



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE  
ESCUELA DE INGENIERÍA

# **DATA AUGMENTATION HELPS TO PREVENT SHORTCUTS AND LEARN REPRESENTATIONS FOR CONTINUAL LEARNING IN NEURAL NETWORKS**

**SEBASTIÁN AMENÁBAR MONTENEGRO**

Thesis submitted to the Office of Research and Graduate Studies  
in partial fulfillment of the requirements for the degree of  
Master of Science in Engineering

Advisor:  
HANS LÖBEL

Santiago de Chile, Noviembre 2021

© MMXXI, SEBASTIÁN AMENÁBAR



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE  
ESCUELA DE INGENIERÍA

# DATA AUGMENTATION HELPS TO PREVENT SHORTCUTS AND LEARN REPRESENTATIONS FOR CONTINUAL LEARNING IN NEURAL NETWORKS

SEBASTIÁN AMENÁBAR MONTENEGRO

Members of the Committee:

HANS LÖBEL

DocuSigned by:

Hans Löbel

9024605F8E47CA7...

ÁLVARO SOTO

DocuSigned by:

Álvaro Soto

DA096D318D8B4F2...

BILLY PERALTA

DocuSigned by:

Billy Peralta

117B2C841D674F5...

DANIEL HURTADO

DocuSigned by:

Daniel Hurtado

EF6C1DAF5DC04B2...

Thesis submitted to the Office of Research and Graduate Studies  
in partial fulfillment of the requirements for the degree of  
Master of Science in Engineering

Santiago de Chile, Noviembre 2021

© MMXXI, SEBASTIÁN AMENÁBAR

*Gratefully to my parents and  
siblings*

## **ACKNOWLEDGEMENTS**

Thanks to my family, for providing me with everything I've ever needed and supporting me throughout all my life, I wouldn't be where I'm right now without them. Thanks to the friends who have been with me the last years, thanks to all the things we have shared, and helping me in beginning to find what I think may be important in this life. Thanks to the amazing group of people of the IALAB, for their friendliness, support, help, and all the thinks they have taught me and helped me understand, academically, but also about all the rest. Finally, thanks to my advisor Hans, for providing me with the opportunity of diving in into this topic, supporting my ideas and encouraging me to follow them, for his great guidance, but most importantly, for his incredible willingness to help me whenever I needed it.



## TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	ix
ABSTRACT	x
RESUMEN	xi
1. INTRODUCTION	1
2. RELATED WORK	4
2.1. Data augmentation in computer vision . . . . .	4
2.2. Learning inductive biases from the data . . . . .	4
3. VISUAL QUESTION ANSWERING	6
3.1. Related work . . . . .	6
3.1.1. Visual Question Answering . . . . .	6
3.1.2. Shortcut learning . . . . .	9
3.1.3. Convolutional neural networks are not translation equivariant . . . . .	10
3.2. Theoretical Framework . . . . .	11
3.2.1. Compositional Attention Networks . . . . .	11
3.3. Dataset . . . . .	14
3.4. Preliminary Experiments . . . . .	15
3.5. Modified CLEVR . . . . .	16
3.6. Experiments . . . . .	16
3.6.1. Evaluation . . . . .	17
3.6.2. Setup . . . . .	17
3.7. Results . . . . .	19
3.8. Analysis . . . . .	19

4. Catastrophic Forgetting	26
4.1. Related work	26
4.1.1. Continual Learning and the Catastrophic Forgetting Problem	26
4.1.2. Meta-learning, few-shot learning, and Model-Agnostic Meta-learning	27
4.2. Theoretical Framework	29
4.2.1. Traditional Supervised Learning	29
4.2.2. The Continual Learning Problem Formulation	29
4.2.3. Online-aware Meta-learning	30
4.2.4. A Neuromodulated Meta-learning	33
4.3. Experiments	35
4.3.1. Dataset	35
4.3.2. Evaluation	36
4.3.3. Setup	37
4.4. Results	39
4.5. Analysis	42
4.5.1. Learned representations	42
4.5.2. Sparsity and dead neurons	46
5. CONCLUSIONS	49
6. FUTURE WORK	50
REFERENCES	51
APPENDIX	62
A. Continual learning representations	63
A.1. Omniglot random samples used for visualization	63
A.2. <i>Single-branch</i> network representations	64
A.3. <i>Neuromodulated</i> network representations	65

## LIST OF FIGURES

3.1	Visual question answering example . . . . .	6
3.2	Object and grid level representations . . . . .	7
3.3	Shortcut learning examples . . . . .	9
3.4	Example of MAC execution . . . . .	13
3.5	CLEVR examples . . . . .	14
3.6	Convolutional features position regression . . . . .	15
3.7	Distribution of objects on the CLEVR and modified CLEVR datasets . . . . .	16
3.8	Examples of perspective transformation . . . . .	18
3.9	Size problems of the model trained without data augmentation . . . . .	20
3.10	Material and shape problems of the model trained without data augmentation . . . . .	21
3.11	Region-based accuracy on the modified CLEVR . . . . .	22
3.12	Example of a question that should not require to reason about an object in the bordering 60 pixels . . . . .	25
4.1	OML network . . . . .	31
4.2	ANML network . . . . .	33
4.3	Examples of Omniglot characters . . . . .	36
4.4	<i>Single-branch</i> network results . . . . .	39
4.5	<i>Neuromodulated</i> network results . . . . .	40
4.6	<i>Single-branch</i> network representations . . . . .	43

4.7	Meta-trained <i>neuromodulated</i> network representations . . . . .	44
4.8	TSL-trained <i>neuromodulated</i> network representations . . . . .	45
A.1	Omniglot random samples used for visualization . . . . .	63
A.2	<i>Single-branch</i> network representations second seed . . . . .	64
A.3	<i>Single-branch</i> network representations third seed . . . . .	64
A.4	Meta-trained <i>neuromodulated</i> network representations second seed . . . . .	65
A.5	Meta-trained <i>neuromodulated</i> network representations third seed . . . . .	66
A.6	TSL-trained <i>neuromodulated</i> network representations second seed . . . . .	67
A.7	TSL-trained <i>neuromodulated</i> network representations third seed . . . . .	67

LIST OF TABLES

3.1 Accuracy on the CLEVR datasets . . . . . 19

3.2 VQA results depending on the presence of objects in the top pixels . . . . . 24

4.1 Average sparsity and dead neurons comparison table . . . . . 47

## ABSTRACT

Inductive biases have had a fundamental role in the success of deep learning, but, in the recent time, models with strong inductive biases have been outperformed by data-centric approaches, that combine big and flexible architectures with a special focus on the data. Nevertheless, these data-centric approaches do not get rid of some of the problems deep learning has, the most relevant to this work being shortcut learning and catastrophic forgetting. Shortcut learning occurs when the network learns decision rules that work well under testing conditions similar to the training ones, but are not robust to shifts in the data distribution, for example, recognizing camels on a pasture, after only seeing camels in the desert. Catastrophic forgetting occurs when the network has to learn from a non-stationary stream of data, without losing or forgetting the already acquired knowledge, but, instead, it fails to do so, commonly forgets, and does not perform well on the data that came earlier in the stream, for example, learning to recognize new animals without forgetting the ones already known.

In this work, we show that data augmentation can be used to mitigate the mentioned problems. First, we observed the shortcut learning problem through the visual question answering task. We found that a flexible architecture learns shortcuts from the data, which causes it to fail when evaluated on samples from a modified distribution, but incorporating data augmentation prevents it from learning these biases and helps its performance on the data of the modified distribution. For the catastrophic forgetting problem, recent work showed that meta-learning can be used to learn a feature extractor less prone to forgetting. In this work, we show that a neural network trained through traditional supervised learning can also be used for this problem, and, we observed that data augmentation can have a big impact on the performance of the model for this problem.

**Keywords:** data augmentation, visual question answering, shortcut learning, catastrophic forgetting, continual learning

## RESUMEN

Los sesgos inductivos han sido fundamentales en el éxito del aprendizaje profundo, pero, recientemente, los modelos con sesgos inductivos fuertes han sido superados por propuestas centradas en los datos, que combinan arquitecturas grandes y flexibles, con una especial atención en los datos. No obstante, estas propuestas centradas en los datos mantienen algunos de los problemas que el aprendizaje profundo tiene, los más relevantes para este trabajo son el aprendizaje de atajos y el olvido catastrófico. El aprendizaje de atajos ocurre cuando la red aprende reglas de decisión que no son robustas a cambios en la distribución de los datos, por ejemplo, reconocer camellos en un pastizal, después de haber visto camellos únicamente en el desierto. El olvido catastrófico ocurre cuando la red tiene que aprender de un flujo no estacionario de datos, sin perder o olvidar el conocimiento ya adquirido, pero falla en lograr esto y tiene un mal desempeño en los datos que vió anteriormente, por ejemplo, aprender a reconocer animales sin olvidar los ya conocidos.

En este trabajo mostramos que la aumentación de datos puede ser utilizada para mitigar los problemas mencionados. Primero, observamos el aprendizaje de atajos en la tarea de respuesta a pregunta visual. Vimos que una arquitectura flexible aprende atajos por lo que falla al modificar la distribución de los datos, pero la incorporación de aumentación previene que el modelo aprenda estas reglas y ayuda a mejorar su desempeño en los datos de la distribución modificada. Para el problema del olvido catastrófico, trabajos recientes mostraron que el meta-aprendizaje puede ser utilizado para aprender un extractor de características menos susceptible a olvidar. En este trabajo, mostramos que una red neuronal entrenada mediante aprendizaje supervisado tradicional también puede ser utilizada para este problema, y observamos que la aumentación de datos puede tener un gran impacto en el desempeño del modelo.

**Palabras Claves:** aumentación de datos, respuesta a preguntas visuales, aprendizaje de atajos, olvido catastrófico, aprendizaje continuo

## 1. INTRODUCTION

Since the breakthrough of deep neural networks in image classification (Krizhevsky, Sutskever, & Hinton, 2012) these systems have become arguably the most powerful tool available to achieve feats previously unthought of, such as predicting protein folding (Senior et al., 2020), flying stratospheric air balloons (Bellemare et al., 2020) or finding known and unknown cosmological principles (Cranmer et al., 2020). Two of the most important factors which allowed deep learning to achieve these accomplishments are the availability of large datasets and the incorporation of inductive biases in neural networks. However, in the recent years, a trend of using general purpose architectures with less inductive biases, such as the transformer (Vaswani et al., 2017), combined with massive amounts of data and computation, has become the best performing solution across a number of problems (Radford, Narasimhan, Salimans, & Sutskever, 2018; Devlin, Chang, Lee, & Toutanova, 2019; Dosovitskiy et al., 2021; Baevski, Schneider, & Auli, 2019; L. H. Li, Yatskar, Yin, Hsieh, & Chang, 2019). Several works that investigated the mechanisms learned by the transformer and its fundamental component, the attention operation, have provided evidence that it is learning the underlying structures of the data directly from it, which is possible thanks to the lower degree of inductive biases of the model (Cordonnier, Loukas, & Jaggi, 2019; Manning, Clark, Hewitt, Khandelwal, & Levy, 2020; Wu et al., 2021). This can be argued to be one of the factors related to the better performance compared to models with hand-designed inductive biases, but also the reason why they are harder to train without large datasets.

As mentioned, thanks to inductive biases, and, more recently, their absence, deep learning has achieved remarkable performance across a large number of fields and tasks, nevertheless, this technique still suffers from several shortcomings that seem to be present for both approaches. One problem is their lack of out-of-domain generalization, in which a model is trained and performs correctly on one or more source domains (the distributions from which the data is obtained), but fails when utilized in a different target domain (Ben-David et al., 2010). An example could be to train a model that has to differentiate



between camels and cows. Since camels are mostly found in deserts and cows in pastures, we can expect that a dataset built for this task will retain this same trend, and, if we train a deep learning model with this data, it will probably fail when we ask it about a camel on a pasture. One of the reasons neural networks struggle with out-of-domain data is called shortcut learning (Geirhos et al., 2020), in which the model learns decision rules that work well under testing conditions similar to the training ones, but are not robust to shifts in the data distribution. In the presented example, the shortcut would be to classify as camel if the background of the image is the color of sand and classify as cow if it is green.

Another problem neural networks have is known as catastrophic forgetting (McCloskey & Cohen, 1989), which is “*the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information*”<sup>1</sup>. Continuing with the cows and camels example, if we take that already trained network, and optimize it to differentiate cats and dogs, it will no longer classify cows and camels correctly, even on data with the same distribution as the training data.

In this work, we studied how data augmentation can help to mitigate the two problems presented. We observed the shortcut learning problem in the visual question answering task (Antol et al., 2015). In this task, the inputs are an image and a question about that image, and the goal is to answer the question correctly, based on the image contents. Moreover, the questions commonly require to perform several sequential steps, thereby, besides relating the image and language contents, the task requires some degree of reasoning. We observed that a synthetic dataset for this task had consistencies in the data, specifically some regions in the images were always empty, and created a modified version of the dataset that did not present this pattern. We found that the model trained on the original dataset had problems in the new setting, and we took the approach of preferring a flexible architecture and letting the model incorporate the required biases from the data. We found that simply incorporating data augmentation helped to mitigate some of the problems the model had and resulted in a better performance in the modified dataset.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Catastrophic\\_interference](https://en.wikipedia.org/wiki/Catastrophic_interference)

For the catastrophic forgetting problem, Javed and White (2019) proposed a meta-learning approach, specifically, to learn a feature extractor whose output representations could be used for learning without interfering with previously acquired knowledge, that showed promising results. However, for the few-shot image classification problem (a task contained in the meta-learning field), Dhillon, Chaudhari, Ravichandran, and Soatto (2019), and, Tian, Wang, Krishnan, Tenenbaum, and Isola (2020) showed that the representations of a feature extractor trained through traditional supervised learning could be used to surpass the performance of more complex meta-learning models or training methods, specifically designed for that task. Based on these results, we posed the following question: *can we use the representations of a model trained through traditional supervised learning for continual learning?* We answer this question positively, by simply initializing the weights of the classification layer with zeroes, a network trained through traditional supervised learning can achieve a performance comparable to the meta-trained models under the same evaluation procedure. However, we found that without any data augmentation the model trained with traditional supervised learning has a lower generalization performance, but including data augmentation eliminated this difference.

The remainder of this document is organized as follows, chapter 2 presents a brief overview of the relevance data augmentation has had in some computer vision tasks and the recent trend of using flexible neural network architectures combined with a data-centric approach.

Because we studied separate tasks, for ease of readability, we divided them into two chapters, in chapter 3 we refer to the visual question answering task and chapter 4 to the catastrophic forgetting problem. For each task and contained in their respective chapters, we present their specific related work, the theoretical framework under which our experiments are conducted, the datasets, experiments performed, and the results with an analysis of them.

Finally, we present our conclusions on chapter 5 and possible extensions to this work on chapter 6.

## **2. RELATED WORK**

### **2.1. Data augmentation in computer vision**

Deep neural networks are highly data-inefficient, they require enormous amounts of samples to work properly, and even in this case may overfit, thus the use of data augmentation has become fundamental for these models to learn the proper invariances (Simard, Steinkraus, & Platt, 2003), for example, the *Alexnet* suffered from overfitting when training on the 1.2 million images of the ImageNet dataset (Deng et al., 2009) without data augmentation, as mentioned by the authors (Krizhevsky et al., 2012). Data augmentation also has a fundamental importance for self-supervised learning, when labels are generated automatically instead of relying on human annotators (T. Chen, Kornblith, Norouzi, & Hinton, 2020; He, Fan, Wu, Xie, & Girshick, 2020; X. Chen, Fan, Girshick, & He, 2020). Finally, reinforcement learning is a setting notoriously plagued with poor data efficiency, but Laskin, Srinivas, and Abbeel (2020), and, Laskin, Lee, et al. (2020) showed that data augmentation allows to train an agent from pixels with better sample efficiency and test-time generalization than other methods that try to improve data efficiency, such as using world models.

### **2.2. Learning inductive biases from the data**

Although inductive biases have been fundamental for successfully training artificial neural networks, for example, the convolution (LeCun, Haffner, Bottou, & Bengio, 1999) or recurrence and memory (Hochreiter & Schmidhuber, 1997), in some cases they may not be trivial to embed in a network design, or even unknown. On the other hand, general purpose architectures, such as the transformer network (Vaswani et al., 2017), combined with large-scale pretraining, have become an established solution across diverse machine learning research fields, such as natural language processing (Radford et al., 2018; Devlin et al., 2019), computer vision (Dosovitskiy et al., 2021; Bertasius, Wang, & Torresani,

2021), speech recognition (Baeovski et al., 2019), and, vision and language multimodality (L. H. Li et al., 2019).

