Few-Shot and Continual Learning with Attentive Independent Mechanisms

Eugene Lee Cheng-Han Huang Chen-Yi Lee
Institute of Electronics, National Chiao Tung University
Hsinchu, Taiwan
{eugenelet.ee06g, huang50213.ee04}@nctu.edu.tw cylee@si2lab.org

Abstract

Deep neural networks (DNNs) are known to perform well when deployed to test distributions that shares high similarity with the training distribution. Feeding DNNs with new data sequentially that were unseen in the training distribution has two major challenges — fast adaptation to new tasks and catastrophic forgetting of old tasks. Such difficulties paved way for the on-going research on few-shot learning and continual learning. To tackle these problems, we introduce Attentive Independent Mechanisms (AIM). We incorporate the idea of learning using fast and slow weights in conjunction with the decoupling of the feature extraction and higher-order conceptual learning of a DNN. AIM is designed for higher-order conceptual learning, modeled by a mixture of experts that compete to learn independent concepts to solve a new task. AIM is a modular component that can be inserted into existing deep learning frameworks. We demonstrate its capability for few-shot learning by adding it to SIB and trained on MiniImageNet and CIFAR-FS, showing significant improvement. AIM is also applied to ANML and OML trained on Omniglot, CIFAR-100 and MiniImageNet to demonstrate its capability in continual learning. Code made publicly available at `https://github.com/huang50213/AIM-Fewshot-Continual`.

1. Introduction

Humans have the ability to learn new concepts continually while retaining previously learned concepts [11]. While learning new concepts, prior concepts that were learned are leveraged to form new connections in the brain [4, 52]. The plasticity of the human brain plays an important role on the forming of novel neuronal connections for learning new concepts. Current deep learning methods are inefficient in remembering old concepts after being fed with new concepts, also widely know as catastrophic forgetting [34, 23]. Deep neural networks (DNNs) trained in an end-to-end fashion also has difficulty in learning new tasks in a sample efficient manner [12]. It is conjectured that the cause of catastrophic forgetting and inefficiency in learning new tasks is from the stability-plasticity dilemma [2]. Stability is required so that previously learned information can be retained through the limitation of abrupt weight changes. Plasticity on the other hand encourages large weight changes, resulting in the fast acquisition of new concepts with the trade-off of forgetting old concepts.

It is believed that by scaling up the currently available architecture, DNNs are able to generalize better [7, 41, 10]. Tremendous effort is placed into neural architecture search (NAS) [28, 54, 49, 39, 32] with the hypothesis that improvements on a structural level introduce inductive bias that improves the generalizability of a neural network. As most of the prior arts are evaluated on benchmark datasets that are distributed similarly to the training set that it is trained on, the evaluation results are not a good measure of the generalization. We argue that the ability to adapt, acquire new knowledge and recall previously learned information plays an important role in reaching true generalization. The importance of learning to learn, i.e. meta-learning, has shone the spotlight on two major research direction that we will focus on — *few-shot learning* and *continual learning*. In few-shot learning [12, 37, 45, 14], the goal is to learn novel concepts with as few samples as possible, i.e. evaluating the capability of adapting to new tasks. Whereas in continual learning, the ability to learn an increasing amount of concepts while not forgetting old ones is evaluated.

Following OML [22], we separate the feature extraction part and the decision making part of the network, defined in OML as representation learning network (RLN) and prediction learning network (PLN) respectively. The fast and slow learning in OML is performed on an architecture level, i.e. RLN is updated in the outer loop (slow weights) and PLN is updated in the inner loop (fast weights). Such approach has proven to be helpful in learning sparse representation that are beneficial for fast adaptation and prevention of catastrophic forgetting. We take one step further by introducing sparsity on an architectural level, accomplished through the introduction of Attentive Independent Mechanisms (AIM). AIM is composed of a set of mechanisms that competitively

attend to the input representation, having mechanisms that are closely related to the input representation being activated during inference. AIM can be understood as a mixture of experts competing to explain an incoming representation, hence only the mechanisms that best explain the input representation will be updated, leading to a sparse representation or modeling on an architectural level. Having sparse modeling on an architectural level for higher-order representations has its benefits, as only the experts or mechanisms that best explain a task will be involved in the learning process, helping in the acceleration of learning new concepts and the mitigation of catastrophic forgetting. To demonstrate the potential of AIM as a fundamental building block for fast learning without forgetting, we demonstrate its strength on few-shot classification [12, 43, 53] and continual learning [5, 22, 23] benchmarks.

Our contributions are as follows: (1) In Section 3, we give a detailed description and formulation of AIM — a novel module that can be used for few-shot and continual learning. (2) We apply AIM on few-shot learning and continual learning tasks in Section 4.1 and Section 4.2 respectively. Qualitative and quantitative results are shown for both learning tasks, giving readers an insight on the importance of having AIM in the context of few-shot and continual learning. For few-shot classification, experiments are performed on CIFAR-FS and MiniImageNet whereas for continual learning, experiments are performed on Omniglot, CIFAR-100 and MiniImageNet. Substantial improvement in accuracy over prior arts are shown.

2. Related Work

Meta-learning revolves on the idea of learning to learn, hoping that through the observation of training iterations on a few tasks, we are able to generalize to unseen tasks with only a few or zero samples. Meta-learning is usually composed of a support set and a query set. The support set is used for fast adaptation and the query set is used to evaluate the adapted model and to meta-learn the adaptation procedure. Model-based meta-learning methods include the work by [35] that uses a meta-learner based on a LSTM [18] which includes all previously seen samples, i.e. all support samples of a task are considered during the class prediction of query samples through an attentive mechanism. Another similar work by [44] augments LSTM with an external memory bank. [36] incorporates fast and slow weights for few-shot classification.

