1. Introduction

Convolutional neural networks have had tremendous success in computer vision [? ?] but, due to their complexity, it is still an open problem how to explain or interpret their predictions. The most concrete achievement in this direction is to localize in an image what regions a prediction can be attributed to, by means of *saliency maps*. *Post-hoc* interpretability methods do so without changing the network architecture or training process and *class activation mapping* (CAM) [?] has been a milestone in their development. CAM-based saliency maps are expressed as a linear combination of feature maps followed by an activation function and it is the definition of weights that determines different methods [? ? ?].

Vision transformers [?] are now strong competitors of convolutional networks, characterized by global interactions between patch embeddings in the form of *self attention*. Based on a classification (CLS) token, their pooling mechanism allows localization by means of *raw attention* maps. However, these maps are class agnostic, they can be of low quality [?] and dedicated interpretability methods are required [?].

In this work, we make an important connection between CAM and the raw attention map of the CLS token. In particular, self attention is defined on all patch tokens, including CLS. Focusing on the cross attention between CLS and patch token embeddings, this is expressed as a collection of dot product similarities between embeddings, followed by softmax. We show that this collection of similarities is in fact a linear combination of feature maps, where the weights are the elements of the CLS token embedding. Hence, **the raw attention map of the CLS token has the same form as a class agnostic CAM-based saliency map**.

In addition, pooling in vision transformers is defined as a weighted average of patch token embeddings, where the weights are given by the raw attention map of the CLS token. This can be seen as reweighting, or soft masking, of the embeddings before *global average pooling* (GAP). By contrast, pooling in convolutional networks is based on GAP only. We thus observe that **attention-based pooling is a form of masking in the feature space**. Masking, mostly in the input space, is common in interpretability methods [?] and their evaluation [? ?] to establish that a prediction is indeed due to a certain object of interest.

Motivated by the above observations, we design an attention-based pooling mechanism as a replacement for GAP in convolutional networks. Since this mechanism has the form of a CAM-based saliency map followed by masking, we aim to study the effect of this network modification in interpreting the network predictions.

Our pooling mechanism, called *Cross Attention Stream* (*CA-Stream*), is implemented as a stream running in parallel with the backbone network. At different stages of the network, it allows interaction between a CLS token and patch embeddings by means of cross attention. The CLS token embedding is initialized as a learnable parameter and, at the output of the stream, provides a global image representation for classification.

We aim at post-hoc interpretability, therefore we keep the network and classifier frozen while learning the parameters of CA-Stream. We then obtain CAM-based saliency maps by existing post-hoc methods for both GAP and CA-Stream and compare their performance in terms of interpretability metrics as well as classification accuracy.

More specifically, we make the following contributions:

1. We show that attention-based pooling in vision transformers is the same as soft masking by a class agnostic CAM-based saliency map (**??**).
2. We design and inject an attention-based pooling mechanism into convolutional networks to replace GAP and study its effect on post-hoc interpretability (**??**). This is a modification of the network but not of its training process.
3. We show that this mechanism helps explain a trained network by improving the performance of existing post-hoc interpretability methods as well as providing a class agnostic raw attention map (**??**).

2. Related Work

2.1. Interpretability

To open up the black-box behavior of deep neural networks, model interpretability is mainly investigated along two directions [? ? ?]:

1. *Post-hoc interpretability* considers the model as a black-box and provides explanations based on inputs and outputs, without modifying the model or its training process.
2. *Transparency* modifies the model or the training process to better explain the behavior of the inner parts of the model.

Post-hoc interpretability Approaches can be grouped into a number of possibly overlapping categories. *Gradient-based methods* [? ? ? ? ? ? ?] use the gradient of a target class score with respect to the input to compute the contribution of different input regions to the prediction. *CAM-based methods* [? ? ? ? ? ?] compute saliency maps as a linear combination of feature maps, with different definitions for the weights. *Occlusion or masking-based methods* [? ? ? ? ?] apply a number of candidate masks in the input space, measure their effect on the prediction and then combine them into a saliency map. Masking in feature space has been explored too [?]. Finally, *learning-based methods* [? ? ? ? ?] learn an additional network or branch on extra data to produce an explanation map for a given input.

Our method shares similarities with learning-based methods. In particular, we train an additional branch on the same training data as the network we aim to explain, but we only use the standard classification loss and we do not provide any explanation as output. Our method is also similar to CAM-based and masking-based methods exactly because we show that attention-based pooling is the same as masking in the feature space with weights obtained by a CAM-based saliency map.

Transparency Approaches are grouped in a number of categories according to the type of the given explanation. *Rule-based methods* [? ?] train a decision tree as a surrogate regularization term to force a network to be easily approximated by a decision tree. *Hidden semantics-based methods* [? ? ? ?] aim to make a convolutional network learn disentangled hidden semantics with hierarchical structure or object-level concepts. *Prototype-based methods* [? ?] learn a set of prototypes or parts as an intermediate representation in the network, which can be aligned with categories. *Attribution-based methods* [? ? ? ?] usually modify the architecture of a network or the training process to help post-hoc methods produce better saliency maps. Unlike [? ?], saliency guided localization [?] does not need ground truth explanations but replaces them with information bottleneck attribution [?]. Finally, saliency-guided training [?] minimizes the KL divergence between the output of original and masked images.

Our method belongs to attribution-based methods. We introduce a learnable cross-attention stream into the network as a pooling mechanism to replace GAP. As a result, post-hoc attribution-based methods can provide a better explanation. Although the network is modified, the training process is not: the network and classifier are pretrained and kept frozen while we learn the paramaters of our stream.

