Generalizing from a Few Examples: A Survey on Few-Shot Learning

YAQING WANG, Hong Kong University of Science and Technology and Baidu Research
QUANMING YAO*, 4Paradigm Inc.
JAMES T. KWOK, Hong Kong University of Science and Technology
LIONEL M. NI, Hong Kong University of Science and Technology

Machine learning has been highly successful in data-intensive applications, but is often hampered when the data set is small. Recently, Few-Shot Learning (FSL) is proposed to tackle this problem. Using prior knowledge, FSL can rapidly generalize to new tasks containing only a few samples with supervised information. In this paper, we conduct a thorough survey to fully understand FSL. Starting from a formal definition of FSL, we distinguish FSL from several relevant machine learning problems. We then point out that the core issue in FSL is that the empirical risk minimizer is unreliable. Based on how prior knowledge can be used to handle this core issue, we categorize FSL methods from three perspectives: (i) data, which uses prior knowledge to augment the supervised experience; (ii) model, which uses prior knowledge to reduce the size of the hypothesis space; and (iii) algorithm, which uses prior knowledge to alter the search for the best hypothesis in the given hypothesis space. With this taxonomy, we review and discuss the pros and cons of each category. Promising directions, in the aspects of the FSL problem setups, techniques, applications and theories, are also proposed to provide insights for future research.[1]

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Learning paradigms**.

Additional Key Words and Phrases: Few-Shot Learning, One-Shot Learning, Low-Shot Learning, Small Sample Learning, Meta-Learning, Prior Knowledge

ACM Reference Format:
Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Comput. Surv.* 1, 1, Article 1 (March 2020), 34 pages. https://doi.org/10.1145/3386252

1 INTRODUCTION

"Can machines think?" This is the question raised in Alan Turing's seminal paper entitled "Computing Machinery and Intelligence" [134] in 1950. He made the statement that "The idea behind digital

*Corresponding Author

[1] A list of references, which will be updated periodically, can be found at *https://github.com/tata1661/FewShotPapers.git*.

Authors' addresses: Yaqing Wang, ywangcy@connect.ust.hk, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Business Intelligence Lab and National Engineering Laboratory of Deep Learning Technology and Application, Baidu Research; Quanming Yao, yaoquanming@4paradigm.com, 4Paradigm Inc.; James T. Kwok, jamesk@cse.ust.hk, Department of Computer Science and Engineering, Hong Kong University of Science and Technology; Lionel M. Ni, ni@ust.hk, Department of Computer Science and Engineering, Hong Kong University of Science and Technology.

0360-0300/2020/3-ART1 $15.00
https://doi.org/10.1145/3386252

computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer". In other words, the ultimate goal of machines is to be as intelligent as humans. In recent years, due to the emergence of powerful computing devices (e.g., GPU and distributed platforms), large data sets (e.g., ImageNet data with 1000 classes [30]), advanced models and algorithms (e.g., convolutional neural networks (CNN) [73] and long short-term memory (LSTM) [58]), AI speeds up its pace to be like humans and defeats humans in many fields. To name a few, AlphaGo [120] defeats human champions in the ancient game of Go; and residual network (ResNet) [55] obtains better classification performance than humans on ImageNet. AI also supports the development of intelligent tools in many aspects of daily life, such as voice assistants, search engines, autonomous driving cars, and industrial robots.

Albeit its prosperity, current AI techniques cannot rapidly generalize from a few examples. The aforementioned successful AI applications rely on learning from large-scale data. In contrast, humans are capable of learning new tasks rapidly by utilizing what they learned in the past. For example, a child who learned how to add can rapidly transfer his knowledge to learn multiplication given a few examples (e.g., $2 \times 3 = 2 + 2 + 2$ and $1 \times 3 = 1 + 1 + 1$). Another example is that given a few photos of a stranger, a child can easily identify the same person from a large number of photos.

Bridging this gap between AI and humans is an important direction. It can be tackled by *machine learning*, which is concerned with the question of how to construct computer programs that automatically improve with experience [92, 94]. In order to learn from a limited number of examples with supervised information, a new machine learning paradigm called *Few-Shot Learning* (FSL) [35, 36] is proposed. A typical example is character generation [76], in which computer programs are asked to parse and generate new handwritten characters given a few examples. To handle this task, one can decompose the characters into smaller parts transferable across characters, and then aggregate these smaller components into new characters. This is a way of learning like human [77]. Naturally, FSL can also advance robotics [26], which develops machines that can replicate human actions. Examples include one-shot imitation [147], multi-armed bandits [33], visual navigation [37], and continuous control [156].

