Computer Vision and Image Understanding

journal homepage: www.elsevier.com

Agglomerator++: interpretable part-whole hierarchies and latent space representations in neural networks

Zeno Sambugaro[a,b], Nicola Garau[a,b], Niccoló Bisagno[a,b], Nicola Conci[a,b,**]

[a] *University of Trento,Via Sommarive 14, Trento, 38123, Italy*
[b] *CNIT - Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Via Sommarive 14, Trento, 38123, Italy*

ABSTRACT

Deep neural networks achieve outstanding results in a large variety of tasks, often outperforming human experts. However, a known limitation of current neural architectures is the poor accessibility in understanding and interpreting the network's response to a given input. This is directly related to the huge number of variables and the associated non-linearities of neural models, which are often used as black boxes. This lack of transparency, particularly in crucial areas like autonomous driving, security, and healthcare, can trigger skepticism and limit trust, despite the networks' high performance. In this work, we want to advance the interpretability in neural networks. We present Agglomerator++, a framework capable of providing a representation of part-whole hierarchies from visual cues and organizing the input distribution to match the conceptual-semantic hierarchical structure between classes. We evaluate our method on common datasets, such as SmallNORB, MNIST, FashionMNIST, CIFAR-10, and CIFAR-100, showing that our solution delivers a more interpretable model compared to other state-of-the-art approaches. Our code is available at `https://mmlab-cv.github.io/Agglomeratorplusplus/`.

1. Introduction

While deep neural networks consistently outperform humans across fields like computer vision (He et al. (2016)), natural language processing (Vaswani et al. (2017)), and data analysis (Sezer et al. (2020)), the achieved high performance often comes at the cost of increased complexity. It is a common practice to train and evaluate neural networks with billions of parameters, primarily guided by experience and heuristics. Consequently, understanding how an individual trainable parameter in the network setup directly affects the desired output from a given input becomes nearly impossible, as shown by Mallat (2016).

Interpretability, defined as the capacity to provide understandable explanations to humans (Linardatos et al. (2021); Doshi-Velez and Kim (2017)), is crucial in many areas, as shown by Grigorescu et al. (2020) in autonomous driving and by Sezer et al. (2020) in finance, although it is generally challenging to understand why a specific decision was made. In contrast, humans excel at understanding objects, their parts, and their interrelationships. We can categorize and recognize an object from its parts and infer its concept from its visual features, as introduced by Biederman (1987). This hierarchical representation is often missing in deep learning networks.

Various techniques have been introduced in the image

[**]Corresponding author.
e-mail: `zeno.sambugaro@unitn.it` (Zeno Sambugaro), `nicola.garau@unitn.it` (Nicola Garau), `niccolo.bisagno@unitn.it` (Niccoló Bisagno), `nicola.conci@unitn.it` (Nicola Conci)

classification field, such as transformers (Vaswani et al. (2017); Dosovitskiy et al. (2020)), neural fields (Mildenhall et al. (2020)), contrastive learning representation (Chen et al. (2020)), distillation (Hinton et al. (2015)), and capsules (Sabour et al. (2017)). These methods have improved interpretability to a certain extent. However, they often lack emphasis on data relationships or model-learned relationships, such as part-whole hierarchies. Inspired by the GLOM framework presented by Hinton (2021), our Agglomerator presented by Garau et al. (2022) aims to address these shortcomings. It integrates multiple methods, such as CNNs (Li et al. (2021a)), transformers, and positional encoding (Vaswani et al. (2017); Dosovitskiy et al. (2020)), contrastive learning representation (Chen et al. (2020)), distillation (Hinton et al. (2015)), and capsules (Sabour et al. (2017)).

Agglomerator++ is an evolution of our previous contribution, Agglomerator, which had proven effective for representing part-whole hierarchies while dealing with the image classification task. Agglomerator++ shares a similar structure but enhances the results while reducing the model size. Using a technique similar to that presented by Mildenhall et al. (2020), we introduced the positional encoding and we adopt a training procedure with masked patches as performed by Xie et al. (2022); He et al. (2022). The sample latent representation after the masked pre-raining allows the architecture to achieve better performances in the classification stage than the previously adopted contrastive pre-training, at a smaller computational cost. We conduct an extensive evaluation of Agglomerator++, focusing on the hyperparameters that are peculiar to our architecture and cannot be found in other neural networks, such as the number of levels and column structure (details in the coming paragraphs). Experimental results show that our model can compete with much larger models on big datasets while retaining a small number of parameters and outperforming capsule networks on smaller ones.

Our contribution is summarised as follows:

- we introduce a novel model, called Agglomerator++, mimicking the functioning of the cortical columns in the human brain (Hawkins (2021));
- we show how our architecture provides interpretability of *relationships contained in data*, namely the hierarchical organization of the feature space closely resembling human lexical similarities (Miller (1995));
- we provide results outperforming or on par with current methods on common datasets, such as SmallNORB (LeCun et al. (2004)), MNIST (LeCun et al. (1998)), Fashion-MNIST (Xiao et al. (2017)), CIFAR-10 and CIFAR-100 (Krizhevsky et al. (2009));
- we show how the input masking during the pre-training for self-supervised reconstruction leads to a better, more efficient neural representation than the previously deployed contrastive pre-training (Garau et al. (2022)).

2. Related work

In the domain of computer vision, a wide range of architectures exists, which have been employed for different tasks, such as classification, segmentation, and tracking. The breakthrough in the application of deep learning techniques, was achieved by Convolutional Neural Networks (CNNs) (He et al. (2016); Simonyan and Zisserman (2014)), which have since been surpassed by Transformers (Dosovitskiy et al. (2020); Khan et al. (2021)). Multi-Layer Perceptrons (MLPs) have resurged in recent years thanks to the availability of more powerful computers, since their straightforward architecture is offset by the huge model size. Capsules networks (Sabour et al. (2017); Hinton et al. (2018); Ribeiro et al. (2020)) and graph neural networks (GNNs) have emerged as the best architectures able to make the part-whole hierarchies in data emerge, with capsules getting the better edge in terms of interpretability of the response to a given input. In the following paragraphs, an extended overview of the techniques mentioned above is presented.

