Optimising for Interpretability: Convolutional Dynamic Alignment Networks

Moritz Böhle, Mario Fritz, and Bernt Schiele

Abstract—We introduce a new family of neural network models called Convolutional Dynamic Alignment Networks (CoDA Nets), which are performant classifiers with a high degree of inherent interpretability. Their core building blocks are Dynamic Alignment Units (DAUs), which are optimised to transform their inputs with dynamically computed weight vectors that align with task-relevant patterns. As a result, CoDA Nets model the classification prediction through a series of input-dependent linear transformations, allowing for linear decomposition of the output into individual input contributions. Given the alignment of the DAUs, the resulting contribution maps align with discriminative input patterns. These model-inherent decompositions are of high visual quality and outperform existing attribution methods under quantitative metrics. Further, CoDA Nets constitute performant classifiers, achieving on par results to ResNet and VGG models on e.g. CIFAR-10 and TinyImagenet. Lastly, CoDA Nets can be combined with conventional neural network models to yield powerful classifiers that more easily scale to complex datasets such as Imagenet whilst exhibiting an increased *interpretable depth*, i.e., the output can be explained well in terms of contributions from intermediate layers within the network.

Index Terms—Explainability in Deep Learning, Convolutional Neural Networks

1 Introduction

Neural networks are powerful models that excel at a wide range of tasks. However, they are notoriously difficult to interpret and extracting explanations for their predictions is an open research problem. Linear models, in contrast, are generally considered interpretable, because the *contribution* ('the weighted input') of every dimension to the output is explicitly given. Interestingly, many modern neural networks implicitly model the output as a linear transformation of the input; a ReLU-based [1] neural network, e.g., is piece-wise linear and the output thus a linear transformation of the input, cf. [2]. However, due to the highly non-linear manner in which these linear transformations are 'chosen', the corresponding contributions per input dimension do not seem to represent the learnt model parameters well, cf. [3], and a lot of research is being conducted to find better explanations for the decisions of such neural networks, cf. [4], [5], [6], [7], [8], [9], [10], [11].

In this work, we introduce a novel network architecture, the **Convolutional Dynamic Alignment Networks (CoDA Nets)**, for which the model-inherent contribution maps are faithful projections of the internal computations and thus good 'explanations' of the model prediction. There are two main components to the interpretability of the CoDA Nets. First, the CoDA Nets are **dynamic linear**, i.e., they compute their outputs through a series of input-dependent linear transforms, which are based on our novel **Dynamic Alignment Units (DAUs)**. As in linear models, the output can thus be decomposed into individual input contributions, see Fig. 1. Second, the DAUs are structurally biased to compute weight vectors that **align with relevant patterns** in their inputs. In combination, the CoDA Nets thus inherently produce contribution maps that are 'optimised for interpretability': since each linear transformation matrix and thus their combination is optimised to align with discriminative features, the contribution maps reflect the most discriminative features *as used by the model*.

With this work, we present a new direction for building inherently more interpretable neural network architectures with high modelling capacity. In detail, we would like to highlight the following contributions:

1) We introduce the concept of Dynamic Alignment Units (DAUs), which improve the interpretability of neural networks and have two key properties: they are *dynamic linear* and need to align their dynamically computed weights their inputs to achieve large outputs, since the norm of their weights is explicitly constrained.
2) Apart from normalising the dynamically computed DAU weights by their vector norm, we show that similar results can be achieved when normalising them by an *upper bound* of their norm which results in more efficient DAUs.
3) We show that networks of DAUs *inherit* the dynamic linearity and the alignment properties from their constituent DAUs. In particular, we introduce Convolutional Dynamic Alignment Networks (CoDA Nets), which are built out of multiple layers of DAUs. As a result, the *model-inherent contribution maps* of CoDA Nets highlight discriminative patterns in the input.

- *Moritz Böhle is the corresponding author. He is with the Department of Computer Vision and Machine Learning, Max Planck Institute for Informatics, Saarbrücken 66123, Germany. E-mail: mboehle@mpi-inf.mpg.de*
- *Bernt Schiele is with the Department of Computer Vision and Machine Learning, Max Planck Institute for Informatics, Saarbrücken 66123, Germany. E-mail: schiele@mpi-inf.mpg.de*
- *Mario Fritz is with the CISPA Helmholtz Center for Information Security, Saarbrücken 66123, Germany. E-mail: fritz@cispa.de.*

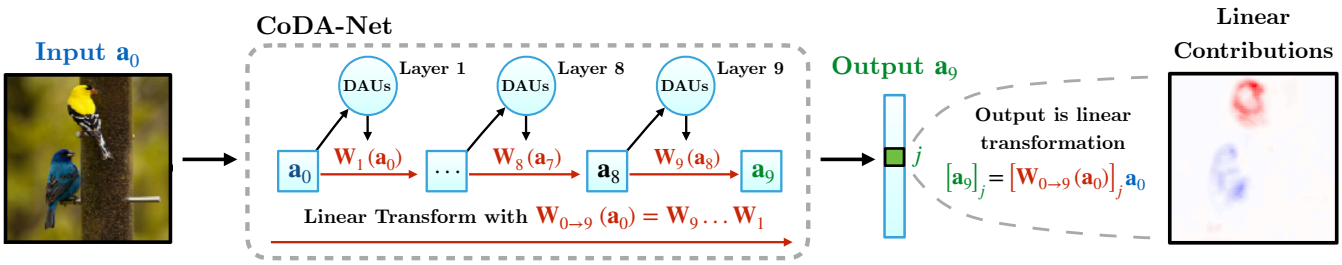


Fig. 1: Sketch of a 9-layer CoDA-Net, which computes its **output $\mathbf{a}_9$** for an **input $\mathbf{a}_0$** as a linear transform via a matrix $\mathbf{W}_{0\to 9}(\mathbf{a}_0)$. As such, the output can be linearly decomposed into input contributions (see right). This 'global' transformation matrix $\mathbf{W}_{0\to 9}$ is computed successively via multiple layers of Dynamic Alignment Units (DAUs). These layers, in turn, produce intermediate linear transformation matrices $\mathbf{W}_l(\mathbf{a}_{l-1})$ that align with the inputs of layer l. As a result, the combined matrix $\mathbf{W}_{0\to 9}$ also aligns well with task-relevant patterns. Positive (negative) contributions for the class 'goldfinch' are shown in red (blue).

4) We further show that the alignment of the DAUs can be promoted by applying a 'temperature scaling' to the final output of the CoDA Nets.
5) We show that the resulting contribution maps perform well under commonly employed *quantitative* criteria for attribution methods. Moreover, under *qualitative* inspection, we note that they exhibit a high degree of detail.
6) We analyse how the models are affected by different normalisation functions in the DAU weight calculation in terms of accuracy, interpretability, as well as efficiency.
7) Beyond interpretability, CoDA Nets are performant classifiers and yield competitive classification accuracies on the CIFAR-10 and TinyImagenet datasets.
8) We show that CoDA Nets can be seamlessly combined with conventional networks. The resulting hybrid networks exhibit an increased 'interpretable depth' whilst taking advantage of the efficiency and strong modelling capacity of the base networks. Such networks hold great potential for designing models that are inherently interpretable up to a user-defined minimal resolution.

2 Related Work

Interpretability. In order to make machine learning models more interpretable, a variety of techniques has been developed. While there are different ways in which interpretability can be defined, in this work we focus on attributing importance values to input features; for an extensive review regarding interpretability in machine learning, see [12].

Current techniques for interpreting neural networks via importance attribution can broadly be split into two categories: deriving *post-hoc* explanations and developing inherently interpretable models. In the following, we will first describe common approaches for post-hoc interpretability.

On the one hand, research regarding post-hoc explanations has been undertaken to develop model-agnostic explanation methods for which the model behaviour under perturbed inputs is analysed; this includes among others [13], [14], [15]. While their generality and the applicability to any model are advantageous, these methods typically require evaluating the respective model several times and are therefore costly approximations of model behaviour. Further, they rely on the assumption that the models generalise to such out-of-distribution (OOD) data (the perturbed / occluded inputs) in a stable manner, such that the outputs on the OOD data can be used to explain model behaviour on in-distribution data.

On the other hand, many techniques that explicitly take advantage of the internal computations have been proposed for explaining the model predictions, including, for example, [4], [5], [6], [7], [8], [9], [10], [11]. Such methods typically distribute importance values layer by layer in a backward pass and introduce different rules for this re-distribution. Similarly, the model-inherent contribution maps in the CoDA Nets can also be obtained as a layer-wise decomposition of the output. However, there is a key difference: producing well-aligned linear decompositions is actually optimised for in the CoDA Nets during training, whereas other methods are developed without taking the optimisation procedure into account. As a result, the *inherent* explanations in the CoDA Nets lend themselves better to understanding the model outputs.

In contrast to techniques that aim to explain models *post-hoc*, some recent work has focused on designing new types of network architectures, which are *inherently* more interpretable. Examples of this are the prototype-based neural networks [16], the BagNet [17] and the self-explaining neural networks (SENNs) [18]. Similarly to our proposed architectures, the SENNs and the BagNets derive their explanations from a linear decomposition of the output into contributions from the input (features). This *dynamic linearity*, i.e., the property that the output is computed via some form of an input-dependent linear mapping, is additionally shared by the entire model family of piece-wise linear networks (e.g., ReLU-based networks). In fact, the contribution maps of the CoDA Nets are conceptually similar to evaluating the 'Input×Gradient' (IxG), cf. [3], on piece-wise linear models, which also yields a linear decomposition in form of a contribution map. However, in contrast to the piece-wise linear functions, we combine this *dynamic linearity* with a structural bias towards an alignment between the contribution maps and the discriminative patterns in the input. This results in explanations of much higher quality, whereas IxG on piece-wise linear models has been found to yield unsatisfactory explanations of model behaviour [3].

Architectural similarities. In our CoDA Nets, the convolutional kernels are dependent on the specific patch that

they are applied on; i.e., a CoDA layer might apply different filters at every position in the input. As such, these layers can be regarded as an instance of dynamic local filtering layers as introduced in [19]. Further, our dynamic alignment units (DAUs) share some high-level similarities to attention networks, cf. [20], in the sense that each DAU has a limited budget to distribute over its dynamic weight vectors (bounded norm), which is then used to compute a weighted sum. However, whereas in attention networks the weighted sum is typically computed over vectors (the 'value vectors') which differ from the input to the attention module (the 'key' and 'query' vectors), a DAU outputs a *scalar* which is a weighted sum of all scalar entries in the input. Moreover, we note that at their optimum (maximal average output over a set of inputs), the DAUs solve a constrained low-rank matrix approximation problem [21]. While low-rank approximations have been used for increasing parameter efficiency in neural networks, cf. [22], this concept has to the best of our knowledge not been used in order to endow neural networks with a structural bias towards finding low-rank approximations of the input for increased interpretability in classification tasks. Lastly, the CoDA Nets are related to capsule networks. However, whereas in classical capsule networks the activation vectors of the capsules directly serve as input to the next layer, in CoDA Nets the corresponding vectors are used as convolutional filters. We include a detailed comparison in the supplement.

3 Dynamic Alignment Networks

In this section, we present our novel type of network architecture: the Convolutional Dynamic Alignment Networks (CoDA Nets). For this, we first introduce Dynamic Alignment Units (DAUs) as the basic building blocks of CoDA Nets and discuss two of their key properties in sec. 3.1. Concretely, we show that these units linearly transform their inputs with dynamic (input-dependent) weight vectors and, additionally, that they are biased to align these weights with the input during optimisation. Given the computational costs of evaluating DAUs, in sec. 3.2 we further present an alternative formulation of the DAUs for increased efficiency. We then discuss how DAUs can be used for classification (sec. 3.3) and how we build performant networks out of multiple layers of convolutional DAUs (sec. 3.4). Importantly, the resulting *linear decompositions* of the network outputs are optimised to align with discriminative patterns in the input, making them highly suitable for interpreting the network predictions.

In particular, we structure this section around the following **three important properties** (P1-P3) of the DAUs:

P1: Dynamic linearity. The DAU output o is computed as a dynamic (input-dependent) linear transformation of the input $\mathbf{x}$, such that $o = \mathbf{w}(\mathbf{x})^T\mathbf{x} = \sum_j w_j(\mathbf{x})x_j$. Hence, o can be decomposed into contributions from individual input dimensions, which are given by $w_j(\mathbf{x})x_j$ for dimension j.

P2: Alignment maximisation. Maximising the average output of a single DAU over a set of inputs $\mathbf{x}_i$ maximises the alignment between inputs $\mathbf{x}_i$ and the weight vectors $\mathbf{w}(\mathbf{x}_i)$. As the modelling capacity of $\mathbf{w}(\mathbf{x})$ is restricted, $\mathbf{w}(\mathbf{x})$ will encode the most frequent patterns in the set of inputs $\mathbf{x}_i$.

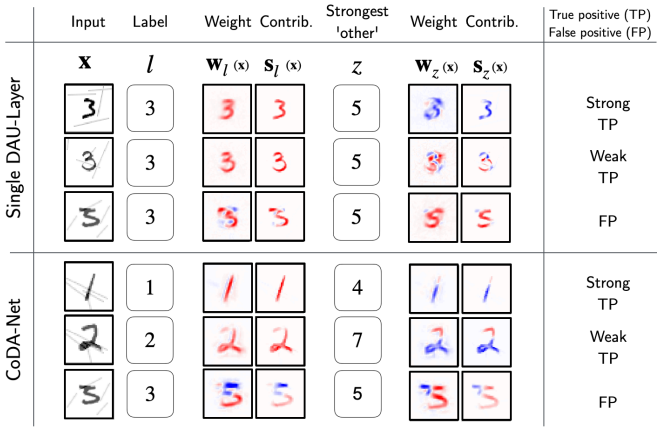


Fig. 2: For different inputs $\mathbf{x}$, we visualise the linear weights and contributions (for the single layer, see eq. (7), for the CoDA-Net eq. (11)) for the ground truth label l and the strongest non-label output z. As can be seen, the weights align well with the input images. The first three rows are based on a single DAU layer, the last three on a 5 layer CoDA-Net. The first two samples (rows) per model are correctly classified and the last one is misclassified.

P3: Inheritance. When combining multiple DAU layers to form a Dynamic Alignment Network (DA Net), the properties P1 and P2 are *inherited*: DA Nets are dynamic linear (P1) and maximising the last layer's output induces an output maximisation in the constituent DAUs (P2).

These properties increase the interpretability of a DA Net, such as a CoDA Net (sec. 3.4) for the following reasons. First, the output of a DA Net can be decomposed into contributions from the individual input dimensions, similar to linear models (cf. Fig. 1, P1 and P3). Second, we note that optimising a neural network for classification applies a maximisation to the outputs of the last layer for every sample. This maximisation aligns the dynamic weight vectors $\mathbf{w}(\mathbf{x})$ of the constituent DAUs of the DA Net with their respective inputs (cf. Fig. 2 as well as P2 and P3).

Importantly, the weight vectors will align with the *discriminative* patterns in their inputs when optimised for classification as we show in sec. 3.3. As a result, the model-inherent contribution maps of CoDA Nets are optimised to align well with *discriminative input patterns* in the input image and the interpretability of our models thus forms part of the global optimisation procedure.

3.1 Dynamic Alignment Units

We define the Dynamic Alignment Units (DAUs) by

$$\mathrm{DAU}(\mathbf{x}) = g(\mathbf{A}\mathbf{B}\mathbf{x} + \mathbf{b})^T\mathbf{x} = \mathbf{w}(\mathbf{x})^T\,\mathbf{x} \quad . \tag{1}$$

Here, $\mathbf{x} \in \mathbb{R}^d$ is an input vector, $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times d}$ are trainable transformation matrices, $\mathbf{b} \in \mathbb{R}^d$ a trainable bias vector, and $g(\mathbf{u}) = \alpha(||\mathbf{u}||)\mathbf{u}$ is a non-linear function that scales the norm of its input. In contrast to using a single matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$, using $\mathbf{AB}$ allows us to control the maximum rank r of the transformation and to reduce the number of parameters; we will hence refer to r as the rank of a DAU. As can be seen by the right-hand side of eq. (1), the DAU linearly transforms the input $\mathbf{x}$ (P1). At the

Fig. 3: Eigenvectors (EVs) of $\mathbf{AB}$ after maximising the output of a rank-3 DAU over a set of noisy samples of 3 MNIST digits. Effectively, the DAUs encode the most frequent components in their EVs, similar to a principal component analysis (PCA).

same time, given the quadratic form ($\mathbf{x}^T\mathbf{B}^T\mathbf{A}^T\mathbf{x}$) and the rescaling function $\alpha(||\mathbf{u}||)$, the output of the DAU is a non-linear function of its input. In the context of DAUs, we are particularly interested in functions that constrain the norm of the weight vectors $\mathbf{w}(\mathbf{x})$, such as, e.g., rescaling to unit norm (L2) or the squashing function (SQ, see [23]):

$$\mathrm{L2}(\mathbf{u}) = \frac{\mathbf{u}}{||\mathbf{u}||_2} \quad \text{and} \quad \mathrm{SQ}(\mathbf{u}) = \mathrm{L2}(\mathbf{u}) \times \frac{||\mathbf{u}||_2^2}{1+||\mathbf{u}||_2^2} \tag{2}$$

In sec. 3.2, we further present an approximation to these rescaling functions, which lowers the computational cost of the DAUs whilst maintaining their bounding property $||\mathbf{w}(\mathbf{x})|| \leq 1$. Given such a bound on $\mathbf{w}(\mathbf{x})$, the output of the DAUs will be upper-bounded by the norm of the input:

$$\mathrm{DAU}(\mathbf{x}) = ||\mathbf{w}(\mathbf{x})||\,||\mathbf{x}||\cos(\angle(\mathbf{x},\mathbf{w}(\mathbf{x}))) \leq ||\mathbf{x}|| \tag{3}$$

As a corollary, for a given input $\mathbf{x}_i$, the DAUs can only achieve this upper bound if $\mathbf{x}_i$ is an eigenvector (EV) of the linear transform $\mathbf{ABx}+\mathbf{b}$. Otherwise, the cosine in eq. (3) will not be maximal[1]. As can be seen in eq. (3), maximising the average output of a DAU over a set of inputs $\{\mathbf{x}_i|\, i = 1,...,n\}$ maximises the alignment between $\mathbf{w}(\mathbf{x})$ and $\mathbf{x}$ (P2). In particular, it optimises the parameters of the DAU such that the *most frequent input patterns* are encoded as EVs in the linear transform $\mathbf{ABx}+\mathbf{b}$, similar to an r-dimensional PCA decomposition (r the rank of $\mathbf{AB}$). In fact, as discussed in the supplement, the optimum of the DAU maximisation solves a low-rank matrix approximation [21] problem similar to singular value decomposition. As an illustration of this property, in Fig. 3 we show the 3 EVs[2] of matrix $\mathbf{AB}$ (with rank $r=3$, bias $\mathbf{b}=\mathbf{0}$) after optimising a DAU over a set of n noisy samples of 3 specific MNIST [24] images; for this, we used $n=3072$ and zero-mean Gaussian noise. As expected, the EVs of $\mathbf{AB}$ encode the original, noise-free images, since this on average maximises the alignment (eq. (3)) between the weight vectors $\mathbf{w}(\mathbf{x}_i)$ and the input samples $\mathbf{x}_i$ over the dataset.

