SWITCHED LINEAR PROJECTIONS AND INACTIVE STATE SENSITIVITY FOR DEEP NEURAL NETWORK INTERPRETABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce *switched linear projections* for expressing the activity of a neuron in a ReLU-based deep neural network in terms of a single linear projection in the input space. The method works by isolating the active subnetwork, a series of linear transformations, that completely determine the entire computation of the deep network for a given input instance. We also propose that for interpretability it is more instructive and meaningful to focus on the patterns that deactive the neurons in the network, which are ignored by the exisiting methods that implicitly track only the active aspect of the network's computation. We introduce a novel interpretability method for the *inactive state sensitivity* (Insens). Comparison against existing methods shows that Insens is more robust (in the presence of noise), more complete (in terms of patterns that affect the computation) and a very effective interpretability method for deep neural networks.

1 INTRODUCTION

It is notoriously hard to interpret how deep networks accomplish the tasks for which they are trained. At the same time, due to the pervasiveness of deep learning in numerous aspects of computing, it is increasingly important to gain understanding of how they work. There are risks associated with the possibility that a neural network might not be "looking" at the "right" patterns (Nguyen et al.; Geirhos et al., 2019), as well as opportunities to learn from the network's capable of *better than human* performance (Sadler & Regan, 2019). Hence, there is ongoing effort to improve the interpretation and interpretability of the internal representation of neural networks.

What makes this interpretation of the inside of a neural network hard is the high dimensionality and the distributed nature of its internal computation. Aside from the first hidden layer, neurons operate in an abstract high-dimensional space. If that was not hard enough, the analysis of individual components of the network (such as activity of individual neurons) is rarely instructive, since it is the intricate relationships and interplay of those components that contain the "secret sauce". The two broad approaches to dealing with this complexity is to either use simpler interpretable models to approximate what a neural network does, or to trace back the elements of the computation into the input space in order to make the internal dynamics relatable to the input. In the latter approach we are typically interested in neurons' *sensitivity* – how the changes in network input affect their output, and *decomposition* – how different components of the input contribute to the output.

In this paper we propose a straightforward and elegant method for expressing the computation of an arbitrary neuron's activity to a *single linear projection* in the input space. This projection consists of a *switched weight vector* and a *switched bias* that easily lend themselves to sensitivity analysis (analogous to gradient-based sensitivity) and decomposition of the internal computation. We also introduce a new approach for interpretability analysis, called *inactive state sensitivity* (Insens), which uses switched linear projections to aggregate the contribution of patterns in the input that *deactivate* neurons in the network. We demonstrate on several networks and image-based datasets that, in comparison to existing interpretability techniques, Insens provides a more comprehensive picture of a deep network's internal computation. The only constraint for the proposed methods is that the network must use ReLU activation functions for its hidden neurons.

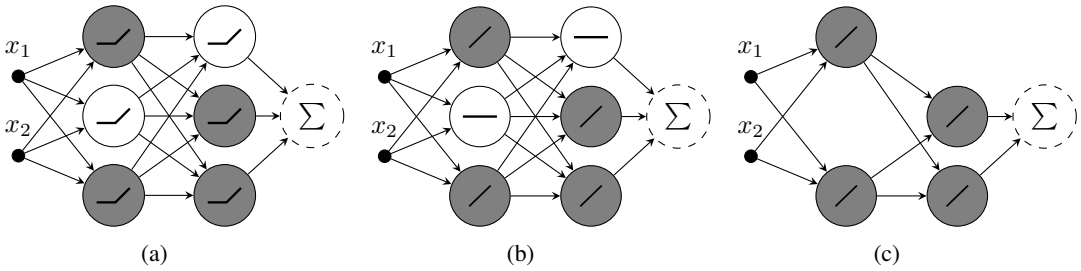

Figure 1: Let's assume that for a particular input $[x_1\ x_2]$ going into the ReLU network shown in (a) the white neurons are *inactive*; then, for this particular input, the network from (a) is equivalent to network in (b) where the inactive neurons are treated as *dead* and the *active* ones operate in the linear part of their ReLU activation fuction; which makes both of these networks equivalent to one in (c); the grey hidden neurons form the active subnetwork.

2 Related work

Previous work on deep learning interpretability is extensive with a wide variety of methods and approaches – Simonyan et al. (2014); Zeiler & Fergus; Bach et al. (2015); Mahendran & Vedaldi; Montavon et al. (2017); Sundararajan et al. (2017); Zhou et al. (2019) being just a selection of the most prominent efforts in this area. Our work on the single linear projection follows the approach akin to Lee et al. and Erhan et al. (2009), where the objective was to interpret the computation performed by an arbitrary neuron for a particular input vector as a projection in the input space. However, whereas these previous attempts were based on Deep Belief Nets (Hinton et al., 2006) and required an approximation of the said projection, our method is a forward computation that gives the neuron's activity in terms of a linear projection in the input space. It works for any neural network, including convolutional ones, as long as all hidden neurons use piecewise linear activation functions.

