Modular Neural Networks with Top-Down and Bottom-Up Signals
Final Project Report

Charbel-Raphaël Ségerie * Hugo Laurençon †
École normale supérieure Paris-Saclay

Abstract

To fully understand information it obtains, the human brain combines top-down and bottom-up signals. Bottom-up signals result from direct perception, while top-down signals take into account past experience. The combination of these two types of signals makes it possible to make a more confident decision when one type is not sufficient for a task. The incorporation of this phenomena in neural networks would allow to increase their performance, because they are currently mostly based only on bottom-up signals, like LSTMs or CNNs. On the other hand, neural networks become difficult to interpret because they are getting bigger and bigger. Encouraging a neural network to be modular would not only increase its performance, but also its interpretability. In this study, we propose an architecture allowing to combine top-down and bottom-up signals with attention, but also to favor modularity by building several bricks within a neural network that can communicate with each other, again through an attention mechanism. We test this new architecture on an image classification task, and we test its ability to generalize on images slightly different than what it was trained on. We obtain reasonable performance on these tasks. These results are encouraging to pursue a more in-depth study on more complicated tasks.

1. Introduction

The attention mechanism is currently used to achieve state-of-the-art performance in many NLP tasks [10], but it is also increasingly used in image classification problems, with performance comparable to that of CNN. One of the strengths is that it provides a means of comparison that allows similar quantities to be grouped together in a certain sense, so that they can help each other without looking at unhelpful information.

In addition, two top-down and bottom-up mechanisms work in parallel when looking at an image. For example, when you are trying to read a word where one of the letters is badly written, you can focus on that letter to get an a priori, but you will also look at the word as a whole to guess the missing letter. We then understand that it can be advantageous to combine these two approaches in a single architecture, in order to improve the classical machine learning models that only work in bottom-up mode. This can be achieved through attention.

Another interesting idea, developed in [3, 5, 7], is the principle of modularity. In the same way that a programmer codes several small independent functions and then makes bigger ones, it could be advantageous to decompose a neural network into several pieces to gain efficiency. These pieces can communicate with each other.

In this study, we use attention and the modularity principle to create a new general neural network architecture based on classical feedforward neural networks. This architecture is applicable to any supervised learning task whose input is fixed (a time series is therefore not suitable) since the loss function will be arbitrary. What makes this new architecture interesting is that it is a generalization of ResNet [4] which are known to be efficient, combined with the promising principle of modularity. Among others, we aim to answer the following questions:

- For a fixed number of parameters in a neural network, is it better to make a classical neural network, or is it better to build an architecture that will be encouraged to separate the final task into several sub-tasks?
- In a ResNet type architecture, is it interesting to add residual connections coming from all the previous layers, whose contribution weights will be determined by attention?

Our approach is somewhat the opposite of the dropout approach. In a classical neural network with dropout, the goal is to be very robust for generalization, by allowing the network to be efficient even if a percentage of neurons are not active. However, this is contrary to the principle of modularity and makes the neural network even less interpretable. We believe that the same performance could

*charbel-raphael.segerie@hotmail.fr
†hugo.laurencon@gmail.com

be obtained with a more interpretable (but not less complicated) model with fewer parameters.

The problem of modularity is potentially very important for the resilience and interpretability of neural networks. According to [3, 5, 7], modularity would allow:

- improved networks robustness in the face of a potential change in the distribution of training and validation data;
- improved performance in complex environments;
- better interpretability of the different modules.

If these promises are fulfilled, it is potentially a step forward for the feasibility of transfer learning in real production environments that are often more complex than development environments. Transfer learning being one of the keys to the democratization of machine learning, it would make artificial intelligence more accessible, but also more robust, especially in environments where training data is scarce.

2. Problem Definition

In order to test the modularity via the proxy of the ability to generalize, we define a set $\mathcal{T}$ containing transformed MNIST [6] images (typically those obtained by data augmentation techniques). Our goal is to find the architecture within the model space $\mathcal{M}$ that, when trained on MNIST only, performs best on $\mathcal{T}$ images.

The classifier space $\mathcal{M}$ that we will study is the set of architectures following the figure 1, by varying the number of bricks, the number of layers, and the presence of transverse and top-down attentions. We will search among this set of classifiers to find the most suitable one to generalize.

3. Related Work

The architecture we propose contains ideas borrowed from [3, 5, 7].

Recurrent Independent Mechanism: In [3], the authors construct what they call a RIM (Recurrent Independent Mechanism), which is an architecture based on several LSTMs running in parallel, and whose communications between the different LSTMs are determined by attention mechanisms. A maximum number of communications is imposed to support the modularity of the neural network. Our architecture is different because it is no longer LSTMs running in parallel but classical feedforward neural networks. Thus, the type of input is not the same. The incorporation of top-down signals is also omitted in this study.

Bidirectional Recurrent Independent Mechanism: In [7], the authors extend the work of [3] to propose an architecture working both bottom-up and top-down. This is done by creating several RIM layers and adding top-down connections, and not only bottom-up connections as in a classical LSTM. We implement in our model this top-down bottom-up idea differently, using after each layer the results of all the previously computed layers.

Neural Function Modules: In [5], the authors define NFMs (Neural Function Modules). It is an architecture in which after each layer, the results of all previous layers are retrieved, with weights determined by attention. It is the architecture that is the closest to ours. The biggest difference is that their algorithm contains several passes in which the network starts again from an initial state, while keeping the results of the layers of the previous passes. Unlike them, we do not use several passes, but we propose several bricks running in parallel to support the modularity in the whole network.

