Visualizing Graph Neural Networks in Order to Learn General Concepts in Power Systems

Øystein Rognes Solheim*†, Gunnhild Svandal Presthus*, Boye Annfelt Høverstad*‡ and Magnus Korpås†
* Dept. of Data Science, Statnett SF, Oslo, Norway, {oystein.solheim, gunnhild.presthus, boye.hoverstad}@statnett.no
† Dept. of Electric Power Engineering, NTNU, Trondheim, Norway, {oysterso, magnus.korpaas}@ntnu.no
‡ Dept. of Computer Science, NTNU, Trondheim, Norway

***Abstract*—Neural networks play an increasingly important role in many complex decision-making systems. However, their lack of interpretability make it difficult to analyze and trust them when applied to critical infrastructure like the power system. This paper presents visualizations and exploratory analyses of the internal representations of a graph neural network based reinforcement learning agent applied to a power system reliability study. We explore three different dimension reduction techniques, and use these to demonstrate how agents with different generalizing capabilities have different internal representations. We study how the inputs to the agent are processed through the layers of the neural network components. Additionally, we use these visualizations to indicate that the agent can learn general concepts of the power system, and we hypothesize that the method can be expected to scale well to larger power systems. We also explore how the agent behaves in a grid expansion scenario with a power system not experienced during training.**

***Index Terms*—Power Systems, Graph Neural Networks, Visualization, Reinforcement Learning**

I. Introduction

The ongoing widespread green transition of the power system means more production from renewable power sources. Due to these increased levels of intermittent production, the system will be more complex and difficult to operate. When evaluating the reliability of this transformed system, we will need to quickly find the remedial actions following a power system failure. In this respect, the use of machine learning models [1], [2] will be an attractive alternative to the traditional time-consuming optimization methods [3], [4].

In both [5] and [6], a reinforcement learning (RL) agent was used in Monte Carlo simulations of the IEEE Reliability 24 bus test system (RTS) [7][1], to find optimal remedial actions following power system failures. A direct encoding of the power system state as a single vector input to the policy neural networks, without any a priori information of the topological structure of the power system was used in [5]. The Monte Carlo reliability simulation based on this agent was much faster and about as accurate as the traditional OPF-based approach. In [6], the method in [5] was improved, by taking advantage of the inherent graph structure of the electrical power system. The resulting graph based integrated actor-critic model was additionally able to perform well in power systems not seen during training.

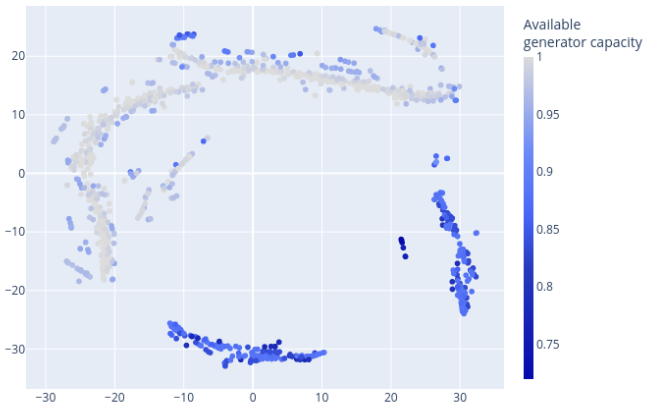
The internal high-dimensional representation and processing of information in neural network make these models challenging to interpret and analyze. In the operation of critical infrastructure like the power system, it is of utmost importance to understand and trust the models at hand. One step in this direction is to visualize these high-dimensional representations in 2D or 3D. Some classical linear techniques are Principal Component Analyses [8] and classical Multidimensional Scaling [9]. Several nonlinear methods have appeared in the last decades, where the t-distributed stochastic neighbor embedding (t-SNE) technique [10] and Uniform Manifold Approximation and Projection (UMAP) [11] have become popular. In this paper we will mostly use t-SNE but have also included PCA and UMAP for comparison.

The major difference between the models in [5] and [6] is that the first uses a deep feed forward neural network and the latter uses a graph neural network architecture. Since the underlying graph structure of the electrical power system can be exploited, we would expect the internal representations of the model in [6] to be more explainable. In this paper, we seek to demonstrate how dimension reduction techniques could help us understand the information processing in graph neural networks. To illustrate this point, in Fig. 1 we have applied the t-SNE method to the first hidden layer of the neural network of the agent policy used in [5] with a perplexity value of 50 and 1000 Monte Carlo simulation time steps. In Fig. 1a the markers are colored by the relative total available generator capacity. We see a clustering of the generators states with 100% in the upper left part, and two sub-clusters with reduced availability in the lower part. Fig. 1b uses the relative total load as the marker color, and in this plot we notice a relatively smooth gradient of the overall load through the individual clusters. Still, even though this information is interesting on a global scale, the direct encoding in [5] offers less information about local phenomena since there are no connection to a graph in the internal structure of the model. It is also difficult to assess how and why changes improve the model beyond metrics.

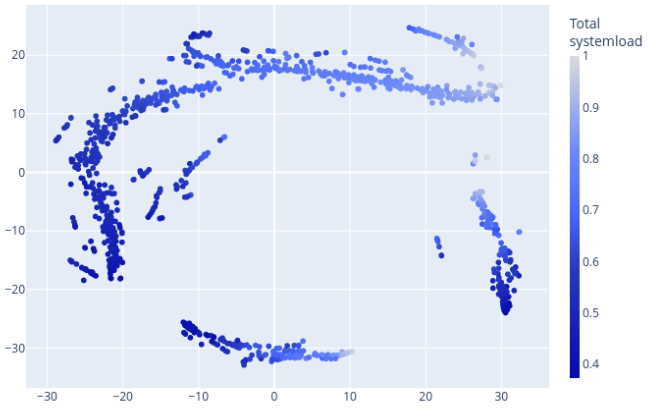
The main research contribution of this paper is to present visualizations and exploratory analysis of the internal representations of this graph neural network based RL agent

This work has been funded by the Norwegian Research Council under grant number 310436.

[1]We have included an illustration of the RTS in Fig. 12 at the end of this paper.



