What's in the Box?
Exploring the Inner Life of Neural Networks with Robust Rules

Jonas Fischer [1] **Anna Oláh** [1] **Jilles Vreeken** [2]

Abstract

We propose a novel method for exploring how neurons within neural networks interact. In particular, we consider activation values of a network for given data, and propose to mine noise-robust rules of the form $X \to Y$, where X and Y are sets of neurons in different layers. We identify the best set of rules by the Minimum Description Length principle, as those rules that together are most descriptive of the activation data.To learn good rule sets in practice, we propose the unsupervised EXPLAINN algorithm. Extensive evaluation shows that the patterns it discovers give clear insight into how networks perceive the world: they identify shared and class-specific traits, compositionality, as well as locality in convolutional layers. Moreover, they are not only easily interpretable, but also super-charge prototyping by identifying which neurons to consider in unison.

1. Introduction

Neural networks achieve state of the art performance in many settings. However, how they perform their tasks, how they perceive the world, and especially, how the neurons within the network operate in concert, remains largely elusive. While there exists a plethora of methods for explaining neural networks, most of these focus either on the mapping between input and output (e.g. model distillation) or only characterize a given set of neurons, but can not identify which set to look at in the first place (e.g. prototyping). In this paper, we introduce a new approach to explain how the neurons in a neural network interact. In particular, we consider the activations of neurons in the network over a given dataset, and propose to characterize these in terms of rules $X \to Y$, where X and Y are sets of neurons in different layers of the network. A rule hence represents that neurons Y are typically active when neurons X are. For robustness we explicitly allow for noise, and to ensure that we discover a succinct yet descriptive set of rules that captures the regularities in the data, we formalize the problem in terms of the Minimum Description Length principle (Grünwald & Roos, 2019). To discover good rule sets, we propose the unsupervised EXPLAINN algorithm and show that the rules we discover give clear insight in how networks performs their tasks. As we will see, these identify what the network deems similar and different between classes, how information flows within the network, and which convolutional filters it expects to be active where. Our rules are easily interpretable, give insight in the differences between datasets, show the effects of fine-tuning, as well as supercharge prototyping as they tell which neurons to consider in unison.

Explaining neural networks is of widespread interest, and especially important with the emergence of applications in healthcare and autonomous driving. In the interest of space we here only shortly introduce the work most relevant to ours, while we refer to surveys for more information (Adadi & Berrada, 2018; Ras et al., 2018; Xie et al., 2020; Gilpin et al., 2018). There exist several proposals for investigating how networks arrive at a decision for a given sample, with saliency mapping techniques for CNNs among the most prominent (Bach et al., 2015; Zhou et al., 2016; Sundararajan et al., 2017; Shrikumar et al., 2017). Although these provide insight on what parts of the image are used, they are inherently limited to single samples, and do not reveal structure across multiple samples or classes. For explaining the inner working of a CNN, research mostly focuses on feature visualization techniques (Olah et al., 2017) that produce visual representations of the information captured by neurons (Mordvintsev et al., 2015; Gatys et al., 2015). Although these visualizations provide insight on how CNNs perceive the world (Øygard, 2016; Olah et al., 2018) it has been shown that concepts are often encoded over multiple neurons, and that inspecting individual neurons does not provide meaningful information about their role (Szegedy et al., 2013; Bau et al., 2017). How to find such groups of neurons, and how the information is routed between layers in the networks, however, remains unsolved.

[1]Max Planck Institute for Informatics, Germany [2]CISPA Helmholtz Center for Information Security, Germany. Correspondence to: Jonas Fischer <fischer@mpi-inf.mpg.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

An orthogonal approach is model distillation, where we train easy-to-interpret white box models to mimic the decisions of a neural network (Ribeiro et al., 2016; Frosst & Hinton, 2017; Bastani et al., 2017; Tan et al., 2018). Rules of the form *(if–then)* are easily interpretable, and hence a popular technique for model distillation (Taha & Ghosh, 1999; Lakkaraju et al., 2017). Existing techniques (Robnik-Šikonja & Kononenko, 2008; Özbakır et al., 2010; Barakat & Diederich, 2005) aim for rules that directly map input to output, rather than providing insight into how information flows through the network. Tran & d'Avila Garcez (2018) restrict themselves to Deep Belief Networks only, and for these propose to mine all sufficiently strong association rules. As such, their method suffers from the well-known pattern explosion. In contrast, Chu et al. (2018) propose to explain NNs by deriving decision boundaries of a network using polytope theory. While this approach permits strong guarantees, it is limited to very small (< 20 hidden neurons) piecewise linear NNs. In sum, existing methods either do not give insight in what happens inside a neural network, and/or, are not applicable to the type or size of state-of-the-art convolutional neural networks. Zhang et al. (2018) show how we can gain insight into convolutional layers of neural networks by building an explanatory graph over sets of neurons. In contrast to what we propose, their method does not elucidate the relation between such filters and subsequent dense layers, nor to the network output.

Instead, we propose to mine sets of rules to discover groups of neurons that act together across different layers in feed forward networks, and so reveal how information is composed and routed through the network to arrive at the output. To discover rules over neuron activations, we need an unsupervised approach. While many rule mining methods exists, either based on frequency (Agrawal & Srikant, 1994; Bayardo, 1998; Moerchen et al., 2011) or statistical testing (Hämäläinen, 2012; Webb, 2010), these typically return millions of rules even for small datasets, thus thwarting the goal of interpretability. We therefore take a pattern set mining approach similar to GRAB (Fischer & Vreeken, 2019), where we are after that set of rules that maximizes a global criterion, rather than treating each rule independently. Although providing succinct and accurate sets of rules, GRAB is limited to conjunctive expressions. This is too restrictive for our setting, as we are also after rules that explain shared patterns between classes, and are robust to the inherently noisy activation data, which both require a more expressive pattern language of conjunctions, approximate conjunctions, and disjunctions. We hence present EXPLAINN, a non-parametric and unsupervised method that learns sets of such rules efficiently.