Convolutional Neural Networks (CNNs) (He et al. (2016); Simonyan and Zisserman (2014)) gained prominence in computer vision tasks, particularly by outperforming existing literature in image classification. However, their increased complexity and size have raised issues concerning their interpretability. Transformers, using self-attention and patch-based approaches, have surpassed CNNs, although they require substantial pre-training on large datasets for optimal performance, as discussed in Dosovitskiy et al. (2020); Khan et al. (2021).

Transformers (Dosovitskiy et al. (2020); Khan et al. (2021); Vaswani et al. (2017)) have surpassed CNNs in performance due to their self-attention and patch-based image analysis capabilities. Despite their more resource-intensive training, they effectively combine information from various image locations. They utilize positional encoding (Vaswani et al. (2017)) to maintain positional information, but typically require extensive pre-training on large datasets for optimal performance. Recently, advancements were made by learning to reconstruct samples from their partially masked versions (Xie et al. (2022)).

The research of the past decade has shown an interest in making part-whole hierarchies in data emerge. Capsule Networks (see Sabour et al. (2017); Hinton et al. (2018); Ribeiro et al. (2020)) and GNNs (Wu et al. (2020)) have shown promising results in this direction. In Capsule networks, the routing algorithm determines which capsules are activated to describe an object in the image, with lower-level capsules describing the parts, and higher-level capsules describing wholes. While effectively routing information from different locations in the image, activated capsules cannot describe every single possible object in the image, thus limiting their effectiveness on more complex datasets (e.g. ImageNet, CIFAR-100), while achieving state-of-the-art results on simpler ones (e.g. MNIST). Moreover capsules have been employed to model the time sequencing between images De Sousa Ribeiro et al. (2024); Ribeiro et al. (2022); Li et al. (2021b). Specifically, in De Sousa Ribeiro et al. (2024) they explore the adaptation of capsule networks to the video domain, addressing challenges in capturing motion information and scaling routing operations.

In recent times, there has been a shift towards biologically constrained AI, aiming to create deep learning networks that replicate the human brain's structure and functionality (Hawkins (2021)). The GLOM framework, which is based on inter-connected columns, each composed of auto-encoders sharing weights, has emerged in this context (Hinton (2021)). While initially GLOM was conceptualised as more of an intuition, it has now been developed into a fully functional system, named Agglomerator, specifically tailored for image classification tasks as presented by Garau et al. (2022). Other works Radwan and Shehata (2024, 2023) have demonstrated the efficacy of the Agglomerator pipeline, particularly with the combination of various pre-trained backbones to aid fine-tuning on larger datasets, achieving impressive results. In this work, we build upon the Agglomerator baseline by enhancing its performance, reducing the model size, and simplifying the pre-training process.

3. Method

The framework we propose aims to replicate a column-like pattern, similar to hyper-columns typical of the human visual cortex (Hawkins (2021)). The proposed network, called *Agglomerator++*, approaches the classification task in a patch-based fashion, constructing the so-called hypercolumns on top of each patch. Each hypercolumn consists of several layers which, from bottom to top, progressively agree on the same representation according to a part-whole paradigm, to the definition of the image content.

Compared to the previous solution proposed in Agglomerator by Garau et al. (2022), the method also benefits from incorporating unsupervised pre-training (Hamilton et al. (2022)), setting up the network as an autoencoder to reconstruct the masked input image as in Xie et al. (2022), which leads to a better neural representation. This procedure, combined with a Transformer-like self-attention (Vaswani et al. (2017)) mechanism on each layer of the columns, aims at reaching consensus between columns. Routing information with layer-based attention and stacked autoencoders allows GLOM to learn different levels of abstraction of the input at different locations and levels in the columns, creating a part-whole structure with a richer representation compared to capsule networks.

3.1. Preliminaries

Here, we introduce the mathematical notation needed to explain the details of the main building blocks of the architecture.

Each input image is transformed into a feature map divided into $N = h \times w$ patches. The n-th patch, with $n \in \{1, \ldots, N\}$, located at coordinates (h, w) is fed to the corresponding column $C(h, w)$.

As shown in Fig. 1, each column $C(h, w)$ consists of K embedding levels $\{l_t^{(h,w),k} \mid k = 0, \ldots, K\}$ connected by a stack of auto-encoders at location (h, w) and time $t \in \{0, 1, \ldots, T\}$, as suggested in Hinton (2021). The (h, w) notation is omitted in subsequent instances of l_t^k for better readability. Each level l_t^k of the column is a vector representation of size d, which encodes the patch information at position k at time t; l_t^{k-1} and l_t^k represent consecutive embedding levels; l_t^{k-1} represents a *part* of the *whole* l_t^k.

We denote the set of all levels L_t^k as the layer containing all l_t^k in all columns sharing the same k. Being K the last layer of our architecture at the last time step T, it is represented as L_T^K.

3.1.1. Patches embedding

At the embedding stage, as in Li et al. (2021a), we apply a convolutional tokenizer to extract the feature map of each image of size $H \times W$ pixels, to provide a richer representation compared to the original image. Following the implementation in Li et al. (2021a), the obtained feature map is of size $h \times w \times d$ where $h = H/H_p$ and $w = W/W_p$, with H_p and W_p being the dimensions in pixels of each patch. We then embed each d-dimensional embedding vector into the bottom levels $l_t^0 \in L_t^0$ at the corresponding coordinates (h, w) of the corresponding column $C(h, w)$. Feeding each patch to a spatially located column $C(h, w)$ resembles the positional encoding of neural fields (see Mildenhall et al. (2020)), where each d-sized embedding l_t^k represents both the sample and its relative observation viewpoint.