3.2 Efficient DAUs: Bounding the Bound

As discussed in the previous section, we introduce a norm constraint for the DAU weights $\mathbf{w}(\mathbf{x})$ to ensure that large outputs can only be achieved for well-aligned weights. However, the explicit norm constraint on $\mathbf{w}(\mathbf{x})$ requires its explicit calculation, which we have observed to significantly impact the evaluation time of DAUs. Therefore, we evaluate an additional formulation of the DAUs in which we only *constrain an upper bound* of the norm of $\mathbf{w}(\mathbf{x})$. For this, we take advantage of the following inequality:

$$||\mathbf{w}(\mathbf{x})|| = ||\mathbf{ABx}|| \leq ||\mathbf{A}||_F\,||\mathbf{Bx}|| \quad . \tag{4}$$

Here, $||\cdot||_F$ denotes the Frobenius norm and $||\cdot||$ the L_2 vector norm; this inequality reflects the fact that the Frobenius norm is *compatible* with the L_2 vector norm. Note that using this approximation for the norm computation *bounds the output bound* of the DAUs and is *at least as tight* as the bound in eq. (3). As a result, without the bias term $\mathbf{b}$, the output of the corresponding DAUs can be calculated as

$$\mathrm{eDAU}(\mathbf{x})|| = ||\mathbf{Bx}||^{-1}\,(\mathbf{Bx})^T\left(\mathbf{A}'^T\mathbf{x}\right) \tag{5}$$

$$\text{with} \quad \mathbf{A}' = ||\mathbf{A}||_F^{-1}\mathbf{A} \tag{6}$$

We will henceforth refer to this non-linear output computation as *weight bounding* (WB). Note that under this formulation the d-dimensional weights $\mathbf{w}(\mathbf{x})$ are never explicitly calculated and the output is instead obtained as a dot product in $\mathbb{R}^r$ between the vectors $\mathbf{Bx}$ and $\mathbf{A}'^T\mathbf{x}$. Further, for convolutional DAUs (see sec. 3.4), the matrix $\mathbf{A}'$ has to be computed only once for all positions. As we show in sec. 5.3, this can result in significant gains in efficiency.

3.3 DAUs for classification

DAUs can be used directly for classification by applying k DAUs in parallel to obtain an output $\hat{\mathbf{y}}(\mathbf{x}) = [\mathrm{DAU}_1(\mathbf{x}),...,\mathrm{DAU}_k(\mathbf{x})]$. Note that this is a linear transformation $\hat{\mathbf{y}}(\mathbf{x})=\mathbf{W}(\mathbf{x})\mathbf{x}$, with each row in $\mathbf{W}\in\mathbb{R}^{k\times d}$ corresponding to the weight vector $\mathbf{w}_j^T$ of a specific DAU j. Consider, for example, a dataset $\mathcal{D}=\{(\mathbf{x}_i,\mathbf{y}_i)|\,\mathbf{x}_i\in\mathbb{R}^d,\mathbf{y}_i\in\mathbb{R}^k\}$ of k classes with 'one-hot' encoded labels $\mathbf{y}_i$ for the inputs $\mathbf{x}_i$. To optimise the DAUs as classifiers on $\mathcal{D}$, we can apply a sigmoid non-linearity to each DAU output and optimise the loss function $\mathcal{L}=\sum_i \mathrm{BCE}(\sigma(\hat{\mathbf{y}}_i),\mathbf{y}_i)$, where BCE denotes the binary cross-entropy and σ applies the sigmoid function to each entry in $\hat{\mathbf{y}}_i$. Note that for a given sample, BCE either maximises (DAU for correct class) or minimises (DAU for incorrect classes) the output of each DAU. Hence, this classification loss will still maximise the (signed) cosine between the weight vectors $\mathbf{w}(\mathbf{x}_i)$ and $\mathbf{x}_i$.

To illustrate this property, in Fig. 2 (top) we show the weights $\mathbf{w}(\mathbf{x}_i)$ for several samples of the digit '3' after optimising the DAUs for classification on a noisy MNIST dataset; the first two are correctly classified, the last one is misclassified as a '5'. As can be seen, the weights align with the respective input (the weights for different samples are different). However, different parts of the input are either positively or negatively correlated with a class, which is reflected in the weights: for example, the extended stroke on top of the '3' in the misclassified sample is assigned *negative weight* and, since the background noise is *uncorrelated* with the class labels, it is not represented in the weights.

In a classification setting, the DAUs thus preferentially encode *the most frequent discriminative patterns* in the linear transform $\mathbf{ABx}+\mathbf{b}$ such that the dynamic weights $\mathbf{w}(\mathbf{x})$ align well with these patterns. Additionally, since the output

1. Note that $\mathbf{w}(\mathbf{x})$ is proportional to $\mathbf{ABx}+\mathbf{b}$. The cosine in eq. (3), in turn, is maximal if and only if $\mathbf{w}(\mathbf{x}_i)$ is proportional to $\mathbf{x}_i$ and thus, by transitivity, if $\mathbf{x}_i$ is proportional to $\mathbf{ABx}_i+\mathbf{b}$. This means that $\mathbf{x}_i$ has to be an EV of $\mathbf{ABx}+\mathbf{b}$ to achieve maximal output.

2. Given $r=3$, the EVs maximally span a 3-dimensional subspace.

for class j is a linear transformation of the input (P1), we can compute the contribution vector $\mathbf{s}_j$ containing the per-pixel contributions to this output by the element-wise product ($\odot$)

$$\mathbf{s}_j(\mathbf{x}_i) = \mathbf{w}_j(\mathbf{x}_i) \odot \mathbf{x}_i \quad , \tag{7}$$

see Figs. 1 and 2. Such linear decompositions constitute the model-inherent 'explanations' which we evaluate in sec. 5.

3.4 Convolutional Dynamic Alignment Networks

The modelling capacity of a single layer of DAUs is limited, similar to a single linear classifier. However, DAUs can be used as the basic building block for deep convolutional neural networks, which yields powerful classifiers. Importantly, in this section we show that such a Convolutional Dynamic Alignment Network (CoDA Net) inherits the properties (P3) of the DAUs by maintaining both the dynamic linearity (P1) as well as the alignment maximisation (P2). For a *convolutional* dynamic alignment layer, each convolutional filter is modelled by a DAU, similar to dynamic local filtering layers [19]. Note that the output of such a layer is also a dynamic linear transformation of the input to that layer, since a convolution is equivalent to a linear layer with certain constraints on the weights, cf. [25]. We include the implementation details in the supplement. Finally, at the end of this section, we highlight an important difference between output maximisation and optimising for classification with the BCE loss. In this context we discuss the effect of *temperature scaling* and present the loss function we optimise in our experiments.

Dynamic linearity (P1). In order to see that the linearity is maintained, we note that the successive application of multiple layers of DAUs also results in a dynamic linear mapping. Let $\mathbf{W}_l$ denote the linear transformation matrix produced by a layer of DAUs and let $\mathbf{a}_{l-1}$ be the input vector to that layer; as mentioned before, each row in the matrix $\mathbf{W}_l$ corresponds to the weight vector of a single DAU[3]. As such, the output of this layer is given by

$$\mathbf{a}_l = \mathbf{W}_l(\mathbf{a}_{l-1})\mathbf{a}_{l-1} \quad . \tag{8}$$

In a network of DAUs, the successive linear transformations can thus be collapsed. In particular, *for any pair of activation vectors* $\mathbf{a}_{l_1}$ and $\mathbf{a}_{l_2}$ with $l_1 < l_2$, the vector $\mathbf{a}_{l_2}$ can be expressed as a linear transformation of $\mathbf{a}_{l_1}$:

$$\mathbf{a}_{l_2} = \mathbf{W}_{l_1 \to l_2}(\mathbf{a}_{l_1})\,\mathbf{a}_{l_1} \tag{9}$$

$$with \quad \mathbf{W}_{l_1 \to l_2}(\mathbf{a}_{l_1}) = \textstyle\prod_{k=l_1+1}^{l_2} \mathbf{W}_k(\mathbf{a}_{k-1}) \quad . \tag{10}$$

For example, the matrix $\mathbf{W}_{0\to L}(\mathbf{a}_0 = \mathbf{x}) = \mathbf{W}(\mathbf{x})$ models the linear transformation from the input to the output space, see Fig. 1. Since this linearity holds between any two layers, the j-th entry of any activation vector $\mathbf{a}_l$ in the network can be decomposed into input contributions via:

$$\mathbf{s}_j^l(\mathbf{x}_i) = [\mathbf{W}_{0\to l}(\mathbf{x}_i)]_j^T \odot \mathbf{x}_i \quad , \tag{11}$$

3. Note that this also holds for convolutional DAU layers. Specifically, each row in the matrix $\mathbf{W}_l$ corresponds to a single DAU applied to exactly one spatial location in the input and the input with spatial dimensions is vectorised to yield $\mathbf{a}_{l-1}$. For further details, we kindly refer the reader to [25] and the implementation details in the supplement of this work.

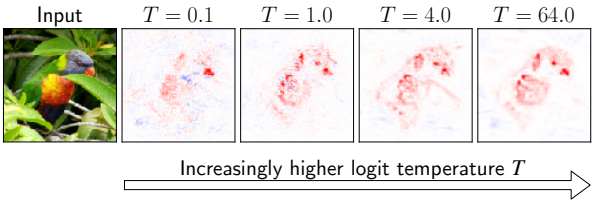


Fig. 4: By lowering the upper bound (cf. eq. (3)), the correlation maximisation in the DAUs can be emphasised. We show contribution maps for a model trained with different temperatures.

with $[\mathbf{W}]_j$ the j-th row in the matrix.

Alignment maximisation (P2). Note that the output of a CoDA Net is bounded independent of the network parameters: since each DAU operation can—independent of its parameters—at most reproduce the norm of its input (eq. (3)), the linear concatenation of these operations necessarily also has an upper bound which does not depend on the parameters. Therefore, in order to achieve maximal outputs on average (e.g., the class logit over the subset of images of that class), all DAUs in the network need to produce weights $\mathbf{w}(\mathbf{a}_l)$ that align well with the class features. In other words, the weights will align with discriminative patterns in the input. For example, in Fig. 2 (bottom), we visualise the 'global matrices' $\mathbf{W}_{0\to L}$ and the corresponding contributions (eq. (11)) for a $L = 5$ layer CoDA Net. As before, the weights align with discriminative patterns in the input and do not encode the uninformative noise.

Temperature scaling and loss function. So far we have assumed that minimising the BCE loss for a given sample is equivalent to applying a maximisation or minimisation loss to the individual outputs of a CoDA Net. While this is in principle correct, BCE introduces an additional, non-negligible effect: *saturation*. Specifically, it is possible for a CoDA Net to achieve a low BCE loss without the need to produce well-aligned weight vectors. As soon as the classification accuracy is high and the outputs of the networks are large, the gradient—and therefore the *alignment pressure*—will vanish. This effect can, however, easily be mitigated: as discussed in the previous paragraph, the output of a CoDA Net is upper-bounded *independent of the network parameters*, since each individual DAU in the network is upper-bounded. By scaling the network output with a temperature parameter T such that $\hat{\mathbf{y}}(\mathbf{x}) = T^{-1}\mathbf{W}_{0\to L}(\mathbf{x})\,\mathbf{x}$, we can explicitly decrease this upper bound and thereby increase the *alignment pressure* in the DAUs by avoiding the early saturation due to BCE. In particular, the lower the upper bound is, the stronger the induced DAU output maximisation should be, since the network needs to accumulate more signal to obtain large class logits (and thus a negligible gradient). This is indeed what we observe both qualitatively, cf. Fig. 4, and quantitatively, cf. Fig. 6 (right column). The overall loss for an input $\mathbf{x}_i$ and the target vector $\mathbf{y}_i$ is thus computed as

$$\mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) = \mathrm{BCE}(\sigma(T^{-1}\mathbf{W}_{0\to L}(\mathbf{x}_i)\,\mathbf{x}_i + \mathbf{b}_0)\,,\,\mathbf{y}_i) \quad . \tag{12}$$

Here, σ applies the sigmoid activation to each vector entry and $\mathbf{b}_0$ is a fixed bias term. As an alternative to the temperature scaling, the explicit representation of the network's computation as a linear mapping allows to directly

regularise what properties these linear mappings should fulfill. For example, we show in the supplement that by regularising the absolute values of the matrix $\mathbf{W}_{0\to L}$, we can induce sparsity in the signal alignments, which also leads to sharper heatmaps.

4 Experimental setup

4.1 Datasets

We evaluate and compare the accuracies of the CoDA Nets to other work on the CIFAR-10 [26] and the Tiny-Imagenet [27] datasets. We use the same datasets for the quantitative evaluations of the model-inherent contribution maps. Additionally, we qualitatively show high-resolution examples from a CoDA Net trained on the first 100 classes of the Imagenet [28] dataset. Lastly, we evaluate hybrid models (see sec. 4.4) on CIFAR10 and the full Imagenet dataset, both in terms of interpretability as well as classification accuracy.

4.2 Models

Our results (secs. 5.1–5.3) are based on models of various sizes denoted by (S/L/XL)-CoDA on CIFAR-10 (S), Imagenet-100 (L), and TinyImagenet (XL); these models have 7-8M[4] (S), 48M (L), and 62M (XL) parameters respectively; see the supplement for details on the model architectures and an evaluation of the impact of model size on accuracy. For the hybrid networks (see secs. 4.4 and 5.4), we use a ResNet-56 (ResNet-50) as a base model on CIFAR-10 (Imagenet) and train CoDA Nets on feature maps extracted at different depths of the base models; see the supplement for details.

4.3 Input encoding

In sec. 3.1, we discussed that the norm-weighted cosine similarity between the dynamic weights and the layer inputs is optimised and the output of a DAU is at most the norm of its input. When using pixels as the input to the CoDA Nets, this favours pixels with large RGB values, since these have a larger norm and can thus produce larger outputs in the maximisation task. In our experiments, we explore two approaches to mitigate this bias: in the first, we add the negative image as three additional color channels and thus encode each pixel in the input as $[r, g, b, 1-r, 1-g, 1-b]$, with $r, g, b \in [0, 1]$.

Secondly, we show that it is also possible to train CoDA Nets on end-to-end optimised patch-embeddings and obtain similar performance in terms of interpretability and classification accuracy. Instead of computing the *per-pixel* contributions to assess the importance of spatial locations (cf. eq. (11)), in our experiments we thus decompose the output with respect to the contributions from the corresponding (learnt) embeddings via

$$\mathbf{s}_j^L(\mathbf{x}_i) = [\mathbf{W}_{0\to L}(E(\mathbf{x}_i))]_j^T \odot E(\mathbf{x}_i) \quad , \tag{13}$$

with $E(\cdot)$ denoting the applied embedding function and L the number of CoDA layers in the network.

4. The models with SQ and L2 non-linearity have 7.8M parameters and the models with WB have 7.1M (without embedding) and 7.2M (with embedding) parameters.

4.4 Interpolating between networks

Training CoDA Nets on learnt patch-embeddings (see sec. 4.3) naturally raises the question of how complex the embedding function should be and how large its receptive field. In particular, are *pixel-wise* importance values more useful than importance values for embeddings of patches of size 3×3? How about 7×7 or 64×64? Of course, there is no single answer to this question and the 'optimal' complexity of the embedding model depends on the dataset, the task, and, ultimately, on the preferences of the end-user of such a model: for example, if a more complex embedding allows for more performant classifiers, one might wish to trade off model interpretability against model accuracy. In order to better understand such trade-offs, we propose to 'interpolate' between a conventional CNN and the CoDA Nets and investigate how this affects both model interpretability and model performance. Specifically, starting from a pre-trained CNN, we successively replace an increasing number of the later layers of the base model by CoDA layers. As such, the model output can be decomposed into *contributions coming from spatially arranged embeddings* computed by the truncated CNN model, which can give insights into how the embeddings are used to produce the classification results.

