CHARACTERIZING THE ACCURACY/COMPLEXITY LANDSCAPE OF EXPLANATIONS OF DEEP NETWORKS THROUGH KNOWLEDGE EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge extraction techniques are used to convert neural networks into symbolic descriptions with the objective of producing more comprehensible learning models. The central challenge is to find an explanation which is more comprehensible than the original model while still representing that model faithfully. The distributed nature of deep networks has led many to believe that the hidden features of a neural network cannot be explained by logical descriptions simple enough to be understood by humans, and that decompositional knowledge extraction should be abandoned in favour of other methods. In this paper we examine this question systematically by proposing a knowledge extraction method using *M-of-N* rules which allows us to map the complexity/accuracy landscape of rules describing hidden features in a Convolutional Neural Network (CNN). Experiments reported in this paper show that the shape of this landscape reveals an optimal trade off between comprehensibility and accuracy, showing that each latent variable has an optimal *M-of-N* rule to describe its behaviour. We find that the rules with optimal tradeoff in the first and final layer have a high degree of explainability whereas the rules with the optimal tradeoff in the second and third layer are less explainable. The results shed light on the feasibility of rule extraction from deep networks, and point to the value of decompositional knowledge extraction as a method of explainability.

1 INTRODUCTION

Recently there has been an increase in interest in explainable Artificial Intelligence (AI). Although in the past decade there have been major advances in the performance of neural network models, these models tend not to be explainable (Andrew Gordon Wilson, 2017). In large part, this is due to the use of very large networks, specifically deep networks, which rely on *distributed representations* to model data accurately (LeCun et al., 2015). In contrast with symbolic AI, in which specific features are often hand picked for a problem, or symbolic Machine Learning (ML), which takes a localist approach (Richardson & Domingos, 2006), the features used by a distributed representation do not necessarily correlate with obviously identifiable features of the data. A distributed representation may owe its strength to weak statistical correlations that a human would not be able to detect or describe in any comprehensible way.

Knowledge extraction seeks to increase the explainability of neural networks by attempting to uncover the knowledge that a neural network has learned implicitly in its weights. One way of doing this is to translate trained neural networks into a set of symbolic rules or decision trees similar to the ones found in symbolic AI, ML and logic programming (Russell & Norvig, 2010) (Frosst & Hinton, 2017). Rule extraction techniques have been around for decades (Towell & Shavlik, 1993) with a number of rule extraction algorithms having been developed over the years (Murphy & Pazzani, 1991) (Craven, 1996) (Tran & d'Avila Garcez, 2016) (d'Avila Garcez et al., 2001). These techniques generally take one of two approaches: *decompositional*, in which the parameters of the network are used to generate rules, or *pedagogical*, in which the behaviour of the network is used to generate rules (Andrews et al., 1995). In either case, the major issue with rule extraction is the complexity of the extracted rules. Even if it is possible to find a symbolic system which describes exactly a neural network (for example,

feedforward, Boolean, deterministic networks can always be written as a logic program), a very large rule set derived from a very large CNN may be no more comprehensible than the original network.

Perhaps the main reason knowledge extraction proves difficult (and in particular decompositional methods of extraction) is the distributed representations found in neural networks (LeCun et al., 2015). This means that important concepts which can be used for reasoning are not always represented by single neurons but by patterns of activity over many neurons. It has been argued that the distributed nature of neural networks plays an important part in many of their capabilities (Smolensky & Legendre, 2006). Distributed representations have been identified as one of the fundamental properties of connectionism (Smolensky, 1988). This has led many to conclude that attempting to explain latent features using symbolic knowledge extraction is a dead end, and that methods akin to distillation should be adopted instead (Frosst & Hinton, 2017). Distillation has also been proposed as a method for improving robustness but it's efficacy has been questioned (Papernot et al., 2015) (Carlini & Wagner, 2016). Other approaches take a more practical view. Rather than attempting to open the black box, one may settle for some guarantees on the network's behaviour, or for visualizations seeking to explain individual classifications rather than the learned model (Johnson & Li, 2015) (Simonyan et al., 2013) (Katz et al., 2017)