2. Theory

We first informally discuss how to discover association rules between neurons. We then formally introduce the concept of robust rules, and how to find them for arbitrary binary datasets, last, we show how to combine these ideas to reveal how neurons are orchestrated within feedforward networks.

2.1. Patterns of neuron co-activation

Similar to neurons in the brain, when they are active, artificial neurons send information along their outgoing edges. To understand flow of information through the network, it is hence essential to understand the activation patterns of neurons between layers. Our key idea is to use recent advances in pattern mining to discover a succinct and non-redundant set of rules that together describe the activation patterns found for a given dataset. For two layers I_i, I_j, these rules $X \rightarrow Y, X \subset I_i, Y \subset I_j$ express that the set of neurons Y are usually co-activated when neurons X are co-activated. That is, such a rule provides us *local* information about co-activations within, as well as the dependence of neurons between layers. Starting from the output layer, we discover rules between consecutive layers I_j, I_{j-1}. Discovering overlapping rules between layers $X \rightarrow Y$ and $Y \rightarrow Z, X \subset I_j, Y \subset I_{j-1}, Z \subset I_{j-2}$, allows us to trace how information flows through the entire network.

Before we can mine rules between two sets of neurons – e.g. layers – I_i and I_j of a network, we have to obtain its binarized activations for a given data set $\mathcal{D} = \{d_k = (s_k, o_k)\}$. In particular, for each sample s_k and neuron set I_i, we take the tensor of activations ϕ_i and binarize it to ϕ_i^b. For networks with *ReLU* activations, which binarize naturally at threshold 0, we might lose some information about activation strength that is eventually used by subsequent layers. This binarization however allows us to derive crisp symbolic, and directly interpretable statements on how neurons interact. Furthermore, binarization reflects the natural on/off state of biological neurons, also captured by smooth step functions such as sigmoid or tanh used in artifical neural networks. We gather the binarized activations into a dataset D where each row t_k corresponds to the concatenation of ϕ_i^b and ϕ_j^b of I_i and I_j for s_k, i.e., $t_k \in D$ is a binary vector of length $|I_i| + |I_j|$. See Fig. 1 for a toy example.

Next, given binary activation data D, our goal is to find that set of rules that together succinctly describe the observed activations. The Minimum Description Length (MDL) principle lends itself as an objective to find such sets. MDL is a statistically well-founded and computable approximation of Kolmogorov Complexity (Li & Vitányi, 1993). First introduced by Rissanen (1978), the essential idea is that the model $M^* \in \mathcal{M}$ that best describes data D is the model that losslessly encodes D using the fewest bits $M^* = \arg\min_{M \in \mathcal{M}} L(D, M)$. Here, our model class $\mathcal{M}$

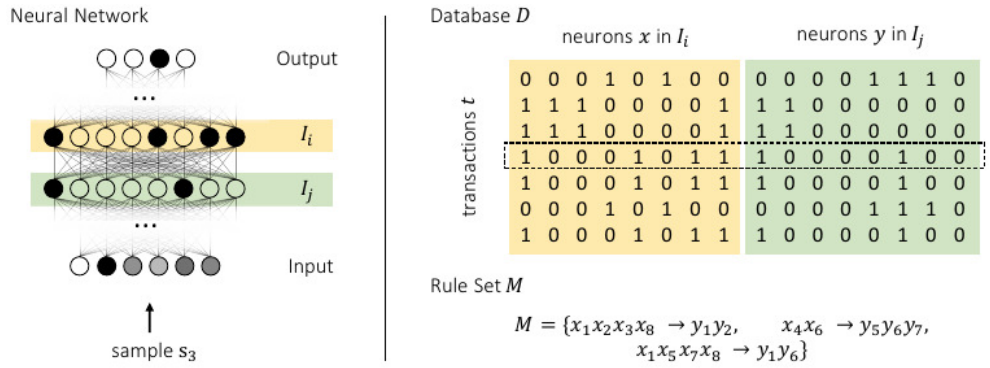


Figure 1: *Overview.* For a given network (left), binarized activations are gathered for the layers I_i, I_j for each sample, and summarized in the binary database D (right). Rules are discovered over D, where a good rule set M is given with at the bottom right, with rules $X \rightarrow Y$, $X \in I_i$, $Y \in I_j$.

is the superset of all possible rules over D, and by MDL we identify the best model M^* as the one that compresses the data best. Traditionally, rule mining is restricted to conjunctions over items, which is not sufficient for our application; neuron activations tend to be noisy, labels are inherently mutually exclusive, and hence we consider an extended language that allows for partial disjunctions of items (neurons, labels) and introduce a codelength function $L(D, M)$ to instantiate MDL for our model class of rule sets.

