Towards Explaining Deep Neural Networks Through Graph Analysis

Vitor A. C. Horta and Alessandra Mileo

Insight Centre for Data Analytics at Dublin City University, Dublin

Abstract. Due to its potential to solve complex tasks, deep learning is being used across many different areas. The complexity of neural networks however makes it difficult to explain the whole decision process used by the model, which makes understanding deep learning models an active research topic. In this work we address this issue by extracting the knowledge acquired by trained Deep Neural Networks (DNNs) and representing this knowledge in a graph. The proposed graph encodes statistical correlations between neurons' activation values in order to expose the relationship between neurons in the hidden layers with both the input layer and output classes. Two initial experiments in image classification were conducted to evaluate whether the proposed graph can help understanding and explaining DNNs. We first show how it is possible to explore the proposed graph to find what neurons are the most important for predicting each class. Then, we use graph analysis to detect groups of classes that are more similar to each other and how these similarities affect the DNN. Finally, we use heatmaps to visualize what parts of the input layer are responsible for activating each neuron in hidden layers. The results show that by building and analysing the proposed graph it is possible to gain relevant insights of the DNN's inner workings.

Acknowledgements

The Insight Centre for Data Analytics is supported by Science Foundation Ireland under Grant Number 17/RC-PhD/3483.

Keywords: Explainable AI · Deep learning · Graph analysis.

1 Introduction

In deep learning, given an extensive set of examples, Deep Neural Networks (DNNs) can be efficiently trained to solve complex problems, such as speech and language processing and image recognition. These tasks often involve high-dimensional data and, while most traditional machine learning methods (ML) fail in these cases, DNNs have been shown to achieve good performance [11].

Although this approach has proven to be powerful, there are still several challenges and open issues regarding the use of this technology. A known gap in the field is the lack of interpretability and the problem is that it is a difficult task to determine why a DNN makes a particular decision [16]. In some domains, such as precision medicine and law enforcement, it is not sufficient to know only the final outcome and how good it is, but it is also important to explain the whole decision making process. Besides explaining the decisions, understanding the inner workings of a DNN could be useful to improve and extend DNN models and architectures, as well as provide useful knowledge that can be reused for transfer learning and reasoning in deep networks.

Some of the challenges involved in this task are the different and complex architectures of deep networks and the difficult to extract knowledge from neurons in hidden layers. This problem has being known and explored since the 90's, when DNNs were not yet as successful as they are now, due to the lack of available training data and computational resources. Existing approaches try to explain DNNs through rule extraction [3][5][16], feature visualization [14][4], measurement of input contribution [1][12] among others. These approaches do not generally tackle the hidden layers and some do not apply to convolutional layers. Instead, some of them use the neural network as an oracle and use its predictions to build other ML models such as decision trees, while others explain only the influence of inputs in feature space over the model outcome. These can be seen as limitations in their attempt to provide a general mechanism to extract knowledge from deep learning models and use it to explain the model itself.

In this paper a novel way to extract and represent knowledge from trained DNNs is proposed. We use activation values to find correlations between neurons at each layer in the DNN, including input layer, dense and convolutional layers and output classes. We represent this information in a graph where nodes correspond to neurons that are connected through weighted edges based on correlations in their activation values. Our hypothesis is that knowledge contained in the proposed graph is compatible with knowledge acquired by the DNN and by using graph analysis tools we can gain insights on how the model works.

In this initial work we focus on evaluating whether the proposed graph is capable of representing knowledge about how the hidden layers of the DNN work. Besides the proposed graph, three main contributions of this paper are: (i) to show how to exploit the proposed graph in order to detect which neurons are relevant to each output class; (ii) to use graph analysis for detecting groups of classes with high similarity; (iii) to visualize features in hidden layers and understand the role of neurons in both fully connected and convolutional layers.