Wu et al. (2021) claimed that one of the reasons the transformer has had such a success, is because it is flexible enough to learn inductive biases from the pretraining tasks, and backed this up by pretraining a transformer with a synthetic task designed to facilitate mathematical reasoning, which outperformed a manually designed hierarchical transformer without pretraining. Manning et al. (2020) also arrived at similar findings, they showed that structures of natural language, such as the tree structure of a sentence, could be reconstructed from the representations learned by a transformer. More evidence of this is seen in computer vision, when replacing the convolutional operator with an attention mechanism, patterns very similar to the convolution emerge (Ramachandran et al., 2019; Cordonnier et al., 2019). Also related is a recent theoretical work that suggested that the linearized version of the transformer architecture resembles a network that generates the weights of another network on-the-fly based on the input (Schmidhuber, 1992; Schlag, Irie, & Schmidhuber, 2021).

We consider important to mention that, despite the fact that most of the works presented above regarding learning inductive biases from the data use the transformer network and we do not use this architecture in our experiments, we consider the present work relevant to this area because we focused on the overlying principle of using a data-centric approach, in our case with the use of data augmentation, rather than the architecture.

### 3. VISUAL QUESTION ANSWERING

In this chapter, we begin by presenting a brief overview of the visual question answering task, the shortcut learning problem and the fact that convolutional neural networks learn to encode spatial information. We then describe the model, dataset and experiments. Finally, we report our results with an analysis of them.

#### 3.1. Related work

##### 3.1.1. Visual Question Answering



Figure 3.1. An example of a visual question answer image-question pair.

Visual question answering (VQA) (Antol et al., 2015) is the task of answering open-ended questions in natural language about an image, as is shown in figure 3.1. This task requires to integrate components from several fields, such as computer vision, natural language processing and understanding, commonsense reasoning, and others. In the image-question pair of figure 3.1 we notice that to successfully answer the question we need to, locate the umbrella, know how umbrellas look upright and extract the umbrella's orientation in the pictures.

Two of the components the models for the VQA task have are a visual stream and language stream. For processing the image, the first systems used grid features of a convolutional neural network, but Anderson et al. (2018) argued that a more natural approach

would be to work at the level of objects, and used an object detector (Ren, He, Girshick, & Sun, 2015) for the visual stream, as shown on the left side of figure 3.2. This approach dominated the VQA leader boards on real-world images for some time (Y. Jiang et al., 2018; Z. Yu, Yu, Cui, Tao, & Tian, 2019), but H. Jiang, Misra, Rohrbach, Learned-Miller, and Chen (2020) revisited the use of grid features and showed that a model that used grid-level features, with a visual stream like the one shown in the right side of figure 3.2, outperformed the same model but using object-level features. Two benefits of using grid features are not being restricted to only the classes known by the object detector, and being able to train the visual stream *end-to-end* on the VQA task, which cannot be done easily with an object detector, since region extraction is an operation not trivial to make differentiable.

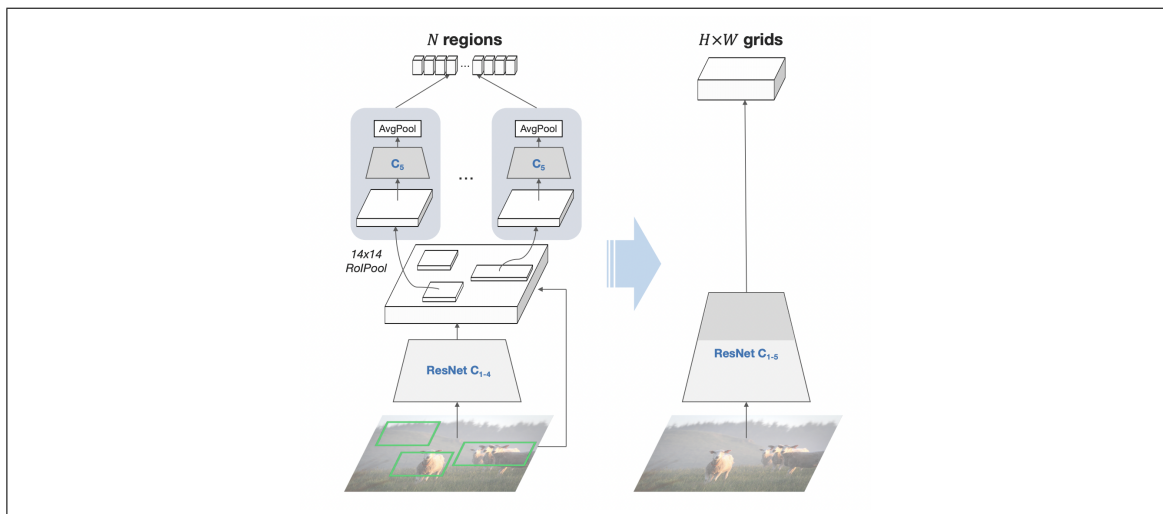


Figure 3.2. On the left it is shown the pipeline of an object-level visual stream and on the right the grid-level alternative. Figure from H. Jiang et al. (2020).

Regarding the question, we again see two high-level approaches. On the one hand, the methods that use a whole-question or per-word (or both) representation of the question (Santoro et al., 2017; Anderson et al., 2018; Perez, Strub, De Vries, Dumoulin, & Courville, 2018; Hudson & Manning, 2018), commonly generated with a recurrent neural

network. On the other hand, the methods that from the question dynamically build an executable program or define a sequence of modules, that are applied to the image (Andreas, Rohrbach, Darrell, & Klein, 2016; Johnson, Hariharan, Van Der Maaten, et al., 2017; Hu, Andreas, Rohrbach, Darrell, & Saenko, 2017; Yi et al., 2018; Mascharka, Tran, Soklaski, & Majumdar, 2018; Mao, Gan, Kohli, Tenenbaum, & Wu, 2018), which we will refer to as program-based methods. The program-based methods require to define the available operations or modules, that can be hand-designed or an automatized alternative could be to obtain them from a syntactic dependency parser, and, in some cases, these methods need additional information about the structure of the images. This helps the program-based methods have a better data efficiency on synthetic VQA datasets, with images and questions generated algorithmically, such as Johnson, Hariharan, van der Maaten, et al. (2017), compared to non-program-based methods, but with small benefit when using more data, and perform worse with human-made questions or real looking images, because their pre-defined action set restricts them.

The most recent works on VQA follow the trend established in natural language processing of pretraining large-scale transformer models (Radford et al., 2018; Devlin et al., 2019), but with modifications for vision-language objectives. We again see that the first methods processed the visual input with an object detector (Lu, Batra, Parikh, & Lee, 2019; L. H. Li et al., 2019; Tan & Bansal, 2019; G. Li, Duan, Fang, Gong, & Jiang, 2020), but were subsequently outperformed by a method that used convolutional grid features (Huang, Zeng, Liu, Fu, & Fu, 2020), although posterior works with an object detector surpassed the grid-based approach, but with larger datasets and models (X. Li et al., 2020; Gan et al., 2020; F. Yu et al., 2020; Zhang et al., 2021). We see that the trend of preferring more flexible architectures is also seen for the VQA task, which is why in our experiments we worked with a less structured model and took a data-centric approach.


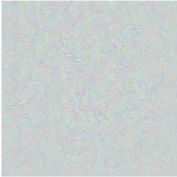
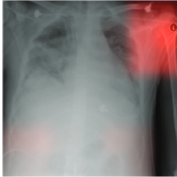
				<p>Article: Super Bowl 50</p> <p>Paragraph: "Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had a jersey number 37 in Champ Bowl XXXIV."</p> <p>Question: "What is the name of the quarterback who was 38 in Super Bowl XXXIII?"</p> <p>Original Prediction: John Elway</p> <p>Prediction under adversary: Jeff Dean</p>
Task for DNN	Caption image	Recognise object	Recognise pneumonia	Answer question
Problem	Describes green hillside as grazing sheep	Hallucinates teapot if certain patterns are present	Fails on scans from new hospitals	Changes answer if irrelevant information is added
Shortcut	Uses background to recognise primary object	Uses features irreco-gnisable to humans	Looks at hospital token, not lung	Only looks at last sentence and ignores context

Figure 3.3. Shortcut learning examples. *“Deep neural networks often solve problems by taking shortcuts instead of learning the intended solution, leading to a lack of generalization and unintuitive failures”* (Geirhos et al., 2020).

### 3.1.2. Shortcut learning

Overfitting is a common problem in neural networks, because of their extreme memorization capacity, they can achieve almost perfect performance on the training set, but poor or considerably lower generalization performance. Geirhos et al. (2020) describe another problem these systems have also related to memorization, which they call *shortcut learning* and define as “[...] decision rules that perform well on standard benchmarks but fail to transfer to more challenging testing conditions, such as real-world scenarios”, rephrasing, neural networks learn rules, shortcuts, that allow them to perform well on test sets taken from the same distribution as the training examples, but fail to generalize to out-of-distribution test sets, on which these rules do not hold, some examples shown in figure 3.3. There are some proposals that aim to improve out-of-distribution generalization (Arjovsky, Bottou, Gulrajani, & Lopez-Paz, 2019), but they have failed to outperform the empirical risk minimization baseline (Gulrajani & Lopez-Paz, 2021), which is fitting a model to the available training data. We consider that more flexible neural networks are more susceptible to this problem, compared to architectures with strong inductive biases, because they are more prone to learn the non-predictive biases of the data, and observed in our experiments that the model learned shortcuts in the VQA task, which caused it to



fail when modifying the data distribution. We show how a data-centric approach, in this case simply using data augmentation, helped to mitigate this problem.

### **3.1.3. Convolutional neural networks are not translation equivariant**

Even though the convolution is a translation equivariant function and we consider this inductive bias fundamental in some of deep learning's achievements, the powerfulness of neural networks is such that they learn to encode the absolute position of the image, the current evidence points out that this occurs due to implicit data leakage from the zero-padding (Islam, Jia, & Bruce, 2019; Kayhan & Gemert, 2020), widely used to retain spatial dimensions and prevent information loss. One may consider the absolute position as useful information, but it may be exploited as shortcuts learned by the network, as reported by (Alsallakh, Kokhlikyan, Miglani, Yuan, & Reblitz-Richardson, 2021).

## 3.2. Theoretical Framework

### 3.2.1. Compositional Attention Networks

In this work, we used the Compositional Attention Network framework (Hudson & Manning, 2018), or MAC (short for Memory, Attention and Composition), designed to facilitate reasoning by decomposing the problem into a series of steps, which are directly inferred from the data in an end-to-end approach, and executed by a recurrent neural network. This can be considered as an inbetween of the program-based and non-program-based methods mentioned in section 3.1.1, where the set of possible operations is inferred from the data instead of being predefined. We chose this method because is one of the best performing non-program-based alternatives on the synthetic CLEVR dataset (Johnson, Hariharan, van der Maaten, et al., 2017), is a more simple and general purpose architecture, and it is computationally cheap and fast to train.

---

**Algorithm 1:** MAC framework algorithm

---

Variables in bold are vectors and the rest are scalars

**Require:**  $I$ : image

**Require:**  $q_1 \dots q_s$ : tokenized question

**Output :**  $y$ : prediction

```

1  $\mathbf{k}_{1,1} \dots \mathbf{k}_{H,W} \leftarrow \text{ImageEncoder}(I)$ 
2  $\mathbf{q}, \mathbf{cw}_1 \dots \mathbf{cw}_s \leftarrow \text{QuestionEncoder}(q_1 \dots q_s)$ 
3  $\mathbf{c}_0 \leftarrow \text{InitializeControl}()$ 
4  $\mathbf{m}_0 \leftarrow \text{InitializeMemory}()$ 
5 for  $i = 1, 2, \dots, k$  do
6    $cv_{i,1} \dots cv_{i,s} \leftarrow \text{QuestionAttentionScores}_i(\mathbf{c}_{i-1}, \mathbf{q}, \mathbf{cw}_1 \dots \mathbf{cw}_s)$ 
7    $\mathbf{c}_i \leftarrow \sum_{j=1}^s cv_{i,j} \cdot \mathbf{cw}_j$ 
8    $rv_{i,1,1} \dots rv_{i,H,W} \leftarrow \text{ImageAttentionScores}(\mathbf{k}_{1,1} \dots \mathbf{k}_{H,W}, \mathbf{m}_{i-1}, \mathbf{c}_i)$ 
9    $\mathbf{r}_i \leftarrow \sum_{h,w=1,1}^{H,W} rv_{i,h,w} \cdot \mathbf{k}_{h,w}$ 
10   $\mathbf{m}_i \leftarrow \text{WriteMemory}(\mathbf{r}_i, \mathbf{m}_0 \dots \mathbf{m}_{i-1}, \mathbf{c}_0 \dots \mathbf{c}_i)$ 
11  $y \leftarrow \text{Answer}(\mathbf{q}, \mathbf{m}_k)$ 

```

---

A high-level overview of the MAC framework is presented in algorithm 1. The image is encoded using the pretrained convolutional neural network ResNet101 (He, Zhang, Ren, & Sun, 2016), with its weights frozen, and two additional trainable convolutional layers, which generates a feature map  $\mathbf{k}_{h,w} \in \mathbb{R}^{H \times W \times D}$ , where  $H, W$  are the spatial dimensions and  $D$  the features' dimensionality. The question is encoded with a bidirectional long-short term memory network (Graves & Schmidhuber, 2005), that generates a question embedding  $\mathbf{q}$  and a per-word context word  $\mathbf{cw}_j$ , which are the hidden states of the words. The network uses a control  $\mathbf{c}_i$  and memory  $\mathbf{m}_i$  vectors, both initialized with zeroes. Then it performs  $k$  fixed reasoning steps, on each step it generates scalar attention scores  $cv_{i,j}$  over the context words, which are used for a weighted sum over them to generate the control vector for the  $i$ -th step  $\mathbf{c}_i$ , that intuitively represents the operation the network performs. Likewise, scalar attention scores are obtained for the image regions  $rv_{i,h,w}$ , where the positions with a higher affinity with the control have a higher value, and are used for a weighted sum to generate the read vector  $\mathbf{r}_i$ . At the end of every iteration, a vector  $\mathbf{m}_i$  is stored as memory. Finally, the model outputs a classification over the set of possible answers. All the functions used in the algorithm, ImageEncoder, QuestionEncoder, QuestionAttentionScores $_i$ , ImageAttentionScores, and Answer, are neural operations learned end-to-end on the VQA task.

The attention scores for the context words  $cv_{i,j}$  and the image feature map  $rv_{i,h,w}$  are probability distributions, non-negative and sum 1, and can be used as a proxy to interpret the operation the model is performing on each step, we show an example of this in figure 3.4.

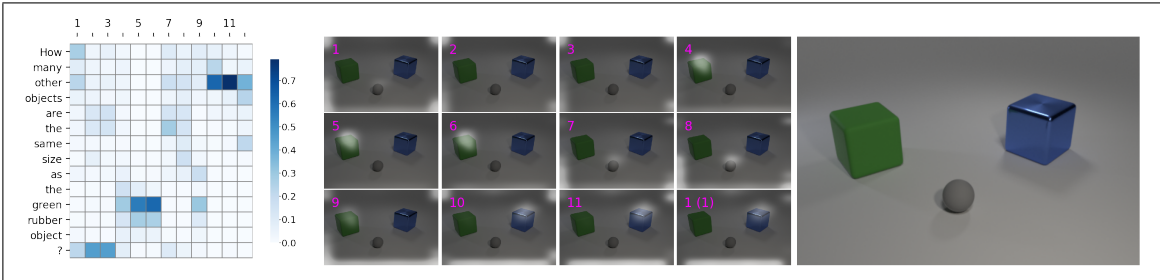


Figure 3.4. Example of the MAC framework with  $k = 12$  steps. The table on the left shows the attention scores over each context word for each step. On the center is the attention the network applies on the image for each step, with clearer spots meaning that the score is higher. The step number is indicated on magenta at the top left corner of each image, and the last image of the grid shows the answer given by the model and the correct answer in brackets. On the right is the image in a larger size for ease of visualization. We can see that at steps 4 through 6 the attention score on the words *green* and *rubber* is high, which translates to the model fixating on an object with these attributes on the images at those steps.

### 3.3. Dataset

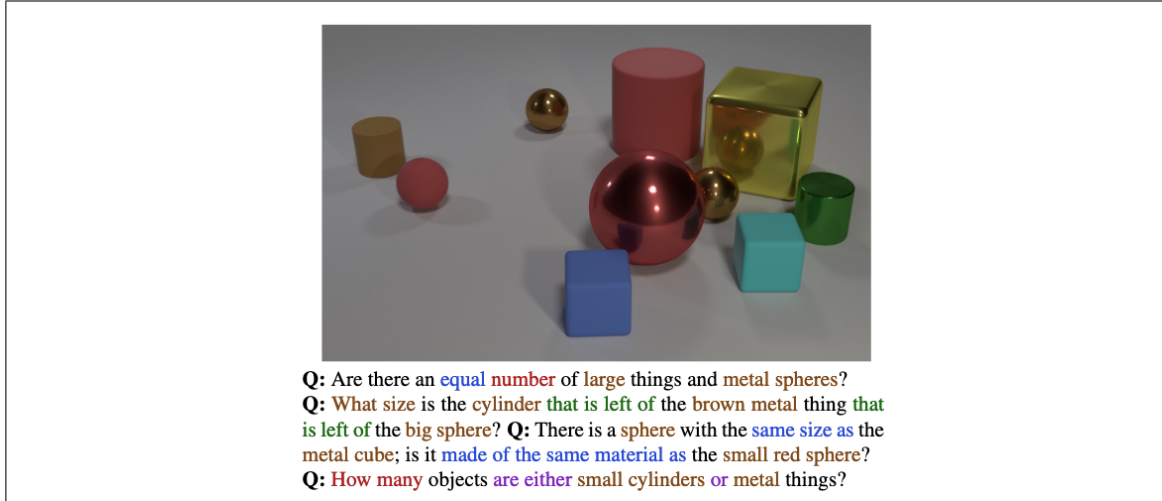


Figure 3.5. Examples of an image and questions of the CLEVR dataset (Johnson, Hariharan, van der Maaten, et al., 2017).

