B-cos Networks: Alignment is All We Need for Interpretability

Supplementary Material

Moritz Böhle
MPI for Informatics
Saarland Informatics Campus

Mario Fritz
CISPA Helmholtz Center
for Information Security

Bernt Schiele
MPI for Informatics
Saarland Informatics Campus

Table of Contents

In this supplement to our work on B-cos DNNs, we provide:

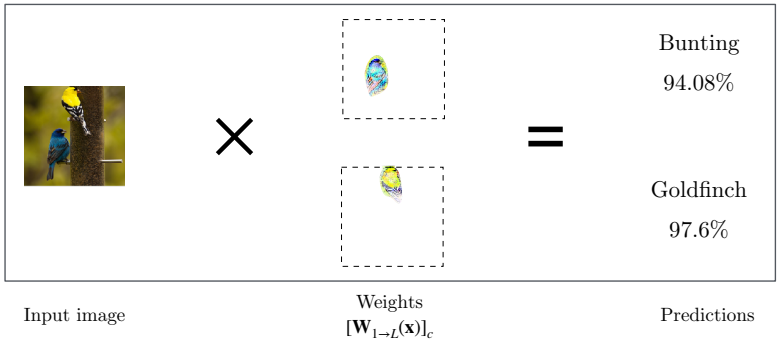


Fig. A1. Illustration of the computations of a B-cos network. For a given input image (left), the model computes an *input-dependent* linear transform $\mathbf{W}_{1\to L}(\mathbf{x})$ (center). The scalar product between the input and the weights $[\mathbf{W}_{1\to L}(\mathbf{x})]_c$ for class c (row c of $\mathbf{W}_{1\to L}(\mathbf{x})$), yields the class logits for the respective class. To obtain class probabilities (right), we apply the sigmoid function. Since the B-cos networks are trained with the BCE loss, they produce probabilities *per class* and *not a probability distribution over classes*. Thus, the probabilities do not sum to 1. For illustration purposes, we only visualise the positive contributions according to $\mathbf{W}_{1\to L}(\mathbf{x})$.

A. Additional qualitative examples

In Fig. A1, we illustrate how the linear mappings $\mathbf{W}_{1\to L}(\mathbf{x})$ are used to compute the outputs of B-cos networks. In particular, with this we would like to highlight that these linear mappings do not only constitute qualitatively convincing visualisations. Instead, they in fact constitute the actual linear transformation matrix that the model effectively applies to the input to compute its outputs and thus constitute an accurate summary of the model computations.

A.1. Additional explanations for class logits [DenseNet-121]

Comparisons between explanation methods In Fig. A2, we show additional comparisons between the model-inherent explanations based on the linear mapping $[\mathbf{W}_{1\to L}(\mathbf{x}_i)]_c$ and some post-hoc methods; in particular, we show results for GradCam (GCam) [S8], LIME [S7], Integrated Gradients (IntG) [S10], DeepLIFT [S9], and RISE [S6] on the most confidently classified image of the first 15 classes in Fig. A3. While GCam highlights similar regions and LIME also yields explanations in color, these explanations are post-hoc approximations of model behaviour. In contrast, the model-inherent explanations are not only of higher visual quality, but also summarise the model computations for the presented classes accurately, cf. Fig. A1.

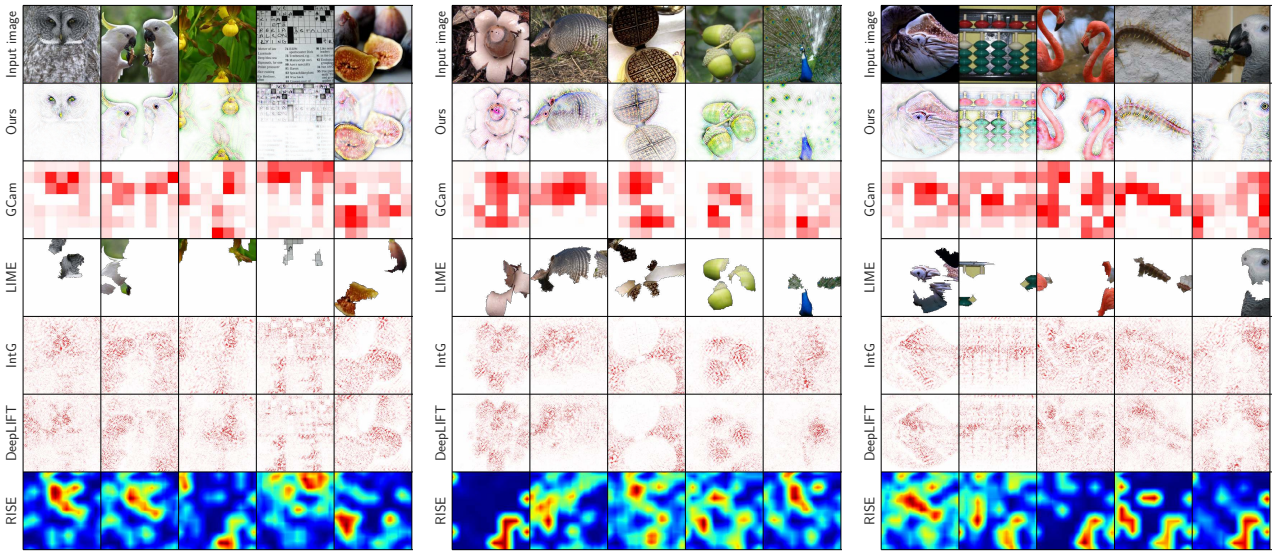


Fig. A2. Comparison between the model-inherent explanations ('Ours') and various post-hoc explanation methods, evaluated for the most confident image for the first 15 of the classes shown in Figs. A3 and A4. Note that for RISE we use its default colormap.