In this paper, we develop a method for empirically examining the explainability of the latent variables in neural networks. We use rule extraction by searching through a space of *M-of-N* rules (Towell & Shavlik, 1993) describing a latent variable, and measuring the error and complexity of each rule. By selecting various error/complexity trade-offs, we are able to map out a rule extraction landscape which shows the relationship between how complex the extracted rules are allowed to be and how accurately they capture the behaviour of a network. When applied to a standard 4-layer CNN trained on fashion MNIST, we find that some layers have very accurate rules whereas this is not the case for others even when using very complex rules. The discovery of a 'critical point' on the rule extraction landscape shows that there is an ideal *M-of-N* rule to describe each latent variable. The accuracy of those rules depends highly on the variable that we are attempting to describe, with the overall explainability trends differing greatly between layers and architectures. All layers showed similarly shaped curves but in the convolutional layers the rules extracted with no penalty in complexity were much more complex relatively than the ones extracted from the fully connected layers with relative complexities over 0.4 in the convolutional layers and complexities of under 0.2 in the fully connected layers. Additionally, it was possible to find rules with near 0% error in the first and final layer whereas rules from the second and third layer could not do much better than 15% error.

In Section 2 we give a brief overview of previous algorithms used for knowledge extraction. In Section 3 we give definitions of accuracy and complexity for *M-of-N* rules and outline the extraction process. In Section 4 we give the experimental results of our rule extraction process for the mapping of the accuracy/complexity landscape before concluding in Section 5.

2 Background and Related Work

One of the first attempts at knowledge extraction used a decompositional approach applied to feedforward networks, in particular the Knowledge-based Artificial Neural Networks (KBANN) (Towell & Shavlik, 1994). This algorithm used the weights of a hidden variable to extract symbolic rules of the form *IF M out of a set of N neurons (or concepts) are activated (or hold) THEN a given neuron (concept) is activated (holds)*, called *M-of-N* rules (Towell & Shavlik, 1993). This was followed by more sophisticated algorithms which generate binary trees in which each node is an *M-of-N* rule (Murphy & Pazzani, 1991) (Craven, 1996) (Notice that these binary trees can be reduced to IF-THEN propositional logic sentences as before). These more recent algorithms are pedagogical in that they select an *M-of-N* rule using the input units as the concepts (called *literals* in logic), based on the maximum information gain with respect to the output. By default these methods do not attempt to explain any of the latent variables of a network and simply treat the model as a black box which can be queried as an oracle to generate data for rule extraction. Many other extraction methods can be described as *eclectic* containing both aspects of pedagogical and decompositional methods (c.f. (d'Avila Garcez et al., 2001) for examples. Other methods abandon the knowledge extraction paradigm and opt for alternative techniques, often more visually oriented (Johnson & Li, 2015) (Frosst & Hinton, 2017). A survey of various methods developed to solve the 'black-box' problem of neural networks can be found in (Guidotti et al., 2018).

Most decompositional rule extraction techniques have been applied only to shallow networks with other techniques focusing on input/output relationships rather than attempting to explain the latent features of a deep network. The multiple hidden layers in a deep network mean that in order to explain an arbitrary hidden feature in terms of the input, a decompositional technique has to produce a hierarchy of rules (see (Tran & d'Avila Garcez, 2016) for an example of hierarchical rule extraction). With many hidden layers, the extracted rules can quickly grow far too complex for a human to understand, unless each constituent of the rule hierarchy is exceedingly simple. Thus, the use of decompositional techniques to explain the features of a deep network end-to-end seems impractical, as argued in (Frosst & Hinton, 2017). Nevertheless, experiments reported in this paper show that some layers of a deep network may be associated with highly explainable rules, and that within a layer some extracted rules may explain the network's behaviour remarkably well in terms of certain features. This opens the possibility of rule extraction being used as a tool for the modular explanation of network models, and it could provide insight into the similarity and disentanglement (Burgess et al., 2018) of latent features by comparing their optimal extracted rules. In what follows, we define the above ideas around rule explainability and optimality formally.

