Anticipating contingengies in power grids using fast neural net screening

Benjamin Donnot, Isabelle Guyon, Marc Schoenauer, Antoine Marot, Patrick Panciatici

▶ To cite this version:

HAL Id: hal-01783669

https://hal.science/hal-01783669

Submitted on 3 May 2018

Anticipating contingengies in power grids using fast neural net screening

Benjamin Donnot[‡ †], **Isabelle Guyon**[‡•]**, Marc Schoenauer**[‡]**,**
Antoine Marot[†]**, Patrick Panciatici**[†]

‡ UPSud and Inria TAU, Université Paris-Saclay, France.
• ChaLearn, Berkeley, California. † RTE France.

Abstract—We address the problem of maintaining high voltage power transmission networks in security at all time. This requires that power flowing through all lines remain below a certain nominal thermal limit above which lines might melt, break or cause other damages. Current practices include enforcing the deterministic "N-1" reliability criterion, namely anticipating exceeding of thermal limit for any eventual single line disconnection (whatever its cause may be) by running a slow, but accurate, physical grid simulator. New conceptual frameworks are calling for a probabilistic risk based security criterion and are in need of new methods to assess the risk. To tackle this difficult assessment, we address in this paper the problem of rapidly ranking higher order contingencies including all pairs of line disconnections, to better prioritize simulations. We present a novel method based on neural networks, which ranks "N-1" and "N-2" contingencies in decreasing order of presumed severity. We demonstrate on a classical benchmark problem that the residual risk of contingencies decreases dramatically compared to considering solely all "N-1" cases, at no additional computational cost. We evaluate that our method scales up to power grids of the size of the French high voltage power grid (over $1\,000$ power lines).

I. Introduction

Although our application domain is power systems, the problem that we address is relatively general. It consists in ranking rapidly items/samples/entities, hereby referred to as "events" with a fast "proxy" method, then re-evaluating results with a slower (more accurate) method. The goal is to identify "bad" events, which should be ranked first. The hope is to gain in speed without sacrificing accuracy of the overall ranking or to increase the accuracy of ranking at no additional computational cost. In our power grid setting, an "event" will be "line disconnection(s)" or "line outage(s)" and will be referred to as a "contingency". Today's, ranking entails running a power grid simulator to compute power flow through all lines at a given time (so-called "load flow"). "Bad" events or "dangerous contingencies" correspond to the disconnection of a power line, that put the grid in an unsecure state: the thermal limit of (at least) one other line is exceeded due to an excess of current flowing through it. If no remedial actions are taken quickly, this overloaded line becomes out of service as well. As the grid weakens, the problem propagates like a snow ball effect (which may lead to a black-out). This can typically happen when the power grid is overloaded at peak consumption times and/or one or several lines have been disconnected. In this paper, we will consider that the system is in quasi-stationary

conditions under given static "injections", corresponding to balanced productions and consumptions. This is typical of the conditions experienced by grid operators (dispatchers), which review grid conditions at intervals of five minutes. In our setting, grid events consist in line disconnections. Our novel contribution is to use neural networks to perform a first fast ranking of would-be "bad contingencies", using an architecture we recently published [1].

The problem of ranking contingencies in power system is not new. In [2] for example, the authors studied voltage stability problems after some unplanned line disconnection (contingency) occurs.

Other authors use machine learning to address power system related problems. Most papers address the problem of classifying grid states according to given security criteria ([3], [4], [5], [6]), or predicting how a system will react after an unplanned event occur ([7]). To the best of our knowledge, using neural networks for screening "bad contingencies" has not been done before.

Closest to our work, [8] use Artificial Neural Networks and Restricted Bolztman Machine to predict "Composite security indices", a concept similar to our scores $\hat{s}$ (see section IV), allowing us to rank the contingencies z. A key difference between their work and ours is that we use our neural network to make predictions under multiple contingencies. Our neural network architecture is more efficient and better adapted to the task at hand.

Our method can also be seen as a particular case of "pointwise approach" by the "learning to rank community" ([9]). These models are widely used in documents retrieval ([10]) or medical drug discovery ([11]). To our knowledge, none of the above methods applies directly to contingencies ranking in power system.

II. Statement of the problem and notations

Formally, given a set $\mathcal{Z}$ of candidate events z, our goal is to identify the subset $\mathcal{B}$ of "bad" events by evaluating with a slow simulator the smallest possible number of events z. We run first a fast "proxy" simulator (our neural network), on all events in $\mathcal{Z}$ to obtain a set $\mathcal{V}$ of "at risk" candidates to be verified with the slow simulator. This saves us the effort of checking the remaining events in set $\mathcal{Z} - \mathcal{V}$.