Model-inherent explanations. In Figs. A3 and A4, we present additional qualitative examples of the linear mappings $[\mathbf{W}_{1\to L}(\mathbf{x}_i)]_c$ that explain the class logit c in the B-cos DenseNet-121 model, see Eq. (13) in the main paper. Specifically, we show the 3 most confidently classified examples for 48 different classes; these classes were selected as those that had the highest mean confidence (sum of class logits) in the three most confidently classified images.

Note that due to the **alignment pressure** induced by the B-cos transform, the linear mappings $[\mathbf{W}_{1\to L}(\mathbf{x}_i)]_c$ align with class-discriminative features in the input images. Interestingly, we find that these features can be highly specific to particular regions in the image (see, e.g., great grey owl, centipede, school bus, planetarium, three-toed sloth, parking meter), but can also cover the entire image and include background features that correlate with the presented classes. For the latter, see e.g., the presented examples of the gondola or the golfcart: for some of these images, the weight matrix also aligns with *context features* in the background. Note, however, that the model has never been explicitly trained to highlight only the respective class objects and it is therefore expected to find that context features are also used by the model to increase its output score for the respective classes.

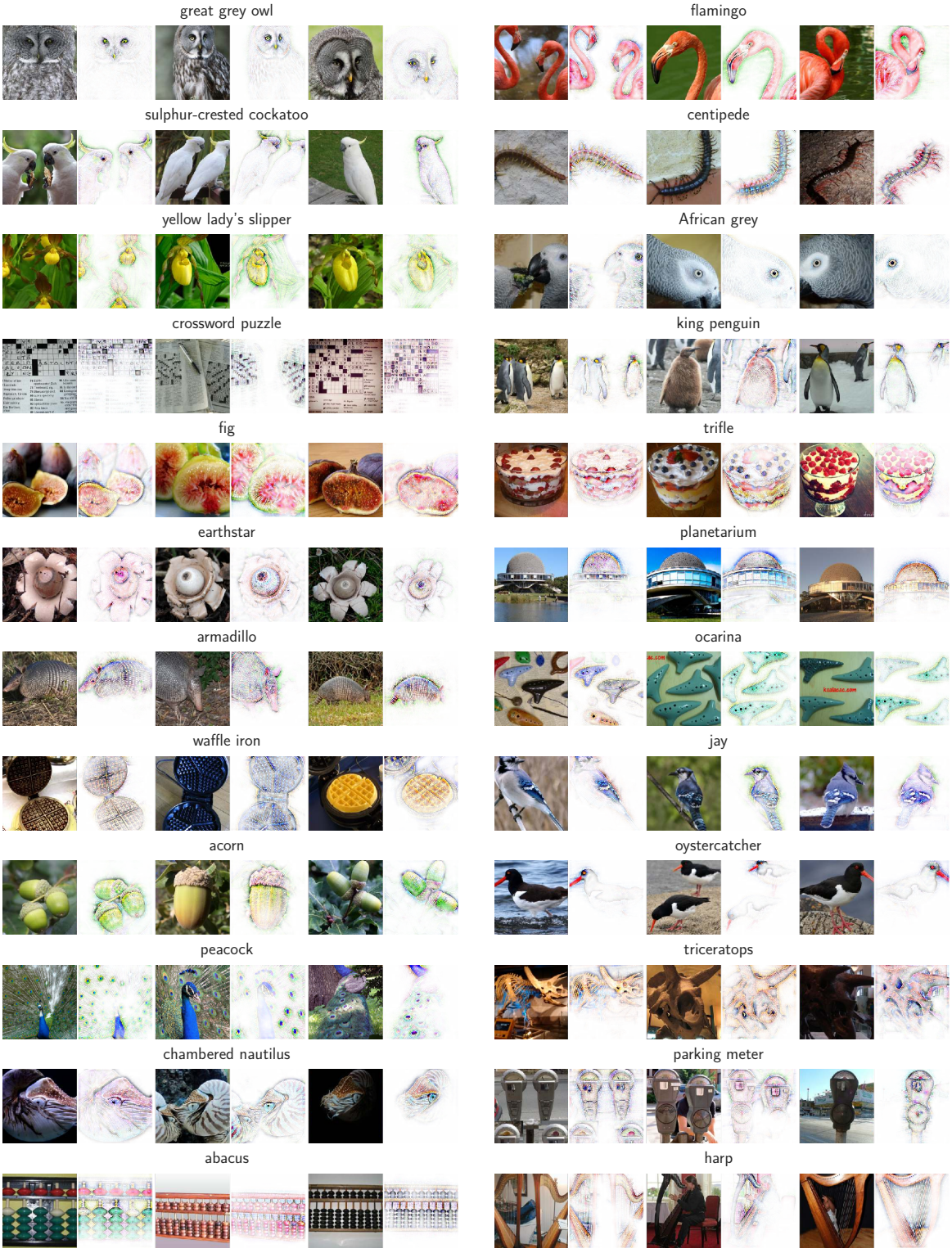


Fig. A3. First three samples $\mathbf{x}_i^c$ and linear mappings $[\mathbf{W}_{1\to L}(\mathbf{x}_i^c)]_c$ for 24 of the most confidently classified classes c from the Imagenet dataset. Specifically, the classes are sorted by the sum of the logits for those three samples. Left: Classes 1-12. Right: Classes 13-24.

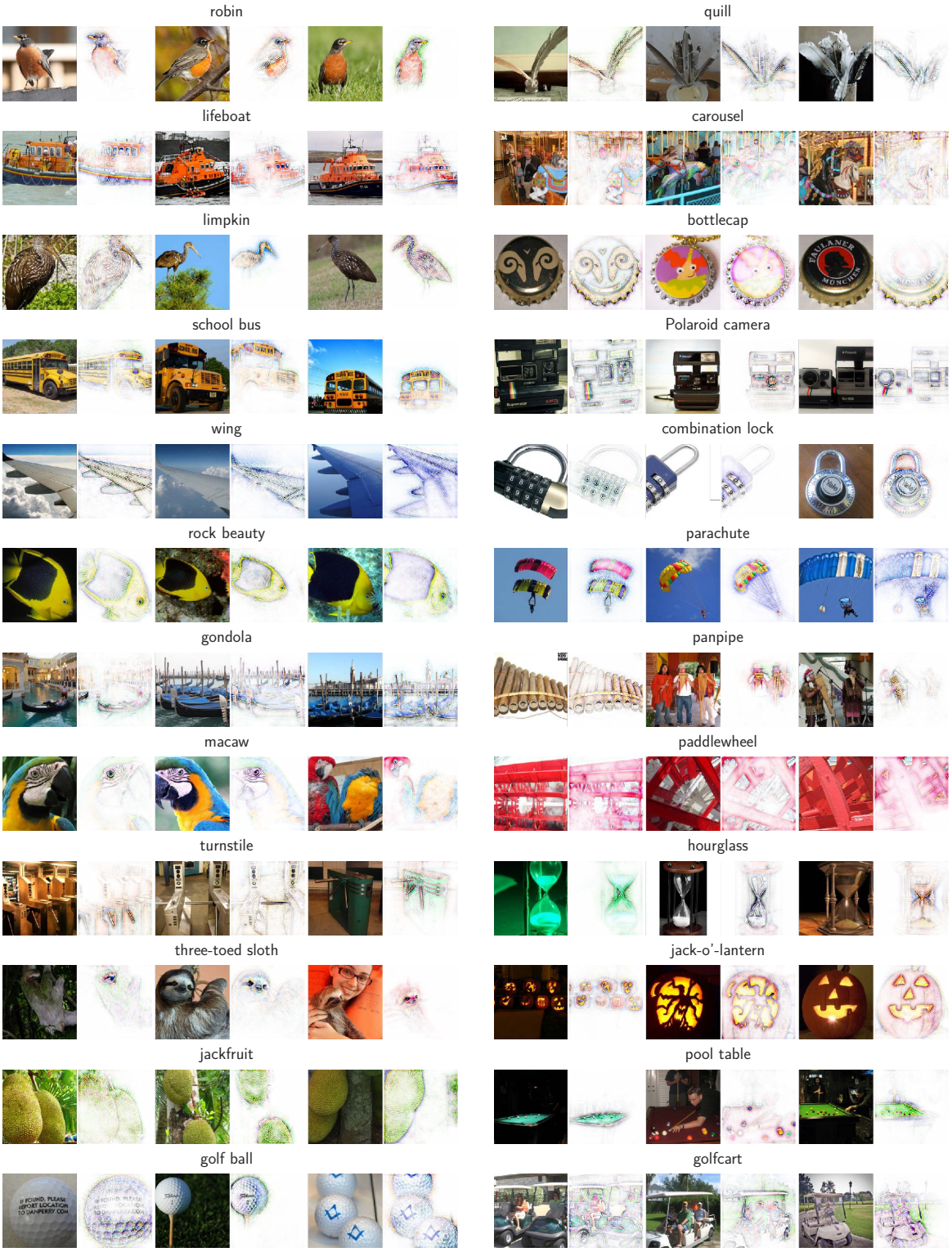


Fig. A4. First three samples $\mathbf{x}_i^c$ and linear mappings $[\mathbf{W}_{1\to L}(\mathbf{x}_i^c)]_c$ for 24 of the most confidently classified classes c from the Imagenet dataset. Specifically, the classes are sorted by the sum of the logits for those three samples. Left: Classes 25-36. Right: Classes 37-48.