(a) Color by available generation capacity



(b) Color by total system load

Fig. 1: Visualizing the hidden layer of the policy of the direct encoded method of [5] using t-SNE.

applied to a power system reliability study. We use PCA, t-SNE and UMAP to show that the agent alters and improves its representation and generalizing capabilities as the input data to the agent is processed through the internal layers of the neural network components. We show how these visualizations can be used to support the view that the trained agent learns general concepts, indicating that the method presented in [6] can be expected to scale well to larger power systems where the agents are able to generalize to marginally larger power systems such as for reliability studies in a grid expansion scenario.

More specifically, we will

1) Visualize and study two different model versions, where each model is trained under slightly different conditions resulting in different generalizing capabilities.
2) Visualize and study how the representation changes during training stages and how it can be interpreted in terms of policy changes.
3) Visualize and analyze the representation of a power system not seen during training.

To the best of our knowledge, this is the first time the internal representation of the policy of a graph based actor critic acting on a power system has been studied.

The rest of this paper is organized as follows: In section II we give a short presentation of power system reliability and the model developed in [6]. In section III-B, we describe the PCA, t-SNE and UMAP methods. In section IV, we present visualizations using the models established in [6]. Finally, in section V, we discuss our findings and point to future research activities.

II. Graph neural networks and reinforcement learning

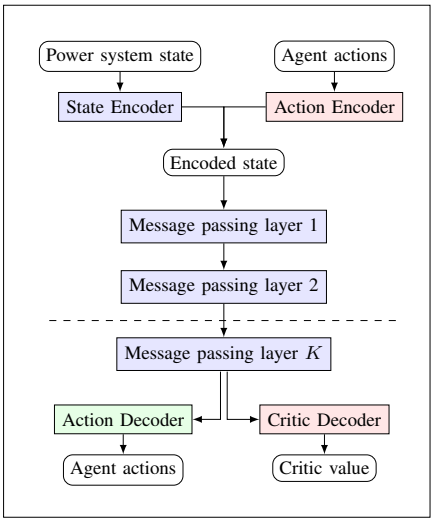


Fig. 2: Integrated Actor-Critic network architecture.

Graph neural networks are a class of machine learning techniques applicable to situations where the data structure can be described by a graph. A power system transmission network is an example of such a structure, where the buses and branches can be represented by nodes and edges in a graph. A key concept in several graph neural network algorithms is the message passing (MP) framework. In such algorithms, the core idea is to update the internal representation by aggregating the information from neighboring nodes, possibly by also using the edge and global graph features. By doing multiple updates, information within the graph can be transferred from a distance and not only from the neighboring nodes. The internal representations created by each MP layer would then presumably contain information related to both the physical data and the local topology of the power system.

Actor-critic methods are popular approaches to train reinforcement learning agents. The model presented in [6] is founded on the MP algorithm [12], where information is distributed through the graph as the depth of the MP layers are processed in the model. The overall model architecture is illustrated in Fig. 2. The model consists of input layers, encoding layers, K MP layers, and the final decoder layers

giving the actions and the real valued critic value. In the MP layers there are separate models for the edge, node and global features which are connected sequentially through a combination of attention and recurrence mechanisms, inspired by the generalized updating methods in [13], section 5.3. Each layer in the model is represented by a graph, with node features, edge features and global graph features. The actions of the agent are given as outputs at the nodes of a graph. For each node in that action graph, the agent has the opportunity to make two actions, corresponding to adjustments of generator production output and load curtailment.

In our model, we consider a simple symmetric directed graph $\mathcal{G}_{\text{STATE}} = (\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V}$ with cardinality $|\mathcal{V}|$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ with cardinality $|\mathcal{E}|$. The *adjacency* matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} = \{a_{ij}\}$ is defined by $a_{ij} = 1$ if $(u_i, u_j) \in \mathcal{E}$, and 0 otherwise. We denote the node features by the column vectors $x_u \in \mathbb{R}^{d_{\text{n}}}$, $\forall u \in \mathcal{V}$, and arrange these into the matrix $X = [... \; x_u \; ...] \in \mathbb{R}^{d_{\text{n}}\times|\mathcal{V}|}$. We denote the edge features by the column vectors $y_{(u,v)} \in \mathbb{R}^{d_{\text{e}}}$, $\forall (u,v) \in \mathcal{E}$, which are collected into the matrix $Y = [... \; y_{(u,v)} \; ...] \in \mathbb{R}^{d_e\times|\mathcal{E}|}$. We denote the graph features by $z \in \mathbb{R}^{d_{\text{g}}}$. In addition, we specifically define the action graph $\mathcal{G}_{\text{ACTIONS}}$ as the graph with the same adjacency matrix A but with node features $X_{\text{ACTIONS}} \in \mathbb{R}^{d_a\times|\mathcal{V}|}$ (no edge or graph features).

For the rest of this paper we will consider the actor part of the model architecture in Fig. 2. This is the left part of the figure, leading from power system state input to agent actions output. This means that the power system state is represented by an input graph $\mathcal{G} = \mathcal{G}_{\text{STATE}}$ with node, edge and graph features. The input graph is then processed by the State Encoder and the graph $\mathcal{G}$ is transferred to an encoded state graph $\mathcal{G}^*$. The State Encoder consists of a graph feed forward neural network for the node, edge and graph features, without any message passing updates.

With $\mathcal{G}^*$ as the input to the K message passing layers, within each message layer k, we define a sequential updating of edge $\mathbf{h}^k_{(u,v)}$, node $\mathbf{h}^k_u$ and graph features $\mathbf{h}^k_{\mathcal{G}}$ in the common message passing layers $k = 1 \dots K$:

$$\mathbf{h}^k_{(u,v)} = \text{UPDATE}_{\text{edge}}(\mathbf{h}^{k-1}_{(u,v)}, \mathbf{h}^{k-1}_u, \mathbf{h}^{k-1}_v, \mathbf{h}^{k-1}_{\mathcal{G}}) \tag{1}$$

$$\begin{aligned} \mathbf{m}^k_{\mathcal{N}(u)} &= \text{AGGR}_{\text{node}}\left(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{(u,v)}, \mathbf{h}^{k-1}_{\mathcal{G}} \quad \forall v \in \mathcal{N}(u)\right) \\ \mathbf{h}^k_u &= \text{UPDATE}_{\text{node}}(\mathbf{h}^{k-1}_u, \mathbf{m}^k_{\mathcal{N}(u)}) \end{aligned} \tag{2}$$

$$\mathbf{h}^k_{\mathcal{G}} = \text{UPDATE}_{\text{graph}}(\mathbf{h}^k_u, \mathbf{h}^k_{(u,v)}, \mathbf{h}^{k-1}_{\mathcal{G}} \; \forall u \in \mathcal{V}, (u,v) \in \mathcal{E}) \tag{3}$$

Here, the learnable UPDATE functions establish the relationship between previous message passing layers and the current. The AGGR function collects information for each node u in its neighbor $\mathcal{N}(u)$.