3.1.2. Hypercolumns

Every auto-encoder bridges consecutive levels in time and space within a column $C(h, w)$. Based on an MLP, these auto-encoders facilitate model reduction (Hinton et al. (2015)) and

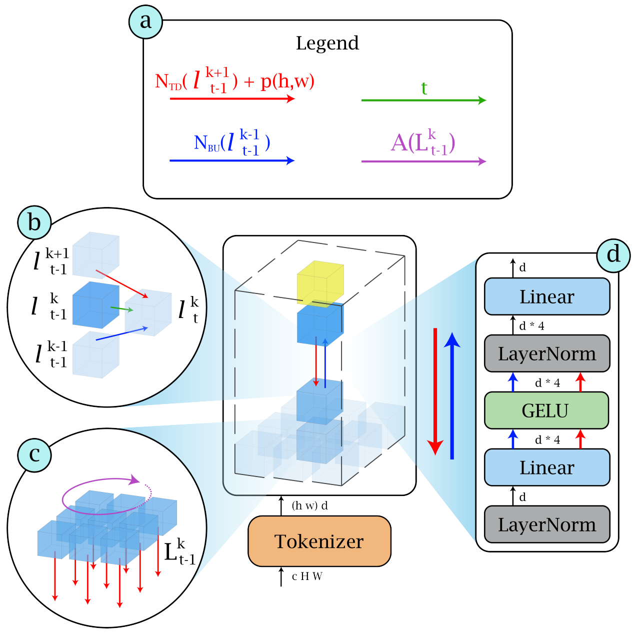


Fig. 1: Architecture of our Agglomerator++ model (center) with information routing (left) and detailed structure of building elements (right). Each *cube* represents a level l_t^k. **Top:** (a) legend for the arrows in the figure, representing the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the positional embedding $p(h, w)$, the bottom-up network $N_{BU}(l_{t-1}^{k-1})$, attention mechanism $A(L_{t-1}^k)$ and time step t. **Left:** (b) Contribution to the value of level l_t^k given by l_{t-1}^k, $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. (c) The attention mechanism $A(L_{t-1}^k)$ facilitates information sharing between $l_{t-1}^k \in L_{t-1}^k$. The positional embedding $p(h, w)$ is different for each column $C(h, w)$. All levels belonging to the same hyper-column $C(h, w)$ share the positional embedding $p(h, w)$. **Center:** bottom to top, the architecture consists of the tokenizer module, followed by the columns $C(h, w)$, with each level l_t^k connected to the neighbors with $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. **Right:** (d) Structure of the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the bottom-up network $N_{BU}(l_{t-1}^{k-1})$.

quicker training. The auto-encoder computes the top-down contribution of level l_{t-1}^{k+1} to the value of the lower level at the next time step l_t^k using a $N_{TD}(l_{t-1}^{k+1})$ top-down decoder. Likewise, it calculates the bottom-up contribution of level l_{t-1}^{k-1} to the value of the level above at the next time step l_t^k using a $N_{BU}(l_{t-1}^{k-1})$ bottom-up encoder.

3.1.3. Routing

The key element of our architecture is how the information is routed to obtain a representation of the input data where the part-whole hierarchies emerge. To propagate batch B through the network, to achieve deep image representations, we employ the *propagation phase*, fostering *consensus* between neighbor levels $l_t^k \in L_t^k$. This process yields similar values across the last layer's neighbor levels $l_t^K \in L_t^K$, each depicting the same *whole*, while lower layers share values amongst smaller groups signifying the same *part*. Vectors with similar values, or *islands of agreement*, symbolize image representation consensus at a given level (Hinton (2021)).

Attention weights. The attention $A(L_t^k)$ allows the information to be routed between embeddings belonging to the same layer. In our architecture we employ a simple version of the attention, which is similar to a weighted average of the embedding belonging to the same levels of the network (Xu et al. (2015)). Each attention weight Ω_n is computed as

$$\Omega_n = \frac{e^{\beta \lambda_n \cdot l_t^k}}{\sum e^{\beta \lambda_n \cdot a(\lambda_n)}} \tag{1}$$

where λ_n represents each possible level l_t^k belonging to the same layer L_t^k, $a(\lambda_n)$ is an indicator function, which indexes all the neighbors levels of λ_n belonging to the same layer L_t^k and β is a parameter that determines the sharpness of the attention.

3.2. Methodology

Hypercolums structure. Differently from Garau et al. (2022), $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$ share a similar structure, as shown in Fig. 1(d). Both these networks use GELU activation functions Hendrycks and Gimpel (2016). All $N_{TD}(l_{t-1}^{k+1}) \mid k = 0, \ldots, K$ networks share the same weights, as do all $N_{BU}(l_{t-1}^{k+1}) \mid k = 0, \ldots, K$ networks. This weight sharing among top-down decoders and bottom-up encoders significantly reduces the number of parameters and model size. Additionally, it promotes the part-whole hierarchy since identical networks connect embeddings at successive levels with similar hierarchical relationships (a part at level $k-1$ to its whole at level k), but varying representation abstractions (a whole at level k is a part relative to level $k+1$). Thus, the same auto-encoder can capture part-whole relationships independently from the level of abstraction.

Positional embedding. Similarly to Dosovitskiy et al. (2020), we define a set of positional embeddings to retain positional information. The same d-dimensional positional embedding $p(h, w)$ is shared among all the levels l_t^k belonging to the same column, one for each patch. As suggested in Hinton (2021), this embedding is used to obtain a rich image representation. Similar to neural fields, the positional embedding vector is added to the output of the top-down network and used by the bottom-up networks to propagate the information to upper levels, causing the emergence of *islands of agreement* between neighbor patches $a(\lambda_n)$ corresponding to neighbor object representations. The top-down network, using both object representations $a(\lambda_n)$ and positional encoding $p(h, w)$, decodesthe whole object representation into different parts. For instance, starting from a uniform vector l_{t-1}^{k+1} representing a dog's face (*whole*), the addition of positional embedding allows the network to decode this vector into diverse embeddings l_t^k, such as a mouth or ears (*parts*). Through the integration of positional embeddings, our network enables the emergence of *islands of agreement* without without relying on contrastive learning.