We used the CLEVR dataset (Johnson, Hariharan, van der Maaten, et al., 2017), that consists of synthetically generated images and questions designed to measure visual reasoning, an example shown in figure 3.5. Because it is synthetic, we have control over its settings, which allowed us to modify it. The CLEVR images are simple scenes that contain between 3 and 10 objects. Each object has a shape, *cube*, *sphere* or *cylinder*, a color, *gray*, *red*, *blue*, *green*, *brown*, *purple*, *cyan* or *yellow*, a size, *big* or *small*, and a material, *rubber* or *metal*. Questions are generated algorithmically based on the image contents by selecting and composing from a set of 26 pre-defined operations, such as *count*, *compare color*, *find intersection*, *relate spatially*, and others. The number of operations used to generate each question varies between 2 and 16, and is a measure of the number of reasoning steps or complexity of the question. The possible answers are *yes*, *no*, an attribute of the object, or a number between 3 and 10. The dataset contains 70.000 training images and 699.989 questions, with close to 10 questions per image, and a validation set of 15.000 images with 149.991 questions.

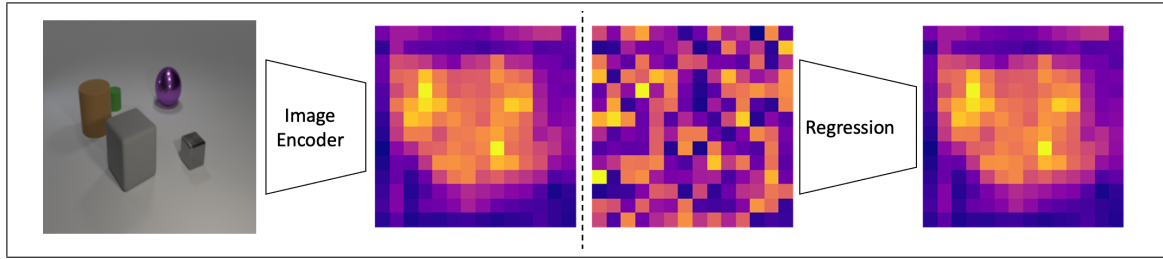


Figure 3.6. Depiction of the preliminary experiment. We used the image encoder to extract the convolutional features of each image, and, with them, trained a multinomial regression that had to predict to which position each vector corresponded to, like reordering the pieces of a puzzle, but each piece is only seen individually.

### 3.4. Preliminary Experiments

This work was motivated by a preliminary analysis of the MAC network. We saw that the attention it applied to the image tended to focus on the borders and corners of it, as can be seen in figure 3.4. What stood out most about this was that to reliably find these regions, the absolute spatial position of each region must be encoded in the representations generated by the convolutional network, a translation equivariant operation. To test this hypothesis we extracted the convolutional features of each image, since the spatial resolution of the grid features is of  $14 \times 14$ , from each image we obtained 196 vectors. We then trained a multinomial regression that received as input an individual vector and had to predict the position it corresponded to, a 196-way classification. This can be seen as reordering the pieces of a puzzle, but each piece is seen only individually, as depicted in figure 3.6. We used the first 1000 images of the CLEVR train dataset to fit our model and evaluated on the following 200. The trained regressor had an accuracy of 99.34%, which validated that convolutional networks learn to encode the spatial position. In retrospect, this is an expected behavior, because it can help on the task the model was trained for, such as image classification. During the time this work was done, several concurrent works were published which reported this phenomenon and inquired deeper into it (Islam et al., 2019; Kayhan & Gemert, 2020; Alsallakh et al., 2021).

### 3.5. Modified CLEVR

We investigated if the tendency the model had to focus on the borders could have a detrimental effect. We conjectured that it arised because on the CLEVR dataset the objects always appear at the center of the image. For this reason we took advantage of the fact that it is a synthetic dataset and built a modified version of it, for which we tried to distribute the objects uniformly across the whole image. We present the pixel occupancy of both datasets in figure 3.7, we see that, although there is still a small frame of low occupancy on the edges of the modified dataset, the objects are much more distributed. The modified dataset has the same number of images and questions as the original CLEVR, both for training a validation.

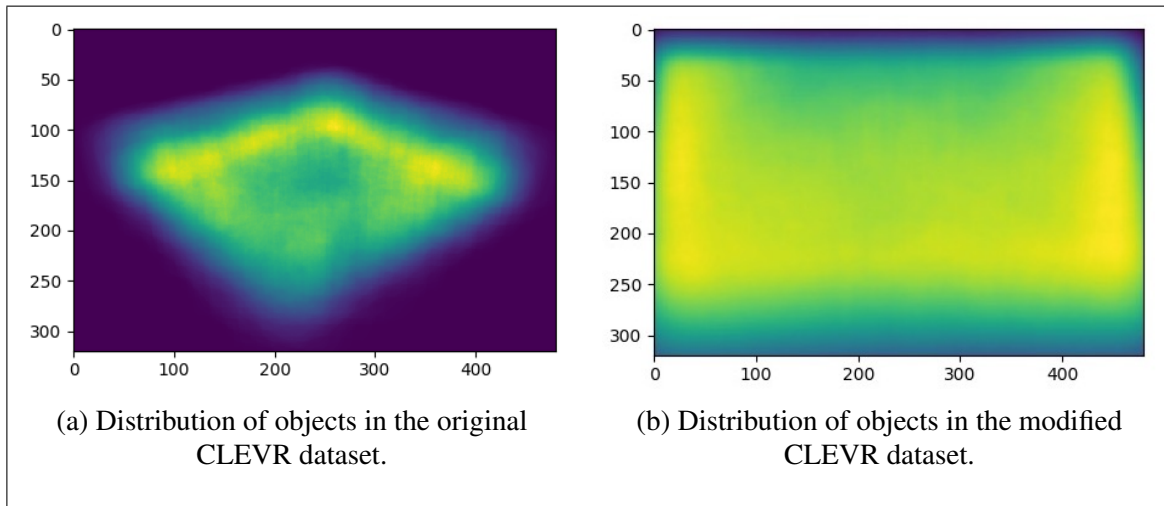


Figure 3.7. Color intensity shows how frequently each pixel was occupied by an object, darker color means less frequent. Objects in the original CLEVR are placed in a diamond shape at the center of the scene, leaving the corners unused, while the placement is much more distributed on the modified dataset.

### 3.6. Experiments

Our goal was to determine if the bias of focusing on the edges could have a negative impact on the model’s generalization capacity. For this reason, we trained the MAC

network on the original CLEVR dataset and evaluated it on the modified dataset. We compared its performance with a model trained directly on the modified CLEVR. Since we saw that the model trained on the original dataset had problems to generalize to the modified dataset, and motivated by the benefits data augmentation has shown, mentioned in section 2.1, we took the approach of incorporating this technique, instead of modifying the architecture of the model, as an additional comparison.

### **3.6.1. Evaluation**

We used the question accuracy, the percentage of questions for which the model predicted the correct answer, as performance metric.

### **3.6.2. Setup**

#### **3.6.2.1. Neural network architecture**

We used the MAC network with  $k = 12$  reasoning steps, that performed the best on its original work. As visual backbone we used a pretrained ResNet101 (He et al., 2016), without modifying its weights while training.

#### **3.6.2.2. Data processing**

The original images have size  $480 \times 320$ , and are resized to  $224 \times 224$  pixels before feeding them into the ResNet. When data augmentation was used, it consisted in a random perspective transformation applied before resizing. Examples of the effect of this transformation are shown in figure 3.8, as it can be seen, in some cases the objects are cut off from the original image, which can result in an unanswerable question. We observed that this caused noise during training but the model still had a good evaluation performance.



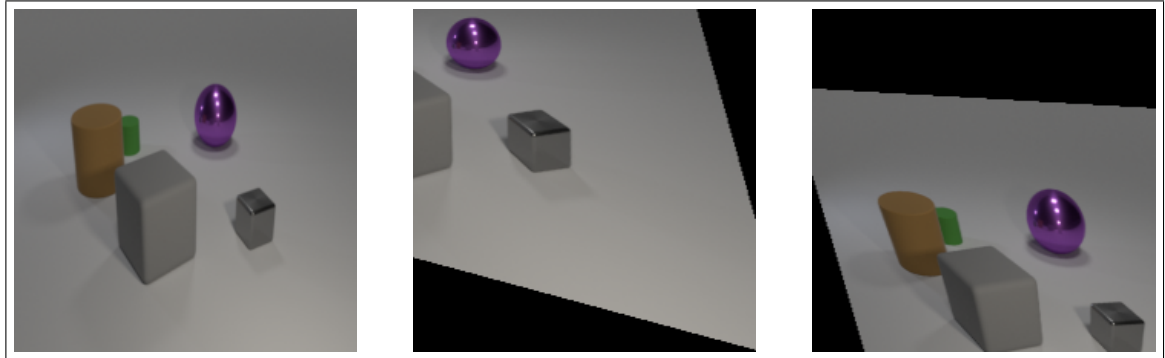


Figure 3.8. Examples of the perspective transformation. The first is the original image.

### 3.6.2.3. Optimization setup

We used the same optimizer and hyperparameters as in the original MAC work, *i.e.*, Adam optimizer (Kingma & Ba, 2014) with learning rate  $1e^{-4}$ . We trained the models without augmentation for 25 epochs and for 50 epochs when augmentation was included, because, as mentioned, it caused a slight instability while training. Each experiment was trained with 3 random seeds.

### 3.6.2.4. Implementation

We used a Pytorch (Paszke et al., 2019) implementation of the MAC network<sup>1</sup>.

---

<sup>1</sup><https://github.com/tohinz/pytorch-mac-network>

### 3.7. Results

On table 3.1 we present the accuracy on the original and modified datasets for our experiments, each ran with three random seeds. We see that the model trained on the original CLEVR had an accuracy of 98.40% on that dataset, but decreased to 84.89% on the modified dataset. The inclusion of data augmentation caused a small decrease of performance on the original dataset, but a big improvement on the modified dataset, with an absolute increase of 5.58% in accuracy. Despite the fact that the accuracy of the model trained with data augmentation is low compared to training directly on the modified dataset, it still is a big reduction of the gap.

Table 3.1. Accuracy on the original and modified datasets. Intervals show the 95% confidence interval.

Training dataset	Accuracy on original CLEVR	Accuracy on modified CLEVR
Original CLEVR	98.40 $\pm$ 0.27	84.89 $\pm$ 1.69
Modified CLEVR	96.89 $\pm$ 0.32	97.10 $\pm$ 0.41
Original CLEVR with augmentation	98.05 $\pm$ 0.17	90.47 $\pm$ 0.64

### 3.8. Analysis

We inspected the behavior and problems the model trained without data augmentation had and compared them with the model trained with data augmentation to find the situations in which data augmentation helped. In total, there were 5790 questions which the three seeds of the model without data augmentation got wrong and all seeds of the model with data augmentation got right, but also 1280 questions in the opposite way, that the three seeds of the model trained with data augmentation got wrong, but all seeds of the

model without data augmentation got right. One problem we managed to identify was that without data augmentation, the model learned a simple rule for the size of the objects based on the number of pixels it occupied on the image, but did not learn the correct notion of perspective, so it thinks that *big* objects on the back of the scene are *small*, and also, *big* objects at the front located at the edges of the image were often considered as *small*, because they were partially visible, as can be seen in figure 3.9.

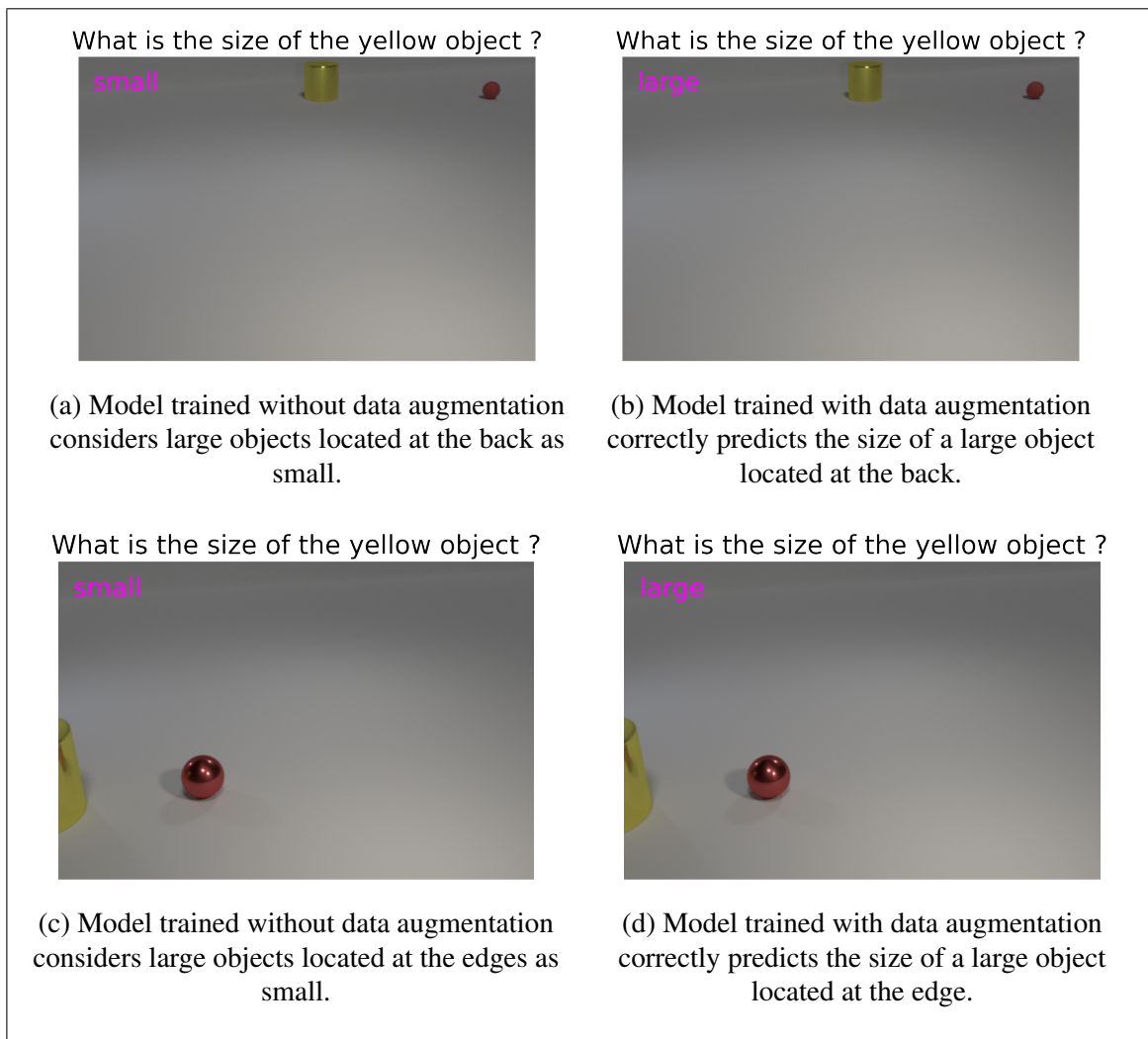


Figure 3.9. Size problems of the model trained without data augmentation. The left side contains the predictions of the model trained without data augmentation and the right side of the model with data augmentation. Each image has the question as title and the answer given by the model in magenta. These images were generated manually apart from the dataset.

We identified two additional problems related to the objects being located farther away, specifically, the model without data augmentation frequently got wrong the material or the shape of the objects, as we show in figure 3.10. We remark that we did not perform an exhaustive search for all possible problems.