Under our assumptions, a "system state" x consists of the power flowing in all lines, resulting from given (fixed) injec-

tions, for a specific grid topology. We always analyze a situation corresponding to a fixed state in this paper and sometimes omit x for brevity of notation. We also omit to specify time ordering, although states are time ordered. What we denote by $z \in \mathcal{Z}$ are **sudden would-be (potentially disruptive) events**, corresponding to a change in grid topology, such as a **line disconnection**, assuming injections remain constant. In some sense, z is a variation Δx. In the language of power systems, such events are also referred to as **"contingencies"**. Therefore in this paper "event" and "contingency" will be synonymous and "bad event" and "bad contingency" will be synonymous. An contingency z might arise with probability $p(z)$[1] and is associated with a loss function $L(z; x)$. The overall risk is defined as[2]:

$$R_{\max}(x) = \sum_{z \in \mathcal{Z}} p(z) L(z; x) \quad (1)$$

where, in our application context, we assume that $L(z; x)$ is the $\{0, 1\}$ loss, with 0 meaning that the contingency z arising in state x is innocuous and 1 that it is risky or "bad" or "dangerous" for our system (*i.e.* at least one line, still in service after z arose, will exceed its thermal limit)[3]. Thus:

$$L(z; x) = \begin{cases} 0 & \text{"No current flowing on any line} \\ & \text{exceeds the line thermal limit} \\ & \text{under contingency } z \\ & \text{in grid state } x" \Rightarrow \text{OK} \\ 1 & \text{"Otherwise"} \Rightarrow \text{"Bad" event} \end{cases} \quad (2)$$

Because of computational cost, we will only evaluate the subset $\mathcal{V} \subset \mathcal{X}$ of events, which will be identified as "potentially bad" by the fast "proxy" (*i.e.* neural network), using the slow (but accurate) simulator (*i.e.* load flow simulator). We assume that corrective actions are taken for all events found "truly bad" in $\mathcal{V}$ using the slow simulator, bringing the loss to zero for such events.

Thus, the *residual* risk, corresponding to events in $\mathcal{Z} - \mathcal{V}$, which we neglected to evaluate with the slow simulator, is:

$$R(\mathcal{V}; x) = \sum_{z \in \mathcal{Z} - \mathcal{V}} p(z) L(z; x) \quad (3)$$

The residual risk R is bounded between:

$$R(\mathcal{Z}; x) = 0 \quad \text{and} \quad R(\emptyset; x) = R_{\max}(x) \quad (4)$$

The fast proxy provides a total ranking of $z \in \mathcal{Z}$ yielding nested subsets $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \cdots \subset \mathcal{Z}$ including an increasing number of events z, starting from those considered most at risk by the screening method. We can thus study $R(\mathcal{V}_i; x)$

as a function of $i = \text{rank}(z) = |\mathcal{V}_i|$, where $|\cdot|$ denotes the cardinal of a set.

In our simulations, the computational cost of the fast proxy is negligible compared to running the slow simulator. Previous experiments indicate that a $2000\times$ speed-up is achievable[4] Hence the computational cost is proportional to the number of events $|\mathcal{V}|$ identified "potentially at risk" by the neural network that are actually run by the simulator (no offset for the neural network computational cost):

$$\frac{\text{rank}(z)}{|\mathcal{Z}|} = \frac{|\mathcal{V}|}{|\mathcal{Z}|} = \frac{C(\mathcal{V})}{C_{max}} \quad (5)$$

In Figure 1 we schematically represent a hypothetical case to illustrate our notations. It is the opposite of a lift curve used in marketing in which revenue = (1 - risk) is plotted as a function of investment. We consider a given state of the system x and study the effect on the residual risk $R(\mathcal{V}; x)$ of various ranking strategies obtained by various screening methods (*i.e.* what we referred to as fast "proxy"). The horizontal axis represents increasing size nested subsets $\mathcal{V}$ of events z "called bad" by a given screening method. Such events incur a computational cost $C(\mathcal{V})$ to evaluate which of them are "truly bad" (using the slow simulator). To limit costs as much as possible, we want $R(\mathcal{V}; x)$ to decrease fast, *i.e.* we want $\mathcal{V}$ to contain as many "truly bad" events as possible. We represented three cases:

- The **blue line** is the expected value of the risk for a random ranking.
- The **orange curve** is the risk for an "ideal" ranking in which all "bad" events are ranked first.
- The **green curve** is the ranking obtained by a given proposed screening method.

To select between alternative methods, we can either fix a maximum budget C^* and compare $R(\mathcal{V}; x)$ or set a maximum risk R^* and compare $C(\mathcal{V})$.

It is worth noting that estimating $R(\mathcal{V}; x)$ is another difficult aspect of the problem. In this paper, because we use a benchmark of modest size, we exhaustively compute $L(z; x)$ for all z with the slow simulator to draw curves such as those of Figure 1. In practice $R(\mathcal{V}; x)$ might have to be approximated by replacing $L(z; x)$ by an approximate loss $\hat{L}(z; x)$, obtained using power flows estimated by our "proxy" simulator. This yields a risk estimator:

$$\hat{R}(\mathcal{V}; x) = \sum_{z \in \mathcal{Z} - \mathcal{V}} p(z) \hat{L}(z; x) \quad (6)$$

III. ASPECTS SPECIFIC TO THE POWER GRID PROBLEM

Our proposed method relies on previously published work in [14] and [1] in which we devised a neural-network trained with "guided dropout", which allows us to predict

---

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

[2]In the power system community this criterion is similar to the "discarding" principle of the GARPUR methodology (see [12] Eq. 3, for instance).

[3]This is a simplification. The real damage of the grid would endure after contingency z would require computing a full "cascading failure" (as presented in [13] for example), which is computationally too expensive to calculate presently, even for a small test case like ours.

