University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

Model-based meta-learning in neural networks

Willian Wang

Final Essay

mac 499 — Capstone Project

Supervisor: Prof.ª Dr.ª Nina S.T. Hirata

2024

Resumo

Willian Wang. **Model-based meta-learning in neural networks**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2024.

Este projeto investiga algoritmos de meta-aprendizagem baseados em modelos, examinando seus fundamentos teóricos e complexidades arquiteturais. Nós fornecemos uma visão geral do cenário da meta-aprendizagem, delineando as várias categorias de algoritmos. Nossa análise então se concentra em métodos baseados em modelos, explorando especificamente Processos Neurais Condicionais (CNPs), Redes Neurais Aumentadas por Memória (MANNs), Meta-Aprendizes Atencionais Neurais Simples (SNAILs) e Meta Redes. Para cada método, discutimos as ideias principais e detalhes de implementação, fazendo observações a partir do trabalho existente. Finalmente, exploramos brevemente conceitos mais experimentais, como redes automodificáveis e inteligência coletiva, examinando seu valor potencial para pesquisas futuras em meta-aprendizagem.

Palavras-chave: aprendizagem profunda. redes neurais. meta-aprendizagem.

Abstract

Willian Wang. **Model-based meta-learning in neural networks**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2024.

This project investigates model-based meta-learning algorithms, examining their theoretical underpinnings and architectural complexities. We provide a comprehensive overview of the meta-learning landscape, outlining the various categories of algorithms. Our analysis then focuses on model-based methods, specifically exploring Conditional Neural Processes (CNPs), Memory-Augmented Neural Networks (MANNs), Simple Neural Attentive Meta-Learners (SNAILs), and Meta Networks. For each method, we discuss the core ideas and implementation details, drawing observations from the existing work. Finally, we briefly explore more experimental concepts like self-modifying networks and collective intelligence, examining their potential value for future research in meta-learning.

Keywords: deep learning. neural networks. meta-learning.

Contents

Introduction

Neural networks are typically constructed by first defining their architecture and then training them from scratch using large datasets. This training process relies on backpropagation, where the network gradually adjusts its parameters to become proficient at a specific task. Although this approach has been successful in many applications, it struggles in scenarios where training data is scarce or where the network must quickly adapt to new tasks beyond the initial training dataset. Meta-learning addresses these limitations by distilling knowledge from prior training experiences, aiming to develop networks that demonstrate superior learning performance on future tasks.

The task distribution view of the meta-learning problem can be formalized as follows. Let $\mathcal{T}$ denote a task sampled from a task distribution $p(\mathcal{T})$. Each task $\mathcal{T}$ consists of a dataset $\mathcal{D}$ divided into a training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$. The goal of meta-learning is to find a set of parameters θ that minimize the expected loss over the test set $\mathcal{D}_{\text{test}}$ after training using $\mathcal{D}_{\text{train}}$ for a new task sampled from the task distribution. Formally, the objective can be expressed as:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathcal{L}(\mathcal{D}_{\text{test}}, f_\theta(\mathcal{D}_{\text{train}}))] \tag{1}$$

where f_θ represents the model parameterized by θ that is trained using $\mathcal{D}_{\text{train}}$.

Meta-learning, often referred to as "learning to learn," focuses on developing models that can effectively learn from limited data and rapidly adapt to new tasks. The field generally categorizes approaches into three main categories: metric-based, optimization-based, and model-based techniques.

This study focuses on exploring the core concepts of model-based meta-learning algorithms. While specific benchmarks are not the primary concern, gaining a thorough understanding of the mechanisms, advantages, and limitations of these algorithms is essential. This includes examining how they leverage model architectures to facilitate rapid learning and adaptation.

Chapter 1

Preliminaries

1.1 General structure

A distinctive aspect of meta-learning is the structure of its training process. Instead of training on a single task, the network is exposed to a distribution of tasks. Each task can be viewed as analogous to the input-output data pairs in typical supervised learning, and is therefore divided into training and testing sets. The training set tasks are used to update the network parameters, while the testing set serves to evaluate the network's performance after learning from the training set.

Within each task, the data is further divided into the **support set** and the **query set**. The model learns from the support set and makes predictions on the query set, similar to a typical supervised learning problem. However, the amount of data in each task's support set is usually very small, making the learning process more challenging. This scarcity of data is a defining characteristic of meta-learning, pushing the model to generalize effectively from limited examples.

The training process can be abstractedly separated into two nested loops: the outer loop and the inner loop. The outer loop optimizes the model's parameters across tasks, guiding the general learning direction. The inner loop fine-tunes the model's parameters on individual tasks using the support set. While the outer loop might seem similar to hyperparameter optimization, the process is considerably more complex due to the multi-task nature and nested structure of the problem.

1.2 Categories of meta-learning algorithms

To explore model-based meta-learning, it is essential to understand the broader landscape of meta-learning algorithms. Although these categories are not mutually exclusive, and more detailed taxonomies have been proposed (Hospedales *et al.*, 2022), this division offers enough flexibility to encompass the studied algorithms.

1.2.1 Metric-based approaches

Metric-based approaches, also known as non-parametric algorithms, aim to learn a metric space where the distance between data points correlates with their label similarity. These methods make predictions by directly comparing test data points with training examples in the learned metric space.

Prototypical networks (Snell *et al.*, 2017) are a popular example of metric-based algorithms. The algorithm works very similarly to the k-nearest neighbors (KNN) algorithm, but instead of storing all training examples, it computes a prototype for each class by averaging the embeddings of the support set examples belonging to that class. During inference, the algorithm computes the distance between the query point and the prototypes of each class, assigning the query point to the class with the closest prototype. Figure 1.1 illustrates this process.