3 LAYERWISE KNOWLEDGE EXTRACTION

3.1 *M-of-N* RULES.

In logic programming, a logical rule is an implication of the form $A \leftarrow B$, called A *if* B. The literal A is called the *head* of the rule and B stands for a conjunction of literals, $B_1 \wedge B_2 \wedge ... \wedge B_n$ called the *body* of the rule. Disjunctions in the body can be modelled simply as multiple rules having the same head. Most logic programs adopt a *negation by failure* paradigm so that A is $true$ if and only if B is $true$ (Fitting, 2002). When using rules to explain a neural network the literals will refer to the states of neurons. For example, if a neuron x takes binary values {0,1} then we define the literal X by $X = True$ if $x = 1$, and $X = False$ if $x = 0$. For neurons with continuous activation values, we can define a literal by including a threshold a such that $X = True$ if $x > a$, and $X = False$ otherwise. In other words, the literal X is shorthand for the statement $x > a$.

In neural networks, a latent variable is usually poorly described by a single conjunctive rule since there are many different input configurations which will activate a neuron. Rather than simply adding a rule for each input pattern that activates a neuron (which essentially turns the network into a large lookup table), we look for ***M-of-N*** rules which have been commonly used in rule extraction starting with (Towell & Shavlik, 1993). ***M-of-N*** rules soften the conjunctive constraint on the body of logical rules by requiring only M of the variables in the body to be true for some specific value of $M < N$ (notice that when $M = N$ we are left with a conjunction). For example, the rule $H \leftarrow 2 - of - \{X_1, X_2, \neg X_3\}$ is equivalent to $H \leftarrow (X_1 \wedge X_2)$ or $(X_2 \wedge \neg X_3)$ or $(X_1 \wedge \neg X_3)$, where $\neg$ stands for negation by failure.

M-of-N rules are an attractive candidate for rule extraction for several reason. First, they offer a compact representation which is more general than a conjunction and which reflects naturally the input/output dependencies of the neurons in a neural network fan-in. Second, ***M-of-N*** rules are only a subset of all propositional formulas (it is easy to see that XOR cannot be represented as an ***M-of-N*** rule), meaning that our rule extraction process will not simply generate a lookup table to explain a neuron. Finally, ***M-of-N*** rules share a structural similarity with neural networks. This can be seen by viewing ***M-of-N*** rules as 'weightless perceptrons'. Any ***M-of-N*** rule can be represented by a perceptron with its output neuron representing the head, and visible neurons representing the body of the rule. In order to encode an ***M-of-N*** rule in a neural network, one just needs to set the bias of the output neuron to M and the weights of each input neuron to 1 or -1 for neurons corresponding to positive or negative literals, respectively. ***M-of-N*** rules have been used in the early days of knowledge extraction but have since been forgotten. This paper brings ***M-of-N*** rules to the forefront of the debate on explainability again.

3.2 GENERATING SPLITS.

When our network has continuous activation values, in order to define the literals to use for rule extraction we must choose a splitting value a for each neuron which will lead to a literal of the form $x > a$. In order to choose splitting values for continuous neurons we use *information gain* (Quinlan,

1986). Given a target neuron we wish to explain, h, we generate a literal for the target neuron by selecting a split based on the information gain with respect to the output labels of the network. That is, given a set of test examples, choose the value of the target neuron h which splits the examples in such a way as to result in the maximum decrease in entropy of the network outputs on the test examples.

The input literals are then generated from the inputs to the target neuron by choosing splits for each input which maximize the information gain with respect to the target literal generated in the previous step. In practice this means that each target literal in a layer will have its own set of input literals, each corresponding to the same set of input neurons but with different splits.

In the case that the layer is convolution, each feature map corresponds to a group of neurons each with a different input patch. Rather than test every single neuron in the feature map we only test the one whose optimal split has the maximum information gain with respect to the network output. This gives us a single rule for each feature map rather than a collection of them.

