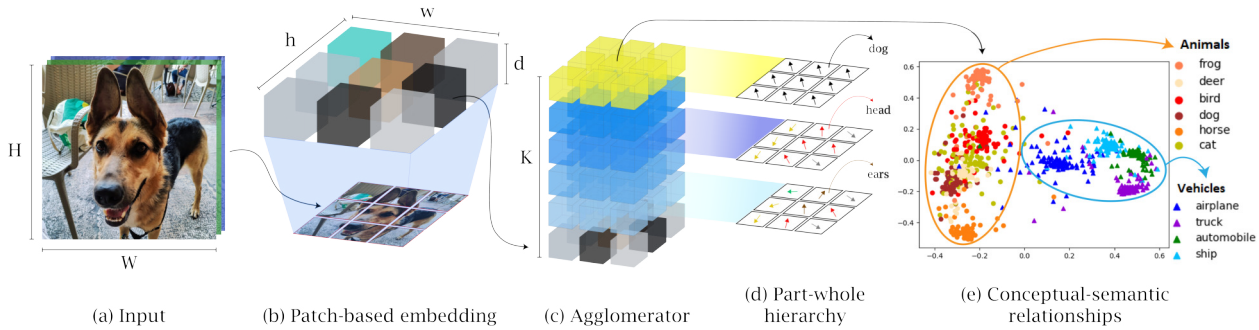
Interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks

Nicola Garau, Niccoló Bisagno, Zeno Sambugaro, and Nicola Conci
University of Trento - Department of Information Engineering and Computer Science - DISI
Via Sommarive, 9, 38123 Povo, Trento TN
nicola.garau,niccolo.bisagno,zeno.sambugaro,nicola.conci@unitn.it



(a) Input (b) Patch-based embedding (c) Agglomerator (d) Part-whole hierarchy (e) Conceptual-semantic relationships

Figure 1. **[Better seen in color]**. Overview of the proposed solution. Our Agglomerator is a novel architecture for vision applications, in which column structure (c) mimics hyper-columns typical of the human visual cortex [14]. The input data (a) is fed to the columns using a patch-based embedding (b). The Agglomerator architecture iteratively routes the information across its structure, creating a neural representation of each image, similar to neural fields [37]. In the neural representation, *part-whole* hierarchies (d) emerge at different levels of the columns. The same column can represent the same patch of the image with different levels of abstraction (e.g., the ears, the head, and the dog) corresponding to each level in the column. Neighbor columns agree on a *part* representation (e.g ears, head) at lower levels, ideally representing the same *whole* (e.g. dog) at the top level. The resulting feature space represents the *conceptual-semantic relationships* between data (e) resembling the human hierarchical organization [38]. Samples belonging to the same super-class (e.g., animals, vehicles) are clustered together, with conceptually close categories (e.g., birds and airplanes) represented on the edge of the super-classes.

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 to understand and interpret the network 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. When it comes to critical applications as autonomous driving, security and safety, medicine and health, the lack of interpretability of the network behavior tends to induce skepticism and limited trustworthiness, despite the accurate performance of such systems in the given task. Furthermore, a single metric, such as the classification accuracy, provides a non-exhaustive evaluation of most real-world scenarios. In this paper, we want to make a step forward towards interpretability in neural networks, providing new tools to interpret their behavior. We present Agglomerator, a framework capable of providing a representation of part-whole hierarchies from visual cues and organizing the input distribution matching the conceptual-semantic hierarchical structure between classes. We evaluate our method on common datasets, such as SmallNORB, MNIST, FashionMNIST, CIFAR-10, and CIFAR-100, providing a more interpretable model than other state-of-the-art approaches.

1. Introduction

The extensive adoption of neural networks and, in general, learning models has been raising concerns regarding our chances, as humans, to explain their behavior. Interpretability would be a highly desirable feature for neural networks, especially in those applications like autonomous driving [13], healthcare [40], and finance [45], where safety, life, and security are at stake.

Deep neural networks have achieved superhuman perfor-

mances in many domains, from computer vision [16, 29] to natural language processing [10, 54], and data analysis [45]. However, the achieved performances have come at the expense of model complexity, making it difficult to interpret how neural networks work [34]. These neural networks are usually deployed as "black boxes", with millions of parameters to be tuned, mostly according to experience and rule of thumb. Interpreting how a trainable parameter in the network setup directly affects the desired output from a given input has nearly zero chances.

According to the literature, interpretability is defined as *"the degree to which a human can understand the cause of a decision"* [39]. When a machine learning model reaches high accuracy on a task such as classification and prediction, can we trust the model without understanding why such a decision has been taken? The decision process is complex and we tend to evaluate the performance of a system in solving a given task using metrics computed at the end of the processing chain. While single metrics, such as the classification accuracy, reach super-human results, they provide an incomplete description of the real-world task [11]. As humans, when looking at an object that has eyes and limbs, we can infer via reasoning and intuition that these are elements (parts) that belong to the same entity (whole) [5], say an animal, and we can explain and motivate why such decision is taken, generally based on past experiences, beliefs and attitude [1]. Moreover, even in presence of an animal never seen before, we can probably tell from the visual features, our frames of reference [14] and our hierarchical organization of objects in the world [38] whether it is a fish or a mammal. We would like neural networks to display the same behavior, so that objects that are *close* in the conceptual-semantic and lexical relations are adjacent in the feature space as well (as shown in Fig. 1e). By doing so, it would be intuitive to identify hierarchical relations between samples and how the model has learned to build a topology describing each sample. Consequently, we can agree on the definition of interpretability in deep learning as the *"extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model"* [42].