We initialize the message passing layer using the features of graph $\mathcal{G}^*$ where

$$\begin{aligned} \mathbf{h}^0_u &= x^*_u, \; \forall u \in \mathcal{V}, \\ \mathbf{h}^0_{(u,v)} &= y^*_{(u,v)}, \; \forall (u,v) \in \mathcal{E}, \\ \mathbf{h}^0_{\mathcal{G}} &= z^*, \end{aligned} \tag{4}$$

where x^*_u, $y^*_{(u,v)}$ and z^* are the node, edge and graph features of the state encoded graph $\mathcal{G}^*$. In section IV we visualize the input layer and the K message passing layers of this model. We refer to [6] for further details concerning the architecture.

The model is trained using a prioritized replay buffer $\mathcal{D}$ [14] where we integrate the critic and target networks as in [15]. The critic loss is defined through $L(\theta) = \mathbb{E}_{\mathcal{D}}[\delta_i]$ where $\delta_i$is given by

$$\delta_i = \left(Q_{\theta^1_{\theta_{\text{c}}}}(g_i, a_i) - \hat{y}_i\right)^2 + \left(Q^2_{\theta_{\text{c}}}(g_i, a_i) - \hat{y}_i\right)^2. \tag{5}$$

Here g_i and a_i are sampled graph states and actions, respectively, $\hat{y}$ are target values, and $Q_{\theta^i_{\theta_{\text{c}}}}$, $i = 1, 2$ are the two critic networks each with parameters θ^i_c. The actor loss is defined through

$$J(\theta_{\text{a}}) = -\mathbb{E}_{\mathcal{D}}[Q_{\theta'}(g_i, \pi_{\theta_{\text{a}}}(g_i))], \tag{6}$$

where θ' is the target critic network parameters and π_g is the policy and θ_{a} its parameters. As in [15] and [6], we define the combined actor and critic loss as $Z(\theta) = L(\theta)+\lambda J(\theta_{\text{a}})$, where λ is an adaptively updated parameter weighting the balance between actor and critic updates.

III. Dimension reduction techniques

In this section we will summarize the linear PCA method and the non-linear methods t-SNE and UMAP.

A. The PCA method

The standard Principal Component Analysis (PCA) is a linear reduction technique based on performing eigenvalue decomposition of the symmetric covariance matrix $S = \frac{1}{N-1}X^{\text{T}}X$. Here, we have ordered the normalized N high dimensional observations $x_i \in \mathbb{R}^d$ into an $N \times d$ matrix X such that each column has mean zero and standard deviation one. S has d orthonormal eigenvectors $u_1, \dots, u_d$ corresponding to a set of sorted eigenvalues $\lambda_1 \geq \dots \geq \lambda_d$. For the PCA analysis we then select the k first eigenvectors which forms the $d \times k$ projection matrix P. The reduced low-dimensional $N \times k$ data matrix Y are then calculated by $Y = XP$.

B. The t-SNE method

The t-SNE method is a statistical tool for visualizing high dimensional data. The main idea is to find data representations in a low dimensional space such that points that have a high probability of being close in the high dimensional space also have a high probability of being close in the low dimensional space. The t-SNE method builds on the SNE method [16]. Both methods use a Gaussian kernel in the high dimensional space, but t-SNE uses the first order Student t-distribution for the low dimensional kernel. This means that if we have N high dimensional objects $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, the conditional probability of a data point x_j of being close to x_i is given by the probabilities

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2\right)}, \tag{7}$$

where σ_i is the variance for the Gaussian centered at x_i. Assuming symmetry, we get $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$. The low dimensional probabilities are then given by the map into N low dimensional objects $\mathbf{y}_i \in \mathbb{R}^d$, where d typically is 2 or 3, given by

$$q_{ij} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}}. \tag{8}$$

The low dimensional probability distribution Q is then found by minimizing the Kullback-Leibler divergence of the two distributions:

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{9}$$

where P is the probability distribution of points in the high dimensional space.

There are many pitfalls when using the t-SNE method, see for example [17]. One important parameter in this respect is the user-specified perplexity. The t-SNE method finds the variance σ_i through a binary search that results in P_i with a fixed value of perplexity, defined as $Perp(P_i) = 2^{H(P_i)}$, where the Shannon entropy $H(P_i)$, which describes the expected amount of information in the $p_{j|i}$ distribution, is defined by $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$.

In the original t-SNE paper [10], the authors state that the "performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50." However, "the perplexity is closely related to the size and density of the data" [18], which is also evident from the definition of the perplexity. The perplexity parameter could thus be interpreted as how much weight to put on local versus global data.

C. The UMAP method

The UMAP method is described in [11], see also [19] and [20]. One difference between t-SNE and UMAP concerns the initialization of the low dimensional distribution, where UMAP uses spectral embedding of the graph Laplacian, instead of a random initialization. In addition, the low dimensional optimizer use stochastic gradient descent instead of general gradient decent used by t-SNE. This leads to UMAP being generally faster and also being able to handle more than 2D/3D for the low-dimensional representation such that in can also be used within other clustering algorithms and not just for visualization.

Similarly to t-SNE, UMAP is also based on using the data to construct a high-dimensional distribution P and then find a low dimensional distribution Q that is similar to P. By the use of Riemannian geometry and fuzzy sets, the UMAP methodology establish a relationship between local metrics and the data through a graph consisting of the k nearest neighbors, where k is a hyperparameter (related to perplexity for t-SNE).

