STEP-WISE SENSITIVITY ANALYSIS: IDENTIFYING PARTIALLY DISTRIBUTED REPRESENTATIONS FOR INTERPRETABLE DEEP LEARNING

Botty Dimanov & Mateja Jamnik
Department of Computer Science and Technology
University of Cambridge
Cambridge, CB3 0FD, UK
{botty.dimanov,mateja.jamnik}@cl.cam.ac.uk

ABSTRACT

We introduce a novel framework for interpreting Deep Neural Networks (DNN) classification decisions that constructs a dependency graph between the relevant neurons across the network to enhance the understanding of the interactions between DNN's internal features. For instance, class-specific dependency graphs share subgraphs across 10 classes, which can cluster the classes into semantically related groups. We propose to use these subgraphs to identify partially-distributed representations. Further, a class-specific dependency graph can compress a sub-optimal DNN in half into a binary classifier for that class. Our work enables the building of more sophisticated techniques that are capable of translating black-box DNNs into interpretable decision trees. Such capabilities would empower DNN developers to address incorrect predictions through informed architectural and design choices.

1 INTRODUCTION

Deep Neural Networks (DNNs) are difficult to interpret due to their highly complex non-linear and interconnected nature. The lack of transparency is a threefold problem. First, it inhibits adoption, especially in industries under heavy regulation and with a high cost of errors. Second, it prevents us from utilising the insights gained from the models for further knowledge discovery. Third, it makes debugging existing models difficult and hampers development progress. Interpretability is crucial to DNN debugging since it can give an intuition of the circumstance under which a DNN might fail and the reasons behind this potential failure, and it can also facilitate regression testing by illustrating the effects of a debugging intervention.

DNN interpretability can be gained from a human-interpretable explanation of the reasons behind the network's choice of output (Ribeiro et al., 2016; Doshi-Velez & Kim, 2017). In a DNN the basis for a decision is encoded in features either as one neuron – local representation; or as a set of neurons – partially-distributed representation (PDR) (Li et al., 2016; Fong & Vedaldi, 2018).

The identification of PDRs and their interactions remains the main hindrance to end-to-end interpretability systems (Olah et al., 2018). Once identified, PDRs enable us to give much finer-grained explanations (e.g., an image is a shark because the network detected a sea, sharp teeth, a long fin, etc.). In this paper, we introduce our novel technique, step-wise sensitivity analysis (SSA), which produces *statistical topological interpretability*. That is, we analyse the network's properties as a directed graph over various inputs to produce a dependency graph between neurons across the layers. The dependency graph highlights the relationships between adjacent layers that are pertinent to the decision, and how these relationships are formed in each layer and across all layers to form a feature representation. Furthermore, the dependency graph is capable of distilling a DNN into a class-specific classifier, which can be further interpreted in more detail or combined into a more powerful ensemble model.

2 Related Work

Existing effort dedicated to DNN interpretability has three main limitations. First, current approaches overlook the partially distributed nature of DNNs and either assume purely local representations (Erhan et al., 2009; Simonyan et al., 2013), or fully-distributed representations (the entire layer) (Mahendran & Vedaldi, 2015). Second, they focus on the input-output relationship treating the entire network either as a black-box function with an input, output and parameters (functional) (Simonyan et al., 2013; Zeiler & Fergus, 2014; Zintgraf et al., 2017; Shrikumar et al., 2017) or as graph (topological) (Landecker et al., 2013; Bach et al., 2015; Montavon et al., 2017). Layer-wise-relevance propagation (LRP) (Bach et al., 2015) and Deep Taylor Decomposition (Montavon et al., 2017) compute a relevance (importance score) between neurons in a layer-by-layer fashion, which can be plugged into our four-step process to redistribute the relevance only to a small number of relevant neurons. Third, model-centric approaches (e.g., activation maximisation (Erhan et al., 2009) and inversion (Mahendran & Vedaldi, 2015)) produce an explanation that can be generalised to every data point. On the other hand, instance-specific methods (Erhan et al., 2009; Simonyan et al., 2013; Bach et al., 2015) give details to reason about particular mistakes or edge-cases operate on the level of a single instance. Our approach aggregates instance-specific results to get a model-centric model explanation, similarly to (Robnik-Šikonja & Kononenko, 2008; Zintgraf et al., 2017).

Net2Vec (Fong & Vedaldi, 2018) proposes a method for identifying and interpreting PDRs by optimising the combinations of filters for classification and segmentation on proxy ad hoc tasks. In contrast, our method identifies the dependencies between neurons and it can ascertain the neuron relevance using the original data set without the need to compile explanatory datasets for various problems. Our work is comparable to excitation backpropagation (Zhang et al., 2016) and PatternNet (Kindermans et al., 2017), which distribute the relevance to a subset of neurons. Both approaches focus on improving the heatmap quality either through a probabilistic winner-take-all sampling approach (Excitation backpropagation) or through the product of weights and activations (PatterNet). In contrast, we investigate the dependencies between PDRs; hence, we deploy a more generalisable linear Taylor approximation and a statistical analysis over multiple inputs and neurons to restrict the relevant neurons.

