ON SPARSE CRITICAL PATHS OF NEURAL RESPONSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Is critical input information encoded in specific sparse paths within the network? The pruning objective — finding a subset of neurons for which the response remains unchanged — has been used to discover such paths. However, we show that paths obtained from this objective do not necessarily encode the input features and also encompass (dead) neurons that were not originally contributing to the response. We investigate selecting paths based on neurons' contributions to the response to ensure that the paths envelop the critical segments of the encoded input information. We show that these paths have the property of being provably locally linear in an ℓ_2-ball of the input, thus having stable gradients. This property is leveraged for proposing a feature attribution paradigm that is guided by neurons, therefore inherently taking interactions between input features into account. We evaluate the attribution methodology quantitatively in mainstream benchmarks.

1 INTRODUCTION

Deep rectified neural networks encode the input information using a sparse set of active neurons (Glorot et al., 2011), and their inference can be deemed as a pursuit algorithm for sparse coding (Papyan et al., 2017; Sulam et al., 2018). Recently, Wang et al. (2018) proposed using the pruning objective and knowledge distillation (Hinton et al., 2006) to show that *significantly higher levels of sparsity* (∼87% for VGG-16 (Simonyan & Zisserman, 2014) on ImageNet (Deng et al., 2009)) can be achieved while keeping the prediction intact. These highly sparse paths are currently seen as the critical paths and are shown to be different for inputs of different classes and adversarially manipulated inputs (Qiu et al., 2019; Wang et al., 2018; Yu et al., 2018). Here, we investigate, whether these highly sparse paths derived from the pruning objective indeed encode critical information of the input features. Crucially, we can show that the *pruning objective has solutions that are not critical paths*, even though they have the same response as the original network. To illustrate how the pruning objective can result in such paths, we construct a greedy pruning algorithm that by design attempts to find a sparse path that encompasses originally dead neurons, while at the same time satisfying the pruning objective. Furthermore, we analyze the paths selected by distillation guided routing (Wang et al., 2018) and observe a similar phenomenon.

If these paths do not encode critical input features, do sparse paths that encode those features exist? How can we find such paths? Thus far we have uncovered the undesirable property of the pruning objective that results in paths consisting of neurons which were originally not contributing. On the other hand, numerous works have studied the importance of individual neurons for the neural response, and how each neuron encodes information specific to one or a subset of classes (Zhou et al., 2018; Bau et al., 2017; Morcos et al., 2018; Olah et al., 2017). It is therefore intuitive that selected sparse paths should encompass *important* neurons for the corresponding response. Importance, or *contribution*, is well defined in feature attribution literature. Many works (Fong & Vedaldi, 2017; Fong et al., 2019; Zintgraf et al., 2019; Ribeiro et al., 2016) implicitly adopt the notion of "marginal contribution", which is the effect of ablation of an input feature on the score function, and other works (Lundberg & Lee, 2017; Ancona et al., 2019; Sundararajan et al., 2017; Sundararajan & Najmi, 2020) adopt a more proper axiomatic definition: the "Shapley value" (Shapley, 1953). We use both aforementioned notions to compute neuron contributions and investigate selecting paths based on neuron contributions instead of the pruning objective. The first section of the work is devoted to analysis and discussion of path selection methods, where we also use neural decoding (Olah et al., 2017; Mahendran & Vedaldi, 2015) to semantically analyze the paths.

The selected paths can help to interpret the response of the network. We show that in rectified neural networks, sparse paths selected by neuron contributions have the property of being provably locally linear. Therefore, the corresponding sparse networks have stable gradients, a property that is exploited for input feature attribution, i.e. we can use the gradients of these paths to uncover their corresponding features in the input. We can select various levels of sparsity, thus we are introducing a feature attribution paradigm where high-level learned features are revealed in the input space in a continuous fashion, based on their criticality for the neural response. The attribution approach has the special property that it considers the distributional interdependence and the correlations between input features. This is possible as the approach in the first step computes the contributions of neurons, which have been trained to encode the complex interactions and correlations between input features. And in the second step, the path gradient reflects the interactions between the coalition of neurons within the path. Therefore, we observe that using a method such as integrated gradients (Sundararajan et al., 2017) on the neurons within our proposed framework performs better in revealing important features of the input compared to conventionally applying integrated gradients directly on the input. The efficacy of the proposed attribution framework is evaluated by several mainstream metrics in the attribution literature (Adebayo et al., 2018; Hooker et al., 2019; Samek et al., 2017).

2 Selection of sparse critical paths

Section 2.1 defines our notation, which is followed by the intoduction of the pruning objective in Section 2.2. By constructing a greedy algorithm we show that pruning can result in undesired paths, and then proceed to analyzing the distillation guided routing of Wang et al. (2018). Subsequently (Section 2.3) we investigate selecting paths based on neuron contributions and perform comparative path analysis with the pruning based methods. Ultimately, we conduct a semantic analysis of the paths selected by all methods in Section 2.4 through neural decoding.

2.1 Setup and notation

Consider a neural network $\Phi_\Theta(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$ with ReLU activation functions, parameters $\Theta = \{\theta^1, ..., \theta^L\}$, and L hidden layers with N_i neurons in layer $i \in \{1, ..., L\}$. The total number of neurons is $N = \sum_{i=1}^{L} N_i$. We use $\mathbf{z}^i \in \mathbb{R}^{N_i}$ to represent pre-activation vector at layer i, and $\mathbf{a}^i \in \mathbb{R}^{N_i}$ for representing the corresponding activation vector, where $\mathbf{a}^i = \text{ReLU}(\mathbf{z}^i)$, $\mathbf{z}^i = \theta^i \mathbf{a}^{i-1} + \mathbf{b}^i$, and $\mathbf{a}^0 = \mathbf{x}$. Note our definition of $\Phi_\theta(\mathbf{x})$ has a single real valued output. The reason is that we are considering the neural path to one neuron, and this neuron could in fact be a hidden neuron in a larger network. Thus the response is defined by $\Phi_\theta(\mathbf{x}) = \theta^{L+1}\mathbf{a}^L + \mathbf{b}^{L+1}$. Each individual neuron in layer i is specified by index $j \in \{1, ..., N_i\}$, thus denoted by $\mathbf{z}^i_j$ and $\mathbf{a}^i_j$. The vectors $\mathbf{z}^i$ and $\mathbf{a}^i$ are specifically associated with input $\mathbf{x}$. Proofs are provided in Appendix A.

2.2 Selection by pruning objective

In this section we describe the commonly used pruning objective and discuss how *a solution satisfying this objective does not necessitate it being critical*. The idea of pruning is to remove unnecessary neurons from the network. Let $\mathbf{m} = \{0, 1\}^N$ be a mask that represents the neurons to be kept and pruned. The pruning objective given an input $\mathbf{x}$ is then defined as:

$$\arg\min_{\mathbf{m}} \mathcal{L}\left(\Phi_\theta(\mathbf{x}), \Phi_\theta(\mathbf{x}; \mathbf{m} \odot \mathbf{a})\right) \quad \text{s.t. } \|\mathbf{m}\|_0 \leq \kappa, \tag{1}$$

where $\odot$ and $\mathcal{L}$ denote the Hadamard product and the loss, respectively. $\mathbf{a}$ denotes neurons in all layers and κ controls the sparsity. Equation (1) is an enormous combinatorial optimization problem and a plethora of solutions have been proposed in the literature (LeCun et al., 1990; Hassibi & Stork, 1993; Han et al., 2015; Molchanov et al., 2016; Lee et al., 2018).

The question is whether such an objective results in paths that encode the input. We already know that a rectified network is encoding the input in a sparse set of *active* neurons. Dead neurons are irrelevant, and their removal does not alter the encoded representation.