[4]Once the data x are loaded, Hades2, the fast high end simulator used by RTE makes approximately 300 ms to compute a load flow, first experiment on neural network show that a they can perform 5900 load-flow per second, provided that all 6000 grid states are already loaded in memory.
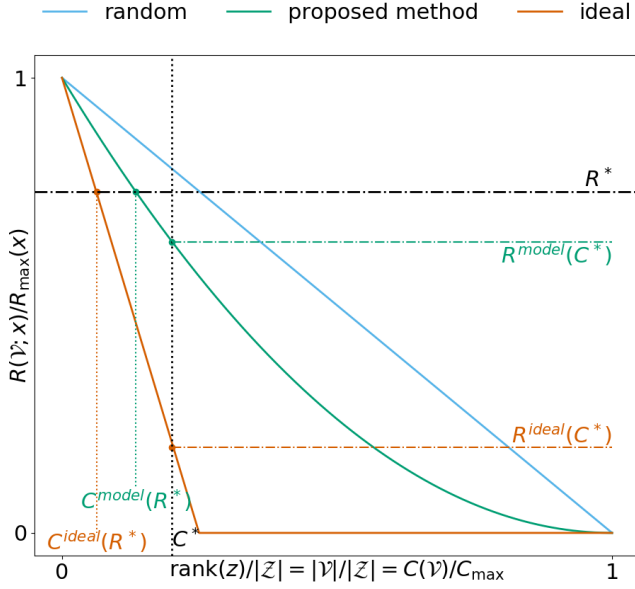
Fig. 1: **Didactic representation of risk as a function of computational cost**. We introduce notations in this schematic plot (not corresponding to actual numerical simulations). The black horizontal dash-dotted line represent R^*: the maximum residual risk we are willing to accept. The Black vertical dotted line represents C^* the maximum computational cost we are willing to pay. We assume for simplifity that all events have the same probablity, which yields staight segments for the blue and orange curves.

power flows in power grids for given topology variants while training only on a small subset of these.

Our new contributions in this paper are (1) to formulate the detection of "bad" events (= dangerous contingencies) as a ranking problem; (2) to use our "guided dropout" network as "proxy" simulator to provide a total ranking of events; and (3) to evaluate the performance of such ranking in several scenarios implied by slow or fast changing environments (distribution of the grid state x). In particular, we examine 2 variants of our "proxy" simulator: one in which training and test data are identically distributed (implying that either the distribution of x does not change over time or that the proxy can be often retrained), and another one where contingencies never seen during training also need to be ranked (testing the ability of the method to deal with novelty in real time, a critical aspect for real time operation in a fast changing environment).

Following the nomenclature of power systems, we recall that events correspond to line outages and are called "contingencies". "Bad" events correspond to contingencies after which at least one power line exceeds its "thermal limit" (a limit on maximum flow essentially set to prevent the line from melting). In this paper, we consider only two kinds of contingencies: "single contingencies", denoted by z_i, representing the disconnection of a single power line, and "double contingencies" $z_{i,j}$ representing the disconnection of two lines.

After the power grid suffers a single contingency, we will say its state is in "n-1" (notice the quotes, this is not an algebraic formula), namely: "it has lost one in "n" lines". If n denotes the number of lines in our power grid, there are exactly n different "n-1" grid states. Similarly, a power grid suffering a double contingency will be referred to as being in a degraded state "n-2". There are exactly $n(n-1)/2$ such "n-2" grid states.

To further elaborate on the specifics of our problem in application to power systems, we explain in the following subsections the security criteria used by TSO's (Transmission System Operator, responsible of the power grid safety) and define a baseline ranking method corresponding to the operational strategy in place in today's French power grid management.
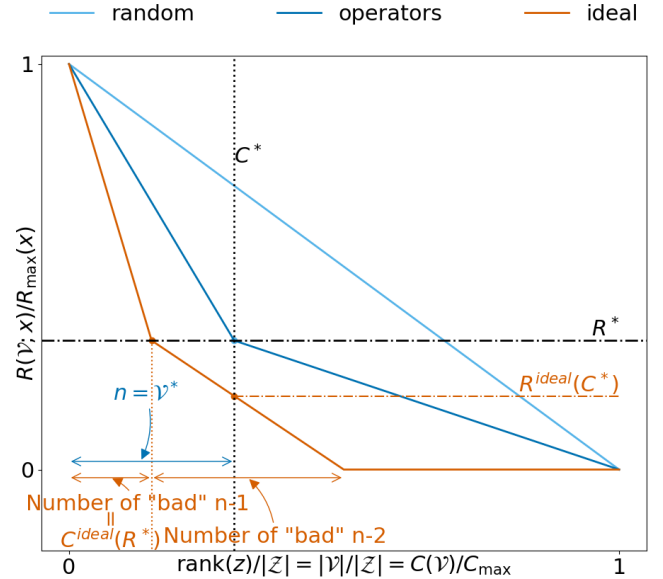
A. Risk modeling

Fig. 2: **Application-specific risk as a function of cost** (didactic representation). We schematically illustrate the random curve (dark blue), baseline curve (light blue) simulating the "n-1" contingencies first, and the optimal curve (orange) simulating first all "bad" events.