3 Step-wise Sensitivity Analysis

Step-wise Sensitivity Analysis (SSA) is a DNN interpretability framework that iteratively follows four steps through the DNN architecture. The basic idea is illustrated in Figure 1. Given a DNN classifier, a set of datapoints, and a set of target labels in the form of relevant neurons in the top layer $n \in \mathbb{S}$, start from the top layer and follow the four steps in Algorithm 1 to produce a set of b relevant neurons $\mathbb{S}^{l-1}$ from the lower layer. Then $\mathbb{S} := \mathbb{S}^{l-1}$ and repeat until the input layer.

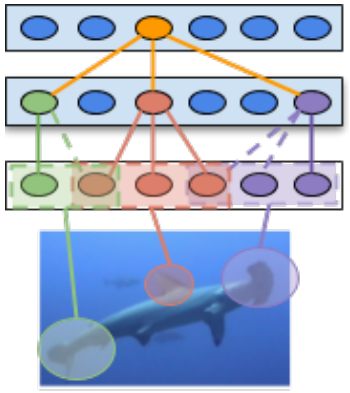


Figure 1: (a) A sketch of how Step-wise Sensitivity Analysis identifies PDRs of relevant neurons that can be used to provide interpretation for a shark prediction.

Algorithm 1 consists of the following steps. Step I. computes the relevance between all neurons between two adjacent layers. Step II. aggregates across datapoints to weight the neuron relevance w.r.t the datapoints under investigation. Step III. aggregate across upper-layer neurons to weight the layer-wise importance of a neuron. This is a proxy of how much a neuron is reused and shared across upper-layer neurons, hence it is a proxy for the likelihood of a neuron being part of a PDR. In order to separate the relevant neurons into distinct PDRs, Step III. can be skipped to preserve the mapping between an upper-layer neuron and its relevant neurons. This enables us to investigate how the relevant neurons across the entire layer are distributed among upper-layer neurons. Consequently, SSA is capable of producing both dependency graphs across the entire network (global) and neuron-specific dependency graphs (local) that indicate the neurons pertinent to the activation of an upper layer neuron. The result of the global execution is a set of relevant neurons in each layer, while the result of the local execution is a set of neurons relevant to an upper-layer target

neuron. Step IV. thresholds relevant neurons based on statistical analysis to get a total of b relevant neurons.[1]

Algorithm 1: Step-wise sensitivity analysis: Identifying partially-distributed representations

INPUT: DNN classifier Φ, a layer $f^l \in \mathbb{R}^d$ from Φ, a set of relevant neurons $n \in \mathbb{S}$, and a set of images $\mathbf{I}_i \in \mathbb{I}$.

STEP I: Compute relevance of neurons using equation 1 in layer f^{l-1} for each n and $\mathbf{I}_i$ so that if f^{l-1} is a:

1. Fully-connected layer: stack results into a relevance tensor $\boldsymbol{\omega}^l_{n,:,:} \in \mathbb{R}^{|\mathbb{S}| \times K}$;
2. Convolutional layer: spatially average the output volume tensor $\boldsymbol{\omega}^l_{n,i,\dots}$ into a relevance tensor $\boldsymbol{\omega}^l_{n,i,:} \in \mathbb{R}^{|\mathbb{S}| \times |\mathbb{I}| \times K}$;
3. Pooling-layers: directly compute for $l-2$: $\boldsymbol{\omega}^l = \nabla_{f^{l-2}} f^l|_{\mathbf{I}_i}$

STEP II: Aggregate a relevance tensor $\boldsymbol{\omega}^l$ across data points to produce a relevance matrix $\boldsymbol{\omega}^l$ that indicates the relevance between the neurons in layers l and $l-1$. Aggregation can be either 1) an averaging aggregation function that yields continuous output; or 2) an outlier aggregation function (Tukey, 1977) that yields binary output.

STEP III: Aggregate a relevance matrix $\boldsymbol{\omega}^l$ across upper layer neurons to produce relevance vector ω^l. The output is either *global* – giving overall importance ranking of all neurons in a **layer** in the form of a relevance vector ω^l; or *local* – skip this step to preserve the relevance with respect to a particular neuron in the form of **neuron-specific** relevance vector $\omega^l_{n,:}$.