3.3 Soundness and Complexity.

The two metrics we are concerned with in rule extraction are comprehensibility and accuracy. For a given rule we can define accuracy in terms of a *soundness* measure. This is simply the expected difference between the predictions made by the rules and the network. More concretely given a neuron h in a neural network with input neurons x_i, we can use the network to compute the state of h from the state of the input neurons which then determines the truth of literal H. Thus we can use the network to determine the truth of H, call this $N(x)$. Furthermore, if we have some rule R relating variables H and X_i, we can use the state of the input x to determine the value of the variables X_i, and then use R to determine the value of H, call this $R(x)$. Given a set of input configurations to test I (not necessarily from the test set of the network) we can measure the discrepancy between the output of the rules and the network as

$$E(R) := \frac{1}{|I|} \sum_{x \in I} |R(x) - N(x)| \tag{1}$$

In other words we measure the average error of the rules when trying to predict the output of the network over a test set.

Comprehensibility is more difficult to define as there is a degree of subjectivity. The approach we take is to look at the *complexity* of a rule. Here, we think of complexity in an analogous way to the Kolmogorov complexity which is determined by a minimal description. Thus we determine the complexity of a rule by the length of its body when expressed by a (minimal) rule in disjunctive normal form (DNF). For an ***M-of-N*** rule, the complexity is simply $M\binom{N}{M}$, where $\binom{}{}$ denotes the binomial coefficient. For our experiments we measure complexity in a relative manner by normalizing w.r.t. a maximum complexity. Given N possible input variables the maximum complexity is $\lceil \frac{N+1}{2} \rceil \binom{N}{\lceil \frac{N+1}{2} \rceil}$, where $\lceil\rceil$ denotes the ceiling function (rounding to the next highest integer). Finally in order to control for growth we take the logarithm giving the following normalized complexity measure.

$$C(R) := \frac{\log(M\binom{N}{M})}{\log(\lceil \frac{N+1}{2} \rceil \binom{N}{\lceil \frac{N+1}{2} \rceil})} \tag{2}$$

As an example, suppose we have a simple perceptron whose output unit has a bias of 1 and two binary visible units with weights $w_{1,1} = 1$ and $w_{2,1} = -0.5$. Then consider the rule $h = 1 \iff$ 1-of-$\{x_1 = 1, \neg(x_2 = 1)\}$. Over the entire input space we see that $R(x) \neq N(x)$ only when $x_1 = 0$ and $x_2 = 1$ giving us an error of 0.25. Furthermore, a $1 - of - 2$ rule is the most complex rule possible for 2 variables as it has the longest DNF of any ***M-of-N*** rule giving us a complexity of 1.
Using Eqs. (1) and (2) we define a loss function for a rule R as a weighted sum in which a parameter $\beta \in \mathbb{R}^+$ determines the trade-off between soundness and complexity.

$$L(R) := E(R) + \beta C(R) \tag{3}$$

By using a brute force search procedure with various values of β we are able to explicitly determine the relationship between the allowed complexity of a rule and its maximum accuracy. For $\beta = 0$ the rule with the minimum loss will simply be the rule with minimum error regardless of complexity,

and for β large enough the rule with the minimum loss will be a rule with 0 complexity, either a $1-of-1$ rule or one of the trivial rules which either always predicts true or always predicts false (these can be represented as *M-of-N* rules by $0-of-N$ and $N+1-of-N$ respectively).

3.4 SEARCH TECHNIQUE (EXTRACTION ALGORITHM).

Given a neuron h_j with n input neurons x_i, we generate splits for each neuron using the technique just described to give us a set of literals H_j and X_i. Then, we negate the literals corresponding to neurons which have a negative weight to h_j. Using these we search through $\mathcal{O}(n^2)$ *M-of-N* rules with variables X_i in the body and H_j in the head, which minimize $L(R)$. To do this we reorder the variables according to the magnitude of the weight connecting x_i to h_j (such that we have $|w_{1,j}| \geq |w_{2,j}| \geq ... \geq |w_{n,j}|$). Then we consider the rule $M-of-\{X_1,...,X_N\}$ for each $1 \leq N \leq n$ and each $0 \leq M \leq N+1$. The search procedure only relies on the ordering of the variables X_i.