All existing methods for interpretability of deep learning, due to the nature of ReLU computation, necessarily provide information only about the active subnetwork of the ReLU-based architecture. Our own observations, as well as other evidence showing that in practice neural networks produce a relatively low number of activation regions (Hanin & Rolnick, 2019), lead us to the hypothesis that the analysis of the patterns in the input that *switch neurons off* gives a better picture of a network's sensitivity. We also take the view that too much *interpretation* in interpretability introduces the risk of showing us what we expect to see and *not* what the network is actually focussing on. For instance, in Deep Taylor Decomposition (Montavon et al., 2017) choices of different root-points for the decomposition of the relevance function lead to different rules for Layerwise Relevance Propagation (LRP)(Bach et al., 2015), which can lead to different interpretations of what is important in the input. The LRP-$\alpha_1\beta_0$ rule, for example, emphasises the computation over the positive weights in the network while discounting the relevance of the information passing through the negative weights. This rule is justified by assumptions about desired properties of the explanation, but this comes with a risk of analysis that gives the answer we *want* to get and not what is truly happening inside the network. Insens is an attempt to take into account the patterns in the input that *cause* the neurons inside the network to produce zero output. The information related to the inactive network may seem irrelevant, since inactive neurons do not directly contribute to the computation of the overall output. However, there is *something* in the input that switches a particular set of neurons off, thus regulating the active computation, and as we show in this paper, this *something* carries a lot of meaningful information.

3 Switched linear projections

The basis of the switched network concept is the fact that neurons that produce output of zero do not contribute to the computation of the overall output of the network. The notion of *dead* neurons, that is neurons that always output zero, is not new, nor is the realisation that these neurons, along with their connecting weights, can be taken out the network without any impact on the computation. In a switched projection, we treat the zero-output neurons as temporarily *dead* for a given instance of input. We refer to these neurons as *inactive*, since they may become *active* for a different network

input. Thus we isolate the subnetwork of the active neurons in a given computation. As it happens, for a ReLU neuron the active neurons are those that pass their activity, the weighted sum of its inputs plus bias, directly to its output[1]. This means that a subnetwork of active ReLU neurons is just a series of linear transformations, which is equivalent to a single linear transformation. As a result, we can express the computation performed by any neuron in a ReLU network as a projection onto a switched weight vector in the input space plus the switched bias. The term *switched* indicates that this weight and bias vector changes when the state of the network changes, the state corresponding to the particular combination of the active and inactive neurons in the network. Figure 1 illustrates the concept graphically, and a formal description is given in the following theorem:

Theorem 1 (Switched linear projections)**.** *Let $\mathbf{x} \in \mathcal{R}^d$ be a vector of inputs, $\mathbf{w}_{li} \in \mathcal{R}^{U_{l-1}}$ the weight vector, and $b_{li} \in \mathcal{R}$ the bias of neuron i in layer l (with U_{l-1} inputs from the previous layer). Let the activity of a neuron i in layer l be defined as:*

$$v_{li}(\mathbf{x}) = \Big(\dots \sigma_r\big(\sigma_r(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\big) \dots \Big)\mathbf{w}_{li} + b_{li}, \tag{1}$$

where $\mathbf{W}_l = \big[\mathbf{w}_{l1}^T \ \dots \ \mathbf{w}_{lU_l}^T\big]$, T denotes transpose, $\mathbf{b}_l = [b_{l1} \ \dots \ b_{lU_l}]$ and $\sigma_r(v) = \max(v, 0)$ is the ReLU activation function. If we define an input-dependent state of the network as
$\mathbf{W}_l^{(\mathbf{x})} = \big[\dot\sigma_r\big(v_{l1}(\mathbf{x})\big)\mathbf{w}_{l1}^T \quad \dots \quad \dot\sigma_r\big(v_{lU_l}(\mathbf{x})\big)\mathbf{w}_{lU_l}^T\big]$ *and*
$b_l^{(\mathbf{x})} = \big[\dot\sigma_r(v_{l1}\big(\mathbf{x})\big)b_{l1} \quad \dots \quad \dot\sigma_r\big(v_{lU_l}(\mathbf{x})\big)b_{lU_l}\big]$,
where $\dot\sigma_r(v) = \frac{d\sigma_r(v)}{dv}$, then for

$\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) = \mathbf{W}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T$ *and*

$\widehat{b}_{li}(\mathbf{x}) = \mathbf{b}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \mathbf{b}_2^{(\mathbf{x})}\mathbf{W}_3^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \dots + \mathbf{b}_{l-1}\mathbf{w}_{li}^T + b_{li}$, *we have*

$$v_{li}(\mathbf{x}) = \mathbf{x}\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) + \widehat{b}_{li}(\mathbf{x}). \tag{2}$$

The proof is provided in Appendix A. Note that the ReLU derivative, $\dot\sigma_r(v)$, is just a convenient definition for a step function, so that

$$\dot\sigma_r(v)\mathbf{w} = \begin{cases} \mathbf{w} & , v > 0 \\ \mathbf{0} & , \text{otherwise.} \end{cases} \tag{3}$$

To simplify the notation, whenever referring to the parameters of the switched projection $\widehat{\mathbf{w}}$, $\widehat{b}$, as well as activity v, we will drop the explicit dependency on $\mathbf{x}$.