2.2. Attention-based architectures

Attention is a powerful mechanism that has been introduced into convolutional networks in several ocaasions [? ? ?]. With the success of vision transformers (ViT) [?], fully attention-based architectures are now competitive with convolutional networks. To benefit from both self-attention and convolutional layers, some hybrid architectures employ convolutional layers before the vision transformer [? ?]. Others, such as Swin [?] and PiT [?], introduce a pyramid structure to share local spatial information while reducing the spatial resolution, as in convolutional networks.

Conformer [?] proposes a dual network structure to retrain and fuse local convolutional features with global representations. Our method merely provides a simple attention-based pooling mechanism inspired by transformers that can work with any architecture, even pretrained and frozen. SCOUTER [?] uses slot attention [?] to build a class specific pooling mechanism, which does not scale well to more than 20 classes. Our mechanism is based standard cross attention, it is class agnostic and scales up to 1000 classes, improving post-hoc interpretability without degrading classification accuracy. PatchConvNet [?] replaces global average pooling by an attention-based pooling layer. Our method is similar in this respect, but uses information collected from the entire network by a parallel processing stream. We introduce our mechanism into any convolutional network and study its effect on post-hoc interpretability.

3. Method

3.1. Preliminaries and background

Notation Let $f : \mathcal{X} \to \mathbb{R}^C$ be a classifier network that maps an input image $\mathbf{x} \in \mathcal{X}$ to a logit vector $\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^C$, where $\mathcal{X}$ is the image space and C is the number of classes. A class probability vector is obtained by $\mathbf{p} = \operatorname{softmax}(\mathbf{y})$. The logit and probability of class c are respectively denoted by y^c and $p^c = \operatorname{softmax}(\mathbf{y})^c := e^{y^c} / \sum_j e^{y^j}$. Let $\mathbf{F}_\ell \in \mathbb{R}^{w_\ell \times h_\ell \times d_\ell}$ be the feature tensor at layer ℓ of the network, where $w_\ell \times h_\ell$ is the spatial resolution and d_ℓ the embedding dimension, or number of channels. The feature map of channel k is denoted by $F_\ell^k \in \mathbb{R}^{w_\ell \times h_\ell}$. By ℓ we may refer to an arbitrary layer of f or a larger compositional block, *e.g.*, a stage.

CAM-based saliency maps Given a class of interest c and a layer ℓ, we consider the saliency maps $S_\ell^c \in \mathbb{R}^{w_\ell \times h_\ell}$ given by the general formula

$$S_\ell^c := h\left(\sum_k \alpha_k^c F_\ell^k\right), \tag{1}$$

where α_k^c are weights defining a linear combination over channels and h is an activation function. Assuming *global average pooling* (GAP) of the last feature tensor $\mathbf{F}_L$ followed by a linear classifier, CAM [?] is defined for the last layer L only, with h being the identity mapping and α_k^c the classifier weight connecting channel k with class c. Grad-CAM [?] is a generalization of CAM defined for any architecture and layer ℓ, with $h = \operatorname{ReLU}$ and weights

$$\alpha_k^c := \operatorname{GAP}\left(\frac{\partial y^c}{\partial F_\ell^k}\right). \tag{2}$$

Self-Attention Let $X_\ell \in \mathbb{R}^{t_\ell \times d_\ell}$ denote the sequence of token embeddings of a vision transformer [?] at layer ℓ, where $t_\ell := w_\ell h_\ell + 1$ is the number of tokens, including patch tokens and the CLS token, and d_ℓ is the embedding dimension. The *query*, *key* and *value* matrices are defined as $Q = X_\ell W_Q$, $K = X_\ell W_K$, $V = X_\ell W_V \in \mathbb{R}^{t_\ell \times d_\ell}$, where $W_Q, W_K, W_V \in \mathbb{R}^{d_\ell \times d_\ell}$ are learnable linear projections. The *attention matrix* $A \in \mathbb{R}^{t_\ell \times t_\ell}$ expresses pairwise dot-product similarities between queries (rows of Q) and keys

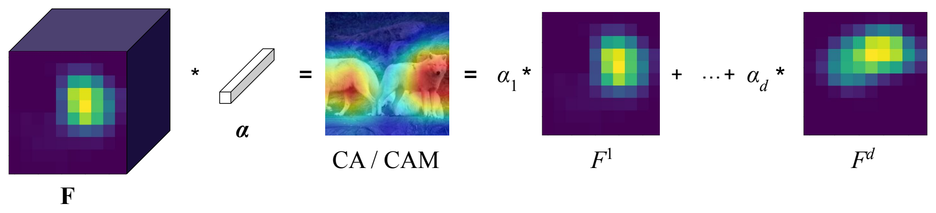

Figure 1. *Visualization of eq. (??).* On the left, a feature tensor $\mathbf{F} \in \mathbb{R}^{w \times h \times d}$ is multiplied by the vector $\boldsymbol{\alpha} \in \mathbb{R}^d$ in the channel dimension, like in 1×1 convolution, where $w \times h$ is the spatial resolution and d is the number of channels. This is *cross attention* (CA) [?] between the query $\boldsymbol{\alpha}$ and the key $\mathbf{F}$. On the right, a linear combination of feature maps $F^1, \dots, F^d \in \mathbb{R}^{w \times h}$ is taken with weights $\alpha_1, \dots, \alpha_d$. This is a *class activation mapping* (CAM) [?] with class agnostic weights. Eq. (??) expresses the fact that these two quantities are the same, provided that $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d)$ and $\mathbf{F}$ is reshaped as $F = (\mathbf{f}^1 \dots \mathbf{f}^d) \in \mathbb{R}^{p \times d}$, where $p = wh$ and $\mathbf{f}^k = \text{vec}(F^k) \in \mathbb{R}^p$ is the vectorized feature map of channel k.