Algorithm 1 Search procedure for finding *M-of-N* rules to explain a hidden feature h

Generate a split, s, for h by choosing the value which maximizes the information gain with respect to the network output. Use this to define the literal H
for Each neuron x which is an input of h, **do**
  Generate a split for x by choosing the value which maximizes the information gain with respect to H. Use this value to define the literal X if the connection between x and h is positive, and use it to define $\neg X$ otherwise
end for
Order the input literals by the magnitude of their weights
for $N : 1 \leq N \leq$ *number of inputs* **do**
  for $M : 1 \leq M \leq N$ **do**
    Create an $M-of-N$ rule, R, whose body consists of the first N literals. Then compute $L(R)$;
  end for
end for
Compute $L(R)$ for the trivial rules $0-of-\{\}$ and $1-of-\{\}$;
return rule with the lowest value of $L(R)$.

A neuron with n input neurons has $\mathcal{O}(2^n)$ possible *M-of-N* rules which makes an exhaustive search intractable. However, here we rely on the assumption that the most accurate *M-of-N* rules with N literals use the literals corresponding to the N neurons with the strongest weights (i.e. the highest weight values). This assumption is easily justified by the conditional independence of hidden units in all layers except for the final one. Here, since we use a softmax function, the conditional independence of the output neurons is no longer valid. One way around this would be to order the literals by their information gains rather than their weights. However, the high accuracy found in the experimental results of the rules extracted from softmax layer when ordering the literals by weight seem to suggest that this is not necessary.

By defining an order on the literals we reduce an exponential search space to a polynomial one. However, this is still computationally difficult when a large number of test examples and input neurons exist. In order to complete the rule extraction in a reasonable time, the algorithm was implemented in Spark and run on IBM cloud services. The examples used to measure the accuracy of the extracted rules were taken from the training set rather than the test set as we evaluate the accuracy of the rules with respect to the output of the network rather than the actual output and using examples that the network was trained on was deemed to be a better representation of the behaviour the network learned. By running the search in parallel, we can map the accuracy/complexity graph for about 50 hidden neurons in the second and third layer in several hours. Increasing the number of examples used in the accuracy calculation greatly increases the time taken and for this reason we only use 1000 examples.

3.5 VISUALIZING RULES EXTRACTED FROM THE FIRST LAYER

To demonstrate the procedure we will examine the extraction process for the first hidden feature in the CNN trained on the fashion MNIST data set. First we select 1000 random input examples from the training set and use them to compute the activations of each neuron in the CNN as well as the predicted labels of the network. Since the CNN pads the input with zeros and the input images are 28×28 we have $28 \times 28 = 784$ neurons per feature in the first layer. Each of these neurons correspond to a different 5×5 patch of the input. To select a neuron to test we find the optimal splitting value of each neuron by computing the information gain of each neuron with respect to the networks predicted labels. For the first feature we find that the neuron with the maximum information gain is neuron 96 which has an information gain of 0.015 when split on the value 0.0004. This neuron corresponds to the image patch centered at $(96/28, 96\%28) = (3, 12)$. With this split we define the variable H by $H := 1$ iff $h_{96} \geq 0.0004$.

Using this variable we define the input splits by choosing the values which result in the maximum information gain with respect to H. Note our test input consists of the 1000 5×5 image patches centered at $(3, 12)$ taken from the input examples. We then search through the M-of-N rules whose bodies consist of the input variables defined by the splits to determine an optimal *M-of-N* rule explaining H for various error/complexity tradeoffs. As we increase the complexity penalty we extract three different rules which are visualized in Figure 1. We can see from the figure that many of