In this paper, we focus on assessing the "N-2" security, that is the state of the system ("secure" / "non secure") after each possible "n-2" OR "n-1". Computing this safety would require $n(n+1)/2$ calls to the high end power grid simulator, since n computations are required for all "n-1" contingencies (loss of one line) and $n(n-1)/2$ for all "n-2" contingencies (loss of two lines). In our notations, this means that the set of all events under consideration is:

$$\mathcal{Z} = \text{"N-2"} \quad \text{with} \quad |\mathcal{Z}| = \frac{n(n+1)}{2}. \quad (7)$$

We ignore the effect (and the residual risk) associated with higher order contingencies ("n-3", "n-4", etc.). As a further

simplification, we assume that all single disconnections z_i have equal probability:

$$p(z_i) \stackrel{\text{def}}{=} \pi_{(1)} \tag{8}$$

and all double disconnection $z_{i,j}$ have equal probability:

$$p(z_{i,j}) \stackrel{\text{def}}{=} \pi_{(2)} = \pi_{(1)}^2. \tag{9}$$

In reality, such probabilities vary depending on factors such as line length, pair of line proximity, local climate, weather variations, etc.

To illustrate the conceptual framework and our notations, we represent in Figure2 a schematic hypothetical case. Our goal is to improve over the "N-1" policy, thus our **baseline method** (represented as a **dark blue** line) will be to rank first all "n-1" contingencies. We define:

$$\mathcal{V}^* = \text{"N-1"} \quad \text{with} \quad |\mathcal{V}^*| = n, \tag{10}$$

corresponding to an associated cost and residual risk:

$$C^* = C(\mathcal{V}^*) \quad \text{and} \quad R^* = R(\mathcal{V}^*, x). \tag{11}$$

The dark blue curve is composed of two straight segments. It corresponds to the expected value of $R(\mathcal{V}, x)$ when ranking first (in random order) all "n-1" events, then (in random order) all "n-2" events. Between $|\mathcal{V}| = 0$ and $|\mathcal{V}| = |\mathcal{V}^*| = n$, we have:

$$R(\mathcal{V}, x) = R_{\max}(x) - \alpha_1 \pi_{(1)} |\mathcal{V}| \tag{12}$$

and between $|\mathcal{V}| = |\mathcal{V}^*| = n$ and $|\mathcal{V}| = |\mathcal{Z}|$, we have:

$$R(\mathcal{V}, x) = R(\mathcal{V}^*, x) - \alpha_2 \pi_{(2)} |\mathcal{V}| \tag{13}$$

where α_1 and α_2 are the fraction of "bad" "n-1" events (contingencies) and "bad" "n-2" events respectively.

To compare our proposed method with the baseline, we will either:

- Evaluate the **reduction of residual risk** $R^* - R(\mathcal{V}, x)$ for $C(\mathcal{V}) = C^*$, C^* being the **maximum (computational) cost** we are willing to incur, or;
- Evaluate the **reduction of cost** $C^* - C(\mathcal{V})$ for $R(\mathcal{V}, x) = R^*$, R^* being the **maximum residual risk** we are willing to incur.

Our proposed method can at best achieve an optimal strategy (**orange curve** in Figure2) in which all "bad" events (first all "n-1" then all "n-2" contingencies) are ranked first. This yields a first segment with slope $-\pi_{(1)}$ as a function of $|\mathcal{V}|$, then a second segment with slope $-\pi_{(2)}$.

B. Parameters setting

We wanted our simulations to be as close as possible to the situation of the real French power system.

Expert dispatchers (TSO operators responsible to operate the grid in security) estimate that a full "N-1" simulation yields approximately 100 "bad" events (dangerous contingencies). This means that approximately 1% of the single events should present a serious risk requiring a corrective action. To respect the order of magnitude of this proportion of "bad" events, we used a calibration dataset, which allowed us to *set* the thermal

limits[5] of each line in our test case grid. Having set these values, we evaluate them on the full "N-1" for 100 different grid states[6], requiring to compute 18 600 load flows. Among all the "n-1" events investigated, 1.75% were found unsafe in our simulations and 4.22% for "n-2" events.

To be as close as possible to the French power system, we also calibrated the failure probabilities $\pi_{(1)}$ and $\pi_{(2)}$. We wanted that the ratio between the risk due to single failures and the one due to double failure to be constant across grid sizes, this leads to consider:

$$\pi_{(1)} \approx 5.4\ 10^{-3} \tag{14}$$

$$\pi_{(2)} \approx 2.9\ 10^{-5} \tag{15}$$

For more information, the section V of the supplemental material[5] shows in detail how we obtained these probabilities.

IV. PROPOSED METHODOLOGY

In this section, we explain how we rank the contingencies z with respect to their estimated severity $\hat{L}(z; x)$ for a given system state x. $\hat{L}(z; x)$ is provided by an artificial neural network, trained on simulated data generated using a high-end load flow simulator.

Consider a fixed grid state x and a given contingency z, we denote by f_i the flow, computed with the high-end simulator, on the i^{nth} line of grid x after contingency z occurs, and by $\bar{f}_i$ the thermal limit for this line.