4. Methodology

4.1. General presentation of our architecture

First, we give a general overview of our architecture, represented on Fig. 1. Details are written in Sec. 4.3.

Let's imagine N_B "small" neural networks in parallel, i.e. shallower networks or networks with narrower layers to compensate for the fact that having several networks implies having more parameters. Each of these small networks, called a brick, will have N_L hidden layers.

Within a brick, we have a top-down and bottom-up operation: for each layer, after applying the activation function, we are going to add to the output a weighted average of all the results of the previous layers of the brick, whose weights are determined by attention.

Within a layer, communication between the bricks is also possible: for each layer, after applying the activation function, we are going to also add to the output a weighted average of all the results of the different bricks of the layer, whose weights are also determined by attention.

If there are differences in size in the weighted averages, up and downsampling techniques can be used in order to have a homogeneous size. To reinforce the principle of modularity, when communicating between the different bricks, one could impose a maximum number of bricks that receive information from other bricks, by taking only the k_A bricks whose attention scores are the highest with other outputs.

As the final loss function is arbitrary, this architecture can be adapted to a large number of supervised learning problems.

4.2. Interpretation and intuition behind the model

4.2.1 Connections with neuroscience

Several links can be made between neuroscience and our architecture in order to obtain an intuition on the model.

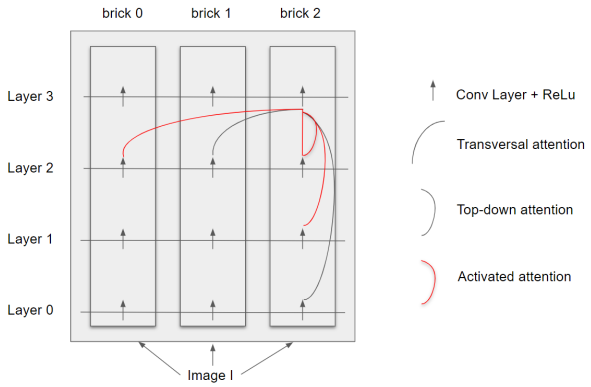

Figure 1: Scheme of our architecture.

- **Modularity:** The brain is not a homogeneous organ, it contains specialized brain areas. Our modular architecture is built on this point by creating different bricks that are ideally specialized in different sub-tasks.

- **Top-down, bottom-up:** Our architecture is inspired by the cerebral mechanisms of the predictive processing theory [2]. Indeed the top-down attention in our architecture is similar to the top-down attention in the predictive processing theory. And our feed forward networks, which are represented in the schematic by a bottom-up arrow, are similar to the bottom-up perceptual inputs in predictive processing theory.

- **Cycles:** The use of top-down and bottom-up information transmission mechanisms runs the risk of introducing cycles into the graph. The brain naturally contains cycles, but we have taken care to avoid them in the calculation of our graph. Indeed, our architecture works with the principle of strong recurrence: in order to move to the $n+1$ layer, we must have computed and stored the results of all previous layers from 1 to n in a "cache" memory. We can notice that, contrary to our architecture, the brain can be interpreted as a cycle that iterates continuously until it reaches a fixed point. In our architecture, we just take the first iteration of this cycle.

4.2.2 Connections with Unet and ResNet

The architecture we propose looks like half a Unet [9] and takes ideas from the ResNet [4].

- **Half-Unet:** The Unet's U-shaped architecture is composed of an encoder (left side of the U) and a decoder (right side of the U). The decoder and the encoder are connected by skip connections. The central part of the U that connects the encoder and the decoder contains the semantic representation. Our architecture is analogous to the encoder of the Unet. Our top-down attention is analogous to the skip connections of the Unet, by connecting the central junction with the left part of the U instead of connecting the left part of the U with the right part.

- **ResNet:** The top-down attention in our model plays the role of the skip connections in a ResNet. These skip connections are modified by the attention matrices, unlike the ResNet which concatenates directly.

4.3. Details of the methods

4.3.1 Attention in our model

The Key-Value Attention, widely used in self-attention models [10], is used in our architecture to determine the scores for the weighted averages we make at each layer of our architecture. Given a set of queries Q, keys K and values V, the attention is defined as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \qquad (1)$$

In practice and in our case, for a matrix X whose rows are the $(h_i)_{1\leq i\leq n}$ and for a fixed $1 \leq j \leq n$, to compute the attention score between h_j and h_i for all $i \in [|1, n|]$, we consider a weight matrix of queries W^Q and a weight matrix of keys W^K, and we compute

$$\text{Att}(X, h_j) = \text{Softmax}\left(\frac{h_j W^Q (W^K)^T X^T}{\sqrt{d}}\right)X \qquad (2)$$

where d is the dimension of each key.

In our case, X is the matrix whose rows are either the outputs of the previous layers for the same brick for the top-down attention, or the outputs of the different bricks for the same layer for the transverse attention.

In our implementation, we chose to keep the V value matrix equal to X. This allows us to get the right dimensions so that the attention output is the same size as the input vector before attention. If we had used multi-head attentions, we could have used p different W^V matrices with a width of k so that by concatenating all the outputs of the multi-head attentions we would get an output of the same size as the input vector before attention.

4.3.2 Parallelization of the attention

An analogy can be made between our transverse attention mechanism and the attention mechanisms in the structure of the transformers [10]. Indeed, we can interpret each of the outputs of the modules of each brick of our architecture as the representation of a token in NLP.