2.2. MDL for Robust Rules

Our goal is to find a set of rules M that, in terms of description length $L(D, M)$, best describes a binary database $D = \{t \mid t \subset \mathcal{I}\}$ that consists of transactions t that are subsets of items $\mathcal{I}$. Each rule is of the form $X \rightarrow Y,\ X, Y \subset \mathcal{I}$, and indicates that Y is strongly associated with, i.e. occurs mostly in transactions where X is present. We say a rule $X \rightarrow Y$ *applies* to a transaction t iff $X \subset t$ and say a rule *holds* for t if additionally $Y \subset t$. We indicate these transactions sets as $T_X = \{i \mid t_i \in D,\ X \subset t_i\}$, respectively $T_{Y|X} = \{i \mid t_i \in T_X,\ Y \subset t_i\}$. Based on these definitions of rule transaction sets, we can now formally introduce our codelength function $L(D, M)$.

Baseline model Our base model $M_{ind} = \{\emptyset \rightarrow I \mid \forall I \in \mathcal{I}\}$ consists of singleton rules only, i.e. it models that all items $\mathcal{I}$ are generated independently. To send the n transactions of D using M_{ind}, we simply send for each item I in which out of all transactions in the database it appears. We can do so optimally using a log binomial code, which is given by $\log \binom{|T_\emptyset|}{|T_{I|\emptyset}|} = \log \binom{n}{|T_I|}$. To unambiguously decode, the recipient needs to know each $|T_I|$, which we can optimally encode via the parametric complexities of the binomials, which are defined as $L_{pc}(n) = \log \left(\sum_{k=0}^{n} \frac{n!}{(n-k)!k!} \left(\frac{k}{n}\right)^k \left(\frac{n-k}{n}\right)^{n-k} \right)$, and can be computed in linear time (Kontkanen & Myllymäki, 2007). We thus have $L(D, M_{ind}) = \sum_{I \in \mathcal{I}} \left(\log \binom{n}{|T_I|} + L_{pc}(n) \right)$. M_{ind} serves as our baseline model, and its singleton rules are a required part of any more complex model as they ensure we can always send any data over $\mathcal{I}$.

Non-trivial models A non-trivial model M contains rules of the form $X \rightarrow Y,\ \ X, Y \subset \mathcal{I}$ that are not part of M_{ind}. The idea is that we first transmit the data for where these non-trivial rules hold, and then send the remaining data using M_{ind}. To determine where such a rule applies, the receiver needs to know where X holds, and hence the data over X needs to be transmitted first. To ensure that we can decode the data, we only consider models M for which the directed graph $G = (\mathcal{I}, E)$ is acyclic, in which there exists an edge between two items i_1, i_2 iff they occur in the head and tail of a rule, that is $E = \{(i_1, i_2) \mid \exists X \rightarrow Y \in M.\ i_1 \in X \wedge i_2 \in Y\}$. We thus get a codelength

$$L(D \mid M \cup M_{ind}) = \left(\sum_{X \rightarrow Y \in M} \log \binom{|T_X|}{|T_{Y|X}|} \right) + \sum_{\emptyset \rightarrow I \in M_{ind}} \log \binom{n}{|T'_I|},$$

where $T'_I = \{t \in D \mid (I \in t) \wedge (\forall X \rightarrow Y \in M.\ I \in Y \implies t \notin T_{Y|X})\}$ is a modified transaction set containing transactions with item I not covered by any non-trivial rule.

In addition to the parametric complexities of the binomial codes, the model cost of a non-trivial model also includes the cost of transmitting the non-trivial rules. To transmit a rule $X \rightarrow Y$, we first send the cardinalities of X resp. Y using the universal code for integers $L_{\mathbb{N}}$ (Rissanen, 1983). For $n \geq 1$, this is defined as $L_{\mathbb{N}}(z) = \log^* z + \log c_0$ with $\log^*(z) = \log z + \log\log z + \ldots$, summing only over the positive components (Rissanen, 1983). To satisfy the Kraft inequality up to equality we set $c_0 = 2.865064$. Knowing

the cardinalities, we can then send the items of X resp. Y one by one using an optimal prefix code given by $L(X) = -\sum_{x \in X} \log \frac{|T_x|}{\sum_{I \in \mathcal{I}} |T_I|}$. For a particular rule $X \to Y \in M$, the model costs for a rule, respectively the full model thus amount to

$$\begin{aligned} L(X \to Y) =& L_{\mathbb{N}}(|X|) + L_{\mathbb{N}}(|Y|) \\ &+ L(X) + L(Y) + L_{pc}(|T_X|) \ , \\ L(M \cup M_{ind}) =& |\mathcal{I}| \times L_{pc}(n) + L_{\mathbb{N}}(|M|) \\ &+ \sum_{X \to Y \in M} L(X \to Y) \ . \end{aligned}$$

We provide an example calculation in Supp. A.1. With these definitions, we have an MDL score that identifies the best rule set M^* for data D as

$$M^* = \arg\min_{M \in \mathcal{M}} \big(L(M \cup M_{ind}) + L(D \mid M \cup M_{ind}) \big) \ ,$$

where $\mathcal{M}$ contains all possible rule sets over the items in D.

Robust Rules In real world applications, we need a score that is robust against noise.The key problem with noisy data is that a single missing item in a transaction can cause a whole rule not to hold or apply. To discover rules that generalize well, we need to explicitly account for noise. The idea is to let rules apply, and hold, also when some items of head respectively tail are missing. Specifying how many items l, and k, out of all items in the rule head, respectively tail, need to be part of a transaction, we relax the original rule definition to account for missing items, or in other words, noise.