(rows of K), normalized by softmax over rows

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_\ell}}\right). \tag{3}$$

For each token, the *self-attention* operation is then defined as an average of all values (rows of V) weighted by attention (the corresponding row of A),

$$\text{SA}(X_\ell) := AV \in \mathbb{R}^{t_\ell \times d_\ell}. \tag{4}$$

At the last layer L, the CLS token embedding is used as a global image representation for classification as it gathers information from all patches by weighted averaging, replacing GAP. Thus, at the last layer, it is only cross attention between CLS and the patch tokens that matters.

3.2. Motivation

Cross attention Let matrix $F_\ell \in \mathbb{R}^{p_\ell \times d_\ell}$ be a reshaping of feature tensor $\mathbf{F}_\ell$ at layer ℓ, where $p_\ell := w_\ell h_\ell$ is the number of patch tokens without CLS, and let $\mathbf{q}_\ell \in \mathbb{R}^{d_\ell}$ be the CLS token embedding at layer ℓ. By focusing on the *cross attention* only between the CLS (query) token $\mathbf{q}_\ell$ and the patch (key) tokens F_ℓ and by ignoring projections W_Q, W_K, W_V for simplicity, attention A (??) is now a $1 \times p_\ell$ matrix that can be written as a vector $\mathbf{a} \in \mathbb{R}^{p_\ell}$

$$\mathbf{a} = A^\top = \text{softmax}\left(\frac{F_\ell \mathbf{q}_\ell}{\sqrt{d_\ell}}\right). \tag{5}$$

Here, $F_\ell \mathbf{q}_\ell$ expresses the pairwise similarities between the global CLS feature $\mathbf{q}_\ell$ and the local patch features F_ℓ. Now, by replacing $\mathbf{q}_\ell$ by an arbitrary vector $\boldsymbol{\alpha} \in \mathbb{R}^{d_\ell}$ and by writing the feature matrix as $F_\ell = (\mathbf{f}_\ell^1 \dots \mathbf{f}_\ell^{d_\ell})$ where $\mathbf{f}_\ell^k = \text{vec}(F_\ell^k) \in \mathbb{R}^{p_\ell}$ for channel k, attention (??) becomes

$$\mathbf{a} = h_\ell(F_\ell \boldsymbol{\alpha}) = h_\ell\left(\sum_k \alpha_k \mathbf{f}_\ell^k\right). \tag{6}$$

This takes the same form as (??), with feature maps F_ℓ^k being vectorized into $\mathbf{f}_\ell^k$ and the activation function is defined as $h_\ell(\mathbf{x}) = \text{softmax}(\mathbf{x}/\sqrt{d_\ell})$. Eq. (??) is visualized in ??. We thus observe the following.

Pairwise similarities between one query and all patch token embeddings in cross attention are the same as a linear combination of feature maps in CAM-based saliency maps, where the weights are determined by the elements of the query.

As it stands, one difference between (??) and (??) is that (??) is class agnostic, although it could be extended by using one query (weight) vector per class. For simplicity, we choose the class agnostic form in the following. We also choose to have no query/key projections. However, we do provide additional experiments with class specific representation as well as projections in ??.

Pooling, or masking We are thus motivated to integrate an attention mechanism into any network such that making a prediction and explaining (localizing) it are inherently connected. In particular, considering cross attention only between CLS and patch tokens (??), equation (??) becomes

$$\text{CA}_\ell(\mathbf{q}_\ell, F_\ell) := F_\ell^\top \mathbf{a} = F_\ell^\top h_\ell(F_\ell \mathbf{q}_\ell) \in \mathbb{R}^{d_\ell}. \tag{7}$$

By writing the transpose of feature matrix as $F_\ell^\top = (\boldsymbol{\phi}_\ell^1 \dots \boldsymbol{\phi}_\ell^{p_\ell})$ where $\boldsymbol{\phi}_\ell^i \in \mathbb{R}^{d_\ell}$ is the feature of patch i, this is a weighted average of the local patch features $F_\ell^\top$ with attention vector $\mathbf{a} = (a_1, \dots, a_{p_\ell})$ expressing the weights:

$$\text{CA}_\ell(\mathbf{q}_\ell, F_\ell) := F_\ell^\top \mathbf{a} = \sum_i a_i \boldsymbol{\phi}_\ell^i. \tag{8}$$

We can think of it as as feature *reweighting* or *soft masking* in the feature space, followed by GAP.

Now, considering that $\mathbf{a}$ is obtained exactly as CAM-based saliency maps (??), this operation is similar to occlusion (masking)-based methods [? ? ? ? ? ? ?] and evaluation metrics [? ?], where a CAM-based saliency map is commonly used to mask the input image. We thus observe the following.

Attention-based pooling is a form of feature reweighting or soft masking in the feature space followed by GAP*, where the weights are given by a class agnostic CAM-based saliency map.*

3.3. Cross attention stream

Motivated by the observations above, we design a *Cross Attention Stream* (*CA-Stream*) in parallel to any network. It takes input features at key locations of the network and uses cross attention to build a global image representation and replace GAP before the classifier. An example is shown in **??**, applied to a ResNet-based architecture.