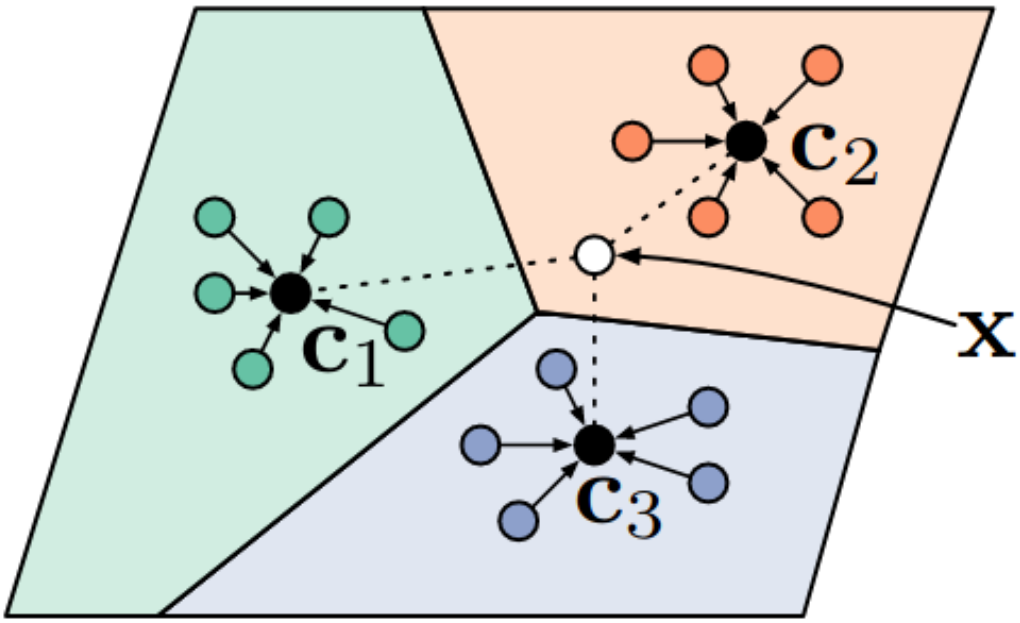

Figure 1.1: *Represention of inference using the prototypical network, a metric-based algorithm that computes the distance between the query point and the prototypes of each class. Image source: Snell et al., 2017.*

A popular set of algorithms often considered outside the scope of meta-learning that have a lot of similarities with metric-based approaches are the contrastive learning algorithms (Radford *et al.*, 2021), used frequently in self-supervised learning with images. The idea of employing data augmentation and comparing augmented versions of the same image to learn a metric space can be seen as a form of metric-based learning, as exemplified by the SimCLR algorithm (Chen *et al.*, 2020).

While these approaches are computationally efficient, they often require domain expertise for effective data augmentation and may struggle in scenarios where the metric

space is not well-defined or when the underlying data manifold is complex. Importantly, some of the ideas from this category, particularly the concept of learning an embedding space, are also present in model-based approaches. This embedding space can be seen as a form of memory representation, which is a common theme in model-based approaches.

1.2.2 Optimization-based approaches

Optimization-based techniques treat the inner-loop update process itself as a learnable function. The goal is to find a set of initial parameters that enable the network to quickly adapt to new tasks with minimal training steps. A prominent example is Model-Agnostic Meta-Learning (MAML) (Finn *et al.*, 2017), which trains networks to find initialization parameters that enable rapid adaptation to new tasks with only a few gradient steps. Figure 1.2 illustrates the MAML algorithm.

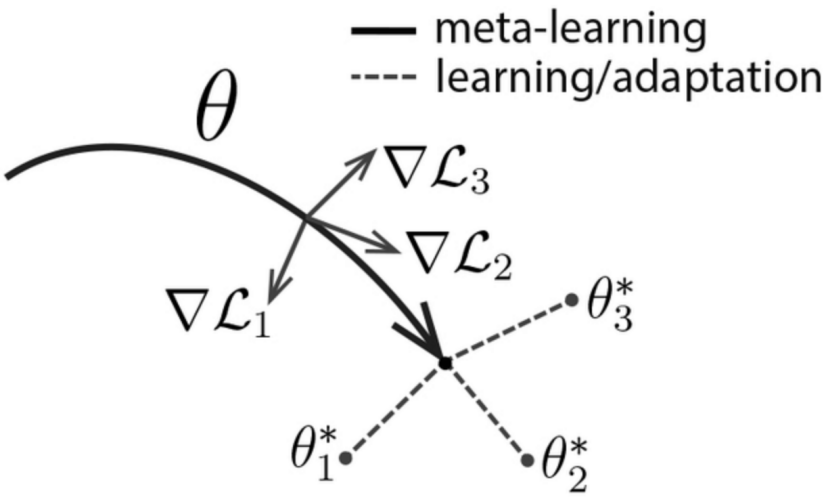

Figure 1.2: *Represention of the model-agnostic meta-learning (MAML) algorithm, optimizing θ so it can be further trained on new tasks with minimal gradient steps. Image source: Finn et al., 2017.*

One of the main advantages of optimization-based approaches is their flexibility to be used in different learning scenarios beyond supervised learning. For instance, MAML has been successfully applied to reinforcement learning too. One of the main limitations of these approaches is the need for second-order optimization, which, although it can be simplified or even avoided, typically results in increased computational cost and complexity.

While these approaches offer considerable flexibility and can be applied to diverse problems, they typically involve complex second-order optimizations and are computationally intensive. Later work, such as Reptile (Nichol *et al.*, 2018), has attempted to decrease the computational cost of these algorithms by employing approximations or simplifications to avoid explicit second-order optimization. However, since this class of algorithms usually does not have a direct intersection with the model-based approaches that are the focus of this study, it will not be covered in further detail.

1.2.3 Model-based approaches

Model-based approaches, also called black-box adaptation algorithms, attempt to directly encode the learning procedure within the model architecture. This often leads to more complex neural network architectures and can sometimes make them less interpretable.

These algorithms are usually easier to classify by their differences from the other two categories than by a common characteristic. However, some of the most common themes in this category are the use of memory modules and networks that are partially or fully parameterized by other networks.

Unlike the other two categories, model-based meta-learning lacks widely adopted "mainstream" techniques. Despite this, it encompasses some of the most radical and innovative ideas in the field. This study aims to review significant algorithms within this category and analyze their underlying mechanisms, advantages, and limitations.

1.3 Dataset

A widely adopted benchmark for meta-learning research is the Omniglot dataset (Lake *et al.*, 2015), comprising 1623 handwritten characters from 50 different alphabets. This dataset boasts a large number of alphabets, each containing a diverse set of unique characters. Each character within each alphabet is represented by 20 handwritten samples, reflecting the inherent variability in handwriting styles. Commonly, the dataset is used for few-shot classification tasks, where the model is presented with 'N-way K-shot' problems. Here, 'N-way' refers to the number of classes and 'K-shot' refers to the number of examples per class in the support set. Figure 1.3 illustrates a 3-way 2-shot classification task using the Omniglot dataset.