Furthermore, as output neurons – the classes – are only active mutually exclusively, rules need to be able to model disjunctions. Setting $l = 1$ and $k = 1$ means that only one of the items of head respectively tail need to be present, thus coincidentally corresponding to a disjunction of items in the head and tail of the rule $X \to Y$, thus allowing to model output neurons correctly, and $l = |X|$ and $k = |Y|$ correspond to the original stringent rule definition. Varying between the two extremes accounts for varying levels of noise. The optimal l and k are those that minimize the MDL score.

To ensure a lossless encoding, we need to make sure that the receiver can reconstruct the original data. Thus, for the previously introduced relaxed definition of when rules hold and apply, we send for each rule the corresponding number of items l that need to be present for it to apply using $L_{\mathbb{N}}(l)$ bits. Knowing each l, the receiver can reconstruct where each rule applies. Sending where a rule holds now leaves the receiver with an approximation of the data. To be able to reconstruct the actual data, Fischer & Vreeken (2019) introduced error matrices that when XORed with the approximation yield the original data. These two matrices $\mathcal{X}^+_{X \to Y}$, and $\mathcal{X}^-_{X \to Y}$ correct for the errors made in the part where the rule applies and holds, respectively applies but does not hold. These error matrices are part of the model M and have to be transmitted with an adapted $L(D, M)$. We provide examples and a short review how how to adapt the codelength function in Supp. A.

Complexity of the search To discover rules over the activations of layers I_i, I_j, we have to explore all rules formed by subsets of neurons in I_i for the head, combined with any subset of neurons of I_j for the tail. There exist $2^{|I_i|} \times 2^{|I_j|}$ such rules, and hence $2^{2^{|I_i|}+|I_j|}$ distinct models would need to be explored. Fischer & Vreeken (2019) showed that the rule set search space does not lend itself to efficient search as it is neither monotone nor submodular, the counterexamples also holding for our model definition. In fact, for robust rules, we additionally have to consider where rules should apply respectively hold – optimizing k and l – which results in approximately $2^{|I_i| \times |I_j| \times 2^{|I_i|+|I_j|}}$ models (details in Supp. A.4). Exhaustive search is therewith infeasible, which is why we present EXPLAINN, a heuristic algorithm to efficiently discover good sets of rules.

2.3. Discovering good rule sets with EXPLAINN

EXPLAINN is based on the idea of iteratively refining the current model by merging and refining already selected rules. The key insight of the algorithm is that for a rule $X \to Y$ to summarize the data well, also rules $X \to Y'$ with only part of the tail, $Y' \subset Y$, should summarize well, as all tail items should be similarly co-occuring with head X. Starting from the baseline model M_{ind} we iteratively and greedily search for better models until we can no longer improve the MDL score. As search steps we consider either introducing a new rule to M, by taking a good set of items $X \subset I_i$ for the head and a single item $A \in I_j$ for the tail and refine the model to $M' = M \oplus \{X \to A\}$, seeing if it decreases the overall MDL costs (Eq. 2.2). Or, we merge two existing rules $r_1 = X \to Y_1 \in M$ and $r_2 = X \to Y_2 \in M$, to form a new rule $r' = X \to Y_1 \cup Y_2$ and refine the model to $M' = M \oplus \{r'\} = (M \setminus \{r_1, r_2\}) \cup \{r'\}$. For a rule r', the refinement operator $\oplus$ is adding the rule $r' = X \to Y$ to M, and removes the merged rules that led to r', if any. Moreover, it updates the singleton transaction lists T_A for all items $A \in Y$, removing all transactions where r' holds.

To permit scaling up to the size of a typical neural net, we next discuss how to efficiently search for candidate rules with heads that can express anything from conjunctions to disjunctions. Immediately after, we present the full algorithm EXPLAINN for mining high quality rule sets for two arbitrary sets of neurons (e.g. layers) of a network.

Searching for candidates A key component of EXPLAINN is the candidate generation process, which implements the two possible steps of generating new and merging existing rules. Given two layers I_i, I_j, to efficiently discover rules that are both robust to noise, and may include disjunctively active neurons in the head, we can not enumerate all possible rule heads for each individual tail neuron, as this would result in $|I_j| \times 2^{|I_i|}$ many rules. Instead, we keep a list H_y for each item $y \in I_j$, storing all head neurons $x \in I_i$ for which y is frequently active when x is active, that is $\sigma_{x,y} = \frac{|T_x \cap T_y|}{|T_x|} > \theta$, where θ is a confidence threshold. We consider a rule $X \to Y$ to be good, if when neurons X are active, the neurons Y are also likely to be active, which is directly represented by the confidence θ. With parameter μ we account for early decisions on rule merging that later hinder us to see a more general trend. The lists are sorted decreasing on σ. We search in each H_y for the rule with highest gain over all unions of first $t = 1 \dots |H_y|$ neurons in the list. We add that rule $X \to y$ with highest gain to the candidate list. To compute the gain, we consider all possible values $k = 1 \dots |X|$ to determine for which transactions $T_X^k = \{t \in D \mid |X \cap t| \geq k\}$ the rule should robustly apply, where $k = 1$ corresponds to disjunction and $k = |X|$ to conjunction of neurons.

For an individual neuron y, such a rule would be optimal, but, our goal is to discover groups of neurons that act in concert. To this end we hence iteratively merge rules with *similar* heads – similar, rather than same, as this gives robustness both against noise in the data, as well as earlier merging decisions of the algorithm. For two rules $X_1 \to Y_1,\ X_2 \to Y_2$ with symmetric difference $X_1 \ominus X_2 = (X_1 \setminus X_2) \cup (X_2 \setminus X_1)$, we consider possible candidate rules $X_1 \cup X_2 \to Y_1 \cup Y_2$ and $X_1 \cap X_2 \to Y_1 \cup Y_2$, iff $|X_1 \ominus X_2| \leq \mu$ for some threshold $\mu \in \mathbb{N}$. For example, $\mu = 1$ corresponds to the case that one head has one label more than the other, all other labels are the same.

