Decision Explanation and Feature Importance for Invertible Networks

Juntang Zhuang[1], Nicha C. Dvornek [2], Xiaoxiao Li[1], Junlin Yang[1], James S. Duncan[1,2,3]
[1] Biomedical Engineering, Yale University, New Haven, CT USA
[2] Radiology & Biomedical Imaging, Yale School of Medicine, New Haven, CT USA
[3] Electrical Engineering, Yale University, New Haven, CT USA
{j.zhuang; nicha.dvornek; xiaoxiao.li; junlin.yang;james.duncan}@yale.edu

Abstract

Deep neural networks are vulnerable to adversarial attacks and hard to interpret because of their black-box nature. The recently proposed invertible network is able to accurately reconstruct the inputs to a layer from its outputs, thus has the potential to unravel the black-box model. An invertible network classifier can be viewed as a two-stage model: (1) invertible transformation from input space to the feature space; (2) a linear classifier in the feature space. We can determine the decision boundary of a linear classifier in the feature space; since the transform is invertible, we can invert the decision boundary from the feature space to the input space. Furthermore, we propose to determine the projection of a data point onto the decision boundary, and define explanation as the difference between data and its projection. Finally, we propose to locally approximate a neural network with its first-order Taylor expansion, and define feature importance using a local linear model. We provide the implementation of our method: https://github.com/juntang-zhuang/explain_invertible.

1. Introduction

Deep learning models have achieved state-of-the-art performance in multiple practical problems, including image classification [11, 16], video processing [1] and natural language processing [4]. However, the black-box nature of the design of most deep learning architectures [3] has raised issues such as lack of interpretation and being vulnerable to adversarial attacks [18]. Previous works have found difficulties in recovering images from hidden representations in the neural network [15, 5], and it is often unclear what information is being discarded [19].

Various methods have been proposed to interpret neural networks. The mainstream is to calculate gradient of the loss function $w.r.t.$ the input image [20, 14]. Dosovitskiy et al. proposed up-convolution networks to invert CNN feature maps back to images [5]. Another direction for model interpretation is to determine the receptive field of a neuron [21] or extract image regions that contribute the most to the neural network decision [23, 8, 10]. Other works focus on model-agnostic interpretations [17, 2, 13, 12]. Different from previous works, we consider explainable neural network models.

The recently proposed invertible network [6, 7, 22] is able to accurately reconstruct the inputs to a layer from its outputs without harming its classification accuracy. For an invertible classifier, information is only discarded at the final pooling layer and fully-connected layer, while preceding layers preserve all information of the input. This property hints at the potential to unravel the black-box and manipulate data both in the input domain and the feature domain.

In this paper, we introduce a novel method to explain the decision of a network. We show that an invertible classifier can be viewed as a two-stage model: (1) an invertible transform from the input space to the feature space; (2) a linear classifier in the feature space. For a linear classifier, we can determine the decision boundary and explain its prediction; using the invertible transform, we can determine the corresponding boundary and explanation in the input space.

After determining the projection onto the decision boundary, we perform Taylor expansion around the projection, to locally approximate the neural net as a linear function. Then we define the importance using the same method as in linear classifier cases.

Our main contributions can be summarized as:

- We explicitly determine the decision boundary of a neural network classifier and explain its decision based on the boundary.
- We use Taylor expansion to locally approximate the neural net as a linear function and define the numerical importance of each feature as in a linear classifier.

2. Invertible Networks

The network is composed of different invertible modules, followed by a global average pooling layer and a fully-connected layer. Details for each invertible module are described in the following sections.

2.1. Invertible Block

An invertible block serves a similar role as a building block like a residual block, except it is invertible. For the invertible block in Fig. 1, we follow the structure of the reversible block in [6]. The input x is split into two parts x_1 and x_2 by channel, such that x_1 and x_2 have the same shape.

Corresponding outputs are y_1 and y_2 with the same shape as the input. F represents some function with parameters to learn, and F can be any continuous function whose output has the **same** shape as input; an example of F is shown in Fig. 2. F can be convolutional layers for 2D inputs and FC layers for 1D inputs. The forward pass and inversion is calculated as:

$$\begin{cases} y_1 = x_2 + F(x_1) \\ y_2 = x_1 \end{cases} \quad \begin{cases} x_1 = y_2 \\ x_2 = y_1 - F(x_1) \end{cases} \tag{1}$$

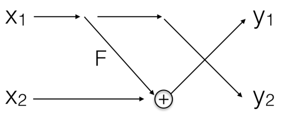


Figure 1: Structure of the invertible residual block.

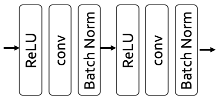


Figure 2: An example of F in the invertible block.