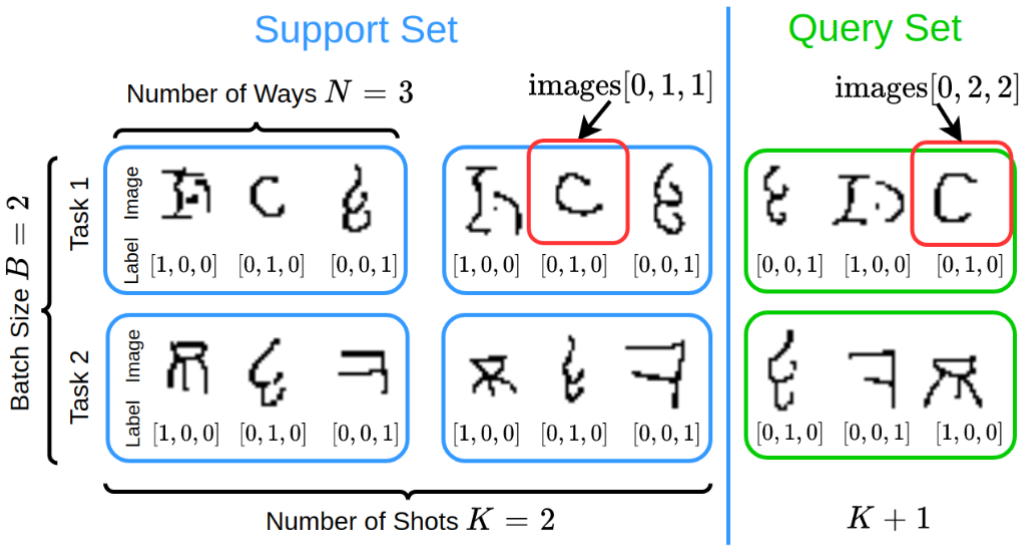

Figure 1.3: *Example of handwritten characters from the Omniglot dataset using in a 3-way 2-shot classification task. Image source:* Finn, 2023.

In a typical meta-learning experiment, the dataset is split such that a model is meta-trained on a subset of alphabets and then evaluated on a completely disjoint set of alphabets. This tests the model's ability to generalize and learn a common structure across different alphabets and handwriting styles. The Omniglot dataset is particularly well-suited for this purpose due to the necessity of generalizing across diverse characters and alphabets instead of relying on numerous examples of the same character.

Chapter 2

Conditional Neural Processes

One of the simplest ideas to handle few-shot learning in classification problems is to use a feedforward network that takes as input all the examples from the support set along with the input query and outputs the prediction directly. Although this approach is simple, it may be improved with the use of a more sophisticated architecture.

Garnelo *et al.*, 2018 introduces Conditional Neural Processes (CNPs), which utilize concepts that resemble the idea of prototypical networks from metric-based approaches. The core idea is to generate an embedding for each support set that represents the task context and then use this embedding to improve the prediction of the query set. The naming makes allusion to the fact that the network used for inference is conditioned on the data from the support set, which makes sense especially because the paper not only looks at the classification and regression problems but also at the generative side with the image completion task.

2.1 Algorithm

The first step is to generate an embedding for each example in the support set using the same encoder network h using both the input and output data. Then, the network computes another embedding from these individual embeddings by using a permutation-invariant function a. This function, which in this case was a simple mean of all the support set embedding vectors, is then used as additional input to another network g that generates the final prediction for the query set. The architecture is illustrated in Figure 2.1.

The training process for CNPs is straightforward and similar to the training of ordinary supervised learning problems. Tasks are sampled from training data and the network predicts all the query set values given the support set. The loss function is the negative log-likelihood of the target values in the query set given the predicted values.

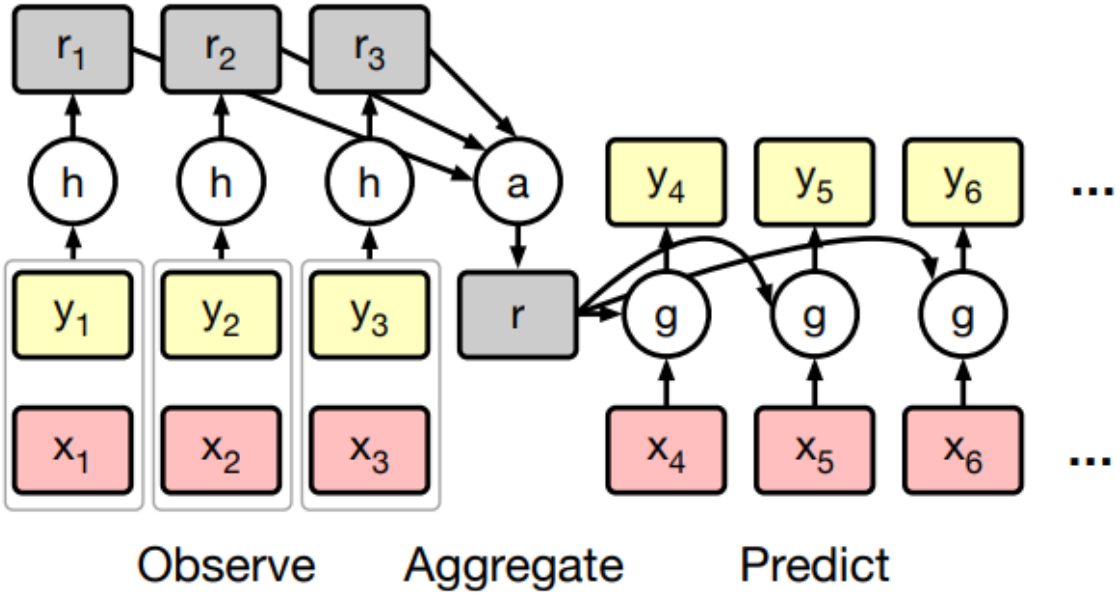

Figure 2.1: *Represention of inference using CNP. It's possible to view this example as 3-shot learning, predicting the value of three or more inputs from the query set. Image source: Garnelo et al., 2018.*

2.2 Results

The results presented in the paper demonstrate that Conditional Neural Processes (CNPs) achieve competitive performance compared to other contemporary meta-learning algorithms, such as Memory-Augmented Neural Networks (MANNs), which will be discussed in the next chapter. Notably, CNPs attained 98.1% accuracy on the 5-way 1-shot Omniglot dataset.