Another classic FSL scenario is where examples with supervised information are hard or impossible to acquire due to privacy, safety or ethic issues. A typical example is drug discovery, which tries to discover properties of new molecules so as to identify useful ones as new drugs [4]. Due to possible toxicity, low activity, and low solubility, new molecules do not have many real biological records on clinical candidates. Hence, it is important to learn effectively from a small number of samples. Similar examples where the target tasks do not have many examples include FSL translation [65], and cold-start item recommendation [137]. Through FSL, learning suitable models for these rare cases can become possible.

FSL can also help relieve the burden of collecting large-scale supervised data. For example, although ResNet [55] outperforms humans on ImageNet, each class needs to have sufficient labeled images which can be laborious to collect. FSL can reduce the data gathering effort for data-intensive applications. Examples include image classification [138], image retrieval [130], object tracking [14], gesture recognition [102], image captioning, visual question answering [31], video event detection [151], language modeling [138], and neural architecture search [19].

Driven by the academic goal for AI to approach humans and the industrial demand for inexpensive learning, FSL has drawn much recent attention and is now a hot topic. Many related machine learning approaches have been proposed, such as meta-learning [37, 106, 114], embedding learning [14, 126, 138] and generative modeling [34, 35, 113]. However, currently, there is no work that provides an organized taxonomy to connect these FSL methods, explains why some methods work while others fail, nor discusses the pros and cons of different approaches. Therefore, in this paper,

we conduct a survey on the FSL problem. In contrast, the survey in [118] only focuses on concept learning and experience learning for small samples.

Contributions of this survey can be summarized as follows:

- We give a formal definition on FSL, which naturally connects to the classic machine learning definition in [92, 94]. The definition is not only general enough to include existing FSL works, but also specific enough to clarify what the goal of FSL is and how we can solve it. This definition is helpful for setting future research targets in the FSL area.
- We list the relevant learning problems for FSL with concrete examples, clarifying their relatedness and differences with respect to FSL. These discussions can help better discriminate and position FSL among various learning problems.
- We point out that the core issue of FSL supervised learning problem is the unreliable empirical risk minimizer, which is analyzed based on error decomposition [17] in machine learning. This provides insights to improve FSL methods in a more organized and systematic way.
- We perform an extensive literature review, and organize them in an unified taxonomy from the perspectives of data, model and algorithm. We also present a summary of insights and a discussion on the pros and cons of each category. These can help establish a better understanding of FSL methods.
- We propose promising future directions for FSL in the aspects of problem setup, techniques, applications and theories. These insights are based on the weaknesses of the current development of FSL, with possible improvements to make in the future.

1.1 Organization of the Survey

The remainder of this survey is organized as follows. Section 2 provides an overview for FSL, including its formal definition, relevant learning problems, core issue, and a taxonomy of existing works in terms of data, model and algorithm. Section 3 is for methods that augment data to solve FSL problem. Section 4 is for methods that reduce the size of hypothesis space so as to make FSL feasible. Section 5 is for methods that alter the search strategy of algorithm to deal with the FSL problem. In Section 6, we propose future directions for FSL in terms of problem setup, techniques, applications and theories. Finally, the survey closes with conclusion in Section 7.

1.2 Notation and Terminology

Consider a learning task T, FSL deals with a data set $D = \{D_{\text{train}}, D_{\text{test}}\}$ consisting of a training set $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{I}$ where I is small, and a testing set $D_{\text{test}} = \{x^{\text{test}}\}$. Let $p(x, y)$ be the ground-truth joint probability distribution of input x and output y, and $\hat{h}$ be the optimal hypothesis from x to y. FSL learns to discover $\hat{h}$ by fitting D_{train} and testing on D_{test}. To approximate $\hat{h}$, the FSL model determines a hypothesis space $\mathcal{H}$ of hypotheses $h(\cdot; \theta)$'s, where θ denotes all the parameters used by h. Here, a parametric h is used, as a nonparametric model often requires large data sets, and thus not suitable for FSL. A FSL algorithm is an optimization strategy that searches $\mathcal{H}$ in order to find the θ that parameterizes the best $h^* \in \mathcal{H}$. The FSL performance is measured by a loss function $\ell(\hat{y}, y)$ defined over the prediction $\hat{y} = h(x; \theta)$ and the observed output y.

2 OVERVIEW

In this section, we first provide a formal definition of the FSL problem in Section 2.1 with concrete examples. To differentiate the FSL problem from relevant machine learning problems, we discuss their relatedness and differences in Section 2.2. In Section 2.3, we discuss the core issue that makes FSL difficult. Section 2.4 then presents a unified taxonomy according to how existing works handle the core issue.

2.1 Problem Definition

As FSL is a sub-area in machine learning, before giving the definition of FSL, let us recall how machine learning is defined in the literature.