In UMAP, we calculate high dimensional unnormalized probabilities (related to the weights of edges in a fuzzy graph) by

$$p_{j|i} = \exp(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}), \tag{10}$$

where $1 \leq j \leq k$, $1 \leq i \leq N$, and ρ_i is defined as the distance to the nearest neighbor of x_i,

$$\rho_i = \min\{d(x_i, x_{i_j}) | 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\}. \tag{11}$$

The parameter σ_i is found by using binary search for the relation $\log_2(k) = \sum_{j=1}^{k} p_{j|i}$, and the symmetric unnormalized probabilities p_{ij} are defined by $p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} * p_{j|i}$. The low dimensional distance probabilities are given by

$$q_{ij} = (1 + a(y_i - y_j)^{2b})^{-1}, \tag{12}$$

where the parameters a and b are found by least-squares fitting to

$$q_{ij} \approx \begin{cases} 1 & \text{if} y_i - y_j \leq \text{min\_dist}, \\ e^{-(y_i - y_j) - \text{min\_dist}} & \text{if} y_i - y_j > \text{min\_dist}. \end{cases} \tag{13}$$

Here min_dist is a hyperparameter determining the minimum distance between points in the low dimensional embedding. Finally, the loss function is defined by the binary cross entropy

$$C = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + (1 - p_{ij}) \log \left(\frac{1 - p_{ij}}{1 - q_{ij}} \right). \tag{14}$$

IV. Visualization

In this section we will visualize the high dimensional embeddings of the graph neural network model presented and described in [6].[2] We also use the RTS as the power system case, and we use the same model parameters as in [6]. All training was done at the NTNU Idun HPC cluster [21] using a single V100 Nvidia 32 GB GPU card. We have used the Julia package "MultivariateStats.jl" for the PCA calculations, the Julia package "TSNE.jl" and the Python package "openTSNE" for the t-SNE calculation, and the Python package "UMAP" [22] for the UMAP calculations.

The PCA, t-SNE and UMAP methods are applied to a power system reliability Monte Carlo simulation with 1000 time steps, giving approximately 24 000 data points[3], as buses disconnected from the power grid will not be included in the time series. We have used a perplexity value of 200 for t-SNE and a $k = 200$ for number of nearest neighbors and $\text{min\_dist} = 0.5$ for UMAP. The only exception is in Fig. 3 and Fig. 4 where we have used 10 000 time steps with a total of approximately 240 000 data points, and a perplexity value of 50 and $k = 50$ for number of nearest neighbors, illustrating the robustness of the perplexity parameter.

A. Two groups of models

In [6], one group of models was trained by using some variant of a data augmentation strategy, Gaussian noise with $\mu = 0.0$ and $\sigma = 0.1$ was added to the curtailment costs which were then randomly permuted among the buses at each point in time when the load or system state changed. Each cost

[2]Code for some of the visualizations can be found at https://github.com/oysteinsolheim/pscc2024-visualizations.

[3]The exact number will vary from simulation to simulation. In the IEEE 24 bus case, bus 7 is a radial bus and experiences disconnections most frequently.

had a cut-off at 2.0 times the original cost. Gaussian noise with $\mu = 0.0$ and $\sigma = 0.2$ was also added to the maximum generator capacities and the loads. Finally, the noise-adjusted loads were randomly permuted among the load buses. The load flow and state of the power system were calculated based on these adjusted input variables. In this visualization section we will use both the noise-free model and the model obtained by using the data-augmentation strategy. Thus, we will consider the following two models:

- **Model 1**: Normally trained.
- **Model 2**: Trained by using noise/data augmentation.

Additionally, we have trained each set of the two RL-agents (model 1 + model 2) two times. One set of trained agents have been used in Fig. 3, Fig. 4, Fig. 8 and Fig. 9, another for the rest of the figures. The RL agents have been trained to the same level of accuracy in the Monte Carlo simulations, but due to different random initialization and the stochastic nature of the training process, the agents does not display identical results. However, both set of trained agents display similar generalizing capabilities.

B. Visualizing models with different generalizing capabilities

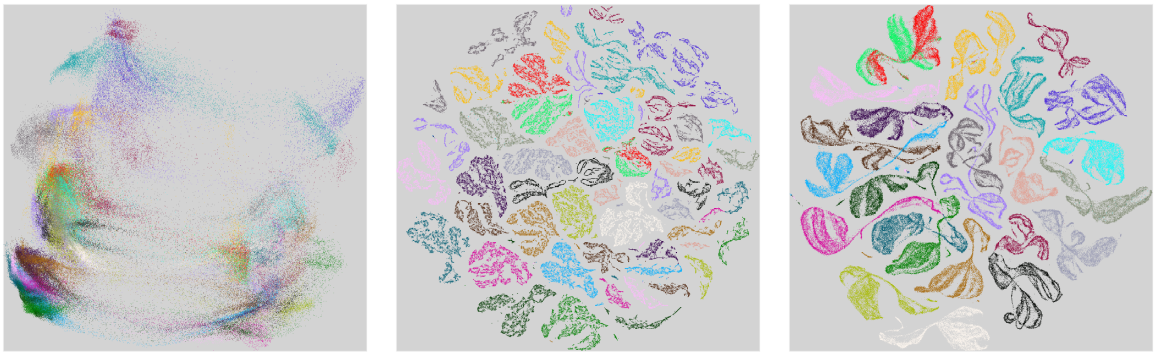
In Fig. 3 we visualize the last message passing layers for model 1 for three different dimension reduction techniques, PCA, t-SNE and UMAP, where each bus (1-24) is given a separate color. As expected, the PCA plot in Fig. 3a does not reveal any clustering of buses and the distribution of the individual bus node representations tend to overlap. This visualization is very similar to 2D PCA visualizations of the MNIST dataset [23], for example as presented in [24]. The t-SNE and UMAP visualizations in Fig. 3b and Fig. 3c, show both a strong clustering by bus and the t-SNE and UMAP representations look very similar, although for these hyperparameter settings, UMAP seems to keep the clusters larger and less divided.