2.2. Invertible Pooling with 2D Wavelet Transform

An invertible pooling can halve the spatial size of a feature map, and reconstruct the input from its output. We use the 2D wavelet transform at level 1 as shown in Fig. 3. Each channel of a tensor is a 2D image. A 2D image is transformed into 4 sub-images whose height/width is half of the original image. Four sub-images are stacked into 4 channels. The inversion can be calculated by the inverse 2D wavelet transform.

2.3. Inverse of Batch Normalization

The forward pass and inverse of a batch normalization layer are listed below:

$$y = \frac{x - \mathbf{E}(x)}{\sqrt{\mathbf{Var}(x) + \epsilon}}\gamma + \beta, \quad x = \frac{y - \beta}{\gamma}\sqrt{\mathbf{Var(x)} + \epsilon} + \mathbf{E}(x) \tag{2}$$

where γ and β are parameters to learn, $\mathbf{E}(x)$ and $\mathbf{Var}(x)$ are approximated as the sample-mean and sample-variance respectively, and all operations are channel-wise.

2.4. Linear Layer

The feature space usually has a high dimension compared to the number of classes for final prediction. The mapping from high-dimension to low-dimension is typically performed with an average pooling and a fully-connected (FC) layer in a convnet. These two steps combined is still a linear transform and can be denoted as:

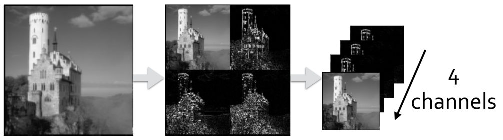


Figure 3: 2D wavelet transform as an invertible pooling.

$$y = ABz = Wz, where\ \ W = AB \tag{3}$$

where z is a $C \times H \times W$ feature vector reshaped to 1D, H, W are spatial sizes, and C is the channel number; B is a block-wise constant matrix of size $C \times CHW$, representing the average pooling operation; A is the weight of a FC layer with size $K \times C$, where K is the number of classes; and $W = AB$ combines the two steps into 1 transform matrix.

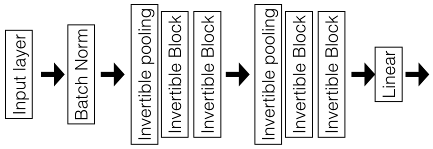


Figure 4: Structure of an invertible network.

2.5. Structure of Invertible Network

The structure of a classification network is shown in Fig. 4. The network is invertible because its modules are invertible. The input image is fed into a batch normalization layer followed by an invertible pooling layer. The invertible pooling layer increases the channel number by 4 and is essential to make the tensor have an even number of channels in order to keep the same shape for x_1 and x_2 as in formula 1.

The network is divided into stages, where an invertible pooling layer connects two adjacent stages. Within each stage, multiple invertible blocks are stacked. The output from the final stage is fed into a linear layer defined in Sec. 2.4. The probability of current data belonging to a certain class is calculated as a $softmax$ of the logits.

2.6. Reconstruction Accuracy of Inversion

We build an invertible network of 110 layers. We train the network on the CIFAR10 dataset [9], and reconstruct the input image from the output of the final invertible block. Results are shown in Fig. 5. The l_2 distance between reconstruction and input is on the order of 10^{-6}, validating the accuracy of inversion.

3. Interpret Model Decision

3.1. Notations of Network

The invertible network classifier can be viewed as a two-stage model:

$$t = T(x), \;\; y = Class(t) \tag{4}$$

(1) The data is transformed from the input space to the feature space by an invertible function T. (2) Features pass through a linear classifier $Class$, whose parameters W and b are defined in Sec. 2.4.

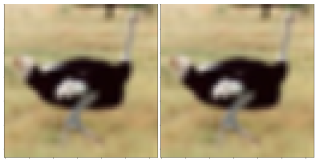


Figure 5: From left to right: input image, reconstructed image from outputs of last invertible block.

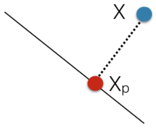


Figure 6: For a linear classifier, X_p is the projection of X onto the decision plane, and the vector (X_p, X) is the explanation for decision.

The operation of $y = Class(t)$ is defined as:

$$\begin{cases} y_k = \langle \vec{t}, \vec{w_k} \rangle + b_k, \quad k = 1, 2, ...K \\ \mathbf{P}(k|x) = \frac{exp(y_k)}{\sum_{i=1}^{K} exp(y_i)} \end{cases} \tag{5}$$

where $\vec{w_k}$ is the weight vector for class k, also is the kth row of W in Sec. 2.4; b_k is the bias for class k; $\langle \cdot, \cdot \rangle$ is the inner-product operation and K is the total number of classes.