STEP IV: Threshold b relevant neurons. For *global output*, perform statistical thresholding of all neurons above a certain percentile such that the resulting number of neurons equals b. For *local output*, select top b neuron values $\boldsymbol{\omega}^l_{n,:}$ for each n in $\mathbb{S}$.

OUTPUT: $\mathbb{S}^{l-1}$ with b relevant neurons.

SSA is a framework, so the first step can apply any method that computes relevance scores, including gradient- (Ancona et al., 2018), statistical- (Zintgraf et al., 2017), or game-theory- (Chen et al., 2019) based approaches. Here we demonstrate how the simplest possible method for computing the relevance importance – *sensitivity analysis* (Baehrens et al., 2010; Simonyan et al., 2013) can be used. Formally, we approximate the activation of $\boldsymbol{o}_n$ with a linear function given an image $\mathbf{I}_0$, a representation function $\boldsymbol{\Phi} : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^d$ such that $\boldsymbol{\Phi}(\mathbf{I}) = \boldsymbol{o}$, and a neuron n. In the neighbourhood of $\mathbf{I}_i$, this is achieved by computing the first-order Taylor expansion: $\boldsymbol{o}_n = \boldsymbol{\Phi}_n(\mathbf{I}) \approx \boldsymbol{\omega}^T \mathbf{I} + b$ where $\boldsymbol{\omega}$ is the gradient of $\boldsymbol{\Phi}_n$ with respect to an image $\mathbf{I}$. The function is evaluated at image $\mathbf{I}_i$ – $\boldsymbol{\omega} = \frac{\partial \boldsymbol{\Phi}_n}{\partial \mathbf{I}}\Big|_{\mathbf{I}_i}$.

Hence, we can interpret the magnitude of the values of $\boldsymbol{\omega}$ as an importance metric corresponding to each pixel. In other words, these values indicate which *pixels* need to be changed the least to change $\boldsymbol{\Phi}(\mathbf{I})$ such that $\boldsymbol{o}_n$ (corresponding to a classification decision) is increased the most.

We propose a much more fine-grained analysis based on the hypothesis that sensitivity analysis can be used in an analogous way to determine the relevance between adjacent layers. Instead of trying to approximate $\boldsymbol{o}_n$ directly, we consider $\boldsymbol{\Phi}$ to be defined as the successive composition of smaller functions that represent the transformations of data between layers – $\boldsymbol{\Phi}(\mathbf{I}) = f^l(\boldsymbol{\Phi}^{l-1}(\mathbf{I})) = f^l \circ f^{l-1} \circ f^{l-2} ... \circ f^1(\mathbf{I})$, where $l = 1...L$, L is the network's depth, and each layer denoted as $f^l : \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$ represents the operation applied by layer l, when d' is the output dimensionality of the input layer f^{l-1} and d is the output dimensionality of layer l. Hence, we can evaluate the Taylor approximation at image $\mathbf{I}$ between a higher and lower layer, respectively l and j:

$$\boldsymbol{\omega}^l_{n,i,:} = \frac{\partial f^l_n\left(\boldsymbol{\Phi}^j(\mathbf{I})\right)}{\partial \mathbf{I}}\Bigg|_{\mathbf{I}_i} \tag{1}$$

4 Results and Discussion

Quantitative Evaluation We conduct a quantitative evaluation of our SSA algorithm on two datasets: circles dataset (one smaller circle inside a bigger one) and CIFAR-10. The accuracy on the original task of the resulting dependency graph is compared to that of the original network. The models are: 2 hidden-layer MLP (with 8 and 16 neurons respectively) for circles; and a convolutional network (Conv-Net such that: conv 3x3x64, max-pool, conv 3x3x64, fully-connected-328, fully-connected-194, soft-max-10 with RELU activations) that achieves **88.19%** and **70.85%** accuracy on the CIFAR-10 training and test sets respectively.

[1] Further details can be found in Appendix A.1.

We compare across 4 alternative techniques for equation 1, which we call relevance functions: *(weight_abs)* the absolute value of the weights; *(activations_abs)* the absolute average activation of a neuron over the target data; *(weight_act_abs)* the absolute average activation of a neuron over the target data multiplied by the absolute value of the weight (in the spirit of Kindermans et al. (2017)); *(gradients)* the absolute gradient values of a neuron w.r.t to the activation of an upper-layer neuron averaged across the target data. We use averaging for STEP II and for STEP IV we threshold only the neurons with values above the 50th percentile.

On the circles dataset, the activations, gradients, and weight_act_abs strategies compress the first class perfectly, while for class 2 weight_act_abs outperforms gradients with 79.88% to 71.81%[2]. This suggests that DNNs could be biased towards one class in binary classification. Table 4 illustrates that for the Conv-Net case, activations, gradients, and weight_act_abs have comparable performances. The benefit of using the gradients technique is that we can infer the sensitivity of an upper-layer neuron w.r.t. all lower-layer ones and thus identify PDRs, which is impossible with the activations strategy.