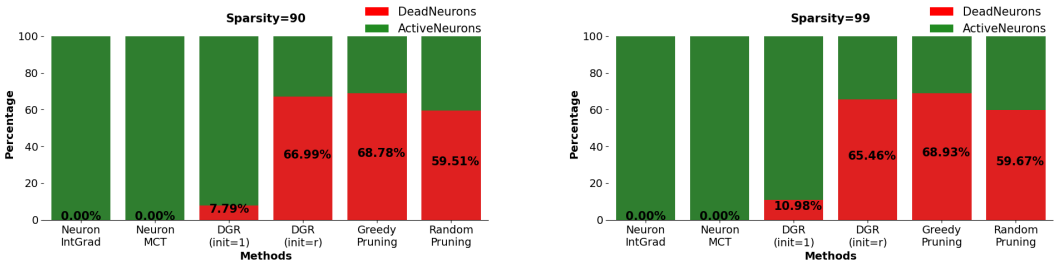

Figure 1: **Dead Neuron Prevalence.** The percentage of originally dead neurons in the selected paths of different methods reported for sparsity of 90% (left) and 99% (right). All paths selected by pruning objective contain originally dead (now active) neurons.

Lemma 1 (Dead Neurons) *Considering* $\mathbf{a}^i$ *as the input at layer* i *to the following layers of the network defined by function* $\Phi_\theta^{>i}(.) : \mathbb{R}^{N_i} \rightarrow \mathbb{R}$*, the Shapley value of a neuron* $\mathbf{a}_j^i$ *defined by* $\sum_{C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i} \frac{|C|!(N_i-|C|-1)!}{N_i} (\Phi_\theta^{>i}(C \cup \mathbf{a}_j^i) - \Phi_\theta^{>i}(C))$ *is zero if the neuron is dead* ($\mathbf{a}_j^i = 0$).

Lemma 1 (proof in Appendix A) shows that dead neurons have a Shapley value of zero, thus do not contribute to the response. Note that removing an active neuron results in a change in inputs to next layer's neurons and thus *may* change their activation value, and this *can* result in activating an originally dead neuron. We consider a sparse selected path *undesirable* if it contains neurons that were originally dead, but have become active due to the pruning of other neurons. Such a selected path is an artificial construct, which does not reflect the original sparse encoding of the input.

Pathological greedy pruning. We construct a greedy algorithm to solve for the pruning objective (Eq. (1)) to illustrate how a solution can select paths where originally dead neurons turn active, and become part of the highly sparse selected path that produces the same network response. Our greedy approach (Section 2.2) first ranks all neurons based on their relevance score for the response. The relevance is determined by the effect of removing a neuron, effectively approximated by Taylor expansion similiar to (Molchanov et al., 2016; Mozer & Smolensky, 1989). The relevance score s_j^i is then:

Algorithm 1 Greedy pruning

initialize $\mathbf{m}_j^i = 1 \quad \forall\, i, j$

while $\|\mathbf{m}\|_0 \geq \kappa$ **do**

  $s_j^i \leftarrow |\mathbf{a}_j^i \nabla_{\mathbf{a}_j^i} \Phi_\theta(\mathbf{x})|$

  if $s_j^i \leq s_\kappa \wedge s_j^i \neq 0$ **then** $\mathbf{m}_j^i \leftarrow 0$;

$$s_j^i = |\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}_j^i := 0)| = |\mathbf{a}_j^i \nabla_{\mathbf{a}_j^i} \Phi_\theta(\mathbf{x})| \; . \tag{2}$$

The next steps are removing the neuron(s) with the lowest rank, and alternating between rank computation and removal on the new subsets. However, we tweak the algorithm to find paths that contain originally dead neurons. At each removal step, we remove the lowest contributing neuron that is not dead. Without this step, dead neurons will be pruned before others as their relevance score is zero.

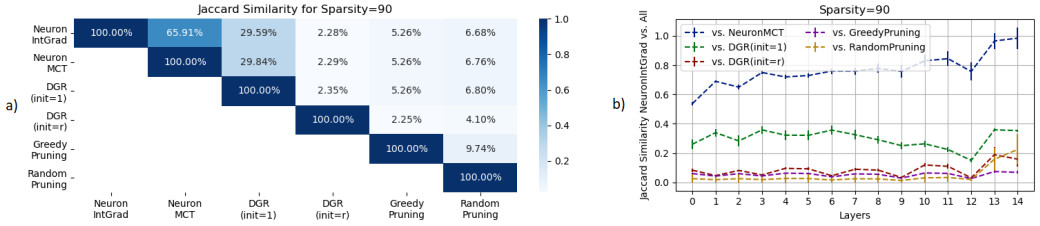

Figure 2: **Path Analysis.** Overlap between paths from different methods: a) overlap in entire network b) layer-wise overlap between paths of all methods and NeuronIntGrad. Among the pruning-based methods, only the path selected by DGR(init=1) overlaps with contribution-based methods.

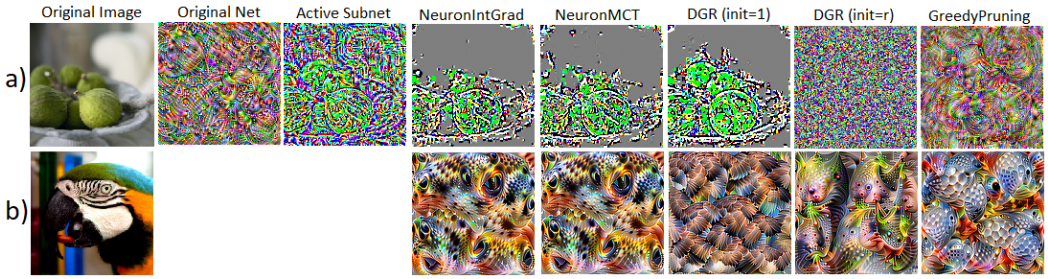

Figure 3: **Path Decoding.** a) Maximizing the network response while restricting the network to a specific path. b) Feature visualization of the top selected neuron in each path. Results confirm that the paths selected by the pruning objective (except for DGR(init=1)) do not encode image features.

By removing a non-zero neuron, the activation pattern can change and some originally dead neurons can activate and become included in the path. This constructed algrithm illustrates that solving for the pruning objective can result in undesirable paths.

Distillation Guided Routing: DGR (Wang et al., 2018) relaxes the pruning objective (Eq. (1)) by replacing $\mathbf{m} = \{0, 1\}^N$ with continuous valued gates $0 \leq \lambda \in \mathbb{R}$. To induce sparsity, the objective is regularized with an L1 norm, *i.e.* $\|\lambda_k\|_1$, where k denotes the index of the corresponding neuron:

$$\min_{\Lambda} \mathcal{L}(\Phi_\theta(\mathbf{x}), \Phi_\theta(\mathbf{x}; \Lambda \odot \mathbf{a})) + \gamma \sum_{k=1}^{N} \|\lambda_k\|_1 \quad \text{s.t. } \lambda_k \geq 0\,, \tag{3}$$

where Λ denotes $[\lambda_k]^N$. After the optimization, the resulting gates are binarized. In our experiments we find that the initial value of Λ plays a significant role in selecting paths. Wang et al. (2018) use $\Lambda_{init} = [1]^N$ without discussing its role. We denote different initilizations with DGR(init=*value*).

2.3 Selection by neuron contribution

The role of individual neurons in the response of the network for specific classes is studied from several perspectives, Bau et al. (2017); Olah et al. (2017) show that single directions are indeed interpretable and mostly correspond to meaningful concepts. Moreover, the importance of individual neurons for different output classes is studied in Zhou et al. (2018), where individual neuron ablation is conducted. The effect of removing a unit, $|\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}^i_j := 0)|$, is called the marginal contribution and satisfies symmetry and null-player axioms. It is a desirable property for a path that is supposed to encode the critical features in the input to contain critical neurons. Therefore we propose selecting paths of critical neurons as the critical path of the network response given an input. Computing the exact value for the marginal contribution of all neurons is computationally expensive. Marginal contribution requires response computation N times. Therefore we use a Taylor approximation similar to Eq. (2), $\mathbf{c}^i_j = |\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}^i_j := 0)| = |\mathbf{a}^i_j \nabla_{\mathbf{a}^i_j} \Phi_\theta(\mathbf{x})|$, where $\mathbf{c}^i_j$ denotes the contribution of neuron $\mathbf{a}^i_j$. Paths selected by this method are hereon referred to as NeuronMCT, where MCT stands for Marginal Contribution Taylor.