Propagation phase. At each time step $t \in \{1, \ldots, T\}$, we compute the values l_t^k as

$$\begin{aligned} l_t^k = avg(&\omega_l l_{t-1}^k, \omega_{BU} N_{BU}(l_{t-1}^{k-1}), \\ &\omega_{TD}(N_{TD}(l_{t-1}^{k+1}) + p(h, w)), \omega_A A(L_{t-1}^k)) \end{aligned} \tag{2}$$

where $avg()$ indicates the arithmetical average, and $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ are trainable weights.[1]

[1]For layer L_t^K, contribution $N_{TD}(l_{t-1}^{k+1})$ is not included, as L_t^{K+1} does not exist.

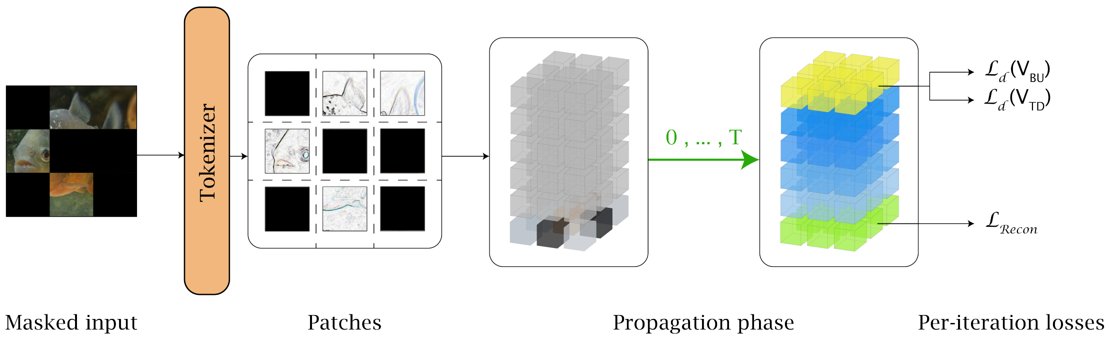


Fig. 2: **Pre-training to obtain a rich neural representation**. During the pre-training phase, a masked input is given to the network. The reconstruction loss $\mathcal{L}_{Recon}$ depends on how well the masked patches of the input image are reconstructed. The loss is attached to level L_t^1 because the network presents more detailed features at a lower level, while the representation becomes more abstract at higher levels, making them less suitable for reconstructing the input image. At the same time, enforcing the minimization of the regularisation losses $\mathcal{L}_d$ on the last level L_t^K encourages the network to display more definite *islands of agreement* at higher levels.

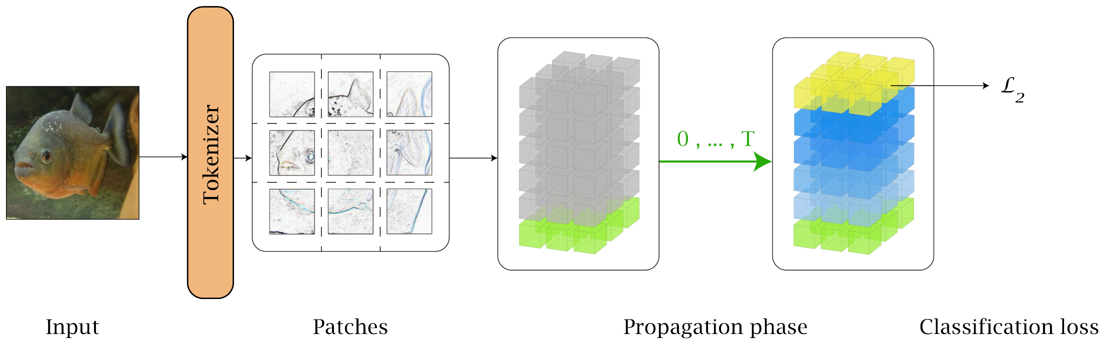


Fig. 3: **Training for image classification**. During the training phase, image samples are fed through the network to obtain their neural representation. The information is routed for at least $2K$ iterations to ensure that it is propagated through the network from the bottom level upward thanks to the bottom-up networks and back downward thanks to the top-down networks. The classification loss $\mathcal{L}_2$ is associated with the last level because the higher-level features are more suitable for the classification task.

Method	Reference	Backbone	Error %					# of params (Millions)	Training Arch.
			S-Norb	MNIST	F-MNIST	C-10	C-100		
E-CapsNet	Sabour et al. (2017)	Caps	2.54	0.26	-	-	-	0.2	GPU
CapsNet	Sabour et al. (2017)	Caps	2.70	0.25	6.38	10.65	82.00	6.8	GPU
Matrix-CapsNet	Hinton et al. (2018)	Caps	1.40	0.44	6.14	11.92	-	0.3	GPU
Capsule VB	Ribeiro et al. (2020)	Caps	1.60	0.30	5.20	11.24	-	0.2	GPU
ResNet-101	He et al. (2016)	Conv	-	2.10	5.10	6.41*	27.76*	23.6	GPU
VGG	Huang et al. (2016)	Conv	-	0.32	6.50	7.74*	28.05*	20.0	GPU
NesT-B	Zhang et al. (2022)	Transf	-	-	-	2.80	17.44	97.2	GPU
CRATE-S	Yu et al. (2023)	Transf	-	-	-	4.00	19.00	13.1	GPU
CRATE-B	Yu et al. (2023)	Transf	-	-	-	3.20	17.30	22.8	GPU
CRATE-L	Yu et al. (2023)	Transf	-	-	-	2.80	16.40	77.6	GPU
SparseSwin	Pinasthika et al. (2024)	Transf	-	-	-	2.57	14.65	17.6	GPU
ViT-L/16	Dosovitskiy et al. (2020)	Transf	-	-	-	0.85*	6.75*	631.9	TPU
ConvMLP-L	Li et al. (2021a)	Conv/MLP	-	-	-	1.40*	11.40*	42.7	TPU
MLP-Mixer-L/16	Tolstikhin et al. (2021)	MLP	-	-	-	1.66*	-	207.2	TPU
Agglom	Garau et al. (2022)	Conv/MLP/Caps	0.01	0.30	7.43	11.15	40.97	72.0	GPU
Agglom++	(**Ours**)	Conv/MLP/Caps	0.01	0.30	5.74	9.35	35.6	1.3	GPU

Table 1: Error percentages on the Top-1 accuracy results on datasets SmallNorb (S-Norb), MNIST, FashionMNIST (F-MNIST), CIFAR-10 (C-10), and CIFAR-100 (C-100). The $*$ notation indicates results obtained with networks pre-trained on ImageNet.

