Explaining ViTs Using Information Flow

Chase Walker[1] **Md Rubel Ahmed**[2] **Sumit Kumar Jha**[3] **Rickard Ewetz**[1]
University of Florida[1] University of Central Florida[2] Florida International University[3]

Abstract

Computer vision models can be explained by attributing the output decision to the input pixels. While effective methods for explaining convolutional neural networks have been proposed, these methods often produce low-quality attributions when applied to vision transformers (ViTs). State-of-the-art methods for explaining ViTs capture the flow of patch information using transition matrices. However, we observe that transition matrices alone are not sufficiently expressive to accurately explain ViT models. In this paper, we define a theoretical approach to creating explanations for ViTs called InFlow. The framework models the patch-to-patch information flow using a combination of *transition matrices* and *patch embeddings*. Moreover, we define an algebra for updating the transition matrices of series connected components, diverging paths, and converging paths in the ViT model. This algebra allows the InFlow framework to produce high quality attributions which explain ViT decision making. In experimental evaluation on ImageNet, with three models, InFlow outperforms six ViT attribution methods in the standard insertion, deletion, SIC and AIC metrics by up to 18%. Qualitative results demonstrate InFlow produces more relevant and sharper explanations. Code is publicly available at `https://github.com/chasewalker26/InFlow-ViT-Explanation`.

1 INTRODUCTION

The rise of end-to-end trainable deep neural models has propelled the field of computer vision (CV) to human-level cognition (Siemens et al., 2022). CV models are broadly being deployed within safety-critical applications such as real-time surveillance (Ho et al., 2019) and medical diagnosis (Esteva et al., 2021). However, their increasing complexity has made them opaque. Nevertheless, understanding the decision-making process of neural models is pivotal to establishing trustworthiness. The most common explanations for CV models are attribution methods, which measure the contribution of the input features to a model's output decision (Das and Rad, 2020; Stassin et al., 2024).

Attribution methods for convolutional neural networks (CNNs) have been proposed based on perturbations (Ancona et al., 2018), Shapely values (Lundberg and Lee, 2017), and gradients (Simonyan et al., 2014). Gradient-based methods utilize the gradients of the input w.r.t. the output to explain neural models (Springenberg et al., 2015). Although these methods are model agnostic, it has been observed that attributions produced for vision transformers (ViT) are of substantially lower quality than for CNNs. We illustrate the integrated gradients (IG) attributions (Sundararajan et al., 2017) for a ViT in Figure 1(b). The poor quality of the attributions stems from the higher degree of non-linearity within self-attention layers and the accompanying saturation effects. This has spurred investigations into architecture-specific explanation approaches for ViTs (Abnar and Zuidema, 2020a).

Early attribution methods for ViTs were proposed based on the raw attention (Attn) weights or the attention gradients of the model's last layer (Vaswani et al., 2017). These were used to adapt popular CNN methods like IG to ViTs by using the attention gradients instead of input gradients. However, these methods ignore the propagation of patch information across all transformer layers, which may explain the moderate results in Figure 1(c). Recent ViT methods are centered on understanding the bottom-up propagation of data within the architecture (Abnar and Zuidema, 2020a; Chen et al., 2023a). The self-attention matrix in each layer is a transition matrix where every row sums to 1. The matrix specifies how the patch embeddings in each transformer layer are mixed together to form the patch

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

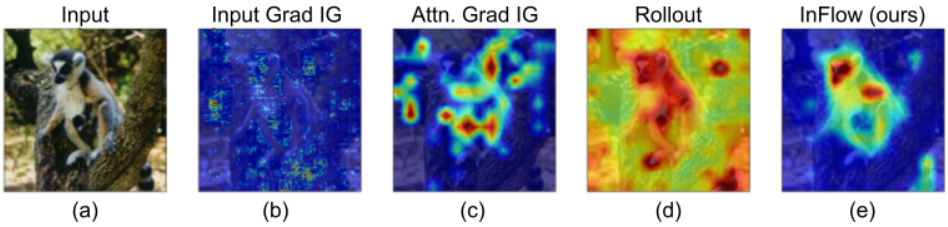


Figure 1: Gradient-based methods perform poorly for ViTs as they are oblivious to the model's information flow (b) and (c) (Sundararajan et al., 2017). Partial information flow modeling yields better results (d) (Abnar and Zuidema, 2020a) and comprehensive flow modeling with InFlow (e), creates the best explanation of the input (a).

embeddings of the next transformer layer. Current approaches model the propagation of patch information across layers as a Markov Decision Process (Yuan et al., 2021a). Attention Rollout therefore attempts to capture the propagation of data by multiplying these transition matrices of all layers in series, while capturing the input residual connection by addition with the identity matrix (Abnar and Zuidema, 2020a). This yields promising initial results as shown in Figure 1(d). Many recent studies aim to improve this approach by incorporating additional model information into the attention matrices with relevance values (Chefer et al., 2021b) or performing post-processing of the rollout operation via gradient multiplication (Yuan et al., 2021a; Chen et al., 2023a). However, we observe that transition matrices alone are not sufficiently expressive to model converging paths within ViTs, which makes it impossible to appropriately capture the impact of residual connections and perceptron (MLP) layers.

In this paper, we propose an information flow modeling framework for computing ViT attributions called InFlow. InFlow captures the flow of information within an ViT using both *transition matrices* and *patch embeddings*. The information flow modeling is performed by defining an *algebra* for updating the representations. We summarize our main contributions as follows:

1. We propose to capture the flow of information within ViTs using both transition matrices and patch embeddings. This new expressive representation allows us to define an algebra for modeling the flow of information for series connections, diverging paths, and converging paths within ViTs.

2. This representation and algebra enables, for the first time, the flow of information through the self-attention block, the MLP block, and both residual connections to be properly modeled in a straightforward manner. This allows the bottom-up information flow to be captured and gradients are incorporated to provide class-specific explanations.

3. Across the standard insertion, deletion, SIC, and AIC evaluation metrics, and on three models, InFlow outperforms six ViT attribution methods quantitatively by up to 18% and provides improved visual qualitative results. A preview of the excellent qualitative results is shown in Figure 1(e).

The remainder of this manuscript is presented as follows: related work in Section 2, the information flow model in Section 3, our method in Section 4, experiments in Section 5, and the conclusion in Section 6.

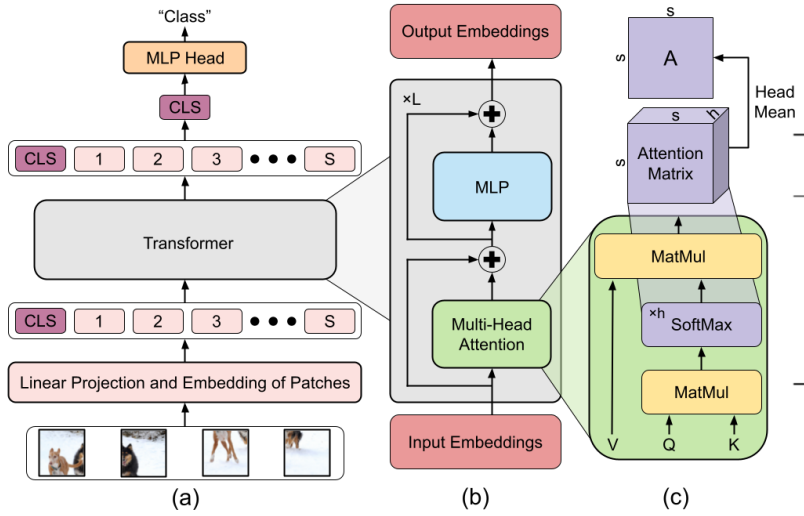
2 BACKGROUND AND RELATED WORK

In this section, we present a discussion of the vision transformer structure and the parts which are relevant to modern ViT explanation methods. Additionally, we discuss how these methods create explanations.

2.1 Vision Transformer

The ViT learns the features of an image by viewing it as a flattened vector of patches which are embedded as tokens (Dosovitskiy et al., 2020). It utilizes self-attention to capture image-wide dependencies between the patches (Vaswani et al., 2017). An overview of the ViT model is shown in Figure 2(a). The model first partitions a 224×224 image into patches of pixels with dimension 16×16, resulting in a new 14×14 image of patches. The patches are tokenized and flattened into a vector of 196 tokens and a special [CLS] token is prepended, resulting in a 197 token sequence. This embedding is passed to twelve series connected transformer layers. Finally, a classification head Z_{CLS} will classify the [CLS] token from the last transformer layer.

The details of the transformer layer are displayed in Figure 2(b). The main components are: the multi-head self-attention (MHA), the input residual connection, the MLP block, and the attention residual connection. The self-attention computation projects the input tokens into matrices Q, K, and V, and performs the operation: $softmax(QK^T)V$. The attention weights are then $A = softmax(QK^T)$, shown in Figure 2(c),



Method	Transition Matrix	Patch Embeddings
Attn		
Attn Grads		
Rollout	✓	
Bi-Attn	✓	
T-Attn	✓	
T-Attr	✓	
InFlow (ours)	✓	✓

(d)

Figure 2: An overview of the ViT architecture (a) with a detailed view of one transformer layer (b) to indicate the 4 components modeled by InFlow and an attention computation overview (c). The table (d) contrasts the components used in the algebra of SOTA methods and our proposed InFlow method as discussed in Section 3.

and the attention gradients are $\partial Z_{CLS}/\partial A$. We highlight these four components which operate on a layer's input embeddings to discuss how existing work models information flow. We omit normalization layers for readability, but they are retained in the model.

2.2 Interpretation of ViT Models

ViT explanations are derived from the [CLS] token of the attention weights or gradients. We first summarize bottom-up approaches using notation introduced in Figure 2. Next, we provide an overview of methods that incorporate class-specific information.

Bottom-up Explanation Approaches: Rollout explained ViTs by modeling the propagation of patches through the model with a series matrix multiplication of all attention matrices across the model's layers (Abnar and Zuidema, 2020a). Each attention matrix A is a transition matrix which captures the patch-to-patch dependencies in a layer. To consider ViTs with h heads, the attention matrix is the average across the heads as seen in Figure 2(c) (Vaswani et al., 2017). Several studies also attempt to model the parallel input residual connection by modeling it as an identity matrix I. To retain the transition matrix property, the resulting representation of a transformer layer is: $(0.5 \cdot A + 0.5 \cdot I)$ (Abnar and Zuidema, 2020a; Chefer et al., 2021b).

This method of modeling the transition matrix for two converging paths is only valid if both paths have equal importance. Unfortunately, we observe that the patch embeddings from the residual connection and the MHA are often of different magnitude, which clearly violates the assumption of equal importance (see results in Figure 7 within Appendix A.3). We now provide a concrete example to illustrate this key limitation of the rollout-based attribution methods in Abnar and Zuidema (2020a); Chefer et al. (2021b); Yuan et al. (2021a); Chen et al. (2023a). Let us have two patch embeddings $E_a = [1]$ and $E_b = [99]$ and the top most row of their associated transition matrices $T_a[1, 0]$ and $T_b = [0, 1]$. Next, these two paths a and b are joined into c and the new patch embedding is obtained as $E_c = E_a + E_b$. The rollout calculation then performs $T_c = 0.5 \cdot [1, 0] + 0.5 \cdot [0, 1] = [0.5, 0.5]$. This disregards that a much greater patch embedding is associated with T_b than T_a. Hence, T_b should have larger weight than T_a when the two transition matrices are combined. This stems from the lack of consideration for patch embeddings when combining the transition matrices. In the next section, we present a new algebra to include the embeddings in the calculation of transition matrices on converging paths. The proposed algebraic approach is contrasted with rollout-based methods in Figure 2(d).