Architecture More formally, given a network f, we consider points between blocks of f where critical operations take place, such as change of spatial resolution or embedding dimension, *e.g.* between stages for ResNet. We decompose f at these points as

$$f = g \circ \text{GAP} \circ f_L \circ \cdots \circ f_0 \tag{9}$$

such that features $F_\ell \in \mathbb{R}^{p_\ell \times d_\ell}$ of layer (stage) ℓ are initialized as $F_{-1} = \mathbf{x}$ and updated according to

$$F_\ell = f_\ell(F_{\ell-1}) \tag{10}$$

for $0 \le \ell \le L$. The last layer features F_L are followed by GAP and $g : \mathbb{R}^{d_L} \to \mathbb{R}^C$ is the classifier, mapping to the logit vector $\mathbf{y}$. As in **??**, p_ℓ is the number of patch tokens and d_ℓ the embedding dimension of stage ℓ.

In parallel, we initialize a classification token embedding as a learnable parameter $\mathbf{q}_0 \in \mathbb{R}^{d_0}$ and we build a sequence of updated embeddings $\mathbf{q}_\ell \in \mathbb{R}^{d_\ell}$ along a stream that interacts with F_ℓ at each stage ℓ. Referring to the global representation $\mathbf{q}_\ell$ as *query* or CLS and to the local image features F_ℓ as *key* or patch embeddings, the interaction consists of cross attention followed by a linear projection $W_\ell \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ to account for changes of embedding dimension between the corresponding stages of f:

$$\mathbf{q}_{\ell+1} = W_\ell \cdot \text{CA}_\ell(\mathbf{q}_\ell, F_\ell), \tag{11}$$

for $0 \le \ell \le L$, where CA_ℓ is defined as in (**??**).

Image features $F_0, \dots, F_L$ do not change by injecting our CA-Stream into network f. However, the final global image representation and hence the prediction do change. In particular, at the last stage L, $\mathbf{q}_{L+1}$ is used as a global image representation for classification, replacing GAP over F_L. The final prediction is $g(\mathbf{q}_{L+1}) \in \mathbb{R}^C$. Unlike GAP, the weights of different image patches in the linear combination are non-uniform, enhancing the contribution of relevant patches in the prediction.

Training In this sense, the network f is pretrained and remains frozen while we learn the parameters of our CA-Stream on the same training set as one used to train f. The classifier is kept frozen too. Referring to (**??**), $f_0, \dots, f_L$ and g are fixed, while GAP is replaced by learned weighted averaging, with the weights obtained by the CA-Stream.

Inference As it stands, CA-Stream is not an interpretability method, but rather a modification of the baseline architecture, *i.e.*, an attention-based pooling mechanism that replaces GAP to enhance the contribution of relevant image regions in the prediction. We are interested in investigating the interpretability properties of this modification. We therefore employ existing post-hoc, CAM-based interpretability methods to generate saliency maps with both baseline GAP and CA-Stream. We then compare interpretability metrics as well as classification accuracy.

4. Experiments

We evaluate the interpretability and recognition capabilities of our approach. In particular, we generate explanations following current state-of-the art post-hoc interpretability methods derived from CAM [?]. We compare the properties of the backbone network f with and without our CA-Stream, where f is pretrained and fixed.

4.1. Experimental setup

Training We train and evaluate our models on the ImageNet ILSVRC-2012 dataset [?], on the training and validation splits respectively. Thus, we experiment with ResNet-based architectures [?] such as ResNet-18 and ResNet-50, and ConvNeXt based architectures [?] such as ConvNeXt-Small and ConvNeXt-Base. We aim at learning our CA-Stream, generating a CLS token that interacts with feature maps at different stages of network f, to serve as an attention-based pooling mechanism in order to interpret the predictions of f. Therefore, we experiment with pretrained models[1], that we keep frozen while the parameters of the CA-Stream are optimized. Details on training hyperparameters are given in the appendix.

Moreover, we present experiments on the bird dataset: CUB-200-2011 [?] and on PASCAL VOC 2012 dataset [?]. Here the ResNet-50 network is fine-tuned to these dataset as baseline. Then, our CA-Stream is learned as for ImageNet.

Evaluation We employ existing post-hoc interpretability methods to generate saliency maps with and without CA-Stream and compare interpretability metrics as well as classification accuracy. Regarding interpretability methods, we use Grad-CAM [?], Grad-CAM++ [?] and ScoreCAM [?]. We note that the evaluation is performed on the entire validation set, unlike the previous approaches.

Following Opti-CAM [?], we use a number of classification metrics for interpretability. In particular, we consider

[1]https://pytorch.org/vision/0.8/models.html

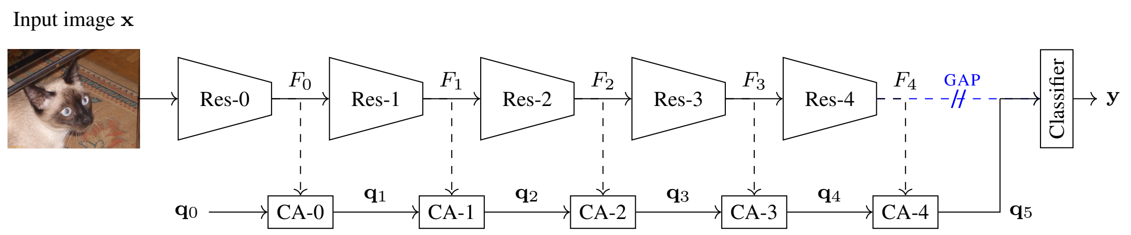

Figure 2. *Cross Attention Stream (CA-Stream) applied to ResNet-based architectures.* Given a network f, we replace global average pooling (GAP) by a learned, attention-based pooling mechanism implemented as a stream in parallel to f. The feature tensor $F_\ell \in \mathbb{R}^{p_\ell \times d_\ell}$ (*key*) obtained by stage Res-ℓ of f interacts with a CLS token (*query*) embedding $\mathbf{q}_\ell \in \mathbb{R}^{d_\ell}$ in block CA-ℓ, which contains cross attention (**??**) followed by a linear projection (**??**) to adapt to the dimension of $F_{\ell+1}$. Here, p_ℓ is the number of patches (spatial resolution) and d_ℓ the embedding dimension. The query is initialized by a learnable parameter $\mathbf{q}_0 \in \mathbb{R}^{d_0}$, while the output $\mathbf{q}_5$ of the last cross attention block is used as a global image representation into the classifier. The network and classifier are pretrained and kept frozen while the parameters of CA-Stream are learned. At inference, we use existing post-hoc interpretability methods like Grad-CAM [**?**] to obtain saliency maps for both the baseline GAP and our CA-Stream. We compare interpretability metrics as well as accuracy.