In the image classification field, available techniques, such as transformers [10, 12, 54], neural fields [37], contrastive learning representation [7], distillation [19] and capsules [44], have achieved state-of-the-art performances, introducing a number of novelties, such as powerful attention-based features and per-patch analysis, positional encoding, similarity-based self-supervised pre-training, model compression and deep modeling of part-whole relationships. Taken as standalone, these methods have contributed to improving the interpretability of networks, while still lacking direct emphasis on either data relationships [7, 10, 12, 37, 54] (e.g. conceptual-semantic relationships) or model-learned relationships [19, 44] (e.g. part-whole relationships). Retrieving part whole hierarchy is not a new task per se, as it has been exploited in different research areas as scene parsing [3, 9] and multi-level scene decomposition [23, 59]. Instead of aiming at learning the part-whole hierarchy as the final goal of our architecture, we focus on learning the part-whole representation as a mean to interpret the network behavior at different levels. In [18], a concept idea on how to represent part-whole hierarchies in neural networks is introduced, which attempts to merge the advantages of the above state-of-the-art frameworks into a single theoretical system (known as *GLOM*). GLOM aims at mimicking the human ability in learning to parse visual scenes. Inspired by the theoretical concepts described in [14, 18], we build a working system, called Agglomerator, which achieves part-whole agreement [20] at different levels of the model (*relationships learned by the model*) and hierarchical organization of the feature space (*relationships contained in data*), as shown in Fig. 1.

Our contribution is summarised as follows:

- we introduce a novel model, called Agglomerator[1], mimicking the functioning of the cortical columns in the human brain [15];
- we explain how our architecture provides interpretability of *relationships learned by the model*, specifically part-whole relationships;
- we show how our architecture provides interpretability of *relationships contained in data*, namely the hierarchical organization of the feature space;
- we provide results outperforming or on par with current methods on multiple common datasets, such as SmallNORB [31], MNIST [30], FashionMNIST [57], CIFAR-10 and CIFAR-100 [27], also relying on fewer parameters.

2. Related work

Convolutional Neural Networks (CNNs) [16, 46] have risen to a prominent role in computer vision when they started to outperform the existing literature in the image classification task of the ImageNet challenge [28]. The convolution operator can effectively describe spatially-correlated data resulting in a feature map, while the pooling operation down-samples the obtained feature map by summarizing the presence of certain features in patches of the image. The pooling operation in CNNs has been the subject of criticism since it does not preserve the information related to the part-whole relationship [48] between features belonging to the same object [44].

[1]The code and the pre-trained models can be found at https://github.com/mmlab-cv/Agglomerator

Transformers [12, 25, 35] have proven able to outperform CNNs, thanks to their ability to encode powerful features using self-attention and patch-based analysis of images. Multi-headed transformers [10] require the query, key, and value weights to be trained differently for each head, which is more costly than training a CNN. The main advantage compared to CNNs is the ability of the multiple heads to combine information from different locations in the image with fewer losses than the pooling operation [32]. However, when compared with CNNs, Transformer-like models usually require intensive pre-training on large datasets, to achieve state-of-the-art performances.

Multi Layer Perceptrons (MLPs) [33, 52] are characterised by fully connected layers, in which each node is connected to every other possible node of the next layer. Even though they are easier to train and have simpler architecture compared to CNNs, the fully connected layers may cause the network to grow too fast in size and number of parameters, not allowing powerful scalability. MLPs have experienced a resurgence, thanks to patch-based approaches [33,52], that allowed reaching state-of-the-art performances. They can also be seen as 1x1 convolutions [18, 33, 52], which do not require the pooling operation.

Capsules networks [21,26,36,41,43,44] try to mimic the way the human brain creates a parse tree of parts and wholes by dynamically allocating groups of neurons (capsules) that can model objects at different levels of the part-whole hierarchy. The routing algorithm determines which capsules are activated to describe an object in the image, with lower-level capsules describing the parts (e.g. eyes and limbs), and higher-level capsules describing wholes (e.g. mammals and fish). 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). While part-whole hierarchies have been investigated in other fields like scene parsing [3,9] and multi-level scene decomposition [23,59], capsule networks aim at building an internal representation of the hierarchy, which allows for better interpretability of the final task (e.g. classification).

There has been a recent push toward the so-called biologically inspired Artificial Intelligence (AI) [14,22], which tries to build deep learning networks able to mimic the structure and functions of the human brain. In [14], the authors propose a column-like structure, similar to hypercolumns typical of the human neocortex. In [53], the authors build upon cortical columns implemented as separate neural networks called Cortical Column Networks (CCN). Their framework aims at representing part-whole relationships in scenes to learn object-centric representations for classification.