One difference is that the transformer architecture uses the same weight matrix for all the tokens while our architecture uses different weight matrices per brick. The consequence is that our query matrix Q is not a matrix but a simple vector, because we have one query matrix per brick and per layer while transformers have a shared query matrix for each layer.

This parallelization has been key in the success of the transformers. In this study, we chose to use single query matrices per layer and per brick in the hope that this would maximize the expressibility of our neural networks. We could also use the same transverse attention for the different bricks in the same way as in a transformer. This would further homogenize the representation between the different bricks, and thus create a common language, in the same way as the Global Workplace Theory [1], where the use of a single language between the bricks allows to communicate more efficiently.

4.3.3 Mathematical formulation of the model

Initialization: We give the same input X_0 to the first module of each brick b:

$$\forall b, X_{0,b} = X_0. \tag{3}$$

Propagation: For a layer l, for all bricks b, we first pass $X_{l,b}$ through its corresponding module denoted $\text{NN}_{l,b}$:

$$\forall b, M_{l,b} = \text{NN}_{l,b}(X_{l,b}). \tag{4}$$

Tranverse Attention: We compute the transverse attention:

$$\forall b, \text{TrAtt}_{l,b} = \text{Att}(M_l, M_{l,b}) \tag{5}$$

with

$$M_l = (M_{l,0}^T, \ldots, M_{l,N_b}^T)^T. \tag{6}$$

Top-down Attention: We compute the top-down attention:

$$\forall b, \text{TDAtt}_{l,b} = \text{Att}(M_b, M_{l,b}) \tag{7}$$

with

$$M_b = (X_{0,b}^T, \ldots, X_{l-1,b}^T, M_{l,b}^T)^T. \tag{8}$$

Communication: We then allow communications by adding attention results:

$$\forall b, \text{X}_{l+1,b} = \text{TrAtt}_{l,b} + \text{TDAtt}_{l,b}. \tag{9}$$

Concatenation: After the final layer, we concatenate results over the bricks:

$$X_{\text{concat}} = \text{Concat}((X_{N_L,b})_{0 \leq b < N_B}). \tag{10}$$

Propagation: We then propagate the concatenated result through fully connected layers:

$$X_{\text{final}} = \text{NN}_{\text{final}}(X_{\text{concat}}). \tag{11}$$

Here, $(M_{l,b})_{1 \leq b < N_B}$ that we compute for the transverse and the top-down attentions need to be stored only during layer l, while $(X_{l,b})_{1 \leq b < N_B}$ need to be stored until the end of the algorithm.

5. Evaluation

The main objective of our experiments is to verify if the structure of our architecture allows generalization in changing environments. The modularity of our architecture must be shown in cases where we change an element of our environment without completely changing the environment. In order to test our architecture, we trained it on the MNIST dataset [6] (which is large enough to be quite complex, but not too large so we do not have to train a model with too many parameters), without using data augmentation techniques. We obtain more intuition on our model by modifying the hyperparameters, mainly the number of bricks and layers and the types of attention used (top-down and/or transverse). For a number of bricks equal to 1, we come back to a model very similar to ResNet [4]. It is interesting to see how performance evolves when we increase the number of bricks while keeping a fixed number of parameters. We then tested the trained model on different types of transformations. We organize our results by type of transformation, by difficulty of generalization, by type of architecture, by type of attention mechanism used, as well as by size of hidden_size which is the size of the representation used by the attention (d in Sec. 4.3.1).

5.1. Transformations

We present here the different transformations used to test modularity. Examples are shown on Fig. 2.

- Linear scale: Multiplying by a constant the pixel values. The constant for *halfScalar*, *twiceScalar*, *scalarTenth* and *scalar10* is respectively 0.5, 2, 0.1 and 10.
- Random rotations: For an angle a, rotate the image by a random angle between $-a$ and a. The angle a for *rot30* and *rot45* is respectively 30 and 45.
- *resizedCrop*: A crop of random size (default: of 0.08 to 1.0) of the original size and a random aspect ratio (default: of 3/4 to 4/3) of the original aspect ratio is made. This crop is finally resized.
- *gaussianBlur*: Introducting white noise in the image with a kernel size of 5, $\sigma = (0.1, 2.0)$.

- *normalize*: Normalize a tensor image with a mean of 1 and a standard deviation of 1. Initially, MNIST has a mean of 0.1 and a standard deviation of 0.3.

- *randomErasing*: Randomly selects a rectangle region in an image and erases its pixels. Function RandomErasing with default parameters of Pytorch ($p = 0.5, scale = (0.02, 0.33), ratio = (0.3, 3.3)$).

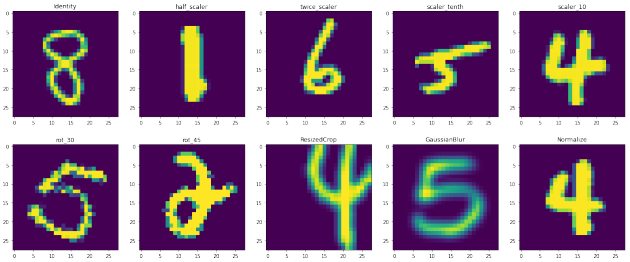

Figure 2: Examples of visualization of our transformations. From top to bottom, left to right: identity, *halfScalar*, *twiceScalar*, *scalarTenth*, *scalar10*, *rot30*, *rot45*, *resizedCrop*, *gaussianBlur*, *normalize*.