Two initial experiments on image classification were conducted to demonstrate our approach. We show that by querying the proposed graph it is possible to find hidden layer neurons highly correlated to output classes and that, when activated in the DNN, these neurons have a high contribution to the prediction value of these classes. Also, graph analysis is performed over the proposed graph to find groups of classes that are more similar to each other. We show that classes with high number of shared neurons (overlapping nodes in the graph) are responsible for most mistakes in the DNN. This is an evidence that overlapping nodes in the proposed graph might indicate which neurons in the DNN should be considered for a fine-tuning process in order to improve the model to better distinguish the overlapping classes. In addition, edges between hidden and input layers allow us to visualize each neuron in both convolutional and dense layers through heatmaps, which helps identifying their role in the decision process.

The rest of the paper is organized as follows. Section 2 shows existing approaches to extract knowledge from neural networks. Section 3 presents the proposed methodology and explains how to generate the proposed graph. In Section 4 a feasibility study is conducted in two experiments on image classification. Section 5 provides our conclusions and what can be done in future works.

2 Related Works

Explaining neural networks is a difficult task that attracts researchers' attention since the 90's and many studies attempt to address this problem from different perspectives. One common approach is to measure the contribution of each input to the outputs of the neural network [1], which helps understanding the model's decision process. This approach can also be extended for measuring the contribution of neurons in the hidden layer [12], but one limitation in the existing works is that they can not be applied to convolutional layers.

Another way to understand the underlying mechanism of neural networks is through rule extraction. In this case some works adopt a decompositional strategy, which extract rules by examining activation and weights in the neural networks [3][8][9]. Another rule extraction strategy is the pedagogical, which uses the decisions made by DNNs to extract rules without exploring the model architecture [5]. It is also possible to use a hybrid of these two approaches, which is the eclectic strategy [13]. The problem is that methods that uses decompositional strategies can not be directly applied to convolutional layers. On other hand, pure pedagogical methods can be applied to any neural network because they do not rely on the architecture. However, they do not explain the inner workings of hidden layers since they keep the DNN as a black box.

A third approach is to use visualization techniques [14][4]. Visualizations can be used to understand which parts of the input are relevant for the model's predictions. It is also possible to understand the role of filters in convolutional layers by visualizing them. Still, performing these visual analyses might require a lot of human intervention, which limits their scalability.

The graph representation proposed in this paper differs from existing works in two main aspects. Firstly, our proposal can represent both convolutional and fully connected layers in any depth of DNNs. Second, it allows the use of automatic methods such as graph analysis tools to discover interesting patterns about the internal workings of DNNs. In this sense, a recent work [6] that proposes a graph representation for embedding vectors is the most similar to our proposal. One difference is that because our graph uses weighted edges, our approach can represent both negative and positive relationships with less nodes than their approach, which uses different nodes for positive and negative contributions. We also include the input layer in our representation, which allow us to visualize of neurons in the hidden layer to understand their roles in the decision process.

3 General Approach: how to generate the co-activation graph

The main goal of this paper is to extract and represent knowledge from trained DNNs in order to better understand how the hidden part of the model works. We propose a graph that connects every pair of neurons of any type (fully connected or convolutional) and located in any layer of the neural network. This section presents the general idea on how to build the proposed graph and in next section we conduct preliminary experiments to assess the feasibility of our approach.

In the proposed graph nodes correspond to neurons in the DNN and weighted edges represent a statistical correlation between them based on their activation values. We call this graph as *co-activation graph*, since the relationships between its nodes represent how their activation values are correlated. The main idea of the co-activation graph is to create a relation between neurons in any depth of hidden layers to neurons in the input feature space and output classes, since the latter are more comprehensible for humans. Given a trained DNN we can generate a co-activation graph using the three steps below:

Extract activation values: the first step is to feed the DNN with some data samples and extract activation values for each prediction. For dense layers this process is straightforward because each neuron outputs a single activation value. Filters in convolutional layers, instead, will output multiple values since they have different activation values for each region in the input. To overcome this and make our approach work for convolutional layers, the average pooling technique is applied to extract a single value for convolutional filters. Although some spatial information is lost in this process it allows the extraction of a single activation value for each filter while keeping the dimensionality low [7].