The more proper definition for the notion of importance is the Shapley value. The Shapley value is the unique definition that satisfies completeness, linearity symmetry, null player axioms. Computing the exact Shapley value requires $\sum_{i=1}^{L} 2^{N_i - 1}$ inference steps. The integrated gradients method with baseline 0 is equivalent to the Aumann-Shapley value, which is an extension of the Shapley value to continuous setting, such as neural networks (Sundararajan & Najmi, 2020). The contribution $\mathbf{c}^i_j$ using integrated gradients with baseline 0 is:

$$\mathbf{c}^i_j = \mathbf{a}^i_j \int_{\alpha=0}^{1} \frac{\partial \Phi_\theta(\alpha \mathbf{a}^i_j; \mathbf{x})}{\partial \mathbf{a}^i_j} \mathrm{d}\alpha \tag{4}$$

Henceforth, the contributions assigned as such are referred to as NeuronIntGrad.

Remark 2 *NeuronMCT and NeuronIntGrad assign* $\mathbf{c}^i_j = 0$ *to a neuron with* $\mathbf{a}^i_j = 0$ *(dead neuron).*

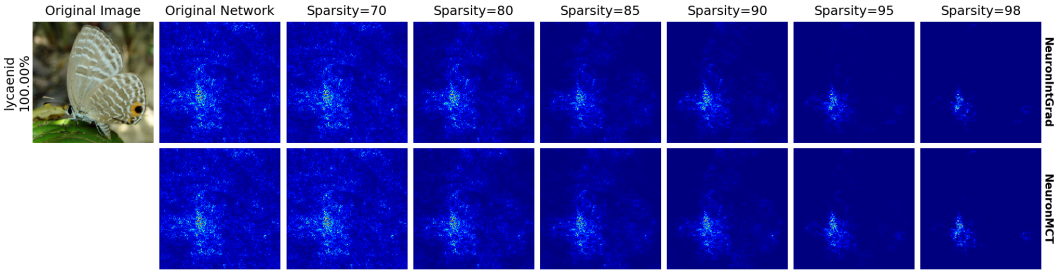

Figure 4: **Gradient Visualization.** The gradients of the locally linear critical paths at different sparsity levels for NeuronIntGrad (top) and NeuronMCT (bottom). More examples in Appendix D.

Having computed the contributions $\mathbf{c}_j^i$, in order to select a path $\mathbf{e} = [\mathbf{e}_j^i]^N$, with sparsity value κ, we select neurons with $\mathbf{c}_j^i \geq \mathbf{c}_\kappa$ where $\mathbf{c}_\kappa$ is the contribution value of the corresponding sparsity κ in a sorted list of contributions, *i.e.* if $\mathbf{c}_j^i \geq \mathbf{c}_\kappa$ then $\mathbf{e}_j^i = 1$, else $\mathbf{e}_j^i = 0$. Selecting the values higher than $\mathbf{c}_\kappa$ is average-case $O(n)$. Thus the computational burden of selecting a path depends on the contribution assignment procedure. NeuronMCT requires one inference, and for NeuronIntGrad we use 50 inference steps in the experiments. For both methods, the ranking of the $\mathbf{a}_j^i$ are performed network-wise, and not layer-wise, *i.e.* we directly compare the contribution of neurons from different layers. This is possible because integrated gradients satisfies completeness, and for each layer i, $\sum_{j=1}^{N_i} \mathbf{c}_j^i = \Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{a}^i := 0)$, making the scores directly comparable.

2.4 Path selection experiments

Path analysis: To corroborate the claim that the pruning objective results in undesirable paths, we evaluate the paths extracted by the discussed methods from a VGG-16 (Simonyan & Zisserman, 2014) network for 1k ImageNet (Deng et al., 2009) images. We report results for the paths of 90% and 99% sparsity. Fig. 1 shows the percentage of previously dead neurons in the selected paths of all methods. We observe that all pruning based methods have converged to selecting undesirable paths. ∼70% of neurons in the top 1% selected neurons of GreedyPruning are originally dead neurons. Another noteworthy observation is the effect of the initial value of gates in the DGR method. When gates are initialized to 1, the paths do not drift away from the original active path as much as they do with random (uniform [0 1]) initialization (DGR(init=r)). Nevertheless, still ∼10% of neurons in the top 1% of DGR(init=1) are originally dead neurons. A path with sparsity 99% is not a subset of path with sparsity 90% for paths selected by pruning objective. We analyze the overlap of the selected paths using the Jaccard similarity between path indicators $\mathbf{e}$ in Fig. 2a. Note the similarity between NeuronIntGrad and DGR(init=1) compared to DGR(init=r). This suggests that when initializing DGR with 1, the selected paths do not drift significantly to undesired paths, and they still roughly contain the critical neurons, explaining why Wang et al. (2018) observed meaningful paths. We also perform a layer-wise similarity analysis between paths in Fig. 2b. We observe that the overlap between paths of NeuronIntGrad and NeuronMCT increases as we move towards final layers, implying that the overall difference between their paths is due to differences in earlier layers.

Path decoding: Neural decoding tries to estimate the input to the network from its encoded neural representation. Previous work (Olah et al., 2017; Mahendran & Vedaldi, 2015) focuses on feature visualization of the entire network. The question we are interested in here is what does the path corresponding to the input, can tell us about that input. This allows us to semantically evaluate the paths derived from different path selection methods. In Fig. 3a, we perform the optimization on the paths selected by different methods. When considering the original network, features related to the predicted ("Fig") class are visualized. When we restricted the network to the active path (Active Subnet), optimization attempts to reconstruct the image. At 90% sparsity we observe that, critical features relevant to the predicted class are reconstructed for contribution-based methods and DGR(init=1). This signifies that the selected sparse path has indeed encoded features relevant to the prediction. However, for paths selected by the pruning objective, the reconstructions resemble noise. In Fig. 3b, we perform a semantic sanity check on the selected top neuron on each path, by

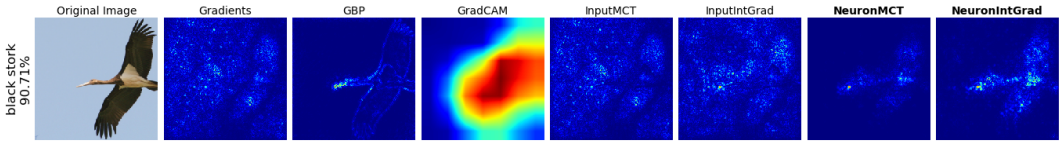

Figure 5: **Comparison with Attribution Methods.** Comparison between attribution maps dervied our proposed methods (right) vs. gradient-based attribution methods on VGG-16. Note the improvement of integrated gradients on the neurons (NeuronIntGrad) over integrated gradients on input (InputIntGrad). More examples for VGG-16 and also ResNet-50 are provided in Appendix D.

analyzing its semantics (Olah et al., 2017). We observe that the selected top neuron by NeuronMCT and NeuronIntGrad is semantically highly relevant to the input, as the neuron is responsible for the bird's eye. DGR(init=1) top neuron is also relevant as it corresponds to feathers. While for the other methods, the top selected neuron is semantically irrelevant, further confirming that the selected paths are not encoding the critical features learned by the network.

3 Interpreting Neural Response via Critical Paths

In Section 3.1 we show that paths selected by NeuronIntGrad and NeuronMCT are locally linear. The local linearity and stability of gradient is later used in Section 3.2 for input feature attribution and understanding to which features in the input the paths correspond.