4.5 Additional details

In our experiments, we observed that rescaling the weight vectors of the DAUs explicitly according to eq. (2) resulted in long training times and high memory usage. To mitigate this, we opted to share the matrix $\mathbf{B}$ between all DAUs in a given layer when using the L2 or SQ non-linearity. This increases efficiency by having the DAUs share a common r-dimensional subspace and still fixes the maximal rank of each DAU to the chosen value of r. In contrast, networks with the WB non-linearity (see eDAUs in eq. (5)) are specifically designed to lower the computational costs of the DAUs and are easier to train. Therefore, for CoDA Nets built with eDAUs, we do not share the matrices $\mathbf{B}$ between the eDAUs. As the inputs are thus not restricted to a common low-dimensional subspace, we expect this to increase the modelling capacity of the CoDA Nets.

5 Experiments

In sec. 5.1 we assess the classification performance of the CoDA Nets. Further, in sec. 5.2 we evaluate the model-inherent contribution maps derived from $\mathbf{W}_{0\to L}$ (cf. eq. (13)) of a CoDA Net and compare them both *qualitatively* (cf. Fig. 5) as well as *quantitatively* (cf. Fig. 6) to other attribution methods. Additionally, in sec. 5.3, we discuss the impact of the different rescaling methods (cf. eqs. (2) and (5)) on model interpretability and evaluation speed. Lastly, in sec. 5.4 we investigate the hybrid models discussed in sec. 4.4. In particular, we analyse how the depth of the embedding function $E(\mathbf{x})$ affects the model interpretability at different depths of the resulting hybrid network architecture.

5.1 Model performance

Classification performance. In Table 1 we compare the performances of our CoDA Nets to several other published

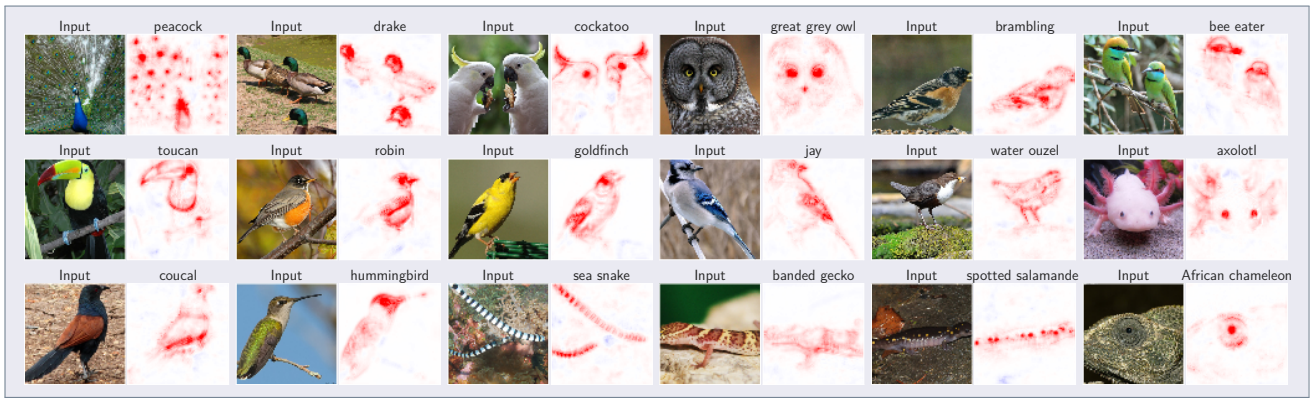


Fig. 5: Model-inherent contribution maps for the most confident predictions for 18 different classes, sorted by confidence (high to low). We show positive (negative) contributions (eq. (11)) per spatial location for the ground truth class logit in red (blue).

Model	C10	Model	T-IM
SENNs [18]	78.5%	ResNet-34 [29]	52.0%
DE-CapsNet [30]	93.0%	VGG 16 [31]	52.2%
VGG-19 [32]	93.4%	VGG 16 + aug [31]	56.4%
ResNet-110 [33]	93.6%	IRRCNN [34]	52.2%
DenseNet [35]	94.8%	ResNet-110 [36]	56.6%
WRN-28-2 [37]	94.9%	WRN-40-20 [38]	63.8%
S-CoDA-SQ	93.2%	XL-CoDA-SQ	54.4%
S-CoDA-L2	93.0%	XL-CoDA-SQ + aug	58.4%
S-eCoDA-WB	94.0%		
S-eCoDA-WB + $E(\mathbf{x})$	94.1%		

TABLE 1: CIFAR-10 (**C10**) and TinyImagenet (**T-IM**) classification accuracies. Results taken from specified references. The prefix of the CoDAs indicates model size, the suffix the non-linearity used (cf. eqs. (2) and (5)). Further, $E(\mathbf{x})$ denotes that a learnt embedding was used as input to the model (see sec. 4.3).

results. Note that the referenced numbers are meant to be used as a gauge for assessing the CoDA Net performance and do not exhaustively represent the state of the art. In particular, we would like to highlight that the CoDA Net performance is on par to that of models from the VGG [39] and ResNet [33] model families on both datasets. Additionally, we list the reported results of the SENNs [18] and the DE-CapsNet [30] architectures for CIFAR-10. Similar to our CoDA Nets, the SENNs were designed to improve network interpretability and are also based on the idea of explicitly modelling the output as a dynamic linear transformation of the input. On the other hand, the CoDA Nets share similarities to capsule networks, which we discuss in the supplement; to the best of our knowledge, the DE-CapsNet currently achieves the state of the art in the field of capsule networks on CIFAR-10. Overall, we observed that the CoDA Nets deliver competitive performances that are fairly robust to the non-linearity (see eqs. (2) and (5)) and the temperature (T); for an ablation study on the latter, see the supplement. Finally, while all models achieve good classification results, we note that the WB-based CoDA Nets perform slightly better than CoDA Nets with SQ or L2 non-linearity despite having a comparable amount of parameters. As discussed in sec. 4.5, we attribute this to the fact that for those models we do not share the matrix $\mathbf{B}$ within the CoDA layers, which increases their modelling capacity.

5.2 Interpretability of CoDA Nets

In the following, we evaluate the model-inherent contribution maps and compare them to other commonly used methods for importance attribution. The evaluations are based on the XL-CoDA-SQ (T=6400) for TinyImagenet and the S-eCoDA-WB (T=$1e6$) for CIFAR-10, see Table 1 for the respective accuracies. The results are very similar for all three non-linearities (cf. sec. 5.3; more results are included in the supplement) and we therefore just show one per dataset as an example. Further, we evaluate the effect of training the S-CoDA-SQ architecture with different temperatures T; as discussed in sec. 3.4, we expect the interpretability to *increase* along with T, since for larger T a stronger alignment is required in order for the models to obtain large class logits. Lastly, in sec. 5.3 we compare how the different non-linearities (L2, SQ, WB) affect the model interpretability. Before turning to the results, however, in the following we will first present the attribution methods used for comparison and discuss the evaluation metrics employed for quantifying their interpretability.

Attribution methods. We compare the model-inherent contribution maps (cf. eq. (11)) to other common approaches for importance attribution. In particular, we evaluate against several perturbation based methods such as RISE [14], LIME [15], and several occlusion attributions [40] (Occ-K, with K the size of the occlusion patch). Additionally, we evaluate against common gradient-based methods. These include the gradient of the class logits with respect to the input image [41] (Grad), 'Input×Gradient' (IxG, cf. [3]), GradCam [7] (GCam), Integrated Gradients [9] (IntG), and DeepLIFT [8]. As a baseline, we also evaluated these methods on a pre-trained ResNet-56 [33] on CIFAR-10, for which we show the results in the supplement.

Evaluation metrics. Our quantitative evaluation of the attribution maps is based on the following two methods: we **(1)** evaluate a localisation metric by adapting the pointing game [42] to the CIFAR-10 and TinyImagenet datasets, and **(2)** analyse the model behaviour under the pixel removal strategy employed in [10]. For **(1)**, we evaluate the attribution methods on a grid of $n \times n$ with $n = 3$ images sampled from the corresponding datasets; in every grid of images, each class may occur at most once. For a visualisation with $n = 2$, see Fig. 7. For each occurring class, we can measure how much positive importance an attribution method assigns to the respective class image. Let $\mathcal{I}_c$ be the image for class c, then the score s_c for this class is calculated

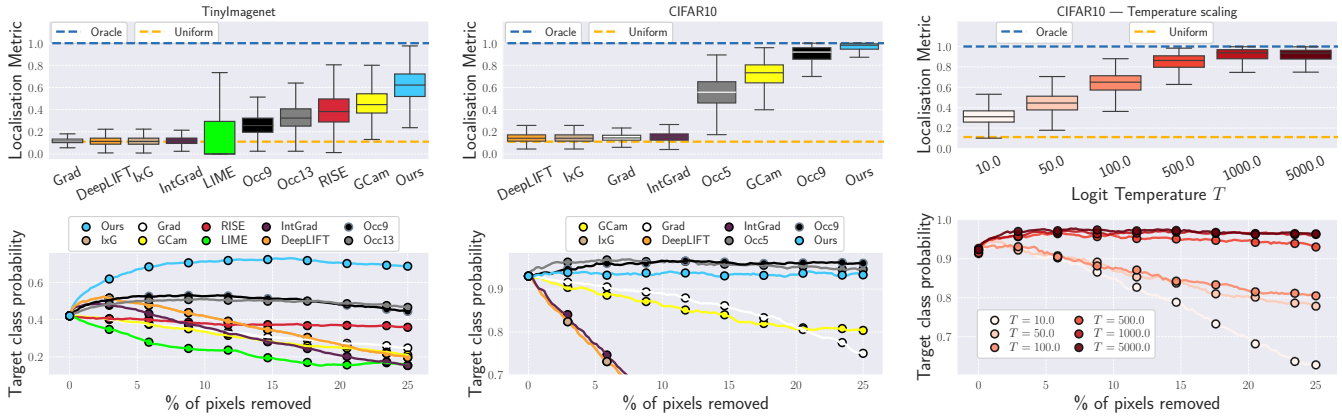


Fig. 6: **Top row:** Results for the localisation metric, see eq. (14). **Bottom row:** Pixel removal metric. In particular, we plot the mean target class probability after removing the $x\%$ of the *least important* pixels. We show the results of a CoDA-Net-SQ trained on TinyImagenet **(left column)**, as well as of a CoDA-Net-WB trained on CIFAR-10 **(center column)**. We observed the results for different rescaling methods (SQ/L2/WB) to be very similar and therefore just show one per dataset; more results can be found in the supplement. Additionally, we show the effect of the temperature parameter on the interpretability of a CoDA-Net with SQ rescaling **(right column)**: as expected, a higher temperature leads to higher interpretability (sec. 3.4).

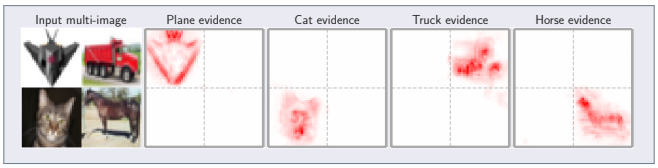


Fig. 7: A multi-image on the CIFAR-10 dataset. The CoDA-Net contribution maps highlight the individual class-images well.

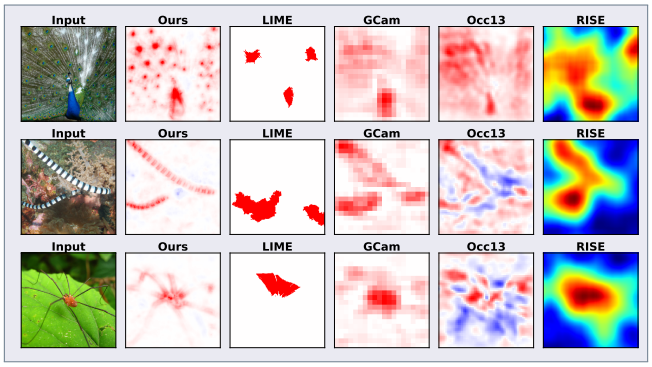


Fig. 8: Comparison to the strongest post-hoc methods. While the regions of importance roughly coincide, the inherent contribution maps of the CoDA-Nets offer the most detail. Note that to improve the RISE visualisation, we chose its default colormap [14]; the most (least) important values are still shown in red (blue).

as

$$s_c = \frac{1}{Z}\sum_{p_c \in \mathcal{I}_c} p_c \quad \text{with} \quad Z = \sum_k \sum_{p_c \in \mathcal{I}_k} p_c \quad , \qquad (14)$$

with p_c the positive attribution for class c assigned to the spatial location p. This metric has the same clear oracle score $s_c = 1$ for all attribution methods (all positive attributions located in the correct grid image) and a clear score for completely random attributions $s_c = 1/n^2$ (the positive attributions are uniformly distributed over the different grid images). Since this metric depends on the classification accuracy of the models, we sample the first 500 (CIFAR-10) or 250 (TinyImagenet) images according to their class score for the ground-truth class[5]; note that since all attributions are evaluated for the same model on the same set of images, this does not favour any particular attribution method.

For (2), we show how the model's class score behaves under the removal of an increasing amount of *least important* pixels, where the importance is obtained via the respective attribution method. Since the first pixels to be removed are typically assigned negative or relatively little importance, we expect the model to initially increase its confidence (removing pixels with *negative* impact) or maintain a similar level of confidence (removing pixels with *low* impact) if the evaluated attribution method produces an accurate ranking of the pixel importance values. Conversely, if we were to remove the *most important* pixels first, we would expect the model confidence to quickly decrease. However, as noted by [10], removing the most important pixels first introduces artifacts in the most important regions of the image and is therefore potentially more unstable than removing the least important pixels first. Nevertheless, the model-inherent contribution maps perform well in this setting, too, as we show in the supplement. Lastly, in the supplement we qualitatively show that they pass the 'sanity check' of [3].

Quantitative results. In Fig. 6, we compare the contribution maps of the CoDA Nets to other attributions under the evaluation metrics discussed above. It can be seen that the CoDA Nets (1) perform well under the localisation metric given by eq. (14) and outperform all the other attribution methods evaluated on the same model, both for TinyImagenet (top row, left) and CIFAR-10 (top row, center); note that we excluded RISE and LIME on CIFAR-10, since the default parameters do not seem to transfer well to this low-resolution dataset. Moreover, (2) the CoDA Nets perform well in the pixel-removal setting: the *least salient* locations according to the model-inherent contributions indeed seem

5. We can only expect an attribution to specifically highlight a class image if this image can be correctly classified on its own. If all grid images have similarly low attributions, the localisation score will be random.

to be among the least relevant for the given class score on both datasets, see Fig. 6 (bottom row, left and center); note that the Occ-K explanations directly estimate the impact of occluding pixels and are thus expected to perform well under this metric. Further, in Fig. 6 (right column), we show the effect of temperature scaling on the interpretability of CoDA Nets with SQ rescaling trained on CIFAR-10. The results indicate that the alignment maximisation is indeed crucial for interpretability and constitutes an important difference of the CoDA Nets to other dynamic linear networks such as piece-wise linear networks (ReLU-based networks). In particular, by structurally requiring a strong alignment for confident classifications, the interpretability of the CoDA Nets forms part of the optimisation objective. Increasing the temperature increases the alignment and thereby the interpretability of the CoDA Nets. While we observe a downward trend in classification accuracy when increasing T, the best model at $T = 10$ only slightly improved the accuracy compared to $T = 1000$ ($93.2\% \rightarrow 93.6\%$); for more details, see supplement.

In summary, the results show that by combining dynamic linearity with a structural bias towards an alignment with discriminative patterns, we obtain models which inherently provide an interpretable linear decomposition of their predictions. Further, given that we better understand the relationship between the intermediate computations and the optimisation of the final output in the CoDA Nets, we can emphasise model interpretability in a principled way by increasing the 'alignment pressure' via *temperature scaling*.

Qualitative results. In Fig. 5, we visualise spatial contribution maps of an L-CoDA-SQ model (trained on Imagenet-100) for some of its most confident predictions. Note that these contribution maps are linear decompositions of the output and the sum over these maps yields the respective class logit. In Fig. 8, we additionally present a visual comparison to the best-performing post-hoc attribution methods; note that RISE cannot be displayed well under the same color coding and we thus use its default visualisation. We observe that the different methods are not inconsistent with each other and roughly highlight similar regions. However, the inherent contribution maps are of much higher detail and compared to the perturbation-based methods do not require multiple model evaluations. Much more importantly, however, all the other methods are attempts at approximating the model behaviour *post-hoc*, while the CoDA Net contribution maps in Fig. 5 are derived from the model-inherent linear mapping that is used to compute the model output.

5.3 Interpretability and efficiency of L2, SQ, and WB

In the following, we present results regarding the impact of the different normalisation methods (L2, SQ, WB) on model interpretability and model efficiency.