The most surprising result is the fact that the average true positive rate (TPR) of all class-specific dependency graphs (dependency graphs extracted to represent a particular class using only data corresponding to this class) outperforms the original network's accuracy substantially – 93.75%, 86.30% vs 88.19% and 70.85% (for training and test set respectively), given that these graphs are half the size of the original network! Hence, apart from their interpretability benefit, dependency graphs may be also used for three additional purposes: 1) compressing large networks into smaller ones; 2) extracting binary classifiers pertinent to a class; or 3) building more powerful ensemble models from existing architectures.

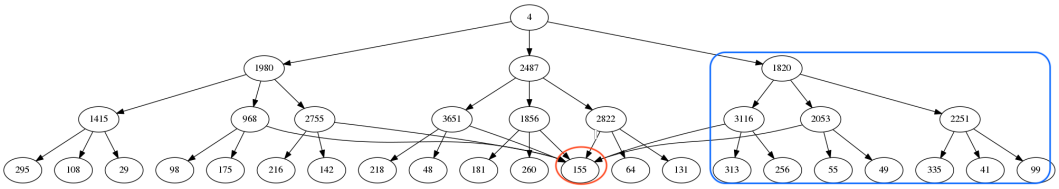
	Train	Tr Class*	Test	Ts Class*
weight_abs	51.60	51.6±23.53	46.86	46.86±23.07
weight_act_abs	51.26	91.54±13.25	45.79	83.22±19.81
gradients	45.28	93.12±8.76	41.54	85.53±17.02
activations_abs	45.79	**93.75**±8.46	42.14	**86.3**±16.67

Table 1: CIFAR-10 Dataset. The table demonstrates the mean±standard deviation accuracy of the dependency graphs over 100 different model initialisations. The columns indicate the training data-set for the relevance functions and the evaluation. The Class* columns indicate the average TPR of class-specific dependency graphs across the 10 classes, while Tr and Ts indicate training and test sets respectively. Compare to the original accuracy of **88.19%** and **70.85%** training and test respectively.

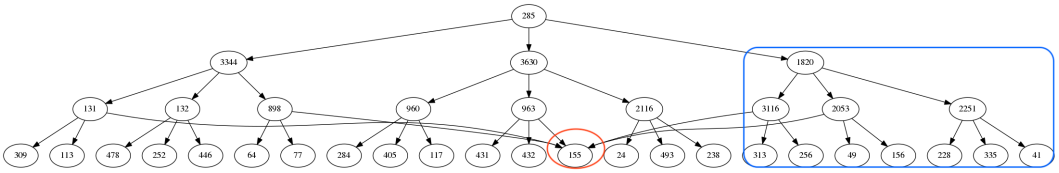
Qualitative Evaluation We conduct qualitative analysis on 10 classes and 100 images per class from ImageNet (Russakovsky et al., 2015) for the 16-layer VGG network (VGG16) (Simonyan & Zisserman, 2014). The two examples of SSA in Figure 2 illustrate that different classes may share significant similarities, sharing 6 out of the 8 most relevant activation maps and connections in `block_5_conv3` (blue rectangle). Additionally, both dependency graphs share multiple incoming connections to the same neuron f_{155}^{b5c3} (red circle). In Appendix A.2 we demonstrate that this piece of information can be used for both targetted investigation of the DNN internal operations and debugging of existing interpretability methods. For instance, we can now see that a the single heatmap visualisation interpretation is insufficient since the shared relevant neuron (f_{155}^{b5c3}) has highly activated large heatmap regions in instances of two classes.

Finally, we hypothesise that pattern matching and analysis of network motifs (Milo et al., 2002) across class-specific dependency graphs illuminates PDRs and their relationships. Once identified, PDRs can be assigned a semantic value and related to adjacent PDRs. The resulting graph would resemble a decision tree, in which the execution of a decision would be easily traceable through semantically interpretable binary decisions represented by PDRs. Therefore, we transform 10 class specific dependency graphs into bag-of-nodes feature representations to investigate patterns within the semantic properties of the dependency graphs. We perform the ward method (Murtagh & Legendre, 2011) for hierarchical agglomerative clustering with cosine distance similarity to group the dependency graphs. The results are inconclusive, with three clusters of most similar dependency graphs – 1) hammerhead and tiger shark; 2) African and Indian elephant; 3) German Sheppard and great white shark. The first two clusters consist of the most semantically and visually similar

[2]Further details can be found in Appendix A.2 Table 3



(a) Class 4: 'Hammerhead shark'



(b) Class 285: 'Egyptian cat'