While Figure 1 illustrates the switching concepts on a small fully connected network, switched linear projections can be computed for networks with convolutional as well as pooling layers. A convolutional layer is just a special case of a fully connected layer with many weights being zero and groups of neurons constrained to share the weight values on their connections. For max pooling, the neurons that do not win the competition, and thus their output does not affect the computation from then on, are deemed to be *inactive* regardless of the output they produce.

3.1 SENSITIVITY

Equation 2 makes it obvious that a given neuron's switched weight vector is just the derivative of its activity with respect to the network input, $\widehat{\mathbf{w}} = \frac{\partial v(\mathbf{x})}{\mathbf{x}}$. Thus, the switched weight vector is analogous to gradient-based sensitivity analysis. Figure 2 shows the heat-maps of the switched weight sensitivity for the same set of hidden neurons with different inputs from the MNIST-trained 2CONV neural network (for details on the network architectures featured in this paper see Appendix B). In this visualisation we show normalised $\widehat{\mathbf{w}}$, with intensity of red corresponding to the larger positive value, and the intensity of blue the negative value. The neurons were chosen from the 2nd

[1] In our terminology, activity denotes output before the activation function and an active neuron is one that produces non-zero output after the activation function; for a ReLU neuron the active and inactive neurons are those that have positive and negative activity respectively.

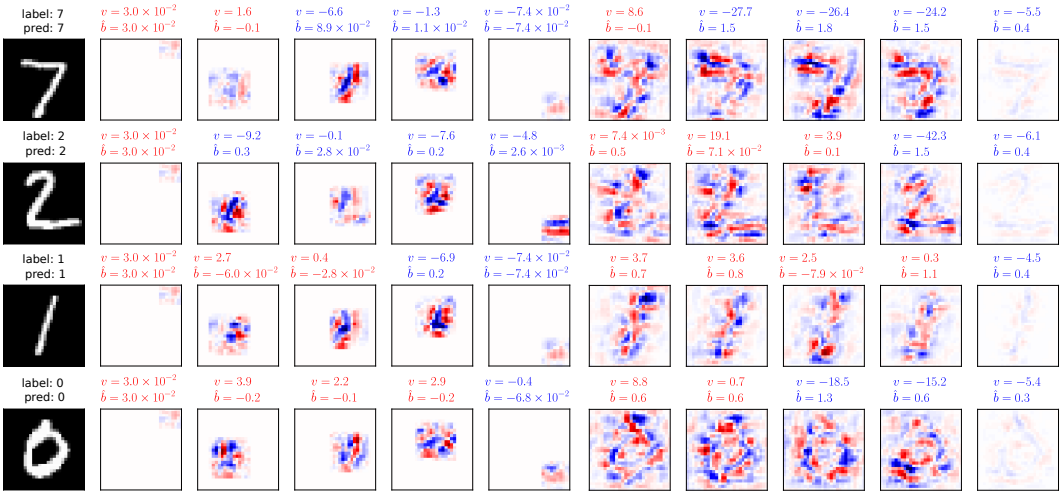

Figure 2: The heat-maps of $\widehat{\mathbf{w}}$ of arbitrary neurons from 4 different MNIST inputs (first column), in the second convolutional layer (next 5 columns) and and the first fully connected layer (last 5 columns) of the 2CONV neural network; the heatmaps have been normalised across the 5 neurons of each layer with intensity of red and blue respectively corresponding to the magnitude of the positive and negative value (with white indicating 0); the activity v and the switched bias $\hat{b}$ are shown above the heatmap of each neuron.

convolutional layer and the penultimate fully connected layer respectively such that for the four considered inputs some neurons were always active, some inactive, and others sometimes active and sometimes not.

Figure 2 makes it clear that a given neuron is not necessarily sensitive to the same pattern for different network inputs – this is most evident in the sensitivity of the neurons of the fully connected layer. Also note that some neurons in the convolutional layers are active despite the fact that they only "see" the part of the input image that is "empty" (all pixels are black). This leads us to the conclusion that a given neuron is not necessarily a detector of a particular pattern in the input space, which is often the underlying assumption of the existing interpretability techniques.

Something also apparent in our switched projection analysis, though not evident from Figure 2, is that for a given input most of the neurons in the network are inactive. On average, only 17% of the neurons were active for a given MNIST input in this architecture. The fact that only a subset of neurons are active in a given computation is not a quirk of one specific network, as observed by (Hanin & Rolnick, 2019). Switched linear projections give us an interpretation of a deep network as a set of linear, input-dependent, transformations. Something about the input activates a subset of neurons, but more importantly, it keeps all the remaining neurons inactive. Traditional sensitivity analysis, as well as the one shown in Figure 2, shows a direction (or magnitude) of the gradient of the input that would increase neurons activity provided the same state of the network remained unchanged. However, often in these networks the state does change even after small perturbations of the input, often resulting in same classification, but different input gradient. We reason that the analysis of the state, including information about what makes the neurons inactive, is more meaningful for interpretability than analysis of the active subnetwork alone.