Both parameters θ and μ are simple, yet effective runtime optimizations. The best results with respect to MDL will always be obtained with the largest search space, i.e. with θ and μ set to 0, respectively $|X_1| + |X_2|$. Besides impacting run-time, many of those rules may be uninteresting from a user-perspective, μ and θ allow to directly instruct EXPLAINN to ignore such rules.

EXPLAINN Assembling the above, we have EXPLAINN, which given two sets of neurons I_i, I_j and a database of activations of these neurons, yields a heuristic approximation to the MDL optimal model M^*. By first introducing all relevant single neuron rules, it then proceeds by iteratively merging existing rules using the approach described above, until it can achieve no more gain. For efficiency, we separate the generation of the new rules from the merging of existing rules. In practice, this does not harm performance, as we allow merging of similar heads and can thus revert too greedy decisions introduced earlier. Furthermore, by observing that independent rules $X_1 \to Y_1,\ X_2 \to Y_2,\ Y_1 \cup Y_2 = \emptyset$ do not influence each others impact on codelength, we can add all independent rules with the highest respective gain at once. We provide pseudocode for candidate generation and the EXPLAINN algorithm in Supp. A.5.

Complexity of EXPLAINN The generation of new rules results in time $O(n \times |I_j| \times |I_i|^3)$, by iterating over each neuron in I_j, and considering each subset of the most overlapping neurons in I_i, and considering each threshold $k = 1 \dots |I_i|$ for when the rule should apply, and the factor n from intersecting transaction lists T to compute the gain. We can have at most $|I_j|$ generated rules before considering rule merges, and in every iteration of merging we combine two rules, reducing the rule set size by 1. In each such step, we consider $|I_j|^2$ merges, for each of which we compute the gain considering noisy head and tail. We thus have a worst case runtime of $O(n \times |I_j|^4 \times |I_i|)$. As MDL ensures we consider models that tend to be succinct and hence capture only relevant structure in the data, EXPLAINN is in practice much faster and easily scales to several thousands of neurons.

3. Experiments

In this section we empirically evaluate EXPLAINN on synthetic data with known ground truth and real world data to explore how CNNs perceive the world. Other approaches to discover patterns based on e.g. frequency measures or statistical testing have already been shown to yield millions or billions of rules or patterns, most spurious and redundant, and many more than anyone would be willing to investigate, see e.g. (Fischer & Vreeken, 2019), we hence focus on evaluating our method for the task of finding activation patterns. Here, we look at CNNs as they count towards the most widespread use of feedforward networks and naturally lend themselves for visualization, which helps us to interpret the discovered rules. We compare to traditional prototyping and activation map approaches on *MNIST* (LeCun & Cortes, 2010), and examine which information is used how to arrive at classification for *ImageNet* (Russakovsky et al., 2015). Finally, we investigate the effect of fine-tuning in transfer learning on the Oxford *Flower* data (Nilsback & Zisserman, 2008). The implementation of EXPLAINN is publicly available.[1] For the below experiments, running on commodity hardware EXPLAINN took minutes for *MNIST* and *Flower*, and up to 6 hours for *ImageNet*— yielding from a few hundred up to 3000 rules, for the smaller, respectively larger networks, and earlier, respectively later layers.

[1] http://eda.mmci.uni-saarland.de/explainn/

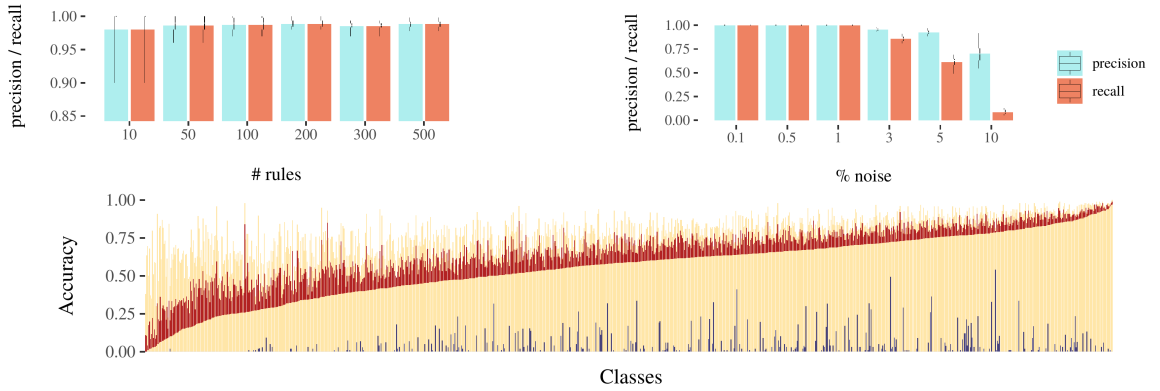


Figure 2: *Evaluation of rule quality.* **Top:** Performance of EXPLAINN as precision and recall on data with varying number of planted rules with mutual exclusive head items (left) and co-occurring head items with varying noise (right). 10% noise corresponds to more noise than signal in the data. We provide the average (bar) and distribution (boxplot) across 10 repetitions. **Bottom:** Accuracy per class of VGG-S before (yellow) and after (blue) intervention on weights connecting neurons to class given by a rule, and 90% quantile of accuracies obtained for randomized intervention (red).