However, the more significant takeaway is the flexibility of the CNP's core concept: using a context embedding as an additional input. This approach can be readily adapted to more complex neural network architectures, opening up possibilities for solving a wider range of problems.

Chapter 3

Memory-Augmented Neural Networks

Unlike traditional Recurrent Neural Networks (RNNs), such as Long Short-Term Memory networks (LSTMs), Memory-Augmented Neural Networks (MANNs) feature explicit external memory modules. These modules can be selectively read from and written to, allowing the network to learn how to store and retrieve information effectively. The seminal work by Santoro *et al.*, 2016 outlines the application of MANNs to one-shot learning problems, demonstrating their ability to rapidly learn new concepts from limited examples. This chapter provides an overview of the MANN architecture, focusing on the memory, the controller, and the training process.

3.1 Algorithm

The core architecture of a MANN comprises a recurrent neural network (RNN) that dynamically interacts with an external memory module. This interaction is mediated by a memory controller, which acts as the intermediary, orchestrating read and write operations to the memory. The controller itself can be implemented as a feedforward neural network, an LSTM, or even another type of RNN. Its primary role is to process incoming input data and generate read and write operations to the memory module. While the specific architectural details of the controller are left flexible in the original paper, the behavior of the memory module is well defined.

The addressing mechanism, which dictates how memory locations are accessed, operates as follows: the controller generates a key vector $\mathbf{k}_t$ from the input data x_t, which is used to compute the cosine similarity $K(\mathbf{k}_t, \mathbf{M}_t(i))$ between the key and each row $\mathbf{M}_t(i)$ of the matrix representing the memory.

For the read operation, the controller generates a read weight vector $\mathbf{w}_t^r$ by applying a softmax function to the cosine similarities. The read vector $\mathbf{r}_t$ is then computed as the weighted sum of the memory rows, where the weights are given by the read weight vector:

$$K(\mathbf{k}_t, \mathbf{M}_t(i)) = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\|\mathbf{k}_t\| \cdot \|\mathbf{M}_t(i)\|} \tag{3.1}$$

$$w_t^r(i) = \text{softmax}(K(\mathbf{k}_t, \mathbf{M}_t(i))) \tag{3.2}$$

$$\mathbf{r}_t = \sum_{i=1}^{N} w_t^r(i)\mathbf{M}_t(i) \tag{3.3}$$

The write operation is designed to update the memory content and follows the read operation. To manage memory updates intelligently, an mechanism called Least Recently Used Access (LRUA) is employed. The LRUA mechanism aims to either write to the least recently accessed memory location, effectively replacing outdated information, or to the most recently used location, potentially updating or refining existing information that might not be accessed again soon.

To implement this mechanism, the controller keeps track of a "usage vector" $\mathbf{w}_t^u$, which represents the usage of each row in the memory. This vector takes into account the read and write operations from previous time steps by decaying their weights and is updated at each time step.

The write vector $\mathbf{w}_t^w$ is computed as a convex combination of the least used row and the last used row $\mathbf{w}_t^{lu}$, which is a binary vector that indicates the least used rows according to the values of $\mathbf{w}_t^u$ and a fixed parameter n that will define number of entries with value 1 at $\mathbf{w}_t^{lu}$ according to $m(\mathbf{w}_t^u, n)$, which is the n-th smallest value of $\mathbf{w}_t^u$.

$$\mathbf{w}_t^u = \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w \tag{3.4}$$

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n) \end{cases} \tag{3.5}$$

$$\mathbf{w}_t^w = \sigma(\alpha)\mathbf{w}_{t-1}^r + (1 - \sigma(\alpha))\mathbf{w}_{t-1}^{lu} \tag{3.6}$$

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i) + w_t^w(i)\mathbf{k}_t, \forall i \tag{3.7}$$

For the meta-learning using a recurrent network, it's necessary to sequentially processes input pairs (x_t, y_{t-1}) at each time step t, generating a prediction $\hat{y}_t$ as output. Each x_t represents an input sample at the current time step, while y_{t-1} is the corresponding label from the previous time step. This sequential processing allows to predict the label of the current input and immediately receive the true label to compare with the prediction in the next time step. Because the network is a recurrent architecture, one important point of care is to ensure that the input pairs and labels are shuffled at the beginning of each task to prevent the network from learning the order of the input pairs. This process is illustrated in Figure 3.1.

For implementation in an n-shot learning scenario, the input sequence is structured into $n+1$ distinct pairs. The initial pair is constructed as $(x_1^{\text{support}}, 0)$, where x_1^{support} signifies the first support set example and the 0 label serves as an initial placeholder. Subsequent

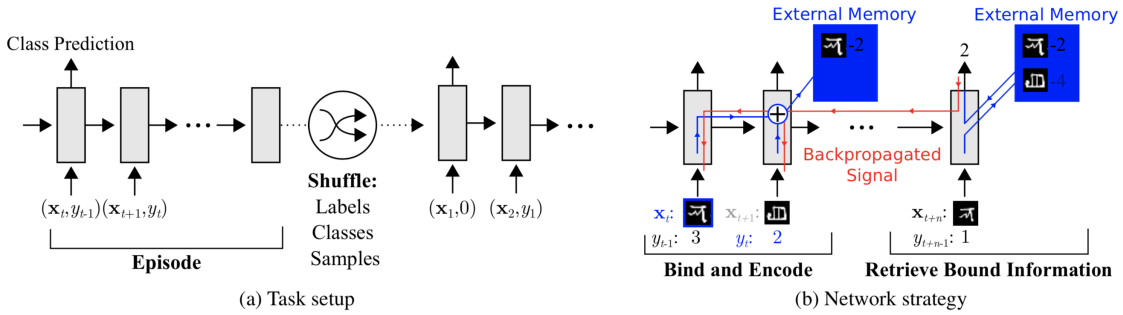

Figure 3.1: *The image (a) represents the process of shuffling the input order to avoid unwanted bias. The image (b) represents the interaction of the recurrent network with the external memory module. Image source: Santoro et al., 2016.*

pairs are formed using subsequent support samples and their corresponding ground-truth labels from the support set. The final pair in this sequence is given by $(x^{\text{query}}, y_n^{\text{support}})$, where x^{query} denotes the query example to be classified, and y_n^{support} is the label of the last example in the support set. Finally, the network's final output, $\hat{y}^{\text{query}}$, represents the predicted label for the query example and is the only output used for training, even though the network generates predictions for all query examples.

