Supplemental material: 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 damage. 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 $1\ 000$ nodes).**

I. Introduction

This paper presents the supplemental material for the submitted paper "Anticipating contingencies in power grids, using fast net screening".

It is organized as followed. In the first section, we will re present the guided dropout algorithm, apply to our specific problem. The second section will be dedicated to the neural network architectures, and how we tune them. Lastly we will expose the thermal limits we use for the fictive grid throughout our experiments.

II. Guided dropout algorithm

Let's formulate our neural network as a function F with input x, l layers of u units each, and output y, and non linearities $\phi(\cdot)$. We have, for layer $k \in \{1, \dots l\}$:

$$y^{(k)} = \phi(W^{(k)}.x^{(k)}) \tag{1}$$

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 here.

The guided dropout (see [1]) consist in "masking" some part of the output depending on external condition.

In our power system example, let's imagine we have 2 power line li_1 and li_2. We decided to "guided mask out" the layer k of our neural network. We would then define (once and for all) two vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$ of 0's and 1's , with the condition if for a components i, $\boldsymbol{m}_i^{(1)} = 0$ then for the same component i, $\boldsymbol{m}_i^{(2)} = 1$. The equation for layer k would become:

$$y^{(k)} = \boldsymbol{m} \odot \phi(W.x^{(k)}) \tag{2}$$

with $\odot$ denote the adamar product (element wise multiplication), and $\boldsymbol{m}$ is a mask with:

$$\boldsymbol{m} = \begin{cases} \text{identity if both } li_1 \text{ and } li_2 \text{ are connected} \\ \boldsymbol{m}^{(1)} \text{ if } li_1 \text{ is disconnected and } li_2 \text{ is connected} \\ \boldsymbol{m}^{(2)} \text{ if } li_2 \text{ is disconnected and } li_1 \text{ is connected} \\ \boldsymbol{m}^{(1)} \odot \boldsymbol{m}^{(2)} \text{ if both } li_1 \text{ and } li_2 \text{ are disconnected} \end{cases} \tag{3}$$

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.

III. Tested neural network architectures

Recall that for these experiments, we used neural network, trained on 500 different grid states, and their "N-1" dataset and a sampling of 5 000 among the 17 205 possible grid state representing the "N-2" dataset. See the original paper, section "Dataset generation" for a more complete description of this process.

75% of this dataset have been used to train the models, and 25% used to find the best architectures. In this section, we will report error only on this validation dataset.

By contrast of the paper presenting the guided dropout architecture (see [1]), we made some improvement of the general architecture.

Instead of considering only active part of the loads, we also add the reactive part of it as input data, and instead of predicting only the current at one end of each line, we ask the neural network to forecast both the current and powerflow, at both end of each power line.

Knowing that the powergrid counts 99 loads, 54 productions and 186 powerlines, each neural network tested have 252 inputs:

$$99 * \underbrace{2}_{\text{for active, and reactive values}} + 54 = 252 \tag{4}$$

and 744 outputs:

$$\underbrace{186}_{\text{number of power lines}} \times \underbrace{2}_{\text{A power line has 2 ends}} \times \underbrace{2}_{\text{At each end, we predicted current flow, and power flow}} = 744$$

We suppose that we give a unique ID to each productions, loads and lines. We also suppose that each power line have an "origin" (one of its end) and an extremity (the other end). We will also denote by $\boldsymbol{p}_a$ the vector (size 54) of all the active production values, $\boldsymbol{c}_a$ (resp. $\boldsymbol{c}_r$) the vector (size 99) of active (resp. reactive) loads values, $\boldsymbol{f}_c^{(or)}$ (resp. $\boldsymbol{f}_c^{(ext)}$) the vector (size 186) of the origin (resp. extremity) current flows on each line of the power grid and finally $\boldsymbol{f}_a^{(or)}$ (resp. $\boldsymbol{f}_a^{(ext)}$) for the active flow at the origin (resp. extremity) on each line of the power grid.