Incorporating Class Information: Transformer Attribution (Chefer et al., 2021b), Transition Attention (Yuan et al., 2021a), and Bidirectional Attention (Chen et al., 2023a) use the concepts of Rollout for bottom-up information propagation modeling. These methods add the incorporation of model gradients in a top-down manner to generate class-specific explanations. Before rollout, T-Attr multiplies each layer's relevance map by the attention gradients measured from the output down to that layer. On the other hand, T-attn performs standard rollout and Bi-attn performs rollout with head importance, then both methods multiply their result with the IG explanation of the last layer's attention

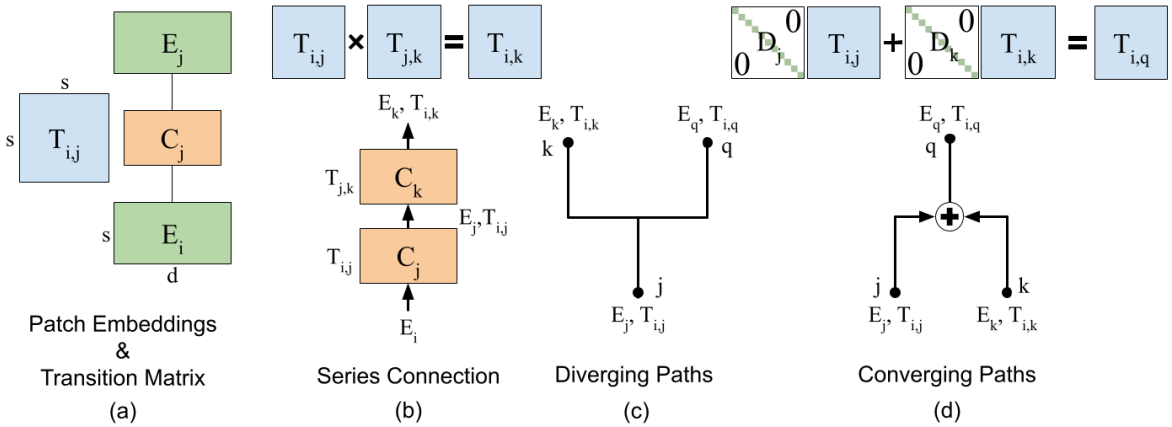


Figure 3: An overview of our algebra which defines our theoretical framework. (a) Defines the preliminaries of patch embeddings and the transition matrix which tracks embedding transformations. (b) Describes the series connection algebra which allows transition matrices in series to be combined into one. (c) Introduces diverging paths where embeddings and transition matrices do not change. Lastly, (d) defines the converging paths algebra which generates a new transition matrix from two pairs of embeddings and transition matrices.

gradients. Instead, we incorporate class-information by multiplying the attention map in each layer by the output gradients *from the input up to that layer*.

3 INFORMATION FLOW MODELING

In this section, we propose our model for capturing the patch-to-patch information flow within a ViT. Our proposed information flow modeling is based on a combination of *transition matrices* and *patch embeddings*. We first formally introduce these two concepts and then define an algebra for updating the representations. The theoretical framework provides a principled foundation for our proposed methodology in the next section. The notation and the algebra is illustrated in Figure 3.

Definition 1. Patch Embedding: Patch embeddings E are the internal representation used in ViTs. The embeddings consist of N image patches and a class token [CLS] patch, for a total of $s = N + 1$ patches embedded into an $s \times d$ matrix where d is the embedding dimension. The representation is shown in Figure 3(a).

Definition 2. Transition Matrix: Let $T_{i,j}$ denote a transition matrix of dimension $s \times s$ which captures the information flow of transforming a patch embedding E_i to E_j. Each entry $T_{i,j}^{nm}$ with $n, m \in [1, s]$ denotes the flow of information from the input patch n to the output patch m. In other words, how much n contributes to the computation of m. Each row in this matrix is a probability vector that sums to 1 which is, by definition, a transition matrix. This is seen in Figure 3(a).

Next, we define an algebra for updating the transition matrix to capture the flow of information within ViTs. Specifically, the algebra captures the impact of series connections, diverging paths, and converging paths.

Definition 3. Series Connection: Consider the input E_i which passes through two series connected components C_j and C_k. They transform E_i to E_j and E_k, respectively, and are represented by the transition matrices $T_{i,j}$ and $T_{j,k}$, respectively. Let us now consider the transformation of E_i to E_k as one transition matrix $T_{i,k}$ which is a function of $T_{i,j}$ and $T_{j,k}$:

$$T_{i,k} = T_{i,j} \times T_{j,k}. \tag{1}$$

This is presented with an example in Figure 3(b).

Definition 4. Diverging Path: Consider an embedding E_j with an associated transition matrix $T_{i,j}$ at a node j. If the embedding is sent along a diverging data path to nodes k and q, the resulting embedding, transition matrix pairs are $E_k, T_{i,k}$ and $E_q, T_{i,q}$. However, if a path has no components, it has a transition matrix I, and therefore:

$$T_{i,j} = T_{i,k} = T_{i,q}. \tag{2}$$

This is exemplified in Figure 3(c).

Definition 5. Converging Path: Consider the embedding and transition matrix pairs $E_j, T_{i,j}$ and $E_k, T_{i,k}$ which are at different nodes j and k. These nodes converge with an addition operation which results in node q and yields the new embedding $E_q = E_j + E_k$ with the transition matrix $T_{i,q}$. This transition matrix, which is a function of the embeddings and transition matrices, is defined as:

$$T_{i,q} = D(E_j, E_k)_j \cdot T_{i,j} + D(E_j, E_k)_k \cdot T_{i,k}. \tag{3}$$

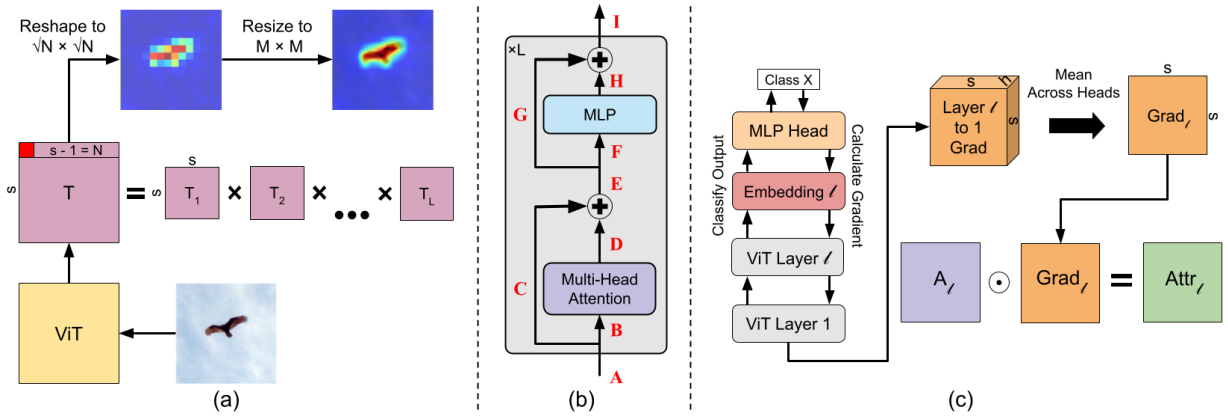


Figure 4: We detail the specifics of the InFlow method. (a) Shows how an explanation is created via bottom-up computation of the model's transition matrix. (b) Shows the components of a transformer layer used to compute the transition matrix of each layer. Lastly, (c) explains how gradients are integrated into our transition matrices.

Here, D is a diagonal matrix which represents the relative importance of each patch between the two different patch embeddings. This matrix is defined as:

$$D(E_a, E_b)_a^{ij} = \begin{cases} \dfrac{||E_a^i||_d^2}{||E_a^i||_d^2 + ||E_b^i||_d^2}, & i == j \\ 0, & i \neq j \end{cases} . \quad (4)$$

Here, E^i is row i of the patch embedding E, i.e., the $1 \times d$ embedding of patch i. The operation $||.||_d^2$ is the L^2 norm operation performed across the embedding dimension d which yields a scalar value. The division operation transforms the scalar into a proportion of the total patch value between E_a^i and E_b^i. The selection of the subscript for D (D_a or D_b) determines the numerator, and thus which patch embedding's relative importance is being calculated. The resulting transition matrix is therefore a weighted sum of the input transition matrices by the input embeddings. Here, patches in the transition matrix with stronger embeddings will dominate the result. This converging path operation is seen in Figure 3(d).

Comparison Against SOTA Approaches: Let us now revisit the example where we have the embedding and transition matrix pairs: $E_a = [1], T_a = [1, 0]$ and $E_b = [99]$ and $T_b = [0, 1]$. Under our new algebra, the converging paths calculation is $T_c = 0.01 \cdot [1, 0] + 0.99 \cdot [0, 1] = [0.01, 0.99]$. This clearly represents both transition matrices fairly, allowing each to contribute proportionally to their patch importance. We empirically observe in Appendix A.3 that the embeddings in converging paths are rarely of equal importance. In addition to this, as Rollout does not use model embeddings, it cannot model the MLP block of the transformer which is only a function of the model embeddings. InFlow is then the sole method which can model the flow of information through the MLP block.

4 THE PROPOSED INFLOW FRAMEWORK

In this section, we introduce the methodology for computing attributions using the InFlow framework. The framework builds on the patch embedding and transition matrix representations and the associated algebra that was described in the previous section. The input to the framework is an image X, a ViT model F, with a class prediction $c = F(X)$. The output is an attribution map Y. We first provide an high-level overview of how the attributions are computed bottom-up using information flow modeling, which is explained in Section 4.1. Next, we describe the detailed modeling of each transformer block in Section 4.2. Lastly, we explain how class-wise gradients are incorporated in the bottom-up process in Section 4.3. An overview of the proposed approach is shown in Figure 4.

4.1 Overview of Bottom-Up Modeling

In this section, we provide an overview of our approach of computing attributions using bottom-up modeling. Consider a transformer with L layers. Let E_l denote the output embedding of layer l and E_0 denote the patch embedding of the input image X. The transition matrix $T_{0,L}$ captures how the output embedding E_L is dependent on the patches of the input embedding E_0. Recall that the [CLS] token is used to perform the final classification using an MLP head. The top-most row in the transition matrix $T_{0,L}$ represents the [CLS] token. By omitting entry $(1, 1)$, the top row can be reshaped into a $\sqrt{N} \times \sqrt{N}$ matrix, which explains each patch's contribution to the classification. Next, the matrix can be projected to the full $M \times M$ input image dimensions. We visualize this entire process in Figure 4(a). We now present computing the transition matrix $T_{0,L}$.

Let $T_{l-1,l}$ denote the transition matrix for the flow of

information between the input and output embeddings of transformer layer l. Note that the input embeddings of layer l are equivalent to the output embeddings to layer $(l-1)$. Next, the transition matrix $T_{0,L}$ can be computed using Eq (1), as follows:

$$T_{0,L} = \prod_{l=0}^{L} T_{l-1,l}, \tag{5}$$

where $T_{-1,0} = I$. Next, we focus on describing how we model each transition matrix $T_{l-1,l}$ using the algebra defined in the previous section.