3.2 Decomposition

Switched linear projections can decompose the activity of a neuron into contributions from its input, in our case the pixels. For this we propose another re-interpretation of the computation of the output that will allow us to distribute the bias over the attributes of the input vector. Note that for a linear projection

$$v = \mathbf{x}\mathbf{w}^T + b = (\mathbf{x} - \mathbf{c})\mathbf{w}^T, \tag{4}$$

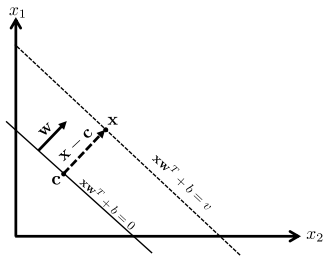

Figure 3: Graphical representation of the concept of a neuron's centre $\mathbf{c}$ in a 2D scenario, where input vector is $\mathbf{x} = [x_1\ x_2]$; the diagram shows input-space coordinate system – the neuron-specific one would have its origin at $\mathbf{c}$.

where $\mathbf{c} = \mathbf{x} - \frac{v}{\mathbf{w}\mathbf{w}^T}\mathbf{w}$, $b = -\mathbf{c}\mathbf{w}^T$, and $\mathbf{c} \in \mathcal{R}^d$. The vector $\mathbf{c}$ can be thought of as a translation of the coordinate system to a neuron-centered one, where $\mathbf{w}$ goes through the origin at $\mathbf{c}$. Montavon et al. (2017) call this vector *the nearest root point*, but we will refer to it as the neuron's *centre*. Figure 3 provides a geometric interpretation of $\mathbf{c}$ in a simple 2D scenario. Since $\mathbf{c} \in \mathbf{R}^d$, we can break down the computation of v such that

$$v = \sum_{j=1}^{d} (x_j - c_j) w_j, \tag{5}$$

and take $(x_j - c_j)w_j$ to be the contribution of the input component j (in our examples, a single pixel) to the computation of neuron's activity. Since the switched linear projection is equivalent to weights and bias of a single neuron, we can compute the *switched centre*

$$\widehat{\mathbf{c}} = \mathbf{x} - \frac{v}{\widehat{\mathbf{w}}^T\widehat{\mathbf{w}}}\widehat{\mathbf{w}}. \tag{6}$$

Visualisations based on decomposition of active neurons do not offer anything more than existing methods. However, the concept of neuron's centre will be used in the method we propose next, for interpretability based on the sensitivity of the inactive network.

4 Inactive state sensitivity

Since a switched linear projection can be found for any neuron in the network, it can just as easily relate what it is about the input that drives a neuron into the negative just as well as positive. Tracking the patterns in the input that make the neuron more inactive tells us about the aspects of the input that would ensure the stability of the network's state. The bigger the magnitude of the negative activity, the less likely the inactive neurons are to switch on, and thus change the switched projection. Some of the inactive neurons are closer and others further away from the point where they would activate. Hence, we propose a definition of inactive sensitivity based on switched linear projection that takes the magnitude of activity into account,

$$\widehat{\boldsymbol{\omega}}_i = \mathbf{x} - \widehat{\mathbf{c}}_i = \frac{v_i}{\widehat{\mathbf{w}}_i\widehat{\mathbf{w}}_i^T}\widehat{\mathbf{w}}_i, \tag{7}$$

where $v_i = (\mathbf{x} - \widehat{\mathbf{c}}_i)\widehat{\mathbf{w}}_i^T$. The difference $\mathbf{x} - \widehat{\mathbf{c}}_i$ can be thought of as the component of the input that projects onto a neuron's switched weight vector $\widehat{\mathbf{w}}_i$, and thus is responsible for the activity v_i. Note that $\widehat{\boldsymbol{\omega}}_i$ is still a vector in the direction of $\widehat{\mathbf{w}}_i$, and so it is a measure of a neuron's sensitivity, but its magnitude is proportional to the absolute value of activity. In an attempt to capture the information about the state of the entire network, we propose averaging the sensitivity of all inactive neurons in the network:

$$\widehat{\Omega}^{(\mathbf{x})} = \frac{1}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \widehat{\boldsymbol{\omega}}_i, \tag{8}$$

where $\mathcal{I}$ is a set of all inactive neurons, $v_i \leq 0$ in the network (excluding the output neurons), and $\widehat{\mathbf{c}}_i$ is the switched centre of neuron i. We refer to $\widehat{\Omega}^{(\mathbf{x})} \in \mathcal{R}^d$ as the *inactive state sensitivity* (Insens) of the network with respect to the input $\mathbf{x}$. The neurons that win maxpooling competitions are not included in the aggregation of $\widehat{\Omega}^{(\mathbf{x})}$ for the reason that the winner is either an active neuron, or one of the inactive neurons from the previous convolutional layer, which is already accounted for.