Model interpretability. In Fig. 9, we show the results of the interpretability metrics for models with different rescaling functions (L2/SQ/WB, see eqs. (2) and (5)) as well as for a model trained with a learnt patch-embedding $E(\mathbf{x})$. As an embedding function, we simply apply a 3x3 convolution with 32 filters, followed by a batch normalisation layer [43]. For comparison to post-hoc methods evaluated on a CoDA Net, we kindly refer the reader to the center column of Fig. 6. As can be seen, it is possible to obtain highly interpretable models under all four settings: (1) the linear contributions allow to localise the class-images well (localisation metric, left) and (2) the models are insensitive to **input features**[6] that are not contributing to the output as per the linear transformation matrix $\mathbf{W}_{0\rightarrow L}$. Note that for the model with a learnt patch-embedding, denoted by E–WB, we show two results for the perturbation metric. First, we 'zero out' the *embeddings* at each location ordered by their assigned importance (blue crosses). As the embeddings are the input features to the CoDA Net, the model confidence shows the expected behaviour of being insensitive to unimportant inputs. In contrast, the assigned importance values do not translate to the center pixels of the embeddings: when zeroing out the *center pixels* according to the contributions of the patch-embeddings, the model confidence drops more quickly (see red crosses). This distinction is important to keep in mind when evaluating CoDA Nets on input embeddings, since it is easy to wrongly interpret such contribution maps. If the input to the CoDA Net is an embedding of an image patch, it depends on the embedding function how the contributions are to be distributed to the image pixels. Lastly, note that different from the center column in Fig. 6, the metrics are evaluated for *four different models* and are thus not comparisons between different explanation methods, but rather between different models under the same explanation. As such, the differences in the localisation metric, for example, do not show that the linear decompositions are generally better suited to explain WB-based models as compared to SQ- or L2-based models; the differences might instead reflect the fact that the models learnt more robust and class-specific representations, which yield both better results in the localisation task as well as higher classification accuracy.

Model efficiency. While all three non-linearities can yield interpretable CoDA Nets, the computational cost of the different approaches for bounding the DAU outputs differs. For example, by avoiding the explicit calculation of the d-dimensional weight vector in eq. (5), the eDAUs are able to save both memory as well as floating point operations—the computed vectors are of size $r \ll d$ and the dot-product in the low-dimensional space requires $O(r)$ operations instead of $O(d)$. Being the fundamental building block of the CoDA Nets, such gains in efficiency can have considerable impact, since the corresponding computations are performed in every layer for every unit and at each spatial position of the input to the respective layer. Accordingly, in practice[7] we observed that the weight bounding approach in the eDAUs (eq. (5)) can yield significant speed-ups and memory savings, especially for high-dimensional inputs. For example, in Fig. 10 we plot the memory consumption and forward-pass speeds for the three different models without learnt embedding function (see Table 1) for varying batch sizes on the CIFAR-10 dataset: while SQ and L2 perform similarly, the WB-based model scales better to larger inputs.

6. For the SQ, L2, and WB model the features are pixels under the static encoding function described in 4.3. For the E-WB model, the input features to the CoDA Net are *learnt* patch embeddings.

7. In our experiments, we rely on the highly optimised implementations for convolutions from the pytorch [44] library.

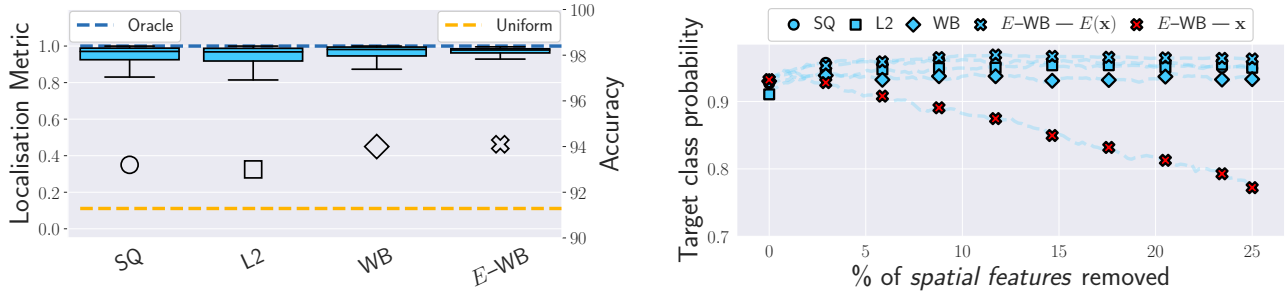


Fig. 9: Localisation (**left**) and perturbation (**right**) metric results on CIFAR10 evaluated for CoDA Networks with different non-linearities (cf. eqs. (2) and (5)) as well as trained with a learnt patch-embedding $E(\mathbf{x})$, here denoted by E–WB. For the model with embedding function $E(\mathbf{x})$, we evaluate the pixel perturbation metric (bottom) directly on the pixels (red crosses) as well as on the learnt patch-embeddings (blue crosses). We further added the models' accuracies (see Table 1) in the plot to the left for comparison (circle / square/ cross markers).

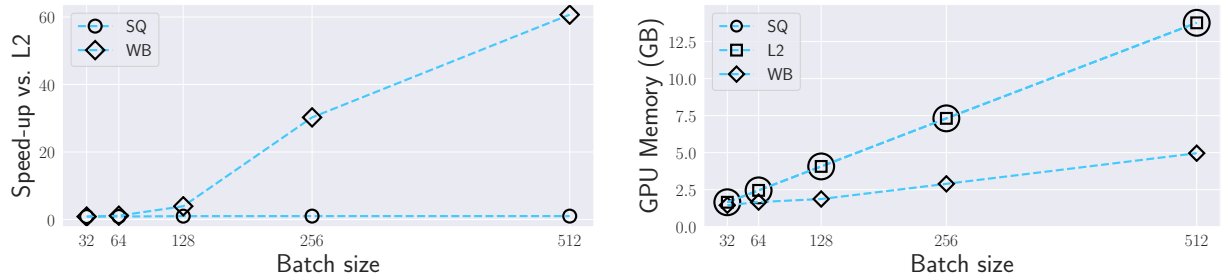


Fig. 10: We show the speed-up per forward pass of the models with WB and SQ rescaling compared to L2 (**left**) as well as the GPU memory consumption (**right**) of the three different models in Table 1 for different batch sizes for both measures. While the models perform similarly for small batch sizes, the WB-based model scales better to large inputs.

Additionally, we measured memory consumption and training time for two models with the same architecture on Imagenet (L-CoDA, see beginning of sec. 5) for the SQ and the WB rescaling methods. For this, we updated the models ≈ 8000 times with a batch size of 16 and recorded the overall time as well as the GPU memory consumption. In these experiments, the WB-based model required more than $3\times$ less memory (9.7GB vs. 30.0GB) and completed the updates more than $1.5\times$ faster (8.7 minutes vs. 14.1 minutes). All experiments regarding evaluation speed were performed on an nvidia Quadro RTX 8000 GPU with 48GB of memory.

5.4 Hybrid CoDA Networks

In this section, we assess the interpretability of hybrid CoDA Nets, which combine conventional CNN layers and CoDA layers in one network model. For our experiments, we use varying numbers of pre-trained CNN layers as feature extractors on top of which a CoDA Net is trained as a classifier. Such a hybrid structure can prove useful in cases where CoDA Nets do not (yet) yield the same accuracy as conventional architectures; we kindly refer the reader to the supplement for details on the network architectures used in these experiments.

In particular, we use the first K layers of a pre-trained ResNet model [33] as feature extractors. Since ResNets are piece-wise linear models, the hybrids are still *dynamic linear* and we can assign importance values to input features according to their effective linear contribution; importantly, the input features can be extracted *at any depth* of the network as the output of a CoDA layer or a ResNet block, or as the actual input pixels. In order to assess whether such hybrids are more interpretable than the base model, we compute spatial contribution maps[8] with respect to different activation maps within the network and evaluate them under the localisation metric (see sec. 5.2, 'Evaluation metrics').

CIFAR10. For the following experiments, we use a pre-trained ResNet-56 obtained from [45]. This model consists of a convolutional layer + batch normalisation [43] (C+B), followed by three times nine *residual blocks* (RBs) as well as a fully connected and a pooling (FC+P) layer; for more details we kindly refer the reader to the original work [33] and the implementation [45] on which we base these experiments. We can summarise this model by [C+B, 9RB, 9RB, 9RB, FC+P]; we will further denote individual *segments* $\mathcal{S}_i$ of the model by their index in this summary counting from the back, e.g., $\mathcal{S}_5$=[C+B] and $\mathcal{S}_1$=[FC+P]. In order to evaluate the interpretability of this model at different depths t, we split it at different points into two virtual parts: an embedding function $E_t(\mathbf{x})$ and a classification head (CH_t). For a given split, we then regard the output of $E_t(\mathbf{x})$ as the input to the classification head and linearly decompose the latter according to the respective

8. The contribution maps according to the dynamic linear mapping can be obtained via 'Input×Gradient', where for the gradient calculation we treat the dynamic matrices in the CoDA layers as fixed.

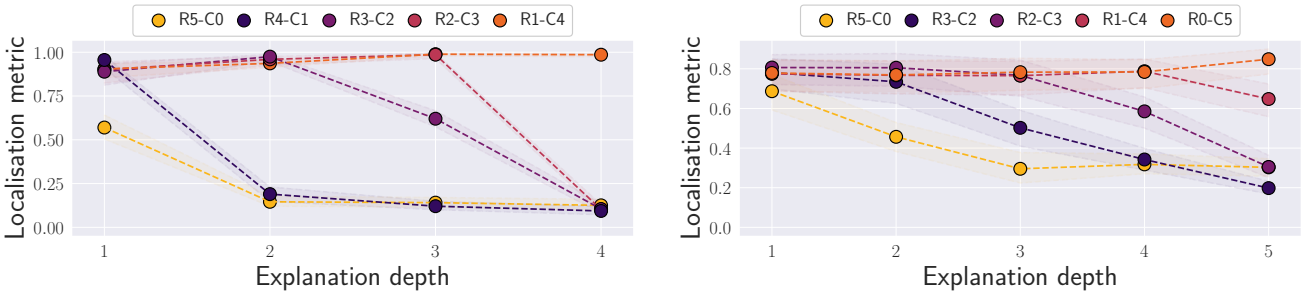


Fig. 11: Localisation metric (mean and standard deviation) for different 'explanation depths' evaluated on four hybrid models trained on CIFAR-10 (**left**) and Imagenet (**right**). Additionally, we show the localisation results of the pretrained ResNets (denoted by R5-C0) that were used as base models. For each evaluation, we extract the effectively applied linear transformations up to a certain depth and compute the corresponding linear contributions to the output logits coming from individual positions in the activation maps. We then use the resulting maps as an explanation of the output logit and assess how well these maps allow for localising the corresponding class images in the localisation task. As can be seen, the more layers of the original ResNet architectures are replaced by CoDA layers, the larger is the 'interpretable depth'.

linear transformation performed by the model; e.g., with an *explanation depth* of 2 we refer to the split in which $\mathcal{S}_2$ is the first element in CH_2 and we evaluate linear contribution maps obtained for the classification head CH_2=[9RB, FC+P] on the preprocessed input $E_2(\mathbf{x})$=[C+B, 9RB, 9RB]$(\mathbf{x})$. By performing this evaluation for various splits, we can assess the 'interpretable depth' of a model. In particular, we evaluate how well the contribution maps at different depths allow for localising the correct class images in the localisation task.

In order to investigate the effect of CoDA layers on the interpretable depth, we train and evaluate four different hybrid models. For this, we replace an increasing number of segments $\mathcal{S}_i$ by CoDA layers, starting from $\mathcal{S}_1$; in Fig. 11 (**left**), we denote the base model by R5-C0 (5 ResNet segments, 0 CoDA segments) and the hybrids according to the number of replaced segments, e.g., for R3-C2 we replaced the last two segments by CoDA layers[9]. *For each of these models*, we can decompose the model outputs in terms of contributions from spatial positions for the embedding functions E_t defined by different splits t. Note that the base model (ResNet-56) is piece-wise linear and we can thus still compute linear contributions at any depth of this hybrid network. In Fig. 11 (**left**) we show the results of the localisation metric for all networks at various depths; the classification accuracies can be found in Table 2. As can be seen, the linear contributions are good explanations of the class logits as long as the classification head entirely consists of CoDA layers and drops as soon as we include a segment with ResNet blocks in the classification head CH. Again, this highlights that *dynamic linearity* alone is not enough to obtain useful linear decompositions of the model outputs, but that the alignment property is crucial for the interpretability of the CoDA layers.

Imagenet. In the following, we show that the gains from interpolating between networks also extend to a more complex dataset. Similar to the interpolation experiments on CIFAR-10 above, the results are based on an interpolation between a pretrained ResNet model (ResNet-50) and a CoDA-based classification head. However, given the high-dimensional representations produced by the later ResNet layers (up to 2048 channels), the parameters of the classification head increase drastically if the high dimensionality is maintained throughout the CoDA layers. Therefore, in the Imagenet experiments, we first compute a low-dimensional projection $\tilde{\mathbf{x}} = \mathbf{P}\mathbf{x}$ of the inputs to the convolutional kernels to which we apply the eDAUs (see eq. (5)); similarly to the dynamic weights of DAUs with L2 normalisation, we normalise the rows of the matrix $\mathbf{P}$ to unit norm to maintain a parameter-independent bound of the network.

For the interpolation experiments, we successively replace the last 5 residual blocks of the ResNet-50 base model (the 'segments' here correspond to FC+P or individual residual blocks) by a single CoDA layer each and assess the interpretability via the localisation metric as well as model accuracy, see Fig. 11 (**right**) and Table 2 respectively. Similar to the CIFAR-10 experiments, we observe an increase in 'interpretable depth' (Fig. 11, right). However, while on CIFAR-10 the accuracy of the base model could be maintained, on Imagenet we observe a trade-off in accuracy. While better results can certainly be achieved by further optimising the network architectures and / or fine-tuning the learnt embeddings, our results show that it is possible

TABLE 2: Classification accuracies for hybrid networks. RX-CY denotes how many segments were replaced by CoDA layers; on Imagenet, maximally up to five ResNet blocks from the end of the network were replaced and all networks thus still rely on a ResNet-based stem. On CIFAR-10 the accuracy can be maintained whilst improving interpretability (see Fig. 11). On Imagenet, on the other hand, we observe a trade-off in accuracy when increasing the 'interpretable depth' of the models.

CIFAR-10	R5-C0	R4-C1	R3-C2	R2-C3	R1-C4
	93.4%	93.6%	93.4%	93.6%	93.8%
Imagenet	R5-C0	R3-C2	R2-C3	R1-C4	R0-C5
	76.1%	74.7%	73.3%	71.7%	71.4%

9. In detail, each segment of 9RB is replaced by a set of 3 CoDA layers. The final network segment [FC+P] is replaced by a single CoDA layer followed by a global pooling operation.

to increase the interpretability of performant classification models by using a classification head comprised of CoDA layers.

6 Discussion and conclusion

We present a new family of neural networks, the CoDA Nets, and show that they are performant classifiers with a high degree of interpretability[10]. In particular, we first introduced the Dynamic Alignment Units (DAUs), which model their output as a dynamic linear transformation of their input and have a structural bias towards alignment maximisation. This bias is induced by ensuring that a DAU can only produce large outputs if its weights are well-aligned with the input, since the dynamically applied weights are explicitly normalised. In order to lower the computational costs of the DAUs, we further introduce the eDAUs, for which we normalise the weights by an upper bound of their norms which is cheaper to compute. Using the DAUs to model filters in a convolutional network, we obtain the Convolutional Dynamic Alignment Networks (CoDA Nets). The successive linear mappings by means of the DAUs within the network make it possible to linearly decompose the output into contributions from individual input dimensions—in contrast to piece-wise linear networks, which are also dynamic linear, the alignment property of the DAUs ensures that the linear decomposition aligns with discriminant patterns in the input. In order to assess the quality of these contribution maps, see eq. (11), we compare against other attribution methods. We find that the CoDA Net contribution maps consistently perform well under commonly used quantitative metrics and are robust to the applied normalisation scheme. Beyond their *interpretability*, the CoDA Nets constitute performant classifiers: their accuracy on CIFAR-10 and the TinyImagenet dataset are on par to the commonly employed VGG and ResNet models. Lastly, we show that CoDA layers can be combined with conventional networks, which yields hybrid models with an increased 'interpretable depth' compared to the base model. We believe that such hybrid models hold great potential, since they take advantage of the high modelling capacity and efficiency of modern neural networks whilst allowing for a user-defined 'minimal interpretability'. For example, such networks could allow for localising regions of importance for the model decision at a desired granularity by restricting the receptive field of the feature extractors.

10. Code is available at github.com/moboehle/CoDA-Nets

References

[1] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *International Conference on Machine Learning (ICML)*, 2010.

[2] G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the Number of Linear Regions of Deep Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[3] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, "Sanity Checks for Saliency Maps," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," in *International Conference on Learning Representations (ICLR), Workshop*, 2014.

[5] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for Simplicity: The All Convolutional Net," in *International Conference on Learning Representations (ICLR), Workshop*, 2015.

[6] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *International Conference on Computer Vision (ICCV)*, 2017.

[8] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features Through Propagating Activation Differences," in *International Conference on Machine Learning (ICML)*, 2017.

[9] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *International Conference on Machine Learning (ICML)*, 2017.

[10] S. Srinivas and F. Fleuret, "Full-Gradient Representation for Neural Network Visualization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[11] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLoS ONE*, 2015.