	Configuration	Error %	# of levels K	Levels embedding *d*	# of params (Millions)
I	Agglomerator Garau et al. (2022)	11.15	2	128	72.0
II	Agglomerator++ (Ours)	9.35	5	192	1.3
III	Decrease K	12.50	3	192	0.8
IV	Increase K	11.90	8	192	1.9

Table 2: Ablation study results of different Agglomerator configurations obtained on CIFAR-10 with a varying number of levels K.

	Configuration	Error %	# of levels K	Levels embedding *d*	# of params (Millions)
I	Agglomerator Garau et al. (2022)	11.15	2	128	72.0
II	Agglomerator++ (Ours)	9.35	5	192	1.3
IV	Decrease d	12.70	5	128	0.6
V	Increase d	9.55	5	512	9.0
VI	Without $A(L_t^k)$	13.99	5	192	1.1
VII	Without $p(h,w)$	13.03	5	192	1.3
VIII	Without regularization $\mathcal{L}$	9.57	5	192	1.3
IX	Without conv tokenizer	16.91	5	192	3.4

Table 3: Ablation study results of different Agglomerator configurations obtained on CIFAR-10 with a varying embedding size d and removing key components.

Masking %	Error %
20	9.39
50	**9.35**
80	9.51

Table 4: Ablation study results of different masking percentages with $K = 5$ and $d = 192$.

3.2.1. Training

The training procedure of our architecture is divided into two steps: (i) a pre-training phase where we train Agglomerator++ as an auto-encoder with a reconstruction loss for masked patches of the image (Xie et al. (2022)) coupled with a *consensus* regularization loss (Hinton (2021)) and (ii) a fine-tuning phase for the image classification task using a Cross-Entropy loss.

Pre-training to obtain a rich neural representation: As shown in Fig. 2 and similarly to Xie et al. (2022), we pre-train our network to reconstruct the masked pixels x_M of image input. Specifically, we set 50% of the total patches to random values before applying the convolutional tokenizer. The final neural image representation is derived from iterative information routing, with reconstruction loss $\mathcal{L}Recon$ attached to layer Lt^1. This detailed representation is a blend of low-level feature encoding and high-level abstract aggregation (Hinton (2021)). The reconstruction loss is defined as:

$$\mathcal{L}_{Recon} = \frac{1}{\Omega(x_M)} ||y_M - x_M||_1 \tag{3}$$

where $x, y \in \mathcal{R}^{3H_pW_p}$ are the input and predicted RGB pixel values, M denotes the set of masked pixels, $\Omega(.)$ is the number of total pixels, and $||.||_1$ is the 1-norm.

As suggested by Hinton (2021), we use a *consensus* loss to regularise the network and encourage the formation of *islands of agreement* at the top level of the architecture. We define the *consensus* vector V_t^k, the bottom-up loss vector $V_{BU}(L_t^K)$ and the top-down loss vector $V_{TD}(L_t^{K-1})$ as

$$V_t^k = l_t^k(1) \frown l_t^k(N) \tag{4}$$

$$V_{BU}(L_t^K) = N_{BU}(l_{t-1}^{K-1}(1)) \frown N_{BU}(l_{t-1}^{K-1}(N)) \tag{5}$$

$$V_{TD}(L_t^{K-1}) = N_{TD}(l_{t-1}^{K}(1)) \frown N_{TD}(l_{t-1}^{K}(N)) \tag{6}$$

where the $. \frown .$ operator denotes the concatenation between all the d-dimensional vectors belonging to the same layer, thus obtaining a vector of size $N \times d$.

Defining the cosine distance loss between two vectors x and y as $\mathcal{L}_d(x, y) = 1 - \cos(x, y) = \frac{x^T y}{||x||||y||}$, the resulting loss to be minimized at the pre-training stage is

$$\mathcal{L}_1 = \mathcal{L}_{Recon} + \mathcal{L}_d(V_{BU}(L_t^K), V_t^K) + \mathcal{L}_d(V_{TD}(L_t^{K-1}), V_t^{K-1}) \tag{7}$$

The last two terms act as a regularizer to improve the coherence of *islands of agreement*.

Fine tuning the architecture As shown in Fig. 3 Once the network is pre-trained using the $\mathcal{L}_1$ loss, we detach the pre-training losses from the architecture. To calculate the input to the cross-entropy loss for the classification, we compute the mean of all d-dimensional l_t^K vectors like in the work by Dosovitskiy et al. (2020). The resulting d-dimensional vector is fed through a layer normalization and a linear stage which reduces the size to c, namely the number of classes to be predicted for each dataset. Then we apply the standard cross-entropy function:

$$\mathcal{L}_2 = CE(y, \hat{y}) = -\frac{1}{c} \sum_{i=1}^{c} y_i \log(\hat{y}_i) \tag{8}$$

where y and $\hat{y}$ are the label and the prediction of the sample taken from the batch, respectively.

4. Experiments

We perform our experiments on the following datasets:

- **SmallNorb (S-NORB)** (LeCun et al. (2004)) is a dataset for 3D object recognition from shape.
- **MNIST** (LeCun et al. (1998)) and **FashionMNIST** (Xiao et al. (2017)).
- **CIFAR-10** and **CIFAR-100** (Krizhevsky et al. (2009)).