Metric-based meta-learning methods include Siamese Network proposed by [24] which predicts whether two images originate from the same class. [50] proposed Matching Networks that uses cosine distance in an attention kernel to measure the similarity of images in its embedding space. [45] later found that using Euclidean distance as a metric instead of cosine distance improves performance. A generalization of all the mentioned work is done by modeling the metric using a graph neural network proposed by [13].

Optimization-based meta-learning includes [42] that proposed using a LSTM meta-learner which provides gradient to a convolutional network-based fast learner. [12, 37] proposed an inner and outer-loop optimization method having fast adaptation in the inner-loop and an outer-loop update that backpropagates through the inner-loop updates. [53] used the concept of inner and outer loop-update by having the context parameters (embeddings of tasks) updated in the inner-loop. LEO [43] has its classifier weights generated by a low-dimensional latent embedding updated in the inner-loop. [15] proposed a similar approach where classification weights are generated using feature vectors that corresponds to the support set. SIB [20] performs transductive inference using synthetic gradient [21] on the feature averaging variant classifier proposed by [15]. Transductive inference was first introduced to the context of few-shot classification by [33], having a graph constructed for the support set and the query set, with labels propagated within the graph. As the architecture proposed by [33] is restrictive, [19] proposed a more general approach that uses a cross attention module that models semantic relevance between the support and query set.

In continual learning, the objective is to mitigate catastrophic forgetting [23]. Earlier works are based on regularization method, with [17] proposing the use of fast and slow training weights, borrowing the idea of plasticity and stability for network training. This idea is then adopted by OML [22] to learn representations that are useful for future learning and helps in mitigating catastrophic forgetting. Similarly, fast and slow learning is applied to ANML [5], having a neuromodulatory network modeled using slow weights. [1] uses task-specific gate module and prediction head to reduce competitive effect between classes. A criterion is designed in [3] to store most-interfered samples in a fixed-sized rehearsal memory.

3. Method

As Attentive Independent Mechanisms (AIM) is used to model higher order information, we place it right after a feature extractor, defined as $\mathbf{z} = f_{\boldsymbol{\psi}}(\mathbf{x})$. $f_{\boldsymbol{\psi}}(\cdot)$ is a series of convolutional layers parameterized by $\boldsymbol{\psi}$, $\mathbf{x}$ is an input sample and $\mathbf{z}$ is its corresponding representation. AIM is a module that is parameterized by $\mathbf{W}$ and is defined as $\mathcal{A}_{\mathbf{W}}$. The representation from AIM is then fed to a linear layer $\boldsymbol{\phi}$ for the task of classification. An illustration of AIM as a module is shown in Figure 1. We also show an illustration on the application of AIM to existing meta-learning frameworks used for few-shot learning and continual learning in Figure 2. We first describe the implementation of AIM as module in Section 3.1 followed by its integration to SIB [20] for few-shot learning in Section 3.2 and to OML [22]

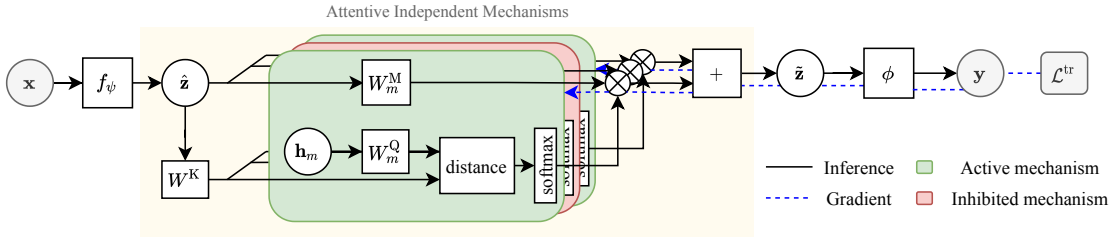


Figure 1: AIM is inserted right after the feature extractor f_ψ and before the output classifier ϕ. Only mechanisms closely related to the input representation are active (green boxes) and updated during the training phase (blue dashed lines).

and ANML [5] for continual learning in Section 3.3.

3.1. Attentive Independent Mechanisms

The goal of AIM is to learn a sparse set of mechanisms, i.e. mixture of experts, to decouple the modeling of higher order information from the feature extraction pipeline. These mechanisms compete and attend to the input representation in a top-down fashion using cross-attention [30, 29]. Through the strict selection of mechanisms, a sparse set of mechanisms will be selected for every task, inducing an architectural bias that helps in fast adaptation to new tasks and mitigating catastrophic forgetting. The structure of AIM is composed of a set of independent mechanisms, each parameterized by its own set of parameters. Each mechanism acts as an independent expert that collaborate with other experts to solve a particular task. AIM can be viewed as a static version of RIMs [16], i.e. temporal modeling of hidden states using LSTM [18] found in RIMs is removed. For RIMs, the model is fed with a continuous stream of inputs, making dynamical modeling using LSTM intuitive. For AIM, the assumption of having continuous stream of inputs does not hold as the practice of few-shot classification and continual learning have i.i.d. data being fed into the model during training and inference. Departing from RIMs, the objective of AIM is to show that through a mixture of experts, new concepts can be easily learned with minimal catastrophic forgetting. We hypothesize that by having a set of independent mechanisms, a sparse set of factorized representations or concepts can be extracted from the input representation. Such concepts have properties that are tasks-invariant which can be helpful in learning new tasks. The learning of concepts in AIM can also be understood as the amortized version of memory based models that stores samples either in the form of images or representations [44], which scales with the size of tasks in the system without limitation. AIM on the other hand performs implicit modeling of samples, analogous to the amortized modeling using a DNN instead of using a non-parametric method that stores samples from the training set for inference [8].