In Fig. 4 we visualize the node embeddings for model 2, using t-SNE at three different model layers. Similarly, as in Fig 3, each bus is given a separate color. Considering the input layer in Fig. 4a, we notice two striking features. First, the long lines are the representations of load buses, where the only feature that changes value is the actual load at the bus. We hypothesize that the reason for this shape it because it is inherently one-dimensional, and the t-SNE-representations of these nodes reflect this. Second, the two circular point clouds near the middle and in the upper right part of the plot are representations of bus 11, 12, 17 and 24. These are buses which have neither load nor generator output connected, and thus none of the input feature values of these buses change during the simulations. The t-SNE-presentation thus represents only random noise around a single point. The rest of the buses are clearly grouped together, although many buses are split in several subgroups, and these are buses with either generation or both load and generation units connected to it. One important point to make here is that neither of the features in the input layers are updated or influenced by information in neighboring nodes.

In Fig. 4b and in Fig. 4c, visualizing the internal representation of the first and last MP layers, these initial modes of straight lines and ellipsoids disappear. Starting from MP layer 1, the internal representation of a bus includes information from neighboring nodes. As noted above, one striking feature of these plots are the clustering of the individual nodes. However, we also expect similar nodes (functionally and topologically) in the power system to have similar representations in the graph neural network. In the RTS case, bus 1 and 2 are two such buses, with similar installed generator capacities (2x10MW + 2x70MW) and approximately similar maximum load (108MW vs. 97MW), and their local grid structure bears great resemblance. These two buses are shown as red and green in Fig. 4 for model 2, which was trained with noise and in [6] proven to generalize better. Going from MP layer 1 in Fig. 4b to MP layer 4 in Fig. 4c we can see that as one proceeds through the MP layers, the internal representations of these two buses become almost indiscriminable. This stands in stark contrast to the representations of these two buses seen in Fig 3b.

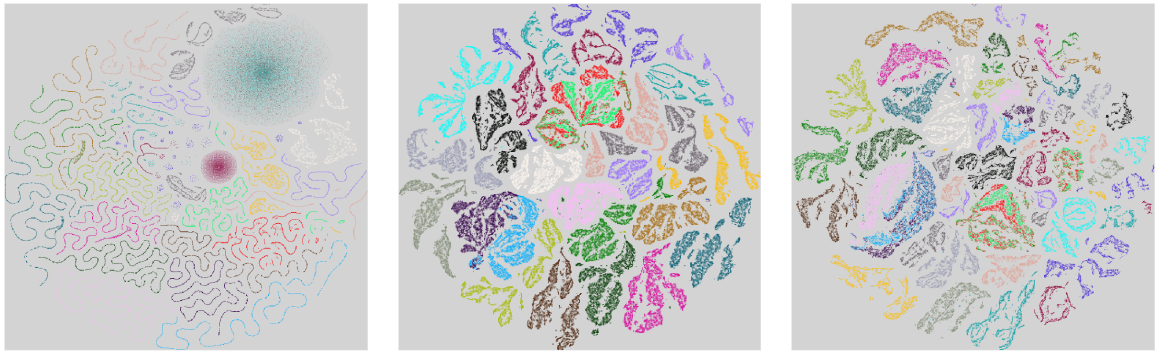
In Fig. 5 we visualize the representation for model 2, showing the t-SNE projection colored by the load value of each bus. The observations where load is zero is omitted from the figure. In the input layer, clusters of buses with only load connected are line shaped, and the lines are ordered by bus load value. We see the same ordering in some clusters of buses with both load and production connected, but not all. The pattern of increasing load across a cluster is also apparent in MP layer 2. As seen in the previous visualizations of the node input layer, the buses only connected to load are mainly grouped together in clusters shaped like thin lines in the t-SNE projection. The buses connected to both load and production are mostly wider clouds. Coloring observations by bus load reveals that the lines are ordered by the value of bus load. The direction of increasing load varies between buses. The fact that the ordering within a cluster is not a global feature should be expected since t-SNE axes are not directly interpretable and since global features are not necessarily preserved in a t-SNE projection.

In Fig. 6 we investigate the representations of three load buses situated in the 138kV part of the power system. Bus 4 and 5 have maximum load of 74MW and 71MW, respectively, whereas bus 6 has 136MW. The local topological structure is also relatively similar, as they all have 2 neighboring nodes and are all connected to either bus 1 or bus 2. The main difference between these buses is that bus 6 is connected to one of its neighboring nodes through a cable and not an overhead line. From Fig. 6a and 6b we see that only model 2 clearly represents bus 4 and 5 similarly in the first MP layer. We also notice here that neither of the models represent bus 6 as particularly similar to bus 4 and 5. However, this changes as we proceed through the layers. In Fig. 6c and 6d we see that model 2 represents bus 6 much more in line with bus 4 and 5. Model 1 still keeps bus 6 at a distance although in the smallest cluster in the lower part of Fig. 6c there does seem to be a stronger connection.



(a) PCA. (b) t-SNE. (c) UMAP.

Fig. 3: PCA, t-SNE and UMAP visualization of for MP layer 4, model 1. Each color represent one bus.



(a) Input layer. (b) MP layer 1. (c) MP layer 4.

Fig. 4: t-SNE visualization of for model 2. Each color represent one bus.

Similar observations are apparent for bus 9 and 10, visualized in Fig. 7. These two buses are also load buses with maximum active load at 175 MW and 195 MW, respectively. As with bus 4, 5, and 6, they share some local properties. For example, they are directly connected to each other and are both connected to either bus 4 or bus 5. They are both connected to five other nodes and are also both connected to transformers with identical properties. The clustering structure in the first MP layer appears quite similar, but when it comes to the last MP layer, the models clearly distinguish themselves. Model 2 represents these two buses in an almost indistinguishable manner. Model 1 does not have any noticeable overlap in its 2-dimensional t-SNE representation of these two buses.

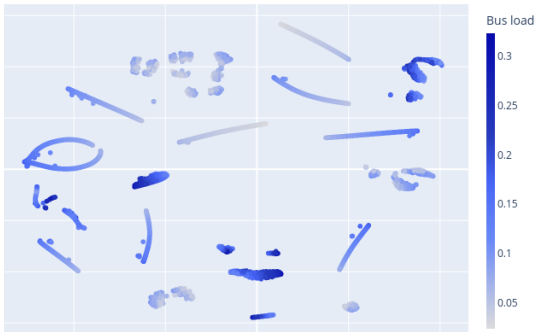
In Fig. 8 and Fig. 9 we show how PCA and UMAP represent the different models. Although the distribution of points are more dispersed in Fig. 8, we can see that in 8d the bus representations do actually confine themselves to a narrower region than the corresponding PCA representation for model 1. Similar observations as for t-SNE are made for the UMAP plots in Fig. 9, where the different visualizations for bus 1 and 2 for model 1 and 2 are especially striking.