[12] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy*, vol. 23, no. 1, p. 18, 2021.

[13] S. M. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] V. Petsiuk, A. Das, and K. Saenko, "RISE: Randomized Input Sampling for Explanation of Black-box Models," in *British Machine Vision Conference (BMVC)*, 2018.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the predictions of any classifier," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2016.

[16] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. Su, "This Looks Like That: Deep Learning for Interpretable Image Recognition," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[17] W. Brendel and M. Bethge, "Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet," in *International Conference on Learning Representations (ICLR)*, 2019.

[18] D. Alvarez-Melis and T. S. Jaakkola, "Towards Robust Interpretability with Self-Explaining Neural Networks," in *Advances in Neural Information Processing (NeurIPS)*, 2018.

[19] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[20] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning (ICML)*, 2015.

[21] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, 1936.

[22] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[24] Y. LeCun, "MNIST handwritten digit database," *http://yann.lecun.com/exdb/mnist*, 1998.

[25] I. Shafkat, "Intuitively Understanding Convolutions for Deep Learning," https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1#ad33, 2018.

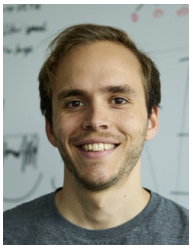
[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[27] J. et al., "Tiny ImageNet Visual Recognition Challenge," https://tiny-imagenet.herokuapp.com/, accessed: 2020-11-10.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[29] L. Sun, "ResNet on Tiny ImageNet," http://cs231n.stanford.edu/reports/2017/pdfs/12.pdf, 2016, accessed: 2020-11-16.

[30] B. Jia and Q. Huang, "DE-CapsNet: A Diverse Enhanced Capsule Network with Disperse Dynamic Routing," *Applied Sciences*, 2020.
[31] learningai.io, "VGGNet and Tiny ImageNet," https://learningai.io/projects/2017/06/29/tiny-imagenet.html, accessed: 2020-11-08.
[32] T. Li, J. Li, Z. Liu, and C. Zhang, "Few sample knowledge distillation for efficient network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 639–14 647.
[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[34] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Improved inception-residual convolutional neural network for object recognition," *Neural Computing and Applications*, 2020.
[35] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017.
[36] C. Termritthikun, Y. Jamtsho, and P. Muneesawang, "An improved residual network model for image recognition using a combination of snapshot ensembles and the cutout technique," *Multimedia Tools and Applications*, 2020.
[37] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in *British Machine Vision Conference (BMVC)*, 2016.
[38] D. Hendrycks, K. Lee, and M. Mazeika, "Using Pre-Training Can Improve Model Robustness and Uncertainty," in *Proceedings of Machine Learning Research (PMLR)*, 2019.
[39] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
[40] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *European Conference on Computer Vision (ECCV)*, 2014.
[41] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller, "How to explain individual classification decisions," *The Journal of Machine Learning Research*, 2010.
[42] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-Down Neural Attention by Excitation Backprop," *Int. J. Comput. Vis.*, 2018.
[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019.
[45] Y. Idelbayev, ""Proper ResNet Implementation for CIFAR10/CIFAR100 in PyTorch","https://github.com/akamaster/pytorch_resnet_cifar10, accessed: 2020-06-05.



Moritz Böhle is a Ph.D. student in Computer Science at the Max Planck Institute for Informatics, working with Prof. Dr. Bernt Schiele and Prof. Dr. Mario Fritz. He graduated with a bachelor's degree in physics in 2016 from the Freie Universität Berlin and obtained his Master's degree in computational neuroscience in 2019 from the Technische Universität Berlin. His research focuses on understanding the 'decision process' in deep neural networks and designing inherently interpretable neural network models.

Mario Fritz Mario Fritz is faculty member at the CISPA Helmholtz Center for Information Security, honorary professor at the Saarland University, and a fellow of the European Laboratory of Learning and Intelligent Systems (ELLIS). Before, he was senior researcher at the Max Planck Institute for Informatics, PostDoc at UC Berkeley and International Computer Science Institute. He studied computer science at the university Erlangen/Nuremberg and obtained his PhD from the TU Darmstadt. His current work is centered around Trustworthy Information Processing with a focus on the intersection of AI & Machine Learning with Security & Privacy. He is associate editor of IEEE TPAMI, a member of the ACM Europe Technical Policy Committee Europe, and a leading scientist of the Helmholtz Medical Security, Privacy, and AI Research Center, where he is coordinating projects on privacy and federated learning in health. He has over 100 publications, including 80 in top-tier journals (IJCV, TPAMI) and conferences (NeurIPS, AAAI, IJCAI, ICLR, NDSS, USENIX Security, CCS, S&P, CVPR, ICCV, ECCV).

Bernt Schiele has been Max Planck Director at MPI for Informatics and Professor at Saarland University since 2010. He studied computer science at the University of Karlsruhe, Germany. He worked on his master thesis in the field of robotics in Grenoble, France, where he also obtained the "diplome d'etudes approfondies d'informatique". In 1994 he worked in the field of multi-modal human-computer interfaces at Carnegie Mellon University, Pittsburgh, PA, USA in the group of Alex Waibel. In 1997 he obtained his PhD from INP Grenoble, France under the supervision of Prof. James L. Crowley in the field of computer vision. The title of his thesis was "Object Recognition using Multidimensional Receptive Field Histograms". Between 1997 and 2000 he was postdoctoral associate and Visiting Assistant Professor with the group of Prof. Alex Pentland at the Media Laboratory of the Massachusetts Institute of Technology, Cambridge, MA, USA. From 1999 until 2004 he was Assistant Professor at the Swiss Federal Institute of Technology in Zurich (ETH Zurich). Between 2004 and 2010 he was Full Professor at the computer science department of TU Darmstadt.

Optimising for Interpretability: Convolutional Dynamic Alignment Networks

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 Convolutional Dynamic Alignment Networks (CoDA-Nets), we provide:

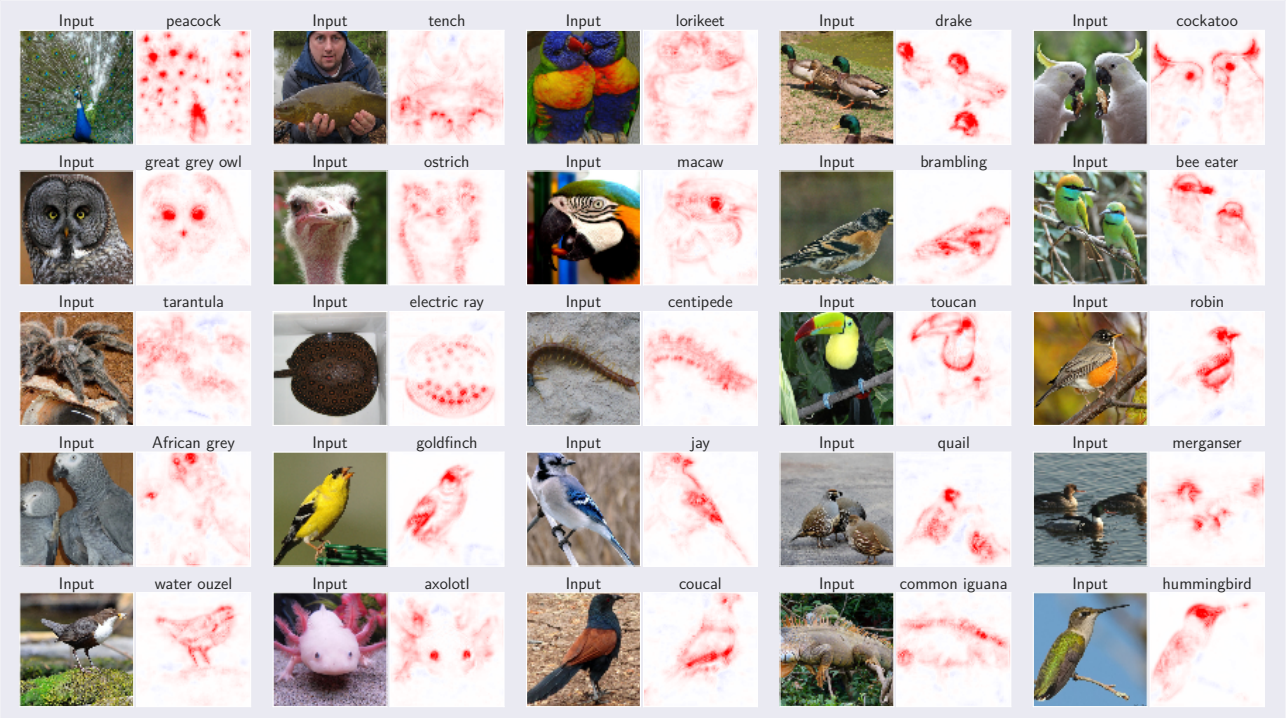


Figure A1: The first 25 most confident classifications decomposed into the contributions from each spatial location, filtered to 1 image per class. Positive (negative) contributions for the ground truth class are shown in red (blue).

A. Additional qualitative results

Additional Imagenet examples. In Figs. A1 and A2 we present additional qualitative examples of the model-inherent contribution maps. In particular, we show the decomposition of the model predictions into input contributions for 50 out of 100 classes; for each class, we show the most confidently classified image and show the classes in sorted order (by confidence). In Figs. A3 and A4, we moreover show the contributions maps for the first 10 of the overall most confidently classified images next to the attribution maps from the post-hoc importance attribution methods for qualitative comparison. We note that GradCam consistently highlights very similar regions to the CoDA-Net contribution maps, but does so at a lower resolution. All contribution maps based on the CoDA-Net use the same linear color scale, which has been set to $(-v, v)$ with v the 99.75th percentile over all absolute values in the contributions maps shown in Figs. A1 and A2. For reproducing the presented contribution maps and more, please visit `github.com/moboehle/CoDA-Nets`.

Regularising the linear mapping on CIFAR-10. As mentioned in the main paper, the explicit representation of the model computations as a linear mapping, i.e.,

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{W}_{0\to L}(\mathbf{x})\mathbf{x} \quad ,$$

allows to explicitly regularise the linear mappings and thereby the model-inherent contribution maps. In Fig. A5 we qualitatively show how the regularisation impacts the contribution maps; for these experiments, we added a regularisation term to the loss function and optimised

$$\mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) = \text{BCE}\left(\sigma(T^{-1}\mathbf{W}_{0\to L}(\mathbf{x}_i)\mathbf{x}_i + \mathbf{b}_0), \mathbf{y}_i\right) + \lambda\langle|\mathbf{W}_{0\to L}(\mathbf{x}_i)|\rangle \tag{A.1}$$

with $\langle|\mathbf{U}|\rangle$ denoting the average absolute value of a matrix $\mathbf{U}$. We see that without any regularisation the contribution maps are difficult to interpret. As soon as even a small regularisation is applied, however, the maps align well with the discriminative parts of the input image.

Sanity check. In [S1], the authors found that many commonly used methods for importance attribution are not *model-faithful*, i.e., they do not reflect the learnt parameters of the model. In order to test this, they proposed to examine how the attributions change if the model parameters are randomised step by step. If the attributions remain stable under model randomisation, they cannot be assumed to explain a specific model, but rather reflect properties of the general architecture

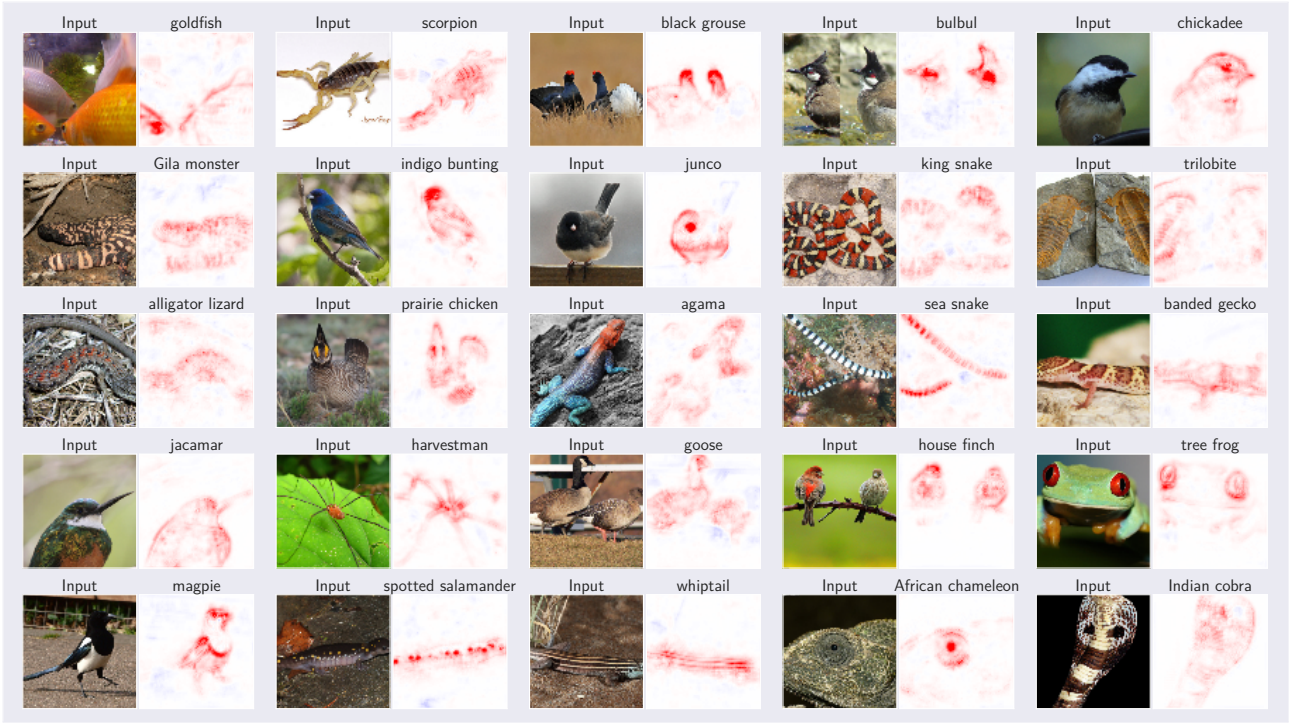


Figure A2: The 26th to the 50th most confident classifications decomposed into the contributions from each spatial location, filtered to 1 image per class. Positive (negative) contributions for the ground truth class are shown in red (blue).

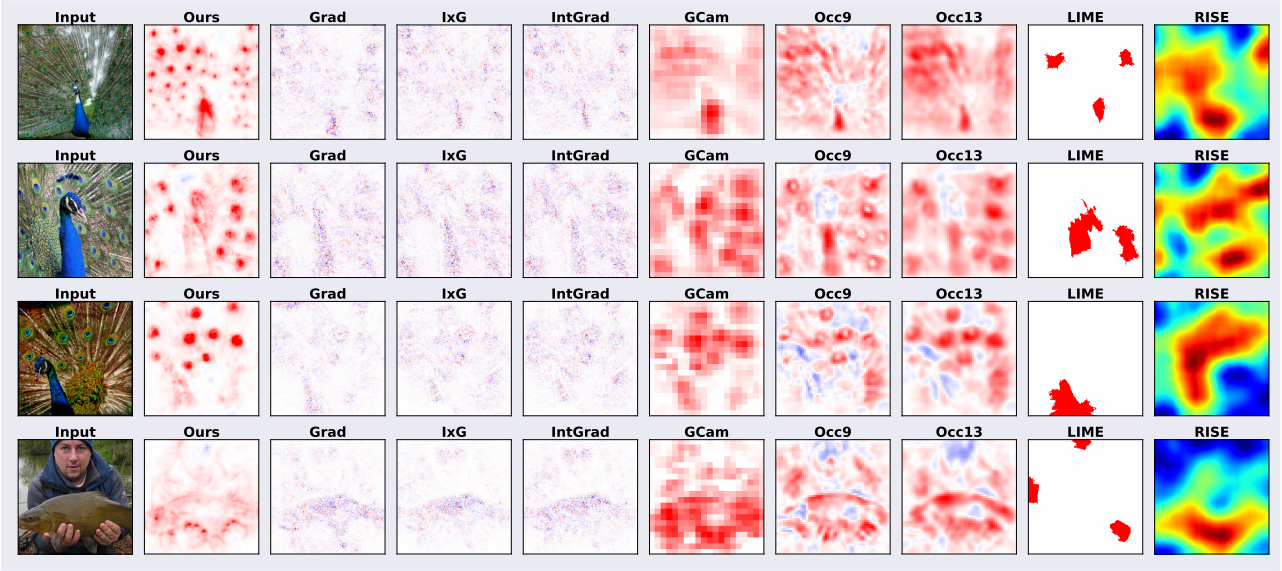


Figure A3: Comparison between attribution methods for the 4 most confident classifications. We show positive importance attributions in red, negative attributions in blue; for RISE we use its default visualisation. Note that the model seems to align the weights well with the ornamental eyespots of the peacocks (also see the peacock in Fig. A4) or the heads of the ducks in Fig. A4. While the latter are also highlighted by GCam, the former constitute a structure that is too fine-grained for GCam to resolve properly.

and the input data. In Fig. A6, we show how the model-inherent contribution maps behave when re-initialising the CoDA-Net