We propose to first train a neural network with "guided dropout", as describe in [1] to approximate rapidly the power flow for the given grid state x and contingency z. We will 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 "over confident" in their predictions (see for example [15]). This overconfidence could lead to a bad ranking in practice with dramatic effects. We propose to calibrate the score of our neural network taken into account a fixed (yet calibrated) uncertainty by assuming:

$$\forall i, (f_i - \hat{f}_i) \sim \mathcal{N}(0, \sigma_i) \tag{16}$$

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 with and also distinct from the final test set we use to evaluate performance. 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:

$$\sigma_i \stackrel{\text{def}}{=} \frac{1}{\text{number of simulations}} \cdot \sum_{\text{simulation}s} (\hat{f}_i - f_i)^2 \tag{17}$$

---

[5] See supplemental material at https://github.com/BDonnot/acpgunn for a detailed value of the thermal limits used.

[6] See the section V-A for a detailed description of this dataset.

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

$$\hat{L}_i \stackrel{\text{def}}{=} 1 - F_{\sigma_i}(\bar{f}_i - \hat{f}_i) \tag{18}$$

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 contingency z, if at least one of its line is above its thermal limit. The score of the power grid, in state x after contingency z, is then obtain with:

$$\hat{L}(z; x) \stackrel{\text{def}}{=} \max_{1 \le i \le n} \hat{L}_i(z; x) \tag{19}$$

We now have defined $\hat{L}(z; x)$. If all contingencies z were equiprobable ($\pi_{(1)} = \pi_{(2)}$), we could use this approximation for ranking. In practice, some events occur more often that others and therefore must be prioritized accordingly.

The first thing we tried was to rank directly the contingencies z with respect to their relative cost $p(z)\hat{L}(z; x)$. This worked, but this order tends to be really conservative: all the single contingencies were ranked first. This can be explained. There is a 10^3 relative factor between $\pi_{(1)}$ and $\pi_{(2)}$, so for a double contingency $z_{i,j}$ to be simulated before a single one z_i, this would mean that $\hat{L}(z_{i,j}; x) > 10^3 . \hat{L}(z_i; x)$. But, we made various hypothesis, in particular that the error on the flow where normally distributed (see equation 16) which is not the case in reality. In general we have then $\hat{L}(z_i; x) - L(z_i; x) >> 10^{-3}$. Even if the contingency z_i is harmless ($L(z_i; x) = 0$), this would imply $\hat{L}(z_i; x) > 10^{-3}$, so this single contingency would be ranked before any "n-2" contingency, which can be different in reality.

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 $\pi_{(1)}/\pi_{(2)}$ leads to the best results in almost every situation $(z; x)$.

To wrap up, all the grid states and contingencies are sorted with respect to their "scores" $\hat{s}(z; x)$ defined as:

$$\hat{s}(z; x) \stackrel{\text{def}}{=} \begin{cases} \hat{L}(z; x) \times \log\left(\frac{\pi_{(1)}}{\pi_{(2)}}\right) \text{if } z \text{ is a single contingency} \\ \hat{L}(z; x) \text{ otherwise} \end{cases} \tag{20}$$

V. EXPERIMENTS

In this section we report extensite experiments validating our proposed methodology, using a standard grid benchmark (the 118 bus grid of the matpower library [16], used for example in [8]). We begin by explaining our data generation methodology to create a dataset with over 2 million samples. Then we report results showing a 50 fold speedup to achieve the same residual risk currently obtained with the "N-1" policy. We also show how the remaining computational time can be exploited to significantly reduce the residual risk by simulating higher order contingencies (for example double contingencies).

A. Datasets generation

We conducted systematic experiments on small size benchmark grid (the 118 bus grid) from Matpower [16], a library commonly used to test power system algorithms [17].

We generated 500 different grid states varying the injections x of the initial grid given in Matpower. We used our knowledge of the French grid to mimic the spatio-temporal behavior of real data [1] when simulating our new datasets. For example, we enforced spatial correlations of productions and consumptions and mimicked production fluctuations, which are sometimes disconnected. Target values were obtained by computing resulting flows in all lines with the proprietary *AC* power flow simulator *Hades2*.

Training and validation sets. On these 500 test cases, we then computed, still using the high-end simulator Hades2, the full "N-1" (making $500 \times 186 = 93.000$ load flow computations). We also simulated $k = 5000$ load flows randomly selected among the $\frac{186 \times 185}{2} = 17\,205$ possible "n-2" double contingencies. This last dataset counts then 2 500 000 rows. We split this dataset in two subsets: 75% for training and 25% for validation/model selection *i.e.* for finding the best architecture and hyper-parameters of the neural networks. The detailed architecture tested can be found in the supplemental material at https://github.com/BDonnot/acpgunn.

Calibration and test sets. For the calibration dataset, used to fit the error of the model $\boldsymbol{\sigma}$, we simulated 75 different grid states x, and the full "N-1" and "N-2" for all of these simulations. The test set is composed of 25 grid states, and their full "N-1" (all single contingencies) and "N-2" (all double contingencies) associated dataset.

B. Neural network architecture

In this part, we will explain more about the neural network architecture used for the experiments, the way we trained it as well as the method used to find the so called "meta parameters" that gives the best results.

The used architecture (introduced in [1]) allows fast assessment of power flows given injections (productions and consumptions). We used this architecture that we called "guided dropout" as a fast proxy of the power flows simulator. The main idea of this purposely designed architecture is to adapt the *architecture* of the neural network to the *topology* of the grid. For example, we switch on/off hidden unit of the neural network depending on the sate off a given line in the power grid (connected / disconnected). We showed in this previous paper, that this architecture allows to super-generalize, eg. even when trained on a small grid configurations, it is able to predict flows properly to unseen grid states.