4.2 Transformer Layer Information Flow

In this section, we describe how information flow is used to capture the transition matrix $T_{l-1,l}$ of a transformer block l. We annotate nine locations within the transformer block with the letters A to I, which is shown in Figure 4(b). Let the input be the patch embedding E_A. The objective is to compute the transition matrix $T_{A,I}$ at I. The patch embeddings $\{E_A, \ldots, E_I\}$ at each location within the transformer block are directly given by the evaluation of the model with respect to the input X. Now, we focus on computing the transition matrix at each location using a bottom-up method.

The transition matrices $T_{A,B}$ and $T_{A,C}$ can be obtained using path divergence from Eq (2). We outline how $T_{A,D}$ is computed from the MHA block below, as this requires a slightly more detailed explanation. Next, the transition matrix $T_{A,E}$ is computed from $T_{A,C}$ and $T_{A,D}$ using the path convergence equation in Eq (3). The equation ensures that an appropriate fraction of the two incoming transition matrices are used based on their respective embeddings. Next, the paths diverge and the transition matrices $T_{A,F}$ and $T_{A,G}$ are obtained. $T_{A,F}$ then goes to the MLP block which results in $T_{A,H}$. We will also explain the details of the $T_{A,H}$ computation below. Finally, we combine $T_{A,G}$, and $T_{A,H}$ via path convergence from Eq (3) to compute $T_{A,I}$.

Modeling of Multi-Headed Attention: We now describe how to calculate $T_{A,D}$ from the multi-head attention block. The attention block has the embedding E_B and transition matrix $T_{A,B}$ as an input. The softmax output of the self-attention operation A is an $h \times s \times s$ representation of the patch-to-patch importance of the patches in the input embedding E_B. By definition, A is a transition matrix where all s rows of s values are vectors with a sum of 1. By removing the number of heads dimension, h, we use this transition matrix directly such that $T_{A,D} = A$. To remove the head dimension, we follow Chen et al. (2023a) and compute $T_{A,D}$ as follows. From the matrix $A^{h\times s\times s}$ and its gradients $\partial A^{h\times s\times s}$, the head importance I_h is computed as: $I_h^{h\times 1} = \text{mean}_{s\times s}(A^{s\times s\times h} \times \partial A^{h\times s\times s})$. The most important attention heads are then chosen: $T_{A,D} = \max_h(I_h^{h\times 1\times 1} \cdot A^{h\times s\times s})$.

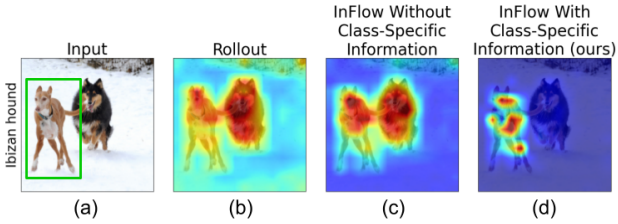


Figure 5: InFlow (c) improvements over Rollout (Abnar and Zuidema, 2020a) (b) for a complex two-class image (a) with the classification of the left dog. The new information flow matrices and algebraic approach yields a low-noise attribution which visually separates the subjects from the background. When gradients are included for the InFlow method (d), the result is a high-quality class-specific explanation.

Modeling of Multi-Layer Perceptron: The MLP block performs scaling on the input embeddings E_F to produce the embeddings E_H. It has been shown that the MLP block acts as a promotion mechanism and applies a scaling factor to each patch embedding Geva et al. (2022). This means only the patch embedding is modified and the source of the patch information stays the same, i.e., $T_{A,F}$ is equal to $T_{A,H}$. We therefore consider the transition matrix of the MLP block to be a diagonal matrix which applies a scaling factor to the each input embedding. Let us consider the L^2 norm of the input and output embedding matrices across the embedding dimension: $||E_F||_d^2$ and $||E_H||_d^2$. We divide these two $s \times 1$ vectors as $||E_H||_d^2/||E_F||_d^2$ to calculate the scaling applied to each patch in E_F. The resulting $s \times 1$ vector is then rearranged into an $s \times s$ matrix where the $s \times 1$ values are on the diagonal, and all off-diagonal values are zeros.

Modeling of the Full Transformer Layer: Now, with these transition matrix representations, we can provide the full equation for a transformer layer. Using the algebra we have defined, we find the transition matrix of a layer l by calculating:

$$\begin{aligned} T_l = (D(E_C, E_D)_D \cdot A_l + D(E_C, E_D)_C \cdot I) \ \times \\ (D(E_G, E_H)_H \cdot \frac{||E_H||_d^2}{||E_F||_d^2} + D(E_G, E_H)_G \cdot I). \end{aligned} \tag{6}$$

The first part (left of the matrix multiplication sign) calculates the information flow through the self-attention block and the first converging connection. The second part then calculates the information flow through the MLP block and the second converging connection.

These are combined via matrix multiplication following the series connection algebra from Eq (1).

In Figure 5 we show the improvements of InFlow (c) over Rollout (b) for an image (a). It is clear that the inclusion of the model embeddings in the flow modeling of InFlow provides substantial improvements in explaining the objects of interest in the image. However, class-specific information is still missing.

4.3 Incorporation of Class Specific Information using Gradients

The InFlow bottom-up approach captures all parts of the input image that are embedded in the class token, however class-specific information cannot be extracted without the use of the MLP classification head. Class-specific gradient information is generally calculated in a top-down (Chefer et al., 2021b) or post-processing (Yuan et al., 2021a; Chen et al., 2023a) manner, which provides a means to identify the specific class features.

We incorporate class-specific information by modifying A to include gradient information. Given a ViT model, $Z_{CLS}(l)$ provides the classification of the output of the embedding calculated by layers 1 to l. We then calculate the "attribution" transition matrix as:

$$\text{Attr}_l = A_l \cdot \frac{1}{d} \sum^{d} \frac{\partial Z_{CLS}(l)}{\partial A_l}, \tag{7}$$

where $A_l^{s \times s}$ is multiplied by the head mean of the attention gradient w.r.t the loss of the given layer's classification, yielding Attr_l with dimensions $s \times s$ as seen in Figure 4(c). Now, we replace A_l in Eq (6) with Attr_l. In Figure 5(d), we show InFlow with gradient information produces a class-specific explanation which accurately explains the left dog target class.

5 EXPERIMENTAL EVALUATION

In this section, we perform experimental evaluation of the proposed InFlow attribution method quantitatively with standard perturbation tests and qualitatively using visual analysis of images. We perform all experiments using PyTorch on the 2012 validation set of ImageNet (Russakovsky et al., 2015). We employ the base ViT-base 16×16 model via Hugging Face (Wu et al., 2020). We also include all results for the ViT-base 32×32 and ViT-tiny 16×16 models from Hugging Face (Wu et al., 2020) in the Appendix. The experiments ran on one desktop with an RTX 4070Ti.

We compare InFlow with Attn (Vaswani et al., 2017), Naive Rollout (N-Roll) (Abnar and Zuidema, 2020a), Rollout (Abnar and Zuidema, 2020a), Bi-Attn (Chen et al., 2023a), T-Attn (Yuan et al., 2021a), and T-Attr (Chefer et al., 2021b). N-Rollout is Rollout where $T_l = A$. It is included to further illustrate the need for information flow modeling. We use the following repositories: (Chefer et al., 2021a) for Attn, and T-Attr, (Yuan et al., 2021b) for IG and T-Attn, (Chen et al., 2023b) for Bi-Attn, and (Abnar and Zuidema, 2020b) for both Rollout methods. Their licences are MIT, Apache-2.0, NA, and NA, respectively. Inputs are reshaped to (224, 224) for all methods, and the methods are not modified from their default parameters unless specified. For all methods, we perform the evaluation without post-processing with IG applied to the attention gradients, which is an ad-hoc approach to improve results. For completeness, we provide results with the post-processing applied in Appendix A.2. We observe that the post-processing can improve the performance of bottom-up methods. Overall, InFlow wins 41/48 tests with a maximum margin of 18%. These results show the promise of InFlow's flow modeling.

5.1 Quantitative Evaluation Metrics

The standard method to efficiently evaluate an attribution map without ground truth is a perturbation test. These tests evaluate if high-value attributions in an explanation are valuable to the model's decision. Perturbation tests come in two forms: insertion or deletion. Insertion blurs the input image and inserts the original image pixels in order of highest attribution. Deletion removes pixels from the input in order of highest attribution. At each perturbation step, the softmax score of the image is measured. A curve of softmax vs perturbation step is generated from the test where the area under the curve is the score. Higher is better for insertion and lower is better for deletion. We use three SOTA perturbation methods: RISE (Petsiuk et al., 2018b), MAS Walker et al. (2024), and PIC (Kapishnikov et al., 2019a) which take varied approaches.

The RISE tests perform insertion and deletion. The blurring baseline for insertion is uniform. Every perturbation step edits N pixels of an $N \times N$ image. The insertion and deletion scores are in the range $[0, 1]$. We use the default configuration from their repository (Petsiuk et al., 2018a) under the MIT license. Test results are reported as the average score for 5000 images.

The MAS tests are also insertion and deletion tests. They compute a density curve which tracks the total attribution inserted or deleted at each step. The softmax curve is then penalized by its distance from this density curve; meaning high value attributions should be important, and low value attributions unimportant. We use the default configuration from their repository (Walker et al., 2024) under the BSD-3 license. Test results are reported as the average score for 5000 images.

The PIC tests: SIC and AIC are insertion tests. SIC

Table 1: A comparison of bottom-up ViT-base 16 × 16 attribution methods against InFlow

	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
Attn (Vaswani et al., 2017)	0.600	0.662	0.759	0.348	0.411	0.649	0.456	0.194
N-Roll (Abnar and Zuidema, 2020a)	0.490	0.559	0.657	0.590	0.068	0.463	0.708	−0.245
Rollout (Abnar and Zuidema, 2020a)	0.689	0.734	0.784	0.243	0.541	0.523	0.485	0.038
T-Attn (Yuan et al., 2021a)	0.632	0.699	0.774	0.414	0.360	0.509	0.573	−0.064
Bi-Attn (Chen et al., 2023a)	0.688	0.748	0.830	0.289	0.541	0.581	0.458	0.123
T-Attr (Chefer et al., 2021b)	0.758	0.814	0.849	0.221	0.628	**0.689**	0.322	0.368
InFlow (ours)	**0.770**	**0.829**	**0.856**	**0.208**	**0.648**	0.687	**0.317**	**0.370**

Table 2: A comparison of the ViT-base 16 × 16 performance improvements from each of InFlow's contributions

	RISE Tests			MAS Tests		
Test Type	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
Rollout	0.774	0.454	0.320	0.512	0.601	−0.088
InFlow (1)	0.826	0.365	0.461	0.560	0.525	0.035
InFlow (2)	0.835	0.319	0.516	0.555	0.498	0.057
InFlow (3) (proposed)	**0.865**	**0.260**	**0.605**	**0.687**	**0.361**	**0.320**

has a modified blur process which uses unique blurring kernels of increasing size from the center of the image to the edges. The size of each blurring segment also determines the portion of the image to be perturbed. As the test progresses, the perturbation size increases. AIC modifies the scoring process. Instead of assembling a curve of softmax scores, it uses binary classification accuracy. We use the default configuration from their repository (Kapishnikov et al., 2019b) under the Apache-2.0 license. AIC and SIC test results are reported as the average score for 2500 images. Fewer images are used than RISE and MAS because the PIC method is computationally more expensive.