Figure 2: Dependency graphs for hammerhead shark and Egyptian cat classes (penultimate 4 layers, excluding the pooling layer) expose the links only between the relevant neurons with a branching factor of 3.

classes – this supports the validity of our approach. In contrast, cluster 3) suggests an unnatural similarity between animals. One possible explanation could be that both of these classes share a similar PDR encoding, which will require further experiments to confirm.[3]

5 Conclusions & Future Work

Our main contribution to condense DNNs into smaller class-specific dependency graphs through the aggregation of instance-specific results has implications from the interpretability, development and debugging perspectives. From the interpretability perspective, we provide a methodology for the identification and guided exploration of PDRs. Our technique is capable of distilling the network into class-specific dependency graphs, which can act as binary classifiers for their corresponding classes. Not only do these classifiers indicate the execution paths in DNNs that contribute to a decision, but they can be also used as an entirely new ensemble model, which significantly increases the predictive accuracy. From the development and debugging perspective, we demonstrate an effective way to compress a DNN by decreasing the number of neurons in half, while boosting the model performance. Hence, we have found a way to quantitatively measure the quality of DNN architectures. Future work could extend the method to semantically interpretable traces through the network that justify the DNN output, which could resemble a program execution stack-trace. That could enable the detection of unnecessary layers.

Furthermore, SSA can be used to evaluate and debug existing interpretability methods. We argue that to gain a sound result in terms of interpretability and development, it is important to conduct analysis across both classes and instances. Analysis conducted merely on single instances is insufficient and misleading.

Step-wise sensitivity analysis opens an opportunity for further and more focused explorations of the internal operations of DNNs. In this paper, we demonstrate that our method is mathematically and conceptually sound. In the future, we will investigate further ways to exploit the approach in areas such as error explanation and decision justification on a much lower level by providing a semantic interpretation of the discovered PDRs through visualisation approaches. We will demonstrate the features that make the difference between semantically similar classes and quantify the interpretability of the resulting PDRs using concept segmentation as in (Fong & Vedaldi, 2018; Bau et al., 2017). Finally, we will investigate the suitability of our approach for defending against adversarial attacks.

Acknowledgments

This research was funded by EPSRC awards. We would like to thank Aaron Stockdill, Duo Wang, Zohreh Shams, Chaitanya Mangla, Edward Ayers, Daniel Raggi, Simeon Spasov, Petar Veličković, Advait Sarkar, Pietro Lio, Ian Lewis, Marek Rei, Marko Robnik-Šikonja, Adrian Weller, and many anonymous reviewers for their support and exceptionally fruitful comments and suggestions.

[3]Results are visualised in Appendix A.2 Figure 3.

REFERENCES

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pp. 9505–9515, 2018.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10 (7):e0130140, 2015.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and MÃžller, K.-R. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun): 1803–1831, 2010.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 3319–3327. IEEE, 2017.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1E3Ko09F7.

Chollet, F. et al. Keras. https://github.com/fchollet/keras, 2015.

Doshi-Velez, F. and Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv e-prints*, February 2017.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341:3, 2009.

Fong, R. and Vedaldi, A. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *arXiv preprint arXiv:1801.03454*, 2018.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

Landecker, W., Thomure, M. D., Bettencourt, L. M., Mitchell, M., Kenyon, G. T., and Brumby, S. P. Interpreting individual classifications of hierarchical networks. In *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pp. 32–38. IEEE, 2013.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. Convergent learning: Do different neural networks learn the same representations? In *Proceedings of International Conference on Learning Representation (ICLR)*, 2016.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5188–5196, 2015.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Murtagh, F. and Legendre, P. Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm. *arXiv preprint arXiv:1111.6285*, 2011.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

Ribeiro, M. T., Singh, S., and Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.

Robnik-Šikonja, M. and Kononenko, I. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600, 2008.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Tukey, J. W. *Exploratory data analysis*. Reading, Mass., 1977.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhang, J., Lin, Z., Brandt, J., Shen, X., and Sclaroff, S. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pp. 543–559. Springer, 2016.

Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M. Visualizing deep neural network decisions: Prediction difference analysis. *CoRR*, abs/1702.04595, 2017. URL `http://arxiv.org/abs/1702.04595`.

A APPENDIX

A.1 METHOD: FURTHER DETAILS

STEP I: COMPUTE RELEVANCE TENSOR

Input: This step requires a network (Φ), a layer f^l, a neuron $n \in f^l$, and an image $\mathbf{l}_i$.
Output: Computes the relevance score of neurons in layer f^{l-1} with respect to a neuron n in layer f^l as a gradient at $\mathbf{l}_i$ using equation 1. Essentially, this produces the relevance of all neurons in layer f^{l-1} to the activation of neuron n.
Method: The relevance for DCNN is computed differently depending on the type of layer f^l.