Define and calculate edge weights: After collecting the activation values for neurons in each layer, the next step is to define the strength in their relationships. We use the activation values to calculate a statistical correlation between each pair of neurons. In this work we chose to use Pearson coefficient as an initial correlation measure and thus edge weights vary in range [-1,1].

Build and analyse the co-activation graph: In this third and final step the co-activation graph can be built and analysed. The nodes represent neurons of any layer in the DNN and weighted edges indicate the correlation between their activation values. We can then explore the graph structure and use graph analysis tools to understand relationships between neurons in hidden layers with the input layer and output classes.

In next section we do a preliminary evaluation of our approach to demonstrate whether the co-activation graph is a suitable way to interpret and understand the way the DNN works and the knowledge encoded in its model.

4 Preliminary Experiments and Initial Results

To evaluate whether the co-activation graph can help understanding how deep learning models work we have conducted two experiments. This section first introduces the datasets and how the experiments were conducted. Then, three analyses are performed, which are the main contributions of this work. In the first analysis we use the co-activation graph to check how relationships between nodes in hidden layers and classes in the output layer can reveal neurons that highly impact the prediction value for these classes. This is the first step to evaluate whether the co-activation graph representation is compatible with the knowledge within the trained DNNs, since it can be used to identify which neurons have high contribution over the decisions taken by the model.

In a second analysis we show how graph analysis performed over the co-activation graph can give insights about the internal workings of the corresponding DNN. For this we use a community detection algorithm to see what properties in the DNN can be revealed by analysing the community structure of co-activation graphs. The third analysis aims to discover the role of each neuron in the DNN. We use relationships between nodes in the input layer and nodes in hidden layers to produce heatmaps for neurons in both convolutional and fully connected layers. The three analyses are used to evaluate the feasibility of using co-activation graphs to extract and represent knowledge from trained DNNs.

4.1 Datasets and experiments setup

In our experiments we used two well known datasets: MNIST handwritten digits [10] and MNIST fashion [17]. Both datasets contain 70000 images separated in ten classes. The classes in *handwritten digits* dataset refer to digits from 0 to 9 while classes in *fashion* dataset are related to clothes. The DNN used for *handwritten digits* contains two convolutional layers and three fully connected layers and the DNN used for *fashion* dataset has three convolutional layers and two fully connected layers. These models achieved an accuracy higher than 97% and although it is possible to find more accurate models we chose to use these ones since we also want to analyse the reason behind mispredictions.

To build a co-activation graph for each DNN the three steps described in the previous section were applied. We have first fed the DNNs with data samples from the testing set in order to extract activation values for each neuron. Then, we calculated the Pearson correlations between those neurons and built a co-activation graph for each DNN. Figure 1 shows the two graphs, where yellow nodes represents neurons in the output layer (classes) and blue nodes are neurons in hidden layers. The input layer was omitted in this visualization for clarity.

4.2 Analysing the relation between co-activation graphs and DNNs

We can see from Figure 1 that for both graphs the visualization technique placed some classes (yellow nodes) very close to each other while keeping other classes more distant. This might indicate that co-activation graphs have some potential to have a community structure. However, before performing such graph analysis it is important to analyse whether these graphs are being able to represent knowledge from their respective DNNs. To check this we have analysed how we can explore the structure of co-activation graphs to understand the DNNs' decision process.

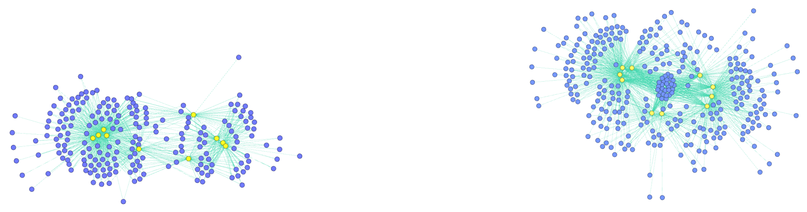


Fig. 1. Visualizations for co-activation graphs (handwritten digits left and fashion right) made on Gephi. Blue nodes are representing neurons in hidden layers and yellow nodes are representing neurons in the last layer (output classes)

By querying the graph it is possible to collect for each class a set of its most correlated neurons based on relationship weights. Since strong relationships indicate high positive correlations in the activation values, it is expected that when these highly correlated neurons are activated in the DNN, they will have a high contribution to increase the prediction value of their classes. For example, if we activate neurons most correlated to a class *sandal*, it is expected that the DNN will output *sandal* as a prediction.