3.1 Local Linearity of Paths of Critical Neurons

Networks with piecewise linear activation functions are piecewise linear in their output domain (Montúfar et al., 2014), and thus are linear at a specific point $\mathbf{x}$, and $\forall i, j$:

$$\Phi_\theta(\mathbf{x}) = (\nabla_\mathbf{x}\Phi_\theta(\mathbf{x}))^\top\mathbf{x} + \mathbf{b}^{L+1} \; ; \; \mathbf{z}^i_j = (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\mathbf{x} + \mathbf{b}^i_j \tag{5}$$

Although the network degenrates into a linear function at a given point, this does not mean it is locally linear. Indeed both the value (Goodfellow et al., 2016) and the gradient (Ghorbani et al., 2019) of the function are unstable around a point. For discussing the local linearity of the rectified network, we need the concept of activation pattern (Raghu et al., 2017; Lee et al., 2019):

Definition 3 (Activation Pattern) *An activation pattern $\mathcal{AP}$ is a set of indicators for neurons denoted by $\mathcal{AP} = \{\mathbb{1}(\mathbf{a}^i_j)\}^N$ where $\mathbb{1}(\mathbf{a}^i_j) = 1$ if $\mathbf{a}^i_j > 0$ and $\mathbb{1}(\mathbf{a}^i_j) = 0$ if $\mathbf{a}^i_j \leq 0$.*

The feasible set $S(\mathbf{x})$ of an activation pattern is the input regions where the activation pattern is constant and thus the function is linear (and has stable gradients). Let $\mathcal{B}(\mathbf{x})_{\epsilon,2} = \{\bar{\mathbf{x}} \in \mathbb{R}^D : ||\bar{\mathbf{x}} - \mathbf{x}||_2 \leq \epsilon\}$ denote the ℓ_2-ball around $\mathbf{x}$ with radius ϵ, and let $\hat{\epsilon}_{\mathbf{x},2}$ denote the largest ℓ_2-ball around $\mathbf{x}$ where the activation pattern is fixed and the function is linear, *i.e.*

$$\hat{\epsilon}_{\mathbf{x},2} \doteq \max_{\epsilon \geq 0 : \mathcal{B}_{\epsilon,2}(\mathbf{x}) \subseteq S(\mathbf{x})} \epsilon \tag{6}$$

$\hat{\epsilon}_{\mathbf{x},2}$ is the minimum ℓ_2 distance between $\mathbf{x}$ and the corresponding hyperplanes of all neurons $\mathbf{z}^i_j$ (Lee et al., 2019). The hyperplane defined by neuron $\mathbf{z}^i_j$ at point $\mathbf{x}$ is $\{\bar{\mathbf{x}} \in \mathbb{R}^D : (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\bar{\mathbf{x}} + \mathbf{b} = 0\}$ or $\{\bar{\mathbf{x}} \in \mathbb{R}^D : (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\bar{\mathbf{x}} + (\mathbf{z}^i_j - (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\mathbf{x}) = 0\}$. If $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$ then the distance between $\mathbf{x}$ and $\mathbf{z}^i_j$ is

$$|(\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\mathbf{x} + (\mathbf{z}^i_j - (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top\mathbf{x})|/||\nabla_\mathbf{x}\mathbf{z}^i_j||_2 = |\mathbf{z}^i_j|/||\nabla_\mathbf{x}\mathbf{z}^i_j||_2 \tag{7}$$

Note for a neuron $\mathbf{z}^i_j$, if $\nabla_\mathbf{x}\mathbf{z}^i_j = 0$, then $\mathbf{z}^i_j = \mathbf{b}^i_j$. In order for the activation of this neuron to change, the $\nabla_\mathbf{x}\mathbf{z}^i_j$ and consequently the $\mathcal{AP}$ has to change. Therefore the distance is goverened by neurons for which $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$. Lee et al. (2019) prove that $\hat{\epsilon}_{\mathbf{x},2} = \min_{i,j} |\mathbf{z}^i_j|/||\nabla_\mathbf{x}\mathbf{z}^i_j||_2$. Since $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$, the existence of a linear region $\hat{\epsilon}_{\mathbf{x},2}$ depends on $|\mathbf{z}^i_j|$ not being equal to zero.

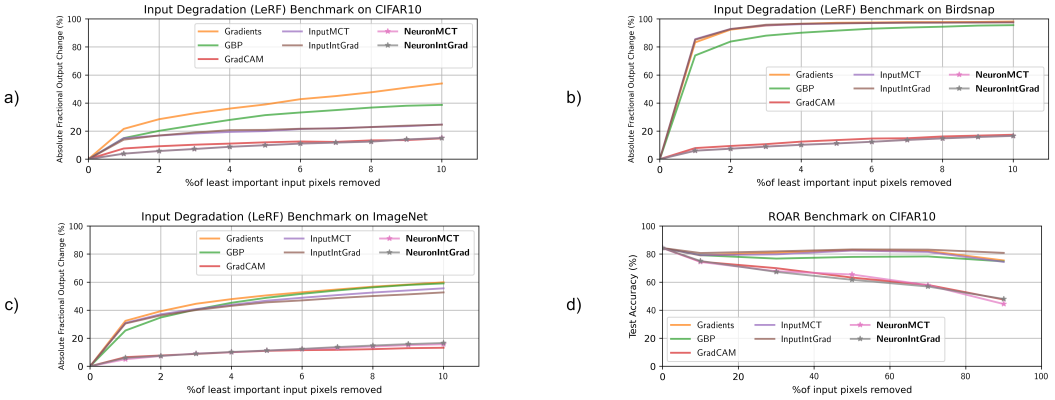

Figure 6: **Feature Importance.** a,b,c) LeRF (ResNet-50) on Cifar10, Bridsnap, and ImageNet. d) Remove and retrain (ROAR) (ResNet-8, Cifar10). In all experiments our methods perform best.

Locally linear network approximation: In order to approximate the original model $\Phi_\theta(\mathbf{x})$ with a selected path $\mathbf{e}$, we replace each neuron $\mathbf{a}^i_j$ which is not in the path, *i.e.* $\mathbf{e}^i_j = 0$ with a constant value equal to the initial value ($\mathbf{a}^i_j$) of that neuron. Note the new constant is not a neuron anymore and thus does not propagate gradient. Replacing the neuron with its initial value keeps $\mathcal{AP}$, and neurons $\mathbf{z}^i_j$ unchanged. We denote such an approximate model by $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$.

Proposition 4 *In a* ReLU *rectified neural network with* $\Phi_\theta(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$*, for a path defined by* $[\mathbf{e}^i_j]^N$*, if* $\mathbf{a}^i_j > 0 \; \forall \; \mathbf{e}^i_j = 1$*, then there exists a linear region* $\hat{\epsilon}_{\mathbf{x},2} > 0$ *for* $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ *at* $\mathbf{x}$*.*

Proposition 5 *Using NeuronIntGrad and NeuronMCT, if* $\mathbf{c}_\kappa > 0$*, then* $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ *at* $\mathbf{x}$ *is locally linear.*

3.2 Input feature attribution through paths of critical neurons

We investigate the correspondence between input features and the selected paths, and use it to understand critical input features for the original network. Approximating the network with a surrogate model with well-behaved gradient is used by Dombrowski et al. (2019), where they find a surrogate low-curvature model, and LIME (Mishra et al., 2017) which finds a linear model that fits the input/output samples in a neighborhood. The weights (gradients) of a linear model represent the contributions of each corresponding input feature. Based on Proposition 5 the approximate model $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ is locally linear for NeuronMCT and NeuronIntGrad. Thus, we can derive linear approximations for the model using the critical high-level features of the model. We use different levels of sparsity and observe the most critical input features for the response (Fig. 4).

Considering interdependencies between input features: If we compute the marginal contribution or Shapley value for a single feature of the input, *e.g.* a pixel, the distributional interdependencies and correlations between the pixels are not considered. For instance, ablating a single pixel from an object in an image does not affect the score of an Oracle classifier, in any coalition. One must consider that all the pixels are related when computing the marginal contribution and Shapley value for the object (all pixels considered as one feature). Neurons inherently consider (learn) the correlations between their input pixels. Thus, by computing the contribution of individual neurons, we are considering a complex group of pixels and their distributional relationships (more in Appendix B.1). In our experiments, using MCT and integrated gradients on neurons results in significantly better attributions compared to applying them only to input pixels (denoted by InputMCT and InputIntGrad).