If f^l is **fully-connected**, the result is a relevance vector $\boldsymbol{\omega}^l_{n,i,:} \in \mathbb{R}^{|f^{l-1}|}$. Repeating this process for all images and neurons in $\mathbb{S}$ yields a relevance tensor $\boldsymbol{\omega}^l$.

If l is a **convolutional** layer, the result of equation 1 is a 3D relevance tensor $\boldsymbol{\omega}^l_{n,i,\dots} \in \mathbb{R}^{H \times W \times K}$, where H, W, K are respectively the height, width, and number of activation maps in $l-1$. Since every activation map k is produced by convolving identical weights onto a lower layer activation map p, k represents the existence of an identical feature across p. Hence, the vector $\boldsymbol{\omega}^l_{n,i,h,w,:}$ represents the relevance of all lower level activation maps (features) at a location (h, w) to the activation of n. Since we are interested in the relative importance of a feature, we perform spatial-averaging over all locations (h, w) to convert $\boldsymbol{\omega}_i^{f^l_n}$ into a relevance vector $\boldsymbol{\omega}_i^{f^l_n} \in \mathbb{R}^K$, where each dimension indicates the relative importance of an activation map across locations. This formulation enables us to repeat the process for all images, neurons and again obtain a 3D relevance tensor $\boldsymbol{\omega}^l$.

The **pooling** layers can be seen as a filter of their predecessors since $\frac{df^l}{d\mathbf{l}_i} = c \times \frac{df^{l-1}}{d\mathbf{l}_i}$, where $c \in \{0, 1\}$. Hence, if f^{l-1} is a pooling layer we compute the relevance tensor directly w.r.t $l-2$: $\boldsymbol{\omega}^l = \nabla_{f^{l-2}} f^l|_{\mathbf{l}_i}$.

Notice that to analyse the evaluation results in Section 4 we can change equation 1 and experiment with different values. For example, instead of gradients in equation 1, it can yield the connecting weights, the actual activations ($\boldsymbol{\Phi}^j(\mathbf{l})$) or the element-wise product between the weights and the activations. In the case of the weights, the next step is redundant since they do not vary with the input samples.

STEP II: AGGREGATE ACROSS DATAPOINTS

Input: This steps requires a relevance tensor $\boldsymbol{\omega}^l$.
Output: The result is a relevance matrix $\boldsymbol{\omega}^l$ that indicates the relevance between the neurons in layers l and $l-1$.
Method: This step aggregates across the dataset dimension of the relevance tensor $\boldsymbol{\omega}^l$. We explore two possible aggregation functions (averaging and outliers). Note that the aggregation functions operate across all data points and as such they yield relative, not absolute results. That is, changing the weights in the model results in an absolute change in all $omega$ values, but relatively there will be no difference.

The averaging technique takes the mean over the datapoints dimension to produce a relevance matrix $\boldsymbol{\omega}^l$, which indicates the average relevance across datapoints between neurons in layers l and $l-1$. On the other hand, preliminary experiments indicated that each row $\boldsymbol{\omega}^l_{n,i,:}$ follows a normal distribution, and consistently exhibits a small number of outliers across i. Therefore, we make the simplifying assumption that these outliers are the only relevant neurons. Finally, we use the Tukey's fences ($1.5\times$ Inter-Quartile Range) outlier detection method (Tukey, 1977) to select relevant neurons from each row $\boldsymbol{\omega}^l_{n,i,:}$.

Observe that in the case of the averaging technique the relevance matrix $\boldsymbol{\omega}^l$ contains continuous values, while in the outliers case it contains binary values.

STEP III: AGGREGATING ACROSS UPPER LAYER NEURONS

Input: Relevance matrix $\boldsymbol{\omega}^l$.
Output: Relevance vector ω^l.
Method: Here we aggregate across the dimension of upper layer neurons to produce a global layer ranking in the form of a relevance vector ω^l. For the current iteration we use averaging; however, in future work, we will explore different alternatives. Notice that it is possible to preserve the local relevance of the neurons in this step, which enables us to produce a neuron-specific relevance vector $\omega^l_{n,:}$. These vectors can be used to explore PDRs as we demonstrate in Section 4.

STEP IV: THRESHOLD

Input: Relevance vector ω^l.
Output: Set $\mathbb{S}^{l-1}$ of all relevant neurons for the lower layer.
Method: For the global relevance case, we perform statistical thresholding of all neurons above a certain percentile such that the resulting number of neurons equals b. This results in a set $\mathbb{S}^{l-1}$ of all relevant neurons for the lower layer. In future work, we will explore alternative ways to perform the thresholding.