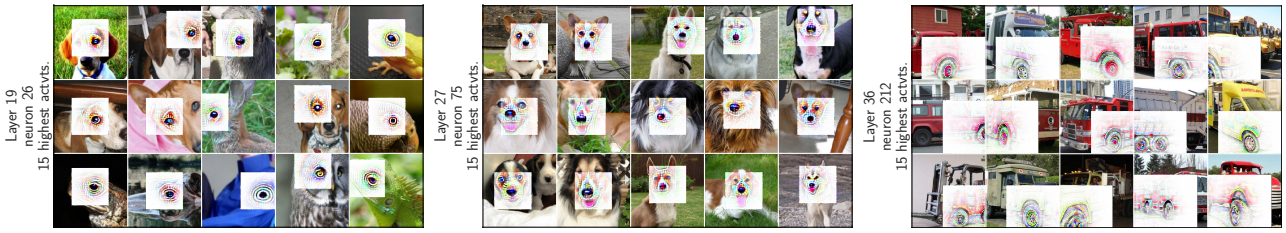
A.2. Additional explanations for intermediate neurons [DenseNet-121]

In Fig. A5, we present additional qualitative examples of the linear mappings $[\mathbf{W}_{1\to l}(\mathbf{x}_i)]_n$ that explain the activations of intermediate neurons n in layer $l{=}87$. Specifically, we show 16 out of the 20 most highly activating neurons and their explanations, which were not already shown in the main paper. We find that all neurons seem to represent specific concepts, such as faces, snouts, water, grass, etc.

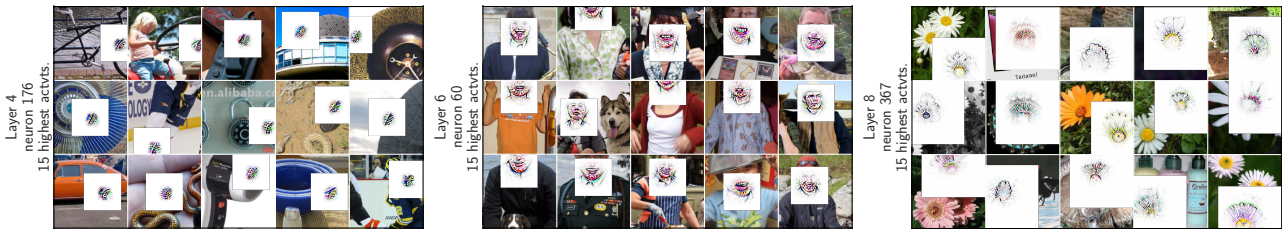
Fig. A5. Additional examples of some of the 20 most highly activating neurons in layer 87 of the B-cos DenseNet-121 model. Similar to the results shown in the main paper, we observe the neurons to represent highly specific concepts.

A.3. Explanations for other B-cos networks

In Fig. A6, we show explanations for neurons in intermediate layers of a B-cos ResNet-34, a B-cos InceptionNet, and a B-cos VGG-11. We observe that the complexity of the neurons tends to increase, similarly to the neurons in the B-cos DenseNet-121 model.



(a) B-cos ResNet-34



(b) B-cos VGG-11

(c) B-cos InceptionNet

Fig. A6. Explanations for intermediate neurons for other B-cos networks. In particular, we show results for B-cos ResNet-34 (a), B-cos VGG-11 (b), and B-cos InceptionNet (c); cf. Tab. 2 in the main paper. Similarly to the DenseNet-121 model, we observe the linear mappings $[\mathbf{W}_{1\to l}]_n$ to be of high visual quality and increase in complexity throughout the layers for all networks.

B. Additional quantitative evaluations

B.1. Localisation scores.

In Figs. B1 and B2, we present the localisation results of the *grid pointing game* [S3].

In particular, in Fig. B1, we show that of all methods, the best explanation for the B-cos network is given by the model-inherent linear transforms $\mathbf{W}_{1\to L}(\mathbf{x})$ (Eq. 13, main paper).

Moreover, from Fig. B2, we can estimate the *interpretability gain* due to replacing the linear transform in conventional models by the B-cos transform: specifically, we see that no method explains the baseline models better than the model-inherent linear transforms explain the respective B-cos network.

We note that LIME and GCam often achieve good localisation scores. However, we would like to highlight the low reliability of those explanations (high variance). Further, LIME requires many forward passes through the model to estimate feature importance, whereas the model-inherent explanations of B-cos models can be extracted in a single forward and backward pass. GCam, on the other hand, only provides explanations with comparably low resolution (cf. also Fig. A2), since it only explains the model's classification head. As such, it does not actually explain the full model, but only a small fraction of it (e.g., 1 out of 121 layers for DenseNet-121). In contrast, the model-inherent explanations of the B-cos networks provide high-resolution explanations *in color* and explain the entire model.