Baseline choice: The baseline value in feature attribution is supposed to model the absence of a feature. In the image input domain, several works (Zeiler & Fergus, 2014; Sundararajan et al., 2017) consider the zero (black image) as baseline. However, zero pixel values do not necessarily reflect the absence of a feature and such a choice can be problematic (Sturmfels et al., 2020; Izzo et al., 2020)

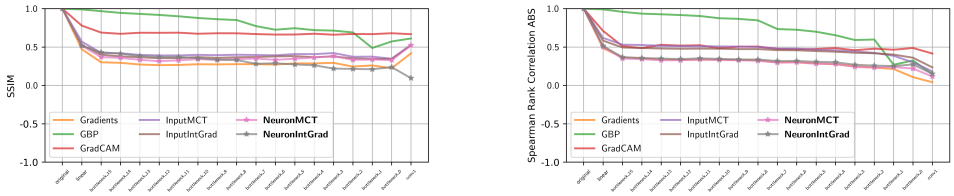

Figure 7: **Randomization-Sensitivity Sanity Check.** Similarity of attributions before and after network (ResNet-50) parameter randomization. Our methods are as sensitive as the network gradient.

in input space. In feature space on the other hand it is intuitive to model the absence with removing a neuron, as the neurons are feature detectors and do not activate if their corresponding feature is absent. The zero baseline is therefore more justified for neurons than input space, as also used by Ancona et al. (2019); Shrikumar et al. (2017). Nonetheless, sampling methods (Lundberg & Lee, 2017; Sturmfels et al., 2020; Sundararajan & Najmi, 2020) could yield further improvements.

3.3 Feature attribution experiments

Input degredation (Samek et al., 2017) and Remove-and-Retrain (ROAR) (Hooker et al., 2019) frameworks evaluate weather the attribution maps are showing important features in the input. Sanity check of Adebayo et al. (2018), checks whether the method is explaining model behavior.

Network parameter randomization sanity check (Adebayo et al., 2018): Several attribution methods, such as LRP$-\alpha 1\beta 0$ (Montavon et al., 2017), Guided Backpropagation (GBP)(Springenberg et al., 2015) generate the same result after the network is randomly initialized, thus not explaining the network (Adebayo et al., 2018; Sixt et al., 2019). Results of randomization-sensitivity of our methods are reported in Fig. 7, showing they are as sensitive as the gradient itself.
Input degradation - LeRF (Samek et al., 2017): Input pixels are perturbed based on their attribution score and the output change is measured. Following Ancona et al. (2017) we remove least relevant features first (LeRF). The results are depicted in Fig. 6(a,b,c). Note the improvement of NeuronIntGrad and NeuronMCT over InputIntGrad (integrated gradients on input) and InputMCT.
Remove and retrain (ROAR) (Hooker et al., 2019): Perturbing input pixels results in artifact images, and thus might cause output change without that pixel being important. ROAR overcomes this problem, by retraining the model on the perturbed dataset. The more the accuracy drops, the better the attribution method has revealed important features. Again we observe that NeuronIntGrad and NeuronMCT are better than their input counterparts (Fig. 6d). (refer to Appendix C for details)

4 Related work

Additional to the work of Wang et al. (2018), and Yu et al. (2018) report that activation paths for inputs within a class overlap and are distinct from other classes. One insightful observation of Wang et al. (2018) is that the paths for adversarial inputs differ. An observation that is more rigorously studied by Qiu et al. (2019), however path selection is not discussed. None of mentioned works leverage the paths for interpreting the response. Schulz et al. (2020) propose feature attribution by placing an information bottleneck (mask) on a chosen layer. The optimized mask is then upsampled to the input. Placing a bottleneck is conceptually similar, however we are masking the entire network then analyze the gradient, whereas Schulz et al. (2020) mask one layer, and upsample the mask.

5 Conclusion

We show that solving the pruning objective does not necessarily yield paths that encode critical input features. We propose finding critical paths based on neurons contributions to the response and show that such paths in rectified networks are provably locally linear. We demonstrate how these critical paths can be leveraged for interpreting the neural response and propose a feature attribution methodology and validate its efficacy by comparative analysis in several popular benchmarks.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

Marco Ancona, Cengiz Öztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for Shapley values approximation. In *36th International Conference on Machine Learning, ICML 2019*, 2019. ISBN 9781510886988.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, pp. 13589–13600, 2019.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2950–2958, 2019.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3681–3688, 2019.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Journal of Machine Learning Research*, 2011.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pp. 164–171, 1993.

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

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 9737–9748, 2019.

Cosimo Izzo, Aldo Lipani, Ramin Okhrati, and Francesca Medda. A baseline for shapely values in mlps: from missingness to neutrality. *arXiv preprint arXiv:2006.04896*, 2020.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Guang He Lee, David Alvarez-Melis, and Tommi S. Jaakkola. Towards robust, locally linear deep networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pp. 537–543, 2017.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 2017. ISSN 00313203. doi: 10.1016/j.patcog.2016.11.008.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, 2014.

Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the imporance of single directions for generalization. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pp. 107–115, 1989.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

Yuxian Qiu, Jingwen Leng, Cong Guo, Quan Chen, Chao Li, Minyi Guo, and Yuhao Zhu. Adversarial defense through network profiling based path extraction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. ISBN 9781728132938. doi: 10.1109/CVPR.2019.00491.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. ISBN 9781510855144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pp. 1135–1144, New York, New York, USA, aug 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL http://dl.acm.org/citation.cfm?doid=2939672.2939778.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. ISSN 21622388. doi: 10.1109/TNNLS.2016.2599820.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xWh1rYwB.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning, ICML 2017*, 2017. ISBN 9781510855144.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. *arXiv*, pp. arXiv–1912, 2019.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Transactions on Signal Processing*, 66(15): 4090–4104, 2018.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *37th International Conference on Machine Learning, ICML 2020*, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. ISBN 9781510855144.

Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8906–8914, 2018.

Fuxun Yu, Zhuwei Qin, and Xiang Chen. Distilling critical paths in convolutional neural networks. *arXiv preprint arXiv:1811.02643*, 2018.

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

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the Importance of Individual Units in CNNs via Ablation. jun 2018. URL `http://arxiv.org/abs/1806.02891`.

Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019.

A PROOFS

A.1 PROOF OF LEMMA 1

Lemma 1 (Dead Neurons) *Considering* $\mathbf{a}^i$ *as the input at layer* i *to the following layers of the network defined by function* $\Phi_\theta^{>i}(.) : \mathbb{R}^{N_i} \to \mathbb{R}$*, the Shapley value of a neuron* $\mathbf{a}_j^i$ *defined by* $\sum_{C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i} \frac{|C|!(N_i-|C|-1)!}{N_i}(\Phi_\theta^{>i}(C \cup \mathbf{a}_j^i) - \Phi_\theta^{>i}(C))$ *is zero if the neuron is dead (*$\mathbf{a}_j^i = 0$*).*

For any layer i, the Shapley value (with baseline zero) of a neuron $\mathbf{a}_j^i$ is defined as:

$$\sum_{C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i} \frac{|C|!(N_i - |C| - 1)!}{N_i}(\Phi_\theta^{>i}(C \cup \mathbf{a}_j^i) - \Phi_\theta^{>i}(C)) \,, \tag{8}$$

where $\Phi_\theta^{>i}$ denotes the neural function after layer i. The input to $\Phi_\theta^{>i}$ is the activation vector $\mathbf{a}^i$. We need to show that for all $\mathbf{a}_j^i$ and all possible coalitions $C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$:

$$\Phi_\theta(C \cup \mathbf{a}_j^i; \mathbf{x}) = \Phi_\theta(C; \mathbf{x}) \,. \tag{9}$$

We know for any $\mathbf{a}^i$ the outputs of neurons in the next layer are:

$$\mathbf{z}^{i+1} = \theta^{i+1}\mathbf{a}^i + \mathbf{b}^{i+1} \,. \tag{10}$$

As the baseline is considered zero, ablating a neuron $\mathbf{a}_j^i$ is done by $\mathbf{a}_j^i = 0$. Thus $\mathbf{z}^{i+1}$ does not change by ablation of $\mathbf{a}_j^i$ for any coalition C. As $\mathbf{z}^{i+1}$ does not change, Φ_θ does not change, thus we get $\Phi_\theta(C \cup \mathbf{a}_j^i; \mathbf{x}) = \Phi_\theta(C; \mathbf{x})$.