For the local relevance case, we perform the aforementioned statistical thresholding across $\boldsymbol{\omega}^l_{n,:}$ for each n to get a relationship $\mathbb{R}$. $\mathbb{R}$ maps the set $\mathbb{S}'$ of b' relevant neurons for each distinct n in $\mathbb{S}$ to the frequency with which they the neuron was deamed relavant. We rank all neurons in $\mathbb{S}'$ based on their frequency values to select top b neurons $\boldsymbol{\omega}^l_{n,i}$ as relevant, where $b = |\mathbb{B}|$ is the branching factor.

For both cases, we set $\mathbb{S}^{l-1}$ to the union of all relevant neurons $\mathbb{B}^n$ ($\mathbb{S} \leftarrow \bigcup_n \mathbb{B}^n$) for the lower layer. For computational efficiency, the magnitude of b is a threshold for the cardinality of each $\mathbb{B}^n$, thus discarding a proportion of potential relevant neurons. We believe that b is an important hyper-parameter since it limits the size of potential PDRs, which recent studies indicate to be typically between 8 and 50 neurons (Fong & Vedaldi, 2018).

The time complexity of our approach in the worst-case is $O(b*d*n)$, where b is the time to perform the backward pass, d is the depth, and n is the maximum number of neurons ($d = 22$, $n = 1024$ in the case of VGG16). The approach is still practical since it is not designed to be executed every time that an explanation is necessary, just as a network is not retrained every time before a prediction.

In summary, our SSA algorithm is capable of producing both, dependency graphs across the entire network (global) and neuron-specific dependency graphs (local) that indicate the neurons pertinent to the activation of an upper layer neuron. The result of the global execution is a set of relevant neurons in each layer, while the result of the local execution is a set of neurons relevant to an upper-layer target neuron.

A.2 ADDITIONAL RESULTS

For the qualitative analysis, We use the publicly available pre-trained model implemented in the deep learning framework keras (Chollet et al., 2015) and we use the outliers aggregation function for STEP II and we threshold the top three omega values for each upper layer neuron.

Toy Problem These experiments evaluate the mathematical soundness of using gradients as a measure of relevance with a toy non-linear binary classification problem of two circles – one smaller circle inside a bigger one. A Multi-Layer Perceptron (MLP) with a single hidden layer of 3 neurons can solve the problem, so we intentionally train a more complex 2 hidden-layer MLP (with 16 and 2 neurons respectively). The overall importance of a neuron can be approximated through ablation experiments, in which the target neuron is disconnected from the network. The gold standard ranks each neuron according to an importance score proportional to the performance degradation resulting from its removal.

The Spearman's ρ rank correlation coefficient between the gold standard and the predicted relevance importance ranking of four alternative techniques to equation 1 is respectively -0.032, 0.309 , **0.371**, 0.647 for: *(weight strategy)* the value of the weights; *(absolute weight)* the absolute value of the weights; *(activations)* the absolute average activation of a neuron over the target data; *(gradients)* the absolute gradient values of a neuron w.r.t to the activation of an upper-layer neuron averaged

TECHNIQUE	SPEARMAN'S ρ
WEIGHTS	-0.032
ACTIVATIONS	0.309
GRADIENTS	**0.371**
ABSOLUTE WEIGHTS	0.647

Table 2: Spearman's ρ rank correlation coefficient between the predicted relevance importance ranking and the gold standard - $+1$ perfect monotone relationship, -1 opposed monotone relationship. Notice that the gradients strategy outperforms the activations, while coming short to the absolute weights technique.

	TRAIN	TEST	CLASS 1	CLASS 2
WEIGHT_ACT	81.41±13.35	80.88±13.29	95.05±13.56	69.38±26.28
WEIGHT	82.72±12.58	82.23±13.0	96.17±12.17	69.37±26.41
WEIGHT_ABS	87.12±10.08	86.47±10.59	97.21±5.63	76.57±22.2
ACTIVATIONS	71.36±15.58	72.27±15.91	100.0±0.0	71.23±27.84
ACTIVATIONS_ABS	71.36±15.58	72.27±15.91	100.0±0.0	71.23±27.84
GRADS	74.14±13.63	73.82±13.94	100.0±0.0	71.81±26.0
GRADS_ABS	74.14±13.63	73.82±13.94	100.0±0.0	71.81±26.0
WEIGHT_ACT_ABS	75.36±10.39	75.92±10.91	100.0±0.0	79.88±16.82

Table 3: Toy Example Dataset. The table demonstrates the mean±standard deviation performance of the dependency graphs over 100 different model initialisations. The columns indicate from which data-set the target data for the compute functions was acquired and for which data-set was the evaluation performed. The class columns indicate which of the two classes is considered.