The author in [18] proposes a conceptual framework, called GLOM, based on inter-connected columns, each of which is connected to a patch of the image and is composed of auto-encoders stacked in levels. Weights sharing among MLP-based [33] auto-encoders allows for an easily trainable architecture with fewer weights, while knowledge distillation [19] allows for a reduction of the training parameters. The patch-based approach combined with the spatial distribution of columns allows for a sort of positional encoding and viewpoint estimation similarly to what is used in *neural fields* [37, 48]. At training time, the author recommends that GLOM should be trained using a contrastive loss function [7]. This procedure, combined with a Transformer-like self-attention [54] mechanism on each layer of the columns, aims at reaching a consensus between columns. Routing the information with layer-based attention and stacked autoencoders would theoretically allow GLOM to learn a different level of abstraction of the input at a different location and level in the columns, creating a part-whole structure with a richer representation if compared to capsule networks [44].

While GLOM is presented in [18] more as an intuition rather than a proper architecture, in this work we develop its foundational concepts and turn them into a fully working system, with application to image classification.

3. Method

The framework we propose aims at replicating the column-like pattern, similar to hyper-columns typical of the human visual cortex [14]. An overview is shown in Fig. 1.

Agglomerator brings together concepts and building blocks from multiple methods, such as CNNs [33], transformers [10, 12, 54], neural fields [37], contrastive learning representation [7], distillation [19], and capsules [44]. 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 $n \in \{1, \dots, N\}$ is fed to the corresponding column $C_n(h, w)$, spatially located at coordinates (h, w). The subscript n is omitted in the next equations for better readability. As shown in Fig. 2, each column $C(h, w)$ consists of K embedding levels $\{l_t^{(h,w),k} \mid k = 0, \dots, K\}$ connected by a stack of auto-encoders at location (h, w) at time $t \in \{0, \dots, t-1, t, t+1, \dots, T\}$, as suggested in [18]. The superscript (h, w) is omitted in the next instances of l_t^k for better readability. Each level l_t^k of the column is an embedding vector representation of size d. Levels l_t^{k-1} and l_t^k represent consecutive levels; l_t^{k-1} represents a *part* of the *whole* l_t^k. We indicate as $l_t^k \in L_t^k$ all the levels l_t^k in all columns $C(h, w)$ sharing the same k value and belonging to the same layer L_t^k. Being K the last layer of our architecture at the last time step T, it is represented as L_T^K.

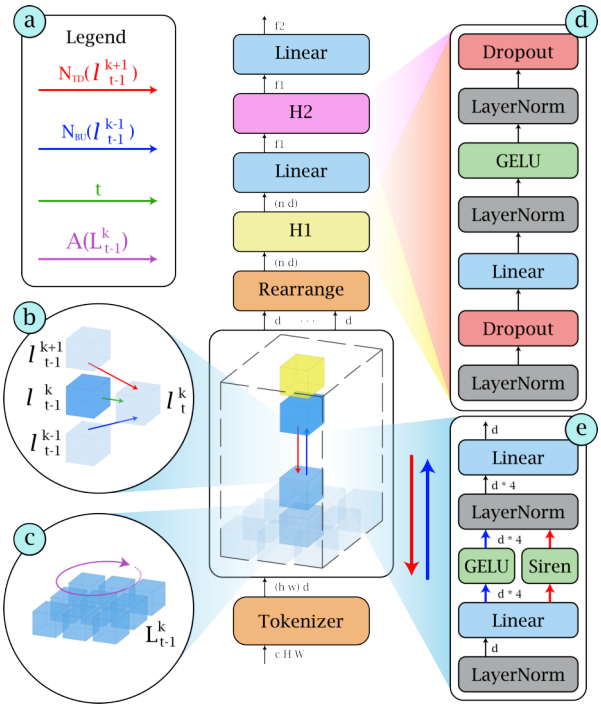


Figure 2. **[Better seen in color]**. Architecture of our Agglomerator model (center) with information routing (left) and detailed structure of building elements (right). Each *cube* represents a level l_t^k. **Left:** (a) legend of the arrows in the figure, representing the top-down network $N_{TD}(l_{t-1}^{k+1})$, the bottom-up network $N_{BU}(l_{t-1}^{k-1})$, attention mechanism $A(L_{t-1}^k)$ and time step t. (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 mechanisms $A(L_{t-1}^k)$ share information between $l_{t-1}^k \in L_{t-1}^k$. **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 $N_{BU}(l_{t-1}^{k-1})$. On top of the structure, the contrastive $H1$ and cross entropy $H2$ heads. **Right:** (d) structure of heads $H1$ and $H2$. (e) Structure of the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the bottom-up network $N_{BU}(l_{t-1}^{k-1})$.

3.1. Patches embedding

At the embedding stage, as in [33], we apply a convolutional Tokenizer to extract the feature map of each image of size $H \times W$ pixels, which provides a richer representation compared to the original image. Following the implementation in [33], the obtained feature map has size $h \times w \times d$ where $h = H/4$ and $w = W/4$. We then embed each of the n d-dimensional embedding vectors 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 the n-th each patch to a spatially located column $C(h, w)$ resembles the positional encoding of neural fields [37], where each d-sized embedding l_t^k represents at the same time the sample and its relative observation viewpoint. At each time step t, we embed each image sample into the first layer of the columns, which is represented as the bottom layer L_t^0.