To present more this architecture we consider the neural network is a function F with input x, l layers each counting u units, and output y (the flows ins our cases), and non linearities $\phi(\cdot)$. We have, for layer $k \in \{1, \ldots l\}$:

$$y^{(k)} = \phi(W^{(k)} . x^{(k)}) \tag{21}$$

where $y^{(k)}$ is the output of layer k and $x^{(k)}$ its input ($x^{(k)} = y^{(k-1)}$), and . denotes the matrix multiplication. For a sake

of simplicity, bias are not written. The "guided dropout" (see [1]) consist in "masking" some part of the output of certain layers depending on external condition (here the state of a line: connected / disconnected).

In our power system example, we decided to "guided mask out" the layer k of our neural network. We then define (once and for all) for each power line i a masking vector $\boldsymbol{m}^{(i)}$ of 0's and 1's. The equation for layer k would become:

$$y^{(k)} = \boldsymbol{m} \odot \phi(W.x^{(k)}) \tag{22}$$

with $\odot$ denote the adamar product (element wise multiplication), and $\boldsymbol{m}$ is a mask build from the $\boldsymbol{m}^{(i)}$'s with:

$$\boldsymbol{m} = \begin{cases} \text{identity if both } li_1 \text{ and } li_2 \text{ are connected} \\ \boldsymbol{m}^{(1)} \text{ if } li_1 \text{ is disconnected and } li_2 \text{ is connected} \\ \boldsymbol{m}^{(2)} \text{ if } li_2 \text{ is disconnected and } li_1 \text{ is connected} \\ \boldsymbol{m}^{(1)} \odot \boldsymbol{m}^{(2)} \text{ if both } li_1 \text{ and } li_2 \text{ are disconnected} \end{cases} \tag{23}$$

This can be viewed as an adaptation of the neural network configuration, depending on the power grid topology. The "presence / absence" of power line will have a direct impact on the "presence / absence" of unit in some layers.

To train the neural networks, we use the "adam" optimizer, variant of the stochastic gradient descent first introduced in [18]. All the neural networks have been trained with the same number of minibatches. This number of minibatches have been calibrated such that the error on the validation set stop decreasing. Among all the meta parameters, we use a cross validation scheme to find the best combination of learning rate, number of layer and number of units per layer. The "relu" (rectifier linear unit) was used as the non linearity, except in the last layer. More details about the architecture used and the meta parameters can be found in the supplemental material at https://github.com/BDonnot/acpgunn.

We trained the same architecture according to two schemes. The first one **"Guided Dropout (trained n-1 only)"** is trained only with single contingencies. During the training we never show input/output pairs coming from a power grid having suffered 2 contingencies. We do that to evaluate the robustness to the method to the input distribution: how the model will generalize facing unseen situations. In the second training scheme refered as "**"Guided Dropout**", we suppose that the distribution of the training set is really close to the one of the test set. This is a viable hypothesis in practice if the neural network can be retrained very often.

C. Results

Figures 3 and 4 represent the lift curves including:
- the random ranking (light blue curve);
- the baseline ranking (dark blue curve) corresponding to ranking the "n-1" events first, as TSO operators would do;
- the ideal ranking (orange curve) in which the "bad" "n-1" events come first followed by the "bad" "n-2" events, and;

- the ranking for two different neural networks (green curves).
  - the plain curve represents the ranking for a neural network trained on the entire training set, which includes all "n-1" and randomly selected "n-2" contingencies; training and test data are similarly distributed.
  - the dashed curve represents the ranking for a neural network trained on the "N-1" dataset only *i.e.* no double contingency is used for training. This neural network therefore exhibits "super-generalization" to multiple contingencies. Such change in distribution of data may occur in actual grid operation situations and the performances obtained are encouraging.

The curves represented in Figure 3 are computed for a grid state x corresponding to a worse case scenario, yielding the highest loads of our dataset. This simulates "peak demand", when loads are $\simeq 40\%$ higher than for the reference case. Studying such extreme cases validates best our approach.

As we can observe in Figure 3, both neural networks perform better than the emulation of TSO operator ranking in every situation. Remarkably, even the neural network that has never seen double contingencies during training is capable of ranking all single AND double contingencies quite well. This is an encouraging result indicating that our methods (called "guided dropout") is capable of "super-generalization" a form of transfer learning to cases of contingencies never seen before.

TABLE I: **Comparison of methods** on the whole test set (mean over all x cinsidered $\pm$ one standard deviation). Gini coefficient (between 0 and 1, 1 is best), residual risk $R(\mathcal{V}^*)$ for a maximum computational budget $\mathcal{V}^*$ and computation cost $C(R^*)$ ensuring that the risk residual risk remains below R^* (in test data). For both $R(\mathcal{V}^*)$ and $C(R^*)$, smallest is best.