the changes in predictive power measured by *average drop* (AD) [**?**] and *average gain* (AG) [**?**], the proportion of better explanations measured by *average increase* (AI) [**?**] and the impact of different extent of masking measured by *insertion* (I) and *deletion* (D) [**?**].

4.2. Qualitative evaluation

We show saliency maps obtained by different interpretability methods using either GAP or CA-Stream, as well as the class-agnostic raw attention coming from our CA-Stream, see **??**.

We observe that the raw attention focuses on objects of interest in the images. In general, saliency maps obtained with CA-Stream are similar but tend to cover larger regions of the object or more instances compared with GAP.Indeed, the differences in saliency maps should not be large, as both methods share the same features maps F_ℓ^k and only the weight coefficients α_k^c differ. Despite the small differences, the following quantitative results show that CA-Stream has a significant impact on the interpretability metrics.

In addition, **??** shows examples of images from the MIT 67 Scenes dataset [**?**] along with raw attention maps obtained by CA-Stream. These images come from four classes that do not exist in ImageNet and the network sees them at inference for the first time. Nevertheless, the attention maps focus on objects of interest in general.

4.3. Interpretabity metrics

Here we measure the effect of employing our CA-Stream approach to pool features *vs*. the baseline GAP on the faithfulness of explanations, using classification metrics for interpretability. Results are reported in **??** for ImageNet and **??** for CUB and Pascal VOC.

?? shows that for different networks, CAM-based interpretability methods and dataset, CA-Stream provides consistent improvements over GAP in terms of AD, AG, AI and I metrics, while performing lower on D. Deletion has raised concerns in previous works [**? ?**]. Indeed, it gradually replaces pixels by black, unlike insertion which starts from a blurred image. This poses the problem of *out-of-distribution* (OOD) data [**? ? ?**], possibly introducing bias related to the shape of black regions [**?**]. Moreover, non-spread saliency maps tend to perform better [**?**], which is likely the reason for lower performance.

Results on CUB in **??** show that our CA-Stream consistently provides improvements when the model is finetuned on a smaller fine-grained dataset.

Regarding Pascal VOC, the results for Score-CAM are similar to the ones on ImageNet and CUB, with consistent improvements on all metrics but Deletion. However, Grad-CAM and Grad-CAM++ only provide improvements on Average Gain and Average Increase. Average Drop, Insertion and Deletion are very similar. In fact, Pascal VOC is a multi-class dataset and our CA-Stream is class agnostic. Thus, the attention-based pooling is the same for different class for a given image, which reduces the benefit of our CA-Stream.

It is also interesting to observe the performance of Score-CAM, as it computes channel weights α_k^c in (**??**) without using gradients. In gradient-based methods, channel weights are modified by CA-Stream due to modified backward gradient flow to features through cross attention blocks rather than GAP. In Score-CAM however, channel weights are only modified in the forward class probabilities computation, due to attention.

4.4. Classification accuracy

Classification accuracy, number of parameters and GFLOPs for both our CA-Stream and the baselines are reported in **??** (top part).

By adding our CA-Stream to the network, classification remains on par with the baseline. Importantly, the network including the classifier remains frozen and the features used for the global image representation remain fixed,