3.2. Hypercolumns

Consecutive levels in time and space in a column $C(h, w)$ are connected by an auto-encoder. The auto-encoders are based on an MLP, which allows for model reduction [19] and faster training time. Each auto-encoder computes the top-down contribution of a level l_{t-1}^{k+1} to the value of the level below at the next time step l_t^k using a $N_{TD}(l_{t-1}^{k+1})$ top-down decoder. Similarly, each auto-encoder computes the bottom-up contribution of a 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. $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$ share a similar structure, but for the activation functions, as described in Fig. 2(e). The top-down network uses GELU activation functions [17], while the bottom up network relies on sinusoidal activation functions [47, 50, 56]. All the $N_{TD}(l_{t-1}^{k+1})$ connecting L_{t-1}^{k+1} to layer L_t^k share the same weights. The same is true for the $N_{BU}(l_{t-1}^{k-1})$ connecting L_{t-1}^{k-1} to layer L_t^k.

3.3. 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.

Before computing the loss, we need to iteratively propagate each batch N through the network, obtaining a deep representation of each image. This procedure, *propagation phase*, encourages the network to reach *consensus* between neighbor levels $l_t^k \in L_t^k$. Ideally, this means that all neighbor levels in the last layer $l_t^K \in L_t^K$ should have similar values, representing the same *whole*; neighbor levels at bottom layers $l_t^k \in L_t^k | k \neq K$ should instead share the value among smaller groups, each group representing the same *part*. Group of vectors that "agree" on a similar value have reached the *consensus* on the image representation at that level, and they are called *islands of agreement* [18]. An example of such representation is shown in Fig. 1(d). In capsules-based approaches [44], group of neurons are activated to represent the part-whole hierarchy with limited expressive power. Our d-dimensional layers l_t^k provide a richer representation of the same hierarchy.

To obtain such representation, at time step $t = 0$, we randomly initialise all the values l_0^k and we embed a batch of B samples into the bottom layer L_0^0. Once the values are initialized, we compute the attention $A(L_t^k)$. Instead of the self-attention mechanism used in Transformers [10, 12, 54], a standard attention weighting is deployed as in [58]. Each attention weight Ω_n is computed as

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

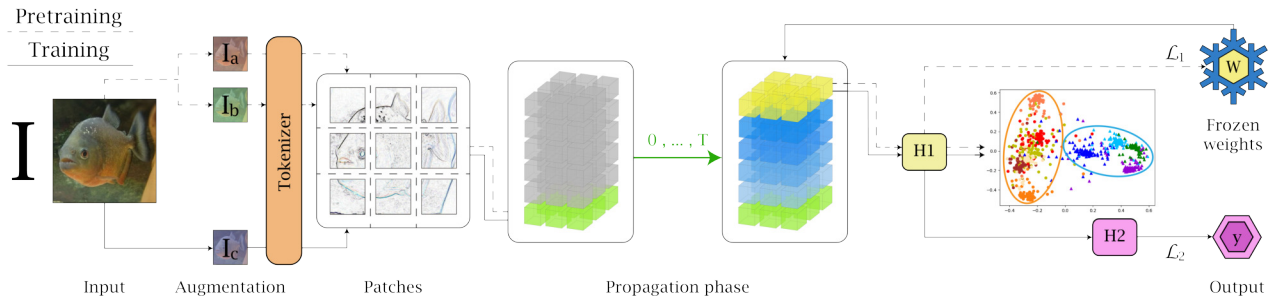


Figure 3. **Contrastive pre-training (dashed lines)** and **supervised training (continuous lines)** procedures. During the contrastive pre-training, two images I_a and I_b are produced by applying random data augmentation to the input image I. Through the Tokenizer, we compute feature maps for both I_a and I_b, which are then divided in patches and embedded into the bottom layer of the columns L_t^0. During the *propagation phase*, the information is routed through the Agglomerator architecture to obtain the neural representation L_T^K for each sample. We pre-train the network with the contrastive head $H1$ using a supervised contrastive loss $\mathcal{L}_1$, obtaining weights W. During the supervised training, we first load the frozen weights W in the network. Then, augmentation RandAugment [8] is applied on the input image I to obtain I_c, which follows the same steps as the pre-training phase. The network, with the classification head $H2$, is trained for the classification task by minimizing the cross-entropy loss $\mathcal{L}_2$.

where λ_n represents each possible level l_t^k belonging to the same layer L_t^k as l_t^k, $N(\lambda_n)$ 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.

At each time step $t \mid t \in \{1, \ldots, T\}$, a batch with B samples is fed to the bottom layer L_t^0 network as described in Sec. 3.1. We compute the values l_t^k as

$$l_t^k = avg(\omega_l l_{t-1}^k, \omega_{BU} N_{BU}(l_{t-1}^{k-1}), \omega_{TD} N_{TD}(l_{t-1}^{k+1}), \omega_A A(L_{t-1}^k)) \quad (2)$$

where $avg()$ indicates the arithmetical average, and $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ are trainable weights. For layer L_t^K, contribution $N_{TD}(l_{t-1}^{k+1})$ is not included, as L_t^{K+1} does not exist. The *propagation phase* takes T time steps to reach the final representation of each image at each layer L_k^T.

3.4. Training

The training procedure of our architecture is shown in Fig. 3. It is divided in two steps: (i) a pre-training phase using a supervised contrastive loss function [7] and (ii) a training phase for the image classification using a Cross-Entropy loss.