To show this we extracted and manually simulated the activation of the *top* k most correlated neurons for each class. These neurons were extracted from the first fully connected hidden layer after the convolutional layers, which is not directly connected to the output and thus more difficult to interpret. We expect that when activating the top k neurons, the model will output a prediction with the respective class. Figure 2 show the results for both datasets. It is possible to see that for the *fashion* dataset when we activate the *top* 7 neurons we get the expected prediction for all the classes and for *handwritten* dataset the model predicts the expected class 80% of the times.

This result indicates that we can query the co-activation graph to find neurons with a high impact over the prediction values for each class. Therefore, it is possible to extract relevant neurons for each class from the co-activation graph even though they are in a layer located in the middle of the DNN, which is the first contribution of this paper. In a subsequent step we performed other graph analysis to see how the co-activation graph can be used to help getting a better understanding of the deep learning models.

4.3 Community structure analysis

Following the intuition from Figure 1, we analysed the community structure in this graph to see if it can help detecting classes that are similar from the DNN point of view. The Louvain community detection algorithm [2] was chosen because besides being a well established algorithm, this method also outputs a modularity coefficient that can be used to check how the community structure differs from random graphs. The value of modularity varies in range [-1,1] and higher values indicates that connections between nodes in same community are stronger than nodes in different ones.

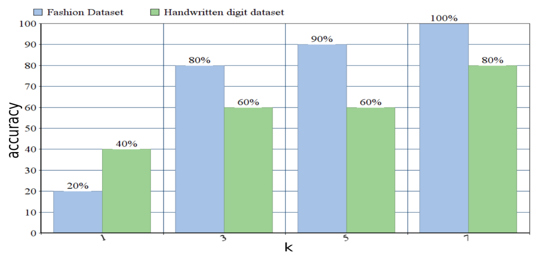


Fig. 2. Result acquired after extracting *top k* most correlated neurons for each class in the co-activation graph and activating then in the DNN. The accuracy indicates how many times the expected class was predicted.

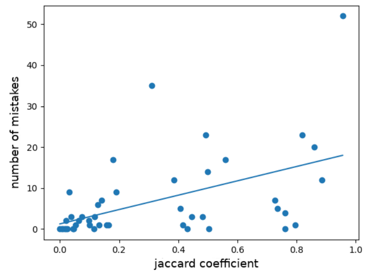
We used the algorithm over the connections between nodes representing neurons in the hidden layers and nodes representing classes in the output layer. Table 1 shows the detected communities and classes contained in each of them. For the *fashion* dataset it can be noted that classes in same communities have a similar semantic meaning. We can notice this because classes like *pullover* and *coat* were put in same community while *sandals* and *sneakers* are in another one. For the *handwritten* it is not possible to conclude the same, since semantic of digits are less clear. It is important to note that for both datasets the modularity was higher than 0.4 which means that found communities are denser regions although they are not totally distinguishable.

Table 1. Classes and their communities in both datasets.

	Fashion	Handwritten Digits
Community	Classes	Classes
C1	T-shirt/Top; Pullover; Coat; Shirt	0;2;4;6
C2	Trouser; Dress;	5;7;8;9
C3	Sandal; Sneaker; Bag; Ankle Boot	1;3
Modularity	0.413	0.45

After noticing that some classes are more similar to each other we investigated the similarities between classes and how they impact the DNNs decisions. The jaccard similarity coefficient was calculated based on the overlaps (shared nodes) between each pair of classes. Then, we collected the number of mistakes between them by counting for every pair of classes A and B how many times the model wrongly predicted A when the correct answer was B, or the other way around.