5.2. Choice of architectures for our tests

In order to test the influence of height, width and different types of transverse and top-down attentions, we created 4 architecture per (width, height), to test for each with/without transverse/top-down attention. We also fixed the number of bricks and layers in order to study the influence of hidden_size. Specifications of the different tested architectures are shown on Tab. 1 and their full performances are shown on Tab. 2 and Tab. 3.

5.3. Effect of transverse and top-down attentions

In order to be able to analyze the effect of transverse and top-down attentions, one can look at the accuracies of each architecture on the MNIST dataset (non-transformed yet) and then make an average. The results can be found in Tab. 4. We can see that using top-down attention slightly reduces performance while using transverse attention clearly reduces performance. But testing the respective effect - of transverse attention on networks with few bricks - or of top-down attention on networks with few layers - is not the most appropriate. We can therefore refine the analysis and study the effects of transverse attention on wide networks, i.e having 12 bricks but only 2 layers (results shown in Tab. 5), as well as the effect of top-down attention on high networks, i.e containing 12 layers but only one brick (results shown in Tab. 6). The results here are contrary to our expectations.

model name	bricks	layers	TR att	TD att	hidden_size
1b4l20hTTD	1	4	Yes	Yes	20
1b4l20hTtd	1	4	Yes	No	20
1b4l20htTD	1	4	No	Yes	20
1b4l20httd	1	4	No	No	20
3b4l20hTTD	3	4	Yes	Yes	20
3b4l20hTtd	3	4	Yes	No	20
3b4l20htTD	3	4	No	Yes	20
3b4l20httd	3	4	No	No	20
1b12l20hTTD	1	12	Yes	Yes	20
1b12l20hTtd	1	12	Yes	No	20
1b12l20htTD	1	12	No	Yes	20
1b12l20httd	1	12	No	No	20
2b6l20hTTD	2	6	Yes	Yes	20
2b6l20hTtd	2	6	Yes	No	20
2b6l20htTD	2	6	No	Yes	20
2b6l20httd	2	6	No	No	20
12b2l20hTTD	12	2	Yes	Yes	20
12b2l20hTtd	12	2	Yes	No	20
12b2l20htTD	12	2	No	Yes	20
12b2l20httd	12	2	No	No	20
3b4l8hTTD	3	4	Yes	Yes	8
3b4l8hTtd	3	4	Yes	No	8
3b4l8htTD	3	4	No	Yes	8
3b4l8httd	3	4	No	No	8
3b4l40hTTD	3	4	Yes	Yes	40
3b4l40hTtd	3	4	Yes	No	40
3b4l40htTD	3	4	No	Yes	40
3b4l40httd	3	4	No	No	40

Table 1: Characteristics of the different architectures. TR att: Transverse attention. TD att: Top-Down attention.

5.4. Ability to generalize

5.4.1 Effects of attention on generalization

As presented in Sec. 2, we can test the effect of attention mechanisms on generalization. In order to be able to compare the different networks with each other, we average the results of each network on the different transformation functions corresponding to the different tasks (summarized in Tab. 2 and Tab. 3). The results can be found in Tab. 7. Unfortunately, it seems that attention mechanisms do not seem to improve generalization.

We also wanted to test the efficiency of the attention on the 5 transformations that decreased the accuracy the most, namely: *scalarTenth*, *scalar10*, *rot45*, *resizedCrop*, and *normalize*. But the average results on these transformations available in Tab. 8 are similar to those obtained previously.

We were disappointed by these results, because we hoped that models with attention linking the different bricks and layers, although they take longer to train, are consequently better able to generalize. But it is in fact established that models using attention are difficult to train from scratch, so it is not impossible that an additional idea may be enough to make the proposed architecture work.

5.4.2 Effect of the number of bricks on generalization

One can also look for the most suitable architectures to generalize by looking at the influence of the number of bricks