Following RIMs, AIM has a null vector $\emptyset$ that is concatenated with the input representation $\mathbf{z}$, giving us $\hat{\mathbf{z}} = [\mathbf{z}^T, \emptyset^T]^T$. The mechanisms then attend to the incoming latent representation $\hat{\mathbf{z}}$ as:

$$\tilde{\mathbf{z}} = \hat{\mathbf{z}} \left(\sum_{m=1}^{M} w_m(\hat{\mathbf{z}}) W_m^{\mathrm{M}} \right), \tag{1}$$

which could be understood as the passing of input representation $\hat{\mathbf{z}}$ through the weighted-summation of the mechanism weights, W_m^{M}. The summation of the outputs of the mechanisms makes the extension to arbitrary number of mechanisms trivial when compared to the concatenation of outputs used in RIMs. Concatenation is also infeasible when the output dimension of W_m^{M} is large, resulting in a wide input dimension for the upcoming layer. The summation of mechanisms also has the property of permutation invariance, reducing the complexity of the output classifier ϕ.

To encourage sparsity, we enforce the mechanisms to compete with each other to attend to the incoming representation. This is done by having only the weights of mechanisms that are closely related to the input representation to be selected, i.e. only top K mechanisms out of a total of M mechanisms are selected for the downstream prediction tasks. The strict selection of mechanisms forces the mechanisms to compete with each other to attend to the incoming signal, modeling the *biased competition theory* of selective attention [9]. The selection of mechanisms is given as:

$$w_m(\hat{\mathbf{z}}) = \begin{cases} \widetilde{w}_m(\hat{\mathbf{z}}), & \text{if } m \in \mathrm{top}_K\left(\widetilde{w}_1(\hat{\mathbf{z}}), \ldots, \widetilde{w}_M(\hat{\mathbf{z}})\right), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The indices corresponding to the top K values from a set is returned by the $\mathrm{top}_K(\cdot)$ operator. The weights used to weight the importance of the selected mechanisms are composed of the softmax of the normalized inner-product, $\langle \cdot, \cdot \rangle$, between the mechanisms' hidden state $\mathbf{h}_m$ and the input representation $\mathbf{z}$ that are first mapped to a lower-dimensional embedding by the query weight W_m^{Q} and key weight W^{K} of output dimension d respectively, given as:

$$\widetilde{w}_m(\hat{\mathbf{z}}) = \mathrm{softmax}\left(\frac{\langle \mathbf{h}_m W_m^{\mathrm{Q}}, \hat{\mathbf{z}} W^{\mathrm{K}} \rangle}{\sqrt{d}} \right). \tag{3}$$

Note that softmax is applied locally for each mechanism, i.e. the transformation of the attention values corresponding to $\mathbf{z}$ and $\emptyset$ into a probabilistic one. The value that corresponds to the input (not null) dimension from (3) is used for the top K comparison in (2).

Intervention during training. The training of AIM can be understood as an intervention procedure with the model selecting a few mechanisms to be included during the forward pass phase of training. Mechanisms that perform well on the training data are rewarded by having gradient update directed to the activated mechanisms, with the sensitivity to novel inputs reflected on $\mathbf{h}_m$. As one can predict, there is a possibility of the occurrence of *mechanism-overfitting*, where only a fixed set of mechanisms are active for all training tasks, losing the original motivation of having a sparse set of mechanisms acting as experts on different tasks. Mechanism-overfitting is also equivalent to having a DNN with multiple residual paths, resembling a single layer of Inception [48], diverting from our original goal of building models that are invariant across tasks.

To prevent the collapsing towards having only a few active mechanisms for all tasks, the trick is to enable the exploration of different amount of mechanisms during training, instead of locking down to the top K mechanisms. Stochasticity is introduced into the selection process by sampling top $K + l$ (also known as *stochastic sampling count*) instead of top K mechanisms. We then perform uniform sampling without replacement of K mechanisms from the top $K+l$ mechanisms, where the original sampling condition of (2) can now be written as:

$$w_m(\hat{\mathbf{z}}) = \begin{cases} \widetilde{w}_m(\hat{\mathbf{z}}), & \text{if } m \in \{\mathcal{K} \subseteq S \mid |\mathcal{K}| = K\}, \\ & \text{s.t. } S = \text{top}_{K+l}\left(\widetilde{w}_1(\hat{\mathbf{z}}), \ldots, \widetilde{w}_M(\hat{\mathbf{z}})\right) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Here, $|\cdot|$ is the cardinality operator to ensure that the sampled subset $\mathcal{K}$ is of size K and is sampled without replacement. Such intervention is analogous to stochastic intervention [25] and dropout [46] which adds stochasticity to the training of AIM, preventing the locking down to a few mechanisms that are attended to upon initialization.

Training and evaluation of AIM. Weight updates in AIM is similar to a typical layer in DNNs, i.e. gradients are backpropagated from the final loss function. A distinct difference from a conventional module in DNNs is that only the mechanisms activated during a forward pass are updated, resulting in a sparse set of weight updates. As AIM is designed to model higher order concepts, it is placed in the higher level of a DNN and has fast weights that are updated in the inner-loop of a meta-learning pipeline. The role of AIM as a module is shown in Figure 1. The procedure for the meta-training of AIM for both few-shot learning and continual learning is shown in Algorithm 1, whereas the meta-testing counterpart is shown in Algorithm 2. The algorithms shown are applicable for both few-shot and continual learning with the distinction between both highlighted with different colors — few-shot learning using SIB in green and continual learning using OML and ANML in blue. Step sizes for the inner-loop and outer-loop are defined as ν^{in} and ν^{out} respectively. Step size for synthetic gradient update used for SIB is defined as ϵ. For few-shot learning, the fast adaptation of AIM is evaluated using the meta-testing test set of the sampled task, i.e. $\mathcal{S}_{\text{test}}$ in the outer-loop. For continual learning, evaluation is performed after the completion of meta-training, and is tested on the entire meta-test train set $\mathcal{S}'_{\text{train}}$ and meta-test test set $\mathcal{S}'_{\text{test}}$.