5.2 Comparison With Previous Work

In Table 1, we evaluate bottom-up methods with ImageNet. InFlow is a clear winner across seven perturbation tests and closely competes with T-Attr in one of them. We see that InFlow achieves up to a 6% score improvement over the next best method. Let us observe the scores of N-Roll, Rollout, T-Attr, and InFlow. As a reminder, these methods provide an increasing depth of information flow modeling. We see this reflected in the scores as they increase with the more comprehensive transition matrix modeling. This makes it clear that the improved information flow of InFlow creates attributions which more accurately reflect the features important for model classification.

In Table 2, we evaluate how all three of InFlow's contributions lead to improvements in attribution quality. These contributions are: (1) the inclusion of patch embeddings in the calculation of information flow seen in Eq (3), (2) the inclusion of the MLP block in information flow calculation seen in Eq (6), and (3) the inclusion of bottom-up gradient information as presented in Section 4.3. These contributions are indicated as InFlow (1), InFlow (2), and InFlow (3) (proposed) and are compared to the baseline Rollout. We evaluate on the ViT-base 16 × 16 model. We see, as expected, every contribution improves the performance of InFlow over the previous contribution for both metrics. These results show that all three contributions of InFlow lead to the overall improved performance over the SOTA.

Now, we provide a runtime test for all presented ViT attribution methods in Table 3. As a reminder, the methods Attn and Rollout compute attributions with a single forward pass. T-Attr and InFlow compute attributions with one forward pass and one backward pass. T-Attn and Bi-Attn however, use IG post-processing and compute $N + 1$ forward passes and N backward passes, where N is the number of IG steps. In this table, we report the mean and standard deviation of the per-image runtime of each attribution method for

Table 3: A comparison of ViT-base 16 × 16 attribution method runtimes

	Per-Image Runtime(s)
Attn	$\mathbf{0.0041} \pm 0.0007$
Rollout	0.0045 ± 0.0004
T-Attr	0.0501 ± 0.0064
T-Attn	0.2290 ± 0.0061
Bi-Attn	0.2313 ± 0.0069
InFlow (ours)	0.0187 ± 0.0055

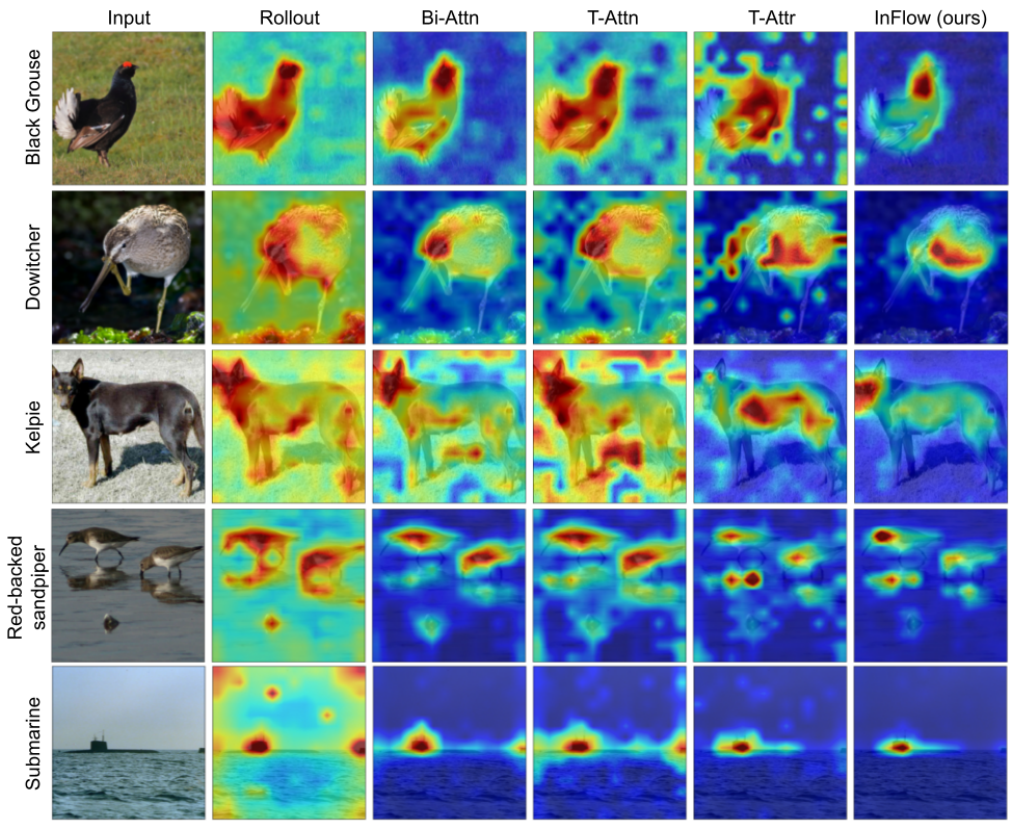


Figure 6: Visual comparison of InFlow against Rollout (Abnar and Zuidema, 2020a), Bi-Attn (Chen et al., 2023a), T-Attn (Yuan et al., 2021a), and T-Attr (Chefer et al., 2021b). InFlow produces better attributions than the competitors with little to no background attributions and increased strength of attributions on the subject.

the ViT-base 16×16 model. We see that InFlow is the third fastest method, only losing to Attn and Rollout which do not require any backpropagation. Although their complexity (in terms of model calls) is the same, InFlow also manages to be faster than the LRP-based T-Attr due to the latter's significant use of recursion which results in a slower execution. Lastly, InFlow is shown to be significantly faster than T-Attn and Bi-Attn due to their use of IG post-processing. This promotes the use of InFlow in real-time systems for downstream applications of attribution methods.

Lastly, for qualitative analysis, we provide visual comparisons of InFlow against the Rollout, T-Attn, Bi-Attn, and T-Attr methods. We show five selected images from ImageNet classes “black grouse”, “dowitcher”, “kelpie”, “red-backed sandpiper", and “submarine". When comparing across each row, we look for high magnitude attributions on the subject, and low magnitude background attributions. We find InFlow explains the most important subject features. It is clear that InFlow provides these qualities, producing high-quality, sharp attributions which better explain the target class of the image. For example, InFlow has substantially less attributions on the background of the "black grouse", "dowitcher", and "kelpie" images. We compare all methods across 144 examples in the Appendix and observe similar trends.

6 CONCLUSION

We propose the bottom-up ViT attribution method InFlow which models patch-to-patch information flow using transition matrices and patch embeddings. This is achieved by defining a new algebra for series connections, diverging paths, and converging paths in the ViT network. This algebra allows for precise modeling of the residual connections, and allows the MLP block to be represented in the transition matrices for the first time. In addition, the inclusion of gradient information allows InFlow to provide high-quality class-specific attributions which achieve SOTA performance on ImageNet. The author's believe InFlow's flow modeling can incite the creation of improved ViT models given the propagation knowledge it provides. Additionally, with the growth of transformers in applications outside of computer vision, the authors believe this work can be extended to improve human trust and understanding of safety-critical systems in many domains.

Acknowledgments

This material is based on research sponsored by DARPA Agreement FA8750-23-2-0501, DOE Awards: DE-SC0023494, DE-SC0024428, and DE-SC0024576, and startup funds from the University of Florida. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, DOE, or the U.S. Government.

References

Abnar, S. and Zuidema, W. (2020a). Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

Abnar, S. and Zuidema, W. (2020b). Rollout code repository. Available at `https://github.com/samiraabnar/attention_flow`.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. (2018). Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*.

Chefer, H., Gur, S., and Wolf, L. (2021a). Transformer attribution code repository. Available at `https://github.com/hila-chefer/Transformer-Explainability`.

Chefer, H., Gur, S., and Wolf, L. (2021b). Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791.

Chen, J., Li, X., Yu, L., Dou, D., and Xiong, H. (2023a). Beyond intuition: Rethinking token attributions inside transformers. *Transactions on Machine Learning Research*.

Chen, J., Li, X., Yu, L., Dou, D., and Xiong, H. (2023b). Bidirectional attention code repository. Available at `https://github.com/jiaminchen-1031/transformerinterp`.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., and Socher, R. (2021). Deep learning-enabled medical computer vision. *NPJ digital medicine*, 4(1):5.

Geva, M., Caciularu, A., Wang, K., and Goldberg, Y. (2022). Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ho, G. T. S., Tsang, Y. P., Wu, C. H., Wong, W. H., and Choy, K. L. (2019). A computer vision-based roadside occupation surveillance system for intelligent transport in smart cities. *Sensors*, 19(8):1796.

Kapishnikov, A., Bolukbasi, T., Viegas, F., and Terry, M. (2019a). Xrai: Better attributions through regions. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4947–4956, Los Alamitos, CA, USA. IEEE Computer Society.

Kapishnikov, A., Venugopalan, S., Avci, B., Wedin, B., Terry, M., and Bolukbasi, T. (2019b). Pic code repository. Available at `https://github.com/PAIR-code/saliency/tree/master/saliency/metrics`.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Petsiuk, V., Das, A., and Saenko, K. (2018a). Rise code repository. Available at `https://github.com/eclique/RISE`.

Petsiuk, V., Das, A., and Saenko, K. (2018b). Rise: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Siemens, G., Marmolejo-Ramos, F., Gabriel, F., Medeiros, K., Marrone, R., Joksimovic, S., and de Laat, M. (2022). Human and artificial cognition. *Computers and Education: Artificial Intelligence*, 3:100107.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). Striving for simplicity: The all convolutional net.

Stassin, S., Corduant, V., Mahmoudi, S. A., and Siebert, X. (2024). Explainability and evaluation of vision transformers: An in-depth experimental study.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Walker, C., Simon, D., Chen, K., and Ewetz, R. (2024). Attribution quality metrics with magnitude alignment. In Larson, K., editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 530–538. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., and Vajda, P. (2020). Base vit 16x16 model from google. Available at `https://huggingface.co/google/vit-base-patch16-224`.

Yuan, T., Li, X., Xiong, H., Cao, H., and Dou, D. (2021a). Explaining information flow inside vision transformers using markov chain. In *eXplainable AI approaches for debugging and diagnosis.*

Yuan, T., Li, X., Xiong, H., Cao, H., and Dou, D. (2021b). Transisiton attention code repository. Available at `https://github.com/XianrenYty/Transition_Attention_Maps`.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] We provide details on how all equations are derived throughout the paper.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] See the supplementary file.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] When we define the theoretical framework in Section 3, we provide assumptions and observations.
   (b) Complete proofs of all theoretical results. [Yes] We do not provide proofs, but we provide experimental results which reinforce the theoretical results.
   (c) Clear explanations of any assumptions. [Yes] When we define the theoretical framework in Section 3, we provide assumptions and observations.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] See the supplementary file.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] See experimental section.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes] See experimental section.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A APPENDIX

In this appendix, we provide supplementary experimental results. We first provide the main paper quantitative results for two additional ViT models in Section A.1. Then we evaluate the quantitative results when post-processing with attention gradients in Section A.2. following this, we evaluate the ratio of embeddings in converging connections in A.3. Lastly, we provide additional qualitative results in Section A.4.