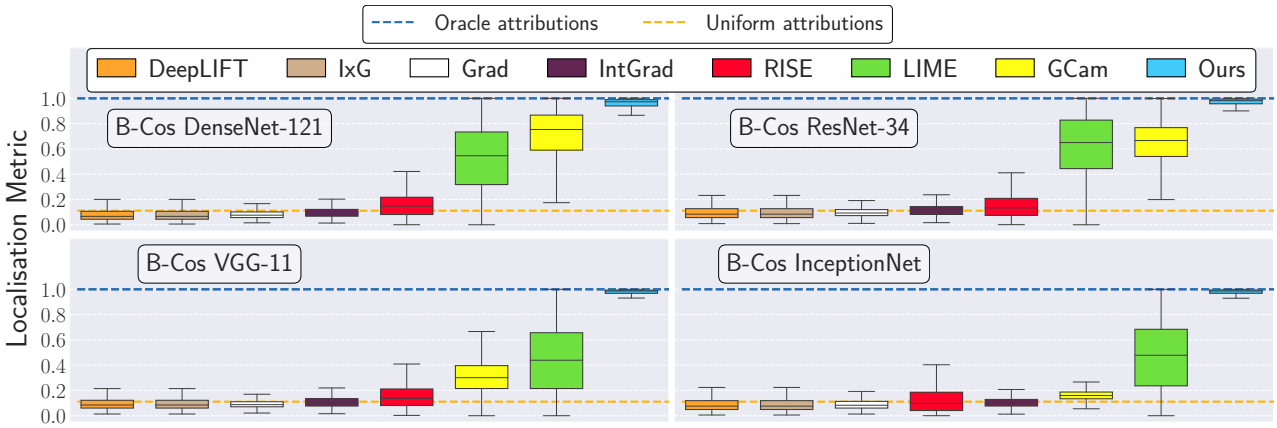


Fig. B1. Localisation metric results for all attribution methods for the converted B-cos models. Note that the DenseNet-121 and InceptionNet results are the same as in the main paper in Fig. 5.

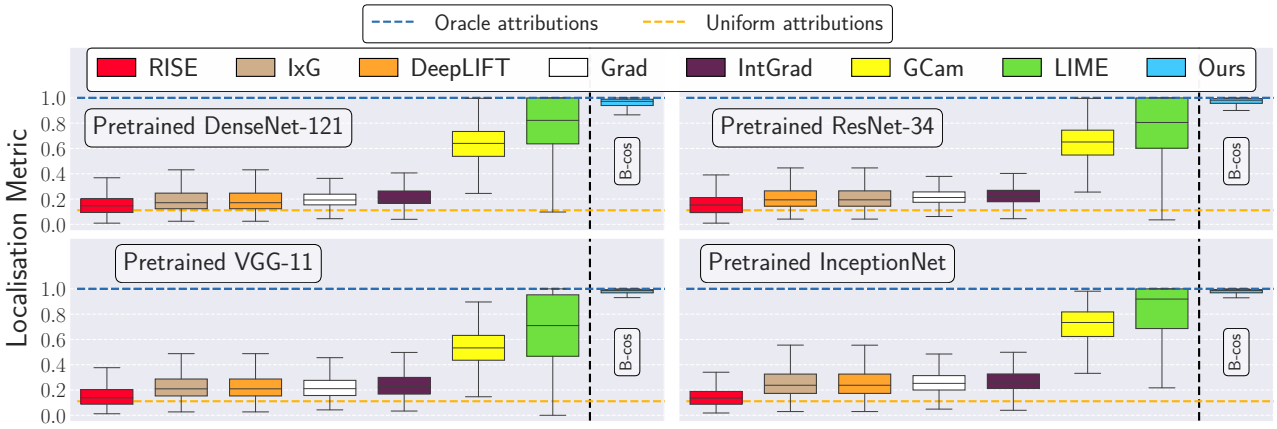


Fig. B2. Results of the localisation metric for all post-hoc attribution methods for the original, pre-trained models. Additionally, we show the B-cos results of the converted models as a reference; note that the equivalent to 'Ours' in piece-wise linear models is given by 'IxG'.

B.2. Impact of model size on performance

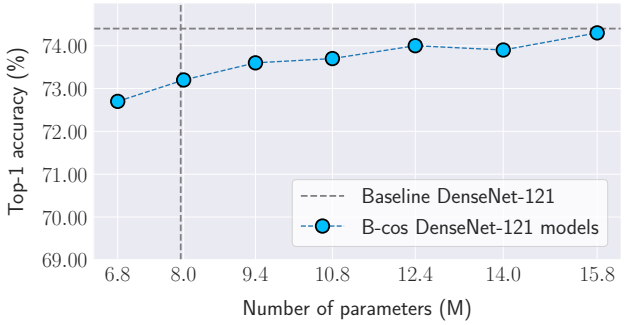


Fig. B3. Top-1 accuracy on ImageNet of B-cos DenseNet-121 models of different sizes (i.e., *growth factors*, see [S4]). These models were trained with 2 MaxOut units. For reference, we indicate the number of parameters and the accuracy results of a conventional DenseNet-121 model (Baseline, dashed lines).