The visualizations presented so far suggest that the two models, with different generalizing capabilities, do differ in their internal representations. In our view, these visualizations indicate that model 2 at least represents the buses with similar properties more effectively and comprehensibly. This insight is helpful in an iterative model-building process.

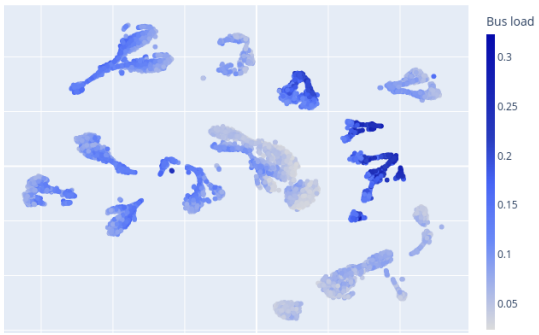
C. Visualizing development in training

In this section we will investigate how the behavior of the two models changes through the training stages. For both models, we collect the model parameters (neural network weights and biases) after 100 000 training steps (1 hour of training time) after 900 000 training steps (12 hours) and after 10 100 000 training steps (7 days). We denote these three models as *First*, *Middle* and *Last*. Fig. 10 visualizes how the final MP layer, layer 4, is represented for both model version at these different stages of training.

In Fig. 10a and 10d we show the representations of model 1 and model 2 for MP layer 4 after 1 hour of training, the *First* models. This short training time implies that the overall model is not able to make near-optimal decisions, partly because the critic part of the model is not accurate enough, and partly because the model has not had enough time to move away from the random initialization. To us, the structure of the clusters in the scatter plot for the first models resembles random noise around some key areas in the high dimensional space, rather than that of any specific structure emerging. We do, however, notice that model 1 and model 2 already at this early stage have

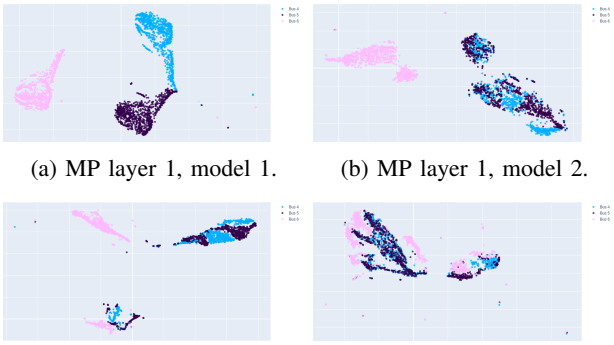


(a) Input layer.



(b) MP layer 2.

Fig. 5: t-SNE visualization of the input layer and MP layer 2 for model 2 with coloring based on bus load.



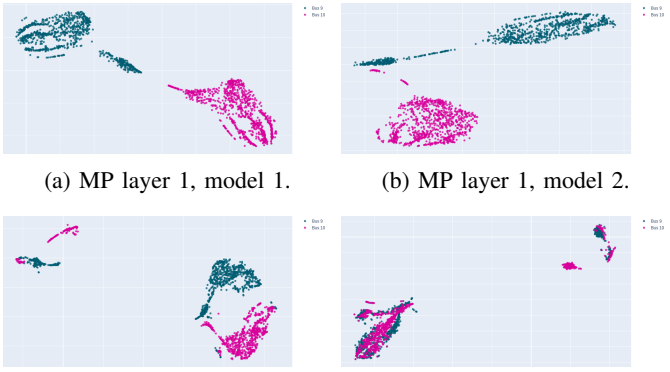
(a) MP layer 1, model 1. (b) MP layer 1, model 2.

(c) MP layer 4, model 1. (d) MP layer 4, model 2.

Fig. 6: t-SNE visualizations for bus 4 (light blue), bus 5 (dark blue) and bus 6 (pink).

different opinions on the representation of bus 6, as discussed in the previous section.

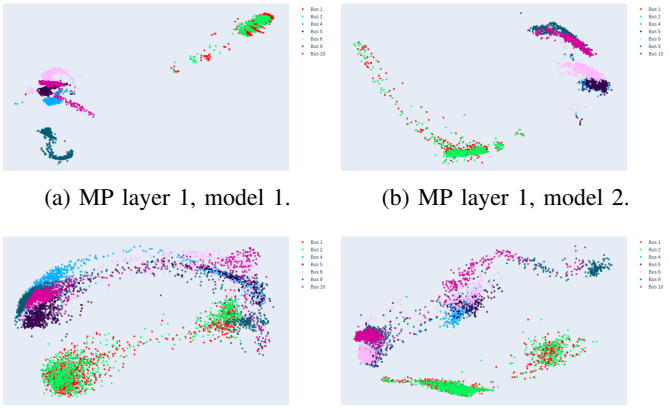
Moving forward to the *Middle* models in Fig. 10, we first notice much more structure in the clusters. For both the low dimensional representation of bus 1 and 2, and bus 9 and 10, we see clearly distinct areas of occupation. For example, both models suggest that bus 9 and 10 have two internal cluster regions. We further notice that the representations of bus 1 and 2 are now almost inseparable, at least for model 2. Model



(a) MP layer 1, model 1. (b) MP layer 1, model 2.

(c) MP layer 4, model 1. (d) MP layer 4, model 2.

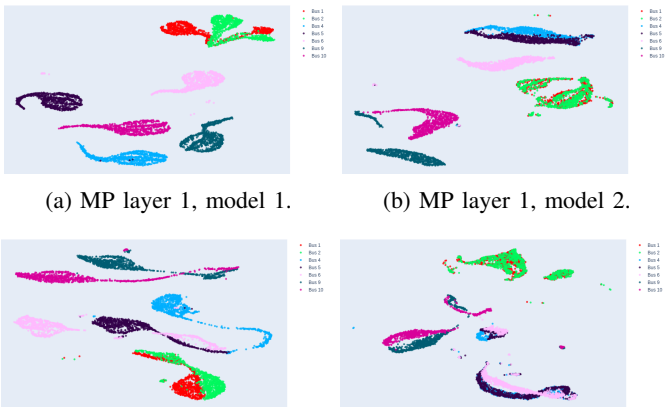
Fig. 7: t-SNE visualizations for bus 9 (green) and bus 10 (red).