3.1. Recovering ground truth

To evaluate how well EXPLAINN can recover the ground truth from data, we first generate synthetic binary data sets of 10000 samples and introduce $\{10, 50, 100, 200, 300, 500\}$ rules with up to 5 items in head and tail, respectively. For each rule, the frequency is drawn from $U(.02, .08)$, the confidence is drawn from $U(.5, 1)$. We introduce noise by flipping 0.1% of the entries chosen uniformly at random, and add 5 noise features with frequency equal to those of rules. Fischer & Vreeken (2019) showed that a similar MDL model works for conjunctive rules, hence we will focus on investigating performance for mutually exclusive rule heads and noise. In the first set of experiments, we set head items mutual exclusively, in line of finding rules over the NN output labels. EXPLAINN achieves high recall and precision (see Figure 2) in terms of retrieving exact ground truth rules, and does not retrieve any redundant rules. Next, we investigate the impact of noise on the performance, generating data of 10000 samples and 100 rules similar to above, with head items now set co-occuring, varying the level of noise in $\{0.1\%, 0.5\%, 1\%, 3\%, 5\%, 10\%\}$ bitflips in the matrix, where 10% noise means more noise than actual signal. EXPLAINN is robust to noise, even when facing almost the same amount of noise and signal (see Fig. 2).

3.2. How neural networks perceive the world

How information is filtered We first consider the *MNIST* data of handwritten digits. We train a simple CNN of 2 convolutional and one fully connected layer using Keras, achieving 99% classification accuracy on test data (see Supp. B.1 for details). We are interested in what the individual filters learn about the digits, and how EXPLAINN reveals shared features across several classes. We compare to average activation maps and single neuron prototypes. Whereas the average activation maps per class do not reveal the purpose of a filter, we see that the rules learned by EXPLAINN, clearly identify which pixels *together* trigger a filter. For example, in filter 2 in layer 1 the prototype looks like a maze and does not reveal any insight, and average activation maps just show the number given by the class, whereas the discovered rules identify shared structure, such as curvatures shared between digits. For filter 36 in layer 2, the discovered rules show that it detects horizontal edges in a class specific manner, whereas prototyping and activation maps again fail to reveal this information. Interestingly, the discovered rules indicate that certain filters learn a negative, with activated areas corresponding to the imprint of the digit. We provide images visualizing rules, prototypes, and average activations in Supp. B.1.

How information flows To understand the inner life of neural networks in a more complex setting, we examine the activations for the *ImageNet* data set of pretrained VGG-S and GoogLeNet architectures (Chatfield et al., 2014; Szegedy et al., 2015). We focus on analyzing the VGG-S results for which an optimized and highly interpretable prototyping method to visualize multiple neurons exists (Øygard, 2016), and provide results for GoogLeNet in Supp. B.2.1. Here, we focus on particular rules, and provide a larger and more diverse set of results in Supp. Fig. 16, 17. We see that rule-derived prototypes generally show highly interpretable features for the corresponding classes. Mining for rules from the output to the last layer, EXPLAINN yields rules with individual heads spanning multiple labels and tails spanning multiple neurons, which together encode the information shared between labels. Examples include the

(a) Visualization for the whole tail

(b) Visualization for the units in the tail individually

Figure 3: *Characteristic faces.* From the data for all dog breed categories, EXPLAINN discovered the rule between the labels {`Japanese spaniel`, `Pekinese`, `Shih-Tzu`, `Lhasa`, `Affenpinscher`, `Pug`, `Brabancon griffon`}, and 5 units from the *FC7* layer, for which a prototype is given in the top image. The units together capture the characteristic face of these breeds, wherease individual units (bottom) give only little insight about the encapsulated information.

faces of certain dog breeds, for which, if we visualize these neurons individually (Fig. 3), it is hard to extract anything meaningful from the images: the information is really encoded in the set of neurons that act together.

We also observe cases where rules describe how the network discriminates between similar classes. We give an example in Fig. 4 for the neurons EXPLAINN discovers to be associated with just huskies, just malamutes, and both of these classes together. These dog breeds are visually similar, sharing a black–white fur pattern, as well as head and ear shapes. These traits are reflected by the neurons corresponding to the rule for both breeds. Looking closer, we can see that distinct traits, the more pointy ears of the husky, respectively the fluffy fur of the malamute, are picked up by the neurons discovered for the individual classes. Beside discovering what shared and distinct traits the network has learned for classes, we also find out when it learns differences *across* samples of the *same* class. As one example, for the dog breed Great Danes, we discover three rules that upon visualization each correspond to visually very different sub-breeds, whereas a simple class prototype does not reveal any such information (Supp. Fig. 15).

Next we investigate the information flow within the network, by iteratively finding rules between adjacent layers, starting with rules $X \rightarrow Y$ from output layer to last fully connected layer FC7. Based on this set of rules, we then apply EXPLAINN to discover rules $Y \rightarrow Z$ between FC7 and FC6, where heads Y are groups of neurons found as tails in the previous iteration. We recursively apply this process until we arrive at a convolutional layer. This gives us traces of neuronal activity by chaining rules $X \rightarrow Y \rightarrow Z \rightarrow \cdots$ discovered in the iterative runs. We visualize two such traces in Fig. 5, which give insight in how the network perceives different classes, passing on information from layer to layer.