A. *Dealing with various type of data*

Input data have not the same dimension, nor the same role. Some representative "active" quantity (physical unit MW), others reactive value (physical unit "MVaR") or others current (physical unit A), and other have no dimension (one-hot encoding vector of line presence/absence).

We found out that "encoding" (or "embedding") each variable type ($\boldsymbol{p}_a$, $\boldsymbol{c}_a$ and $\boldsymbol{c}_r$) independently at the beginning of the neural network improve convergence speed, while reducing the number of parameters. If we denote by $e_{p;a}$, $e_{c;a}$ and $e_{c;r}$ neural network with respective inputs $\boldsymbol{p}_a$, $\boldsymbol{c}_a$ and $\boldsymbol{c}_r$, counting each 2 hidden layers, and 200 units per layer, as well as 200 output, the input that we feed the "main" neural network is the concatenation of the outputs of each of the encoding. So the data are then "embedded", before being concatenate and fed to the neural network. The embedding size, as well as its depth have not be calibrated for this particular study.

For the same reason, we also use 2 "decoding" layer, independently for each output variable So, with the notation of previous section, we have:

$$x = (e_{p;a}(\boldsymbol{p}_a), e_{c;a}(\boldsymbol{c}_a), e_{c;r}(\boldsymbol{c}_r)) \tag{5}$$

all the "encoder" above are trained with the main neural network. This means that data are "pre processed" our neural network, before being treated as a whole by the guided dropout neural network.

Let "n_l" denotes the number of layers for the intermediate network, and "s_l" its size, the final network representation is then :

$$\begin{tikzcd}
\text{input} & \boldsymbol{p}_a \arrow[d] & \boldsymbol{c}_a \arrow[d] & \boldsymbol{c}_r \arrow[d] & \\
\text{embedding} & e_{p;a}(\cdot) \arrow[dr] & e_{c;a}(\cdot) \arrow[d] & e_{c;r}(\cdot) \arrow[dl] & \\
\text{concatenation} & & \text{x} \arrow[d] & & \\
n_l \text{ guided dropout layer} & & F(\cdot) \arrow[d] & & \\
\text{decoding} & & d(\cdot) \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{output} & \hat{\boldsymbol{f}}_a^{(or)} & \hat{\boldsymbol{f}}_a^{(ex)} & \hat{\boldsymbol{f}}_c^{(or)} & \hat{\boldsymbol{f}}_c^{(ex)}
\end{tikzcd}$$

B. *Using residual architecture*

In all the neural network presented here, we use a residual architecture (as presented in [2] and [3]) for most part of our networks. This means that the equation for one layer k (see equation 1) now becomes:

$$y^{(k)} = \phi(W_2^{(k)}.\phi(W_1^{(k)}.x^{(k)})) + x^{(k)} \tag{6}$$

where $y^{(k)}$ is still the output of layer k and $x^{(k)}$ its input ($x^{(k)} = y^{(k-1)}$). For a guided dropout version of this type of layer, we choose to implement it as:

$$y^{(k)} = \boldsymbol{m} \odot \left(\phi(W_2^{(k)}.\phi(W_1^{(k)}.x^{(k)})) + x^{(k)}\right) \tag{7}$$

where $\boldsymbol{m}$ is the current mask (see section II for a more complete explanation)

C. *Other meta parameters*

For all the experiments, we also used the "Adam" optimizer [4] with default parameters in tensorflow. The minibatch size have been set to 100. All models were trained for 1000 epoch. A typical training curve is shown in figure 1.