(a) MP layer 1, model 1. (b) MP layer 1, model 2.

(c) MP layer 4, model 1. (d) MP layer 4, model 2.

Fig. 8: PCA visualizations of bus 1, 2, 4, 5, 6, 9 and 10.



(a) MP layer 1, model 1. (b) MP layer 1, model 2.

(c) MP layer 4, model 1. (d) MP layer 4, model 2.

Fig. 9: UMAP visualizations of bus 1, 2, 4, 5, 6, 9 and 10.

2 also has large overlapping regions of bus 4 and 5, while still keeping bus 6 close. Model 1 has separate regions for bus 4, 5 and 6, and bus 4 and 5 only have a small region of overlap, if any.

In Fig. 10c and 10f, the *Last* models, we see the final MP

layers having established themselves as discussed earlier. It is illuminating to see, for both models, how the representation changes to the better through the training stages.

D. Exposing bus types with higher perplexity

In this section we have studied the effect of extending the RTS with an extra bus that neither model experienced during training. As in [6], we have added a new bus between bus 21 and 22, with a load equal to the load at bus 19. The line characteristics between this new bus and bus 21 and 22 are equal to the characteristics of the line between buses 21 and 22. We set the curtailment cost for this load to \$4 000/MWh, meaning only bus 9 has lower curtailment costs. Following the discussion of the perplexity in Section III-B, we have experimented with much higher perplexity values, and especially a perplexity value of 2 000 has revealed interesting information.

In Fig. 11, we show an example of these high perplexity plots for the last MP layer of both model 1 and model 2. The main difference between these two plots is that for model 2 in Fig. 11b we see four clusters of data points. Each of these groups represent the bus type in the sense that the upper left cluster are buses with neither load nor generation connected to them, in the lower left corner a large cluster of load only-buses, in the upper right part are the three generator only-buses, and below this are buses with both generators and load connected to them. Importantly, model 2 also correctly clusters the new load bus together with the remaining load buses. The same is not apparent for model 1 in Fig 11a.

V. Summary and future work

In this paper we have visualized the internal representations of a graph-based actor-critic RL agent as it acts within a power system reliability environment. By using the dimension reduction techniques method, we have shown how agents with different generalizing capabilities have different internal representations, and we have studied how the input data to the agent is processed through the internal layers of the graph neural network components. Our visualizations have also pointed at how the representation changes to the better through the model training stages. An important question that we have sought to illuminate is how we can use these visualizations to guide our choice of graph neural network architecture and model size. These visualizations can also help us to understand and build confidence in the models developed. It is difficult to base the model architecture on the visualizations alone, but we believe that in an iterative process, these visualizations can play an important role in deciding which model has the desired properties, including the ability to generalize to new power system topologies. In our experience, the graph neural network is easier to analyze than a direct encoding, as the topology of the system is available to support our interpretations.

One future direction for research would be to study these visualizations for larger power system cases. This would increase the inherent variability of the input data to the graph neural network. One hypothesis would be that it might not be necessary to train the model with a data augmentation strategy, and visualizations could help decide to what extent that strategy would be needed.

We would also like to further explore additional grid expansion scenarios, beyond adding a single load bus. The visualizations done in this paper could help us find where the limits of a model's generalizing capabilities lie.

In this paper we have exclusively studied the internal representation of the actor part of the integrated actor-critic network. We think it would also be interesting to study the representations in the critic part of our model as well as the representations directly connected to the agent actions. Seeing the visualizations of all parts of the model could bring attention to modelling aspects previously not considered.

References

[1] X. Pan, T. Zhao, M. Chen, and S. Zhang, "DeepOPF: A Deep Neural Network Approach for Security-Constrained DC Optimal Power Flow," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1725–1735, 2021.

[2] L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent Developments in Machine Learning for Energy Systems Reliability Management," *Proceedings of the IEEE*, vol. 108, no. 9, pp. 1656–1676, 2020.

[3] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*. Wiley, 1995.

[4] O. S. Laengen and V. V. Vadlamudi, "DC and AC Contingency Solvers in Composite Power System Adequacy Assessment," in *Advances in RAMS Engineering*, D. R. Karanki, G. Vinod, and S. Ajit, Eds. Cham: Springer International Publishing, 2020, pp. 3–48.

[5] Ø. R. Solheim, B. A. Høverstad, and M. Korpås, "Deep reinforcement learning applied to Monte Carlo power system reliability analysis," in *2023 IEEE Belgrade PowerTech*. IEEE, 2023, pp. 01–08.

[6] ——, "Using graph neural networks in reinforcement learning with application to monte carlo simulations in power system reliability analysis," Sep. 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/Using_Graph_Neural_Networks_in_Reinforcement_Learning_with_application_to_Monte_Carlo_simulations_in_Power_System_Reliability_Analysis/24190764

[7] IEEE Committee Report, "IEEE Reliability Test System," *IEEE Trans. on Power Apparatus and Systems, PAS-98*, 1979.

[8] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933. [Online]. Available: https://api.semanticscholar.org/CorpusID:144828484

[9] W. S. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, pp. 401–419, 1952.

[10] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[11] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2020.