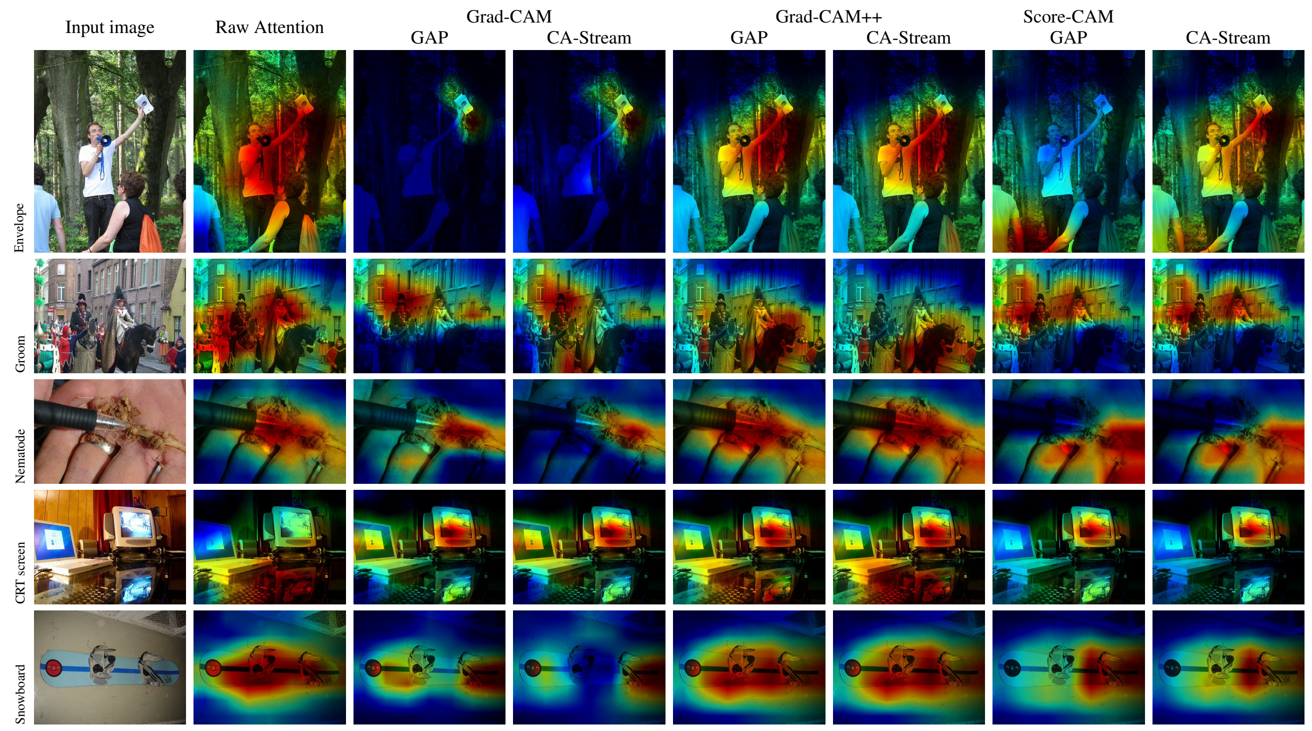

Figure 3. Comparison of saliency maps generated by different CAM-based methods, using GAP and our CA-Stream, on ImageNet images. The raw attention is the one used for pooling by CA-Stream.

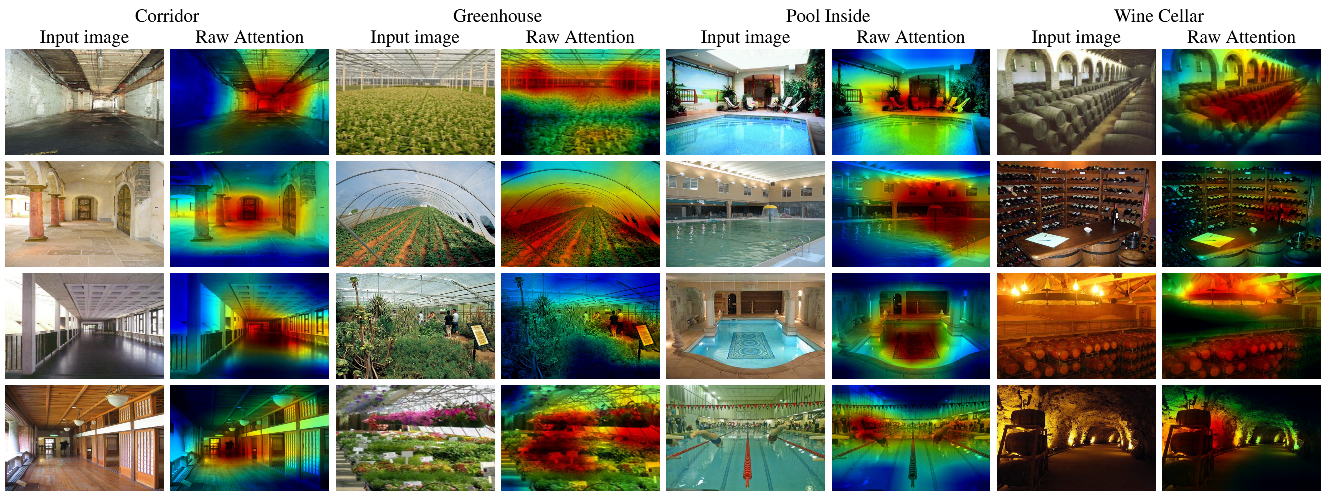

Figure 4. Raw attention maps obtained from our CA-Stream on images of the MIT 67 Scenes dataset [?] on classes that do not exist in ImageNet. The network sees them at inference for the first time.

meaning that any change in accuracy is due to the attention-based pooling mechanism. We further report the number of GFLOPs for one forward pass and the parameters count of both methods. Our CA-Stream has little computation cost and the parameter overhead depends on the embedding dimension because of projection W_ℓ in (**??**) and is small in general, except for ResNet-50. Thus, with small overhead in resources, CA-Stream achieves superior explanations of the classifier predictions, while maintaining accuracy.

4.5. Ablation

We conduct ablation experiments on ResNet50 because of its modularity and ease of modification. We investigate the effect of the cross attention block design, the placement of the CA-Stream relative to the backbone network. The appendix further includes an extra experiments regarding

Accuracy and Parameters					
Network	Pool	GFLOPs	#Param	Param%	Acc↑
ResNet-18	GAP	3.648	11.69M	3.71	67.28
	CA	3.652	12.13M		67.54
ResNet-50	GAP	8.268	25.56M	27.27	74.55
	CA	8.288	32.53M		74.70
ConvNeXt-S	GAP	17.395	50.22M	1.95	83.26
	CA	17.400	51.20M		83.14
ConvNeXt-B	GAP	30.747	88.59M	1.96	83.72
	CA	30.753	90.33M		83.51