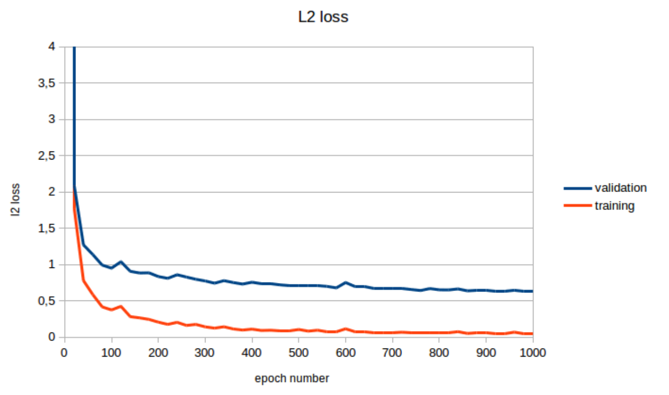

Fig. 1: **Learning curve of the retained model**. This figure represents the training loss (l_2 loss) on the training set (orange) and validation set (blue) as a function of the training epoch

Each dataset is pre-processed by centering/reducing it along each dimension (column). Weights matrices were initialized with "Xavier initialization" ([5]) and 0 for the biases.

Previous experiments show us a learning rate of 510^{-5} achieved the best performance in our settings. We did not try to recalibrate this meta parameter for this experiments.

TABLE I: **Best model for each architecture**. In this table we show the "MAPE@90" (in percentage) for the best model of each architecture.

	G. Dropout			One hot		
number of unit:	150	300	600	150	300	600
2 layers	0.74	**0.57**	0.85	0.98	0.98	**0.92**
4 layers	0.95	0.95	0.94	1.01	0.94	1.03
8 layers	1.04	1.03	1.08	1.21	1.06	0.95

D. *Meta parameter selection*

Among all meta parameters of our architecture, only the number of hidden unit for the "main" neural network, as well as their size have been search through a grid search, in 2 cases:

- one hot encoding, with "N-1" data only
- guided dropout encoding, with "N-1" data only

We did not check the error on the "N-2" dataset of the model tested on "N-1" data only. To select which model performs best, we use an error adapt to our porblem the "MAPE@90". This error function will compute the Mean Absolute Percentage Error (MAPE) for the 10% highest values (in absolute values) for the flows. In the paper, we were interested in overflow, so with models that predicted best the high flow values. Formally, for a *single* power line l_i, with real flow vector $\boldsymbol{f}^{(i)}$ and predicted flow $\hat{\boldsymbol{f}}^{(k)}$ computed over n_s simulations, we have:

$$\text{MAPE@90}(\boldsymbol{f}^{(i)}, \hat{\boldsymbol{f}}^{(i)}) \overset{\text{def}}{=} \tag{8}$$

$$\frac{1}{n_s} \sum_{1 \leq k \leq n_s} \frac{\left| \hat{\boldsymbol{f}}_k^{(i)} - \boldsymbol{f}^{(i)} \right|_k}{\boldsymbol{f}_k^{(i)}} . \text{Ind} \left(\boldsymbol{f}_k^{(i)} \geq \text{quantile}(90; \boldsymbol{f}_k^{(i)}) \right) \tag{9}$$

with Ind denoting the "indicator function" (see below), and quantile$(90; z)$ the 90% quantile of the vector z.

$$\text{Ind}(z) = \begin{cases} 1 \text{If } z \text{ is true} \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

MAPE@90$(\boldsymbol{f}^{(i)}, \hat{\boldsymbol{f}}^{(i)})$ are then average for the 186 powerline of the power grid.

Then, 5 models for each architectures have been trained, and the best one have been selected. An extensive grid search have been made for the "N-1" trained model. In both cases (one hot and guided dropout) the same architecture have been used to train a neural network on the N-1 and n-2 dataset. As one can see on the table I, the best architecture found for the guided dropout counts 2 layers and 300 units per layer, and performs (much) better than the best model of the best tested architecture for the one hot model (counting still 2 layers, but 600 units per hidden layer).

IV. Thermal limits used

In the paper, we mention that, compared to the original test case, we modified the thermal limits. We did that to keep the ratio of "bad" "N-1" contingencies approximately to the same value as it is for the most tension grid state in French grid (approximately $1-2\%$).