A.2 PROOF OF PROPOSITION 4

Proposition 4 *In a* ReLU *rectified neural network with* $\Phi_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$*, for a path defined by* $[\mathbf{e}_j^i]^N$*, if* $\mathbf{a}_j^i > 0 \;\forall\; \mathbf{e}_j^i = 1$*, then there exists a linear region* $\hat{\epsilon}_{\mathbf{x},2} > 0$ *for* $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ *at* $\mathbf{x}$*.*

In Section 3.1 we discuss that for $\Phi_\theta(\mathbf{x})$ the existence of linear region $\hat{\epsilon}_{\mathbf{x},2} > 0$ depends on the value of $|\mathbf{z}_j^i|$ not being equal to zero. We are selecting a path $[\mathbf{e}_j^i]^N$ where for each neuron $\mathbf{a}_j^i > 0$ and thus $\mathbf{z}_j^i > 0$. If we replace every neuron not on the path with a constant value equivalent to the original value of the activation of that neuron, the activation pattern $\mathcal{AP}$ remains constant, and thus we get a new approximate neural network $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$, where all neurons $\mathbf{z}_j^i > 0$. Therefore $\hat{\epsilon}_{\mathbf{x},2} \neq 0$ and there exists a linear region.

A.3 PROOF OF PROPOSITION 5

Proposition 5 *Using NeuronIntGrad and NeuronMCT, if* $\mathbf{c}_\kappa > 0$*, then* $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ *at* $\mathbf{x}$ *is locally linear.*

$$\mathbf{c}_j^i = |\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}_j^i := 0)| = |\mathbf{a}_j^i \nabla_{\mathbf{a}_j^i} \Phi_\theta(\mathbf{x})| \tag{11}$$

For NeuronMCT and NeuronIntGrad the contributions are assigned by:

$$\mathbf{c}_j^i = \mathbf{a}_j^i \int_{\alpha=0}^{1} \frac{\partial \Phi_\theta(\alpha \mathbf{a}_j^i; \mathbf{x})}{\partial \mathbf{a}_j^i} \mathrm{d}\alpha \tag{12}$$

and

$$\mathbf{c}_j^i = \mathbf{a}_j^i \int_{\alpha=0}^{1} \frac{\partial \Phi_\theta(\alpha \mathbf{a}_j^i; \mathbf{x})}{\partial \mathbf{a}_j^i} \mathrm{d}\alpha \tag{13}$$

respectively. It is evident that if $\mathbf{c}_j^i > 0$ then $\mathbf{a}_j^i > 0$. Hence according to Prop. 4 the selected paths and the approximate $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ is locally linear.

A.4 PROOFS FOR AXIOMS OF MARGINAL CONTRIBUTION

Defining marginal contribution of neuron $\mathbf{a}_j^i$ at layer i as:

$$\mathbf{c}_j^i = \Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i}) - \Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i) \tag{14}$$

A.4.1 NULL PLAYER

The null player axiom asserts that if a neuron is a null player, *i.e.*

$$\Phi_\theta^{>i}(S \cup \mathbf{a}_j^i) = \Phi_\theta^{>i}(S)\,, \tag{15}$$

for all $S \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$, then $\mathbf{c}_j^i$ must be zero.

Eq. 15 is assumed for all S, therefore by substituting $S = \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$, in Eq. 15 we get:

$$\Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i}) = \Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i)\,, \tag{16}$$

which results in $\mathbf{c}_j^i = 0$.

A.4.2 SYMMETRY

The symmetry axiom asserts for all $S \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \{\mathbf{a}_j^i, \mathbf{a}_k^i\}$ if

$$\Phi_\theta^{>i}(S \cup \mathbf{a}_j^i) = \Phi_\theta^{>i}(S \cup \mathbf{a}_k^i) \tag{17}$$

holds, then $\mathbf{c}_k^i = \mathbf{c}_j^i$.

Eq. 17 is assumed for all S, therefore by substituting $S = \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \{\mathbf{a}_j^i, \mathbf{a}_k^i\}$, in Eq. 15 we have:

$$\Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i) = \Phi_\theta^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_k^i)\,. \tag{18}$$

By substituting into Eq. 14, we obtain $\mathbf{c}_k^i = \mathbf{c}_j^i$.

B FURTHER DISCUSSIONS

B.1 INTERDEPENDENCIES BETWEEN INPUT FEATURES

If we compute the marginal contribution or Shapley value for a single feature of the input, *e.g.* a pixel, the distributional interdependencies, and correlations between the pixels are not considered. This is not to be confused with the interdependency that the Shapley value considers by taking different coalitions into account. For instance, ablating a single pixel from an object in an image does not affect the score of an Oracle classifier, in any coalition. One must consider that all the pixels are related and exist in one object when computing the marginal contribution and Shapley value for the object (all pixels considered as one feature). We can observe a consequence of this phenomenon, in the different results obtained by Ancona et al. (2017) when removing single pixels (occlusion-1) or removing patches, where the latter results in more semantic attribution maps. Several works implicitly consider such interdependency by masking a group of pixels. The question is what mask should we look for, as the prior information about the dependency of pixels is not available. There are 2^N possible masks that one can select. Moreover, a larger mask containing a feature might get the same or higher contribution score as the mask of the feature. Therefore in (Fong & Vedaldi, 2017; Fong et al., 2019) priors such as the size of the mask are used. These methods look for the smallest masks with the highest contribution. In the regime of neural networks, we encounter more problems with mask selection. If we do not enforce any prior, we can get adversarial masks (Fong & Vedaldi, 2017; Goodfellow et al., 2014). Therefore, several works(Fong & Vedaldi, 2017; Fong et al., 2019) enforce priors such as smoothness of the masks. On the other hand, if we use the prior encoded in

the network (which is learned from the distribution of the data), we implicitly consider the group of pixels that are correlated with each other. Thus by computing the contribution of individual neurons, we are considering a complex group of pixels and their distributional relationships.

C IMPLEMENTATION DETAILS

The sparsity level of ResNet-50 is 70% and VGG-16 is 90% in the experiments, unless stated otherwise.

C.1 NETWORK PARAMETER RANDOMIZATION SANITY CHECK (ADEBAYO ET AL., 2018)

All attribution methods are run on ResNet50 (PyTorch pretrained) and on 1k ImageNet images. The acquired attribution maps from all methods are normalized to [-1 1] as stated by (Adebayo et al., 2018). The layers are randomized from a normal distribution with mean=0 and variance=0.01 in a cascading manner from the last to the first. After the randomization of each layer, the similarity metrics (SSIM and Spearman Rank Correlation) are calculated between the map from the new randomized model and the original pretrained network. Methods that are not sensitive to network parameters (like GBP) would hence lead to high levels of similarity between maps from normalized networks and the original map.

C.2 INPUT DEGRADATION - LERF (SAMEK ET AL., 2017)

We report results on CIFAR using a custom ResNet8 (three residual blocks), Birdsnap using ResNet-50, and ImageNet (validation set) using ResNet-50. We show the absolute fractional change of the output as we remove the least important pixels. Lower curves mean higher specificity of the methods. Note that, for NeuronMCT and NeuronIntGrad, the pixel perturbation process is performed on the original model not on the critical paths selected by these methods. The critical paths are only used to obtain the attribution maps and not after.

C.3 REMOVE AND RETRAIN (ROAR) (HOOKER ET AL., 2019)

We perform the experiments with top 30; 50; 70; 90 of pixels perturbed. The model is retrained for each attribution method (8 methods) on each percentile (5 percentiles) 3 times. Due to the large number of retraining sessions required, we cannot report this benchmark on other datasets. We evaluate this benchmark on CIFAR-10 (60k images, 32x32) with a ResNet-8 (three residual blocks).

D SUPPLEMENTARY RESULTS

D.1 REMOVE AND RETRAIN (ROAR)

Table 1: ROAR: AUCs reported for each attribution method. The lower the AUC the better.