Interpretability Metrics							
Network	Method	Pool	AD↓	AG↑	AI↑	I↑	D↓
ResNet-18	Grad-CAM	GAP	17.64	12.73	41.21	63.13	**10.66**
		CA	**16.99**	**17.22**	**44.95**	**65.94**	10.68
	Grad-CAM++	GAP	19.05	11.16	37.99	62.80	**10.75**
		CA	**19.02**	**14.76**	**40.82**	**65.53**	10.82
	Score-CAM	GAP	13.64	12.98	44.53	62.56	**11.37**
		CA	**11.53**	**18.12**	**50.32**	**65.33**	11.51
ResNet-50	Grad-CAM	GAP	13.04	17.56	44.47	72.57	**13.24**
		CA	**12.54**	**22.67**	**48.56**	**75.53**	13.50
	Grad-CAM++	GAP	**13.79**	15.87	42.08	72.32	**13.33**
		CA	13.99	**19.29**	**44.60**	**75.21**	13.78
	Score-CAM	GAP	8.83	17.97	48.46	71.99	**14.31**
		CA	**7.09**	**23.65**	**54.20**	**74.91**	14.68
ConvNeXt-S	Grad-CAM	GAP	42.99	1.69	12.60	48.42	**30.12**
		CA	**22.09**	**14.91**	**32.65**	**84.82**	43.02
	Grad-CAM++	GAP	56.42	1.32	10.35	48.28	**33.41**
		CA	**51.87**	**9.40**	**20.55**	**84.28**	52.58
	Score-CAM	GAP	74.79	1.29	10.10	47.40	**38.21**
		CA	**64.21**	**8.81**	**18.96**	**82.92**	57.46
ConvNeXt-B	Grad-CAM	GAP	33.72	2.43	15.25	52.85	**29.57**
		CA	**19.45**	**13.96**	**32.89**	**86.38**	45.29
	Grad-CAM++	GAP	**34.01**	2.37	15.60	52.83	**29.17**
		CA	36.69	**8.00**	**21.95**	**85.39**	53.42
	Score-CAM	GAP	43.55	2.23	15.67	50.96	**39.49**
		CA	**23.51**	**11.04**	**27.35**	**83.41**	60.53

Table 1. *Accuracy, parameters and interpretability metrics* of CA-Stream *vs*. baseline GAP for different networks and interpretability methods on ImageNet. #Param: total parameters; Param%: percentage of CA-Stream parameters relative to backbone.

CUB-200-2011 - ResNet-50		
	Pooling	Acc↑
	GAP	76.96
	CA	75.90

Interpretability Metrics						
Method	Pooling	AD↓	AG↑	AI↑	I↑	D↓
Grad-CAM	GAP	10.87	10.29	45.81	65.71	**6.17**
	CA	**10.44**	**17.61**	**53.54**	**74.60**	6.56
Grad-CAM++	GAP	11.35	9.68	44.32	65.64	**5.92**
	CA	**11.01**	**16.50**	**51.63**	**74.64**	6.21
Score-CAM	GAP	9.05	10.62	48.90	65.58	5.94
	CA	**6.37**	**19.50**	**60.41**	**74.22**	**2.14**

Pascal VOC 2012 - ResNet-50		
	Pooling	mAP↑
	GAP	78.32
	CA	78.35

Interpretability Metrics						
Method	Pooling	AD↓	AG↑	AI↑	I↑	D↓
Grad-CAM	GAP	**12.61**	9.68	27.88	**89.10**	59.39
	CA	12.77	**15.46**	**34.53**	88.53	**59.16**
Grad-CAM++	GAP	**12.25**	9.68	27.62	**89.34**	54.23
	CA	12.28	**16.76**	**34.87**	89.02	**53.34**
Score-CAM	GAP	14.8	6.76	36.41	71.10	**39.95**
	CA	**10.96**	**21.35**	**43.82**	**89.21**	51.44

Table 2. Accuracy, respectively mean Average Precision, and interpretability metrics of CA-Stream *vs*. baseline GAP for ResNet-50 on CUB and Pascal dataset.

Block Type	#Params	Accuracy
CA	6.96M	74.70
proj→CA	18.13M	74.41

Table 3. *Different cross attention block design for CA-Stream.* Classification accuracy and parameters using ResNet-50 on ImageNet. #Param: parameters of CA-Stream only.

class agnostic *vs*. class specific representations.

Cross attention block design Following transformers [? ?], it is possible to add more layers in the cross attention block. We consider a variant referred to as proj→CA, which uses linear projections $W_\ell^K, W_\ell^V \in \mathbb{R}^{d_\ell \times d_\ell}$ on the key and value

$$\textsc{ca}_\ell(\mathbf{q}_\ell, F_\ell) := (F_\ell W_\ell^V)^\top h_\ell(F_\ell W_\ell^K \mathbf{q}_\ell) \in \mathbb{R}^{d_\ell}, \quad (12)$$

while equation (**??**) remains.

Results are reported in **??**. We observe that the stream made of vanilla CA blocks (**??**) offers slightly better accuracy than projections (**??**), while having less parameters. We also note that most of the computation takes place in the last residual stages, where the channel dimension is the largest. To keep our design simple, we choose the vanilla solution without projections (**??**) by default.

CA-Stream placement To validate the design of CA-Stream, we measure the effect of its depth on its performance *vs*. the baseline GAP in terms of both classification accuracy / number of parameters and classification metrics for interpretability. In particular, we place the stream in parallel to the network f, starting at stage ℓ and running through stage L, the last stage of f, where $0 \leq \ell \leq L$. Results are reported in **??**.

From the interpretability metrics as well as accuracy, we observe that stream configurations that allow for iterative interaction with the network features obtain the best performance, although the effect of stream placement is small in general. In many cases, the lightest stream of only one cross attention block ($S_4 - S_4$) is inferior to options allowing for more interaction. Since starting the stream at early stages has little effect on the number of parameters and performance is stable, we choose to start the stream in the first stage ($S_0 - S_4$) by default.