We first pre-train our network using an image-based contrastive loss [7]. Given a batch with B samples, we duplicate each image I to obtain pairs of samples (I_a, I_b), for a total of $2B$ data points. We then apply data augmentation RandAugment [8] to both (I_a, I_b). Both samples are fed to the network as described in Sec. 3.1, and we perform the propagation phase in Sec. 3.3 to obtain the representation at the last layer L_T^K. Then we rearrange the n levels $l_T^K \in L_T^K$ to obtain a vector of dimensions $n \times d$, given as input to the contrastive head $H1$, as described in Fig. 2. At the output of the contrastive head, each sample is described by a feature vector of dimension $f1$. We take all the possible sample pairs (I_a, I_b) from the batch and we compute the contrastive loss defined as:

$$\mathcal{L}_1 = ContrLoss(I_a, I_b) = -log \frac{e^{sim(I_a, I_b)}}{\sum_{k=1}^{2B} \mathcal{I}_{[k \neq a]} e^{sim(I_a, I_b)}} \quad (3)$$

where $sim(u, v) = u^T v / \|u\| \|v\|$ indicates the dot product between the normalised version of u and v, $\mathcal{I}_{[k \neq a]}$ is a indicator function valued 0 if k and a belong to the same class, and 1 otherwise.

Once the network is pre-trained using the contrastive loss, the weights are frozen. We apply augmentation [8] to each sample I_c in a batch of size B, which is then fed to the network for the *propagation phase* to obtain for each sample the representation L_T^K. Then, the cross-entropy head $H2$ is added on top of the contrastive head $H1$. A linear layer resizes $f1$-dimensional features to dimension $f2$, which corresponds to the number of classes to be predicted for each dataset. The new layers are then trained using the cross-entropy function:

$$\mathcal{L}_2 = CE(y, \hat{y}) = -\frac{1}{f2} \sum_{i=1}^{\text{f2}} y_i \log(\hat{y}_i) \quad (4)$$

where y is the label of a sample taken from the batch and $\hat{y}$ is the label to be predicted.

4. Experiments

We perform our experiments on the following datasets:

SmallNorb (S-NORB) [31] is a dataset for 3D object recognition from shape. It consists of roughly 200000 images of size 96×96 pixels of 5 classes of toys.

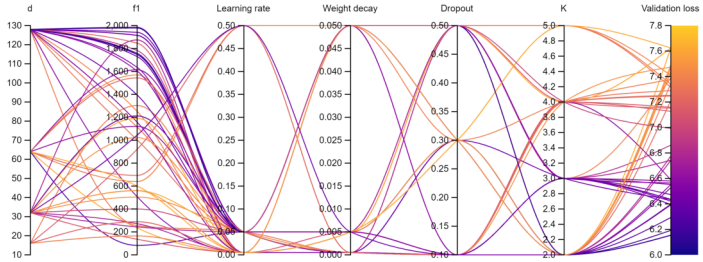


Figure 4. Hyper-parameters sweep. Each line represents a combination of parameters setup, with the darker lines representing the models achieving the lowest validation loss. Image obtained with [6].

	Configuration	Error % (after 100 epochs)
I	Vanilla (proposed)	12.8
II	ReLU activation only	12.6
III	Without attention	12.7
IV	Linear columns layers	13.5
V	Linear contrastive head	15.8
VI	Linear embedding	17.2

Table 1. Ablation study results of different Agglomerator configurations obtained on CIFAR-10 trained for 100 epochs.

MNIST [30] and **FashionMNIST** [57], consist of 60000 training images and 10000 test images of grayscale handwritten digits and Zalando's articles of size 28×28 pixels.

CIFAR-10 and **CIFAR-100** [27] both consist of 50000 training images and 10000 test images of size 32×32 pixels, with 10 and 100 classes, respectively.

Our network is trained in an end-to-end fashion using PyTorch Lightning on a single NVIDIA GeForce RTX 3090. Input images for each dataset are normalized using each standard dataset's normalization. We train our network on each dataset's native resolution, except for SmallNorb, which is resized to 32×32 pixels, following the standard procedure as in [21, 43]. 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 deploy the following hyper-parameters: 300 epochs, cyclic learning rate [49] in the range $[0.002, 0.05]$, batch size $B = 1024$, levels embedding $d = 128$, number of levels $K = 3$, number of iterations $T = 2K = 6$, dropout value 0.3, contrastive features dimension $f1 = 512$, and weight decay $5e^{-4}$. During the training phase, we resume the network training with the same hyper-parameters, $f2$ being the number of classes corresponding to each dataset.

5. Quantitative results

We report the quantitative results for each dataset in Tab. 2. Capsule-based models [21, 36, 41, 43, 44] can achieve good performances on simple datasets (SmallNorb, MNIST, and FashionMNIST), but they fail to generalize to datasets with a higher number of classes (CIFAR-100). Convolutional-based models [2, 16, 24, 46] can generalize to different datasets, at the expense of weak model interpretability, mainly due to the max-pooling operation. Transformer-based [12] and MLP-based methods [33, 52] are able to achieve the best performances on more complex datasets, but they do not provide tests for smaller datasets. However, to achieve such levels of accuracy they rely on long pretraining (thousands of TPU days) on expensive computational architectures, implementing data augmentation on ImageNet [28] or the JFT-300M [51] dataset, not available publicly. As can be seen, our method performs on par with capsule-based methods on simpler datasets, while achieving better generalization on more complex ones. In addition, our method has fewer parameters than most transformer-based and MLP-based methods, and it can be trained in less time on a much smaller architecture.