Algorithm 1 Meta-Training: Training of AIM

Require: N sequential tasks $\mathcal{T}$; step size $\nu^{\text{in}}, \nu^{\text{out}}, \epsilon$; inner iterations T; modules $f_\psi, \mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}, \boldsymbol{\theta}$ (SIB only)

while not done **do**
  $\{\mathcal{S}_{\text{train}}, \mathcal{S}_{\text{test}}\} \sim \mathcal{T}$ ▷ SIB: i.i.d.; continual: sequential
  for $t \leftarrow 1, T$ **do**
    Update fast weights using $\mathcal{S}_{\text{train}}$ ▷ step size: ν^{in}
    SIB: $\mathcal{A}_{\mathbf{W}}$ OML: $\mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}$ ANML: $f_{\psi^{\text{P}}}, \mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}$
  end for
  Update $\boldsymbol{\phi}$ using transductive inference ▷ step size:ϵ
  Update slow weights using $\mathcal{S}_{\text{test}}$ ▷ step size: ν^{out}
    SIB: $\boldsymbol{\theta}$ OML: $f_{\boldsymbol{\psi}}$ ANML: $f_{\psi^{\text{NM}}}$
end while

Algorithm 2 Meta-Testing: Evaluation of AIM

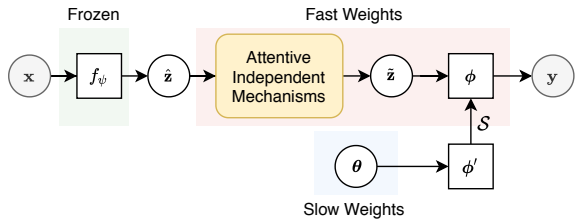
Require: N sequential unseen tasks $\mathcal{T}$; step size $\nu^{\text{in}}, \epsilon$; inner iterations T; modules $f_\psi, \mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}, \boldsymbol{\theta}$ (SIB only)

$\mathcal{S}'_{\text{train}} = \{\}; \mathcal{S}'_{\text{test}} = \{\}$ ▷ initialize empty set

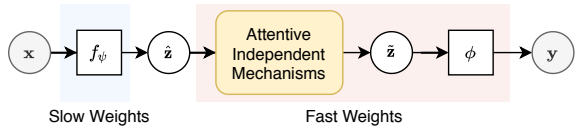
for $n \leftarrow 1, N$ **do**
  $\{\mathcal{S}_{\text{train}}, \mathcal{S}_{\text{test}}\} \sim \mathcal{T}_n$ ▷ SIB: i.i.d.; continual: sequential
  $\mathcal{S}'_{\text{train}}, \mathcal{S}'_{\text{test}} = \{\mathcal{S}'_{\text{train}}, \mathcal{S}_{\text{train}}\}, \{\mathcal{S}'_{\text{test}}, \mathcal{S}_{\text{test}}\}$ ▷ store trajectory
  for $t \leftarrow 1, T$ **do**
    Update fast weights using $\mathcal{S}_{\text{train}}$ ▷ step size: ν^{in}
    SIB: $\mathcal{A}_{\mathbf{W}}$ OML: $\mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}$ ANML: $f_{\psi^{\text{P}}}, \mathcal{A}_{\mathbf{W}}, \boldsymbol{\phi}$
  end for
  Update $\boldsymbol{\phi}$ using transductive inference ▷ step size:ϵ
  Evaluate on $\mathcal{S}_{\text{test}}$
end for
Evaluate on $\mathcal{S}'_{\text{train}}$ ▷ end of meta-test training trajectory
Evaluate on $\mathcal{S}'_{\text{test}}$ ▷ eval on entire meta-test testing set

3.2. Few-Shot Learning Using SIB

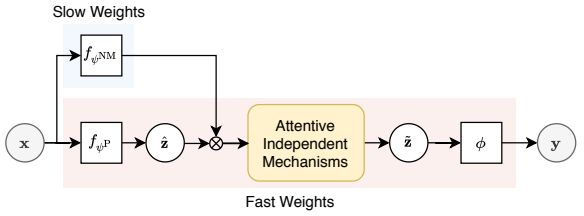
SIB is composed of two works: synthetic gradient modeling [21] and a feature averaging classifier [15]. In [21],



(a) Synthetic Information Bottleneck (SIB) [20]



(b) Online aware Meta-Learning (OML) [22]



(c) A Neuromodulated Meta-Learning Algorithm (ANML) [5]

Figure 2: Applying AIM on both few-shot learning ((a) SIB) and continual learning ((b) OML and (c) ANML) frameworks. For all frameworks, AIM (yellow) is placed directly after the feature extractor, $f_{\psi}(\cdot)$. With different learning scheme (fast and slow) used in meta-learning, weights or modules that correspond to fast update are highlighted in red, slow update are in blue and frozen weights are in green.

the idea is to use a synthetic gradient model, $\mathcal{S}$, that is meta-learned to generate gradient when labeled data is absent for transductive inference, i.e. update of weights without gradients propagated from a loss that is dependent on label. In [15], a classifier is defined as the cosine similarity between feature representations $\tilde{\mathbf{z}}$ and classification weight vectors $\boldsymbol{\phi}$. $\boldsymbol{\phi}$ is generated using an external classification weight generator $G_{\boldsymbol{\theta}}(\cdot)$ parameterized by $\boldsymbol{\theta}$ followed by iterative update by the synthetic gradient model $\mathcal{S}$. Feature vectors of P training samples of a novel category $\bar{\mathbf{Z}} = \{\bar{\mathbf{z}}^{(i)}\}_{i=1}^{P}$ are fed as input to generate a new set of weights for classification, $\boldsymbol{\phi}' = G_{\boldsymbol{\theta}}(\bar{\mathbf{Z}})$. In SIB, feature averaging based weight inference is used, i.e. the classification weight vector is obtained as $\boldsymbol{\phi}' = \boldsymbol{\theta} \odot \mathbf{w}_{\text{avg}}$, where $\odot$ is the Hadamard product and $\mathbf{w}_{\text{avg}} = \frac{1}{P}\sum_{i=1}^{P} \bar{\mathbf{z}}^{(i)}$ ($\bar{\mathbf{z}}$ is the ℓ_2-normalized version of $\tilde{\mathbf{z}}$). The classification weight vector $\boldsymbol{\phi}'$ is then updated iteratively using the synthetic gradient model in SIB, given as $\boldsymbol{\phi} = \mathcal{S}(\boldsymbol{\phi}')$. Both the synthetic gradient model and the weights of the weight generator $\boldsymbol{\theta}$ are meta-learned, i.e. updated in the outer-loop. To encourage sparse modeling of higher order concepts in the network, AIM is inserted right after the feature extractor $f_{\psi}(\cdot)$ and before the output linear classifier $\boldsymbol{\phi}$ that is generated using $G_{\boldsymbol{\theta}}(\bar{\mathbf{Z}})$ and $\mathcal{S}$, or,