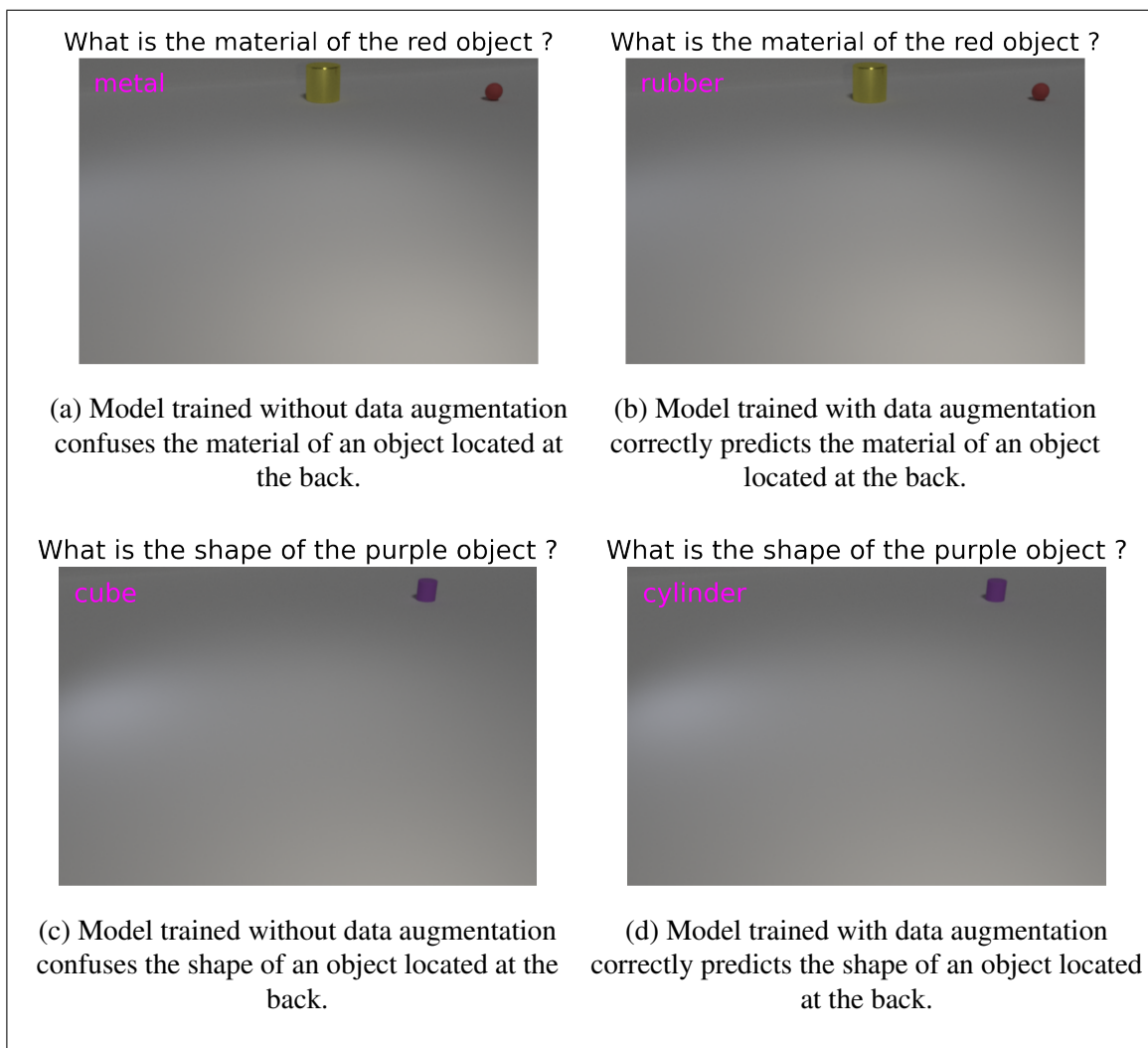


Figure 3.10. Material and shape problems of the model trained without data augmentation. The left side contains the predictions of the model trained without data augmentation and the right side of the model with data augmentation. Each image has the question as title and the answer given by the model in magenta. The errors occur when objects are farther away and may be hard to distinguish even to the human eye. These images were generated manually apart from the dataset.

To have a better quantification of the impact of the objects' positions and data augmentation have, we divided the images of the modified CLEVR in regions of  $20 \times 20$  pixels, and, for each region we computed the accuracy the model had on the questions that to be answered required to reason about an object located in that region, we explain below how we made this division. We present this analysis in figure 3.11. We see that the model trained on the original dataset shows a clear deficiency when it has to reason about objects located in the back, and also on the lateral and front edges to a lower degree. The model trained with data augmentation has a much better performance over all regions and the impact of having to reason about objects in the back of the scene appears to be much lower.

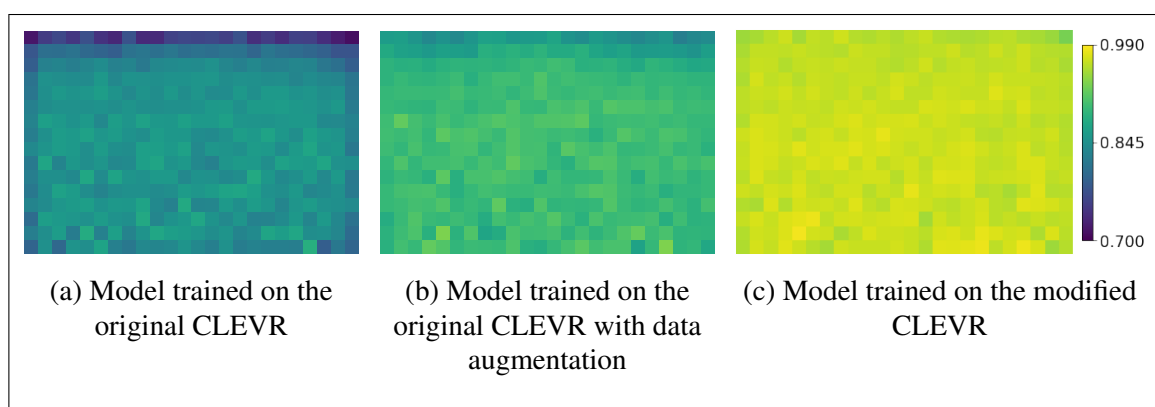


Figure 3.11. Image divided in regions of  $20 \times 20$  pixels and accuracy of the model on the questions that to be answered require to reason about an object located in that region. The model trained on the original dataset shows clear problems when it needs to reason about objects away from the center.

Regarding our initial goal of investigating if the tendency the model had of focusing on the edges had an unwanted effect, our hypothesis was that when evaluated in the modified CLEVR, a model trained on the original CLEVR would focus on the borders where objects are now present, and read incorrect information. Based on the observed behavior, we divided the situations in which the model focused on the edges into two. First, when it was searching for the presence of an object with a certain description, but there was not any object that satisfied it on the scene. Secondly, the cases in which no clear instructed

could be interpreted from the context words attention, we think that this may effectively be a *skip* or *do nothing* instruction, because the model performs a fixed amount of reasoning steps (12 in our case), but some questions are much simpler and do not require so many steps. As a side note, methods that aim to directly avoid performing *skip* steps (Graves, 2016; Eyzaguirre & Soto, 2020) are important to this specific problem, but, as mentioned, in this work we focused on the effect of data augmentation.

We quantitatively analyzed if the tendency of focusing on the borders had an impact by separating the question-image pairs of the modified CLEVR into three splits, (i) the ones where the image does not contain any object in the bordering 60 pixels, (ii) the ones where the image contains objects in the bordering 60 pixels, but the question does not require to reason about them, and (iii) the ones where the question requires to reason about an object located in the bordering 60 pixels. To determine if a question required to reason about an object located in the borders, we used the available information for each scene and the operations to generate each question, specifically, each operation returns a set of objects, for example, the operation *filter\_color* receives as input a set of objects and a color, and returns as output all objects of the input set that are of the requested color. We considered that a question required to reason about an object if it was present in any of the outputs of the operations used to create the question. We present the results of this analysis on table 3.2.

First, we noted that for case (i) the accuracy of the model without data augmentation is of 98.69%, and of 98.62% when including data augmentation, showing that, as expected, the problems occur when objects are located at the borders. For case (ii), the accuracy without data augmentation is of 96.93% and of 97.78% with data augmentation, we would have expected a bigger difference if our hypothesized problem had occurred constantly, for this reason, we delved into this and comment about our findings in the following paragraph. Finally, for case (iii) the model trained without data augmentation had an accuracy of 84.17% and using data augmentation increased it to 90.03%, which provides evidence of the benefits of data augmentation.

Table 3.2. Accuracy on the modified CLEVR question-pairs split based on the presence of objects in the bordering 60 pixels and if the questions requires to reason about them. In parenthesis the number of questions and the intervals show the 95% confidence interval.

Trained on	(i) Accuracy on questions that do not have objects in the bordering 60 pixels (2.420)	(ii) Accuracy on questions that have objects in the bordering 60 pixels, but do not require to reason about them (5.740)	(iii) Accuracy on questions that require to reason about an object in the bordering 60 pixels (141.840)
Original CLEVR	$98.69 \pm 0.42$	$96.93 \pm 1.01$	$84.17 \pm 2.28$
Modified CLEVR	$99.34 \pm 0.79$	$99.28 \pm 0.03$	$96.97 \pm 0.54$
Original CLEVR with augmentation	$98.62 \pm 0.88$	$97.78 \pm 0.45$	$90.03 \pm 0.85$

As mentioned, we delved into the questions of case (ii). From the 5740 questions that had objects in the bordering 60 pixels but did not require to reason about them, we extracted the questions that the three seeds of the model trained without data augmentation got wrong, and the three seeds of the model with data augmentation got right, which amounted to 44 questions. We found that almost all errors were due to the fact that according to the criteria we used to divide the dataset, the question did not require to reason about an object in that location, but if the model got an attribute of an object wrong, it could have used that object in its reasoning. We show an example of this for the model trained without data augmentation in figure 3.12. Most importantly, from our inspection none of the

errors were caused by the model reading incorrect information from the borders, which disproves our inquired hypothesis.

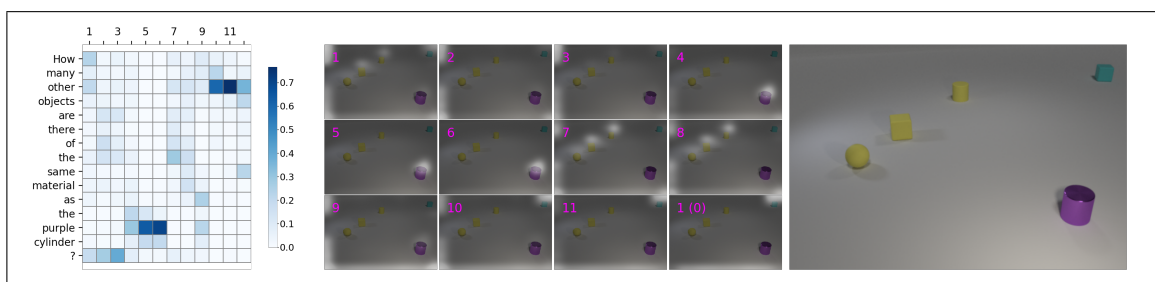


Figure 3.12. Example of a question that according to the criteria we used should not require to reason about an object in the bordering 60 pixels, answered by model trained without data augmentation. Because the model thought the *cyan block* was *metallic* it answered incorrectly.

Finally, from the questions of case (ii), there were 22 such that the three seeds of the model trained with data augmentation got wrong and the model trained without data augmentation got right. We observed that in some cases, the model trained with data augmentation also confused the material of objects located at the back, which caused it to answer incorrectly. There were some other issues related to perspective, for example, the model did not consider an object *left* from another, but from the scene information it was, possibly caused by the transformation we used.



## 4. CATASTROPHIC FORGETTING

In this chapter, we begin by giving a brief overview of the solutions to tackle the catastrophic forgetting problem and research on meta-learning that motivated the present work. We then present the theoretical framework of the performed experiments. Finally, we describe our experiments and present our results and analysis.

### 4.1. Related work

#### 4.1.1. Continual Learning and the Catastrophic Forgetting Problem

On one side, neural networks are trained through mini-batch stochastic gradient descent, in which we repeatedly sample a small subset (batch) of the training data and minimize its loss function, on the other side, humans learn sequentially, once we are taught to differentiate between cats and dogs, no further retraining is required. We would want a neural network to behave just as humans do and be able to learn continually, without constantly revisiting information, but McCloskey and Cohen (1989) found that “*training on a new set of items may drastically disrupt performance on previously learned items*”, and coined this as the catastrophic forgetting problem.

The solutions proposed to tackle the catastrophic forgetting problem can be divided into three categories. First, the replay-based methods that store samples of the data stream, and, when new information arrives, they interleave the new data with the stored data, and add some of the new samples to the stored data for later use (Aljundi, Lin, Goujaud, & Bengio, 2019; Chaudhry, Marc’Aurelio, Rohrbach, & Elhoseiny, 2019; Lopez-Paz & Ranzato, 2017; Rebuffi, Kolesnikov, Sperl, & Lampert, 2017; Riemer et al., 2018). Due to possible constraints in storing information, some variants simultaneously train a generative network that is used to produce synthetic versions of the older information for interleaving, effectively using the weights of the generator network as a data storage (Shin, Lee, Kim, & Kim, 2017). It is also possible to store the intermediate representations of the network instead of the raw data, which showed an improved efficiency (van de Ven,

Siegelmann, & Tolias, 2020). The use of a data storage is also commonly used in reinforcement learning, where agents act sequentially in an environment, making it prone to suffer from catastrophic forgetting, and using old stored data has a substantial importance to the performance of some algorithms (Lin, 1993; Wawrzyński, 2009; Mnih et al., 2013).

A second class of proposals try to prevent the network from forgetting the relevant information for older data by reducing the plasticity of the weights relevant for those tasks (Zenke, Poole, & Ganguli, 2017; Kirkpatrick et al., 2017; Hurtado, Lobel, & Soto, 2021). In some cases, this requires computing the weight's importance for the task and adding a regularization term while learning new data that penalizes the changes to the important weights.

The third category of methods rely on learning representations that are not as vulnerable to catastrophic forgetting, and are mainly focused on the evidence that sparse representations are better suited for continual learning (French, 1991; X. Liu et al., 2018; Masse, Grant, & Freedman, 2018; V. Liu, Kumaraswamy, Le, & White, 2019). Another approach tries to skip the manual specification of some desired properties for the representations, and leverages meta-learning to directly optimize for the task of continual learning (Javed & White, 2019). A follow up work incorporated a neuromodulatory network, a parallel branch of the neural network, that acts as a gating mechanism to turn off neurons of the main branch and easily generate sparsity (Beaulieu et al., 2020).

#### **4.1.2. Meta-learning, few-shot learning, and Model-Agnostic Meta-learning**

Meta-learning is a subfield of machine learning where the objective is to learn how to learn (Schmidhuber, 1987), it includes the few-shot learning problem, in which a model has to learn new tasks (or classes) with a small number of training (support) examples, this problem can be thought of as a mitigation to the massive data requirements deep learning commonly has. Among the meta-learning methods, the most relevant to this work is the Model-Agnostic Meta-learning (MAML) (Finn, Abbeel, & Levine, 2017), that poses the training procedure as a bi-level optimization, composed of an inner and outer loop. In

the inner loop it is emulated that the model is learning one or more tasks (each with a small number of examples) using gradient descent, and, in the outer loop, a loss function measures how well the model learned the tasks. The gradients of this outer loss function are backpropagated through the inner optimization to modify the parameters of the model before beginning the inner loop. This can be interpreted as finding an initialization from which new tasks can be learned rapidly.

In the original MAML setting, the weights of all layers are modified in the inner loop, but Raghu, Raghu, Bengio, and Vinyals (2019) showed that modifying only the classification layer can reach a similar performance. Furthermore, even when the weights of all the layers are modified, the ones of the nonclassification layers remain very similar to their values before the inner loop. These findings suggest that MAML is approximately learning a fixed feature extractor. Related to this, Tian et al. (2020) showed that using the representations of a neural network trained through traditional supervised learning with a multinomial regression can perform similarly or surpass the performance of models trained with meta-learning methods, specifically designed for the few-shot learning task, which further gives hints on how powerful simple deep learning can be. Our work is strongly related to Tian et al. (2020), since we investigated if the representations of a network trained through traditional supervised learning can perform on par with the representations of a network trained with a meta-learning method for catastrophic forgetting.