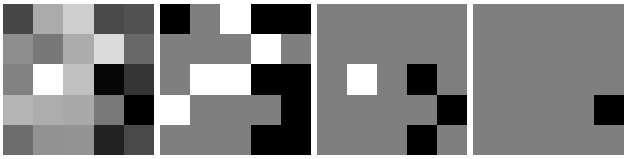

Figure 1: An example of the various rules extracted for the first feature. The first image represents the weights of the neruon and the following three are rules of decreasing complexity explaining the neuron. Here grey indicates that the input feature is not included in the *M-of-N* rule, white indicates a positive literal, and black indicates a negative literal

the weights are filtered out by the rules. The most complex rule is a 5-of-13 rule which has a 0.025 error, or in other words the output of the rule agrees with the network 97.5% of the time. Adding a mild penalty to complexity changes the optimal rule to the much simpler 3-of-4 rule but raises the error to 0.043. Finally a heavy penalty to complexity produces the trivial 1-of-1 rule which has the significantly higher error of 0.13

4 EXPERIMENTAL RESULTS

4.1 DNA PROMOTER

In order to demonstrate the results on a simple network known to be amenable to rule extraction we apply the technique to the classic example of the DNA promoter dataset. Training a feed forward network with a single hidden layer of 100 nodes We find that, like we will see with the Fashion MNIST dataset, the relationship between complexity and error in the first layer is exponential, suggesting an ideal complexity/error tradeoff (See Fig 2). Furthermore, in the output layer we find that the rule $1 - of - \{H_{39}, H_{80}\}$ gives 100% fidelity to the network. Since the splits for the hidden layer are defined only by the information gain with respect to the output we can describe each of the literals in our *1-of-2* rule with an *M-of-N* rule extracted from the input layer. Extracting with no complexity penalty, the rules in question are of the form *64-of-119* for the variable $H_3 9$ which produces a single incorrect classification of $H_3 9$, and a rule of the form *32-of-61* for the variable $H_8 0$ which produces no incorrect classifications for $H_8 0$. Denoting L_{119} as the set of literals in the rule explaining $H_3 9$ and L_{61} as the set of literals in the rule explaining $H_8 0$ the output of the network can be predicted by the rule $1 - of - \{32 - of - L_{61}, 64 - of - L_{119}\}$. Or in other words $\{32 - of - L_{61}\} \vee \{64 - of - L_{119}\}$. In general the errors through a layer propagate when stacking rules that don't perfectly approximate each layer. This is exacerbated by the fact that in most networks, when we choose splits for the input neurons of a layer and then move down a layer we will

end up choosing different splits for the same layer when they are treated as output. In order to replace a network with an end to set of heirarchical rules we must decide on a single set of splits for each layer. This can be done by simpy moving down one layer at a time and selecting the input splits based on the information gain against all configurations of the new output layer but it introduces more error. With this in mind we conduct our experiments layer by layer independently in order to provide an idealized complexity/error curve that can provide a baseline for the best achievable rules when doing rule extraction with *M-of-N* rules. By doing this we can examine under what circumstances *M-of-N* rule extraction might be useful as well as provide a method for evaluating other extraction algorithms.

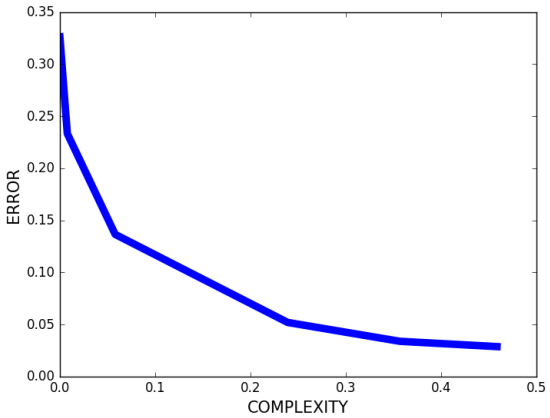

Figure 2: The Complexity/Error relationship for rules extracted from the first layer of a network with a single hidden layer trained on the DNA promoter dataset