	B-cos DenseNet-121 models							
MaxOut	**no**	**Maxout with 2 units**						
g-factor	32	20	22	24	26	28	30	32
#Params. (M)	7.9	6.8	8.0	9.4	10.8	12.4	14.0	15.8
Accuracy (%)	72.6	72.8	73.2	73.6	73.7	74.0	73.9	74.3

Standard DenseNet-121: accuracy 74.4; parameters 8.0; g-factor 32

Tab. B1. Top-1 accuracies (%) on ImageNet of B-cos DenseNet-121 models with different growth factors (g-factors), see [S4], and with and without MaxOut. Note that the B-cos DenseNet-121 model without MaxOut does not employ any non-linearities other than B-cos. Results of a standard DenseNet-121 model shown for reference.

In Fig. B3 and Tab. B1, we present the results of two ablation studies. On the one hand, we show the results for a B-cos DenseNet-121 model trained without MaxOut, which therefore has a similar number of parameters as the baseline model (a few less due to not using BatchNorm nor biases). On the other hand, we evaluate DenseNet models with two MaxOut units per neuron of different sizes. For this, we modify the *growth factor* of the model architectures, see [S4].

In particular, in Tab. B1, we show that despite not employing any non-linearity apart from the B-cos transform, the model with no MaxOut units also achieves competitive performance (left-most column in Tab. B1). Specifically, we only observe a minor drop in performance with respect to a conventional DenseNet-121 model ($74.4 \rightarrow 72.6$, $\Delta{=}1.8$).

Further, the accuracy of B-cos networks improves with model size; note that the B-cos DenseNet-121 with a growth factor of 22 and 2 MaxOut units has a similar size to the pretrained baseline DenseNet-121 model. While there is a drop in performance ($74.4 \rightarrow 73.2$, $\Delta{=}1.2$), the B-cos version still shows competitive accuracy results. Lastly, note that increasing model size via maxout is computationally more efficient than just increasing the growth factor in a model with a single unit, as the number of feature channels remains unchanged.

B.3. Class-discriminative information content in watermark neurons

As discussed in the main paper, we observed that some neurons seem to specifically respond to watermarks in images. While this might not seem like a semantically meaningful feature, we find that the distribution of watermarks is in fact highly skewed. In particular, in Fig. B4, we plot the distribution of classes among the images corresponding to the 500 highest neuron activations of the 'watermark neuron' (index 341); we manually inspected those images and found that neuron 341 consistently activated on text within or overlayed over the images. These images clearly exhibit a non-uniform class distribution, indicating that watermarks indeed represent a highly informative feature for the classification task.

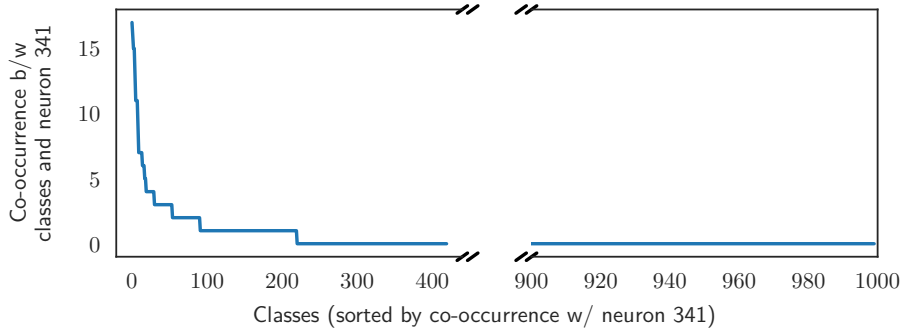


Fig. B4. Class distribution among the images corresponding to the 500 highest activations of the watermark neuron (341). The co-occurrence distribution between classes and watermarks is indeed highly skewed and only a fraction of all classes is represented among these images. This indicates high discriminative power of the watermark for classification.

C. Implementation details

Here, we provide implementation details regarding implementation of a convolutional B-cos transform (Algorithm 1), the training procedure (C.1) and the *post-hoc* attribution methods (C.2).

Algorithm 1: Pseudocode for B-cos-Conv2d, cf. Eq. (9) in the main paper.

```
# x: input, Ŵ: normed weights, k: kernel size, d_f: index of feature dimension
def bcos_conv2d(x, Ŵ, k, B):
    linear_out = conv2d(x, Ŵ) # = Ŵx
    norm = sumpool2d(x.pow(2).sum(d_f), k).sqrt()
    cos = linear_out / norm.unsqueeze(d_f)
    scaling = cos.abs().pow(B-1) # = |c(x; Ŵ)|^(B-1)
    return scaling * linear_out # = |c(x; Ŵ)|^(B-1) Ŵx
```

C.1. Training and evaluation procedure

C.1.1 CIFAR10

Architecture. For our CIFAR10 experiments, we used a 9-layer architecture with the following specifications: kernel size $k = [3, 3, 3, 3, 3, 3, 3, 3, 1]$, stride $s = [1, 1, 2, 1, 1, 2, 1, 1, 1]$, padding $p = [1, 1, 1, 1, 1, 1, 1, 1, 0]$, and output channels $o = [64, 64, 128, 128, 128, 256, 256, 256, 10]$ for layers $l = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ respectively.