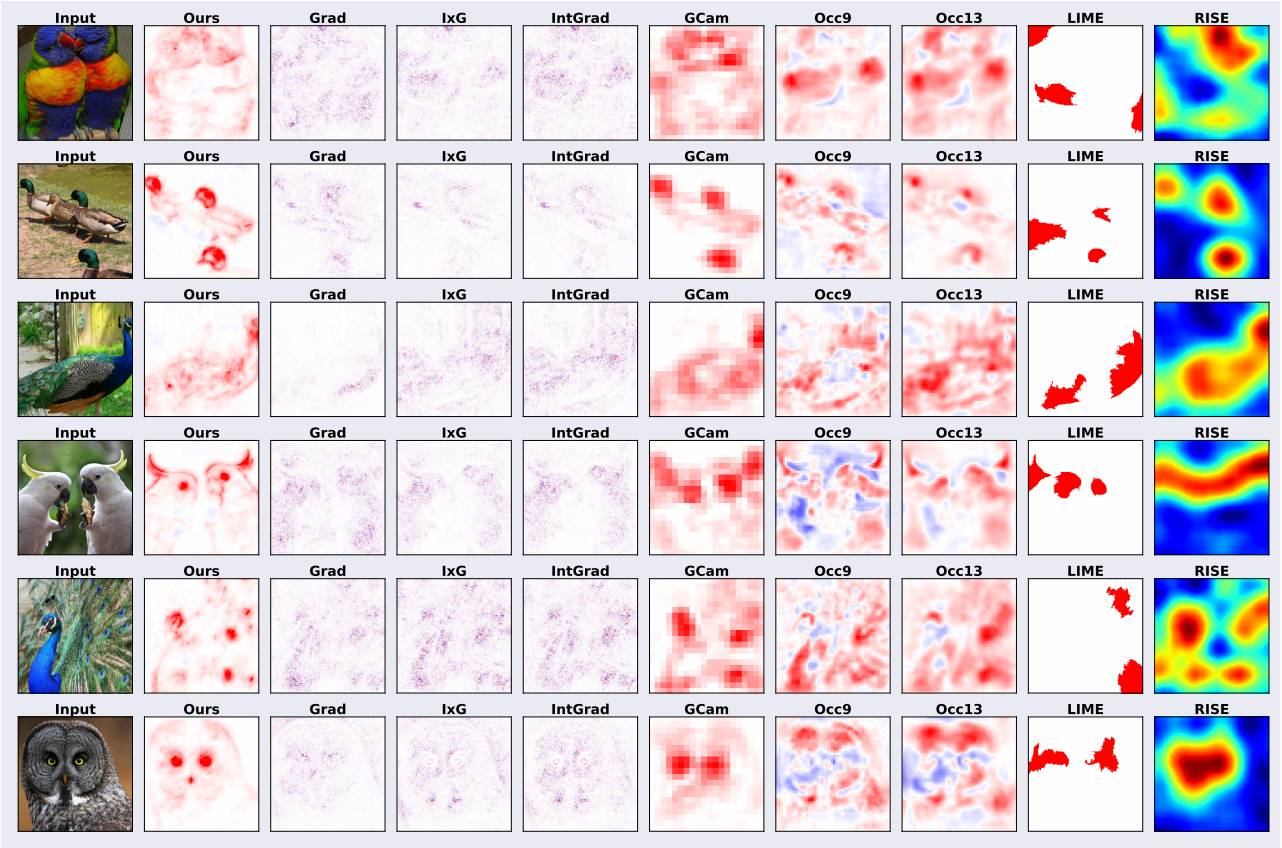


Figure A4: Comparison between attribution methods for the 5th to the 10th most confident classifications. We show positive importance attributions in red, negative attributions in blue; for RISE we use its default visualisation.

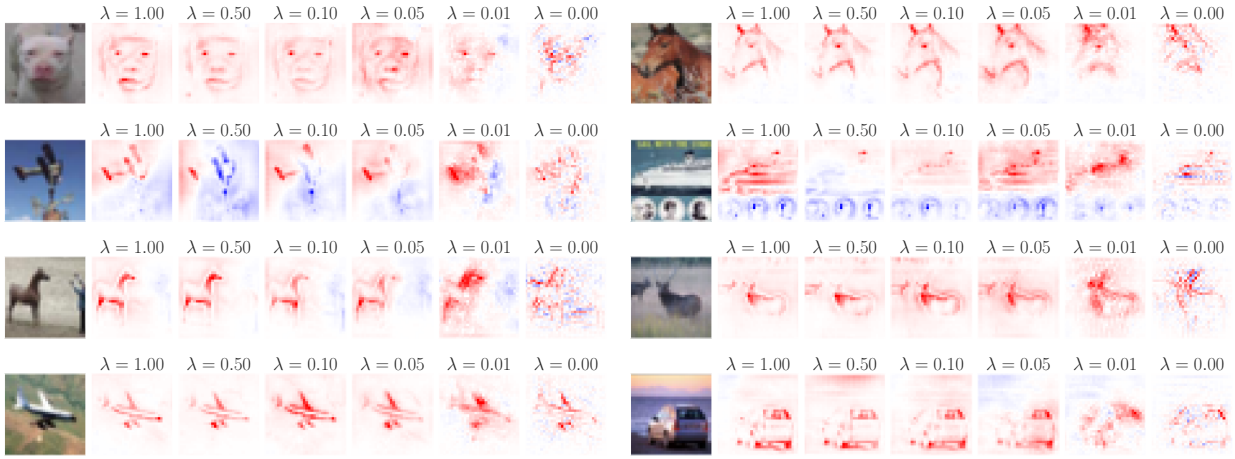


Figure A5: Visualising the qualitative effect of explicitly regularising the linear mapping $\mathbf{W}_{0\to L}$ on CIFAR-10. While the contribution maps without regularisation are noisy, they become sharper as soon as a regularisation is applied.

layers one at a time to a random parameter setting, starting from the deepest layer. As can be seen, the contribution maps get significantly perturbed with every layer that is reset to random parameters; thus, the contribution maps pass this 'sanity check' for attribution methods.

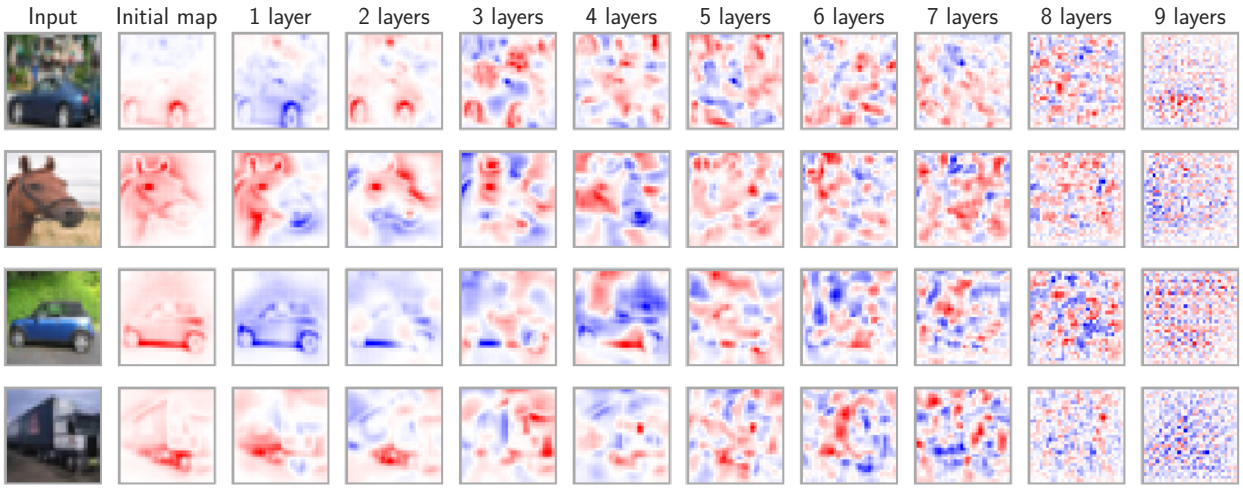


Figure A6: Sanity check experiment as in [S1]. If the importance attributions remain stable under parameter randomisation, they cannot be assumed to faithfully reflect the learnt parameters of the model. Since the contribution maps get significantly perturbed when re-initialising layers from network output (left) to network input (right), the model-inherent contribution maps thus pass this sanity check. Positive contributions are shown in red, negative contributions in blue.

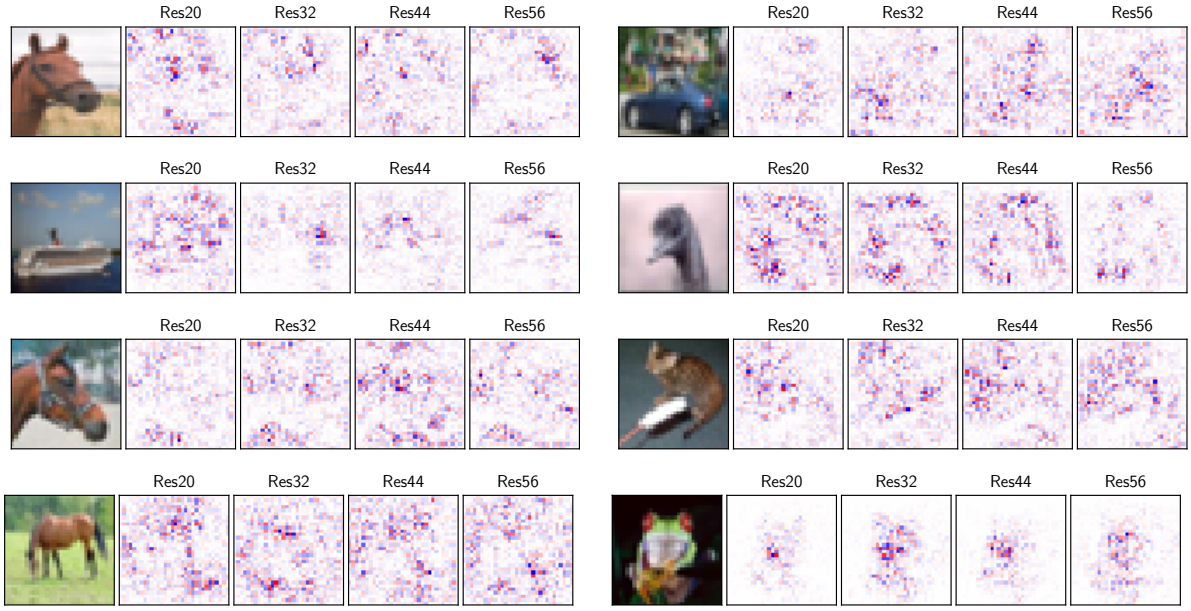


Figure A7: 'Input×Gradient' evaluated on different ResNets. Since ResNets are piece-wise linear functions, s.t. $\mathbf{y}(\mathbf{x}) = \mathbf{M}(\mathbf{x})\mathbf{x} + \mathbf{b}(\mathbf{x})$, this is the ResNet-based equivalent to the CoDA-Net contribution maps.

Contribution maps of a piece-wise linear model. In Fig. A7 we show contribution maps obtained from different pre-trained ResNet architectures obtained from https://github.com/akamaster/pytorch_resnet_cifar10. In particular, we visualise the 'Input×Gradient' method. Note that this yields contribution maps, since piece-wise linear models, such as the ResNets, produce input-dependent linear mappings, similar to the CoDA-Nets. These contribution maps, however, are rather noisy and do not reveal particularly relevant features.

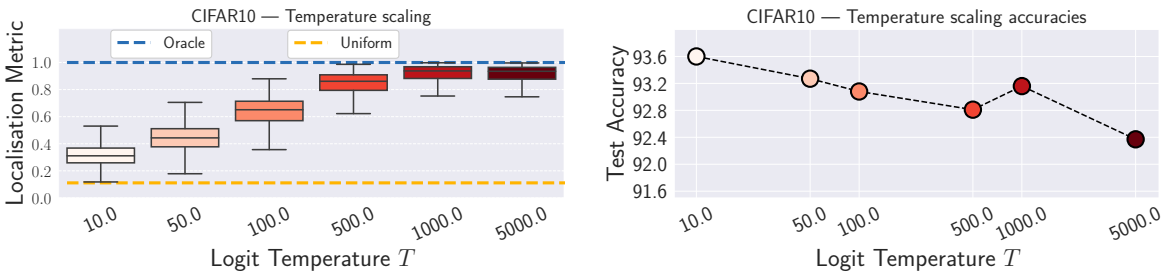


Figure B1: **Left:** Localisation metric results for models trained with different temperatures T, same as in main paper (see Fig. 6, top right). **Right:** Corresponding accuracies of the models on the CIFAR10 test set. There seems to be a trade-off between the interpretability and the accuracy of the models due to the regularising effect of T.

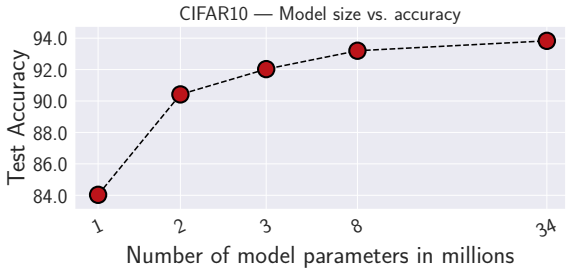


Figure B2: Effect of scaling the rank of the DAUs in the CoDA-Nets on accuracy. Specifically, all ranks in the S-CoDA-SQ model presented in the main paper were scaled with the same factor, thereby changing the model size.

B. Additional quantitative results

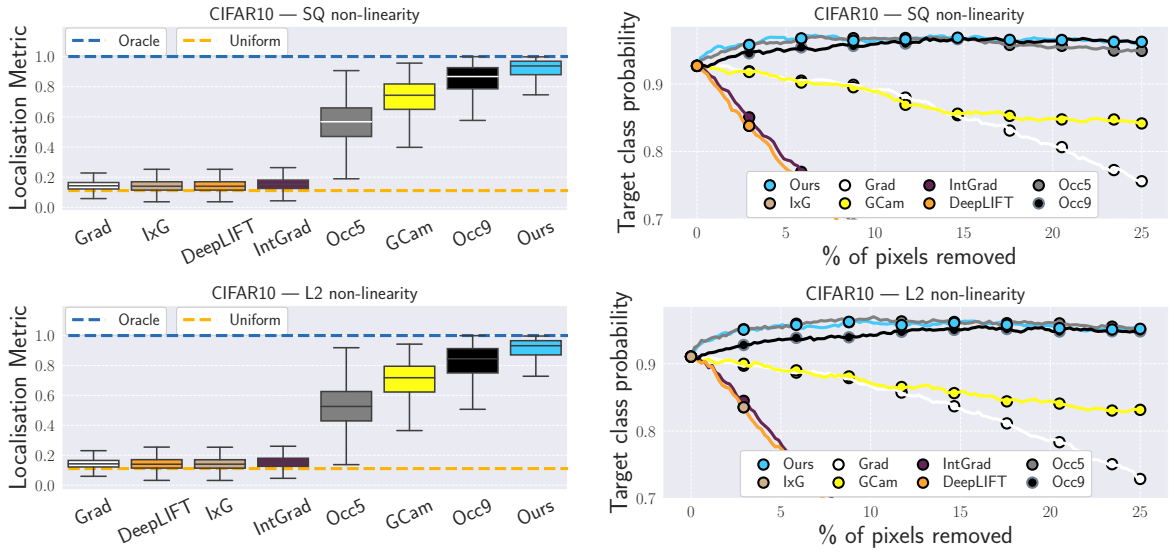
Performance / interpretability trade-off. While the CoDA-Nets were observed to train and perform well over a wide range of choices for the logit temperature T, there seems to be a trade-off between the accuracies of the network and their interpretability—the implicit alignment regularisation comes at a cost. For example, in Fig. B1, we contrast the gain in interpretability (left, same figure as in main paper) with the corresponding accuracies (right).

Model size vs. accuracy. Given the quadratic form in the DAUs (cf. eq. (1), $\approx \mathbf{x}^T\mathbf{M}\mathbf{x}$), the number of parameters per DAU scales quadratically with the input dimensions. In order to limit the model size, we decided to explicitly limit the rank of the DAUs by factorising the matrix $\mathbf{M}$ into $\mathbf{AB}$ with $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times d}$. While this allows to be more parameter efficient, it, of course, affects the modelling capacity of the DAUs. In Fig. B2, we present how the accuracy changes with the model size; for this, we scaled the ranks of all DAUs per layer with factors of 1/8, 1/4, 1/2, and 4.5 compared to the S-CoDA-SQ model presented in the main paper. This results in models with 1.1M, 2.0M, 4.0M, and 34.6M parameters respectively. For comparison, the original model is also included in Fig. B2 (8M parameters).

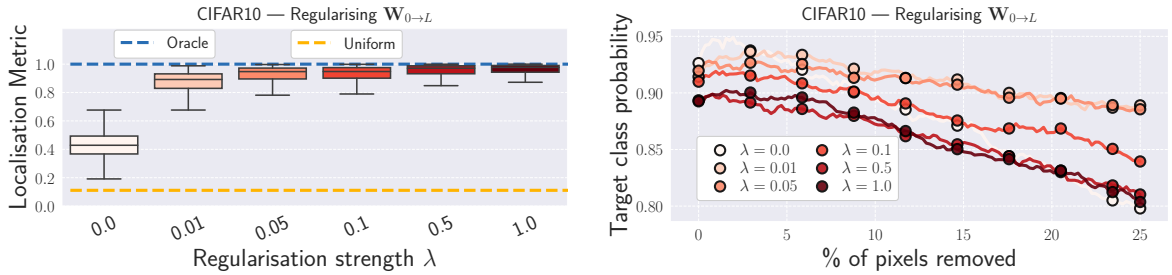
Interpretability results for L2 and SQ non-linearity. In Fig. B3a we show the results of evaluating the different methods for importance attribution on a model with the L2 and SQ non-linearities, see eq. (2) in the main paper. As can be seen, the results are very similar to those presented in the main paper in Fig. 6 (center column) for a model trained with the WB rescaling method; in particular, the model-inherent contribution maps outperform the other methods under the localisation metric and are on par with the occlusion methods under the pixel removal metric; note, however, that the occlusion methods are a direct estimate of the behaviour under pixel removal and therefore expected to perform well under this metric.