$$\mathbf{y} = \boldsymbol{\phi}\left(\mathcal{A}_{\mathbf{W}}\left(f_{\psi}(\mathbf{x})\right)\right), \quad \text{where } \boldsymbol{\phi} = \mathcal{S}\left(G_{\boldsymbol{\theta}}\left(f_{\psi}(\bar{\mathbf{X}})\right)\right). \tag{5}$$

Training. Following the training pipeline in SIB [20], the weights of the feature extractor ψ are frozen to simplify the training procedure. The weights of the AIM, $\mathbf{W}$, and the output linear classifier, $\boldsymbol{\phi}$, are updated as fast weights, i.e. inner-loop. Only the weights of the classification weight generator $\boldsymbol{\theta}$ are updated as slow weight, i.e. outer-loop. The application of AIM to SIB is shown in Figure 2a.

3.3. Continual Learning: Learning Fast and Slow

It is shown in the task of continual learning that learning fast and slow from the context of meta-learning is helpful for the mitigation of catastrophic forgetting [22, 5]. OML [22] and ANML [5] are example frameworks for continual learning that uses this methodology, showing promising results. To validate our claim on the importance of incorporating sparse modeling on an architectural level for the mitigation of catastrophic forgetting, we insert AIM into both OML and ANML and observe the resulting performance.

OML. The entire architecture is split into two parts — *representation learning network* (RLN) and *prediction learning network* (PLN). RLN uses slow weights and PLN uses fast weights. Following our notations, RLN is the feature extractor in our work, f_{ψ}, and PLN is the classifier (not limited to a single layer), $\boldsymbol{\phi}$, in our work. AIM is inserted after the RLN and before the PLN, or,

$$\mathbf{y} = \boldsymbol{\phi}\left(\mathcal{A}_{\mathbf{W}}(f_{\psi}(\mathbf{x}))\right). \tag{6}$$

AIM is trained jointly with PLN, i.e. they have fast weights. The application of AIM to OML is shown in Figure 2b.

ANML. Two set of feature extractors are used in ANML — a *neuromodulatory network*, $f_{\psi^{\text{NM}}}$, and a *prediction network*, $\boldsymbol{\phi} \cdot f_{\psi^{\text{P}}}$. The role of the neuromodulatory network is to modulate the latent representation of the prediction network, i.e. the output of $f_{\psi^{\text{P}}}$ in Figure 2c. The output of the neuromodulatory network is element-wise multiplied with the outout of $f_{\psi^{\text{P}}}$ before passing to the final classifier, or $\mathbf{y} = \boldsymbol{\phi}\left(f_{\psi^{\text{NM}}}(\mathbf{x}) \odot f_{\psi^{\text{P}}}(\mathbf{x})\right)$. Only the neuromodulatory network has slow weights and the entire prediction network has fast weights. Similar to SIB and OML, AIM is inserted right after the feature extractor and uses fast weights,

$$\mathbf{y} = \boldsymbol{\phi}\left(\mathcal{A}_{\mathbf{W}}\left(f_{\psi^{\text{NM}}}(\mathbf{x}) \odot f_{\psi^{\text{P}}}(\mathbf{x})\right)\right). \tag{7}$$

4. Experiments

4.1. Few-Shot Learning

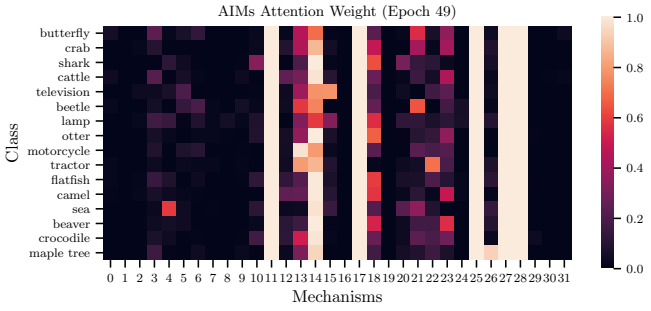
Datasets. For all datasets, class splits are disjoint. MiniImageNet [50] contains a total of 100 classes which are split into 64 training, 16 validation and 20 testing classes; images are of size 84×84. CIFAR-FS [6] is created by dividing CIFAR-100 into 64 training, 16 validation and 20 testing classes; images are of size 32×32. For few-shot classification, each task (episode) consists of a train set and a test set. For each task, k classes are sampled from the class pool mentioned. For each class, n examples are drawn and are relabeled as k disjoint classes forming the train set. For the test set, $15k$ samples are used. We show results of $k = 5$ for both $n = 1$ and $n = 5$.