	TRAIN	TR CLASS*	TEST	TS CLASS*
WEIGHT_ACT	16.83	0.0±0.0	16.05	0.0±0.0
WEIGHT	16.80	16.8±26.79	16.29	16.29±27.21
WEIGHT_ABS	51.60	51.6±23.53	46.86	46.86±23.07
WEIGHT_ACT_ABS	51.26	91.54±13.25	45.79	83.22±19.81
GRADS	45.28	93.12±8.76	41.54	85.53±17.02
GRADS_ABS	45.28	93.12±8.76	41.54	85.53±17.02
ACTIVATIONS	45.79	93.75±8.46	42.14	86.3±16.67
ACTIVATIONS_ABS	45.79	93.75±8.46	42.14	86.3±16.67

Table 4: CIFAR-10 Dataset. The table demonstrates the mean±standard deviation performance of the dependency graphs over 100 different model initialisations. The columns indicate from which data-set the target data for the compute functions was acquired and for which data-set was the evaluations performed. The Class* columns indicate that the results are averaged across the 10 classes, while Tr and Ts indicate training and test set respectively.

across the target data. Although the absolute weights strategy exhibits the highest correlation with the gold standard, it is limited to global results. That is, the relevance ranking corresponds solely to the overall performance of the network in contrast to the local performance for a specific class, which can be produced by the gradients.

Application of Step-wise Sensitivity Analysis As demonstarted in Section 4, Step-wise sensitivity analysis allows developers to focus analysis and interpretation efforts on the most pertinent regions of a DNN. For example, Figure 2 illustrates the importance of neuron f_{155}^{b5c3} (red circle). Figures 4(a) & 4(b) display targeted visualisation through guided backpropagation Springenberg et al. (2014) of the especially relevant neuron f_{155}^{b5c3}. Had we relied on a single visualisation, we would have erroneously presumed that the neuron perfectly encodes either the idea of a shark or of a cat. However, step-wise sensitivity analysis exposes that the neuron is equally important for both classes, and forms a part of a shared sub-structure. Therefore, it must encode a more abstract concept. These results confirm previous results in line with the fragility of sensitivity analysis-based visualisaiton (Adebayo et al., 2018).

Exploring a neuron or activation map in isolation is simplistic. In reality, the semantics are expressed within the combination of neurons within the PDR. In future work, we will apply activation maximisation Erhan et al. (2009) to an entire PDR to investigate its semantic properties.

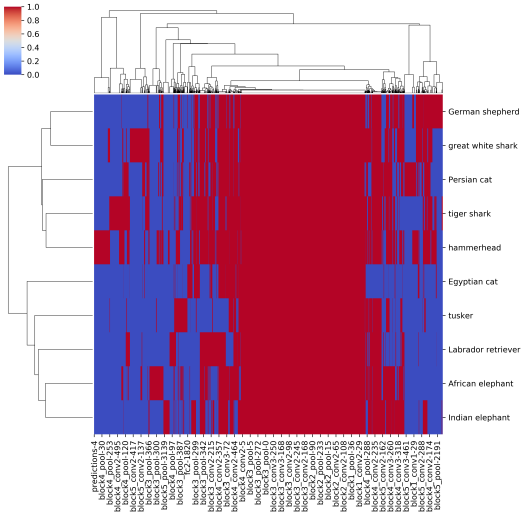
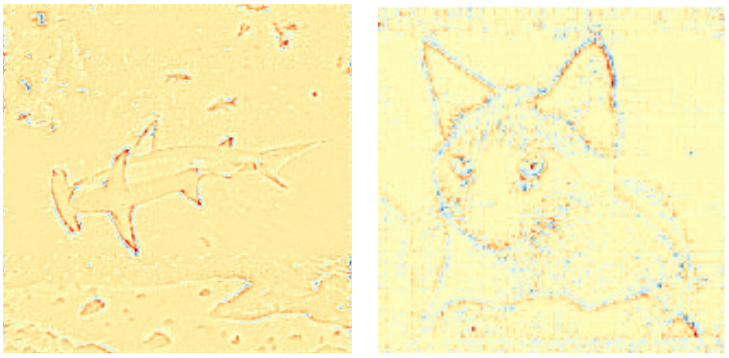


Figure 3: A clustered heatmap (clustermap) of each of the 10 dependency graphs (spanning the entire network) into a bag-of-nodes features representation. The x,y, z-axis respectively represent neuron, class, and presence of the neuron in the dependency graph – red present, blue absent. The dendrograms on the side indicate the relative distance between points and clusters. As expected, most of the lower layer activation maps are shared across all classes since they encode very abstract features. Notice the three small clusters of semantically similar classes on the side.



(a) Class 4: 'Hammerhead shark' (b) Class 285: 'Egyptian cat'

Figure 4: a) & b) Guided-backpropagation of activation map f_{155}^{b5c3} indicating the regions of the image from the corresponding class. Red and blue respectively correspond to positive or negative contribution to the activation.