Name	*Epoch*	*Identity*	*halfScalar*	*twiceScalar*	*scalerTenth*	*scaler10*	*rot30*	*rot45*	*resizedCrop*	*gaussianBlur*	*normalize*	*randomErasing*	*Generalization*	*Hard Generalization*
1b4l20hTTD	1	93,79	93,72	93,78	92,87	93,76	82,39	68,53	37,58	93,5	75,44	84,6	81,617	73,636
1b4l20hTtd	1	96,43	96,27	96,35	96,11	96,35	86,89	72,05	42,64	95,74	89,87	86,42	85,869	79,404
1b4l20htTD	1	91,95	91,97	92,05	91,11	92,25	82,31	66,18	29,32	91,96	69,95	83,31	79,041	69,762
1b4l20httd	1	96,35	96,44	96,38	96,06	96,34	86,03	75,75	45,87	95,48	87,59	85,98	86,192	80,322
3b4l20hTTD	1	86,03	86,31	86,48	85,11	86,37	73,7	65,11	28,62	85,83	62,52	75,17	73,522	65,546
3b4l20hTtd	1	63,06	63,01	63,22	61,25	62,85	54,06	48,67	26,24	63,25	29,14	53,45	52,514	45,63
3b4l20htTD	1	93,11	93,46	93,04	92,76	93,39	82,87	62,91	35,45	92,8	76,05	84,4	80,713	72,112
3b4l20httd	1	96,4	96,45	96,5	96,02	96,6	88,92	74,44	31,27	96,28	91,16	88,01	85,565	77,898
1b12l20hTTD	1	56,82	57,04	56,43	55,78	55,98	49,68	41,73	23,55	56,17	38,47	46,68	48,151	43,102
1b12l20hTtd	1	93,84	93,34	93,84	93,3	93,74	83,96	70,09	36,26	92,93	86,75	82,54	82,675	76,028
1b12l20htTD	1	34,82	35,68	35,62	34,68	36,35	31,63	27,89	18,88	34,97	22,51	31,2	30,941	28,062
1b12l20httd	1	93,53	93,69	93,6	93,13	93,82	84,95	73,05	33,96	92,71	86,59	82,92	82,842	76,11
2b6l20hTTD	1	72,05	71,97	72,16	71,54	71,59	62,61	54,5	26,31	72,91	47,72	62,6	61,391	54,332
2b6l20hTtd	1	36,28	37,11	36,77	35,17	37,29	31,25	28,2	19,88	36,49	18,7	31,84	31,27	27,848
2b6l20htTD	1	82,94	83,07	83,42	82,44	83,71	71,86	57,6	22,27	84,15	45,18	72,97	68,667	58,24
2b6l20httd	1	95,8	95,57	95,72	95,54	95,51	86,55	72,84	38,73	95,25	92,42	86,31	85,444	79,008
12b2l20hTTD	1	95,2	95,19	95,17	94,95	95,41	87,03	70,24	30,17	94,76	83,94	86,03	83,289	74,942
12b2l20hTtd	1	93,54	93,71	93,7	93,11	93,73	84,11	73,08	30,09	93,2	79,3	84,48	81,851	73,862
12b2l20htTD	1	97,12	97	97,05	96,8	97,1	91,32	77,84	35,61	96,87	89,67	87,67	86,693	79,404
12b2l20httd	1	97,31	97,21	97,26	96,97	97,16	90,91	76,7	41,64	96,83	89,27	87,59	87,154	80,348
3b4l8hTTD	1	84,77	84,33	84,46	84,04	84,86	70,97	64,49	34,14	84,52	55,92	73,89	72,162	64,69
3b4l8hTtd	1	64,36	64,59	64,8	62,81	64,42	54,36	48,5	22,09	64,53	36,18	55,3	53,758	46,8
3b4l8htTD	1	92,61	92,79	92,55	91,82	92,53	82,9	64,98	27,72	93,03	73,97	83,56	79,585	70,204
3b4l8httd	1	96,78	96,79	96,96	96,51	96,84	86,7	75,01	38,16	96,33	91,23	87,86	86,239	79,55
3b4l40hTTD	1	86,95	86,92	87,24	86,58	87,4	75,73	64,99	32,41	87,29	60,32	78,35	74,723	66,34
3b4l40hTtd	1	67,3	67,21	67,42	66,61	67,57	56,25	44,51	25,58	68,52	33,39	56,35	55,341	47,532
3b4l40htTD	1	92,36	92,29	92,43	91,8	92,55	81,7	67,4	30,19	92,55	67,31	83,12	79,134	69,85
3b4l40httd	1	96,35	96,27	96,54	95,9	96,47	87,18	75,79	40,7	96,08	91,59	85,01	86,153	80,09
1b4l20hTTD	2	95,28	95,35	95,15	94,9	95,2	83,38	73,45	38,35	94,1	80,87	86,67	83,742	76,554
1b4l20hTtd	2	97,03	96,8	96,94	96,73	96,77	88,41	75,16	35,49	96,09	90,36	88,4	86,115	78,902
1b4l20htTD	2	94,42	94,59	94,2	93,52	94,29	83,5	71,86	30,37	93,91	77,2	85,18	81,862	73,448
1b4l20httd	2	97,05	97,08	97,14	96,85	97,16	90,41	77,61	35,67	96,96	87,62	86,95	86,345	78,982
3b4l20hTTD	2	89	89,12	89,22	88,92	89,27	77,35	67,02	24,89	88,66	62,63	79,13	75,621	66,546
3b4l20hTtd	2	74,45	74,88	74,24	74,29	75,19	62,05	55,06	27,03	74,84	35,75	65,02	61,835	53,464
3b4l20htTD	2	94,95	95,26	95,2	94,89	95,02	83,66	74,39	34,88	94,52	78,23	84,21	83,026	75,482
3b4l20httd	2	97,01	97,05	97,01	96,89	97,1	87,55	76,29	34,89	96,75	91,7	86,36	86,159	79,374
1b12l20hTTD	2	62,6	62,42	61,74	61,24	60,83	52,12	46,03	23,63	62,48	43,01	53,45	52,695	46,948
1b12l20hTtd	2	95,17	95,17	95,3	94,86	95,17	88,17	72,99	36,16	94,32	90,37	85,95	84,846	77,91
1b12l20htTD	2	53,16	53,42	53,53	52,34	53,38	46,4	41,15	20,3	53,07	34,17	44,92	45,268	40,268
1b12l20httd	2	95,1	95,14	95,11	95	95	85,94	74,74	35,61	94,54	89,2	86,13	84,641	77,91
2b6l20hTTD	2	82,26	81,89	82,19	81,83	82,14	72,8	59,81	32,08	82,49	59,1	70,28	70,461	62,992
2b6l20hTtd	2	69,64	68,8	68,86	68,26	69,44	59,9	48,14	26,71	68,57	36,8	62,3	57,778	49,87
2b6l20htTD	2	90,37	90,24	90,31	90,05	90,29	74,59	63,36	30,24	90,31	55,29	81	75,568	65,846
2b6l20httd	2	96,35	96,7	96,69	96,63	96,67	87,14	75,83	34,79	95,93	92,64	87,32	86,034	79,312
12b2l20hTTD	2	95,98	96,05	95,92	95,9	96,08	84,98	75,88	33,26	95,57	84,67	88,14	84,645	77,158
12b2l20hTtd	2	94,1	93,9	93,96	94,03	94,33	80,05	67,8	35,98	93,64	78,42	83,15	81,526	74,112
12b2l20htTD	2	97,62	97,57	97,65	97,42	97,51	90,95	81,83	36,04	97,03	87,81	84,11	86,792	80,122
12b2l20httd	2	97,61	97,59	97,61	97,21	97,47	90,78	79,71	38,87	97,2	87,7	83,86	86,8	80,192