The described process is repeated for each task in a batch of tasks and trained using a loss function that compares the predicted label $\hat{y}^{\text{query}}$ with the true label y^{query} and backpropagates the error through the network. The memory itself is always zeroed at the beginning of each task, ensuring a memory controller that is agnostic to the task at hand. There are no major differences to a common supervised learning problem in the outer loop of the training process.

3.2 Results

Although the results of the original MANN paper were impressive compared to previous techniques like the simpler LSTM and kNN, the architecture has since been surpassed by most other algorithms mentioned in this work. However, the MANN architecture served as a crucial baseline for subsequent work in the field. Many later meta-learning methods, such as MAML and Prototypical Networks, used MANNs as a benchmark for comparison, demonstrating the impact of the work in establishing a standard for few-shot learning performance.

One particularly interesting later development is the popularization of the Transformer architecture, which, when used to train very large models with massive datasets and compute resources, has demonstrated impressive few-shot learning abilities (Brown *et al.*, 2020). This capability might be partly attributed to the inherent associative nature of their attention mechanism, which allows them to quickly relate new inputs to relevant information stored implicitly within their weights during pre-training (Nichani *et al.*, 2024; Ramsauer *et al.*, 2021). Although the Transformer architecture is not explicitly designed for memory management like MANNs, it still shows the importance of memory mechanisms in few-shot learning tasks.

Chapter 4

Simple Neural Attentive Meta-Learner

4.1 Algorithm

The Simple Neural Attentive Meta-Learner (SNAIL), introduced by Mishra *et al.*, 2017, occupies a middle ground between the recurrent nature of Memory-Augmented Neural Networks (MANNs) from Chapter 3 and the fixed-size embeddings of Conditional Neural Processes (CNPs) from Chapter 2. SNAIL strategically combines convolutional layers with soft attention mechanisms, creating a flexible network adaptable to a wide range of tasks.

SNAIL applies convolution across the temporal dimension of the input data—the sequence of (x_t, y_t) pairs from the support set. This approach is reminiscent of MANNs but eliminates the need for a one-step time shift. In MANNs, the time shift is necessary to prevent the network from "peeking" at future labels during training. Since SNAIL is not recurrent and does not predict at each step, this is unnecessary. A key argument against recurrent architectures in this context is their limitation on information flow between steps. SNAIL employs Temporal Convolution (TC), first introduced in Oord *et al.*, 2016. TC is a type of causal convolution, meaning it only considers past information, as illustrated in Figure 4.1. Furthermore, TC leverages exponential dilation rates to expand the receptive field, allowing it to capture longer-range dependencies.

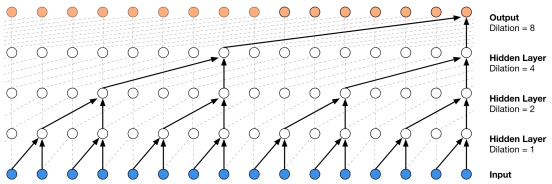

Figure 4.1: *Visualization of a stack of dilated causal convolutional layers. Image source: Oord et al., 2016.*

Although the number of causal convolutional layers needed to encompass the entire sequence grows logarithmically ($O(\log n)$) with exponential dilation, the network may

still struggle with exceptionally long sequences. Here, the soft attention mechanism, as introduced by Vaswani *et al.*, 2017, offers a valuable solution. Attention excels at capturing relationships between samples in the support set. However, this comes at the cost of increased computational complexity, scaling quadratically with the sequence length. Furthermore, attention layers inherently lack positional dependence. While this characteristic can be advantageous in scenarios like reinforcement learning, where the order of input data is important, it can be a limitation in other applications, similar to the requirement of shuffling input pairs in MANNs. The combination of TC layers and attention layers in SNAIL can be seen in Figure 4.2.

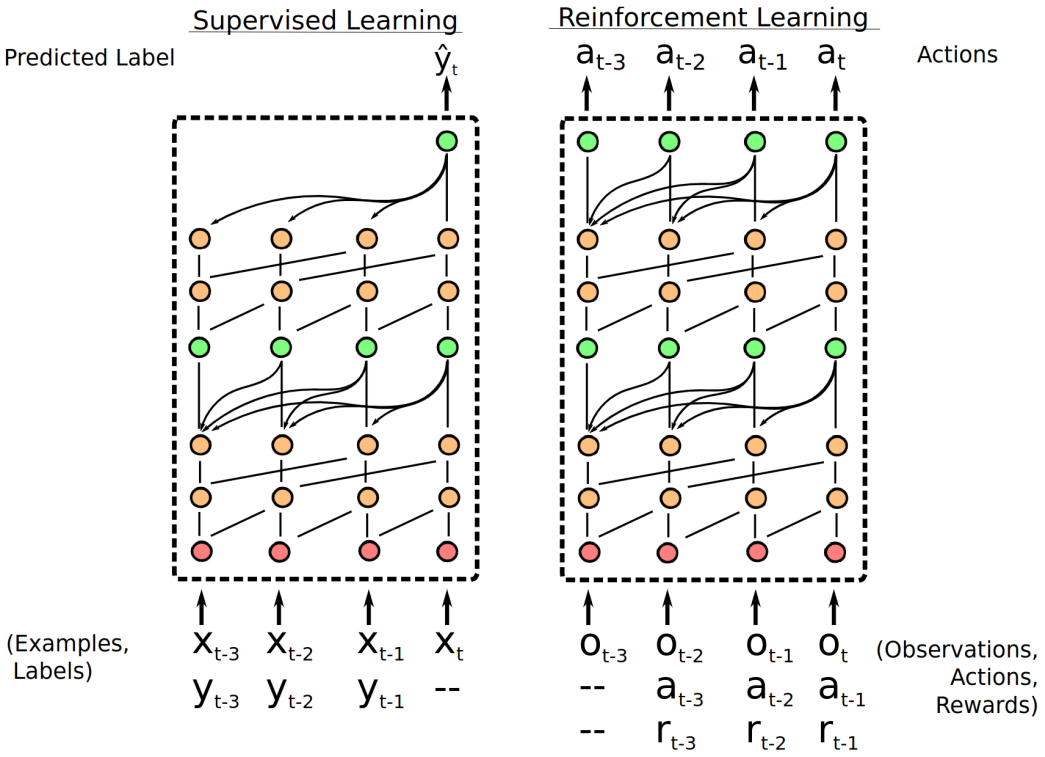

Figure 4.2: *Visualization of SNAIL used for both supervised and reinforcement learning. Blocks of TC layers (orange) are followed by attention layers(green). Image source: Mishra et al., 2017.*

4.2 Results

In 5-way 1-shot classification tasks, SNAIL outperforms MANNs significantly (99.07% vs. 82.8%) and remains competitive with optimization-based methods like MAML (98.7%). This demonstrates that carefully designed architectural modifications can lead to substantial performance improvements, without relying on complex optimization techniques or metric learning.

