen grid topologies. The "guided dropout" achieves a high accuracy (up to 99% of precision for flow predictions) with a 300 times speedup compared to physical grid simulators based on Kirchoff's laws even for unseen contingencies, without detailed knowledge of the grid structure. We also showed that guided dropout can be used to rank contingencies that might occur in the order of severity. In this application, we demonstrated that our algorithm obtains the same risk as currently implemented policies while requiring only 2% of today's computational budget. The ranking remains relevant even handling grid cases never seen before, and can be used to have an overall estimation of the global security of the power grid.