	GRADIENT	GBP	GRADCAM	INPUTMCT	INPUTINTGRAD	NEURONMCT	NEURONINTGRAD
CIFAR-10	0.728	0.702	0.584	0.723	0.741	**0.580**	**0.574**

D.2 INPUT DEGRADATION - LERF

Table 2: Input degredation (LeRF): AUCs reported for each attribution method. The lower the AUC the better.

	GRADIENT	GBP	GRADCAM	INPUTMCT	INPUTINTGRAD	NEURONMCT	NEURONINTGRAD
CIFAR-10	0.037	0.028	0.010	0.019	0.019	**0.009**	**0.009**
BIRDSNAP	0.090	0.085	0.012	0.090	0.090	**0.010**	**0.010**
IMAGENET	0.046	0.044	**0.009**	0.043	0.041	0.010	0.010

D.3 DEAD NEURON PREVALENCE

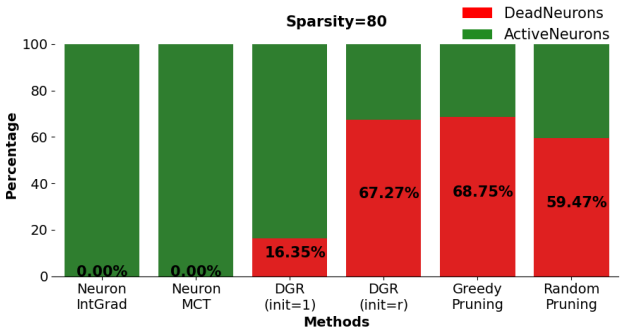

Figure 8: **Dead Neuron Prevalence.** The percentage of originally dead neurons in the selected paths of different methods reported for sparsity of 80%. All paths selected by pruning objective contain originally dead (now active) neurons

D.4 PATH ANALYSIS - ENTIRE NETWORK

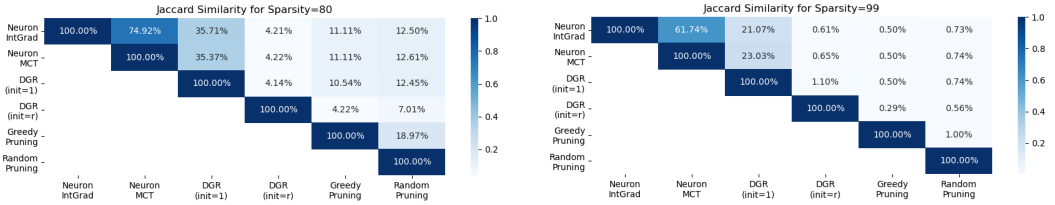

Figure 9: **Path Analysis.** Overlap between paths from different methods in entire network. Among the pruning-based methods, only the path selected by DGR(init=1) overlaps with contribution-based methods.

D.5 PATH ANALYSIS - LAYERWISE

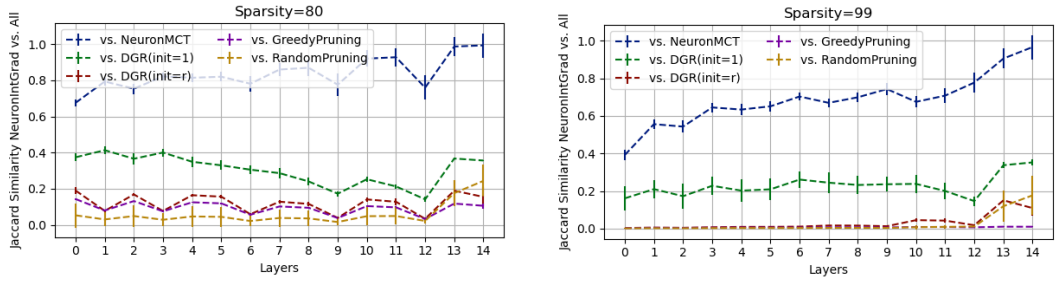

Figure 10: **Path Analysis.** Overlap between paths from different methods in different layers of VGG-16. Among the pruning-based methods, only the path selected by DGR(init=1) overlaps with NeuronIntGrad.

D.6 GRADIENT VISUALIZATION

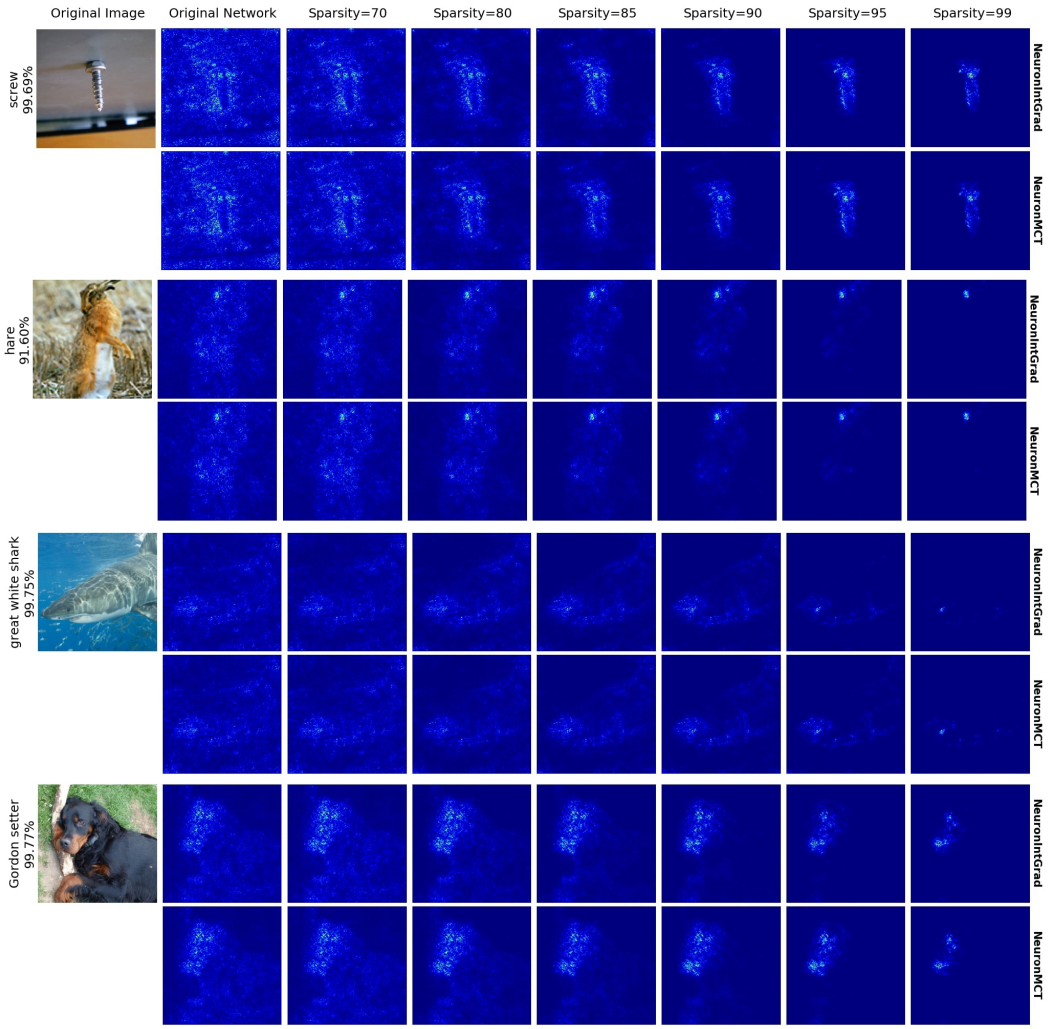

Figure 11: **Gradient Visualization.** The gradients of the locally linear critical paths at different sparsity levels for NeuronIntGrad (top) and NeuronMCT (bottom).