Our network is trained in an end-to-end fashion on a single NVIDIA GeForce RTX 3090. All datasets are employed at native resolution, but SmallNorb, which is resized to 32×32 pixels as defined by Ribeiro et al. (2020); Hinton et al. (2018). The tokenizer embedding creates $n = H/4 \times W/4$ patches represented by n d-dimensional vectors, where H and W are the pixels dimension of the input image. Thus, the corresponding number of columns is 8×8 for CIFAR-10, CIFAR-100, and SmallNorb, and 7×7 for MNIST FashionMNIST. During the pre-training, we set the following hyper-parameters: 1000 epochs, cyclic learning rate (Smith (2017)) in the range $[0, 1e^{-3}]$, batch size $B = 512$, levels embedding $d = 192$, number of levels $K = 5$, number of iterations $T = 2K = 10$, and weight decay $5e^{-2}$.

4.1. Quantitative results

Quantitative results from Tab. 1 indicate our Agglomerator++ performs well on both simple (SmallNorb, MNIST, Fashion MNIST) and complex datasets (CIFAR10, CIFAR100) without architecture modification or pre-training. It matches capsule-based models on simple datasets (Hinton et al. (2018);

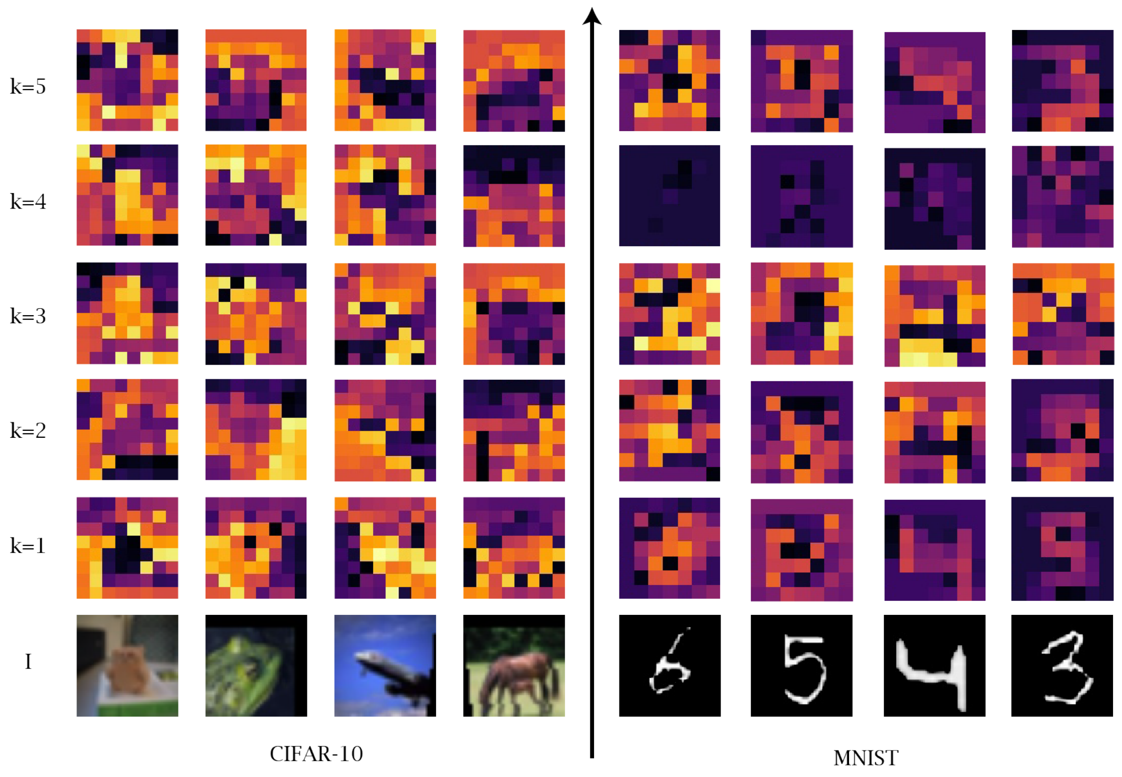


Fig. 4: Illustration of the evolving islands of agreement at varying K levels for MNIST and CIFAR-10 datasets samples. Displaying agreement vectors for each patch at each k level post 300 epochs of pre-training. Level $k = 1$ functions akin to a feature extractor with minimal neighbor agreement. Lastly, at level $k = 5$, two islands surface representing the object and the background.

Sabour et al. (2017); Ribeiro et al. (2020)), outperforms them on complex datasets, yet requires fewer parameters and training time than transformer-based (Dosovitskiy et al. (2020); Yu et al. (2023); Pinasthika et al. (2024); Zhang et al. (2022)) and MLP-based methods (Li et al. (2021a); Tolstikhin et al. (2021)). While convolutional models (He et al. (2016); Huang et al. (2016)) generalize well, they lack model interpretability. Agglomerator++ further enhances efficiency by sharing weights among networks and reducing extra layer requirements, thereby achieving comparable parameter count to capsule networks and superior performance on complex datasets. Agglomerator (Garau et al. (2022)) has fewer parameters than most transformer-based and MLP-based methods, and it requires less training time on a much smaller architecture. While improving the numerical results, Agglomerator++ further reduces the number of parameters by enforcing the sharing of the weights among all bottom-up and top-down networks, and it avoids building extra layers on top of the architecture to perform the contrastive pre-training and the classification as in our previous approach. Thus, the number of network parameters is now comparable with capsule-based networks. The improvement in performance on more complex datasets with respect to capsules highlights the expressive power of our d-dimensional embeddings with respect to a group of neurons.

Ablation study. Our Agglomerator++ architecture's key components—attention $A(L_t^k)$, positional embedding $p(h, w)$, number of levels K, and embedding dimension d—underwent ablation studies, showing their impact on performance. Moreover we ablate the percentages of masked patches.