3.2. Determine the Decision Boundary

Note that on the decision boundary probabilities of two classes are the same. Using the same notation as in formula (4) and (5), the decision boundary between class i and j in the feature domain is:

$$\langle \vec{t}, \vec{w_i} \rangle + b_i = \langle \vec{t}, \vec{w_j} \rangle + b_j \tag{6}$$

The solution $\vec{t}$ to formula (6) lies on a high-dimensional plane, and can be solved explicitly.

Since T is invertible, we can map the decision boundary from the feature space to the input domain.

3.3. Model Decision Interpretation

3.3.1 Interpret linear models

We first consider a linear classifier for a binary problem as in Fig. 6. For a data point X and its projection X_p onto the decision boundary, X_p is the *nearest* point to X on the boundary; the vector (X_p, X) could be regarded as the explanation for the decision, as shown below:

$$Explanation = X - X_p \tag{7}$$

3.3.2 Interpret non-linear model

The last layer of a neural network classifier is a linear classifier, and we can calculate X_p from X as in linear case. With invertible networks, we can find their corresponding inputs, denoted as $T^{-1}(X)$ and $T^{-1}(X_p)$ respectively, where T is the transform function as in equation (4). Vector $(T^{-1}(X), T^{-1}(X_p))$ is the explanation in the input domain. The explanation can be denoted as:

$$Explanation = T^{-1}(X) - T^{-1}(X_p) \tag{8}$$

where $X = T(x)$ is the point in the feature space, corresponding to x in the input space; X_p is the projection of X onto the boundary in the feature space; and x_p is the inversion of X_p, as shown in Fig. 7.

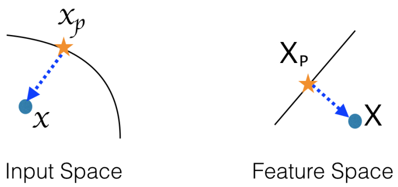


Figure 7: Explanation of invertible non-linear classifiers. Left figure is the input space, right figure is the feature space. Black line is the decision boundary, X_p is the projection of X onto the decision boundary. Vector (X, X_p) is perpendicular to the boundary in feature space. Dashed vector can be viewed as the explanation for model decision.

3.3.3 Feature importance

Linear case For a linear model, ignoring the bias, the function for log-probability is:

$$f(x) = \sum_{i}^{d} w_i(x_i - x_{p,i}) \tag{9}$$

where d is the dimension of x; x_p is the projection of x onto the boundary, which is also the nearest point on the boundary; and w_i is the weight for dimension i.

The explanation in dimension i is $x_i - x_{p,i}$; the contribution to $f(x)$ is $w_i(x_i - x_{p,i})$. Therefore, we define $|w_i(x_i - x_{p,i})|$ as the importance of feature i for data x.

Non-linear case We use Taylor expansion around x_p to approximate the neural network with a linear model locally:

$$f(x) = f(x_p) + \nabla f(x_p)^T (x - x_p) + O(||x - x_p||_2^2) \tag{10}$$

For a local linear classifier, the importance of each feature is:

$$Importance = |\nabla f(x_p) \otimes (x - x_p)| \tag{11}$$

where $\otimes$ is the element-wise product, and $Importance$ is a vector with the same number of elements as x.

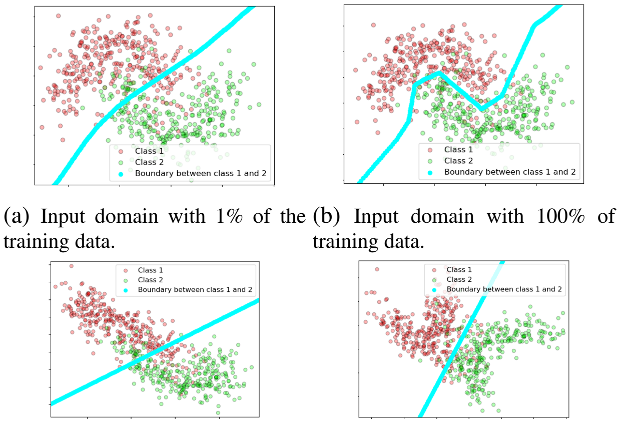
4. Experiments

4.1. Decision Boundary Visualization

For a d-dimensional input space, the decision boundary is a $(d-1)$-dimensional subspace. For the ease of visualization, we perform experiments on a 2D simulation dataset, whose decision boundary is a 1D curve.

The data points for two classes are distributed around two interleaving half circles. As shown in Fig. 8, two classes are colored with red and green. The decision boundary is colored in blue. We visualize the decision boundary in both the input domain and the feature domain.