Ablation study. We analyze the contribution of the different components of our architecture evaluating their influence on the validation loss after 50 epochs. The considered parameters, in descending order of correlation with the validation loss value are: the embedding dimension d, the contrastive feature vector $f1$, learning rate, weight decay, dropout, and the number of levels K. The results are reported in Fig. 4. We perform 50 different training on CIFAR-10 with different combinations of parameters.

In Tab. 1 we show how our network configuration (I) performs similarly with (II) and (III). Both sinusoidal activations and shared attention in (I) are key to providing interpretable results, allowing islands of *agreement to emerge*. Simplified versions using only a linear layer instead of column layers (IV), of the contrastive head (V), or of the linear embedding (VI) lead to a decrease in performance.

6. Qualitative results: interpretability

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.

Island of agreement as a representation of multi-level part-whole hierarchy. During the *propagation phase*, neighbor levels on the same layer L_t^k are encouraged to reach a *consensus* by forming *islands of agreement*. The *islands of agreement* represent the part-whole hierarchies at different levels. In Fig. 5, we provide a few examples of the islands of agreement obtained on MNIST and CIFAR-10

Method	Ref	Backbone	Error %					# of params (Millions)	Training Arch.
			S-Norb	MNIST	F-MNIST	C-10	C-100		
E-CapsNet	[36]	Caps	2.54	0.26	-	-	-	0.2	GPU
CapsNet	[41, 44]		2.70	0.25	6.38	10.6	82.00	6.8	GPU
Matrix-CapsNet	[21]		1.40	0.44	6.14	11.9	-	0.3	GPU
Capsule VB	[43]		1.60	0.30	5.20	11.2	-	0.2	GPU
ResNet-110	[2, 16, 24]	Conv	-	2.10	5.10	6.41*	27.76*	1.7	GPU
VGG	[2, 46]		-	0.32	6.50	7.74*	28.05*	20	GPU
ViT-L/16	[12]	Transf	-	-	-	0.85*	6.75*	632	TPU
ConvMLP-L	[33]	Conv/MLP	-	-	-	1.40*	11.40*	43	TPU
MLP-Mixer-L/16	[52]	MLP	-	-	-	1.66*	-	207	TPU
Ours		Conv/MLP/Caps	0.01	0.30	7.43	11.15	40.97	72	GPU

Table 2. 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.