A.1 Extended Evaluation of Bottom-Up Attribution Methods

In this section, we present results for the bottom-up attribution tests for the ViT-base 32×32 and ViT-tiny 16×16 models in Tables 4 and 5, respectively. These results are gathered with the same parameters as those discussed in the paper. When observing the results we see the same expected performance from the previous table. InFlow is a high-performer, winning a majority of tests (14/16) against the SOTA methods. We see the largest win margins in Table 5, with an 18% win in RISE deletion. These results indicate that InFlow extends to multiple ViT models and maintains a high level of performance.

A.2 Evaluation of Post-Processing with Attention Gradients

We evaluate the different methods with attention gradient post-processing for the ViT-base 16×16, ViT-base 32×32, and ViT-tiny 16×16 in Tables 6, 7, and 8. The post-processing is performed by element-wise multiplication of the attribution map with the IG attention gradients Sundararajan et al. (2017). We note the significant increase in scores that almost all methods have when IG post-processing is applied. Nonetheless, when post-processing is applied, InFlow maintains its win rate over all eight test types, winning 20/24 tests. Here, the maximum win margin is nearly 10% in the ViT-tiny 16×16 MAS deletion test.

A.3 Evaluation of Embeddings in Converging Connections

In this section, we provide a case study of the values of embeddings in the two converging connections in the ViT model. Across $10,000$ images we compare the ratio of the embeddings entering the converging connection after the MHA block and after the MLP block. In Figure 7(a) and (b), we study the ratio of the embeddings on the residual connection (C) and after the MHA block (D) $D(E_C, E_D)_C : D(E_C, E_D)_D$ for every token in the first and last transformer layers. In (a) we see the ratio is near 2 for every patch in the embeddings. In (b) we see the ratio varies from 1 to 4. In Figure 7(c) and (d), we study the ratio of the embeddings on the residual connection (G) and after the MLP block (H) $D(E_G, E_H)_G : D(E_G, E_H)_H$. Here, we see the first layer (c) does maintain a ratio near 1, but this significantly changes by the last layer where the ratio approaches 10. Altogether, this makes it clear that Rollout's assumption of equal importance embeddings is flawed Abnar and Zuidema (2020a). In addition, this verifies the foundation for Eq (3).

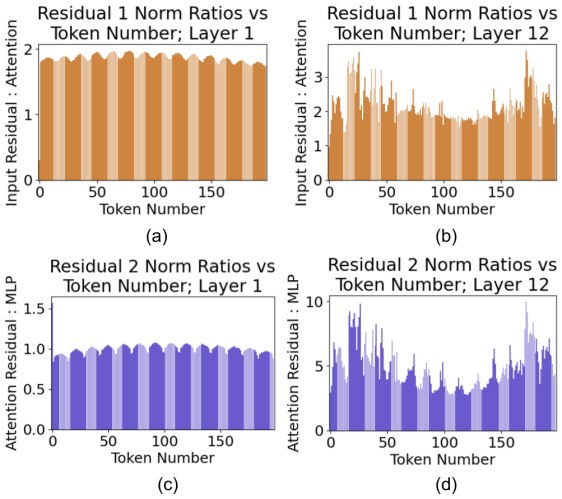


Figure 7: The ratio of embeddings in each converging path for the first and last transformer layer. (a) and (b) present the first convergence. (c) and (d) present the second. As the ratio is greater than 1 for a majority of tokens, the use of Eq (3) is reinforced.

A.4 Additional Qualitative Results

In this section, we provide 144 additional qualitative results for InFlow and SOTA methods. We compare InFlow with Raw Attention (Attn) Vaswani et al. (2017), GradCam (GC) Selvaraju et al. (2017), Integrated Gradients (IG) Sundararajan et al. (2017), Naive Rollout Abnar and Zuidema (2020a), Rollout Abnar and Zuidema (2020a), Bidirectional Attention (Bi-Attn) Chen et al. (2023a), Transition Attention (T-Attn) Yuan et al. (2021a), and Transformer Attribution (T-Attr) Chefer et al. (2021b). We provide the examples for ViT-base 16×16 model on pages 15 to 17, ViT-base 32×32 on pages 18 to 20, and ViT-tiny 16×16 on pages 21 to 23. Notice the consistent clarity provided by InFlow. This is indicated by a lack of attributions on the background, and sharper attributions on the class subject which only highlight the unique features of the subject. We believe InFlow bests the SOTA methods in many images and performs as well as the best SOTA competitor for all other images.

Table 4: A comparison of ViT-base 32×32 attribution methods against InFlow

	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
Attn (Vaswani et al., 2017)	0.687	0.739	0.823	0.272	0.551	0.703	0.370	0.332
N-Roll (Abnar and Zuidema, 2020a)	0.487	0.561	0.648	0.510	0.138	0.464	0.620	−0.156
Rollout (Abnar and Zuidema, 2020a)	0.578	0.643	0.732	0.382	0.350	0.488	0.554	−0.066
T-Attn (Yuan et al., 2021a)	0.584	0.647	0.733	0.383	0.350	0.489	0.554	−0.065
Bi-Attn (Chen et al., 2023a)	0.676	0.730	0.826	0.285	0.541	0.548	0.487	0.062
T-Attr (Chefer et al., 2021b)	**0.750**	**0.796**	0.864	0.217	0.647	0.674	0.355	0.319
InFlow (ours)	0.748	0.794	**0.867**	**0.202**	**0.665**	**0.683**	**0.335**	**0.349**

Table 5: A comparison of ViT-tiny 16×16 attribution methods against InFlow

	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
Attn (Vaswani et al., 2017)	0.619	0.680	0.798	0.158	0.640	0.697	0.245	0.452
N-Roll (Abnar and Zuidema, 2020a)	0.474	0.543	0.634	0.289	0.345	0.449	0.483	−0.034
Rollout (Abnar and Zuidema, 2020a)	0.598	0.659	0.774	0.190	0.584	0.529	0.447	0.082
T-Attn (Yuan et al., 2021a)	0.592	0.654	0.770	0.195	0.575	0.526	0.450	0.077
Bi-Attn (Chen et al., 2023a)	0.607	0.665	0.793	0.170	0.624	0.587	0.386	0.201
T-Attr (Chefer et al., 2021b)	0.657	0.715	0.827	0.131	0.697	0.668	0.294	0.374
InFlow (ours)	**0.699**	**0.754**	**0.852**	**0.107**	**0.746**	**0.675**	**0.292**	**0.383**

Table 6: A comparison of ViT-base 16×16 attribution methods against InFlow with IG post-processing

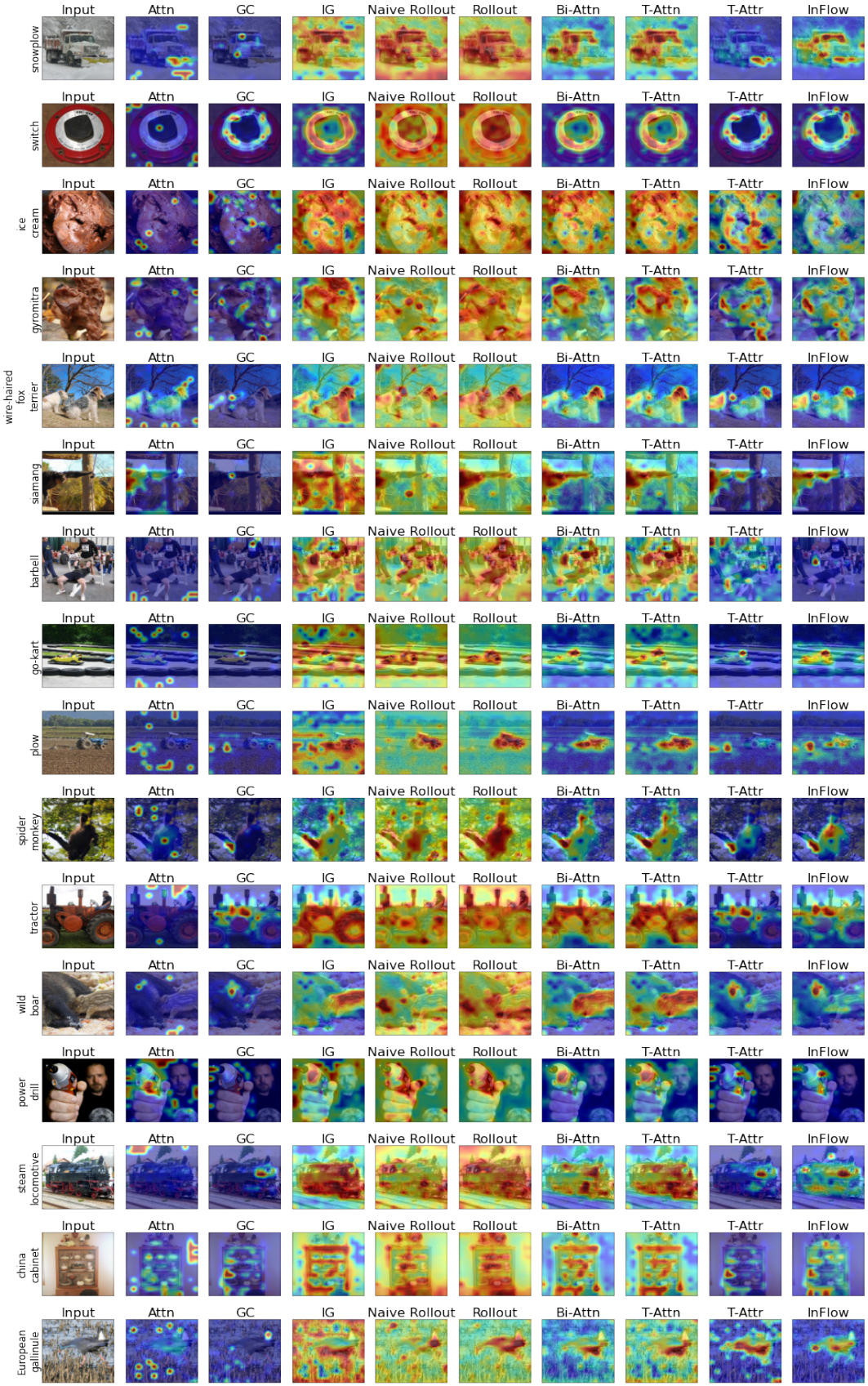
	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
N-Roll (Abnar and Zuidema, 2020a)	0.743	0.807	0.835	0.249	0.586	0.642	0.371	0.272
Rollout (Abnar and Zuidema, 2020a)	0.762	0.825	0.849	0.217	0.632	0.655	0.341	0.314
T-Attn (Yuan et al., 2021a)	0.757	0.816	0.850	0.217	0.633	0.657	0.339	0.317
Bi-Attn (Chen et al., 2023a)	0.772	0.833	**0.856**	0.207	0.649	0.680	0.313	0.367
T-Attr (Chefer et al., 2021b)	0.760	0.819	0.851	0.210	0.641	**0.731**	0.275	0.456
InFlow (ours)	**0.774**	**0.834**	0.852	**0.202**	**0.650**	0.728	**0.267**	**0.461**