## 4.2. Theoretical Framework

### 4.2.1. Traditional Supervised Learning

In the most common deep learning for supervised learning scenario, which we will refer to as traditional supervised learning (TSL), we have a dataset of training examples  $\mathcal{D}^{train} = \{(\mathbf{x}_t, y_t)\}_{t=1}^T$ , and the goal is to learn a function  $f_{\theta}$  that for every input  $\mathbf{x}_t$  produces as output the corresponding label  $y_t$  of the dataset. This function, and specifically its parameters  $\theta$ , is learned through mini-batch gradient descent, where  $\theta$  are iteratively modified in the direction that minimizes the loss function  $\mathcal{L}$  of a sampled subset of  $\mathcal{D}^{train}$ , this subset is called batch and has size  $B$ . This is expressed in the equation 4.1.

$$\theta = \arg \min_{\theta} \mathbb{E}_{\{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_B, y_B)\} \sim \mathcal{D}^{train}} \left[ \frac{1}{B} \sum_{i=1}^B \mathcal{L}(f_{\theta}(\mathbf{x}_i), y_i) \right] \quad (4.1)$$

### 4.2.2. The Continual Learning Problem Formulation

A continual learning problem (CLP) consists of an unending stream of samples, or trajectory  $\mathcal{T}$ , where,

$$\mathcal{T} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t), \dots \quad (4.2)$$

and a model  $f$  that has to learn from this stream of data. The state of the model at time  $t$  is denoted as  $\mathbf{s}_t$  and a learning algorithm  $\mathcal{A}$  is used to update it,  $\mathbf{s}_t = \mathcal{A}(\mathbf{s}_{t-1}, \mathbf{x}_t, y_t)$ , the model uses its state to generate a prediction  $\hat{y} = f(\mathbf{s}, \mathbf{x})$ . The goal of the continual learning problem is to find a model such that after updating its state with the  $t$ -th datapoint, it maintains a high performance on all previously seen datapoints  $\{(\mathbf{x}_i, y_i)\}_{i \leq t}$ . If we denote  $p(\mathcal{T})$  as the trajectories' distribution and  $\mathcal{L}$  as the loss of a single datapoint, the loss function for the continual learning problem is expressed in the following equation 4.3 (we omit possible normalization factors).

$$\mathcal{L}_{CLP} = \mathbb{E}_{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_j, y_j), \dots = \mathcal{T} \sim p(\mathcal{T})} \left[ \sum_{t \geq 1} \sum_{i=1}^t \mathcal{L}(f(\mathbf{s}_t, \mathbf{x}_i), y_i) \right] \quad (4.3)$$

### 4.2.3. Online-aware Meta-learning

Hand-designed solutions have dominated the approaches to tackle the catastrophic forgetting problem, because the loss presented in equation 4.3 can be infeasible to optimize directly, due to the fact that the trajectories are expected to contain hundreds or thousands of samples, which would need big computational requirements to store the computation graph of all state updates for backpropagation, and, a phenomenon like vanishing or exploding gradients could be expected from all these passes through the network.

Javed and White (2019) proposed to use an approximation of the CLP loss, and, instead of hand-designing a new mechanism, they adapted the MAML framework to directly optimize for continual learning, which they called Online-aware Meta-learning (OML). The method divides the trajectory in sub-trajectories referred to as tasks  $\tau_j$ , for example, the trajectory could contain images of digits between 0 and 9, with 10 images per digit, and each task contains the 10 images of a single digit. It also assumes that the model has a set of known tasks  $\mathcal{T}^{known}$  that it can perform properly, and train in the MAML inner-outer loop fashion. In the inner loop, the model has to sequentially learn  $n \geq 1$  new tasks,  $\mathcal{T}^{new}$ , with multiple samples per task, for simplicity we will assume that each task has  $k$  samples. To learn these tasks, the model sequentially performs a gradient descent step on each sample separately to update its parameters, which translates in  $n \times k$  gradient steps. After the inner loop, on the outer loop, it is measured that the model learned to perform the tasks it just saw,  $\mathcal{T}^{new}$ , and also that it remembers how to perform the tasks it knew before,  $\mathcal{T}^{known}$ , *i.e.*, that it learned sequentially and did not forget. The gradients of the parameters before the inner loop with respect to this loss function are computed by backpropagating through the inner optimization and used to update them, which can be interpreted as finding a network that retains the information it has, even after updating its parameters on new

tasks. This formulation avoids the potential thousands of updates of the full trajectory, and instead performs  $n \times k$  updates, which can be a few tens.

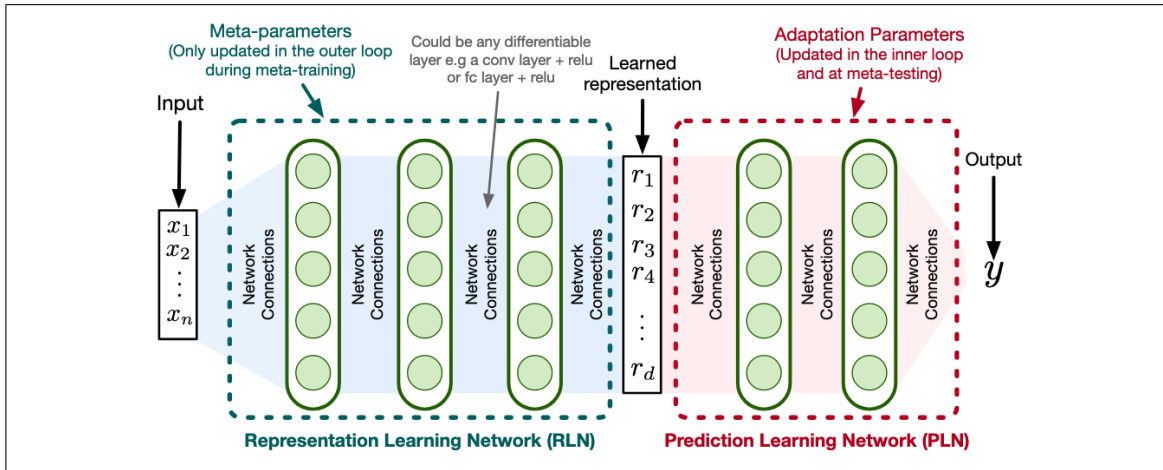


Figure 4.1. An example of the architecture proposed by Javed and White (2019). During the inner loop only the parameters of the prediction learning network (PLN) are updated by taking a gradient step with respect to each individual sample of the subtrajectory. In the outer loop both the representation learning network and PLN parameters are updated by taking a gradient step with respect to the approximate CLP loss. Figure taken from Javed and White (2019).

The bi-level optimization is repeated, sampling different inner and outer tasks each time, until convergence, when the model should be able to continually learning. Some additional considerations, to simulate that the model does not know how to perform the tasks to be seen in the inner loop, the parameters of the classification layer corresponding to those tasks are reinitialized (without this the model would simply memorize all new tasks, since there is a limited amount of data available). Additionally, the model has two sets of parameters, the *meta parameters* that belong to a representation learning network (RLN), and the *adaptation parameters* of a prediction learning network (PLN). The RLN acts as a fixed feature extractor during the inner loop and its parameters are only modified in the outer loop. The adaptation parameters of the PLN are modified both during the inner and outer loops. A schematic of this architecture is shown in figure 4.1.

Using the CLP notation, the model  $f$  is a neural network, the state is composed of the RLN parameters  $\theta$  and the PLN parameters  $\phi$ , and the learning algorithm is  $\mathcal{A}$  is presented in the following equation 4.4,

$$\mathcal{A}((\theta, \phi), \mathbf{x}, y) = (\theta, \phi - \alpha \nabla_{\phi} \mathcal{L}(f(\theta, \phi, x), y)) \quad (4.4)$$

where  $\alpha$  is the inner learning rate hyperparameter. The complete OML training procedure is summarized in algorithm 2.

---

**Algorithm 2:** Online-aware Meta-Learning training procedure

---

**Require:**  $p(\mathbf{X}^{known})$ : distribution of samples of known tasks  
**Require:**  $p(\mathbf{T}^{new})$ : distribution over new tasks  
**Require:**  $\alpha, \beta$ : inner and outer step size hyperparameters  
**Output :** RLN and PLN parameters  $(\theta, \phi)$

- 1 Randomly initialize RLN  $\theta$  and PLN  $\phi$  parameters
- 2 **while not done do**
- 3     Sample new tasks  $\tau_1 \dots \tau_n \sim p(\mathbf{T}_{new})$  without repetition
- 4     **for**  $i = 1, 2, \dots, n$  **do**
- 5          $\phi \leftarrow$  Reinitialize classification weights for task  $\tau_i$  on  $\phi$   
        // The  $i$ -th column of the weights
- 6      $\phi' \leftarrow \phi$
- 7      $(\mathbf{X}^{new\_outer}, \mathbf{Y}^{new\_outer}) \leftarrow \{\}, \{\}$
- 8     **for**  $i = 1, 2, \dots, n$  **do**
- 9          $(\mathbf{X}^{new\_inner}, \mathbf{Y}^{new\_inner}) \leftarrow$  Sample  $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_k, y_k)$  from  $\tau_i$
- 10         **for**  $(\mathbf{x}_j, y_j) \in (\mathbf{X}^{new\_inner}, \mathbf{Y}^{new\_inner})$  **do**
- 11              $\phi' \leftarrow \phi' - \alpha \nabla_{\phi'} \mathcal{L}(f(\theta, \phi', \mathbf{x}_j), y_j)$  // Apply the learning  
            algorithm
- 12         Sample and add  $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_q, y_q)$  from  $\tau_i$  to  $\mathbf{X}^{new\_outer}$  and  $\mathbf{Y}^{new\_outer}$
- 13      $(\mathbf{X}^{known}, \mathbf{Y}^{known}) \leftarrow$  Sample  $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_c, y_c)$  from  $p(\mathbf{X}^{known})$
- 14      $(\theta, \phi) \leftarrow (\theta, \phi) - \beta \nabla_{\theta, \phi} \mathcal{L}(f(\theta, \phi, \mathbf{X}^{new} \cup \mathbf{X}^{known}), \mathbf{Y}^{new} \cup \mathbf{Y}^{known})$

---

#### 4.2.4. A Neuromodulated Meta-learning

Beaulieu et al. (2020) built on the evidence that suggested that gating or inhibition mechanisms helped to mitigate catastrophic forgetting, and complemented the OML method with a two-branched neural network that they called A Neuromodulated Meta-learning Algorithm (ANML). This architecture is composed of a neuromodulatory network (NM) and a prediction network (PN), as seen in figure 4.2. A sigmoid activation function is applied to the output of the NM to obtain values between 0 and 1, and this result is used in an elementwise multiplication with the output of the PN, to selectively turn off some activations of the PN and generate sparsity.

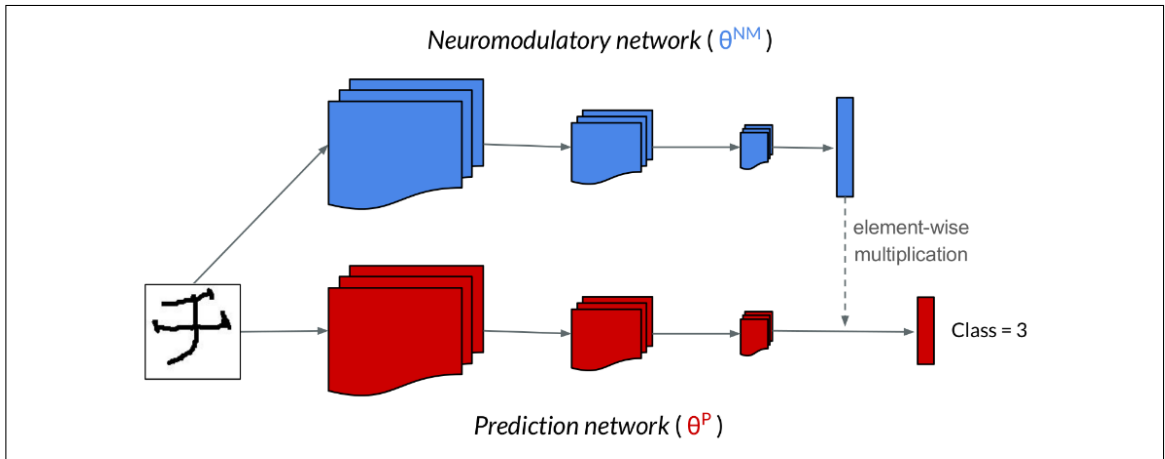


Figure 4.2. The ANML network proposed in Beaulieu et al. (2020). The neuromodulatory network (NM) is used to inhibit the activations of the prediction network (PN). During training, only the parameters of the prediction network are modified in the inner loop and the parameters of both networks are modified in the outer loop. Figure taken from Beaulieu et al. (2020).

Under the CLP notation, there are two neural network functions,  $h$  for the PN with parameters  $\theta^P$  and  $g$  for the NM with parameters  $\theta^{NM}$ , the state is  $s = (\theta^{NM}, \theta^P)$ , and  $\theta^P$  is also subdivided in the parameters of the body of the PN and the classification parameters  $\theta^P = (\theta_{body}^P, W)$ . The function that defines the ANML network is  $f(\theta^{NM}, \theta^P, \mathbf{x}) =$



$W(\sigma(g(\theta^{NM}, \mathbf{x})) \cdot h(\theta_{body}^P, \mathbf{x}))$ , where  $\sigma$  is the sigmoid activation function,  $\cdot$  is an elementwise multiplication and  $W$  is multiplied as a matrix product (we omit the bias in the classification layer for simplicity). This network is trained using the OML training procedure, and only the parameters of the PN are modified during the inner loop and both during the outer loop, the learning algorithm  $\mathcal{A}$  is expressed in the following equation 4.5.

$$\mathcal{A}((\theta^{NM}, \theta^P), \mathbf{x}, y) = (\theta^{NM}, \theta^P - \alpha \nabla_{\theta^P} \mathcal{L}(f(\theta^{NM}, \theta^P, x), y)) \quad (4.5)$$

### 4.3. Experiments

Our goal was to determine if the representations of a model trained through TSL can be used for continual learning, as Tian et al. (2020) showed for the image few-shot learning case. For this reason we replicated their setting, pretrain a neural network (on the *meta-training* set, which we will describe bellow) and freeze all but the classification layer during evaluation. We compared against the OML method, because it is the most prominent approach that focuses on learning representations before continual learning evaluation, most of the other approaches learn the representations *online*, during evaluation. As mentioned, on the OML framework, before each training inner loop, the weights of the classification layer for the tasks to be learned are randomly reinitialized, we found that reinitializing them with zeroes provided a small improvement, and included this variant in the comparison and denoted it with *zero-init*. We also investigated if the architecture had an impact, for this reason we performed experiments with both a standard *single-branch* convolutional neural network (the network in the original OML work), as showed in figure 4.1, and a *neuromodulated* network (the ANML network) depicted in figure 4.2.

Motivated by the importance data augmentation has in computer vision, presented in section 2.1, we also conducted experiments that incorporated data augmentation, to see if it had an impact on the continual learning problem.

#### 4.3.1. Dataset

We used the Omniglot dataset (Lake, Salakhutdinov, & Tenenbaum, 2015), composed of images of characters from 50 different alphabets, each alphabet containing several classes, totaling 964 classes for training and 659 for testing, and each class has 20 datapoints. For hyperparameters search, we split the training set and used the first 664 for training and the remaining 300 for validation. For testing, we used the models trained on all 964 training classes with the setup determined during validation. To prevent confusion, in the meta-learning field the training set is often referred to as *meta-training* set and the

testing set as *meta-testing* set. Examples of some Omniglot characters are shown in figure 4.3.



Figure 4.3. Characters of the Omniglot dataset from 8 different alphabets.

#### 4.3.2. Evaluation

We used the same evaluation protocol as in Javed and White (2019), and, Beaulieu et al. (2020), a trajectory of  $t \in \{10, 50, 75, 100, 200, 300, 400, 600\}$  classes is sampled from the test split, and for each class 15 datapoints are sampled. The classification layer of the trained models is updated sequentially, one sample at a time, with gradient descent. On the shortest trajectory, 150 gradient updates are performed, and 9000 on the longest. This process is called *meta-test training*.

After *meta-test training*, the accuracy of the resulting model is evaluated on all the datapoints of the sampled trajectory. This is called *meta-test training* accuracy, it measures raw sequential memorization of the seen datapoints. Finally, for each task in the trajectory we evaluate the accuracy of the model on the remaining 5 datapoints that were not in the *meta-test training* trajectory. This aims to measure the capacity of the model to generalize to unseen datapoints while learning sequentially. This is referred to as the *meta-test testing* accuracy. For each trajectory length  $t$  we repeat this process 50 times, with different tasks and *meta-test training* and *meta-test testing* datapoints each time.

We also observed that shuffling and interleaving classes of different alphabets during testing had an impact. For example, if in a sampled trajectory there were two alphabets,

each with two classes, without shuffling, the model would see the two characters of the first alphabet and then the characters of the second alphabet, when shuffling, this order is no longer necessarily maintained. We report the results for both shuffled and unshuffled classes.

### 4.3.3. Setup

#### 4.3.3.1. Neural network architectures

For the *single-branch* network, we used a network with 6 convolutional layers with ReLU activation function, and a final linear classification layer. For the *neuromodulated* network, both the neuromodulatory branch and the prediction branch had 3 convolutional layers, each convolution followed by an instance normalization (Ulyanov, Vedaldi, & Lempitsky, 2016) and a ReLU activation. The neuromodulatory branch has a linear layer with sigmoid activation function at the end and the prediction network has a linear classification layer.

During training with the OML method, for the *single-branch* network only the parameters of the classification layer are modified during the inner loop. For the *neuromodulated* network all the parameters of the prediction network are modified in the inner loop.

During evaluation, for the TSL trained networks, we initialize the classification layer with zeroes for all weights and biases. For the OML trained models we report the results of initializing the classification layer randomly or with zeroes. For both methods and both models, during evaluation only the weights of the classification layer are modified and the remaining parameters are kept fixed.

#### 4.3.3.2. Data processing

As in the original works, the *single-branch* network received as input images of  $84 \times 84$  pixels and for the *neuromodulated* network the images were downsized to  $28 \times 28$  pixels. When data augmentation was used, it consisted of a random crop with zero padding.