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

The table I presents the Gini coefficient (normalized area under lift curve; an index between 0 and 1, highest is best)[19]

as well as the residual risk at a fixed computational budget $R(\mathcal{V}*)$, and the cost for achieving a residual risk below R^*. Our neural network method is always better that the TSO operator strategy, with respect to all metrics. The most promising results of Table I is in the 3rd row. Our neural networks perform as well as the optimal strategy. On average, we need to simulate only $3 - 4$ contingencies 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" contingencies to be representative of the French power grid, we expected speed-
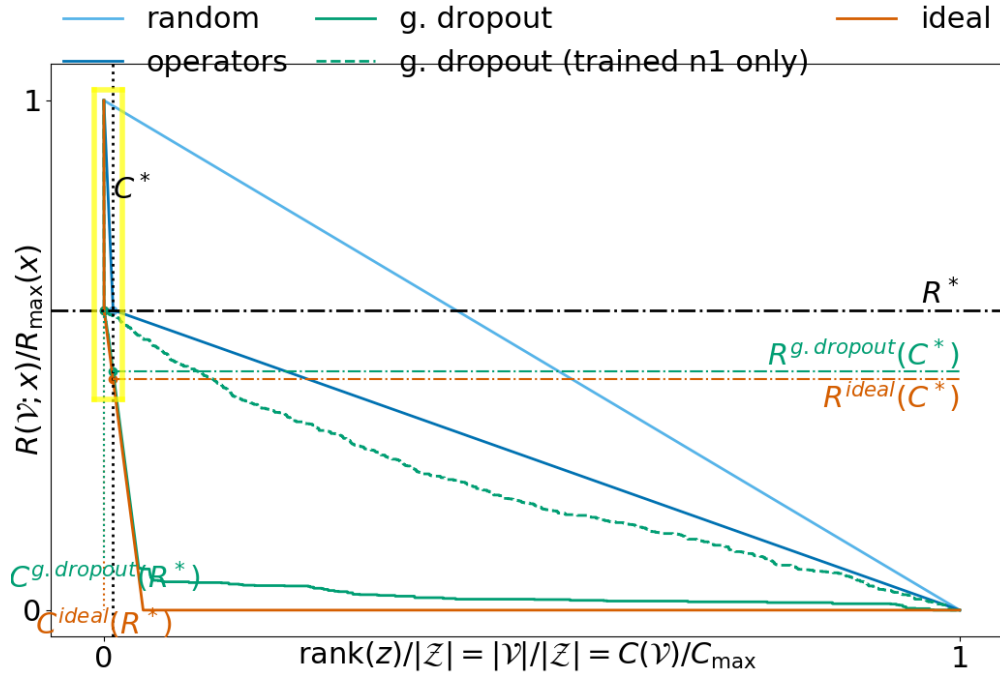
Fig. 3: **Contingency ranking using two neural networks**. The first network (plain line) is trained with both "n-1" and "n-2" contingencies, and it is tested on a situation not seen during training. The second neural network (dashed green line) is trained only on the "n-1" data: it has seen during training only "single contingencies" (a small fraction of all the contingencies it is tested on).
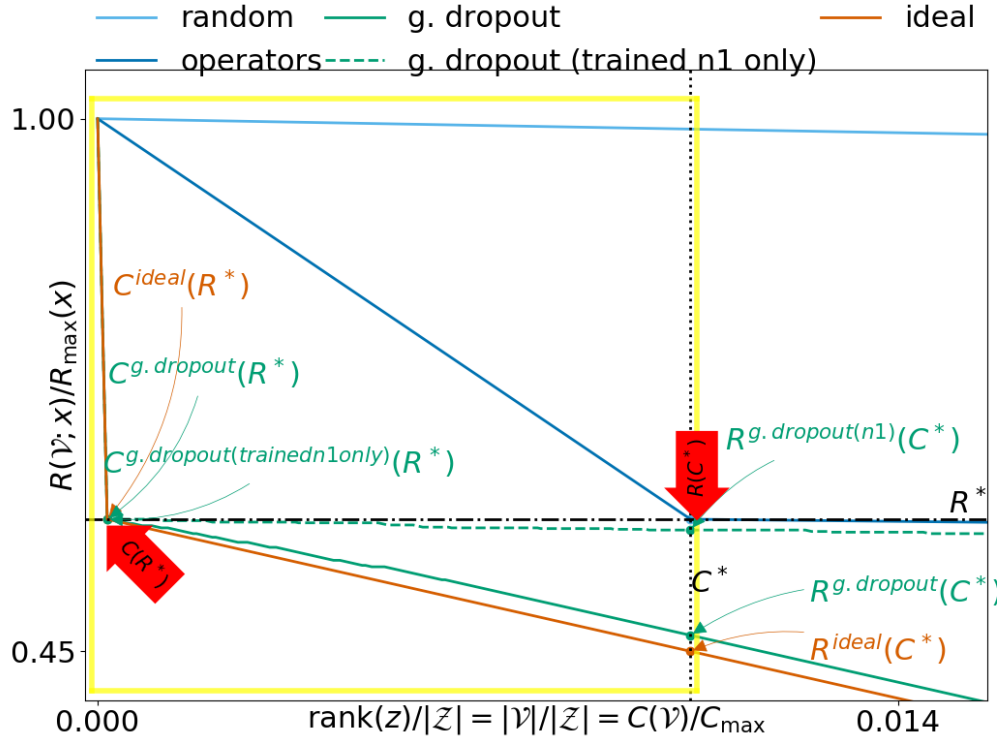

Fig. 4: **Zooming on the contingency ranking**. This figure corresponds to zooming the yellow rectangle in the previous figure, focusing on the region between $|\mathcal{V}| = 0$ and $|\mathcal{V}| = n$ in which we hope to improve over the current "N-1" TSO security policy.