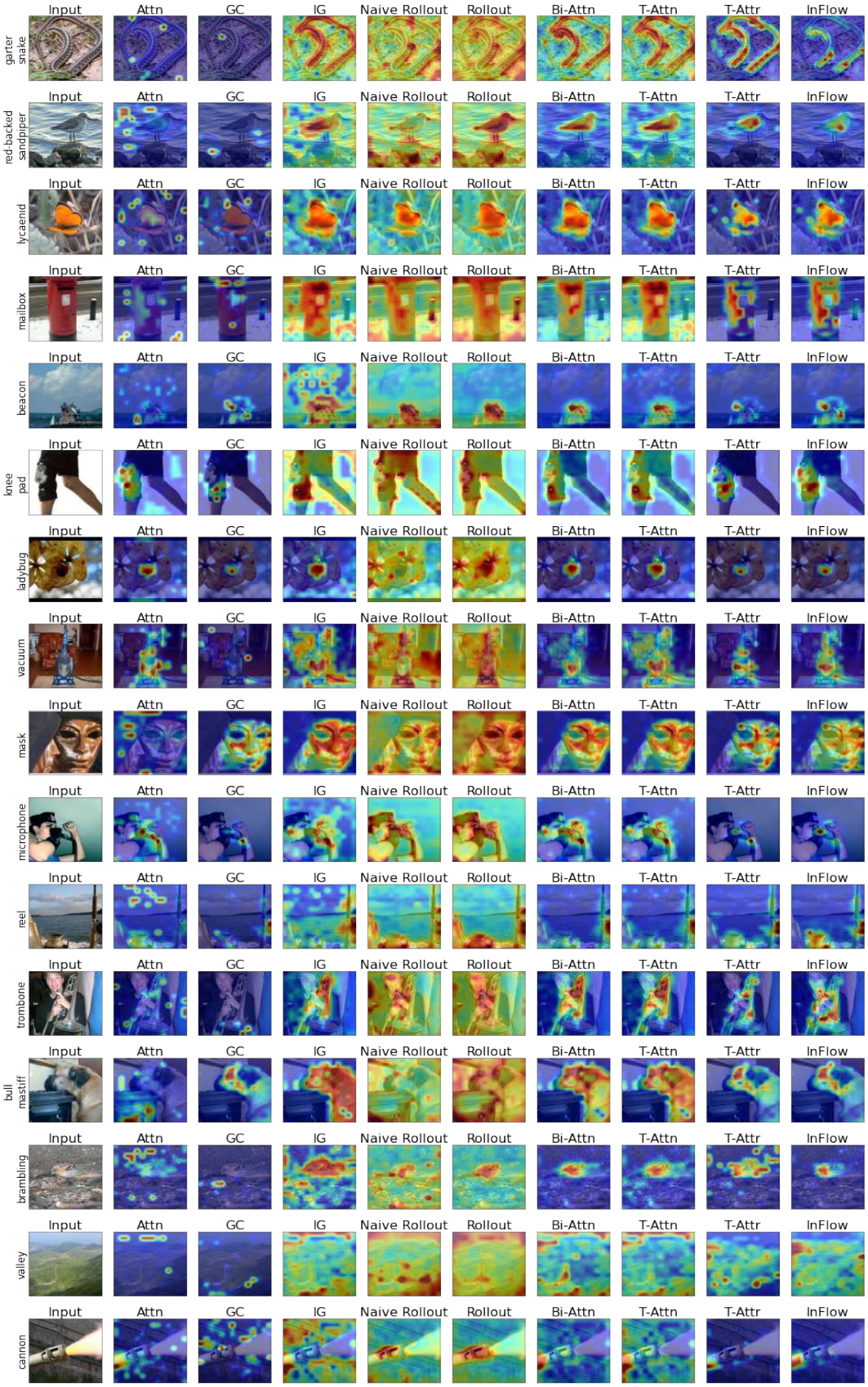
Table 7: A comparison of ViT-base 32×32 attribution methods against InFlow with IG post-processing

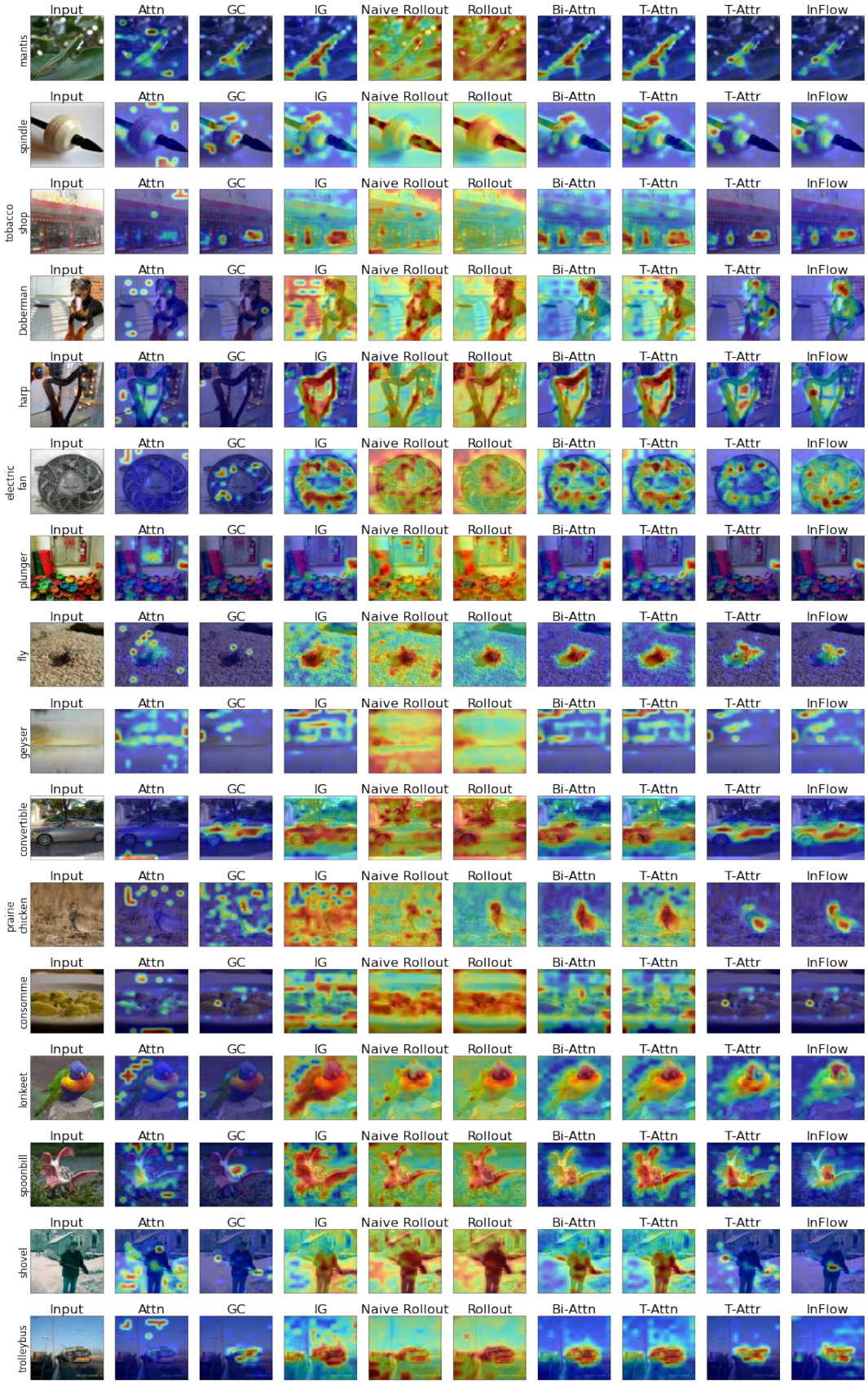
	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
N-Roll (Abnar and Zuidema, 2020a)	0.687	0.740	0.811	0.299	0.512	0.577	0.464	0.114
Rollout (Abnar and Zuidema, 2020a)	0.744	0.788	0.862	0.218	0.645	0.592	0.425	0.167
T-Attn (Yuan et al., 2021a)	0.745	0.790	0.863	0.217	0.646	0.593	0.425	0.169
Bi-Attn (Chen et al., 2023a)	0.763	0.804	**0.881**	0.201	0.680	0.623	0.394	0.229
T-Attr (Chefer et al., 2021b)	**0.767**	0.807	0.871	0.201	0.670	0.681	0.339	0.342
InFlow (ours)	0.765	**0.807**	0.871	**0.190**	**0.681**	**0.691**	**0.316**	**0.375**

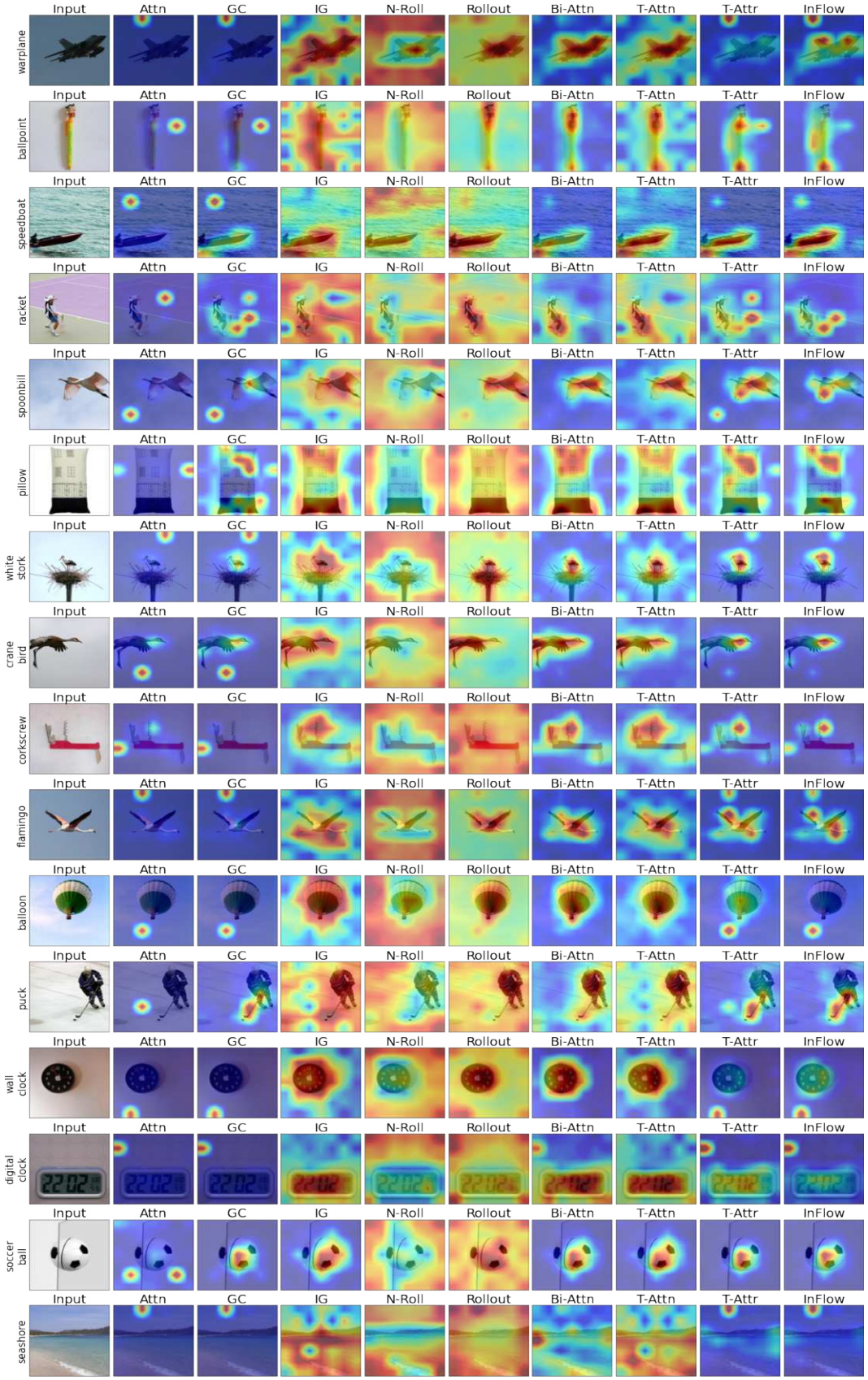
Table 8: A comparison of ViT-tiny 16×16 attribution methods against InFlow with IG post-processing