When increasing the parameter B, we observed the input signal to decay strongly over the network layers, which resulted in zero outputs and hindered training. To overcome this, we scaled all layer outputs with a fixed scalar γ, which we set such that $\log_{10}\gamma = 1.5 \times B - 1.75$, which improved signal propagation. To counteract the artificial upscaling of the signal at the network output, we divided the network output by a fixed constant T for each B, such that $\log_{10} T = [-3, -3, -2, 1, 2, 2, 3]$ for B $= [1, 1.25, 1.5, 1.75, 2, 2.25, 2.5]$ respectively. In future work, we aim to examine how to automatically set an optimal scale for a given network in more detail.

Training. We trained our CIFAR10 models for 100 epochs with Adam [S5], an initial learning rate of 1×10^{-3}, and a batch size of 64. Further, we used a cosine learning rate schedule and decayed the learning rate to 1×10^{-5} over the 100 epochs and applied horizontal flipping and padded random cropping for augmenting the data. We used a bias term of $\mathbf{b} = \log(0.1/0.9)$, which yields a uniform probability distribution for zero inputs ($[\mathbf{f}(\mathbf{x} = \mathbf{0})]_i = [\sigma(\mathbf{W}_{1\to L}\,\mathbf{0} + \mathbf{b})]_i = 0.1\ \forall\ i$).

C.1.2 ImageNet

Training. Similar to the CIFAR10 experiments, we observed signals to decay quickly for deep networks and to be dependent on the number of channels used. To overcome this, we scaled the layer outputs by $\gamma = s/\sqrt{d}$ with s a network-dependent hyperparameter and d the input dimensionality (i.e., k^2c for a convolutional layer with kernel size k and an input with c feature channels). Specifically, we chose $s = 100$ for DenseNets and ResNets, $s = 200$ for the InceptionNet, and $s = 1000$ for the VGG model.

Moreover, as in the CIFAR10 experiments, we divided the network outputs by a temperature parameter T. In detail, for the results in this paper we set $\log_{10}\ T = -3$ for the DenseNet models, $\log_{10}\ T = 1$ for ResNet, $\log_{10}\ T = 0$ for InceptionNet, and $\log_{10}\ T = -1$ for the VGG model. These parameters were experimentally determined to achieve good accuracies and stable training behaviour. In future work, we plan to investigate how to set the temperature parameter automatically.

Finally, we added the auxiliary loss in the InceptionNet with a weighting of $\lambda = 1$, used images of size $s = 299$ for Inception (224 otherwise), and employed RandAugment with $n = 2$ and $m = 9$. The bias term $\mathbf{b}$ was set to $\mathbf{b} = \log(0.01/0.99)$ for all ImageNet experiments.

C.2. Attribution methods

We compare the model-inherent explanations, given by the linear transform $\mathbf{W}_{1\to L}(\mathbf{x})$, against the following post-hoc attribution methods: the vanilla gradient (Grad, [S2]), 'Input×Gradient' (IxG, cf. [S1]), Integrated Gradients (IntGrad, [S10]), DeepLIFT ([S9]), GradCam (GCam, [S8]), LIME ([S7]), and (RISE [S6]).

For all methods except RISE, LIME, and GCam, we rely on the captum library (`github.com/pytorch/captum`). For IntGrad, we set n_steps $= 50$ for integrating over the gradients. For RISE and LIME, we used the official implementations

available at github.com/eclique/RISE and github.com/marotcr/lime respectively. We generated 500 masks for RISE and set the hyperparameters s and p to their default values of $s = 8$ and $p = 0.1$. Similarly, we used 500 samples for LIME, and used the default values for the kernel size ($k = 4$) and the number of features ($n = 5$).

C.2.1 Localisation metric

We evaluated all attribution methods on the *grid pointing game* [S3]. For this, we constructed 500 3×3 grid images. For an example of a 2×2 grid, see Fig. 3 in the main paper. As was done in [S3], we sorted the images according to the models' classification confidence for each class and then sampled a random set of classes for each multi-image. For each of the sampled classes, we then included the most confidently classified image in the grid that had not already been used in a previous grid image.

D. Additional derivations and discussions

D.1. On the B-cos transform in matrix form

In the following, we provide additional details on how to express the B-cos transform in matrix form.

As shown in Eq. (3) in the main paper, the B-cos transform is given by

$$\text{B-cos}(\mathbf{x}; \mathbf{w}) = ||\widehat{\mathbf{w}}||\ ||\mathbf{x}|| \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}} \times \operatorname{sgn}(c(\mathbf{x}, \widehat{\mathbf{w}}))\ , \tag{D.1}$$

$$\text{with} \quad c(\mathbf{x}, \mathbf{w}) = \cos(\angle(\mathbf{x}, \mathbf{w}))\ , \tag{D.2}$$

$$\widehat{\mathbf{w}} = \mathbf{w}/||\mathbf{w}||\ , \tag{D.3}$$

$\angle(\mathbf{x}, \mathbf{w})$ returning the angle between $\mathbf{x}$ and $\mathbf{w}$, and sgn the sign function. Note that the sign function can be expressed as $\operatorname{sgn}(a) = a/|a|$ for $|a| \neq 0$ and zero otherwise. Hence, Eq. (D.1) can be expressed as