4.2 FASHION MNIST

In order to examine the rule extraction landscape of a neural network trained on a practical example, we tested the layerwise rule extraction search on a basic CNN trained on fashion MNIST in tensorflow. The CNN had a standard architecture with a 32 feature convolutional layer with a 5×5 convolutional window followed by a 2×2 max pooling layer, then a 64 feature convolution with a 5×5 convolutional window and another 2×2 max pooling layer followed by a final hidden fully connected layer of 1024 units. All units used a rectified linear activation function; 1000 random inputs from the fashion MNIST training data were chosen to test extracted rules against the network. For the convolutional layers each feature was tested on the image patches corresponding to the maximum information gain as described in Section 3.2. For the third layer 50 features were chosen at random to test. In the third layer we limited the search to rules with 1000 literals or less to save time as it was very rare for even the most complex rules extracted to have over 900 literals. For the final layer the output was tested as 10 one-hot neurons, each of which underwent the rule searching procedure. For each layer, we repeated the search procedure for 5 different values of β, $0, 0.1, 0.2, 1, 5, 1000$. This produced 5 different sets of extracted rules each with a different error/complexity trade-off. For a given value of β, we averaged the complexities and errors of rules extracted for each target neuron. This allowed us to produce a graph mapping out the error/complexity landscape for rules extracted from each layer (see Figure 1.)

Here we can see that the Complexity/Error trade-off for the extracted rules from each layer are different. For the first and final layers we were able to extract quite accurate rules approaching a near 0 error whereas the second and third layer had a similar accuracy/complexity tradeoff with second layer showing a very slight improvement in accuracy with much more complex rules whereas the minimum error of the third layer is not able to be improved at all with increasingly complex rules. Additionally the trivial rules perform much worse on the second than the third layer.

Importantly we see that the optimal accuracy/complexity tradeoff is not simply a function of the number of input nodes since the third layer performs about as well as the second layer despite

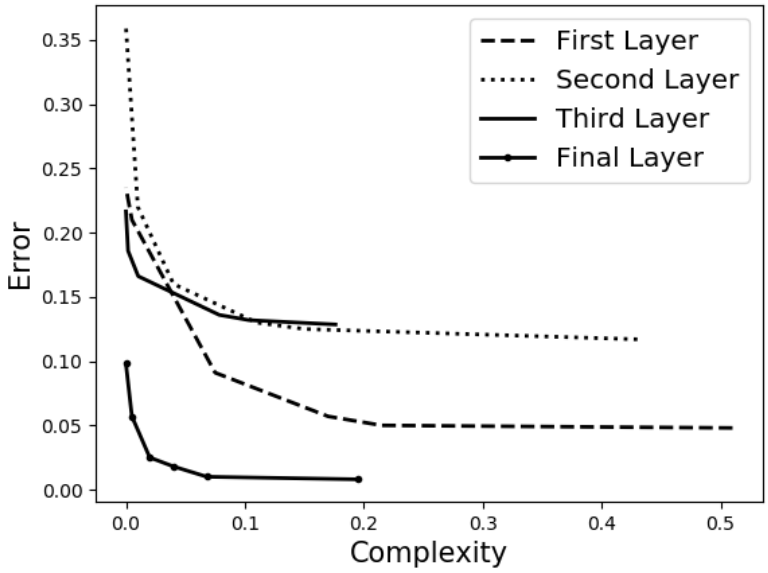

Figure 3: The Complexity/Error relationship for rules extracted from each layer of a CNN.

having 3136 vs 800 input nodes. Similarly the final layer provides much more accurate rules that are relatively less complex than the first layer despite having 1024 vs 25 input nodes.

Despite the actual values varying for each layer, the results paint a similar picture for the shape of the rule extraction landscape in each case. Rather than a gradual decline in accuracy there is a critical point at which the error starts to increase rapidly as the penalty on complexity increases. This suggests that there is a 'natural' set of rules for explaining the latent features. Although ultimately the trade off is subjective, the set of rules existing at the critical points cannot be made significantly more accurate if they are more complex, and will start to do exponentially worse if they are made any simpler. Current rule extraction algorithms do not explicitly take complexity into account for their optimization. Although they may include steps to attempt to simplify the extracted rules, it is not clear *a priori* where on the Error/Complexity graph these rules will land. To the best of our knowledge, this paper is the first to make rule complexity an integral part of the extraction algorithm, as exemplified in the analysis of Figure 1.