Figure 3 shows that there is a positive correlation between class similarity and the number of mistakes involving them. This shows that pairs of classes with many overlapping nodes in the co-activation graph tend to cause a high number of mistakes in the DNN. This brings to our second contribution, as we can analyse communities and node similarities in the co-activation graph to understand which classes are similar from the DNN point of view. Also, this indicates that overlapping nodes between two classes might be recommended for tuning process in order to improve the model to better distinguish these classes. However, since our focus is to analyse the fidelity of the graph representation, evaluating the latter is beyond the scope of this work.



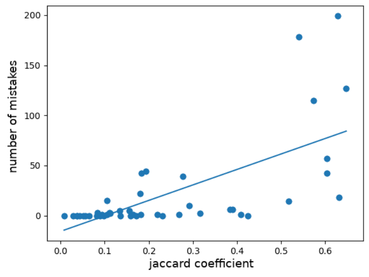


Fig. 3. Correlations of mistakes between two classes and their jaccard similarity. The Pearson correlation is 0.5042 for handwritten digits (left) and 0.6625 for fashion (right)

4.4 Visualizing neurons in hidden layers

So far we have only analysed the relationships between neurons in the hidden layer and neurons in the output layer. A third analysis was then conducted to see if the co-activation graph can help understand how nodes in the input layer impact the activation of neurons in the hidden layer. To do that, we used the edges between nodes in hidden layer and input layer to plot heatmaps for each neuron. Thus, colors in the heatmap are defined based on these edge weights.

Figure 4 shows heatmaps for three nodes strongly connected to classes *sneaker*, *trouser* and *pullover* respectively. We can see from these heatmaps that the inputs correlated to theses neurons are clearly related to their respective strongly connected classes. Figure 5 shows instead nodes that are overlapping between two or more classes. Note that for the nodes shared by only two classes (left) it is still possible to visualize some aspects of their classes (*bag* and *trouser*) in the heatmap. Yet, it is more difficult to identify which classes are related to node on right image in Figure 5, since it is overlapping in three different classes. This is another evidence that overlapping nodes in the co-activation graph indicate neurons that are not being able to make a distinction between some classes in the DNN. Thus, they might be considered for a tuning process in order to improve the model to separate these classes and reduce possible classification errors.

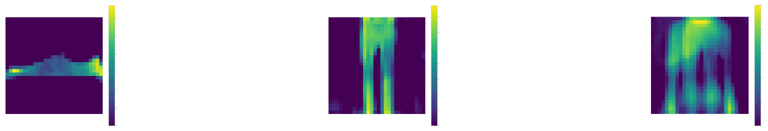


Fig. 4. Heatmaps from nodes correlated to *sneaker*, *trouser* and *pullover* respectively.

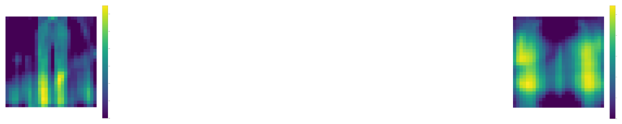


Fig. 5. On the left side is a heatmap from a node overlapping in classes *bag* and *trouser* and on right side a heatmap from node in *ankleboot*, *pullover* and *bag*.

This brings to our third contribution, as it is possible to visualize what parts of the image activate each neuron. Also, the heatmaps confirm the relationship between nodes and their most correlated classes.

To summarize the achieved results, we have first shown that by querying the co-activation graph we can detect which neurons have high impact over each class in the DNN. Then we found that by using community detection methods in this graph it is possible to detect groups of classes with similar semantics. In addition, we showed that classes with high similarity (shared nodes in the graph) are responsible for most mistakes in the model. Finally we used the relationships between neurons in hidden layers and input layer to visually analyse their roles.

5 Final remarks

In this work we focused on the problem of explaining deep learning models. We proposed a method to extract knowledge from trained DNNs and to represent it as a graph, which is called co-activation graph. In the co-activation graph, nodes represent neurons in a DNN and weighted relationships indicate a statistical correlation between their activation values. Thus, it is a representation that connects neurons in any layer of the neural network, including hidden (convolutional and dense), input and the output layer through statistical correlations.