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! Testing on new contingencies only improve results, as seen on Figure 3. Thus, the time saved with our method can be used to further reduce the residual risk by investigating more candidate "contingencies".

To sum up, both neural networks are capable of correctly ranking first all "bad" single contingencies, and single contingencies are ranked before double contingencies. As in every ranking problem, there is a tradeoff between the hit rate (detecting true "bad" contingencies) and the false alarm rate (falsely diagnosing harmless contingencies as "bad"). From the point of view of ensuring the security of a power grid, the severity of 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 residual risk curve $R(\mathcal{V})$ decreases very fast, ensuring that the "hit rate" be high initially. Remarkably, both neural networks rank first the "bad" "n-1" contingencies., then chose among the worst "n-2" contingencies. The neural networks privileged a high "hit rate" over lowering the "false alarm rate", a risk adverse behavior, which is expected to operate power systems in security.

VI. CONCLUSION

In this paper, we proposed a novel approach to rank the most dangerous contingencies in power grids (namely those leading to lines exceeding their thermal limits and posing problems of grid security). This approach is quite general: it can rank single contingencies (one line disconnection) or multiple contingencies (two or more lines disconnected), with different probabilities of occurring. In this paper we studies the case of single and double contingencies.

Our methodology in a nutshell can be summarized as follows: (1) Train a neural network to mimic a load flow simulator. (2) Use it (on new test data) to evaluate how close each line it to its thermal limits.(3) Rank contingencies accordingly in decreasing order of severity. Our simulations results on a standard benchmark case are quite promising:

First, with this approach, we can accurately identify the most dangerous contingencies. In all cases, we could use a computational budget of only 2% of the "N-1" policy budget (the strategy presently delyed in production).

Second, this method is robust to changes in distribution between training and test time. Training a neural network with only 0.5% of all possible contingencies (single contingencies) is enough to achieve a better performance than today's "N-1" criterion.

We evaluated that this method is scalable to the full Ultra High Voltage powergrid, and may result in speed-ups by a factor of 2 000 or more in power flow calculations.

Although our approach was developed for power systems and demonstrated on a particular case study, it is rather generic and could easily be extended to other domains in which risk can be evaluated by a high-end but slow simulator and simulations must be prioritized for efficiency reasons.

REFERENCES

[1] B. Donnot, I. Guyon, M. @bullet, A. Marot, and P. Panciatici, "Fast power system security analysis with guided dropout," Apr. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01695793

[2] E. Vaahedi, C. Fuchs, W. Xu, Y. Mansour, H. Hamadanizadeh, and G. Morison, "Voltage stability contingency screening and ranking," *IEEE Transactions on power systems*, vol. 14, no. 1, pp. 256–265, 1999.

[3] L. Wehenkel, "Machine learning approaches to power-system security assessment," *IEEE Expert*, vol. 12, no. 5, pp. 60–72, 1997.

[4] L. A. Wehenkel, *Automatic learning techniques in power systems*. Springer Science & Business Media, 2012.

[5] I. Saeh and A. Khairuddin, "Static security assessment using artificial neural network," in *Power and Energy Conference, 2008. PECon 2008. IEEE 2nd International*. IEEE, 2008, pp. 1172–1178.

[6] S. Fliscounakis, P. Panciatici, F. Capitanescu, and L. Wehenkel, "Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4909–4917, 2013.

[7] L. Duchesne, E. Karangelos, and L. Wehenkel, "Machine learning of real-time power systems reliability management response," in *2017 IEEE Manchester PowerTech*. IEEE, jun 2017. [Online]. Available: https://doi.org/10.1109/ptc.2017.7980927

[8] R. Sunitha, S. K. Kumar, and A. T. Mathew, "Online static security assessment module using artificial neural networks," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4328–4335, 2013.

[9] H. Li, "A short introduction to learning to rank," *IEICE TRANSACTIONS on Information and Systems*, vol. 94, no. 10, pp. 1854–1862, 2011.

[10] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," in *Advances in neural information processing systems*, 2003, pp. 961–968.

[11] S. Agarwal, D. Dugar, and S. Sengupta, "Ranking chemical structures for drug discovery: A new machine learning approach," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 716–731, 2010, pMID: 20387860. [Online]. Available: http://dx.doi.org/10.1021/ci9003865

[12] 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.

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

[14] B. Donnot and et al., "Introducing machine learning for power system operation support," in *IREP Symposium*, Espinho, Portugal, Aug. 2017. [Online]. Available: https://hal.inria.fr/hal-01581719

[15] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014. [Online]. Available: http://arxiv.org/abs/1412.1897

[16] R. D. Zimmerman and et al., "Matpower," *IEEE Trans. on Power Systems*, pp. 12–19, 2011.

[17] O. Alsac and B. Stott, "Optimal load flow with steady-state security," *IEEE transactions on power apparatus and systems*, no. 3, pp. 745–751, 1974.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] C. Gini, ""concentration and dependency ratios" (in italian – english translation in rivista di politica economica, 87 (1997), 769–789)," 1909.