Regularisation of $\mathbf{W}_{0\to L}$. In Fig. B3b, we show the results of the localisation metric and the pixel removal metric for models trained on CIFAR-10 with different regularisation strengths λ, see eq. (A.1). We note that the localisation metric benefits from an increase in the regularisation strength λ. For the pixel removal metric, we observe that while the models improve at first under this metric, a strong regularisation makes the predictions more brittle.

Removing the *most important* pixels first. In Fig. B4a, we show the results of the pixel removal metric when removing the most important pixels first. As can be seen, the model-inherent contributions better predict the importance of pixels and



(a) The quantitative results for CoDA-Nets trained with the L2 and SQ non-linearities are very similar to those of a model trained with the WB non-linearity as shown in the main paper (Fig. 6, center column). In particular, we observe that the CoDA-Net outperforms the other methods under the localisation metric and achieves similar performance to the occlusion attribution method, which directly estimates the change in model prediction when removing a pixel.

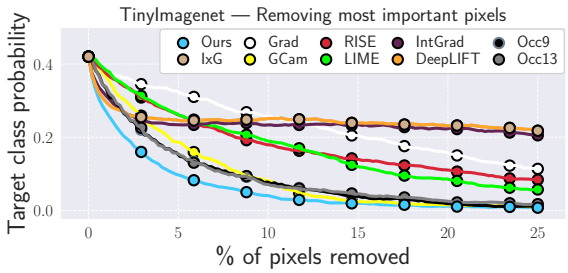


(b) Similar to the effect of temperature scaling, cf. Fig. 6 (right column) in the main paper, here we show quantitative results for different regularisation strengths λ, see eq. (A.1). Similar to increasing the temperature, stronger regularisations also improve the localisation metric (left). While the pixel perturbation metric (right) also improves at first ($\lambda = 0.01$ and $\lambda = 0.05$), the models become less stable for stronger regularisation strengths. In comparison, we found that increasing the temperature also improves the performance of the models under this metric, see Fig. 6 (right column) in the main paper.

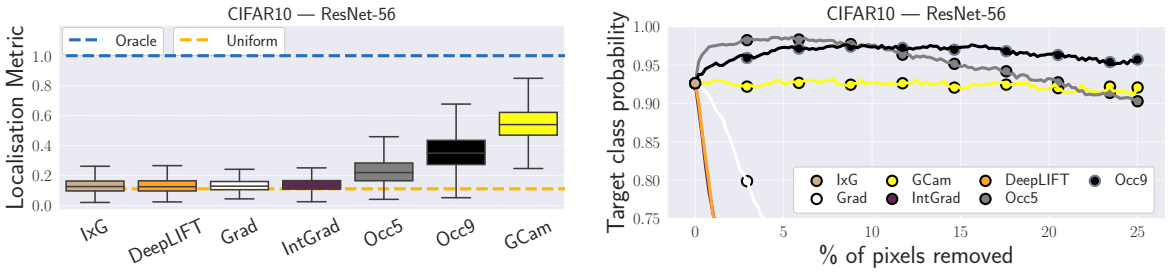
Figure B3: Quantitative results for two ablations: (a) using the L2 or SQ non-linearities and (b) increasing the regularisation of the linear mapping $\mathbf{W}_{0\to L}$, see eq. (2) in the main paper and eq. (A.1) respectively. Compare with Fig. 6 in the main paper.

outperform the other attribution methods under this metric; in particular, the confidence drops most rapidly when removing pixels according to the ranking given by the model-inherent contribution maps.

Evaluating a pre-trained ResNet. In order to establish a baseline for the performance of the different attribution methods on a classical neural network architecture, we additionally evaluated the attribution methods on a pre-trained ResNet and show the results in Fig. B4b. Specifically, we rely on a publicly available pre-trained ResNet-56 obtained from https://github.com/akamaster/pytorch_resnet_cifar10, which achieves a classification accuracy of 93.39%. The results show that the performance of the CoDA-Net-derived contribution maps is not only strong when comparing them to attribution methods evaluated *on the same model*. Instead, they also perform well in comparison to those methods evaluated *on a different model*. In particular, the model-inherent contribution maps of the CoDA-Net outperform attribution methods evaluated on the pre-trained ResNet-56 under the localisation metric. Further, only the occlusion attributions produce similarly strong pixel importance rankings for the ResNet; note, however, that the occlusion methods are a



(a) Results for the pixel removal metric when first removing the most important pixels, evaluated on the S-CoDA-SQ model. As can be seen, the ranking given by the model-inherent contribution maps seems to best reflect the pixel importance, since the confidence most rapidly drops when removing pixels according to this ranking.



(b) Quantitative results for methods for importance attribtution evaluated on a pre-trained ResNet-56 on CIFAR-10. Comparing these results to Fig. 6 (center column) in the main paper or Fig. B3a, it can be seen that the model-inherent contribution maps of the CoDA-Net are also strong when compared to importance attributions evaluated on a different model.

Figure B4: In (a) we show the results of the pixel-removal metric when removing those pixels first that are considered the most important ones according to the importance attribution method. Moreover, in (b) we plot the quantitative results for the evaluation metrics of the importance attributions for a pre-trained ResNet-56.

direct estimate of the behaviour under pixel removal and therefore expected to perform well under this metric.

C. Implementation details

C.1. Training and architecture details

C.1.1 Pure CoDA Networks

Architectures. The architectures used for the results in Table 1 in the main paper are given in Table C1. All activation maps are padded with $(k-1)/2$ zeros on each side, such that the spatial dimensions are only reduced by the strides; here, k refers to the kernel size. As can be seen, the activation maps thus still have a spatial resolution after the last layer, which we further reduce with a global sum-pooling layer. Note that global sum-pooling is just a linear layer with no trainable parameters and therefore still allows for linear decomposition. For the models which use a static embedding (the image and its negative, see sec. 4.3), the input itself consists of 6 channels; hence, the first layer takes an input with 6 channels per pixel. For the models with a learnt embedding function, the input has 32 channels. Note that for the eCoDA model the matrices $\mathbf{B}$ are not shared and each DAU has its own matrix $\mathbf{B}$.

Training details. We use the pytorch library [S9] and optimise all networks with the Adam optimiser [S8] with default values. As for the loss function, we use the binary cross entropy loss to optimise class probabilities individually (as 'one-vs-all' classifiers). For all networks, we used a base learning rate of 2.5×10^{-4}; for the Imagenet experiment and the S-eCoDA experiments on CIFAR-10, we employed learning rate warm-up and linearly increased the learning rate from 2.5×10^{-4} to 1×10^{-3} over the first 15 (10) epochs. Further, we trained for 200 epochs on CIFAR-10, for 100 epochs on TinyImagenet, and for 60 epochs on the Imagenet subset; we decreased the learning rate by a factor of 2 after every 60/30/20 epochs on CIFAR-10/TinyImagenet/Imagenet; for the eCoDA experiments on CIFAR-10 the learning rate was decayed according to a cosine schedule to a final learning rate of 10^{-5} after the initial linear increase. We used a batch size of 16, 128, and 64 for CIFAR-

10, TinyImagenet, and Imagenet respectively (and 64 for e-CoDA experiments on CIFAR-10). For the Imagenet subset, we additionally used RandAugment [S3] with parameters $n = 2$ and $m = 9$; for this, we relied on the publicly available implementation at https://github.com/ildoonet/pytorch-randaugment and followed their augmentation scheme. The qualitatively evaluated model (see Figs. 5, 8, and A1-A4) for the Imagenet subset was trained with $T = 1e5$ and achieved a top-1 accuracy of 76.5%. For comparison, we trained several ResNet-50 models (taken from the pytorch library [S9]) with the exact same training procedure, i.e., batch size, learning rate, optimiser, augmentation, etc.). The best ResNet-50 out of 4 runs achieved 79.16% top-1 accuracy[1], which outperforms the CoDA-Net but is nevertheless comparable. While it is surely possible to achieve better accuracies for both models, long training times for the CoDA-Nets have thus far prevented us from properly optimising the architectures both on the 100 classes subset, as well as on the full Imagenet dataset. In order to scale the CoDA-Net models to larger datasets, we believe it is important to first improve the model efficiency in future work. Lastly, when regularising the matrix entries of $\mathbf{W}_{0\to L}$, see eq. (A.1), we regularised the absolute values for the true class c, $[\mathbf{M}_{0\to L}]_c$, and a randomly sampled incorrect class per image.

C.1.2 Hybrid CoDA Nets

CIFAR10. As described in the main paper, the interpolation experiments for CIFAR-10 are based on a ResNet-56 obtained from [S6]. This model consists of a convolutional layer + batch normalisation [S7] (C+B), followed by three times nine *residual blocks* (RBs) as well as a fully connected and a pooling (FC+P) layer; for more details we kindly refer the reader to the original work [S5] and the implementation [S6] on which we base these experiments. We can summarise this model by [C+B, 9RB, 9RB, 9RB, FC+P] and denote individual *segments* $\mathcal{S}_i$ of the model by their index in this summary counting from the back, e.g., $\mathcal{S}_5$=[C+B] and $\mathcal{S}_1$=[FC+P]. Similarly, we divide the S-eCoDA (see Table C1) into segments, which we define as [C+B, 3eCoDA, 3eCoDA, 2eCoDA, eCoDA+P]. We 'interpolate' between these two networks by successively replacing segments from the ResNet model by its corresponding segment in the CoDA Net—each replacement yields one hybrid model, which we train on the fixed representations[2] obtained from the ResNet-based stem. We use the same training scheme as in the S-eCoDA-based models on CIFAR-10 without learning rate warm-up. Further, the models with $[1, 2, 3, 4]$ replaced segments are trained with temperatures of $10^{[1,3,5,5]}$.

Imagenet. The interpolation experiments on the Imagenet dataset are based on a pre-trained ResNet-50 obtained from the pytorch [S9] library. Here, we define individual segments by the final classification layer + pooling and individual ResNet bottlneck blocks. For the corresponding eCoDA Net, the segments contain exactly one eCoDA layer. We train with a batch size of 128, DAU ranks of 8, 100 epochs, a cosine learning rate decay from 10^{-3} to 10^{-5}, a temperature $T = 100$ and apply RandAugment [S3] with default parameters on random crops of size 192×192.

C.2. Convolutional Dynamic Alignment Units

In Algorithm 1, we present the implementation of the convolutional DAUs (CoDAUs). As can be seen, a Convolutional Dynamic Alignment Layer applies dynamic (input-dependent) filters to each of the patches extracted at different spatial locations. In detail, the Dynamic Alignment Units are implemented as two consecutive convolutions (lines 11 and 15), which are equivalent to first applying matrix $\mathbf{B}$ (line 24) and then $\mathbf{A}$ to each patch and adding a bias term $\mathbf{b}$ (line 29). After applying the non-linearity (line 33), we obtain the dynamic weight vectors for CoDAUs as described in eq. (1) in the main paper. In particular, for every patch $\mathbf{p}_{hw}$ extracted at the spatial positions hw in the input, we obtain the dynamic weight $\mathbf{w}_j(\mathbf{p}_{hw})$ for the j-th DAU as

$$\mathbf{w}_j(\mathbf{p}_{hw}) = g(\mathbf{A}_j\mathbf{B}\mathbf{p}_{hw} + \mathbf{b}_j) \; ; \tag{C.1}$$

note that the projection matrices $\mathbf{B}$ are thus shared between the DAUs. These weights are then applied to the respective locations (line 41) to yield the outputs of the DAUs per spatial location. As becomes apparent in line 41, the outputs are linear transformations (weighted sums) of the input and can be written as

$$\mathbf{a}_{l+1}(\mathbf{a}_l) = \mathbf{W}(\mathbf{a}_l)\mathbf{a}_l \quad , \tag{C.2}$$

with $\mathbf{a}_l \in \mathbb{R}^d$ the vectorised input to layer l and $\mathbf{W} \in \mathbb{R}^{f\times d}$ and f the number of filters (DAUs). The rows in matrix $\mathbf{W}$ correspond to exactly one filter (DAU) applied to exactly one patch $\mathbf{p}_{hw}$ and are non-zero only at those positions that correspond to this specific patch in the input.

[1] The best test accuracies per run are given by 79.16%, 79.04%, 78.86%, and 78.7% respectively.

[2] I.e., the ResNet weights are fixed in these experiments.

C.3. Attribution methods

In section 5.2, we compare the model-inherent contribution maps of the CoDA-Nets to those of the following methods for importance attribution: the gradient of the class logits with respect to the input image [S2] (Grad), 'Input×Gradient' (IxG, cf. [S1]), GradCam [S13] (GCam), Integrated Gradients [S16] (IntG), DeepLIFT [S14], several occlusion sensitivities (Occ-K, with K the size of the occlusion patch) [S17], RISE [S10], and LIME [S11].

For RISE and LIME, we relied on the official implementations available at `https://github.com/eclique/RISE` and `https://github.com/marcotcr/lime` respectively. For RISE, we generated 6000 masks with parameters $s = 6$ and $p_1 = 0.1$. For LIME, we evaluated on 256 samples per image and used the top 3 features for the attribution; for the segmentation, we also used the default parameters, namely 'quickshift' with $max\_dist = 200$, $ratio = 0.2$, and a kernel size of 4.

For Grad, GCam, IxG, IntG, DeepLIFT, and the occlusion sensitivities, we relied on the publicly available pytorch library 'captum' (`https://github.com/pytorch/captum`). GCam was used on the last activation map before global sum-pooling. The occlusion sensitivities were used with strides of 2 on CIFAR-10 and strides of 4 for TinyImagenet. Finally, for IntG we used 20 steps for the integral approximation.

C.4. Evaluation metrics

In section 5.2, we evaluated the attribution methods against 2 quantitative metrics: **(1)** the adapted *pointing game* [S18] and **(2)** the prediction stability under removing the *least important pixels* as in [S15]. In section B, we further show results for removing the *most important pixels* first.

For **(1)**, we constructed 500 (250) 3×3 multi-images for CIFAR-10 (TinyImagenet); for an example with 2×2, see Fig. 7 in the main paper. In each of these multi-images, every class occurred at most once. As stated in section 5.2, we measured the fraction of positive contributions falling inside the correct mini-image. Further, the images were sorted according to their confidence for each of the classes. For every multi-image, a random set of classes was sampled. For each of the classes, we included the most confidently classified class image in the multi-image that had not been used yet in previous multi-images.

For **(2)**, we followed [S15] and successively replaced one (embedded) pixel at a time, replacing its embedding by a corresponding zero vector, until up to 25% of the image were removed. The pixels were removed in order, sorted by their assigned importance.

Algorithm 1: Implementation of a Convolutional Dynamic Alignment Layer

```
from torch import nn
import torch.nn.functional as F

class DAUConv2d(nn.Module):

    def __init__(self, in_channels, out_channels, rank, kernel_size, stride, padding, act_func):
        # act_func: non-linearity for scaling the weights. E.g., L2 or SQ.
        # out_channels: Number of convolutional DAUs for this layer.
        # rank: Rank of the matrix AB.
        # 'dim_reduction' applies matrix B.
        self.dim_reduction = nn.Conv2d(in_channels, rank, kernel_size, stride, padding, bias=False)
        # Total dimensionality of a single patch
        self.patch_dim = in_channels * kernel_size * kernel_size
        # 'weightings' applies matrix A and adds bias b.
        self.weightings = nn.Conv2d(rank, out_channels * self.patch_dim , kernel_size=1, bias=True)
        self.act_func = act_func
        self.out_channels = out_channels
        self.kernel_size = kernel_size
        self.stride = stride
        self.padding = padding

    def forward(self, in_tensor):
        # Project to lower dimensional representation, i.e., apply matrix B. This yields Bp for every patch p.
        reduced = self.dim_reduction(in_tensor)
        # Get new spatial size height h and width w
        h, w = reduced.shape[-2:]
        batch_size = in_tensor.shape[0]
        # Apply matrix A and add bias b, yielding ABp + b for every patch p.
        weights = self.weightings(reduced)
        # Reshape for every location to size patch_dim×out_channels
        weights = weights.view(batch_size, self.patch_dim, out_channels, h, w)
        # Apply non-linearity to the weight vectors, yielding w(p) = g(ABp + b) as in eq. (1) for every patch p.
        weights = self.act_func(weights, dim=1)
        # Extract patches from the input to apply dynamic weights to patches.
        patches = F.unfold(in_tensor, self.kernel_size, padding=self.padding, stride=self.stride)
        # Reshape for applying weights.
        patches = patches.view(batch_size, self.patch_dim, 1, h, w)
        # Apply the weights to the patches.
        # As can be seen, the output is just a weighted combination of the input, i.e., a linear transformation.
        # The output can thus be written as o = W(x)x.
        return (patches * weights).sum(1)
```

Network	Input dimensions	Layer	Number of DAUs	Rank of $\mathbf{AB}$	Kernel size	Stride
S-CoDA	$6 \times 32 \times 32$	1	16	32	3	1
		2	16	32	3	1
		3	32	64	3	2
		4	32	64	3	1
		5	32	64	3	1
		6	64	64	3	2
		7	64	64	3	1
		8	64	64	3	1
		9	10	64	1	1
S-eCoDA	$6 \times 32 \times 32$	1	16	16	3	1
		2	16	16	3	1
		3	32	16	3	2
		4	32	16	3	1
		5	32	32	3	1
		6	64	32	3	2
		7	64	32	3	1
		8	64	32	3	1
		9	10	32	1	1
L-CoDA	$6 \times 240 \times 240$	1	16	64	7	3
		2	32	64	3	1
		3	32	64	3	1
		4	64	128	3	2
		5	64	128	3	1
		6	64	128	3	1
		7	64	256	3	2
		8	64	256	3	1
		9	100	256	3	1
XL-CoDA	$6 \times 64 \times 64$	1	16	64	5	1
		2	32	64	3	1
		3	32	128	3	2
		4	64	128	3	1
		5	64	128	3	1
		6	64	256	3	2
		7	64	256	3	1
		8	64	256	3	1
		9	200	256	3	2

Table C1: Architecture details for the results presented in Table 1 in the main paper.