Visualization of the decision boundary can be used to understand the behavior of a neural network. We give an example to visualize the influence of training set size on the decision boundary in Fig. 8. From left to right, the figure shows the decision boundary when training with 1% and 100% of data, respectively. As the number of training examples increases, the margin of separation in the feature domain increases, and the decision boundary in the input domain gradually captures the moon-shaped distribution. Furthermore, the decision boundary can be used to determine how the network generalizes to unseen data.



(a) Input domain with 1% of the training data.

(b) Input domain with 100% of training data.

(c) Feature domain with 1% of training data.

(d) Feature domain with 100% training data.

Figure 8: Visualization of the decision boundary varying with the size of training set on a 2D toy dataset. Top row shows results in input domain, and bottom row shows results in feature domain. Columns left (right) shows training with 1% (100%) of data.

4.2. Feature Importance

We validated our proposed feature importance method on a simulation dataset using $make\_classification$ in $scikit-learn$. We created a 2 class, 10-dimensional dataset, of which only 3 dimensions are informative.

We train an invertible network and computed the importance of each dimension. Results are shown in Fig. 9. An oracle model should give equal importance to 3 informative variables (indexed by 1, 3 and 9), while set 0 to other variables. Our invertible network successfully picks informative features, and generates feature importance comparable to random forest. Both models select the correct features.

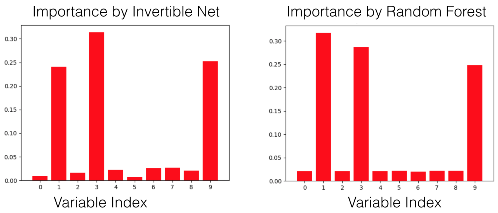


Figure 9: Left: feature importance from invertible network. Right: feature importance from random forest.

4.3. Explain a Convolutional Invertible Network

We train a convolutional invertible classifier, achieving over 99% accuracy on the MNIST test set. For an input image, we select classes with the top 2 predicted probabilities, determine the decision boundary between these two classes as in Sec. 3.2, calculate the projection onto the boundary, and interpolate between the input image and its projection onto the boundary in the feature domain.

Results are shown in Fig. 10. Note that for each row, only one input image (left most) is provided; the projection (right most) is calculated from the model, instead of searching for a nearest example in the dataset. So the projection demonstrates the behavior of the network. As discussed in Sec. 3.3, the difference between a data point and its projection onto the boundary can be viewed as the explanation. For example, for an image of 8, its left half vanishes in the interpolation, which explains why it's not classified as 3; for an image of 7, a bottom line appears in the interpolation, which explains why it's not classified as 2.

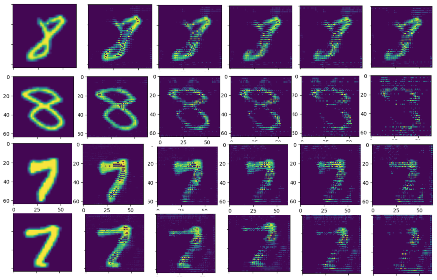


Figure 10: Interpolation (performed in the feature domain) between input (left most) and its projection on the boundary (right most). Top two rows shows 8 transforms to 3, bottom two rows show 7 to 2.

5. Conclusion

We propose a method to explicitly determine the decision boundary of an invertible neural network classifier and define the explanation for model decision and feature importance. We validate our results in experiments, and demonstrate that the transparency of invertible networks has great potential for explainable models.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011. 1

[2] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. 1

[3] D. Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016. 1

[4] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 1

[5] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016. 1

[6] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2214–2224, 2017. 1, 2

[7] J.-H. Jacobsen, A. Smeulders, and E. Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018. 1

[8] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017. 1

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2

[10] D. Kumar, A. Wong, and G. W. Taylor. Explaining the unexplained: A class-enhanced attentive response (clear) approach to understanding deep neural networks. In *IEEE Computer Vision and Pattern Recognition (CVPR) Workshop*, 2017. 1

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1

[12] X. Li, N. C. Dvornek, Y. Zhou, J. Zhuang, P. Ventola, and J. S. Duncan. Efficient interpretation of deep learning models using graph structure and cooperative game theory: Application to asd biomarker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 718–730. Springer, 2019. 1

[13] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017. 1

[14] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 1

[15] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016. 1

[16] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 1

[17] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017. 1

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[19] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015. 1

[20] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1

[21] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018. 1

[22] J. Zhuang, N. C. Dvornek, X. Li, P. Ventola, and J. S. Duncan. Invertible network for classification and biomarker selection for asd. *arXiv preprint arXiv:1907.09729*, 2019. 1

[23] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017. 1