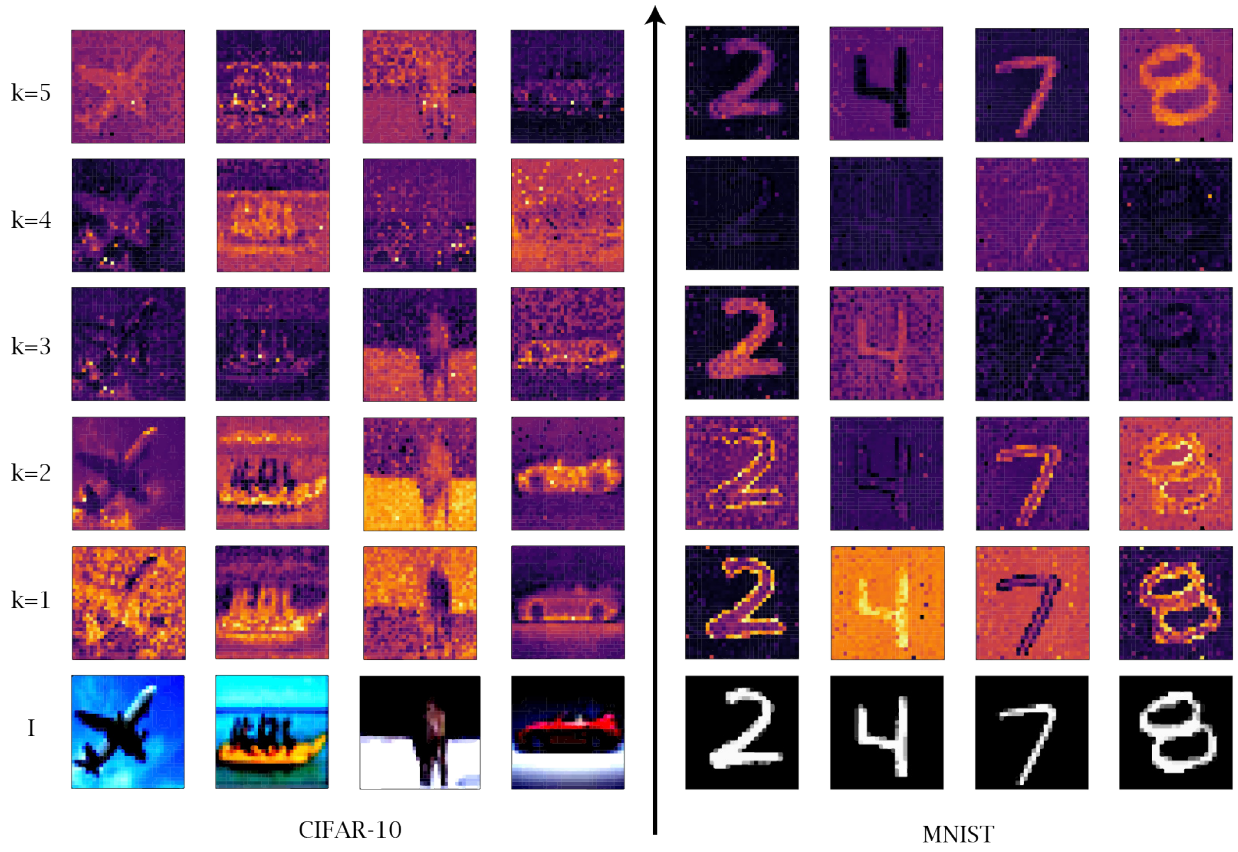
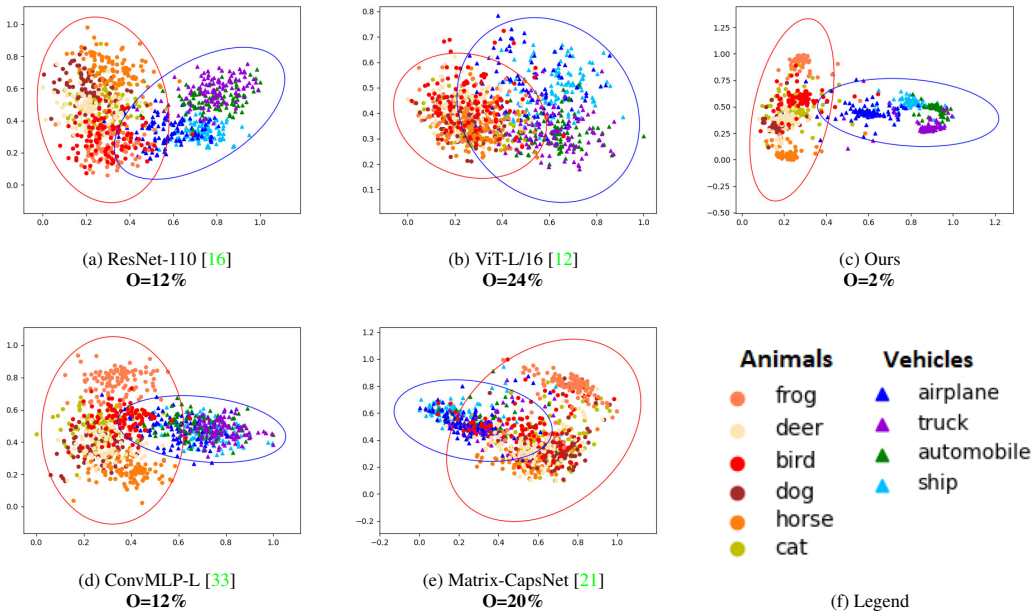


Figure 5. Vectorial representation of emerging islands of agreement at different K levels of sample from MNIST and CIFAR-10 datasets. We show the vectors of agreement for each patch at each level k after 100 epochs of contrastive pre-training. At level $k = 1$, the network acts similarly to a feature extractor, where each cell represent a spatial feature with little agreement between neighbors. At intermediate levels, $k = 2, 3, 4$ neighbor cells reach agreement on specific parts of the image, creating different island for different part of the plane. At the last level $k = 5$, two island emerge, agreeing on the representation of the object and of the background. Since we are training the network for the classification task, the distance between the color of the two island is small since all the parts of the image tend to agree that the image represents the same whole.

trained with $K = 5$ levels. Each arrow represents the value of a level l_k^t at location (h, w), reduced from d-dimensional to 2D using a linear layer. As k for L_t^k increases, neighbor $l_k^t \in L_t^k$ tend to agree on a common representation of the *whole* represented in the image sample. At lower levels, smaller islands emerge, each representing a part of the *whole*. Samples of MNIST present fewer changes in the islands across levels because the data is much simpler, indi-



(a) ResNet-110 [16] **O=12%**

(b) ViT-L/16 [12] **O=24%**

(c) Ours **O=2%**

(d) ConvMLP-L [33] **O=12%**

(e) Matrix-CapsNet [21] **O=20%**

(f) Legend

Figure 6. 2D representation of the latent space for multiple methods trained only on the CIFAR-10 dataset obtained using Principal Component Analysis (PCA) [55]. The PCA provides a deterministic change of base for the data from a multidimensional space into a 2D space. The legend (f) displays the classes, which are divided between super-classes *Vehicles* and *Animals* following the WordNet hierarchy [38]. The different methods (a,b,c,d,e) are all able to cluster the samples between the two super-classes. However, while (a,b,e) display a latent space where classes are close to each other, the two MLP-based methods (c,d) are able to provide a clearer separation between the super-classes. Both methods show conceptual-semantically close samples on the edge of each superclass, such as airplanes and birds. Inside each superclass, semantically close samples are represented contiguously, such as deers and horses, or cars and trucks. Our method (c) provides better inter-class and intra-class separability. The overlap percentage O is reported for each method. The overlap area is the area where a mistake with a higher hierarchical severity [4] has a higher probability to occur.

cating that fewer levels in the hierarchy can be sufficient to obtain similar results. Our Agglomerator is thus able to represent a patch differently at different levels of abstraction. At the same level, spatially adjacent patches take the same value, agreeing on the representation of parts and wholes.