Empirical evaluation of popular extraction algorithms should be an important step in their validation. The limitations, as well as potential, of rule extraction algorithms are also outlined in these results. In certain cases, such as we see in the second and third layer of the CNN, there are simply no simple explanations for the features in terms of their input with even the most complex rules having an error rate of 15%. However, in other cases, such as in the final layer in the CNN, the behaviour of the output neurons can be accurately captured by relatively simple rules with extracted rules from the final layer obtaining near 0% error even for rules with complexity of under 0.05. This lends weight to the idea that decompositional rule extraction as a general method of explainability is not possible, but opens up the possibility of selective use of decompositional algorithms depending on which layer we wish to explain in terms of the previous layer.

5 Conclusion

The black box problem of neural networks presents an obstacle to their deployment into society. The black box problem has been an issue for neural networks since their creation, but as neural networks have become more integrated into society, the need for explainability has attracted considerably more attention. The success of knowledge extraction in this endeavor has overall been mixed with most large neural networks today remaining difficult to interpret and explain. Traditionally

knowledge extraction has been a commonly used paradigm and it has been applied to various tasks. Critics, however, point out that the distributed nature of neural networks makes the specific method of decomposition rule extraction unfeasible as individual latent features and unlikely to represent anything of significance. We test this claim by applying a novel search method for *M-of-N* rules to explain the latent features of a CNN, and find that generally latent features can be described by an 'optimal' rule representing an ideal error/complexity trade-off for the explanation. We do this by including rule complexity as an explicit measure in the search for extracted rules. The large discrepancy in this trade-off between neurons in different layers, neurons in different layers with different architectures, and even different neurons in the same layer, suggests that rule extraction as a general technique is unlikely to provide adequate descriptions for all, or even most latent variables.

However, the fact that in many cases the explanations can be made much simpler without reducing the accuracy of the rules suggests that rule extraction can be a useful tool when examining networks with features that are likely to be easily understandable. These results indicate that decompositional rule extraction may still be an important tool for understanding the behaviour of networks. Further research would examine the effects on the accuracy/interpretability landscape of using different transfer functions, other data sets, different architectures, and various forms of regularization of the learning.

REFERENCES

Patrice Simard Rich Caruana William Herlands Andrew Gordon Wilson, Jason Yosinski (ed.). *Proceedings of NIPS 2017 Symposium on Interpretable Machine Learning*, December 2017.

Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373 – 389, 1995.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. *ArXiv e-prints*, April 2018.

Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, 2016. URL `http://arxiv.org/abs/1607.04311`.

Mark William Craven. *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, 1996.

Artur d'Avila Garcez, Krysia Broda, and Dov Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125(1):155 – 207, 2001. ISSN 0004-3702.

Melvin Fitting. Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1):25 – 51, 2002. ISSN 0304-3975. Mathematical Foundations of Programming Semantics 1996.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML*, pp. 879–888, 2017.

Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *CoRR*, abs/1802.01933, 2018. URL `http://arxiv.org/abs/1802.01933`.

Andrej Karpathy Justin Johnson and Fei-Fei Li. Visualizing and understanding recurrent networks. 2015.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Relluplex: an efficient smt solver for verifying deep neural networks. In *Computer Aided Verification*, pp. 97–117, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(2):436—-444, 2015.

Patrick M. Murphy and Michael J. Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. Technical report, 1991.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015. URL http://arxiv.org/abs/1511.04508.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986. ISSN 1573-0565.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-5833-1. URL http://dx.doi.org/10.1007/s10994-006-5833-1.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearsonl, 3 edition, 2010.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL http://arxiv.org/abs/1312.6034.

Paul Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11(1): 1–23, 1988.

Paul Smolensky and Géraldine Legendre. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic GrammarVolume I: Cognitive Architecture (Bradford Books)*. The MIT Press, 2006.

Geoffrey G. Towell and Jude W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, 1993.

Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70:119–165, 1994.

Son Tran and Artur d'Avila Garcez. Deep logic networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):246–258, 2016.