Network architecture. We follow the setup in [20, 15, 40, 14] by using a 4-layer convolutional network with 64 feature channels (Conv-4-64) or a WideResNet (WRN-28-10) [51] as our feature extractor, f_ψ. f_ψ is pretrained in a typical end-to-end supervised learning fashion, i.e. the entire training set is used for batch update. Our classifier is adopted directly from [20, 15] having $\phi = G'_{\boldsymbol{\theta}}(\bar{\mathbf{Z}})$. For transductive inference [20], the synthetic gradient network is modeled by a MLP of 3 layers and hidden size $8k$. Classification is done by using the cosine-similarity based classifier found in [20, 15]. For AIM, all weights are linear layers. The hidden state $\mathbf{h}_m$ of the mechanisms are of dimension 256. The key and query weights (W^{K} and W^{Q}_m) maps the input and hidden state to a dimension of 128 to perform distance measurement. For the output dimension of the mechanism weights, W^{M}_m, we picked 400 for CIFAR-FS trained on Conv-4-64 and 800 for the rest; this decision is based on the dimension of the flattened feature map at the output of the feature extractor (not cherry-picked).

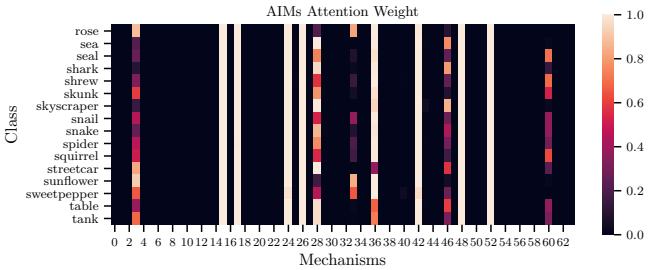
Training details. We use $M = 32$ mechanisms with top $K = 8$ mechanisms selected during inference with induced stochasticity by having $l = 2$ during training. SGD is used a batch size of 1 for 50,000 steps with learning rate $\epsilon = 10^{-3}$ for SIB's classifier synthetic update, $\nu^{\mathrm{out}} = 5 \times 10^{-3}$ for outer-loop update and $\nu^{\mathrm{in}} = 3 \times 10^{-3}$ for inner-loop update. The feature extractor is frozen during training. 1,000 tasks are sampled from the validation set for hyperparameter selection at each training epoch. All experiments are run on a single GTX1080Ti using PyTorch. A complete run of Conv-4-64 on CIFAR-FS and WRN-28-10 on MiniImageNet takes less than 2 hours and 5 hours respectively.

4.1.1 Qualitative Study: Activation of AIM

We show heatmaps that illustrate mechanisms activated for different classes from the validation set in Figure 3. The heatmap is plotted by averaging the mechanisms' activity for each class over the entire validation set, with 1 and 0 indicating active and inhibited mechanism respectively. We can observe that there's a set of mechanisms that are shared among tasks and another set that are distributed sparsely. The sharing of mechanisms can be understood as different classes sharing similar concepts. The sparse allocation of mechanisms over different classes show that there are features that are invariant for certain classes only, improving resiliency to covariate shift among distributions.



(a) SIB + AIM using Conv-4-64 on CIFAR-FS, 1-shot.



(b) OML + AIM on CIFAR-100.

Figure 3: With 1 indicating an active mechanism and 0 indicating an inhibited mechanism and having top K mechanisms selected for every inference, the average of the activation for the same class across the entire validation set is taken here. The active mechanisms can be categorized into two sets: 1. fixed set of shared active mechanisms; 2. sparse set of mechanisms with class-dependent activations.

4.1.2 Quantitative Study

Stochastic sampling count. To show the importance of inducing stochasticity in the mechanism selection process for inference, we perform an empirical study by varying the stochastic sampling count, $K + l$. We fix $K = 8$ and vary l from 0 to 24. As we can see from Figure 4a, the accuracy obtained by varying l have different maximum for different datasets, models and number of shots. For most cases, the peak accuracy usually occurs at small value of l and slowly deteriorates as more stochasticity is introduced.

Table 1: Average classification accuracies with 95% confidence intervals on the test-set of MiniImageNet and CIFAR-FS. 2000 episodes are sampled for MiniImageNet and CIFAR-FS using Conv-4-64 and WRN-28-10 as the feature extractor.

Method	Backbone	Transductive	MiniImageNet, 5-Way		CIFAR-FS, 5-Way	
			1-shot	5-shot	1-shot	5-shot
Matching Net [50]	Conv-4-64		44.2%	57%	-	-
MAML [12]	Conv-4-64		$48.7 \pm 1.8\%$	$63.1 \pm 0.9\%$	$58.9 \pm 1.9\%$	$71.5 \pm 1.0\%$
Prototypical Net [45]	Conv-4-64		$49.4 \pm 0.8\%$	$68.2 \pm 0.7\%$	$55.5 \pm 0.7\%$	$72.0 \pm 0.6\%$
Relation Net [47]	Conv-4-64		$50.4 \pm 0.8\%$	$65.3 \pm 0.7\%$	$55.0 \pm 1.0\%$	$69.3 \pm 0.8\%$
TPN [33]	Conv-4-64	✓	55.5%	69.9%	-	-
Gidaris *et al.* [14]	Conv-4-64		$54.8 \pm 0.4\%$	$71.9 \pm 0.3\%$	$63.5 \pm 0.3\%$	$79.8 \pm 0.2\%$
SIB [20]	Conv-4-64	✓	$58.0 \pm 0.6\%$	$70.7 \pm 0.4\%$	$68.7 \pm 0.6\%$	$77.7 \pm 0.4\%$
SIB + Linear layer	Conv-4-64	✓	$60.07 \pm 0.59\%$	$73.70 \pm 0.38\%$	$68.75 \pm 0.62\%$	$79.99 \pm 0.39\%$
AIM (ours)	Conv-4-64	✓	$\mathbf{61.90 \pm 0.57\%}$	$\mathbf{74.55 \pm 0.38\%}$	$\mathbf{71.09 \pm 0.62\%}$	$\mathbf{80.48 \pm 0.40\%}$
TADAM [38]	ResNet-12		$58.5 \pm 0.3\%$	$76.7 \pm 0.3\%$	-	-
SNAIL [44]	ResNet-12		$55.7 \pm 1.0\%$	$68.9 \pm 0.9\%$	-	-
CTM [31]	ResNet-18	✓	$64.1 \pm 0.8\%$	$80.5 \pm 0.1\%$	-	-
LEO [43]	WRN-28-10		$61.8 \pm 0.1\%$	$77.6 \pm 0.1\%$	-	-
Gidaris *et al.* [14]	WRN-28-10		$62.9 \pm 0.5\%$	$79.9 \pm 0.3\%$	$73.6 \pm 0.3\%$	$86.1 \pm 0.2\%$
SIB [20]	WRN-28-10	✓	$70.0 \pm 0.6\%$	$79.2 \pm 0.4\%$	$80.0 \pm 0.6\%$	$85.3 \pm 0.4\%$
SIB + Linear layer	WRN-28-10	✓	$67.38 \pm 0.54\%$	$80.54 \pm 0.34\%$	$78.02 \pm 0.55\%$	$79.91 \pm 0.38\%$
AIM (ours)	WRN-28-10	✓	$\mathbf{71.22 \pm 0.57\%}$	$\mathbf{82.25 \pm 0.34\%}$	$\mathbf{80.20 \pm 0.55\%}$	$\mathbf{87.34 \pm 0.36\%}$