Chapter 5

Meta network

Training conventional neural networks is often bottlenecked by gradient descent, an iterative optimization algorithm that fundamentally requires numerous passes through the entire dataset to converge, even with variants like stochastic gradient descent. Meta Networks (MetaNets), introduced by Munkhdalai and Yu, 2017, offer a novel approach to rapidly adapt network parameters to new tasks. MetaNets leverage concepts similar to those explored in previous chapters, such as memory and embedding generation, to achieve this rapid adaptation.

5.1 Algorithm

MetaNets consist of two main components: the *base-learner* and the *meta-learner*. The base-learner is the network that performs the final task prediction, while the meta-learner generates task-specific parameters (fast weights) for the base-learner. Each network has two sets of weights in its feedforward layers: *slow weights* and *fast weights*. Slow weights are updated using standard gradient descent, while fast weights are dynamically generated by another network based on the slow weight's loss gradients. These two sets can be run independently, as shown in Figure 5.1, and the fast weights are generated using the gradient of the loss function with respect to only the slow weights of the same network, as will be explained later.

The first step is to train the meta-learner. The meta-learner, denoted as f, has slow weights θ and fast weights θ^+. Initially, only the slow weights are used to train an embedding of the inputs, minimizing the cross-entropy loss. During each training step t, the loss gradients with respect to the slow weights, $\nabla_\theta L_t$, are fed into an LSTM, denoted as F. The input dimension of this LSTM matches the number of parameters in f. The fast weights θ^+ are generated by the LSTM at the end of the training process and are not used during training.

After training the meta-learner, the base-learner, denoted as g, is trained. It also has slow weights ϕ and fast weights ϕ^+. Similar to the meta-learner, the base-learner is initially trained using only its slow weights. However, instead of an LSTM, the loss gradients $\nabla_\phi L_t$ are directly input into a feedforward neural network G to generate fast weights that are

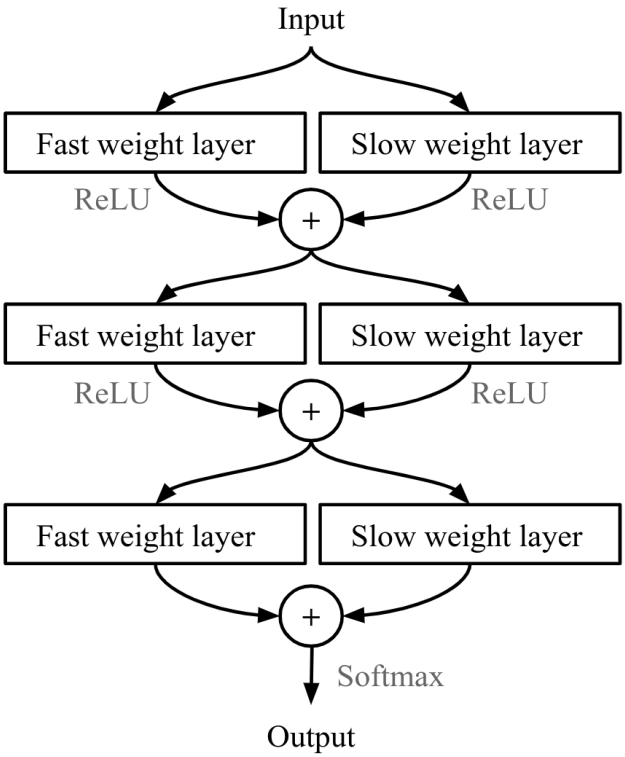

Figure 5.1: *Closer look at how the fast and slow weights are not dependant of each other. The results from fast weight layers are simply summed to the slow weight layer results. Image source: Munkhdalai and Yu, 2017.*

specific to the input data: $\phi_t^+ = G(\nabla_\phi L_t)$. These generated fast weights are then stored in the memory, using as keys the embedding generated by the recently updated meta-learner $f_\theta(x)$, since the fast weights are not used during the training process.

Finally, after training both networks on the support set, during the query phase, the base-learner retrieves the necessary fast weights from memory on the fly. For each input, the embedding function $f_{\theta,\theta^+}(x)$ is used as a query to the memory, employing cosine similarity as in MANNs. The final prediction is then made using $g_{\phi,\phi^+}(x)$, where the fast weights ϕ^+ are those retrieved from memory. An overview of the process is illustrated in Figure 5.2.

5.2 Results

Despite the architectural complexity and the need for second-order derivatives to train the fast weight generating networks, MetaNets achieved state-of-the-art results on the one-shot 5-way classification task using the Omniglot dataset, as previously described. They reported an accuracy of 98.95%.