#### 4.3.3.3. Optimization setup

As in Tian et al. (2020), we use SGD optimizer with momentum of 0.9, weight decay of  $5e^{-4}$  and mini-batch size 64 to train with the TSL objective. The training schedule was determined using the validation setup described in section 4.3.1, for the *single-branch* architecture we use a learning rate of 0.05 for 600 epochs. We trained the *neuromodulated* architecture for 700 epochs with initial learning rate of 0.05 and decay once after 600 epochs with factor 0.1.

We use the configuration of the original works to optimize with the OML method, *i.e.*, Adam optimizer (Kingma & Ba, 2014) with learning rates  $1e^{-3}$  for the *neuromodulated* architecture and  $1e^{-4}$  for the *single-branch* architecture.

#### 4.3.3.4. Implementation

For comparison, we present the results of the original implementations for the *single-branch*<sup>1</sup> and *neuromodulated*<sup>2</sup> models trained with the OML objective. We also reimplemented them in JAX (Bradbury et al., 2018), and used this framework as well for the TSL experiments.

---

<sup>1</sup><https://github.com/khurramjaved96/mrc1>

<sup>2</sup><https://github.com/uvm-neurobotics-lab/higherANML>

#### 4.4. Results

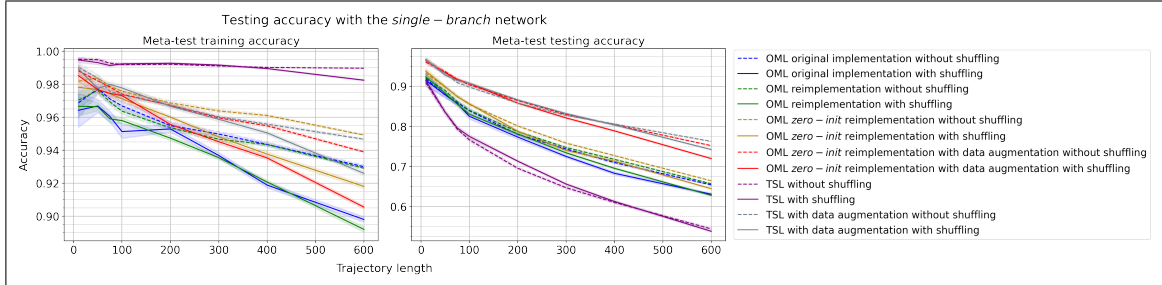


Figure 4.4. Evaluation performance of the *single-branch* network training with the OML or TSL loss. The  $x$  axis indicates the number of classes in the trajectory and the  $y$  axis the accuracy on the seen tasks on *meta-test training* and *meta-test testing*. Comparing dotted and solid lines, we see that shuffling the order of classes had a slight negative impact on performance. The TSL-trained network without data augmentation had a remarkable capacity to memorize inputs sequentially, but lacked on generalization. The inclusion of data augmentation boosted its generalization capacity to perform on par to the OML-trained model, sacrificing raw memorization, but still on level with the meta-trained model. Error bars show the 95% confidence interval for three random seeds, each tested on 50 trajectories for each trajectory length.

On figure 4.4 we show the results for the *single-branch* architecture. We see that shuffling the order of the classes had a slight negative impact on performance for all methods, which suggests that interleaving unrelated classes has a negative effect while learning sequentially. Also, we see that initializing the classification weights of the OML-trained method with zeroes provided an accuracy improvement in all cases.

The model trained with TSL but without data augmentation had a surprising raw memorization capacity and could sequentially memorize 600 tasks (9000 images in total) with an accuracy of 98%, but lacked generalization with an accuracy around 56%, attributable to how prone neural networks are to overfitting. On the other hand, the *zero-init* OML-trained method without data augmentation had a lower memorization, but still high with

an accuracy of over 90% on 600 tasks, and had a better generalization performance compared to the TSL-trained model, with over 62% accuracy, which shows that meta-learned features provide better generalization in this case.

Finally, data augmentation caused a *meta-test training* accuracy decrease for both methods, with the TSL-trained method suffering the most, but still performed on par with the OML-trained method. On the *meta-test testing* side, we see that both methods benefited from data augmentation, with a bigger boost for the TSL-trained method that reached an accuracy of  $\sim 74\%$  on the 600 tasks, slightly higher than the  $\sim 72\%$  reached by the *zero-init* OML-trained method.

The results for the *neuromodulated* network are shown in figure 4.5. The first thing to note is that with this architecture shuffling the order of the classes had a smaller impact, this may be attributable to the use of instance normalization, but we do not further inquire on this. Like for the *single-branch* network, zero initialization provided a small improvement.

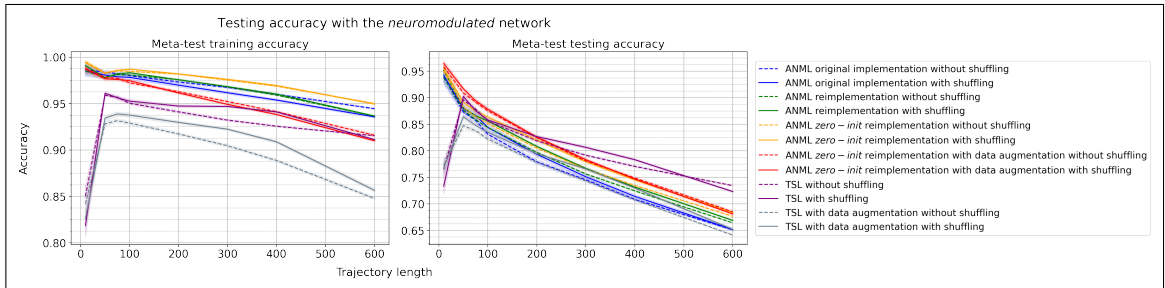


Figure 4.5. Evaluation performance of the *neuromodulated* network training with the OML (ANML) or TSL loss. The  $x$  axis indicates the number of tasks in the trajectory and the  $y$  axis the accuracy on the seen tasks on *meta-test training* and *meta-test testing*. Comparing dotted and solid lines, shuffling did not have a big impact on performance. The TSL-trained network without data augmentation had lower *meta-test training* performance, but surpassed the level of the ANML model in *meta-test testing*. The inclusion of data augmentation had a noticeably negative impact on the TSL-trained model, and, although it decreased the *meta-test training* performance of the ANML model, it provided a slight improvement on *meta-test testing*. Error bars show the 95% confidence interval for three random seeds, each tested on 50 trajectories for each trajectory length.

Without data augmentation, the TSL-trained method underperformed the ANML model on *meta-test training* accuracy, possibly because the normalization acts as a strong regularization, but still had a good performance with over 90% when learning 600 tasks, while the *zero-init* ANML reached  $\sim 95\%$  accuracy. However, on the *meta-test testing* performance, the TSL-trained model had an accuracy of  $\sim 72\%$ , higher than the  $\sim 68\%$  of the meta-trained model.

Regarding data augmentation, we see that it had a big negative impact on the TSL-trained model, both during *meta-test training* and *testing*, we think this may be because this shallow network with the TSL loss was unable to tolerate the increased complexity of characters moving around. On the other hand, for the ANML model, data augmentation caused a small decrease in *meta-test training* and small increment in *meta-test testing*.

Finally, despite the fact that the TSL-trained model reached a performance on par with the ANML model on long trajectories with over 300 classes, it was noticeably worse on short trajectories, of 100 classes or less, on the analysis we observed what may cause this.



## 4.5. Analysis

### 4.5.1. Learned representations

Our results showed that TSL can suffice to learn representations suitable for continual learning. Regarding the characteristics of the learned representations, Javed and White (2019) saw that, on one side, sparsity organically appeared in the representations of the OML-trained models, meaning that a high percentage of neurons had an activation with value zero per datapoint, but, on the other side, a small number of the activations were always zero across the dataset, which can be interpreted as having a high efficiency or usage of its resources. In this section we compared these representations with the ones of the TSL-trained models.

#### 4.5.1.1. *Single-branch* network

In figure 4.6 we show the activations for one random seed of the *single-branch* architecture trained either with OML or TSL, for 3 random *meta-test* instances on the first three columns, and the mean activation across all test datapoints on the fourth column. The used samples are included in appendix A.1 and the representations of the other random seeds in appendix A.2. These representations are the output of the RLN, that has 2304 neurons, and are reshaped to form an image with  $32 \times 72$  pixels, for ease of visualization. The per sample activations are normalized by the maximum value of that sample. We observed that the per-instance activation for both training methods appears to be highly sparse, with no noticeable difference between the methods or the use of data augmentation. Likewise, no clear difference can be seen across the OML variants on their mean activation, but the TSL-trained model with data augmentation appears to have a smoother mean activation, which suggests that it learned to distribute the representations across the available resources on a higher degree than the other methods.

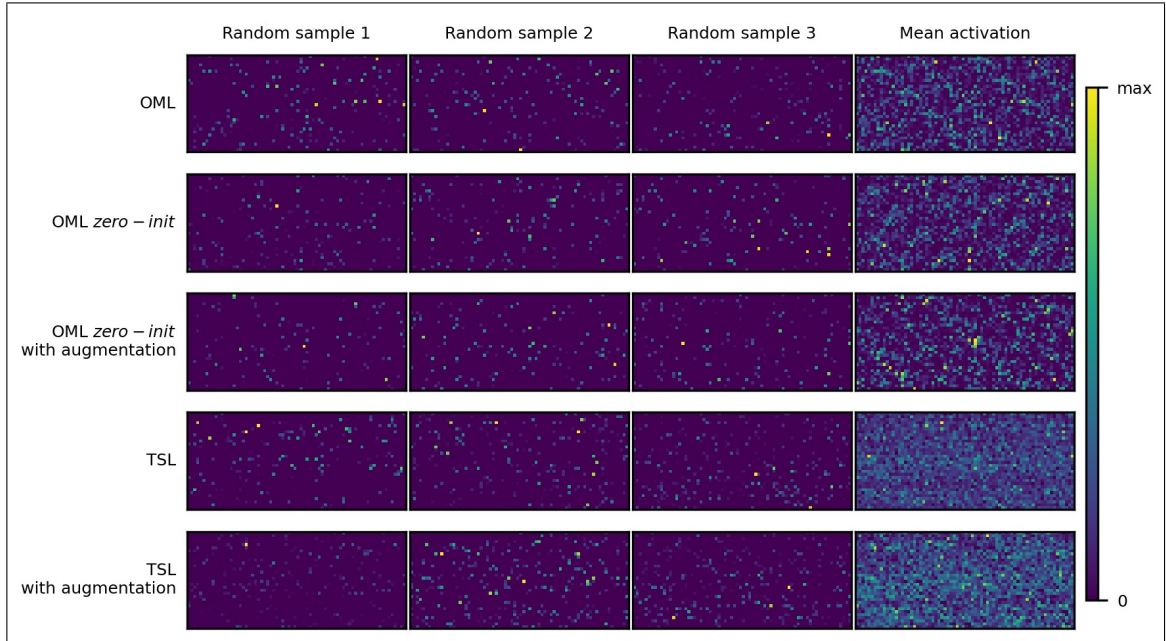


Figure 4.6. Representation learning network activations of the *single-branch* network. The first three columns show the representations of 3 sampled test instances for different training methods, normalized by the maximum value of that representation. The fourth column shows the mean activation across all test datapoints. No noticeable difference is seen between the per-instance representations based on the training method, but the TSL-trained model with data augmentation has a visually smoother mean activation.

#### 4.5.1.2. Neuromodulated network

The representations of a single random seed of the meta-trained *neuromodulated* architecture are shown in figure 4.7, the representations of the other random seeds are included in appendix A.3.1. As reported by Beaulieu et al. (2020), the NM network acts as a mechanism to generate sparsity on the representations, which are highly dense before applying the gating signal, but the modulation turns off most of the activations. In addition, the mean activation after the gating signal is highly distributed across the available neurons, reflecting a high resource efficiency.

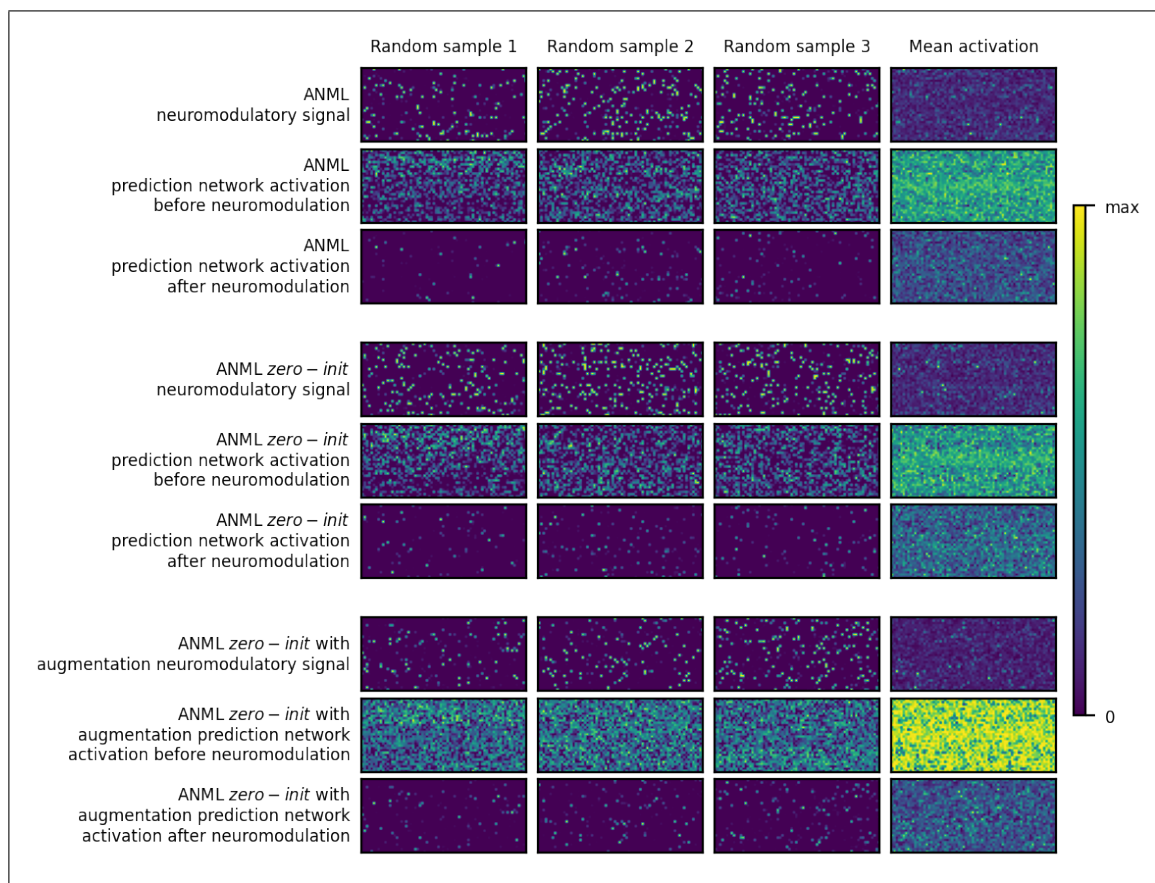


Figure 4.7. Activations for the meta-trained *neuromodulated* network. The first horizontal block shows the activation for the network trained with random reinitialization and without data augmentation, the second block is the *zero-init* variant, and the last block shows the *zero-init* with data augmentation. Each block contains three rows, the first shows the output of the neuromodulatory branch, the second the output of the prediction network before the neuromodulatory signal is applied and the third row the result after applying the modulation. The first three columns show the activations for random test samples, the last column shows the mean activation across all test data. We can see that the neuromodulatory signal acts as a strong mechanism to generate sparsity on the representations, which are highly dense before it is applied. And, despite the fact that each individual representation after modulation is sparse, the average activation is highly dense, meaning that the network efficiently uses the available resources.