In Tab. 2 we ablate the number of levels K. It illustrates how Agglomerator++ (I) surpasses the original network version (Garau et al. (2022)) (II) while decreasing parameter count through weight sharing across all networks and levels. This demonstrates that it is possible to add levels K to the network. Modifying the number of levels K (III)(IV) also affects performance, leading to a slight decrease. This likely occurs because

$K = 5$ is the optimal number of levels to construct a part-whole hierarchy that is sufficiently complex to capture the dataset's intricacies with respect to $K = 3$, without overcomplicating it as with $K = 8$.

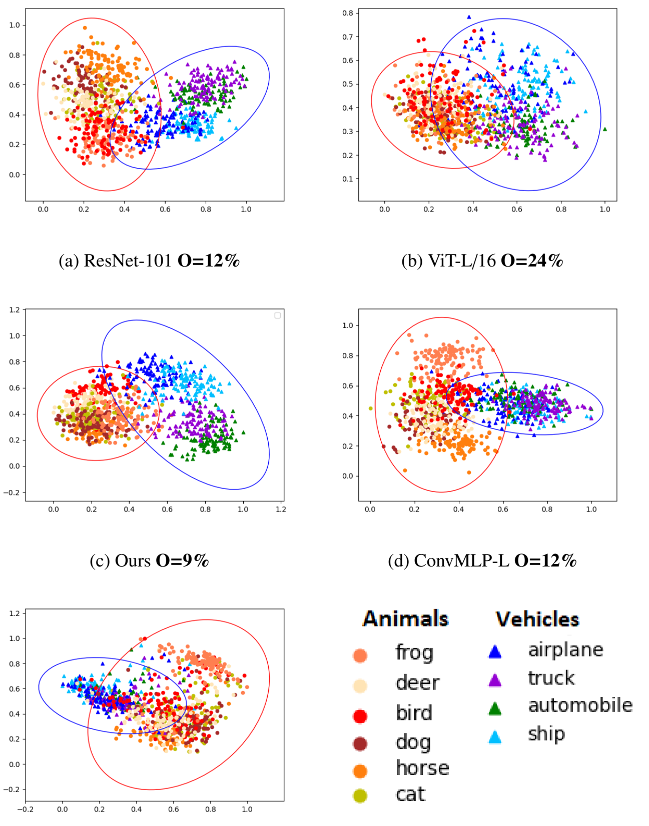
In Tab. 3 we ablate the embedding dimensions d, as well as other key components. Modifying the embedding dimensions d (IV)(V) has a huge impact on the number of parameters, with a smaller number leading to worse results (IV) and a bigger one leading to similar performance, but a huge increase in the number of parameters (V). Performance deteriorates when attention $A(L_t^k)$ (VI) or positional embedding $p(h, w)$ (VII) are removed, indicating their crucial role. The elimination of the two regularization losses $\mathcal{L}_d(x, y)$ (IX) has minor impact on performance, primarily affecting island formation. Removing the convolutional tokenizer (X) worsens results, hindering inter-patch information exchange at the embedding stage.

In Tab. 4 we ablate different masking percentages of the patches. The results show that masking a small percentage of the patches leads to a slight decrease in performance, as the network is not sufficiently challenged to learn representative features. Similarly, masking a too high percentage of patches proves too challenging for the network, resulting in worse performance.

4.2. *Qualitative results*

Our method provides interpretability of the *relationships learned by the model* by explicitly modeling the part-whole hierarchy, and of the *relationships contained in data* through the hierarchical organization of the feature space.

Part-whole hierarchy via island of agreement. In 4, we demonstrate that the model is able to capture meaningful hierarchical representations of the objects within the scenes. This is evidenced by the colour-coding of embedding vectors, where similar coloured patches denote spatial regions that the model associates with objects or their parts. Notably, as we progress up the hierarchy, (vertically along the columns), we observe a convergence of embedding vectors into predominantly two main "islands" in most examples. These islands represent the primary object and the background. More specifically, during



(a) ResNet-101 **O=12%** (b) ViT-L/16 **O=24%** (c) Ours **O=9%** (d) ConvMLP-L **O=12%** (e) Matrix-CapsNet **O=20%** (f) Legend

Fig. 5: The 2D latent space representation of multiple methods trained on the CIFAR-10 dataset through PCA is illustrated. The PCA reduces data from multidimensional to 2D. The legend (f) classifies samples into *Vehicles* and *Animals* following WordNet hierarchy (Miller (1995)). All methods (a,b,c,d,e) cluster samples between super-classes. The MLP-based methods (c,d) offer superior super-class separation, while placing similar samples together. Our method (c) optimizes inter-class and intra-class separability. The overlap percentage O denotes areas prone to severe hierarchical errors (Bertinetto et al. (2020)).

the *propagation phase*, neighbor levels on layer L_t^k are driven to form *islands of agreement*, showcasing part-whole hierarchies. Examples from MNIST and CIFAR-10 with $K = 5$ levels are shown. Instead of the convolutional tokenizer, a linear embedding, although lowering numerical results (Tab. 3), is used for better visualization. Vectors from the same island are grouped by cosine similarity between each embedding l_k^t and all level K embeddings.

Therefore, as k for L_t^k increases, neighbors tend to agree on the *whole* representation. Lower levels display smaller islands each representing a part. Our Agglomerator++ represents a patch at different abstraction levels and the same level's patches agree on the representation.