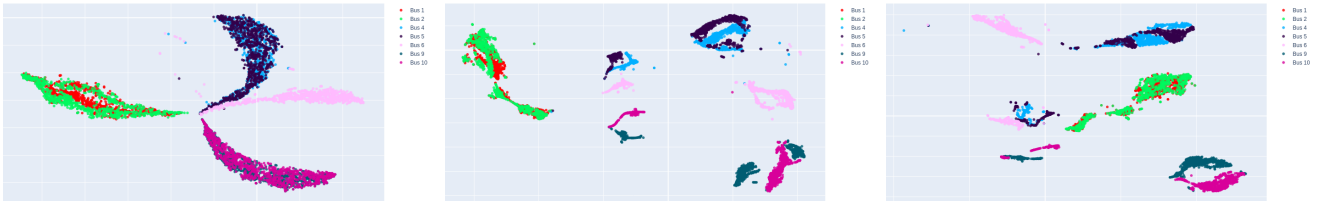
[12] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," pp. 1–23, 2017. [Online]. Available: http://arxiv.org/abs/1709.05584

[13] W. L. Hamilton, "Graph Representation Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

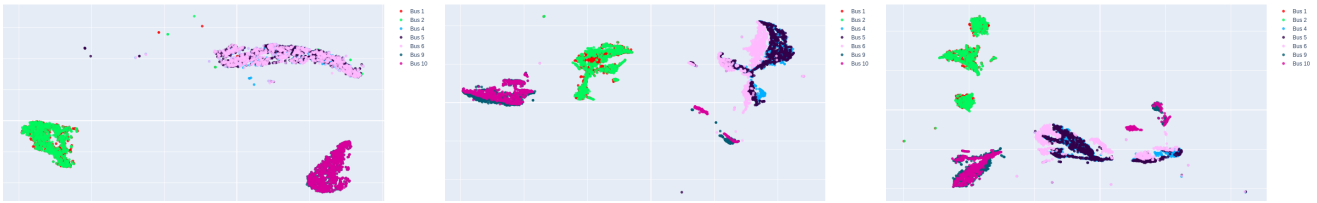
[14] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, "PRIORITIZED EXPERIENCE REPLAY," Tech. Rep., 2016.

[15] J. Zheng, M. N. Kurt, and X. Wang, "Integrated Actor-Critic for Deep Reinforcement Learning," in *Artificial Neural Networks and Machine Learning – ICANN 2021*. Cham: Springer International Publishing, 2021, vol. 12894, pp. 505–518. [Online]. Available: https://link.springer.com/10.1007/978-3-030-86380-7_41

[16] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. MIT Press, 2002.

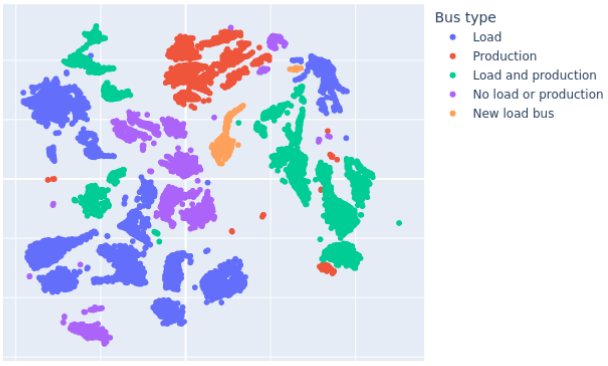


(a) *First*, MP layer 4, model 1. (b) *Middle*, MP layer 4, model 1. (c) *Last*, MP layer 4, model 1.

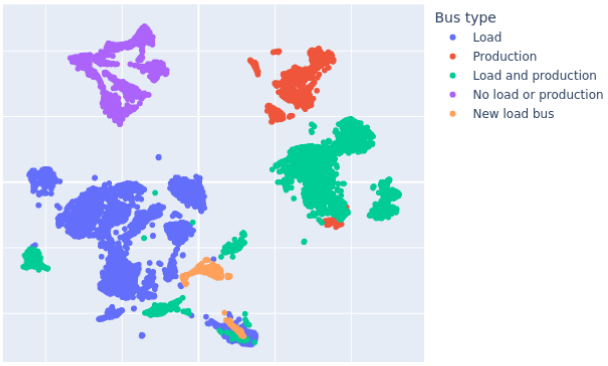


(d) *First*, MP layer 4, model 2. (e) *Middle*, MP layer 4, model 2. (f) *Last*, MP layer 4, model 2.

Fig. 10: t-SNE visualizations of bus 1, 2, 4, 5, 6, 9 and 10 during different stages (*First*, *Middle*, *Last*) of training.



(a) MP layer 4, model 1.



(b) MP layer 4, model 2.

Fig. 11: t-SNE visualization of model 1 and model 2 for RTS with extra bus.

[17] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-SNE effectively," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/misread-tsne

[18] C. Xiao, S. Hong, and W. Huang, "Optimizing graph layout by t-SNE perplexity estimation," *International Journal of Data Science and Analytics*, vol. 15, no. 2, pp. 159–171, Mar. 2023. [Online]. Available: https://doi.org/10.1007/s41060-022-00348-7

[19] Andy Coenen and Adam Pearce, "Understanding UMAP." [Online]. Available: https://pair-code.github.io/understanding-umap/

[20] Nikolay Oskolkov, "How exactly UMAP works," 2019. [Online]. Available: https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

[21] M. Själander, M. Jahre, G. Tufte, and N. Reissmann, "EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure," 2019.

[22] T. Sainburg, L. McInnes, and T. Q. Gentner, "Parametric UMAP embeddings for representation and semisupervised learning," *Neural Computation*, vol. 33, no. 11, pp. 2881–2907, 2021.

[23] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[24] M. Min, D. Stanley, Z. Yuan, A. Bonner, and Z. Zhang, "A deep non-linear feature mapping for large-margin kNN classification," Dec. 2009, pp. 357–366.

[25] A. Abedi, J. Beyza, F. Romerio, J. A. Domínguez Navarro, and J. Yusta, "MCDM approach for the integrated assessment of vulnerability and reliability of power systems," *IET Generation, Transmission and Distribution*, Oct. 2019.

APPENDIX

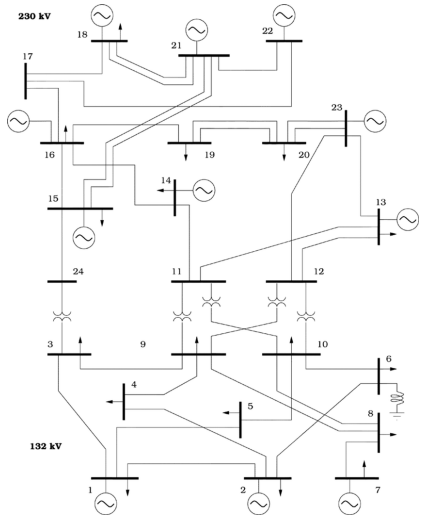


Fig. 12: The IEEE Reliability 24 bus test system (RTS) [25].