One example of a discovered trace is for the class `totem pole` (Fig. 5a). We observe that the set of neurons discovered for FC7 and FC6 each yield prototypes that clearly resemble the animalistic ornaments of such totem poles, which can also be found in the training data. Interestingly, we see that the neuron sets found for different filters of the last convolutional layer CONV5 together detect parts of the object, including the vertical pole, and the rooftop-like structure, decorated with animalistic shapes with eyes, that is typically found at the top of a totem. These filters act in a highly specific manner, detecting only specific parts of the image, such as thinner or wider vertical structures in the center, or objects at the top center of the image.

We also find signs of overfitting, e.g. when considering the information trace for a set of dog breeds (Fig. 5b). Note that due to space, we here only show a subset of the discovered rules. We observe that the prototypes for FC7 and FC6 both show side-views of animals. The networks seems to learn features that are specific to side photos of dogs, which are prevalent in the training data, also indicated by the filter prototypes. For the filters, we see that the network acts on very specific parts of the image, detecting structures at the bottom that resemble paws and pairs of front and hind legs, and at the top of the image, which resemble dog faces and clouds. We also find more abstract features with groups of filters detecting horizontal edges, which reminds of the back of the dog in side-view. While there is room for interpretation of prototypes, the discovered traces provide evidence on how the network perceives the world, as information from prototypes can be interpreted across layers, and in combination with the spatial location of activations in the filters.

(a) Top: Malamute, Bottom: Siberian Husky

`Siberian husky` AND NOT `Malamute`

`Siberian husky` AND `Malamute`

`Malamute` AND NOT `Siberian husky`

(b) Rule prototypes for FC7

Figure 4: *Neurons discriminating Huskies and Malamutes.* a) Huskies and Malamutes are very similar looking dog breeds. b) Prototypes for rules $X \rightarrow Y$ discovered for classes X, `Siberian husky` (red frame), class `Malamute` (yellow frame), resp. both (orange frame) and neurons Y in FC7. The neurons associated with both classes represent typical features shared between the two classes, those associated only with `Siberian huskies` show their slightly sharper, more defined head, while those associated only with `Malamutes` capture their more fluffy fur.

Rules carry class information To quantitatively assess the rules that EXPLAINN discovers, we here consider the VGG-S network for *ImageNet* and intervene on those neurons in the last fully connected layer that EXPLAINN finds to be class-associated. For each class c, we set incoming weights from neurons y to 0, for which we have discovered a rule $X \rightarrow Y, c \in X, y \in Y$, comparing classification rate before and after intervention. As baseline, we additionally intervene on an equally sized random subset of all weights leading to class c, again measuring classification rate after intervention. We see that for all classes, performance drops much more strongly for the actual interventions than for the random ones, in most cases even to 0 (see Fig. 2 bottom). This gives evidence that the discovered rules capture information necessary for classification. We further observe that under intervention the model often predicts closely related classes, e.g. `Fire Salamander` to `Spotted Salamander`, `Barbell` to `Dumbbell`, or `Palace` to `Monastery`, which gives insight towards similarity of classes, robustness of predictions, and therewith sensitivity to adversarial attacks.

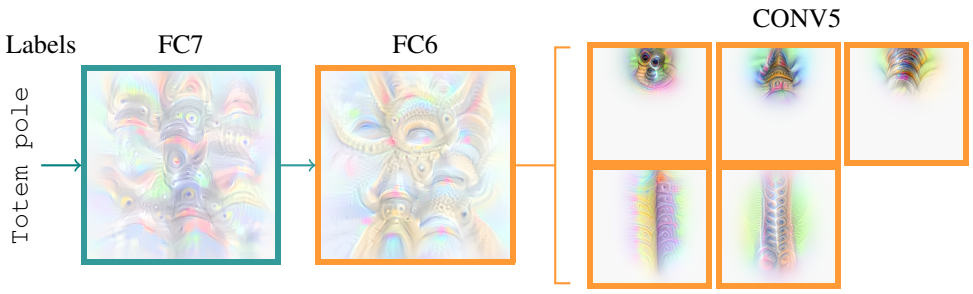
The effect of fine tuning Finally, we show that EXPLAINN provides insight into the effect of fine-tuning in transfer learning. For this we consider Oxford *Flower* data (Nilsback & Zisserman, 2008), which consists of 8k images of flowers of 102 classes. For investigation, we consider both the vanilla VGG-S network trained on *ImageNet* from above, and a fine-tuned version from the Caffee model zoo.[2] We run EXPLAINN to obtain rules between the output and the final layer of both networks. We provide examples in Supp. Fig. 18. The visualizations show, as expected, a strong emphasis on colour and shape of the corresponding flower. Interestingly, the visualizations of the same neurons for the original VGG-S show almost identical shapes and pattern, but with less intense colour, and in both observe prototypes with animal-like features such as eyes or beaks.

[2]https://github.com/jimgoo/caffe-oxford102

4. Discussion and Conclusion

The experiments show that EXPLAINN is able to discover distinct groups of neurons that *together* capture traits shared and distinct between classes, within-class heterogeneity, and how filters are used to detect shared features, segment background, or detect edges locally. Neither of these are revealed by activation maps, which miss the local information that patterns provide, nor by saliency maps, which investigate network attention for an individual image alone. Prototyping is a great tool for visualizing neuron information content, but, by itself is limited by the massive number of possible combinations of neurons, requiring thousands of hours to painstakingly handpick and connect the information of just individual neurons (Olah et al., 2020). Combining EXPLAINN with prototyping permits exploring networks beyond single neurons, by automatically discovering which neurons act in concert, which information they encode, and how information flows through the network.