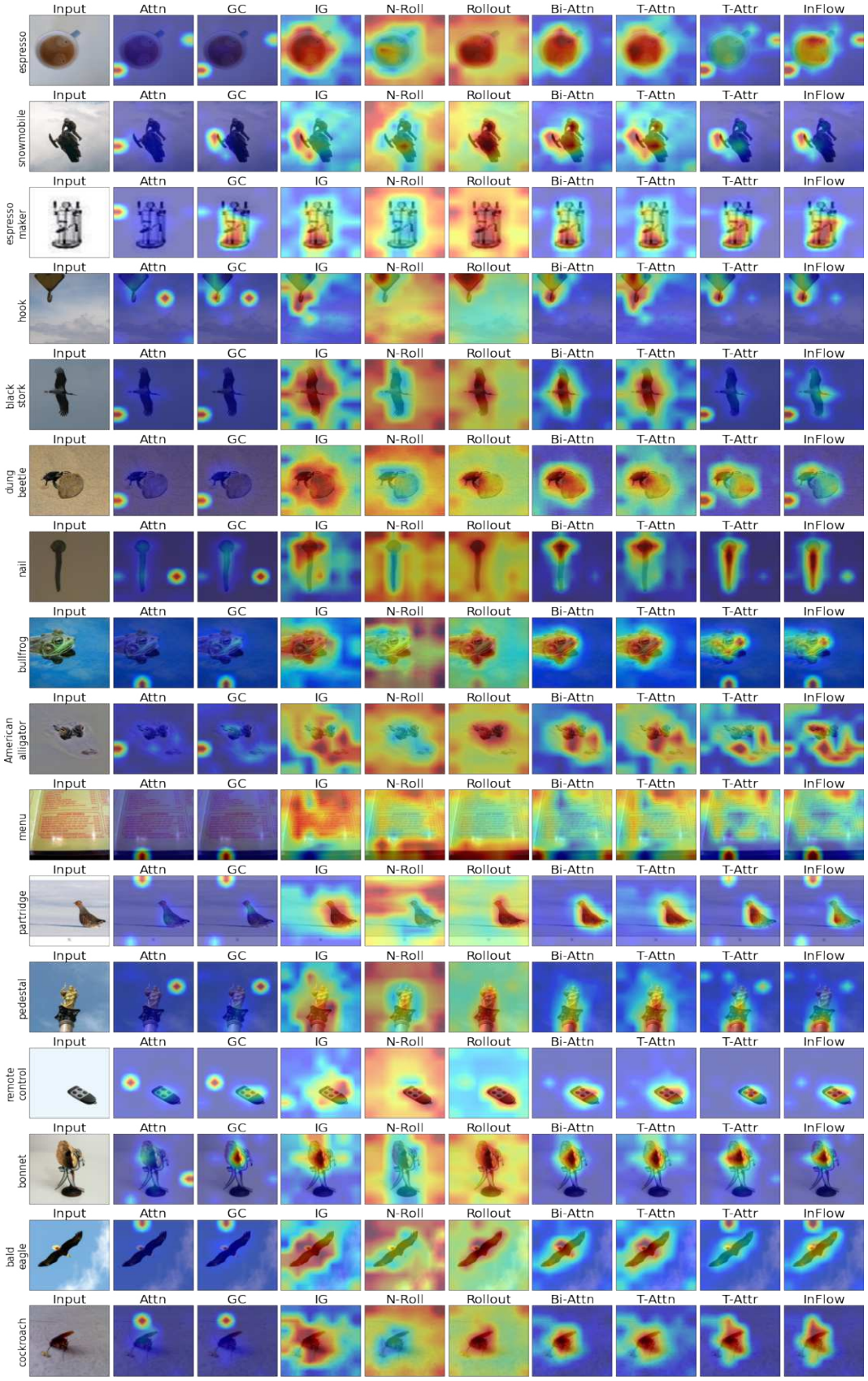
	PIC Tests		RISE Tests			MAS Tests		
Test Type	**SIC (↑)**	**AIC (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**	**Ins (↑)**	**Del (↓)**	**Ins - Del (↑)**
N-Roll (Abnar and Zuidema, 2020a)	0.575	0.641	0.743	0.219	0.524	0.587	0.361	0.226
Rollout (Abnar and Zuidema, 2020a)	0.624	0.688	0.794	0.171	0.622	0.624	0.331	0.293
T-Attn (Yuan et al., 2021a)	0.624	0.689	0.792	0.174	0.618	0.622	0.333	0.290
Bi-Attn (Chen et al., 2023a)	0.680	0.736	0.814	0.157	0.657	0.658	0.301	0.356
T-Attr (Chefer et al., 2021b)	0.639	0.701	0.833	0.131	0.701	0.712	0.238	0.474
InFlow (ours)	**0.690**	**0.746**	**0.841**	**0.120**	**0.721**	**0.727**	**0.215**	**0.513**

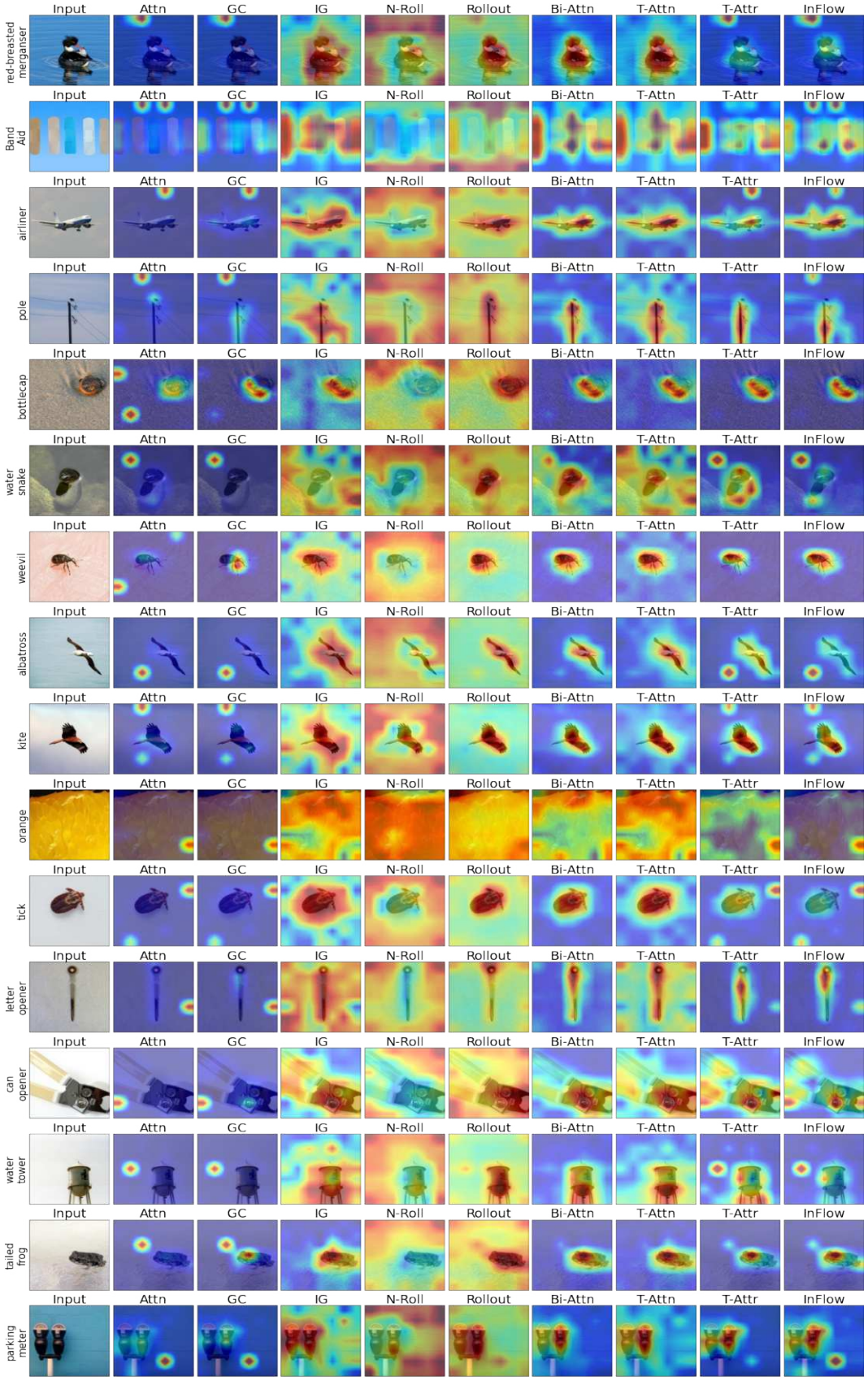

Input
Attn
GC
IG
Naive Rollout
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
snowplow
switch
ice cream
gyromitra
wire-haired fox terrier
siamang
barbell
go-kart
plow
spider monkey
tractor
wild boar
power drill
steam locomotive
china cabinet
European gallinule