3b4l8hTTD	2	86,33	86,21	86,39	85,86	86,39	76,2	62,67	27,19	86,2	57,84	76,96	73,191	63,99
3b4l8hTtd	2	69,81	69,88	68,79	69,2	69,53	59,87	51,52	24,66	70,08	35,24	58,35	57,712	50,03
3b4l8htTD	2	95,34	95,21	95,14	94,52	95,36	86,04	71,52	38,09	94,91	76,01	82,58	82,938	75,1
3b4l8httd	2	96,6	96,74	96,51	96,45	96,73	89,97	73,56	39,4	96,13	90,91	85,86	86,226	79,41
3b4l40hTTD	2	90,44	90,73	90,73	90,13	90,47	79,44	68,87	33,74	90,14	65,3	80,5	78,005	69,702
3b4l40hTtd	2	76,88	77,43	76,91	76,5	76,99	66,15	54,53	28,61	76,98	42,09	65	64,119	55,744
3b4l40htTD	2	96,28	96,32	96,22	95,93	96,26	85,25	76,9	37,14	95,87	68,54	85,88	83,431	74,954
3b4l40httd	2	96,93	97,03	97,05	96,59	96,83	88,19	79,49	40,47	96,35	92,44	84,45	86,889	81,164
1b4l20hTTD	5	95,8	96,07	96,02	95,09	96	85,22	72,96	34,62	95,18	81,39	83,95	83,65	76,012
1b4l20hTtd	5	97,39	97,3	97,5	97,25	97,37	89,26	78,43	44,49	96,75	90,28	84,45	87,308	81,564
1b4l20htTD	5	95,78	95,98	95,73	95,4	95,88	86,6	74,19	30,84	95,16	80,03	82,78	83,259	75,268
1b4l20httd	5	97,46	97,66	97,72	97,42	97,59	88,47	79,77	40,82	97,26	87,11	85,46	86,928	80,542
3b4l20hTTD	5	92,55	92,33	92,24	91,6	92,3	79,83	70,56	36,13	92,11	63,6	81,09	79,179	70,838
3b4l20hTtd	5	85,86	85,92	85,98	85,38	85,9	72,9	64,02	27,98	85,63	56,43	72,39	72,253	63,942
3b4l20htTD	5	96,93	96,79	96,84	96,39	96,81	87,21	77,73	42,07	95,91	72,97	85,63	84,835	77,194
3b4l20httd	5	97,65	97,69	97,62	97,63	97,66	90,75	84,06	38,31	97,41	92,01	82,94	87,608	81,934
1b12l20hTTD	5	74,58	74,7	74,63	74,51	74,65	62,64	52,27	26,08	74,02	50,22	64,11	62,783	55,546
1b12l20hTtd	5	96,96	96,91	96,84	96,66	96,81	88,42	75,12	39,83	96,05	93,32	84,69	86,465	80,348
1b12l20htTD	5	59,24	59,2	58,83	58,32	59,62	51,17	41,98	19,35	59,29	30,62	50,1	48,848	41,978
1b12l20httd	5	96,6	96,75	96,55	96,56	96,47	88,99	82,77	37,88	96,04	91,44	86,1	86,955	81,024
2b6l20hTTD	5	89,89	90,21	90,01	89,38	90,22	78,95	69,88	31,97	89,64	62,16	78,77	77,119	68,722
2b6l20hTtd	5	80,7	80,88	80,4	81,15	80,74	69,65	56,86	31,8	80,43	50,39	71,7	68,4	60,188
2b6l20htTD	5	95,67	95,55	95,65	95,05	95,5	84,11	75,45	31,13	95,36	54,28	84,51	80,659	70,282
2b6l20httd	5	97,64	97,49	97,85	97,6	97,79	88,78	79,94	45,17	96,99	93,73	86,8	88,214	82,846
12b2l20hTTD	5	96,91	96,82	96,86	96,76	97,01	89,17	78,35	38,54	96,51	83,82	87,48	86,132	78,896
12b2l20hTtd	5	95,68	95,67	95,74	95,33	95,6	84,58	73,7	38,5	95,1	80,54	84,21	83,897	76,734
12b2l20htTD	5	97,91	97,86	97,84	97,75	97,91	90,35	80,36	42,34	97,76	87,13	88,32	87,762	81,098
12b2l20httd	5	98,03	97,85	97,99	97,83	98,03	90,37	81,94	42,57	97,5	86,59	85,9	87,657	81,392
3b4l8hTTD	5	92,68	92,83	92,84	92,37	92,56	81,3	67,12	34,36	92,17	61,94	83,54	79,103	69,67
3b4l8hTtd	5	84,89	85,01	85,02	84,9	85,18	73,39	63,43	26,71	84,56	46,52	73,23	70,795	61,348
3b4l8htTD	5	96,26	96,26	96,39	95,96	96,61	86,88	76,24	37,98	96,14	76,95	83,61	84,302	76,748
3b4l8httd	5	97,89	97,68	97,55	97,4	97,75	88,54	74,44	46,8	97,19	92,27	85,65	87,527	81,732
3b4l40hTTD	5	93,72	93,69	93,84	93,18	93,56	83,3	72,81	34,57	92,93	65,02	83,66	80,656	71,828
3b4l40hTtd	5	88,67	88,78	88,85	88,52	88,59	76	64,7	30,19	88,7	49,52	77,12	74,097	64,304
3b4l40htTD	5	97,21	97,27	97,26	96,94	97,25	88,81	78,19	31,13	96,44	69,41	86,02	83,872	74,584
3b4l40httd	5	97,55	97,49	97,49	97,22	97,48	89,63	76,5	42,02	96,77	92,07	83,36	87,003	81,058
1b4l20hTTD	20	97,66	97,66	97,56	97,27	97,71	88,31	77,95	37,73	97,06	78,42	82,46	85,213	77,816
1b4l20hTtd	20	98,14	98,16	98,07	97,9	98,29	90,99	83,14	44,65	97,56	91,08	85,82	88,566	83,012
1b4l20htTD	20	97,5	97,71	97,55	97,19	97,56	89,3	78,05	46,19	96,85	68,88	82,38	85,166	77,574
1b4l20httd	20	97,74	97,88	97,84	97,54	97,79	89,56	75,22	39,98	97,35	87,91	81,04	86,211	79,688
3b4l20hTTD	20	96,1	96,23	96,2	96,13	96,37	86,74	72,55	32,62	95,6	71,51	82,88	82,683	73,836
3b4l20hTtd	20	93,93	93,46	93,77	93,65	93,82	82,13	69,2	36,96	93,59	66,11	81,37	80,406	71,948
3b4l20htTD	20	97,75	97,6	97,73	96,82	97,64	88,22	75,48	37,7	97,2	77,17	82,1	84,766	76,962
3b4l20httd	20	97,81	97,91	97,83	97,87	97,91	90,71	77,09	45,35	97,38	90,1	85,36	87,751	81,664
1b12l20hTTD	20	83,82	83,82	83,33	83,13	83,81	70,36	58,78	29,24	83,91	44,64	71,29	69,231	59,92
1b12l20hTtd	20	97,57	97,61	97,66	97,34	97,62	92,44	82	41,03	96,88	92,22	85,36	88,016	82,042
1b12l20htTD	20	85,65	85,84	85,8	85,57	85,46	75,78	63,31	31,28	86,72	22,7	72,93	69,539	57,664
1b12l20httd	20	97,83	97,84	97,74	97,54	97,77	88,52	80,22	45,58	97,2	91,16	83,05	87,662	82,454
2b6l20hTTD	20	95,3	94,79	95,04	94,33	94,91	84,16	75,99	27,84	94,66	70,78	84,31	81,681	72,77