Accuracy and Parameters				
Placement	CLS dim	#Param	Acc↑	
$S_0 - S_4$	64	6.96M	**74.70**	
$S_1 - S_4$	256	6.95M	74.67	
$S_2 - S_4$	512	6.82M	74.67	
$S_3 - S_4$	1024	6.29M	74.67	
$S_4 - S_4$	2048	4.20M	74.63	

Interpretability Metrics						
Method	Placement	AD↓	AG↑	AI↑	I↑	D↓
Grad-CAM	$S_0 - S_4$	**12.54**	**22.67**	48.56	75.53	13.50
	$S_1 - S_4$	12.69	22.65	48.31	75.53	13.41
	$S_2 - S_4$	**12.54**	21.67	**48.58**	75.54	13.50
	$S_3 - S_4$	12.69	22.28	47.89	**75.55**	13.40
	$S_4 - S_4$	12.77	20.65	47.14	74.32	**13.37**
Grad-CAM++	$S_0 - S_4$	13.99	19.29	44.60	75.21	13.78
	$S_1 - S_4$	13.99	19.29	44.62	75.21	13.78
	$S_2 - S_4$	13.71	**19.90**	**45.43**	75.34	13.50
	$S_3 - S_4$	13.69	19.61	45.04	**75.36**	13.50
	$S_4 - S_4$	**13.67**	18.36	44.40	74.19	**13.30**
Score-CAM	$S_0 - S_4$	**7.09**	23.65	54.20	74.91	14.68
	$S_1 - S_4$	**7.09**	23.65	54.20	74.92	14.68
	$S_2 - S_4$	**7.09**	**23.66**	**54.21**	74.91	14.68
	$S_3 - S_4$	7.74	23.03	52.92	**74.97**	14.65
	$S_4 - S_4$	7.52	19.45	50.45	74.19	**14.46**

Table 4. *Effect of stream placement* on accuracy, parameters and interpretability metrics for ResNet-50 on ImageNet. $S_\ell - S_L$: CA-Stream runs from stage ℓ to L (last); #Param: parameters of CA-Stream only.

5. Conclusion

We believe we are the first to observe that attention-based pooling in transformers is the same as forming a class agnostic CAM-based saliency map and then using this map to mask the features before global average pooling, much like we mask inputs to confirm that the prediction is due to a certain object. This observation establishes that transformers have a built-in CAM-based interpretability mechanism and allows us to design a similar mechanism for convolutional networks. Masking in feature space is much more efficient than in the input space as it requires only one forward pass, although of course it is not equivalent because of interactions within the network.

Although the saliency maps obtained with our CA-Stream are not very different than those obtained with GAP, our approach improves a number of CAM-based interpretability methods on a number of convolutional networks according to most interpretability metrics, while preserving classification accuracy. By doing so, it also enhances the differences in performance between interpretability methods, facilitating their evaluation. Further study may be needed to improve the differentiation of saliency maps themselves, to possibly make a class specific representation more competitive and to apply the approach to more architectures, including transformers.

CA-Stream: Attention-based pooling for interpretable image recognition *Supplementary material*

Anonymous WACV submission

Paper ID 140

A1. More on experimental setup

Implementation details Following the training recipes from the pytorch models[1], we choose the ResNet protocol given its simplicity. Thus, we train over 90 epochs with SGD optimizer with momentum 0.9 and weight decay 10^{-4}. We start our training with a learning rate of 0.1 and decrease it every 30 epochs by a factor of 10. Our models are trained on 8 V100 GPUs with a batch size 32 per GPU, thus global batch size 256.

We follow the same protocol for both ResNet and ConvNeXt, though a different protocol might lead to improvements on ConvNeXt.

A2. More ablations

Class-specific CLS As discussed in section 3.2, the formulation of single-query cross attention as a CAM-based saliency map (6) is class agnostic (single channel weights α_k), whereas the original CAM formulation (1) is class specific (channel weights α_k^c for given class of interest c). Here we consider a class specific extension of CA-Stream using one query vector per class. In particular, the stream is initialized by one learnable parameter $\mathbf{q}_0^c$ per class c, but only one query (CLS token) embedding is forwarded along the stream. At training, c is chosen according to the target class label, while at inference, the class predicted by the baseline classifier is used instead.

Results are resported in **??**. We observe that the class specific representation for CA-Stream provides no improvement over the class agnostic representation, despite the additional complexity and parameters. We thus choose the class agnostic representation by default. The class specific approach is similar to [50] in being able to generate class specific attention maps, although no fine-tuning is required in our case.

[1] https://github.com/pytorch/vision/tree/main/references/classification

ACCURACY AND PARAMETERS						
	REPRESENTATION			#PARAM		ACC↑
	Class agnostic			32.53M		74.70
	Class specific			32.59M		74.68
INTERPRETABILITY METRICS						
METHOD	REPRESENTATION	AD↓	AG↑	AI↑	I↑	D↓
Grad-CAM	Class agnostic	12.54	22.67	48.56	75.53	13.50
	Class specific	12.53	22.66	48.58	75.54	13.50
Grad-CAM++	Class agnostic	13.99	19.29	44.60	75.21	13.78
	Class specific	13.99	19.28	44.62	75.20	13.78
Score-CAM	Class agnostic	7.09	23.65	54.20	74.91	14.68
	Class specific	7.08	23.64	54.15	74.99	14.53

Table A5. *Effect of class agnostic* vs. *class specific representation* on accuracy, parameters and interpretability metrics of CA-Stream for ResNet-50 and different interpretability methods on ImageNet. #PARAM: parameters of CA-Stream only.