In particular, we discover distinct groups of neurons in fully connected layers that capture shared respectively distinct traits across classes, which helps in understanding how the network learns generality but still can discriminate between classes. Due to the local information that our rules provide, we can also detect differences in the perception across samples of a single class, where for example different groups of neurons describe visually different sub-breeds of a class of dogs. By connecting rules that we find across several layers, we trace how information is gathered and combined to arrive at a classification, from filters that detect typical class specific features in the image, through fully connected layers where multiple neurons together encode the combined



(a) Information flow for class `Totem Pole`.

(b) Part of an information flow for *{`Black-and-tan coonhound, english foxhound, borzoi, ibizan hound, saluki, scottish deerhound, curly-coated retriever, entle bucher, mexican hairless`}.

Figure 5: *Information flow.* Example rule cascades discovered for *ImageNet*. For each rule $X \rightarrow Y$, the group of neurons of tail Y are used to generate a prototype (images in colored frames). To discover these rule cascades, we first mine rules between output and FC7. We use the tails of these rules (neurons of FC7) as heads to mine rules to the next layer (FC6). Finally, we use the tails of those rules to mine rules between FC6 and CONV5.

information, up to the final classification output. Applying ExplaiNN to investigate the impact of fine-tuning in transfer learning, we found that for groups of neurons in the given fine-tuned CNN, surprisingly, the contained information is almost identical to the original CNN, but capturing the traits of the new classes almost perfectly. For the given task, fine-tuning thus mostly resulted in routing information differently, rather than learning to detect new features.

Overall, ExplaiNN performs well and finds surprising results that help to understand how CNNs perceive the world. While many important tasks are solved by such networks, attention based architectures play an important role in e.g. language processing. Although rules can likely also help to understand what these models learn, these networks encode an entirely different type of information that is inherently hard to understand and visualize, and hence an exciting challenge for future work. Here, our main interest was characterizing information flow through neural networks, and hence, we focused on subsequent layers. ExplaiNN, however, operates on arbitrary sets of neurons, thus naturally allows investigating e.g. residual networks, where the previous *two* layers contribute information. Currently scaling to thousands of neurons, it will make for engaging future work to scale to entire networks at once.

Acknowledgements

Anna Olah and Jonas Fischer are supported by scholarships from the International Max Planck Research School for Computer Science (IMPRS-CS).

References

Adadi, A. and Berrada, M. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago de Chile, Chile*, pp. 487–499, 1994.

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.

Barakat, N. and Diederich, J. Eclectic rule-extraction from support vector machines. *International Journal of Computational Intelligence*, 2(1):59–62, 2005.

Bastani, O., Kim, C., and Bastani, H. Interpreting blackbox

models via model extraction. *CoRR*, abs/1705.08504, 2017.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

Bayardo, R. Efficiently mining long patterns from databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA*, pp. 85–93, 1998.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

Chu, L., Hu, X., Hu, J., Wang, L., and Pei, J. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pp. 1244–1253. Association for Computing Machinery, 2018. ISBN 9781450355520.

Fischer, J. and Vreeken, J. Sets of robust rules, and how to find them. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Würzburg, Germany*. Springer, 2019.

Frosst, N. and Hinton, G. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Gatys, L. A., Ecker, A. S., and Bethge, M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89. IEEE, 2018.

Grünwald, P. and Roos, T. Minimum description length revisited. *International Journal of Mathematics for Industry*, 11(01):1930001, 2019.

Hämäläinen, W. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowledge and information systems*, 32(2):383–414, 2012.

Kontkanen, P. and Myllymäki, P. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233, 2007. ISSN 0020-0190.

Lakkaraju, H., Kamar, E., Caruana, R., and Leskovec, J. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Li, M. and Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, Heidelberg, 1993. ISBN 0387940537.

Moerchen, F., Thies, M., and Ultsch, A. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Knowledge and Information Systems*, 29(1):55–80, 2011.

Mordvintsev, A., Olah, C., and Tyka, M. Inceptionism: Going deeper into neural networks. 2015.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

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.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in/.

Øygard, A. M. Peeking inside convnets. `https://www.auduno.com/2016/06/18/peeking-inside-convnets/`, June 2016.

Özbakır, L., Baykasoğlu, A., and Kulluk, S. A soft computing-based approach for integrated training and rule extraction from artificial neural networks: Difaconn-miner. *Applied Soft Computing*, 10(1):304–317, 2010.

Ras, G., van Gerven, M., and Haselager, P. Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 19–36. Springer, 2018.

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, 2016.

Rissanen, J. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Rissanen, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

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*, 115(3):211–252, 2015.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3145–3153. JMLR.org, 2017.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3319–3328. JMLR.org, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Taha, I. A. and Ghosh, J. Symbolic interpretation of artificial neural networks. *IEEE Transactions on knowledge and data engineering*, 11(3):448–463, 1999.

Tan, S., Caruana, R., Hooker, G., Koch, P., and Gordo, A. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

Tran, S. N. and d'Avila Garcez, A. S. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):246–258, 2018.

Webb, G. I. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *ACM Transactions on Knowledge Discovery from Data*, 4(1):1–20, 2010.

Xie, N., Ras, G., van Gerven, M., and Doran, D. Explainable deep learning: A field guide for the uninitiated. *arXiv preprint arXiv:2004.14545*, 2020.

Zhang, Q., Cao, R., Shi, F., Wu, Y. N., and Zhu, S. Interpreting CNN knowledge via an explanatory graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 4454–4463, 2018.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.