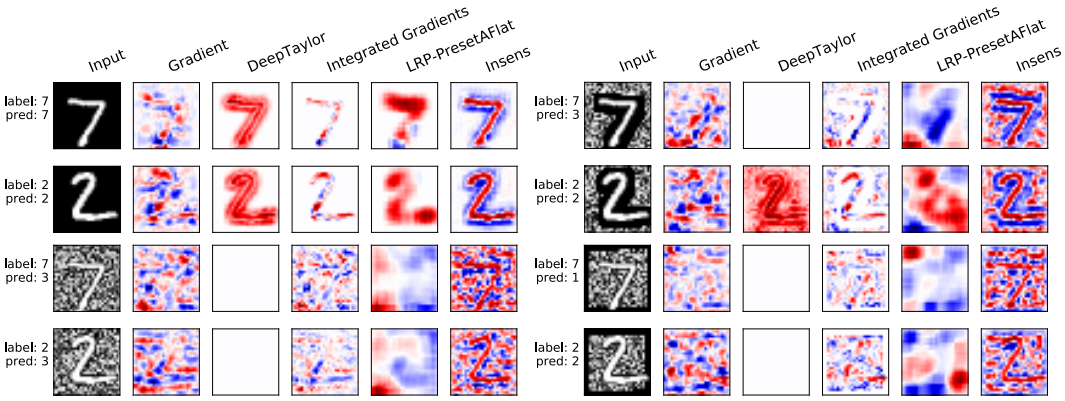

Figure 4: Visualisations for a set of interpretability methods of a single 2CONV neural network trained on the clean MNIST data in response to clean as well as noisy input; top-left shows clean MNIST, top-right noisy bordered MNIST, bottom-left noisy background MNIST and bottom-right noisy framed MNIST; the intensity of red and blue correspond respectively to the magnitude of the positive and negative values in the heatmaps.

4.1 Evaluation

We evaluated Insens interpretability against plain gradient sensitivity, Deep Taylor decomposition (Montavon et al., 2017), Integrated gradients (Sundararajan et al., 2017) and Layerwise relevance propagation (Bach et al., 2015) as implemented by the iNNvestigate toolbox (Alber et al., 2018). For visualisations of the Insens-based interpretability we simply show $\widehat{\Omega}^{(\mathbf{x})}$ as a heatmap, with intensity of the red pixels relating the magnitude of its positive components, and the intensity of blue the magnitude of the negative component of the vector. Since the individual $\widehat{\boldsymbol{\omega}}_i$ gives a weighted gradient with respect to the input, and thus the change vector that would make neuron i less prone to become active, we take the average $\widehat{\Omega}^{(\mathbf{x})} \in \mathcal{R}^d$ to be indicative of the pattern in the input that is related to the stability of the network's state induced by $\mathbf{x}$.

In the first evaluation, we examine visualisations of the 2CONV network (described in Appendix B.1) trained on the MNIST dataset (Lecun et al., 1998). Figure 4 shows visualisations from existing methods against the Insens heatmaps for different instances of the input and the same network. In the first instance we use clean MNIST input (on which test accuracy of 2CONV is over 99%). Note that Insens visualisation shows something that other methods hint at but do not show explicitly – that the network is sensitive to the black and white contrast of the digit and its outline. The red and blue areas in the Insens heatmaps suggest that respectively lightening and darkening these regions would make the current network state more stable (though in this case it is not possible to make black pixels darker nor white pixels any lighter). To verify the information provided by Insens, we added different types of noise to the images, also shown in Figure 4, and used them as input to the same network. We wanted to confirm that adding noise to the background, but not the digit outline, as suggested by Insens, does not impact performance. Indeed, testing with images that contain random gaussian noise in the background, but not within the 3-pixel outline of the digit (Figure 4 top-right) still gives 87% accuracy over 10000 test images. If we use images with random gaussian background everywhere (except the digit itself) or images with the same number of clean pixels as in the 3-pixel outline, but located around the frame of the image (Figure 4 bottom-left and bottom-right), the accuracy drops to 32% and 45% respectively. This confirms that Insens patterns provide meaningful information about the network's sensitivity. Also note that the other methods tend to lose the patterns of the digits in the visualisations over noisy images, whereas Insens still shows the digits, to which the network is still sensitive, albeit also being affected by the noise.

In the second evaluation we test Insens visualisations on four 2CONV networks trained on variants of the smallNORB dataset (LeCun et al.) (for details of the training see Appendix B.1). These variants differed in the encoding of the grayscale intensity of the pixels, where regular smallNORB (top-left of Figure 5) encoded black to white grayscale from 0 to 1, smallNORB neg (top-right of Figure 5) reversed the encoding from 1 to 0, smallNORB mean (bottom-left of Figure 5) encoded