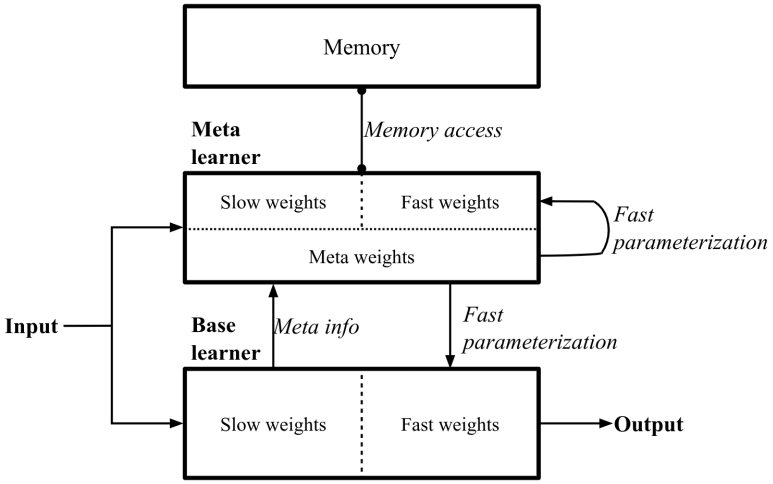

Figure 5.2: *Visualization of the communication between different components of the Metanet. Image source: Munkhdalai and Yu, 2017.*

Chapter 6

Other Algorithms

This chapter explores research in related fields like lifelong learning, continual learning, and biologically inspired systems. While these works may not explicitly address meta-learning, they share conceptual overlaps with the algorithms discussed in previous chapters. Specifically, many draw inspiration from neuroplasticity and Hebbian learning, offering potentially valuable insights for enhancing multitask learning through a meta-learning lens. Although direct comparisons to established meta-learning algorithms are often absent, the core principles presented in these papers could serve as inspiration for future research in the field.

6.1 Networks that modify networks

The concept of fast weights was not new from Metanets (chapter 5), and it was actually explored a few times decades ago (Jurgen Schmidhuber, 1987; Hinton and Plaut, 1987). More recently, they are best known for their use in Hypernetworks (Ha, Dai, *et al.*, 2016) and Neural Architecture Search (Ren *et al.*, 2021), both of which are focused on generating neural networks that are more efficient for a specific task and have their own field of research. On the other side, there are works that focus on the idea of having networks that can modify themselves during the training process, one of them being self-referential neural networks.

Self-referential networks (Kirsch and Jürgen Schmidhuber, 2022) focus on eliminating the need to train the meta-learning process itself (sort of learning to learn how to learn) and beyond, by handling the task to the network itself. The idea is to adopt an evolutionary-like algorithm where the modification is stochastic, and the networks with the highest fitness scores are selected for the next generation. The authors also discuss the relationship with memory-based meta-learning, arguing that because the memory influences the future updates to the network itself, the network can be seen as a self-referential system.

6.2 Collective Intelligence

Ha and Tang, 2022 reviews multiple areas of research in deep learning that can incorporate concepts of self-organization, emergent behavior, swarm optimization, and cellular automata to improve the models' ability to handle the issues from each of these areas. One of the mentioned areas is meta-learning, citing some recent works that explore the idea of having neural networks composed of multiple agents that learn how to interact with each other in such a way that the network exhibits emergent behaviors that can mimic the learning process of a typical neural network or even improve it.

One of these works is Sandler *et al.*, 2021, which introduces a kind of generalized neural network by representing each neuron and synapse as multiple parameters. By doing so, the typical neural network can be viewed as a special case where the data passed between each neuron is a two-dimensional vector, one for the activation and the other for the loss gradient. The interaction rules between the neurons are governed by a low-dimensional genome, which is trained using evolutionary algorithms.

Another work mentioned is Variable Shared Meta-Learning (VSML) (Kirsch and Jürgen Schmidhuber, 2021). The authors argue that one bottleneck of current recurrent neural networks is how the number of parameters of the RNN is on the order of $O(n^2)$, while the hidden state/activations are on the order of $O(n)$, which makes the network prone to overfitting. This complements the argument made with SNAIL (chapter 4) to drop a recurrent architecture in favor of a convolutional/attention-based one. By using multiple smart manipulations with the weight matrices to increment their dimensions while keeping the activations on the same order of magnitude, the authors show a few possible perspectives to the modified architecture, one of them being a feed-forward network where every neuron of a layer is represented by the exact same RNN. Figure 6.1 shows a visualization of the inference and training process of VSML.

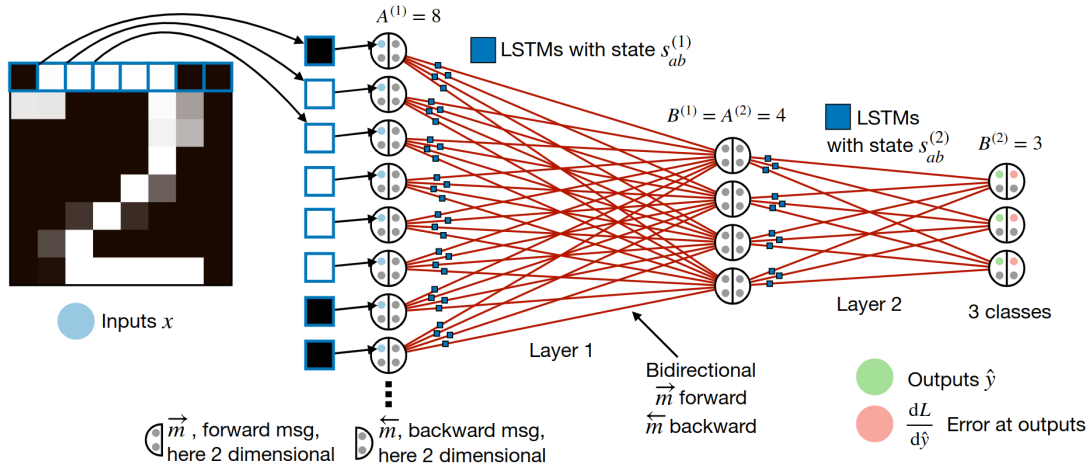

Figure 6.1: *Visualization of the inference and training process of VSML. Image source: Kirsch and Jürgen Schmidhuber, 2021.*

Because these techniques offer very flexible bidirectional communication between the neurons, backpropagation can be viewed as just a special case of the forward pass, where

the loss is propagated from the output layer to the input layer. By breaking free from global update rules from backpropagation, the network can explore different local update rules for the neurons. This exploration of local update rules for neurons and synapses is a direct consequence of the idea of Hebbian learning.

Both Sandler *et al.*, 2021 and Kirsch and Jürgen Schmidhuber, 2021 actually show results indicating that their architectures are able to be trained for MNIST classification but converging faster than a typical feedforward network. These results suggest that alternative, biologically inspired architectures might offer advantages for certain learning tasks and provide inspiration for developing more efficient meta-learning algorithms.