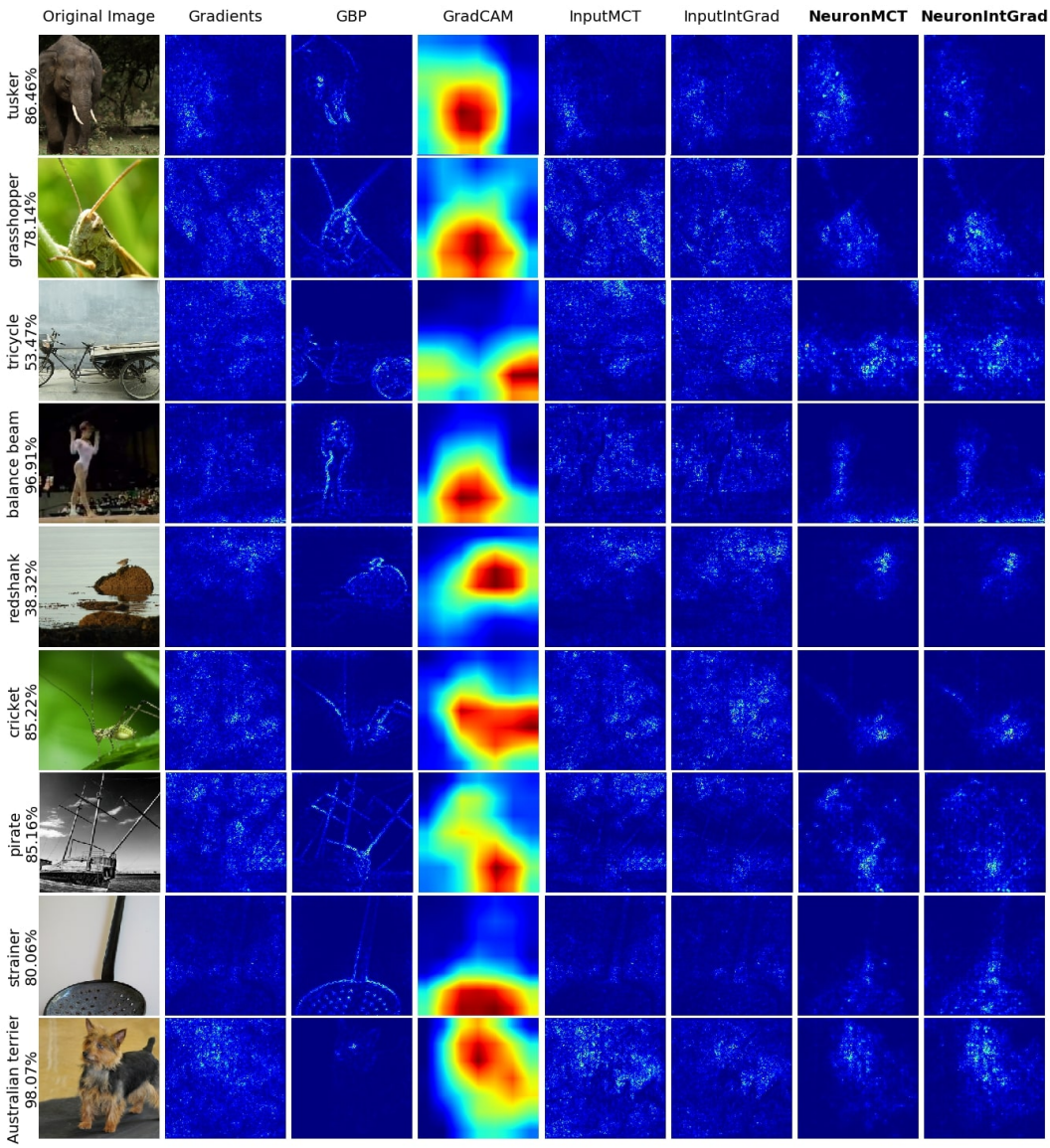

Figure 12: **Comparison with Attribution Methods.** Comparison between attribution maps dervied our proposed methods (right) vs. gradient-based attribution methods on **ResNet-50**. Note the improvement of integrated gradients on the neurons (NeuronIntGrad) over integrated gradients on input (InputIntGrad).

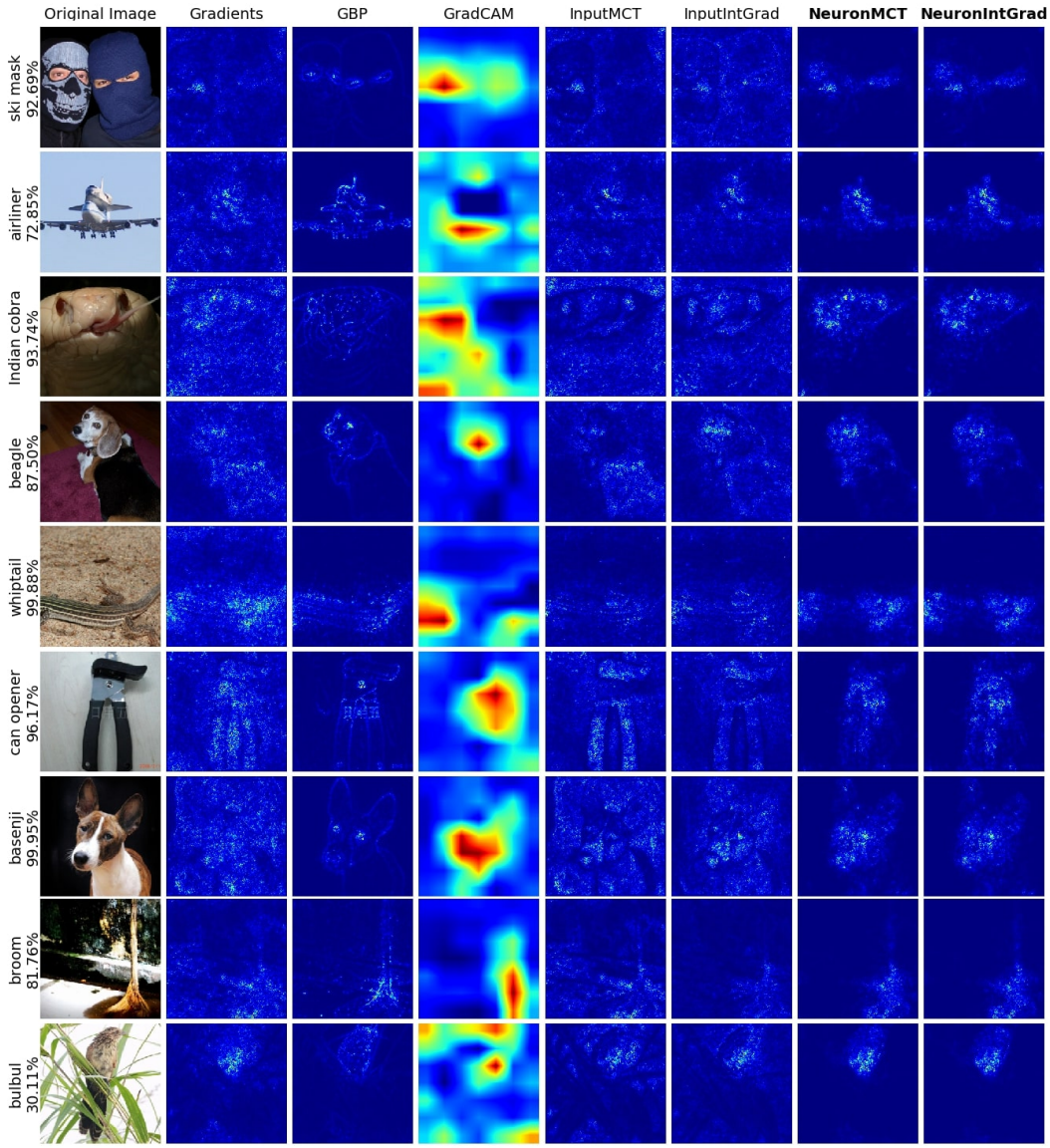

Figure 13: **Comparison with Attribution Methods.** Comparison between attribution maps dervied our proposed methods (right) vs. gradient-based attribution methods on **VGG-16**. Note the improvement of integrated gradients on the neurons (NeuronIntGrad) over integrated gradients on input (InputIntGrad).