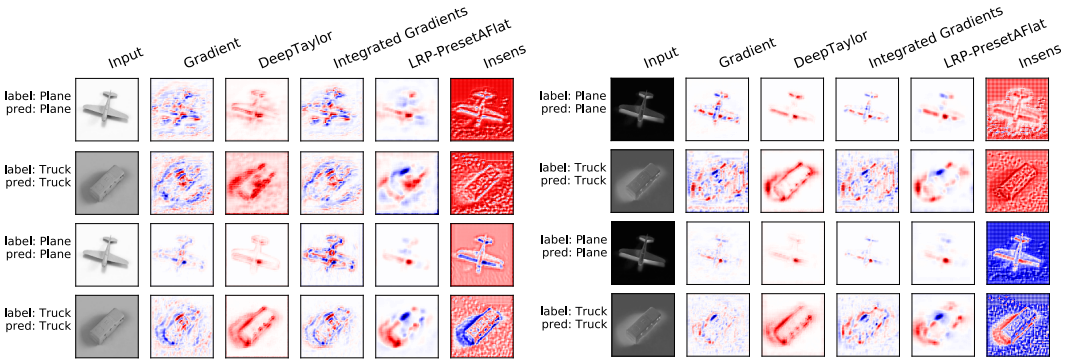

Figure 5: Visualisations for a set of interpretability methods of four 2CONV neural networks trained on four variants of encoding of the smallNORB images – each shown for two example images; top-left shows 2CONV trained on zero-black and one-white encoding of pixel intensity, top-right a network trained on zero-white and one-black encoding, bottom-left negative-one-black and positive-one-white encoding and bottom-right negative-one-white and positive-one-black encoding; for the last two encoding input images were normalised back to 0-1 range for the visualisation); the intensity of red and blue correspond respectively to the magnitude of the positive and negative values in the heatmaps.

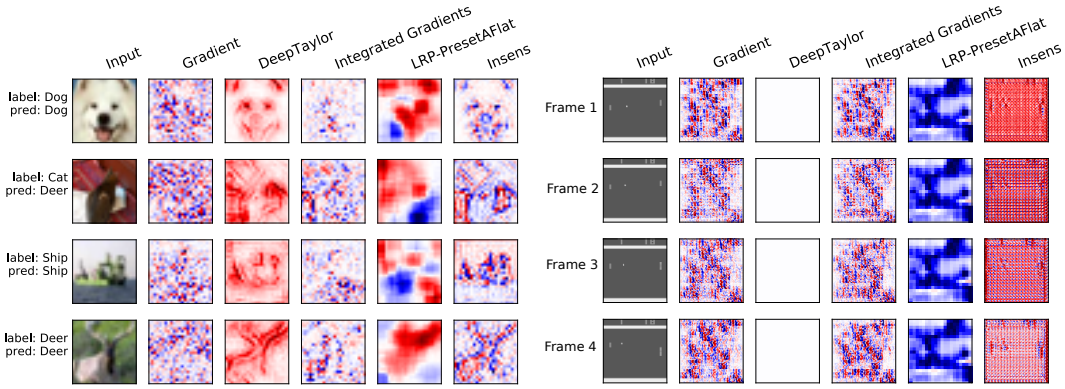

Figure 6: Visualisations for a set of interpretability methods for: (left) 4 CIFAR10 images and a 2CONV neural network trained on CIFAR10 dataset; (right) 4 consecutive frames corresponding to a single state and input of the RLNET neural network trained to play the Atari Pong game; the intensity of red and blue correspond respectively to the magnitude of the positive and negative values in the heatmaps.

black to white in range -1 to +1 and smallNORB neg mean (bottom-right of Figure 5) in reverse from +1 to -1. In all cases Insens shows what other methods miss – that the background plays an important role in the classification, and the nature of that role changes depending on the pre-processing. In the top-left of Figure 5, Insens shows that the regular smallNORB network seems to be tuned-in to the negative space of the scene, with the outline of the shape being a cut out of the background in the sensitivity visualisation. For smallNORB neg the network seems to process the positive space patterns. In the smallNORB mean and smallNORB neg mean networks, where the mean of the input range is centred around zero, the Insens heat maps are perfectly reversed between the two flipped variants of the encodings. Other methods tend to show the outlines of the shapes of interest, giving a sense that the networks always looks at the positive space and that background is ignored by the network, when in fact it is not.

For the third evaluation, we trained the 2CONV network on the CIFAR-10 dataset (Krizhevsky, 2009) (for details of the training see Appendix B.1). The resulting visualisations are shown on the left-side of Figure 6. Note that Insens, as also in this case the Deep Taylor decomposition, provides a sensible explanation for why the image of the cat is mislabelled by the network – the network

mistakes the composition of the pattern on the carpet and cat's ear for a set of antlers. For the image of the ship on the water, only Insens clearly shows the importance of the contrast between the water and sky, which is likely a bias that the network relies on for accurate ship identification.

The fourth evaluation looks at an RLNET, a neural network that was trained to play the Atari 2600 Pong game (for details of the architecture and training see Appendix B.2). Visualisation of a single state of the game given by four consecutive frames is shown on the right-hand side of Figure 6. While none of the visualisations are particularly clean, on close inspection, one can make out irregularities in the Insens heatmap that correspond to the location of the paddles and the ball. The fact that Insens visualisations, as well as other methods, are so noisy suggest that the background does play an important role in the computation that leads to decisions based on the policy given by that network. This is consistent with experiments that show that agents are unable to cope, and continue to play well, when the colour of the background changes after training (Hsu et al., 2018).

5 Conclusion

The switched linear projection is an interpretation of the computation inside a ReLU network that distinguishes between the active and inactive parts of the deep neural network architecture. The active subnetwork tends to be a smaller subset of the deep network (see Appendix B.3 for details) and the linear projection it provides is somewhat arbitrary, in the sense that it does not matter what the orientation of the switched weight vector is, as long as it produces the desired output. In addition, since it is the input that determines the particular pattern of neuron activity and inactivity, which we refer to as the network's state, relatively minor perturbations in the input can induce state changes affecting the switched projection and thus the nature of the active computation. Hence interpretability analysis based on the active subnetwork may be limited and not give the full picture of the patterns from the input that the network *relies* on in order to produce its computation.