Chapter 7

Conclusion

This work explored model-based meta-learning algorithms, emphasizing their conceptual frameworks and trade-offs. While these approaches often exhibit greater architectural complexity compared to metric-based or optimization-based counterparts—manifested through multiple specialized modules like memory networks or attention mechanisms—this complexity does not inherently preclude strong performance. The analysis of Meta Networks demonstrated that sophisticated designs can effectively balance expressiveness with empirical results, albeit at increased computational costs. Conversely, simpler architectures like Conditional Neural Processes (CNPs) achieved notable efficiency without sacrificing capabilities. This suggests that complexity should be contextually justified rather than universally avoided.

Several algorithms revealed how revisiting classical concepts through modern lenses can yield innovation. For instance, SNAIL's replacement of recurrent layers with causal convolutions and attention mechanisms illustrates how hybridizing past techniques with contemporary components can address traditional limitations. Furthermore, the work briefly surveyed more radical ideas like self-modifying networks and networks that can learn to backpropagate. While computationally expensive, these ideas may still offer insights as experimental frameworks to aid in the research of new concepts.

References

[Brown *et al.* 2020] Tom B. Brown *et al.* "Language models are few-shot learners". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546 (cit. on p. 11).

[Chen *et al.* 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020 (cit. on p. 3).

[Finn 2023] Chelsea Finn. *CS 330: Deep Multi-Task and Meta Learning, Fall 2023*. Stanford University. Available at https://cs330.stanford.edu/. 2023 (cit. on p. 5).

[Finn *et al.* 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1126–1135 (cit. on p. 4).

[Garnelo *et al.* 2018] Marta Garnelo *et al.* "Conditional neural processes". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1704–1713. URL: https://proceedings.mlr.press/v80/garnelo18a.html (cit. on pp. 7, 8).

[Ha, Dai, *et al.* 2016] David Ha, Andrew Dai, and Quoc V Le. "Hypernetworks". *arXiv preprint arXiv:1609.09106* (2016) (cit. on p. 17).

[Ha and Tang 2022] David Ha and Yujin Tang. "Collective intelligence for deep learning: a survey of recent developments". *Collective Intelligence* 1.1 (Sept. 2022). DOI: 10.1177/26339137221114874. URL: https://doi.org/10.1177/26339137221114874 (cit. on p. 18).

[Hinton and Plaut 1987] Geoffrey E Hinton and David C Plaut. "Using fast weights to deblur old memories". In: *Proceedings of the ninth annual conference of the Cognitive Science Society*. 1987, pp. 177–186 (cit. on p. 17).

[Hospedales *et al.* 2022] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. "Meta-learning in neural networks: a survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (2022), pp. 5149–5169. doi: 10.1109/TPAMI.2021.3079209 (cit. on p. 2).

[Kirsch and Jürgen Schmidhuber 2021] Louis Kirsch and Jürgen Schmidhuber. "Meta learning backpropagation and improving it". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 14122–14134. url: https://proceedings.neurips.cc/paper_files/paper/2021/file/7608de7a475c0c878f60960d72a92654-Paper.pdf (cit. on pp. 18, 19).

[Kirsch and Jürgen Schmidhuber 2022] Louis Kirsch and Jürgen Schmidhuber. "Self-referential meta learning". In: *First Conference on Automated Machine Learning (Late-Breaking Workshop)*. 2022. url: https://openreview.net/forum?id=WAcLlCixQP7 (cit. on p. 17).

[Lake *et al.* 2015] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". *Science* 350.6266 (2015), pp. 1332–1338. doi: 10.1126/science.aab3050. eprint: https://www.science.org/doi/pdf/10.1126/science.aab3050. url: https://www.science.org/doi/abs/10.1126/science.aab3050 (cit. on p. 5).

[Mishra *et al.* 2017] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and P. Abbeel. "A simple neural attentive meta-learner". In: *International Conference on Learning Representations*. 2017. url: https://api.semanticscholar.org/CorpusID:3503426 (cit. on pp. 12, 13).

[Munkhdalai and Yu 2017] Tsendsuren Munkhdalai and Hong Yu. "Meta networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2554–2563. url: https://proceedings.mlr.press/v70/munkhdalai17a.html (cit. on pp. 14–16).

[Nichani *et al.* 2024] Eshaan Nichani, Jason D. Lee, and Alberto Bietti. "Understanding factual recall in transformers via associative memories". In: *NeurIPS 2024 Workshop on Mathematics of Modern Machine Learning*. 2024. url: https://openreview.net/forum?id=PtYojIoW0u (cit. on p. 11).

[Nichol *et al.* 2018] Alex Nichol, Joshua Achiam, and John Schulman. *On First-Order Meta-Learning Algorithms*. 2018. arXiv: 1803.02999 [cs.LG]. url: https://arxiv.org/abs/1803.02999 (cit. on p. 4).

[Oord *et al.* 2016] Aaron van den Oord *et al.* *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD]. url: https://arxiv.org/abs/1609.03499 (cit. on p. 12).

[Radford *et al.* 2021] Alec Radford *et al.* *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020 (cit. on p. 3).

[Ramsauer *et al.* 2021] Hubert Ramsauer *et al.* "Hopfield networks is all you need". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=tL89RnzIiCd (cit. on p. 11).

[Ren *et al.* 2021] Pengzhen Ren *et al.* "A comprehensive survey of neural architecture search: challenges and solutions". *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–34 (cit. on p. 17).

[Sandler *et al.* 2021] Mark Sandler *et al.* "Meta-learning bidirectional update rules". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9288–9300. URL: https://proceedings.mlr.press/v139/sandler21a.html (cit. on pp. 18, 19).

[Santoro *et al.* 2016] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. "Meta-learning with memory-augmented neural networks". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1842–1850 (cit. on pp. 9, 11).

[Jurgen Schmidhuber 1987] Jurgen Schmidhuber. "Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook". Diploma Thesis. Technische Universitat Munchen, Germany, 14 May 1987. URL: http://www.idsia.ch/~juergen/diploma.html (cit. on p. 17).

[Snell *et al.* 2017] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4080–4090. ISBN: 9781510860964 (cit. on p. 3).

[Vaswani *et al.* 2017] Ashish Vaswani *et al.* "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964 (cit. on p. 13).

[Weng 2018] Lilian Weng. *Meta-Learning: Learning to Learn Fast*. Nov. 2018. URL: https://lilianweng.github.io/posts/2018-11-30-meta-learning/.