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 [38], does not decrease [4]. As shown in Fig. 6, our network semantically organizes the input data resembling the human lexical hierarchy.

7. Limitations

Our method introduces new types of hyper-parameters in the network structure, such as embedding dimensions, number of levels, and size of patches, which need to be tuned. We believe a better parameters setting can be found for all the datasets, increasing accuracy while still retaining interpretability. Moreover, a higher number of parameters generally causes architectures to be more prone to over-fitting and more difficult to train. To improve the accuracy of our network, we would need a pre-training on large datasets (e.g., on ImageNet), which requires large computational resources to be performed in a reasonable time frame. While hoping that powerful TPU architectures become publicly available in the future, we are currently investigating efficient pre-training strategies for our network.

8. 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 artificial intelligence and deep learning. With this work, we intend to promote a preliminary implementation and the corresponding results on the image classification task, and we hope to inspire other researchers to adjust our solution to solve more complex and diverse tasks.

References

[1] Dolores Albarracin and Robert S Wyer Jr. The cognitive impact of past behavior: influences on beliefs, attitudes, and future behavioral decisions. *Journal of personality and social psychology*, 79(1):5, 2000. 2

[2] Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Denser: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, 20(1):5–35, 2019. 6, 7

[3] Daniel M Bear, Chaofei Fan, Damian Mrowca, Yunzhu Li, Seth Alter, Aran Nayebi, Jeremy Schwartz, Li Fei-Fei, Jiajun Wu, Joshua B Tenenbaum, et al. Learning physical graph representations from visual scenes. *arXiv preprint arXiv:2006.12373*, 2020. 2, 3

[4] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12506–12515, 2020. 8

[5] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 2

[6] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 6

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2, 3, 5

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 5

[9] Fei Deng, Zhuo Zhi, Donghun Lee, and Sungjin Ahn. Generative scene graph networks. In *International Conference on Learning Representations*, 2020. 2, 3

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 3, 4

[11] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017. 2

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3, 4, 6, 7, 8

[13] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. 1

[14] Jeff Hawkins. A thousand brains: A new theory of intelligence, 2021. 1, 2, 3

[15] Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in neural circuits*, 11:81, 2017. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 6, 7, 8

[17] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[18] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021. 2, 3, 4

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3, 4

[20] Geoffrey E Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990. 2

[21] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018. 3, 6, 7, 8

[22] Kjell Jørgen Hole and Subutai Ahmad. A thousand brains: toward biologically constrained ai. *SN Applied Sciences*, 3(8):1–14, 2021. 3

[23] Yining Hong, Li Yi, Josh Tenenbaum, Antonio Torralba, and Chuang Gan. Ptr: A benchmark for part-based conceptual, relational, and physical reasoning. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 3

[24] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 6, 7

[25] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021. 3

[26] Adam R Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. *arXiv preprint arXiv:1906.06818*, 2019. 3

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 6

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 2, 6

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 2

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2, 6

[31] Yann LeCun, Fu Jie Huang, and Leon Bottou. 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.*, volume 2, pages II–104. IEEE, 2004. 2, 5

[32] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019. 3

[33] Jiachen Li, Ali Hassani, Steven Walton, and Humphrey Shi. Convmlp: Hierarchical convolutional mlps for vision. *arXiv preprint arXiv:2109.04454*, 2021. 3, 4, 6, 7, 8

[34] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2021. 2

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 3

[36] Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. Efficient-capsnet: Capsule network with self-attention routing. *arXiv preprint arXiv:2101.12491*, 2021. 3, 6, 7

[37] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 3, 4

[38] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 1, 2, 8

[39] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019. 2

[40] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018. 1

[41] Rinat Mukhometzianov and Juan Carrillo. Capsnet comparative performance evaluation for image classification. *arXiv preprint arXiv:1805.11195*, 2018. 3, 6, 7

[42] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. 2

[43] Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos Kollias. Capsule routing via variational bayes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3749–3756, 2020. 3, 6, 7

[44] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017. 2, 3, 4, 6, 7

[45] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020. 1, 2

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 6, 7

[47] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 4

[48] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 2, 3

[49] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017. 6

[50] Josep M Sopena, Enrique Romero, and Rene Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. 1999. 4

[51] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 6

[52] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 3, 6, 7

[53] Toon Van de Maele, Tim Verbelen, Ozan Catal, and Bart Dhoedt. Disentangling what and where for 3d object-centric representations through active inference. *arXiv preprint arXiv:2108.11762*, 2021. 3

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3, 4

[55] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 8

[56] Kwok-wo Wong, Chi-sing Leung, and Sheng-jiang Chang. Handwritten digit recognition using multilayer feedforward neural networks with periodic and monotonic activation functions. In *Object recognition supported by user interaction for service robots*, volume 3, pages 106–109. IEEE, 2002. 4

[57] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 2, 6

[58] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015. 4

[59] Song-Chun Zhu and David Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007. 2, 3