Table 2: All the results obtained (1/2).

Name	*Epoch*	*Identity*	*halfScalar*	*twiceScalar*	*scalerTenth*	*scaler10*	*rot30*	*rot45*	*resizedCrop*	*gaussianBlur*	*normalize*	*randomErasing*	*Generalization*	*Hard Generalization*
2b6l20hTtd	20	93,51	93,66	93,65	93,28	93,46	81,49	72,63	37,24	93,29	76,37	79,9	81,497	74,596
2b6l20htTD	20	97,05	96,94	96,8	96,26	97,05	88,1	75,97	39,55	96,59	56,7	81,99	82,595	73,106
2b6l20httd	20	97,95	97,84	97,97	97,54	97,96	89,25	76,9	44,79	97,57	92,26	84,3	87,638	81,89
12b2l20hTTD	20	97,43	97,7	97,64	97,4	97,73	89,2	76,4	43,25	97,06	79,24	84,84	86,046	78,804
12b2l20hTtd	20	96,88	96,83	96,78	96,62	96,89	87,58	74,83	39,59	96,45	81,06	84,85	85,148	77,798
12b2l20htTD	20	98,2	98,22	98,23	98,08	98,13	92,13	79,93	42,11	97,92	83,39	84,26	87,24	80,328
12b2l20httd	20	98,13	97,98	97,82	97,62	98,03	89,04	80,99	41,13	97,5	88,24	86,56	87,491	81,202
3b4l8hTTD	20	96,46	96,48	96,47	96,27	96,5	85,74	74,44	38,44	95,87	60,56	82,64	82,341	73,242
3b4l8hTtd	20	93,66	93,79	93,55	93,94	93,82	82,84	71,01	32,06	93,39	65,25	82,7	80,235	71,216
3b4l8htTD	20	97,84	97,73	97,83	97,33	97,72	86,89	79,51	42,88	97,29	81,43	81,59	86,02	79,774
3b4l8httd	20	98,02	97,97	98,11	97,71	98,1	91,65	81,09	34,21	97,54	91,18	84,31	87,187	80,458
3b4l40hTTD	20	96,51	96,31	96,49	96,14	96,33	87,03	68,03	38,57	96,01	70,16	83,27	82,834	73,846
3b4l40hTtd	20	95,29	95,39	95,56	95,17	95,27	84,8	73,03	34,51	94,89	57,88	85,34	81,184	71,172
3b4l40htTD	20	97,78	97,91	97,87	97,41	97,79	90,17	79,95	35,21	97,13	73,69	81,9	84,903	76,81
3b4l40httd	20	98	98,19	98,02	97,73	98	88,22	74,92	43,11	97,69	91,65	87,31	87,484	81,082

Table 3: All the results obtained (2/2).