D. Low-rank matrix approximations

In the following, we first introduce the standard formulation of the low-rank matrix approximation problem and present the well-known solution via singular value decomposition. We then introduce an additional constraint to this standard formulation and show that at their optimum (maximal average output), the DAUs solve this constrained approximation problem.

Low-rank approximation problem. Given a data matrix[3] $\mathbf{M} \in \mathbb{R}^{m \times n}$, the low-rank approximation problem is typically formulated as $\min_{\mathbf{M}'}(||\mathbf{M} - \mathbf{M}'||_F)$ with *rank*$(\mathbf{M}') \leq r$ and $||\cdot||_F$ the Frobenius norm. The Frobenius norm can be written as a sum over the column differences $||\mathbf{\Delta}_i||_F^2 = ||\mathbf{m}_i||_2^2 + ||\mathbf{m}'_i||_2^2 - 2\mathbf{m}_i^T\mathbf{m}'_i$ with $\mathbf{m}_i, \mathbf{m}'_i \in \mathbb{R}^m$ the i-th columns of the matrices $\mathbf{M}$ and $\mathbf{M}'$ respectively. Note that since the $\mathbf{m}_i$ are fixed (they just correspond to the fixed input matrix $\mathbf{M}$), the optimisation objective is equivalent to

$$\min \textstyle\sum_i ||||\mathbf{m}'_i||_2^2 - 2\mathbf{m}_i^T\mathbf{m}'_i = \tag{D.1}$$

$$\min \textstyle\sum_i ||||\mathbf{m}'_i||_2^2 - 2||\mathbf{m}_i||||\mathbf{m}'_i|| \cos(\alpha_i) \quad , \tag{D.2}$$

[3] In the context of our work, we can think of $\mathbf{M}$ as storing n data samples of dimensionality m; e.g., when considering images, each column might correspond to one vectorised image from a dataset of n images. The low-rank approximation problem aims to approximate this dataset with r independent variables (dimensionality reduction).

with α_i the angle between $\mathbf{m}_i$ and $\mathbf{m}'_i$. Hence, optimising the Frobenius norm of the difference matrix finds a trade-off between aligning the directions (angles) of the columns of $\mathbf{M}$ and $\mathbf{M}'$ and approximating the correct norms of the original columns in $\mathbf{M}$. The optimal solution for this problem is given by singular value decomposition (SVD) [S4] of $\mathbf{M}$, and the optimal $\mathbf{M}'$ can be written as

$$\mathbf{M}' = \mathbf{U}_r\mathbf{U}_r^T\mathbf{M} = \mathbf{U}\mathbf{\Sigma}_r\mathbf{V}^T \quad , \tag{D.3}$$

where $\mathbf{U}_r$ is the matrix of left singular vectors of $\mathbf{M}$ up to the r-th vector[4]; the remaining vectors are replaced by zero-vectors. Note that the first formulation of the solution ($\mathbf{U}_r\mathbf{U}_r^T\mathbf{M}$) emphasises a property that we will encounter again for the DAUs. In particular, this formulation highlights the fact that the optimal matrix $\mathbf{M}'$ is given by reconstructing the individual data points from the r-dimensional eigenspace spanned by the eigenvectors of the data points; i.e., each column $\mathbf{m}'_i$ can be calculated as $\mathbf{U}_r\mathbf{U}_r^T\mathbf{m}_i$. The right hand side of the equation shows the conventional SVD-notation of the low-rank approximation solution, with $\mathbf{U} \in \mathbb{R}^{m\times m}, \mathbf{\Sigma}_r \in \mathbb{R}^{m\times n}$ $\mathbf{V} \in \mathbb{R}^{n\times n}$ and $\mathbf{\Sigma}_r$ being a rectangular diagonal matrix with the first r singular values as non-zero entries.

Constrained low-rank approximations Now we diverge from the standard low-rank approximation formulation and impose an additional constraint on $\mathbf{M}'$ (red box):

$$\min_{\mathbf{M}'}(||\mathbf{M} - \mathbf{M}'||_F) \ , \tag{D.4}$$

$$\text{s.t.} \quad \textit{rank}(\mathbf{M}') \leq r \ \boxed{\wedge \ ||\mathbf{m}'_i||_2^2 = 1 \ \forall i} \ . \tag{D.5}$$

When combining eqs. (D.2) and (D.5), it becomes clear that the optimisation problem can now be formulated as

$$\max \textstyle\sum_i \mathbf{m}_i^T\mathbf{m}'_i = \max \sum_i ||\mathbf{m}_i||_2 \cos\left(\angle(\mathbf{m}_i, \mathbf{m}'_i)\right) \ , \tag{D.6}$$

since the norm of $\mathbf{m}'_i$ is fixed and the addition of or multiplication with a fixed constant does not change the location of the optima.

As a corollary, we note that *any* matrix $\mathbf{M}'$ with a maximum rank r that maximises eq. (D.6) gives a solution to the constrained approximation problem, *independent* of the norm of its columns: the columns can be normalised *post-hoc* to fulfill the new constraint, since rescaling does not change neither the rank of the matrix, nor the score of the objective function in eq. (D.6).

Further, we note that the unnormalised optimal matrix $\widetilde{\mathbf{M}}'$ can be factorised as

$$\widetilde{\mathbf{M}}' = \mathbf{A}\mathbf{B}\mathbf{M} \ , \tag{D.7}$$

$$\text{s.t.} \quad \mathbf{M}' = G(\widetilde{\mathbf{M}}') = G(\mathbf{A}\mathbf{B}\mathbf{M}) \tag{D.8}$$

with $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times n}$, similar to the solution to the unconstrained matrix approximation problem in eq. (D.3). Here, G is a function that normalises the columns of a matrix to unit norm (as discussed in the previous paragraph, in order to fulfill the norm constraint, we can normalise the columns *post-hoc*).

Finally, this allows us to rewrite the objective function in eq. (D.6) as

$$\max \sum_i g(\mathbf{A}\mathbf{B}\mathbf{m}_i)\mathbf{m}_i = \max \sum_i \mathbf{w}(\mathbf{m}_i)\mathbf{m}_i \tag{D.9}$$

with $g(\mathbf{v})$ normalising its input vector to unit norm. Comparing this with eq. (1) in the main paper, we see that this is equivalent to maximising a DAU without bias term for maximal output over the n columns in $\mathbf{M}$ and therefore, at their optimum, the DAUs solve the constrained low-rank matrix approximation problem in eq. (D.5). In particular, the DAUs achieve this by encoding common inputs in the eigenvectors of $\mathbf{A}\mathbf{B}$, which allows for an optimal angular reconstruction of the inputs, similar to reconstructing from the first PCA components in a PCA decomposition. The main difference to PCA is that PCA yields optimal reconstructions under the L2-norm, whereas the DAUs yield optimal angular reconstructions.

[4] A short proof of the equality to the right can be found at the end of this section.

Proof of $\mathbf{U}_r\mathbf{U}_r^T\mathbf{M} = \mathbf{U}\mathbf{\Sigma}_r\mathbf{V}^T$.
In order to see that this equality holds, we first write matrix $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad \text{(SVD-form)} \tag{D.10}$$

Now, we multiply both sides with $\mathbf{U}_r\mathbf{U}_r^T$ from the left:

$$\mathbf{U}_r\mathbf{U}_r^T\mathbf{M} = \mathbf{U}_r\mathbf{U}_r^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{D.11}$$

$$\xrightarrow{\mathbf{U}_r^T\mathbf{U}=\mathbf{I}_r} \mathbf{U}_r\mathbf{U}_r^T\mathbf{M} = \mathbf{U}_r\mathbf{I}_r\mathbf{\Sigma}\mathbf{V}^T \tag{D.12}$$

$$\Rightarrow \mathbf{U}_r\mathbf{U}_r^T\mathbf{M} = \mathbf{U}\mathbf{\Sigma}_r\mathbf{V}^T \qquad \square. \tag{D.13}$$

Here, we made use of the fact that $\mathbf{U}$ is an orthogonal matrix and therefore $\mathbf{U}^T\mathbf{U} = \mathbf{I}$; since we only use the first r eigenvectors, we obtain the truncated identity matrix $\mathbf{I}_r$ when multiplying $\mathbf{U}_r^T\mathbf{U}$. Further, in the last line, we used that $\mathbf{U}\mathbf{\Sigma}_r = \mathbf{U}_r\mathbf{\Sigma}_r$.

E. Comparison to capsule networks

In order to discuss the relationship to capsule networks, in section E.1 we will first rewrite the classical capsule formulation in 'Dynamic Routing Between Capsules' by Sabour et al. [S12] to mitigate the notational differences between their work and our work. In section E.2 we then show that while capsules and Dynamic Alignment Units share some computations, there are several important differences that we summarise in Table E1.

	Classical capsules	**Dynamic Alignment Units**
Non-Lin. g	SQ	SQ, L2, ...
Activations	$g\left(\mathbf{V}\mathbf{x}\right)$	$g\left(\mathbf{A}\mathbf{B}\mathbf{x}+\mathbf{b}\right)$
Routing	**yes**	**no**
Low-rank	**no**	**yes**
Output	$\mathrm{CAP}(\mathbf{x})$	$\mathrm{CAP}(\mathbf{x})^T\mathbf{x}$

Table E1: Comparison between capsules and Dynamic Alignment Units (DAUs). Importantly, DAUs produce a linear transformation of the input by multiplying the 'capsule activations' with the input (see 'Output') and allow for constraining the rank of the transformation. The dynamic weights in the DAUs can be seen as the activations of a capsule, s.t. $\mathbf{w}(\mathbf{x}) = \mathrm{CAP}(\mathbf{x})$, see 'Output' in the table.

E.1. Reformulating capsules

In this subsection, we will show that the classical capsule formulation (eq. (E.1)), in which input capsules 'vote' for the activations of an output capsule, can be written as a simple linear transformation $\mathbf{s} = \mathbf{V}\mathbf{x}$ if just one iteration of the dynamic routing algorithm is applied; here, $\mathbf{s}$ is a vector containing the activations of the output capsule, $\mathbf{V}$ stores the 'votes' of the input capsules to an output capsule, and $\mathbf{x}$ is a vector containing the activations of all input capsules. In the following, we will start from how capsules are formulated in [S12] and rewrite this formulation step by step.

In [S12] eq. (2), the authors calculate the activations $\mathbf{s}$ of a capsule[5] *before* any routing or non-linearity as

$$\mathbf{s} = \sum_i c_i\hat{\mathbf{u}}_i \ , \quad \hat{\mathbf{u}}_i = \mathbf{W}_i\mathbf{u}_i \quad . \tag{E.1}$$

Here, $\mathbf{u}_i$ is the capsule vector of capsule i from the incoming layer and $\mathbf{W}_i$ a matrix which transforms the capsule activations to generate the votes $\hat{\mathbf{u}}_i$, i.e., the vote of the i-th incoming capsule to the output capsule in the current layer. Note that c_i

[5] As we only discuss a single output capsule, we omitted the subscript j for the j-th output capsule for simplicity.

are the coefficients for the dynamic routing algorithm. If no routing is applied, they can be combined with $\mathbf{W}_i$ to yield $\widetilde{\mathbf{W}}_i = c_i \mathbf{W}_i$, which simplifies eq. (E.1) to

$$\mathbf{s} = \sum_i \hat{\mathbf{u}}_i \ , \quad \hat{\mathbf{u}}_i = \widetilde{\mathbf{W}}_i \mathbf{u}_i \quad . \tag{E.2}$$

Further, we note that the linear transformation of $\mathbf{u}_i$ can be represented as votes of the individual entries t of $\mathbf{u}_i$:

$$\hat{\mathbf{u}}_i = \widetilde{\mathbf{W}}_i \mathbf{u}_i = \sum_t \left[\mathbf{u}_i\right]_t \left[\widetilde{\mathbf{W}}_i^T\right]_t \quad . \tag{E.3}$$

Here, $\left[\mathbf{W}\right]_t$ denotes the t-th row in a matrix $\mathbf{W}$ (equivalently for a vector). Inserting this formulation of $\hat{\mathbf{u}}_i$ in eq. (E.2) yields

$$\mathbf{s} = \sum_i \sum_t \left[\mathbf{u}_i\right]_t \left[\widetilde{\mathbf{W}}_i^T\right]_t \quad . \tag{E.4}$$

Hence, $\mathbf{s}$ can be represented as the result of votes from all neurons u contained in any of the incoming capsules (note that we sum over all entries in all capsules). If we represent the activations of these neurons in a single vector $\mathbf{x}$, denote their activations by x_u, and their respective votes for the output capsule by $\mathbf{v}_u$ we can write eq. (E.4) as:

$$\mathbf{s} = \sum_i \sum_t \overbrace{\left[\mathbf{u}_i\right]_t}^{\widehat{=} x_u} \overbrace{\left[\widetilde{\mathbf{W}}_i^T\right]_t}^{\widehat{=} \mathbf{v}_u} = \sum_u x_u \mathbf{v}_u \quad . \tag{E.5}$$

The sum on the right hand side in eq. (E.5), in turn, can be expressed as a simple matrix-vector multiplication, such that

$$\mathbf{s} = \sum_u x_u \mathbf{v}_u = \mathbf{V}\mathbf{x} \tag{E.6}$$

with $\mathbf{v}_u$ as the columns of $\mathbf{V}$. The result is, of course, trivial and only shows that the capsule activations (a weighted sum of the inputs, eq. (E.1)) are obtained as a linear transformation of the input if no dynamic routing is applied.

Finally, we note that subsequent to this linear transformation, the authors in [S12] apply the squashing function SQ (see eq. (2) in our main paper) to the output vector $\mathbf{s}$, which yields the final capsule output $\mathrm{CAP}(\mathbf{x})$:

$$\mathrm{CAP}(\mathbf{x}) = \mathrm{SQ}(\mathbf{s}(\mathbf{x})) = \mathrm{SQ}(\mathbf{V}\mathbf{x}) \quad . \tag{E.7}$$

E.2. Differences to capsules

In the previous section we showed that the activation of a single capsule is computed as a linear transformation of all input activations, which is subsequently squashed (see eq. (E.7)). Comparing this with the computation of the DAU outputs (eq. (1) in our main paper), we see that this is equivalent to the dynamic weight computation if $\mathbf{V} = \mathbf{A}\mathbf{B}$ and $\mathbf{b} = \mathbf{0}$. As such, Dynamic Alignment Units and capsules are related. However, there are important differences, which we discuss in the following and summarise in Table E1.

First, in [S12], the squashed capsule activations $\mathrm{CAP}(\mathbf{x})$ are used as input to the next layer (after potentially applying the dynamic routing algorithm first). Instead of forwarding the squashed activations directly, in our DAUs they are used to linearly transform the input. Second, by factorising the matrix $\mathbf{V}$ into two matrices $\mathbf{A}\mathbf{B}$, we are able to control the rank of the linear transformation. Third, when computing the weights in the DAUs we allow for an additional bias term, which in the context of capsules can be considered a 'default vote'. Fourth, we generalise the non-linearity in the DAUs to any non-linearity that only changes the norm. Lastly, we do not apply dynamic routing in the DAUs.

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. 14, 15, 18, 23

[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*, 2010. 23

[S3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019. 22

[S4] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936. 26

[S5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 22

[S6] Yerlan Idelbayev. "Proper ResNet Implementation for CIFAR10/CIFAR100 in PyTorch". https://github.com/akamaster/pytorch_resnet_cifar10. Accessed: 2020-06-05. 22

[S7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 22

[S8] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR), Conference Track Proceedings*, 2015. 21

[S9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019. 21, 22

[S10] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *British Machine Vision Conference (BMVC)*, 2018. 23

[S11] 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. 23

[S12] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 14, 27, 28

[S13] 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. 23

[S14] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *International Conference on Machine Learning (ICML)*, 2017. 23

[S15] Suraj Srinivas and François Fleuret. Full-Gradient Representation for Neural Network Visualization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 23

[S16] 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. 23

[S17] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision (ECCV)*, 2014. 23

[S18] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-Down Neural Attention by Excitation Backprop. *Int. J. Comput. Vis.*, 2018. 23