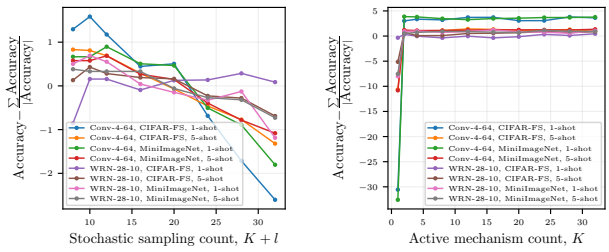


Figure 4: Illustrates the accuracy obtained by varying (a) stochastic sampling count ($K = 8$ and l is manipulated) and (b) active mechanisms count ($l = 0$ and K is manipulated). The zero mean-ed accuracy is shown to better demonstrate the change in accuracy across different model-dataset pairs. $|\cdot|$ is the cardinality operator.

Number of active mechanisms. An interesting question would be how maybe active mechanisms are required to reap the benefits of sparse activations. Empirical study is performed as shown in Figure 4b, showing the accuracy obtained by varying the number of active mechanisms K from 1 to 32. The results show that accuracy is low when K is small and saturates for larger values of K. This shows that a limited set of active mechanisms is sufficient. Sparsity in representation can still be met when the number of active mechanisms is large, but it will be cost inefficient during both training and inference.

Benchmark evaluation. As AIM is introduced as an additional component that's integrated into SIB [20], the gain in accuracy shows the importance of having a mixture of experts for fast adaptation. We also show the results for SIB with a linear layer (parameters equal the total parameters found in the AIM module) added before the classifier (SIB + Linear) to show that the gain in accuracy from AIM is not solely from the increase in parameters. From Table 1, we can see that AIM outperforms all existing few-shot classification methods by a noticeable margin. As only a single layer of AIM is explored, the coupling between AIM as found in RIMs [16] is not considered here. We believe that further improvements can be attained if layers of AIM are stacked, with coupling between them considered.

4.2. Continual Learning

Datasets. Omniglot [27] has over 1,623 characters from 50 diferent alphabets, where each character has 20 handwritten images of size 28×28. The dataset is split into 963 classes for meta-training and 660 classes for meta-testing. In each trajectory, 15 images are used for training and 5 images for testing in both meta-training and meta-testing. CIFAR-100 [26] is composed of 60,000 images of size 32×32 distributed uniformly over 100 classes, i.e. 500 train images and 100 test images for each class. 70 classes are used for meta-training and 30 classes are used for meta-testing. MiniImageNet [50] has 64 training classes and 20 testing classes with images of size 84×84. Each class has 600 images with 540 for training and 60 for testing. 30 training images are sampled for each class. In each trajectory of CIFAR-100 and MiniImageNet, we sample 30 train images for training all test images for testing for both meta-training and meta-testing.

Network architecture. We adopt the model from OML [22] and ANML [5] with a slight modification for our experiments. For OML, the feature extractor f_ψ is a 6-layer convolutional network with 112 channels and the classifier ϕ is a single linear layer with AIM $\mathcal{A}_W$ in between f_ψ and

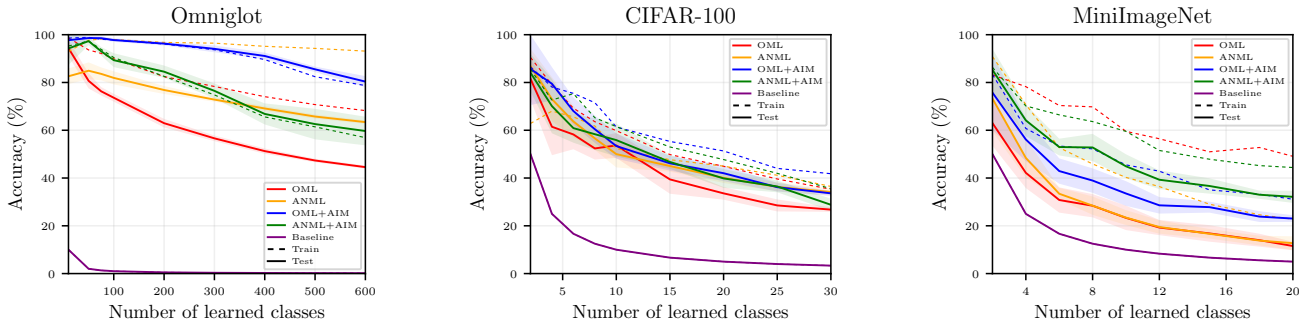


Figure 5: Evaluation of continual learning methods using dataset of various scales. Meta-test testing (training) trajectories are shown in solid (dashed) lines. All curves are averaged over 10 runs with standard deviation shown.