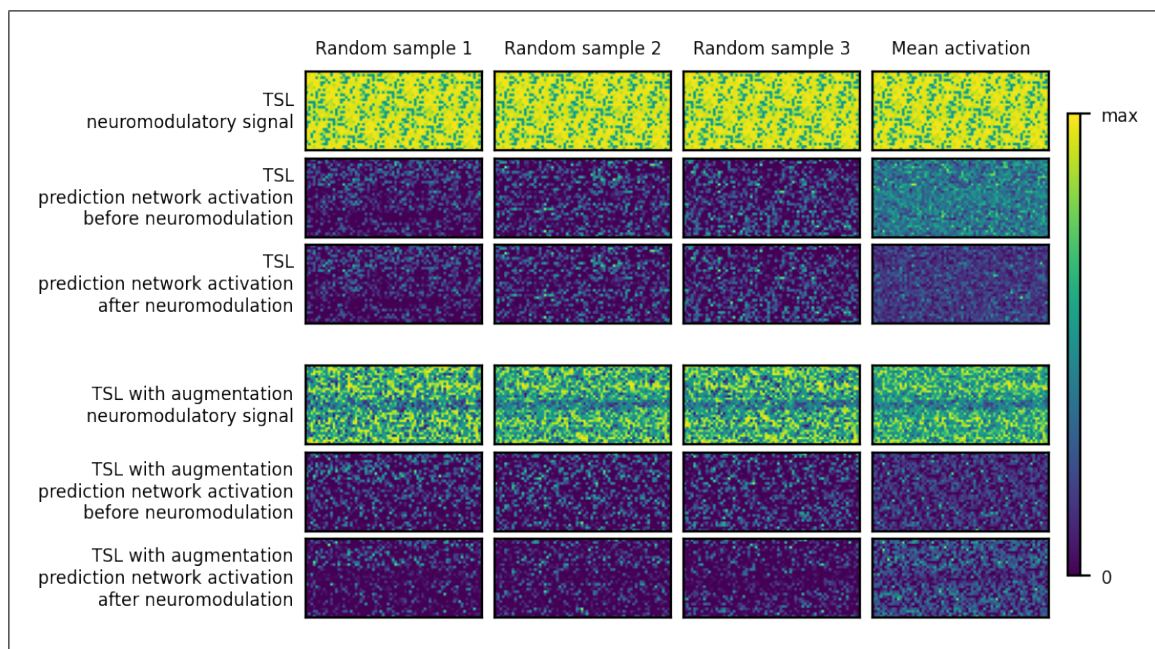


Figure 4.8. Activations for the TSL-trained *neuromodulated* network. The first horizontal block shows the activations for the network trained without data augmentation, and the second block for the activations when incorporating data augmentation. Each block contains three rows, the first shows the output of the neuromodulatory branch, the second the output of the prediction network before the neuromodulatory signal is applied and the third row the result after applying the modulation. The first three columns show the activations for random test samples, the last column shows the mean activation across all test data. We see that the model produced a constant neuromodulation for all samples, and the output representations are highly dense.

The representations of a single random seed of the TSL-trained *neuromodulated* network are shown in figure 4.8, the representations of the other random seeds are included in appendix A.3.2. Two interesting observations can be made, first, the output of the NM network is constant and does not change across the input, meaning that the network did not learn to use it, and probably was closer to a constraint during training. We also think this may be related to the fact that data augmentation had a negative impact for this training method and not for the meta-trained variant. Second, in this case the per instance activations do not appear to be as sparse compared to the meta-trained model or the

*single-branch* network. We attribute this to two factors, the shallowness of the network that may not allow sufficient complexity to generate sparsity, unlike the *single-branch* network case, and the use of instance normalization, because it normalized the activations to have a mean value of 0 (before the ReLU). We also think that this denser representations may be the reason this model had a lower performance on short trajectories, sparsity may be needed to properly learn short trajectories, which is why the meta-trained model was forced to use the neuromodulatory branch, but the fact that the TSL-trained method had a better performance when learning 300 or more classes makes us think that sparsity has a reduced relevance for longer trajectories.

#### 4.5.2. Sparsity and dead neurons

In table 4.1 we show the quantitative measure of sparsity, the average percentage of neurons with activation  $< 0.01$ , relative to the maximum value each neuron has across all training datapoints, and the percentage of dead neurons, the neurons with activation 0 for all datapoints. We include the data for the *meta-training* and *meta-testing* data splits.

For the *single-branch* architecture we observe that the OML-trained models have the highest sparsity, with a mean over 93%. We also see that for the TSL-trained models, data augmentation causes a  $\sim 2\text{-}3\%$  decrease in sparsity. Moreover, for most cases the number of dead neurons hovers between 1-4%, but surprisingly for the meta-trained model with data augmentation it is much higher, around 18%.

And, for the *neuromodulated* architecture, we see that the number of dead neurons is close to 0% for all cases, we think that this, again, can be related to the use of instance normalization. Regarding the sparsity, as expected we see a high level of sparsity for the meta-trained models, with over 94%, but for the TSL-trained models it lies in the  $\sim 32\text{-}34\%$ .

Table 4.1. Average sparsity and dead neurons with 95% confidence interval. The sparsity measures the percentage of neurons which activations was  $< 0.01$  relative to the maximum value each neuron had across all training datapoints, averaged across all datapoints in the corresponding split. The dead neurons indicates the percentage of neurons with activation 0 across all datapoints of the corresponding split. Confidence interval is omitted for values with mean zero.

Architecture	Training method	Data split			
		<i>Meta-train</i>		<i>Meta-test</i>	
		Sparsity	Dead neurons	Sparsity	Dead neurons
<i>Single-branch</i>	OML	93.18 $\pm 1.65$	2.92 $\pm 1.79$	93.28 $\pm 1.65$	3.63 $\pm 1.94$
	OML <i>zero-init</i>	93.72 $\pm 1.31$	1.97 $\pm 2.59$	93.87 $\pm 1.29$	2.69 $\pm 2.90$
	OML <i>zero-init</i> with data augmentation	94.97 $\pm 0.98$	17.90 $\pm 3.83$	94.95 $\pm 0.98$	18.26 $\pm 4.05$
	TSL	92.18 $\pm 0.92$	1.43 $\pm 3.92$	91.87 $\pm 0.82$	1.43 $\pm 3.92$
	TSL with data augmentation	89.38 $\pm 0.22$	1.53 $\pm 4.62$	88.51 $\pm 0.19$	1.61 $\pm 4.92$
<i>Neuromodulated</i>	ANML	94.29 $\pm 0.19$	0	94.45 $\pm 0.28$	0
	ANML <i>zero-init</i>	94.40 $\pm 0.45$	0	94.55 $\pm 0.51$	0
	ANML <i>zero-init</i> with data augmentation	94.75 $\pm 0.27$	0	94.78 $\pm 0.26$	0
	TSL	32.85 $\pm 0.20$	0	32.40 $\pm 0.20$	0
	TSL with data augmentation	35.32 $\pm 3.00$	1.43 $\pm 2.24$	35.10 $\pm 3.04$	1.43 $\pm 2.24$

Finally, about our claim that sparsity loses relevance on long trajectories, we don't observe correlations to support this. Specifically, the OML-trained *single-branch* architecture with data augmentation performed better on longer trajectories, but also had a slightly higher sparsity, compared to training with the same method but without data augmentation. On the contrary, when training with TSL data augmentation caused a reduction in sparsity and better performance on long trajectories. For the *neuromodulated* architecture we did not observe any clear signals related to this.

## 5. CONCLUSIONS

As we introduced in chapter 1 and then expanded on in chapter 2, despite the fact that inductive biases have been fundamental for deep learning, the current dominating trend is a data-centric approach in which the biases are directly inferred from the data. In this thesis, we specifically studied how data augmentation and pretraining could be leveraged to mitigate some of the problems deep learning has, using standard architectures and training techniques.

First, we showed that on a synthetic dataset for the visual question answering task, a simple deep learning model had problems for out-of-domain generalization and learned shortcuts which caused it to fail when modifying the data distribution, but data augmentation had a big positive impact in reducing some of the problems, which translated in a better generalization.

Secondly, we explored how important meta-learning is to learn representations for the catastrophic interference problem. In our experiments we saw that the representations of a model pretrained through traditional supervised learning can be as effective to learn continuously, and data augmentation had a big impact on the generalization capacity for a *single-branch* neural network. We think that our work opens the possibility of using large-scale pretraining to have models that are better for continual learning without forgetting, as was the case for the few-shot learning task.

Finally, we hope that this thesis increases the focus on the existing and upcoming research that studies data-centric approaches, because we consider that they have a broader applicability, a longer lasting validity and a more practical impact, than many of the new architectural designs.



## 6. FUTURE WORK

Our work related to shortcut learning was performed on a synthetic dataset, so a future work would be to evaluate this problem and the impact of data augmentation on a real world dataset. Additionally, the effect data augmentation has depends on the goal task, so designing general purpose augmentations can be a very impactful line of work, or the possibility to automatically select the augmentations that benefit a given task. We also think that how to properly leverage generative models to increase the available data can be an important future work.

Regarding the catastrophic forgetting problem, further work with more complex architectures and datasets has to be made to validate if our findings hold. We also think that the online learning algorithm we used for evaluation, which consisted of a single gradient step per datapoint, is not suitable for any neural network architecture, optimizer, or training method, for this reason we think that an online learning algorithm that can handle this variation is a necessary next step. With our work, we expected to provide initial evidence that pretraining can be very useful for continual learning, and meta-learning is not strictly required, for this reason, we think that self-supervised learning can be a relevant pretraining alternative.

Finally, it is also of importance to study how our findings can be extended in other research areas such as reinforcement learning, where pretraining with methods like curiosity, that autonomously seek novel states, have also been very important for fast policy learning, so, naturally, the next step is to study their effect on continual learning.

## REFERENCES

- Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019). Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 11816–11825.
- Alsallakh, B., Kokhlikyan, N., Miglani, V., Yuan, J., & Reblitz-Richardson, O. (2021). Mind the pad – cnns can develop blind spots. In *International conference on learning representations*.
- Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6077–6086).
- Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Neural module networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 39–48).
- Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the ieee international conference on computer vision* (pp. 2425–2433).
- Arjovsky, M., Bottou, L., Gulrajani, I., & Lopez-Paz, D. (2019). Invariant risk minimization. *ArXiv, abs/1907.02893*.
- Baevski, A., Schneider, S., & Auli, M. (2019). vq-wav2vec: Self-supervised learning of discrete speech representations. In *International conference on learning representations*.
- Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., & Cheney, N. (2020). Learning to continually learn. In G. D. Giacomo et al. (Eds.), *ECAI 2020*

(Vol. 325, pp. 992–1001). IOS Press.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., ... Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836), 77–82.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1), 151–175.

Bertasius, G., Wang, H., & Torresani, L. (2021). Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs*.

Chaudhry, A., Marc'Aurelio, R., Rohrbach, M., & Elhoseiny, M. (2019). Efficient lifelong learning with a-gem. In *7th international conference on learning representations, iclr 2019*.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).

Chen, X., Fan, H., Girshick, R., & He, K. (2020). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Cordonnier, J.-B., Loukas, A., & Jaggi, M. (2019). On the relationship between self-attention and convolutional layers. In *International conference on learning representations*.

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., & Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (p. 248-255). doi: 10.1109/CVPR.2009.5206848

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).

Dhillon, G. S., Chaudhari, P., Ravichandran, A., & Soatto, S. (2019). A baseline for few-shot image classification. In *International conference on learning representations*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International conference on learning representations*.

Eyzaguirre, C., & Soto, A. (2020). Differentiable adaptive computation time for visual reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12817–12825).

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135).

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference* (Vol. 1, pp. 173–178).

Gan, Z., Chen, Y.-C., Li, L., Zhu, C., Cheng, Y., & Liu, J. (2020). Large-scale adversarial training for vision-and-language representation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 6616–6628). Curran Associates, Inc.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11), 665–673.

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005*. (Vol. 4, pp. 2047–2052).

Gulrajani, I., & Lopez-Paz, D. (2021). In search of lost domain generalization. In *International conference on learning representations*.

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9729–9738).

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

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., & Saenko, K. (2017). Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision* (pp. 804–813).

Huang, Z., Zeng, Z., Liu, B., Fu, D., & Fu, J. (2020). Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. *arXiv preprint arXiv:2004.00849*.

Hudson, D. A., & Manning, C. D. (2018). Compositional attention networks for machine

reasoning. In *International conference on learning representations*.

Hurtado, J., Lobel, H., & Soto, A. (2021). Overcoming catastrophic forgetting using sparse coding and meta learning. *IEEE Access*, 9, 88279-88290. doi: 10.1109/ACCESS.2021.3090672

Islam, M. A., Jia, S., & Bruce, N. D. (2019). How much position information do convolutional neural networks encode? In *International conference on learning representations*.

Javed, K., & White, M. (2019). Meta-learning representations for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.

Jiang, H., Misra, I., Rohrbach, M., Learned-Miller, E., & Chen, X. (2020). In defense of grid features for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10267–10276).

Jiang, Y., Natarajan, V., Chen, X., Rohrbach, M., Batra, D., & Parikh, D. (2018). *Pythia v0.1: The winning entry to the vqa challenge 2018*.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., & Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Cvpr*.

Johnson, J., Hariharan, B., Van Der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., & Girshick, R. (2017). Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision* (pp. 2989–2998).

Kayhan, O. S., & Gemert, J. C. v. (2020). On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14274–14285).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv*

*preprint arXiv:1412.6980.*

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... others (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13), 3521–3526.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097–1105.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 1332–1338.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., & Srinivas, A. (2020). Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33.

Laskin, M., Srinivas, A., & Abbeel, P. (2020, 13–18 Jul). CURL: Contrastive unsupervised representations for reinforcement learning. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 5639–5650). PMLR.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision* (pp. 319–345). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/3-540-46805-6\_19

Li, G., Duan, N., Fang, Y., Gong, M., & Jiang, D. (2020). Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 11336–11344).

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., ... others (2020). Oscar: Object-  
semantics aligned pre-training for vision-language tasks. In *European conference on com-  
puter vision* (pp. 121–137).

Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. *PhD Thesis*.

Liu, V., Kumaraswamy, R., Le, L., & White, M. (2019). The utility of sparse represen-  
tations for control in reinforcement learning. In *Proceedings of the aaai conference on  
artificial intelligence* (Vol. 33, pp. 4384–4391).

Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A. M., & Bagdanov, A. D.  
(2018). Rotate your networks: Better weight consolidation and less catastrophic forget-  
ting. In *2018 24th international conference on pattern recognition (icpr)* (pp. 2262–2268).

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learn-  
ing. In *Proceedings of the 31st international conference on neural information processing  
systems* (pp. 6470–6479).

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolin-  
guistic representations for vision-and-language tasks. In *Neurips*.

Manning, C. D., Clark, K., Hewitt, J., Khandelwal, U., & Levy, O. (2020). Emergent  
linguistic structure in artificial neural networks trained by self-supervision. *Proceedings  
of the National Academy of Sciences*, 117(48), 30046–30054.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2018). The neuro-symbolic  
concept learner: Interpreting scenes, words, and sentences from natural supervision. In  
*International conference on learning representations*.

Mascharka, D., Tran, P., Soklaski, R., & Majumdar, A. (2018). Transparency by design:  
Closing the gap between performance and interpretability in visual reasoning. In *Proceed-  
ings of the ieee conference on computer vision and pattern recognition* (pp. 4942–4950).



Masse, N. Y., Grant, G. D., & Freedman, D. J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44), E10467–E10475.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Elsevier.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., & Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding with unsupervised learning.

Raghu, A., Raghu, M., Bengio, S., & Vinyals, O. (2019). Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International conference on learning representations*.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-alone self-attention in vision models. In *Neurips*.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental

classifier and representation learning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2001–2010).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. In *Proceedings of the 28th international conference on neural information processing systems-volume 1* (pp. 91–99).

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International conference on learning representations*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., & Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 4974–4983).

Schlag, I., Irie, K., & Schmidhuber, J. (2021). Linear transformers are secretly fast weight memory systems. *arXiv preprint arXiv:2102.11174*.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook* (Unpublished doctoral dissertation). Technische Universität München.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1), 131–139.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., ... others (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792), 706–710.

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 2994–3003).

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh international conference on document analysis and recognition, 2003. proceedings.* (Vol. 3, pp. 958–958).

Tan, H., & Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 5103–5114).

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., & Isola, P. (2020). Rethinking few-shot image classification: A good embedding is all you need? In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer vision – eccv 2020* (pp. 266–282). Cham: Springer International Publishing.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *ArXiv, abs/1607.08022*.

van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1), 1–14.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6000–6010).

Wawrzyński, P. (2009). Real-time reinforcement learning by sequential actor–critics and experience replay. *Neural Networks*, 22(10), 1484–1497.

Wu, Y., Rabe, M., Li, W., Ba, J., Grosse, R., & Szegedy, C. (2021). Lime: Learning inductive bias for primitives of mathematical reasoning. *arXiv preprint arXiv:2101.06223*.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. B. (2018). Neural-symbolic vqa: disentangling reasoning from vision and language understanding. In *Proceedings of the 32nd international conference on neural information processing systems*

(pp. 1039–1050).

Yu, F., Tang, J., Yin, W., Sun, Y., Tian, H., Wu, H., & Wang, H. (2020). Ernie-vil: Knowledge enhanced vision-language representations through scene graph. *arXiv preprint arXiv:2006.16934*.

Yu, Z., Yu, J., Cui, Y., Tao, D., & Tian, Q. (2019). Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6281–6290).

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International conference on machine learning* (pp. 3987–3995).

Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., . . . Gao, J. (2021). Vinvl: Making visual representations matter in vision-language models. *CVPR 2021*.

## **APPENDIX**

## A. CONTINUAL LEARNING REPRESENTATIONS

### A.1. Omniglot random samples used for visualization

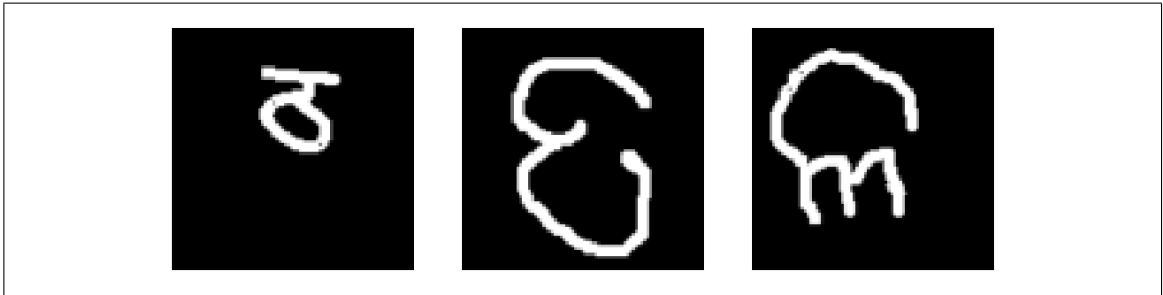


Figure A.1. The three randomly selected samples of the Omniglot dataset that were used for visualization.