$$\text{B-cos}(\mathbf{x}; \mathbf{w}) = ||\widehat{\mathbf{w}}||\ ||\mathbf{x}|| \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}} \times \operatorname{sgn}(c(\mathbf{x}, \widehat{\mathbf{w}})) \tag{D.4}$$

$$\text{(replace sgn)} \qquad = ||\widehat{\mathbf{w}}||\ ||\mathbf{x}|| \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}} \times c(\mathbf{x}, \widehat{\mathbf{w}})/|c(\mathbf{x}, \widehat{\mathbf{w}})| \tag{D.5}$$

$$\text{(combine cos terms)} \qquad = ||\widehat{\mathbf{w}}||\ ||\mathbf{x}|| \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}-1} \times c(\mathbf{x}, \widehat{\mathbf{w}}) \tag{D.6}$$

$$\text{(reorder)} \qquad = ||\widehat{\mathbf{w}}||\ ||\mathbf{x}|| \times c(\mathbf{x}, \widehat{\mathbf{w}}) \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}-1} \tag{D.7}$$

$$\text{(write first three factors as linear transform)} \qquad = \widehat{\mathbf{w}}^T\mathbf{x} \times |c(\mathbf{x}, \widehat{\mathbf{w}})|^{\mathrm{B}-1}\ . \tag{D.8}$$

For clarity, we marked the changes between lines in the above equations in red.

From Eq. (D.8) it becomes clear that a B-cos transform simply computes a rescaled linear transform. Thus, multiple units in parallel (i.e., a layer $\mathbf{l}^*$ of B-cos units) can easily be expressed in matrix form via

$$\mathbf{l}^*(\mathbf{x}) = |c(\mathbf{x}, \widehat{\mathbf{W}})|^{\mathrm{B}-1} \times \widehat{\mathbf{W}}\mathbf{x}\ . \tag{D.9}$$

Here, the $\times$, cos, and absolute value operators are applied element-wise and the rows of $\widehat{\mathbf{W}}$ are given by $\widehat{\mathbf{w}}_n$ of the individual units n.

Hence, the output of each unit (entry in output vector $\mathbf{l}^*$) is the down-scaled linear transform from Eq. (D.8). Note that Eq. (D.9) is the same as Eq. (9) in the main paper.

D.2. On the relevance of image encoding for the visualisations

As we describe in the main paper, we encode image pixels as $[r, g, b, 1-r, 1-g, 1-b]$. This has two important advantages.

On the one hand, as argued by [S3], this overcomes a bias towards bright pixels. For this, note that the model output is computed as a linear transform of the input $\mathbf{x}$. As such, the contribution to the output per pixel is given by the weighted input strength. In particular, a specific pixel location (i, j) with color channels c contributes $\sum_c w_{(i,j,c)}x_{(i,j,c)}$ to the output. Under the conventional encoding—i.e., $[r, g, b]$—, a black pixel is encoded by $x_{(i,j,c)}=0$ for $c \in \{1, 2, 3\}$ and can therefore not contribute to the model output. Since we train the model to maximise its outputs (binary cross entropy loss, see Sec. 3.2.2 in the main paper), the network will preferentially encode bright pixels, as these can produce higher contributions for maximising the output than dark pixels. In contrast, under the new encoding dark and bright pixels have the same amount of 'signal' that can be weighted, i.e., $\sum_c x_{(i,j,c)}=3\ \forall\ (i, j)$.

Moreover, this encoding allows to unambiguously infer the color of a pixel solely based on the angle of the pixel vector $[r, g, b, 1-r, 1-g, 1-b]$. To contrast this with the original encoding, consider a pixel that is (almost) completely black and given by $[r, g, b]$ with $g=0, b=0, r=0.001$. This pixel has the same angle as a red pixel, given by $r=1, g=0, b=0$. Thus, these two colors cannot be disambiguated based on their angle. By adding the three additional color channels $[1-r, 1-g, 1-b]$, each color channel is uniquely encoded by the direction of the color channel vector, e.g.,$[r, 1-r]$. Finally, note that the B-cos transform induces an alignment pressure on the weights, i.e., the model weights are optimised such that $\mathbf{W}_{1\to L}$ points in the same direction as (important features in) the input. Consequently, the weights will reproduce the *angles* of the pixels, but there is no constraint on their *norm*. Since the angle is sufficient for inferring the color, we can nevertheless decode the angles of the weight vectors into RGB colors, as, e.g., shown in Figs. A2, A3, A4, A5 and A6.

References

[S1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 10

[S2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research (JMLR)*, 2010. 10

[S3] Moritz Böhle, Mario Fritz, and Bernt Schiele. Convolutional Dynamic Alignment Networks for Interpretable Classifications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8, 11

[S4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9

[S5] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 10

[S6] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *British Machine Vision Conference (BMVC)*, 2018. 2, 10

[S7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2016. 2, 10

[S8] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *International Conference on Computer Vision (ICCV)*, 2017. 2, 10

[S9] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *International Conference on Machine Learning (ICML)*, 2017. 2, 10

[S10] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In Doina Precup and Yee Whye Teh, editors, *International Conference on Machine Learning (ICML)*, 2017. 2, 10