A preliminary evaluation was conducted to check whether the co-activation graph representation is compatible with the knowledge encoded in the DNN. Two experiments showed that by exploring the graph structure we can find neurons that have a high contribution for predicting some classes. This is crucial for better understanding: since these neurons highly impact the prediction values for some classes we can analyse them to explain decisions taken by the model.

Then, we showed that community detection methods over this graph can detect classes that are more similar from the DNN point of view, and that classes highly overlapping in the co-activation graph are responsible for most mistakes in the model. This is another interesting finding as it indicates that overlapping nodes in the graph can be considered for a tuning process for model improvement in order to better distinguish similar classes. At last we used the co-activation graph to show that relationships between hidden layers and the input layer can be used to visually explain what parts of the input layer activate each neuron in the hidden layer regardless of its type and location. By plotting heatmaps for each neuron we showed how this can help identifying their roles in the DNN.

Although we still consider these as initial results, they demonstrate the potential of using co-activation graphs to extract knowledge from DNNs. Our goal is to investigate this further and in a more formal way. As part of our next steps, we plan to consider other correlation coefficients to see if a non-linear correlation gives a better representation than Pearson coefficient. We also want to explore the notion of graph centrality to clarify whether nodes with a high centrality measure plays a specific role in the DNN. Community structure should also be better investigated especially in terms of their overlaps. Since some classes have shown to have a high similarity, methods for detecting overlapping communities might give better clusters than disjoint ones. Finally, experiments with bigger neural networks such as VGG16 CNN [15] should be conducted to assess if our approach can extract knowledge from more complex networks and how this knowledge can be used to improve the model.

References

1. B. Bartlett, E.: Self determination of input variable importance using neural networks. Neural, Parallel Scientific Computations **2**, 103–114 (03 1994)
2. Blondel, V., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics Theory and Experiment **2008** (04 2008). https://doi.org/10.1088/1742-5468/2008/10/P10008
3. Chan, V., Chan, C.W.: Development and application of an algorithm for extracting multiple linear regression equations from artificial neural networks for nonlinear regression problems. In: 2016 IEEE 15th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC). pp. 479–488 (Aug 2016)
4. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Technical Report, Univerist de Montral (01 2009)
5. Junque de Fortuny, E., Martens, D.: Active learning-based pedagogical rule extraction. IEEE transactions on neural networks and learning systems **26** (01 2015)
6. Garcia-Gasulla, D., Vilalta, A., Parés, F., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., Suzumura, T.: Building graph representations of deep vector embeddings. CoRR **abs/1707.07465** (2017), http://arxiv.org/abs/1707.07465
7. Garcia-Gasulla, D., Vilalta, A., Parés, F., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., Suzumura, T.: An out-of-the-box full-network embedding for convolutional neural networks. 2018 IEEE International Conference on Big Knowledge (ICBK) pp. 168–175 (2018)
8. Kim, D., Lee, J.: Handling continuous-valued attributes in decision tree with neural network modelling. In: ECML (2000)
9. Krishnan, R., Sivakumar, G., Bhattacharya, P.: A search technique for rule extraction from trained neural networks. Pattern Recognition Letters **20**(3), 273–280 (1999)
10. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
11. Liu, B., Wei, Y., Zhang, Y., Yang, Q.: Deep neural networks for high dimension, low sample size data. pp. 2287–2293 (08 2017). https://doi.org/10.24963/ijcai.2017/318
12. Mak, B., Blanning, R.W.: An empirical measure of element contribution in neural networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **28**(4), 561–564 (1998)
13. Mohamed, M.H.: Rules extraction from constructively trained neural networks based on genetic algorithms. Neurocomput. **74**(17), 3180–3192 (Oct 2011)
14. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR **abs/1312.6034** (2013), http://arxiv.org/abs/1312.6034
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), http://arxiv.org/abs/1409.1556
16. Towell, G.G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. Mach. Learn. **13**(1), 71–101 (Oct 1993)
17. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)