Inactive state sensitivity (Insens), the proposed method for interpretability of ReLU networks, aggregates weighted sensitivity of the inactive neurons. This sensitivity relates the gradient of the input that would potentially drive the activity of the inactive neurons further away from zero, thus corresponding to the patterns in a particular input that would keep the network state stable and the output decision the same. As the name implies the method isolates the patterns in the input to which the network, in a sense, is *insensitive*. Our evaluations show that these patterns give a more comprehensive, and more explicit picture of what the network *reacts* to in a given input.

Since switched linear projections are just an interpretation of the actual computation inside a neural network, they may also become a useful tool for complexity analysis of deep networks. For instance, it might be possible to develop new regularisation methods based on switched weights, biases and centres of the neurons in the network. It remains to be investigated how the nature of the inactive subnetwork, and potential ways of manipulating it during training, would affect generalisation.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate neural networks! *arXiv reprint: arXiv 1808.04260*, 2018.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, 2009.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. *arXiv reprint: arXiv 1906.00904*, 2019.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Shu-Hsuan Hsu, I-Chao Shen, and Bing-Yu Chen. Transferring deep reinforcement learning with adversarial objective and augmentation. *arXiv reprint: arXiv 1809.00770*, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area v2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pp. 873–880.

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

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65(C):211–222, 2017.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 427–436.

M. Sadler and N. Regan. *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, 2019.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

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

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 818–833.

B. Zhou, D. Bau, A. Oliva, and A. Torralba. Interpreting deep visual representations via network dissection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2131–2145, 2019.

A PROOF OF THEOREM 1

Theorem 1 (Switched linear projections)**.** *Let $\mathbf{x} \in \mathcal{R}^d$ be a vector of inputs, $\mathbf{w}_{li} \in \mathcal{R}^{U_{l-1}}$ the weight vector, and $b_{li} \in \mathcal{R}$ the bias of neuron i in layer l (with U_{l-1} inputs from the previous layer). Let the activity of a neuron i in layer l be defined as:*

$$v_{li}(\mathbf{x}) = \Big(\dots \sigma_r\big(\sigma_r(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\big) \dots \Big)\mathbf{w}_{li} + b_{li}, \tag{1}$$

where $\mathbf{W}_l = \big[\mathbf{w}_{l1}^T \ \dots \ \mathbf{w}_{lU_l}^T\big]$, T denotes transpose, $\mathbf{b}_l = [b_{l1} \ \dots \ b_{lU_l}]$ and $\sigma_r(v) = \max(v, 0)$ is the ReLU activation function. If we define an input-dependent state of the network as
$\mathbf{W}_l^{(\mathbf{x})} = \big[\dot\sigma_r\big(v_{l1}(\mathbf{x})\big)\mathbf{w}_{l1}^T \quad \dots \quad \dot\sigma_r\big(v_{lU_l}(\mathbf{x})\big)\mathbf{w}_{lU_l}^T\big]$ and
$b_l^{(\mathbf{x})} = \big[\dot\sigma_r\big(v_{l1}(\mathbf{x})\big)b_{l1} \quad \dots \quad \dot\sigma_r\big(v_{lU_l}(\mathbf{x})\big)b_{lU_l}\big]$,
where $\dot\sigma_r(v) = \frac{d\sigma_r(v)}{dv}$, then for

$\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) = \mathbf{W}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T$ *and*

$\widehat{b}_{li}(\mathbf{x}) = \mathbf{b}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \mathbf{b}_2^{(\mathbf{x})}\mathbf{W}_3^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \dots + \mathbf{b}_{l-1}\mathbf{w}_{li}^T + b_{li}$*, we have*

$$v_{li}(\mathbf{x}) = \mathbf{x}\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) + \widehat{b}_{li}(\mathbf{x}). \tag{2}$$

Proof. By definition from Equation 1, the activity of neuron i in layer l is

$$v_{li}(\mathbf{x}) = \sum_{j=1}^{U_l} \sigma_r\Big(v_{l-1j}(\mathbf{x})\Big)w_{lij} + b_{li}, \tag{9}$$

where w_{lij} is the weight on the connection between neuron j of layer $l-1$ and neuron i of layer l, and b_{li} is the bias of neuron i in layer l.

Since $\sigma_r(v) = \begin{cases} v & v > 0, \\ 0 & \text{otherwise}, \end{cases}$ and $\dot\sigma_r(v) = \frac{d\sigma_r(v)}{dv} = \begin{cases} 1 & v > 0 \\ 0 & \text{otherwise}, \end{cases}$ we have

$$\sigma_r\Big(v_{l-1j}(\mathbf{x})\Big)w_{lij} = \begin{cases} v_{l-1j}(\mathbf{x})w_{lij} & v > 0, \\ 0 & \text{otherwise}, \end{cases}$$

and thus

$$\sigma_r\Big(v_{l-1k}(\mathbf{x})\Big)w_{lik} = v_{l-1k}(\mathbf{x})\dot\sigma_r\Big(v_{l-1k}(\mathbf{x})\Big)w_{lik}.$$