ϕ. For ANML, both neuromodulatory network $f_{\psi^{\text{NM}}}$ and prediction network $f_{\psi^{\text{P}}}$ have a 3-layer convolutional network and ϕ is a single linear layer with AIM placed after $f_{\psi^{\text{NM}}}$ and $f_{\psi^{\text{P}}}$. $f_{\psi^{\text{NM}}}$ has 112 channels while $f_{\psi^{\text{P}}}$ has 256 channels. For CIFAR-100 and MiniImageNet, an additional linear layer is placed before AIM for dimension reduction. The hidden state $\mathbf{h}_m \in \mathbb{R}^{128}$. $W^{\text{K}} \in \mathbb{R}^{\dim(\hat{\mathbf{z}})\times 128}$ and $W^{\text{Q}}_m \in \mathbb{R}^{128\times 128}$ maps theirs corresponding inputs to $\mathbb{R}^{128}$.

Training details. We use $M = 64$ mechanisms in our system and top $K = 10$ mechanisms are selected during inference with induced stochasticity by having $l = 2$ during training. We follow the 1st-order MAML strategy in [22, 5]. We use a batch size of 1 for 20,000 steps with step size of $\nu^{\text{out}} = 1 \times 10^{-3}$ for the outer-loop (slow weights) and $\nu^{\text{in}} = 1 \times 10^{-2}$ for the inner-loop (fast weights). A complete meta-training of AIM using OML or ANML on Omniglot, CIFAR-100 and MiniImageNet takes less than 2 hours, 3 hours and 6 hours respectively.

4.2.1 Qualitative Study: Activation of AIM

Following the settings in few-shot learning, activations of AIM when applied to OML are shown in Figure 2b. The activations are similar to what we observed in few-shot learning, i.e. a set of common mechanisms for all classes and another set for mechanisms that are sparsely activated.

4.2.2 Quantitative Study

To evaluate the capability of AIM to continually learn new concepts and mitigating catastrophic forgetting, we show the results of meta-test training and testing in Figure 5. To demonstrate that the accuracy gain using AIM is not due to the increase in parameters, *baseline* is plotted and is defined as the swapping of AIM with a linear layer containing the same amount of parameters as AIM added to OML. Samples of new classes are continuously fed without replacement, and samples of old classes are not stored. Prior works use the results from meta-test training as a measure of forgetting and meta-test testing to measure both forgetting and generalization error. We argue that memorizing features that doesn't transfer well to the testing set is not a good measure of forgetting. Results show that through the application of AIM, the difference between train and test accuracy is marginal, i.e. small generalization error, demonstrating that AIM is not only useful for the adaptation to new knowledge and mitigation of catastrophic forgetting, it also plays an important role in the learning of concepts that are generalizable to the test set. Consistent improvement in accuracy is observed when AIM is applied to existing continual learning frameworks. The only exception is the application of AIM to ANML trained on Omniglot, which could be remedied through a better selection of hyperparamters.

5. Conclusion

We have shown that AIM as a mixture of experts is an important building block for modeling higher-order concepts, translating to the capability of fast adaptation and mitigation of catastrophic forgetting. Through the sparse modeling of higher-order concepts, substantial improvement over prior arts can be seen for both few-shot and continual learning. It would be interesting to see the extension of AIM to multiple layers for hierarchical modeling of higher-order concepts.

Acknowledgement

This project is supported by MOST under code 107-2221-E-009 -125 -MY3. Eugene Lee is partially supported by Novatek Ph.D. Fellowship Award. The authors are grateful for the suggestions provided by Dr. Eugene Wong from University of California in Berkeley and Dr. Jian-Ming Ho from Academia Sinica of Taiwan.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.

[2] Wickliffe C Abraham and Anthony Robins. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005.

[3] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019.

[4] Andrew James Bauer and Marcel Adam Just. Monitoring the growth of the neural representations of new animal concepts. *Human brain mapping*, 36(8):3213–3226, 2015.

[5] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[6] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[9] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995.

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

[11] Joël Fagot and Robert G Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proceedings of the National Academy of Sciences*, 103(46):17564–17567, 2006.

[12] 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*, pages 1126–1135. JMLR. org, 2017.

[13] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[14] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8059–8068, 2019.

[15] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[16] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[17] 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*, pages 177–186, 1987.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Advances in Neural Information Processing Systems*, pages 4005–4016, 2019.

[20] Shell Xu Hu, Pablo Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. In *International Conference on Learning Representations (ICLR)*, 2020.

[21] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635. JMLR. org, 2017.

[22] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1820–1830, 2019.

[23] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *arXiv preprint arXiv:1708.02072*, 2017.

[24] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[25] Kevin B Korb, Lucas R Hope, Ann E Nicholson, and Karl Axnick. Varieties of causal intervention. In *Pacific Rim International Conference on Artificial Intelligence*, pages 322–331. Springer, 2004.

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

[27] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[28] Eugene Lee and Chen-Yi Lee. Neuralscale: Efficient scaling of neurons for resource-constrained deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1478–1487, 2020.

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

[30] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–216, 2018.

[31] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[32] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[33] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[34] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[35] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[36] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org, 2017.

[37] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[38] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[39] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[40] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[41] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[42] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[43] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[44] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[45] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[47] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[49] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[51] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[52] Dagmar Zeithamova, Michael L Mack, Kurt Braunlich, Tyler Davis, Carol A Seger, Marlieke TR van Kesteren, and Andreas Wutz. Brain mechanisms of concept learning. *Journal of Neuroscience*, 39(42):8259–8266, 2019.

[53] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. *arXiv preprint arXiv:1810.03642*, 2018.

[54] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.