TABLE II: **Thermal limits**. In this table we show the thermal limits used in the paper.

to	from	th. limits (Amps)	to	from	th. limits (Amps)	to	from	th. limits (Amps)
1	2	2010	38	65	9090	74	75	6090
1	3	3050	39	40	5190	75	77	5430
2	12	3250	40	41	4840	75	118	5330
3	5	5770	40	42	6170	76	77	5620
3	12	4230	41	42	6690	76	118	3140
4	5	10180	42	49	8470	77	78	7000
4	11	7310	42	49	8470	77	80	8490
5	6	8370	43	44	6690	77	80	5140
5	11	9230	44	45	7820	77	82	3270
6	7	5600	45	46	5560	78	79	4670
7	12	5200	45	49	6200	79	80	7140
8	5	10720	46	47	4220	80	96	2800
8	9	9850	46	48	2830	80	97	3130
8	30	8960	47	49	4300	80	98	2620
9	10	9630	47	69	5100	80	99	2540
11	12	8810	48	49	4130	81	80	4300
11	13	8150	49	50	4440	82	83	6390
12	14	10390	49	51	5450	82	96	4910
12	16	10620	49	54	3890	83	84	3070
12	117	1240	49	54	3830	83	85	4440
13	15	9540	49	66	11700	84	85	3650
14	15	11190	49	66	11700	85	86	1370
15	17	16890	49	69	4550	85	88	5570
15	19	7440	50	57	3460	85	89	7800
15	33	6170	51	52	2760	86	87	1030
16	17	11900	51	58	2770	88	89	8410
17	18	10010	52	53	2410	89	90	10430
17	31	6340	53	54	3920	89	90	10570
17	113	6180	54	55	1370	89	92	14580
18	19	7260	54	56	3030	89	92	8990
19	20	4110	54	59	2220	90	91	6060
19	34	5960	55	56	1830	91	92	5050
20	21	5120	55	59	2290	92	93	4120
21	22	5880	56	57	2720	92	94	3900
22	23	6480	56	58	3030	92	100	2250
23	24	10300	56	59	1920	92	102	2930
23	25	16910	56	59	2020	93	94	3460
23	32	11950	59	60	8200	94	95	4460
24	70	6100	59	61	8470	94	96	4100
24	72	5300	60	61	11850	94	100	3920
25	27	15980	60	62	7270	95	96	3260
26	25	7660	61	62	7520	96	97	2220
26	30	8030	62	66	7060	98	100	2160
27	28	5320	62	67	4830	99	100	2950
27	32	5450	63	59	4820	100	101	1750
27	115	4040	63	64	4820	100	103	7860
28	29	4380	64	61	3620	100	104	6140
29	31	3340	64	65	5610	100	106	5390
30	17	12510	65	66	4710	101	102	2630
30	38	5840	65	68	4620	103	104	3560
31	32	5740	66	67	6470	103	105	4200
32	113	5770	68	69	5340	103	110	4340
32	114	4260	68	81	3890	104	105	4880
33	37	6120	68	116	4950	105	106	3880
34	36	4040	69	70	11760	105	107	1370
34	37	8830	69	75	10020	105	108	2660
34	43	5710	69	77	9400	106	107	1740
35	36	5210	70	71	7760	108	109	2530
35	37	7150	70	74	3260	109	110	2130
37	39	4100	70	75	4760	110	111	2360
37	40	4070	71	72	6580	110	112	2050
38	37	7490	71	73	2170	114	115	4330

The list of all thermal limits used is shown in table II page 3. Thermal limits are expressed in Amps (A). To make sure the load-flow we used was appropriate, we scaled the injections so that the total active load (eg the sum all the consumptions) is equal to 65 MW.

References

[1] B. Donnot, I. Guyon, M. @bullet, A. Marot, and P. Panciatici, “Fast Power system security analysis with Guided Dropout,” in *European Symposium on Artificial Neural Networks*, Bruges, Belgium, Apr. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01695793

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[3] ——, “Identity mappings in deep residual networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.
[4] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
[5] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.