Mean acc	W/o TD att	W TD att
W/o TR att	97.95	97.75
W/ TR att	95.29	96.46

Table 4: Average Accuracy of Transverse and Top-Down Attention. acc: accuracy. W/o: Without. W/: With. TD att: Top-Down attention. TR att: Transverse attention.

Mean acc	W/o TD att	W TD att
W/o TR att	97.82	97.765
W/ TR att	94.89	96.445

Table 5: Average accuracy of transverse attention on broad networks. acc: accuracy. W/o: Without. W/: With. TD att: Top-Down attention. TR att: Transverse attention.

Mean acc	W/o TD att	W TD att
W/o TR att	95.77	95.89
W/ TR att	58.21	69.46

Table 6: Average accuracy of top-down attention on high networks. acc: accuracy. W/o: Without. W/: With. TD att: Top-Down attention. TR att: Transverse attention.

Mean acc	W/o TD att	W TD att
W/o TR att	87.35	83.58
W TR att	82.89	81.43

Table 7: Average accuracy of transverse and top-down attention evaluated on the validation set. acc: accuracy. W/o: Without. W/: With. TD att: Top-Down attention. TR att: Transverse attention.

and layers, with a number of fixed parameters, by setting the number of modules to 12. Results are shown in Tab. 9. Interestingly, the generalization capacity of networks does not seem to increase with the number of layers, we can even notice that for difficult generalization tasks, the performance seems to increase slightly with the number of bricks. Even if this slight increase is subject to statistical fluctuations, it seems that for many equal parameters, networks with bricks

Mean acc	W/o TD att	W TD att
W/o TR att	81.21	75.97
W TR att	74.60	72.90

Table 8: Average accuracy evaluated on the validation set obtained by the 5 most severe transformations. acc: accuracy. W/o: Without. W/: With. TD att: Top-Down attention. TR att: Transverse attention.

seem to have the same generalization capabilities as deep networks. Our interpretation is that with this type of network defined with bricks without attention mechanism, the semantic representation is certainly less deep, but the different bricks are then no longer correlated, which allows us to obtain a network implementing an ensemble method, such as random forests, with classifiers that are each weak, but that work quite well together.

nb bricks / layers	generalization tasks	hard generalization tasks
4-bricks-3-layers	87.47	81.07
6-bricks-2-layers	87.64	81.89
12-bricks-1-layers	87.66	82.45

Table 9: Average accuracy for generalization tasks for no-attention networks with a fixed number of modules (12 modules).

6. Conclusions

In this study, we tested a new architecture whose goal was to encourage modularity to make neural networks more interpretable, and to add top-down connections in addition to bottom-up connections to obtain a more general architecture close to that of the brain.

We tested our architecture for different choices of hyperparameters on the MNIST database, and then on transformed images from this database to test the ability to generalize of our architecture. The results are a bit disappointing for the transverse attention, but are not all the time very inferior compared to a classical neural network with the same number of parameters (for example, the top-down attention gives roughly the same performance). These results lead us

to believe that it is more difficult to train our models because even if they are theoretically more general and expressive, it should be easier to find their way into a bad local minimum. Therefore, it could be better to use simpler networks for easy tasks like MNIST but maybe the top-down attention could actually improve performances on harder tasks.

Since this study was carried out for the validation of a course, and not being a full-time research, it was not possible for us to make an exhaustive study of everything we wanted to test. There are a lot of possible improvements, which we give here in opening, which could be able to significantly improve performance.

First of all, an architecture with CNNs instead of simple feedforward neural networks would allow better capture of image information and would make the architecture applicable to more problems.

Moreover, for simplicity in attention calculations, we limited ourselves to layer output vectors having the same dimensions whatever the layer and brick considered. This can potentially create a bottleneck, which we can solve by resizing the vectors before the attention calculation by doing up-sampling or down-sampling as in [5].

We can also use multi-head attention instead of the single-head we used here. This would enhance the effect of the attention and allow for greater expressiveness.

In order to encourage modularity, it is possible to consider a maximum number n_A of bricks receiving the result of the other bricks per attention at each layer. This would reduce the application of the attention mechanism to the n_A bricks that can benefit the most from it.

Finally, it would remain to apply all these improvements on a larger dataset, such as ImageNet, with the classical techniques to improve performance (dropout, batch normalization, etc...) to have a better idea of the performance of this type of neural network.

References

[1] Yoshua Bengio et al. Synchronizing neural modules through a shared workspace. *Currently unpublished*. 4

[2] Andy Clark. Surfing uncertainty: Prediction, action, and the embodied mind. 2017. 3

[3] Anirudh Goyal et al. Recurrent independent mechanisms. *arXiv 1909.10893*, 2020. 1, 2

[4] Kaiming He et al. Deep residual learning for image recognition. *CoRR*, 2015. 1, 3, 4

[5] Alex Lamb et al. Neural function modules with sparse arguments: A dynamic approach to integrating information across layers. *arXiv 2010.08012*, 2020. 1, 2, 8

[6] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. 2, 4

[7] Sarthak Mittal et al. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1, 2

[8] Adam Paszke et al. Automatic differentiation in pytorch. *31st Conference on Neural Information Processing Systems*, 2017.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 3

[10] Ashish Vaswani et al. Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017. 1, 3