As a result

$$v_{li}(\mathbf{x}) = \sum_{j=1}^{U_l} v_{l-1j}\dot\sigma_r\Big(v_{l-1j}(\mathbf{x})\Big)w_{lij} + b_{li} = \sum_{j\in\mathcal{A}} v_{l-1j}w_{lij} + b_{li} \tag{10}$$

where $\mathcal{A}$ is the set of neurons with activity $v > 0$, the active neurons. Substituting the expression for activity from Equation 10 into its recursive definition in Equation 9, where $v_{0i}(\mathbf{x}) = x_i$, reveals that the overall computation is a series of linear transformations of $\mathbf{x}$ equivalent to a single linear transformation

$$v_{li}(\mathbf{x}) = \mathbf{x}\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) + \widehat{b}_{li}(\mathbf{x}).$$

where $\widehat{\mathbf{w}}_{li}^T(\mathbf{x}) = \mathbf{W}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T$ and

$\widehat{b}_{li}(\mathbf{x}) = \mathbf{b}_1^{(\mathbf{x})}\mathbf{W}_2^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \mathbf{b}_2^{(\mathbf{x})}\mathbf{W}_3^{(\mathbf{x})} \dots \mathbf{W}_{l-1}^{(\mathbf{x})}\mathbf{w}_{li}^T + \dots + \mathbf{b}_{l-1}\mathbf{w}_{li}^T + b_{li}$. □

Table 1: 2CONV network (conv = convolution; fc = fully connected)

Layer	Type	Filters/ neurons	Window	Stride	Activation
1	conv	32	5x5	1x1	relu
2	maxpool	-	2x2	2x2	relu
3	conv	64	5x5	1x1	relu
4	maxpool	-	2x2	2x2	relu
5	fc	512	-	-	relu
5	fc	K	-	-	-

Table 2: RLNET network (conv = convolution; fc = fully connected)

Layer	Type	Filters/ neurons	Window	Stride	Activation
1	conv	32	8x8	4x4	relu
2	conv	64	4x4	3x3	relu
3	conv	64	3x3	1x1	relu
4	fc	512	-	-	relu
5	fc	18	-	-	-

B NEURAL NETWORK ARCHITECTURES

The evaluation section of this paper tested the Insens interpretability on several network architectures and different datasets.

B.1 2CONV ARCHITECTURE

This network consists of two convolutional layers each followed by a maxpool layer that down-samples the feature map, followed by a single fully connected layer of 512 neurons and K output neurons, where K is the number of classes in the dataset of interest (see Table 1 for details). For training this network we used Adam optimiser minimising the softmax cross-entropy without regularisation. In all training a portion of the training set was set aside for validation for early stopping.

For the first, MNIST evaluation, the network was trained on cleaned MNIST data to test accuracy of 99.4%. The network was then tested on several variant of the MNIST dataset: MNIST Gauss images were given random Gaussian background (test accuracy of 31.7%), MNIST outline images with same Gaussian background except for outline within 3 pixels of the digit (test accuracy of 86.0%), and MNIST frame with Gaussian background and the same number of clean pixels as in the MNIST outline version of a given image around the frame (test accuracy of 44.9%).

For the second, smallNORB evaluation, the network was trained four times on variants of the dataset differing in the encoding of the images: smallNORB with grayscale intensity given by value between 0 and 1 for black to white pixel respectively, smallNORB neg with grayscale intensity between 0 and 1 for white to black pixel, smallNORB mean with grayscale between -1 and +1 for black to white pixel and smallNORB neg mean with grayscale between -1 and 1 for white to black pixel. The classifciation accuracy on these four networks tested on 24300 image of each dataset was respectively 84.3%, 87.2%, 81.4% and 83.6%.

For the third, CIFAR-10 evaluation, a single network was trained on augmented (by random contrast adjustment and crop to a 24x24 size) images of the CIFAR-10 dataset giving test accuracy of 81.0%.

B.2 RLNET ARCHITECTURE

For the fourth evaluation, a single convolutional neural network was trained to play Atari 2600 Pong with a reinforcement learning algorithm. The RLNET is the policy network taking four 84x84 grayscale frames of the game as input, producing 18 outputs or actions corresponding to different combinations of joystick movements and pressing of the fire button (see Table 2 for details of the

Table 3: Average percentage of active neurons in a neural network

Dataset	Network	Total # neurons	Average % active neurons
MNIST	2CONV	38144	17%
MNIST gauss	2CONV	38144	25%
MNIST outline	2CONV	38144	27%
MNIST border	2CONV	38144	23%
smallNORB	2CONV	442880	20%
smallNORB neg	2CONV	442880	15%
smallNORB mean	2CONV	442880	27%
smallNORB mean neg	2CONV	442880	17%
CIFAR-10	2CONV	28160	19%
Pong	RLNET	20896	32%

architecture). Visualisations are shown for the network that was trained to obtain an average score of 17.3, where 17 corresponds to beating the computer driven opponent 21 to 4.

B.3 NUMBER OF ACTIVE NUEURONS

Table 3 shows the average percentage of neurons that are active during input-output mappings for the trained networks evaluated in this paper.