Latent space organization as the representation of conceptual-semantic relationship in data. Recent networks aim at maximizing inter-class distances and minimizing intra-class distances between samples in the latent space. While the accuracy is high, they provide little interpretability in their data representation. As a result, mistakes are less likely to happen, but the mistake severity, defined as the distance between two classes in WordNet lexical hierarchy (Miller (1995)), does not decrease (Bertinetto et al. (2020)). As shown in Fig. 5, our network semantically organizes the input data resembling the human lexical hierarchy, even though, differently from Garau et al. (2022), no contrastive loss is enforced. It is interesting to note that our network, unlike others, can enforce a more dispersed distribution of the samples belonging to the ”Vehicles” superclass. This distribution better mirrors our lexical hierarchy, as the vehicles sharing the most similarities with samples from the ”Animals” superclass are airplanes and birds. Additionally, the organization within the ”Animals” superclass shows closer resemblance to human lexical organization, with all quadrupeds clustered together in the latent space and birds positioned farther apart.

5. Conclusion

We presented Agglomerator++, a method that makes a step forward towards representing interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks. We believe that interpretable networks are key to the success of AI and deep learning. In this work, we show that by employing a self-supervised reconstruction training phase, we can significantly enhance the representations learned by our network, achieving better performance compared to similar networks trained with contrastive loss. The biological processes happening in our brains when learning are probably a combination of the two processes, which we view as the next step to obtain even richer and better explainable representations.

6. Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve readability and to correct typos. After using this tool/service, the author(s) reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Bertinetto, L., Mueller, R., Tertikas, K., Samangooei, S., Lord, N.A., 2020. Making better mistakes: Leveraging class hierarchies with deep networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12506–12515.

Biederman, I., 1987. Recognition-by-components: a theory of human image understanding. Psychological review 94, 115.

Chen, T., Kornblith, S., Norouzi, M., Hinton, G., 2020. A simple framework for contrastive learning of visual representations, in: International conference on machine learning, PMLR. pp. 1597–1607.

De Sousa Ribeiro, F., Duarte, K., Everett, M., Leontidis, G., Shah, M., 2024. Object-centric learning with capsule networks: A survey. ACM Computing Surveys 56, 1–291.

Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 .

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

Garau, N., Bisagno, N., Sambugaro, Z., Conci, N., 2022. Interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13689–13698.

Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G., 2020. A survey of deep learning techniques for autonomous driving. Journal of Field Robotics 37, 362–386.

Hamilton, M., Zhang, Z., Hariharan, B., Snavely, N., Freeman, W.T., 2022. Unsupervised semantic segmentation by distilling feature correspondences. arXiv preprint arXiv:2203.08414 .

Hawkins, J., 2021. A thousand brains: A new theory of intelligence.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R., 2022. Masked autoencoders are scalable vision learners, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition , 770–778.

Hendrycks, D., Gimpel, K., 2016. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 .

Hinton, G., 2021. How to represent part-whole hierarchies in a neural network. arXiv preprint arXiv:2102.12627 .

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 .

Hinton, G.E., Sabour, S., Frosst, N., 2018. Matrix capsules with em routing, in: International conference on learning representations.

Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q., 2016. Deep networks with stochastic depth, in: European conference on computer vision, Springer. pp. 646–661.

Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M., 2021. Transformers in vision: A survey. arXiv preprint arXiv:2101.01169 .

Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images .

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86.

LeCun, Y., Huang, F.J., Bottou, L., 2004. Learning methods for generic object recognition with invariance to pose and lighting, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., IEEE. pp. II–104.

Li, J., Hassani, A., Walton, S., Shi, H., 2021a. Convmlp: Hierarchical convolutional mlps for vision. arXiv preprint .

Li, J., Zhao, Q., Li, N., Ma, L., Xia, X., Zhang, X., Ding, N., Li, N., 2021b. A survey on capsule networks: Evolution, application, and future development, in: 2021 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), IEEE. pp. 177–185.

Linardatos, P., Papastefanopoulos, V., Kotsiantis, S., 2021. Explainable ai: A review of machine learning interpretability methods. Entropy 23, 18.

Mallat, S., 2016. Understanding deep convolutional networks. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences , 20150203.

Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R., 2020. Nerf: Representing scenes as neural radiance fields for view synthesis, in: European conference on computer vision, Springer.

Miller, G.A., 1995. Wordnet: a lexical database for english. Communications of the ACM 38, 39–41.

Pinasthika, K., Laksono, B.S.P., Irsal, R.B.P., Yudistira, N., et al., 2024. Sparseswin: Swin transformer with sparse transformer block. Neurocomputing 580, 127433.

Radwan, A., Shehata, M.S., 2023. Distilling part-whole hierarchical knowledge from a huge pretrained class agnostic segmentation framework, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 238–246.

Radwan, A., Shehata, M.S., 2024. Fedpartwhole: Federated domain generalization via consistent part-whole hierarchies. arXiv preprint arXiv:2407.14792 .

Ribeiro, F.D.S., Duarte, K., Everett, M., Leontidis, G., Shah, M., 2022. Learning with capsules: A survey. arXiv preprint arXiv:2206.02664 .

Ribeiro, F.D.S., Leontidis, G., Kollias, S., 2020. Capsule routing via variational bayes, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3749–3756.

Sabour, S., Frosst, N., Hinton, G.E., 2017. Dynamic routing between capsules. arXiv preprint arXiv:1710.09829 .

Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M., 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied Soft Computing 90, 106181.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

Smith, L.N., 2017. Cyclical learning rates for training neural networks, in: 2017 IEEE winter conference on applications of computer vision (WACV), IEEE. pp. 464–472.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al., 2021. Mlp-mixer: An all-mlp architecture for vision. arXiv:2105.01601 .

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

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y., 2020. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems 32, 4–24.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:cs.LG/1708.07747.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q., Hu, H., 2022. Simmim: A simple framework for masked image modeling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9653--9663.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention, in: International conference on machine learning, PMLR. pp. 2048--2057.

Yu, Y., Buchanan, S., Pai, D., Chu, T., Wu, Z., Tong, S., Haeffele, B., Ma, Y., 2023. White-box transformers via sparse rate reduction. Advances in Neural Information Processing Systems 36, 9422--9457.

Zhang, Z., Zhang, H., Zhao, L., Chen, T., Arik, S.Ö., Pfister, T., 2022. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3417--3425.