A.2. *Single-branch* network representations

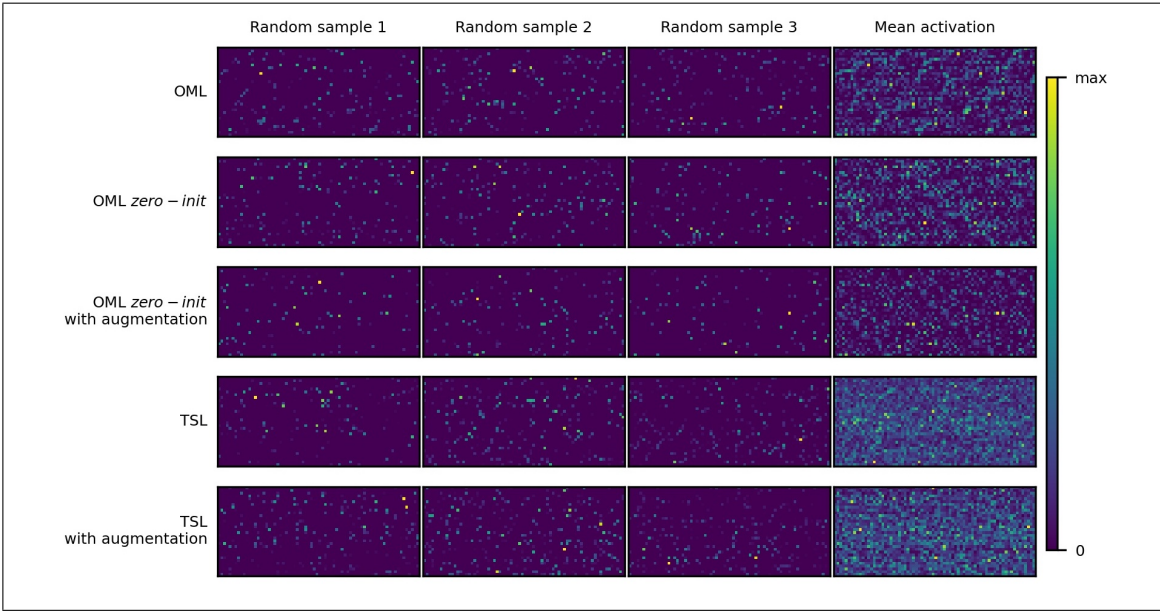


Figure A.2. The representations of the trained *single-branch* network for the second random seed.

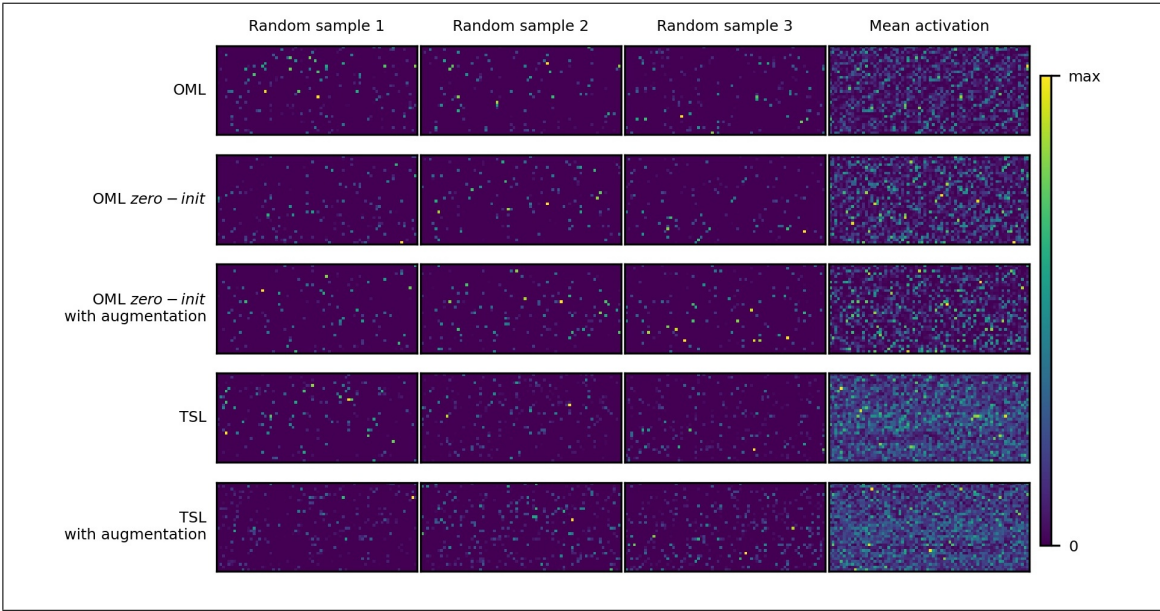


Figure A.3. The representations of the trained *single-branch* network for the third random seed.

A.3. Neuromodulated network representations

A.3.1. Meta-trained models

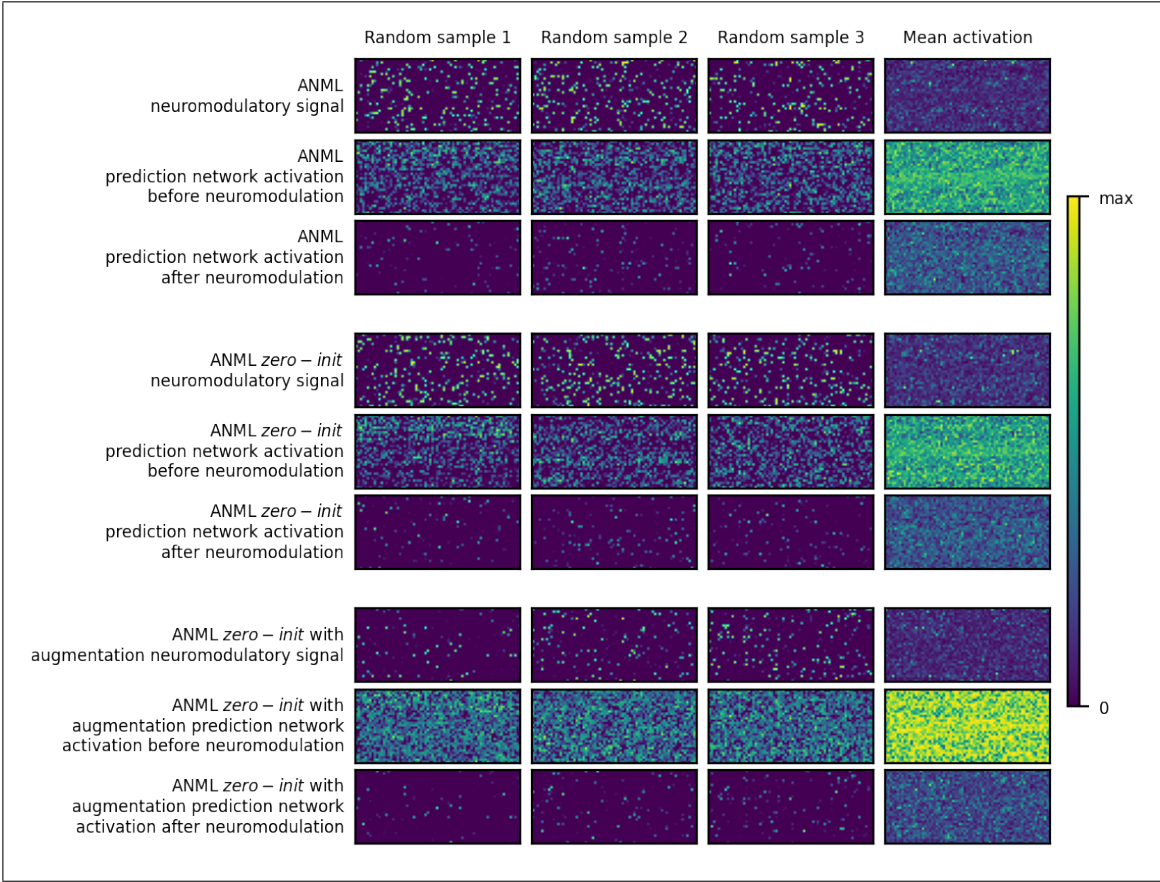


Figure A.4. The representations of the ANML network for the second random seed.



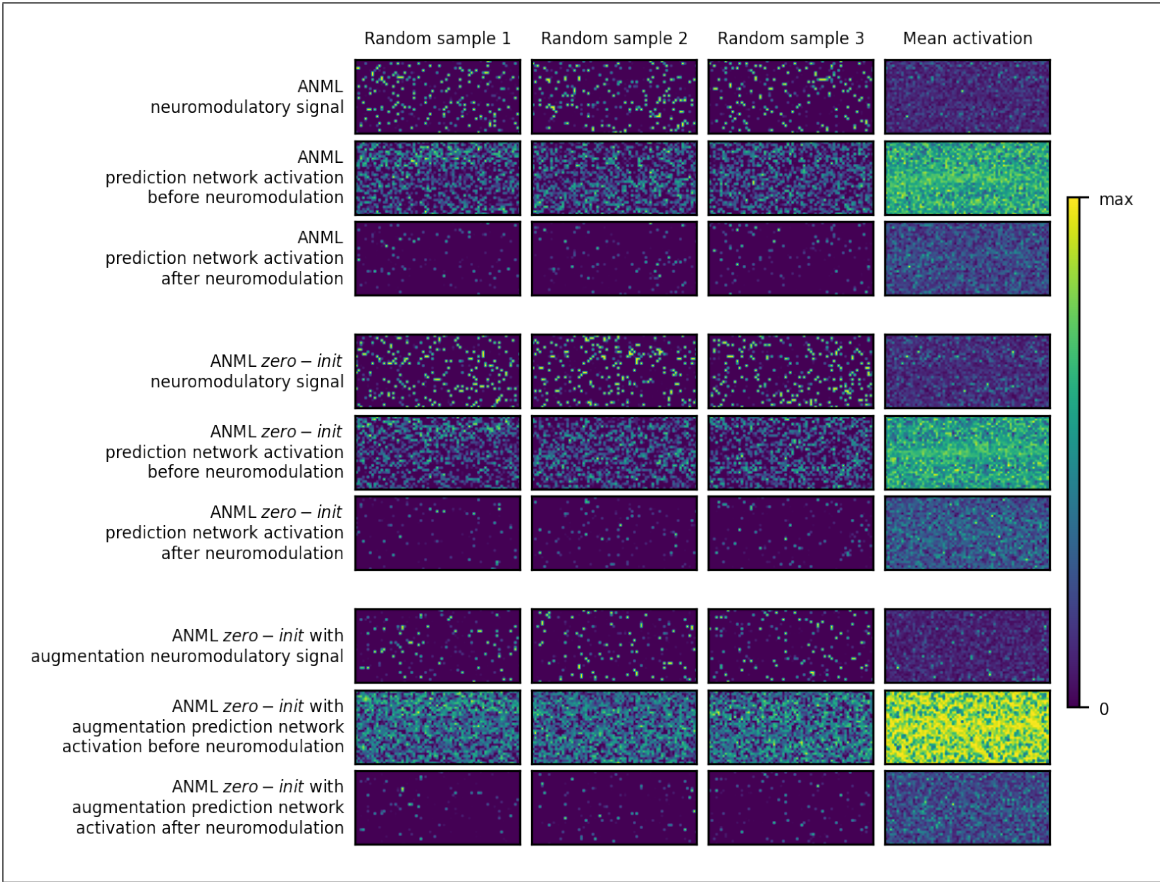


Figure A.5. The representations of the ANML network for the third random seed.

A.3.2. TSL-trained models

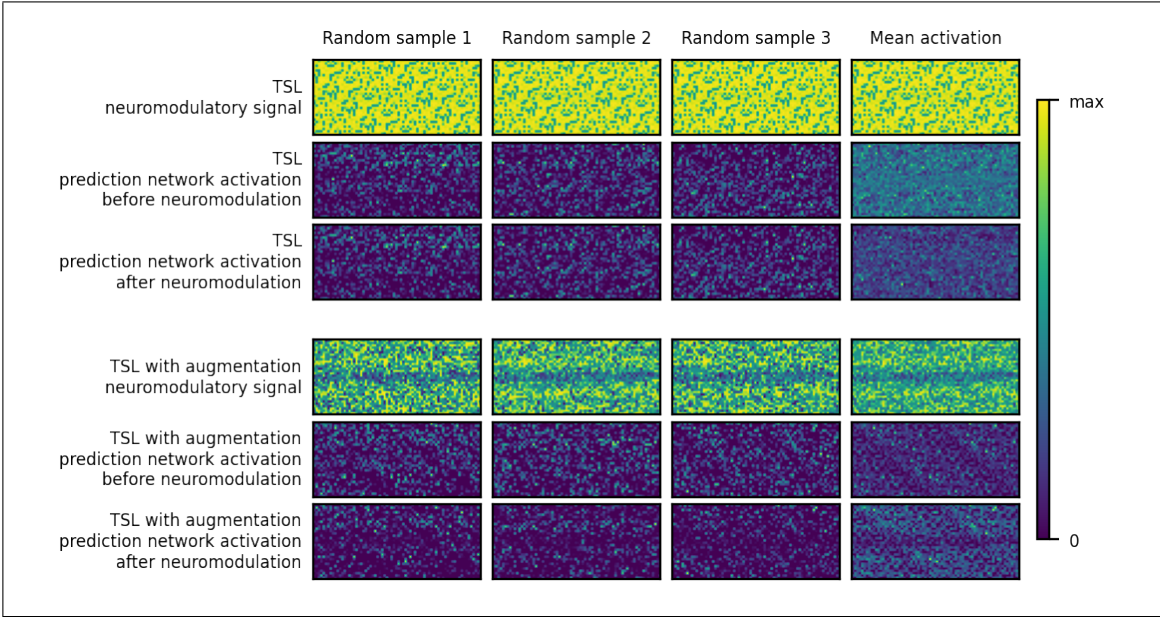


Figure A.6. The representations of the TSL-trained *neuromodulated* network for the second random seed.

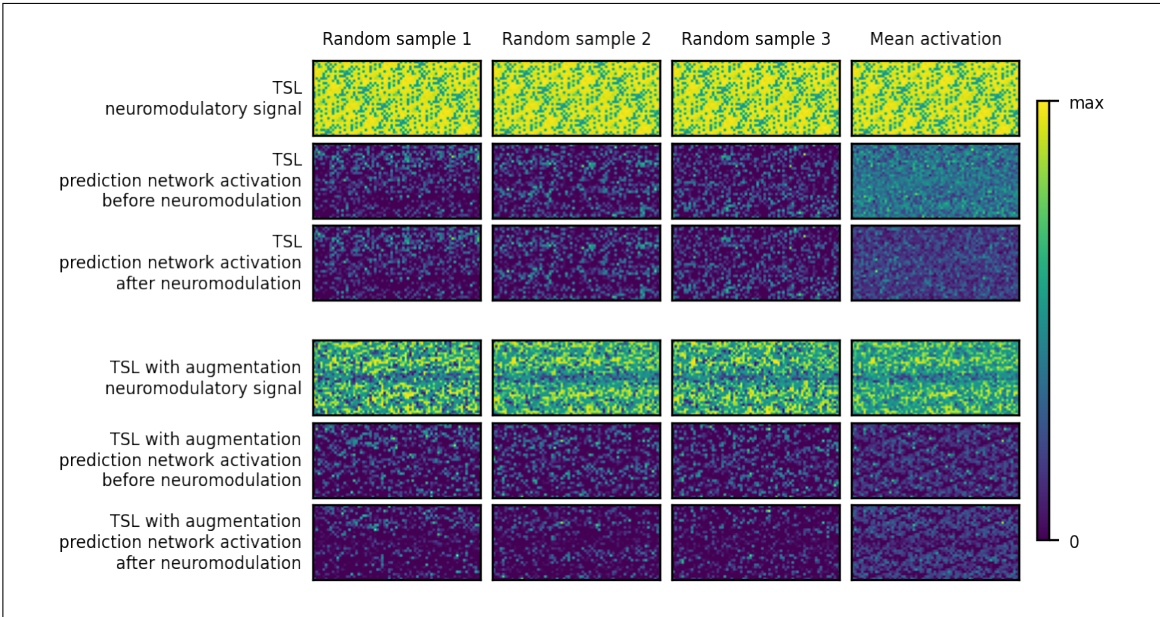


Figure A.7. The representations of the TSL-trained *neuromodulated* network for the third random seed.