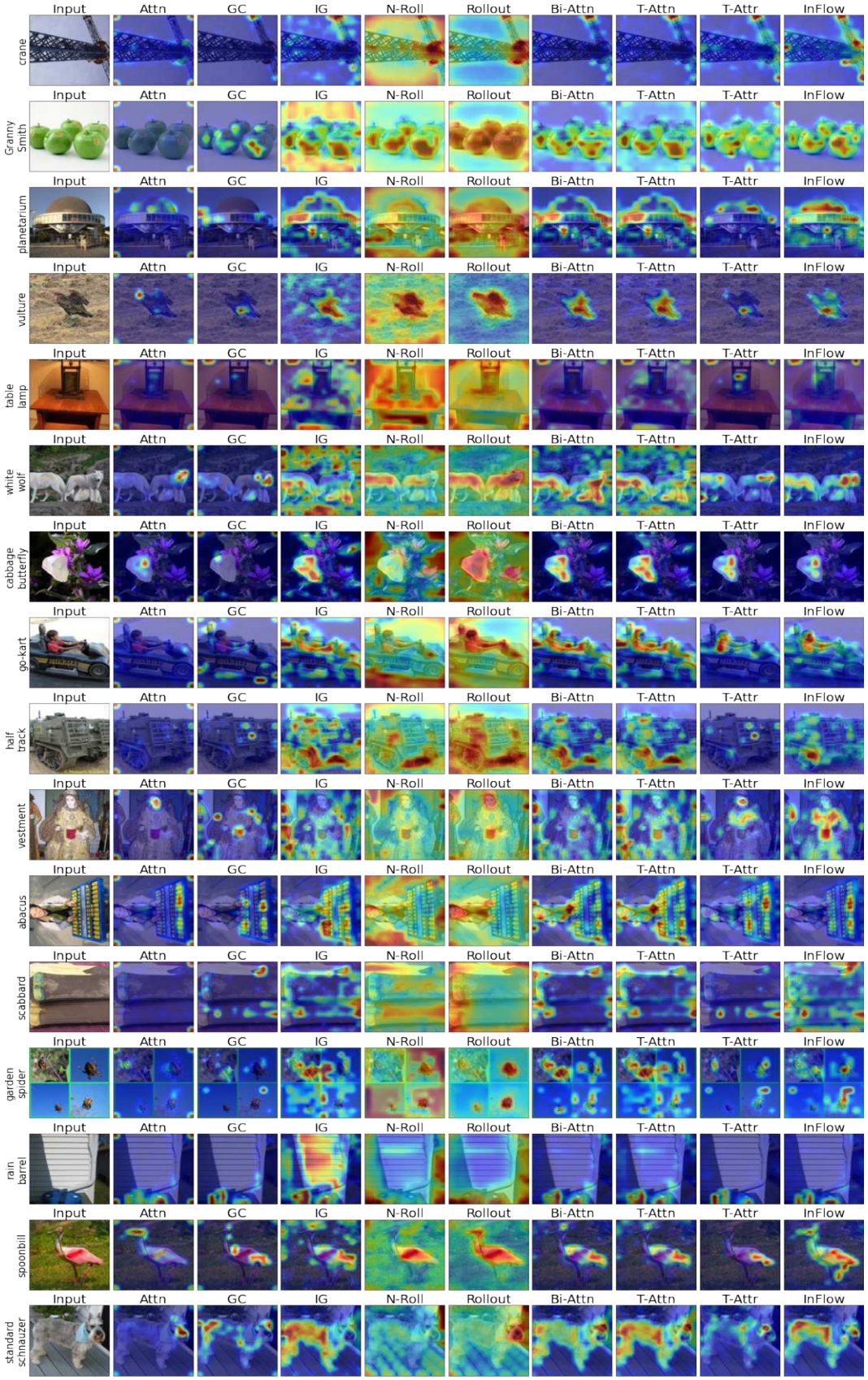

Input
Attn
GC
IG
Naive Rollout
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
garter snake
red-backed sandpiper
lycaenid
mailbox
beacon
knee pad
ladybug
vacuum
mask
microphone
reel
trombone
bull mastiff
brambling
valley
cannon

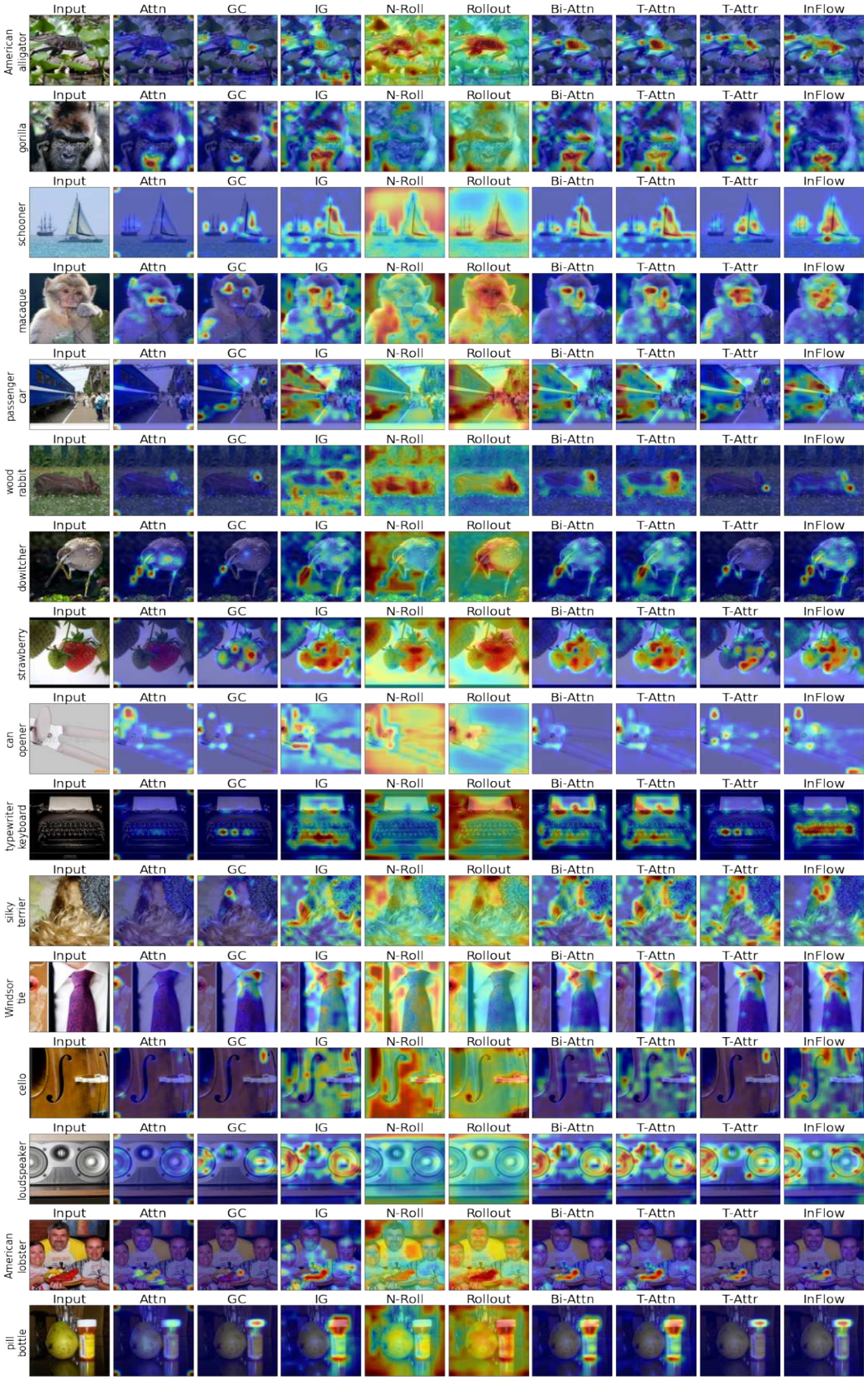

Input
Attn
GC
IG
Naive Rollout
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
mantis
spindle
tobacco shop
Doberman
harp
electric fan
plunger
fly
geyser
convertible
prairie chicken
consomme
lorikeet
spoonbill
shovel
trolleybus

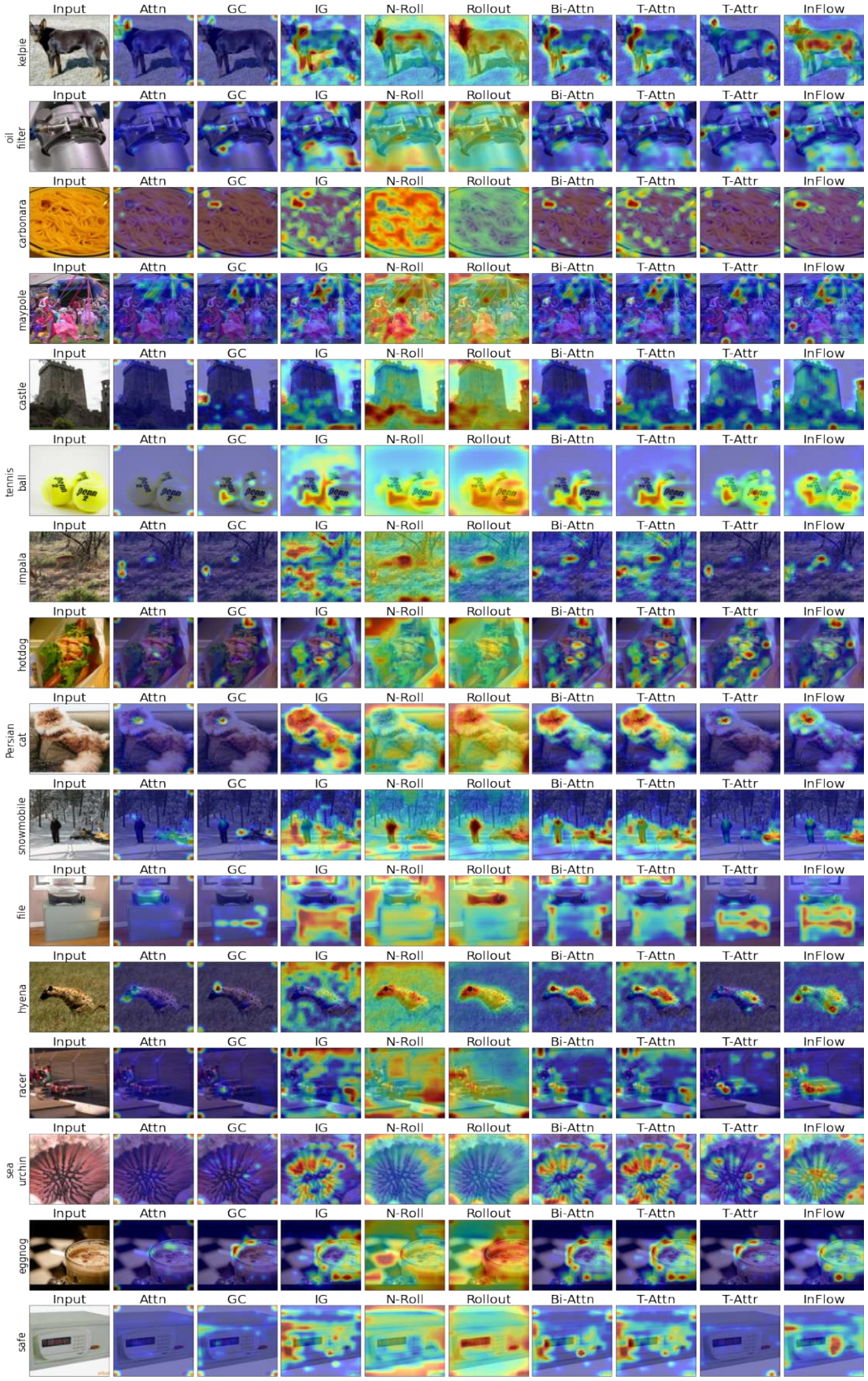

Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
warplane
ballpoint
speedboat
racket
spoonbill
pillow
white stork
crane bird
corkscrew
flamingo
balloon
puck
wall clock
digital clock
soccer ball
seashore


Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
espresso
snowmobile
espresso maker
hook
black stork
dung beetle
nail
bullfrog
American alligator
menu
partridge
pedestal
remote control
bonnet
bald eagle
cockroach


Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
red-breasted merganser
Band Aid
airliner
pole
bottlecap
water snake
weevil
albatross
kite
orange
tick
letter opener
can opener
water tower
tailed frog
parking meter


Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
crane
Granny Smith
planetarium
vulture
table lamp
white wolf
cabbage butterfly
go-kart
half track
vestment
abacus
scabbard
garden spider
rain barrel
spoonbill
standard schnauzer


Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
American alligator
gorilla
schooner
macaque
passenger car
wood rabbit
dowitcher
strawberry
can opener
typewriter keyboard
silky terrier
Windsor tie
cello
loudspeaker
American lobster
pill bottle


Input
Attn
GC
IG
N-Roll
Rollout
Bi-Attn
T-Attn
T-Attr
InFlow
kelpie
oil filter
carbonara
maypole
castle
tennis ball
impala
hotdog
Persian cat
snowmobile
file
hyena
racer
sea urchin
eggnog
safe