*Definition 2.1 (**Machine Learning** [92, 94]).* A computer program is said to learn from experience E with respect to some classes of task T and performance measure P if its performance can improve with E on T measured by P.

For example, consider an image classification task (T), a machine learning program can improve its classification accuracy (P) through E obtained by training on a large number of labeled images (e.g., the ImageNet data set [73]). Another example is the recent computer program AlphaGo [120], which has defeated the human champion in playing the ancient game of Go (T). It improves its winning rate (P) against opponents by training on a database (E) of around 30 million recorded moves of human experts as well as playing against itself repeatedly. These are summarized in Table 1.

Table 1. Examples of machine learning problems based on Definition 2.1.

task T	experience E	performance P
image classification [73]	large-scale labeled images for each class	classification accuracy
the ancient game of Go [120]	a database containing around 30 million recorded moves of human experts and self-play records	winning rate

Typical machine learning applications, as in the examples mentioned above, require a lot of examples with supervised information. However, as mentioned in the introduction, this may be difficult or even not possible. FSL is a special case of machine learning, which targets at obtaining good learning performance given limited supervised information provided in the training set D_{train}, which consists of examples of inputs x_i's along with their corresponding output y_i's [15]. Formally, we define FSL in Definition 2.2.

Definition 2.2. ***Few-Shot Learning*** (FSL) is a type of machine learning problems (specified by E, T and P), where E contains only a limited number of examples with supervised information for the target T.

Existing FSL problems are mainly supervised learning problems. Concretely, *few-shot classification* learns classifiers given only a few labeled examples of each class. Example applications include image classification [138], sentiment classification from short text [157] and object recognition [35]. Formally, using notations from Section 1.2, *few-shot classification* learns a classifier h which predicts label y_i for each input x_i. Usually, one considers the N-way-K-shot classification [37, 138], in which D_{train} contains $I = KN$ examples from N classes each with K examples. *Few-shot regression* [37, 156] estimates a regression function h given only a few input-output example pairs sampled from that function, where output y_i is the observed value of the dependent variable y, and x_i is the input which records the observed value of the independent variable x. Apart from few-shot supervised learning, another instantiation of FSL is *few-shot reinforcement learning* [3, 33], which targets at finding a policy given only a few trajectories consisting of state-action pairs.

We now show three typical scenarios of FSL (Table 2):

- *Acting as a test bed for learning like human.* To move towards human intelligence, it is vital that computer programs can solve the FSL problem. A popular task (T) is to generate samples of a new character given only a few examples [76]. Inspired by how humans learn, the

computer programs learn with the E consisting of both the given examples with supervised information and pre-trained concepts such as parts and relations as prior knowledge. The generated characters are evaluated through the pass rate of visual Turing test (P), which discriminates whether the images are generated by humans or machines. With this prior knowledge, computer programs can also learn to classify, parse and generate new handwritten characters with a few examples like humans.

- *Learning for rare cases.* When obtaining sufficient examples with supervised information is hard or impossible, FSL can learn models for the rare cases. For example, consider a drug discovery task (T) which tries to predict whether a new molecule has toxic effects [4]. The percentage of molecules correctly assigned as toxic or non-toxic (P) improves with E obtained by both the new molecule's limited assay, and many similar molecules' assays as prior knowledge.
- *Reducing data gathering effort and computational cost.* FSL can help relieve the burden of collecting large number of examples with supervised information. Consider few-shot image classification task (T) [35]. The image classification accuracy (P) improves with the E obtained by a few labeled images for each class of the target T, and prior knowledge extracted from the other classes (such as raw images to co-training). Methods succeed in this task usually have higher generality. Therefore, they can be easily applied for tasks of many samples.

Table 2. Three FSL examples based on Definition 2.2.

task T	experience E		performance P
	supervised information	prior knowledge	
character generation [76]	a few examples of new character	pre-learned knowledge of parts and relations	pass rate of visual Turing test
drug toxicity discovery [4]	new molecule's limited assay	similar molecules' assays	classification accuracy
image classification [70]	a few labeled images for each class of the target T	raw images of other classes, or pre-trained models	classification accuracy

In comparison to Table 1, Table 2 has one extra column under "experience E" which is marked as "prior knowledge". As E only contains a few examples with supervised information directly related to T, it is natural that common supervised learning approaches often fail on FSL problems. Therefore, FSL methods make the learning of target T feasible by combining the available supervised information in E with some prior knowledge, which is "any information the learner has about the unknown function before seeing the examples" [86]. One typical type of FSL methods is Bayesian learning [35, 76]. It combines the provided training set D_{train} with some prior probability distribution which is available before D_{train} is given [15].

Remark 1. *When there is only one example with supervised information in E, FSL is called* ***one-shot learning*** *[14, 35, 138]. When E does not contain any example with supervised information for the target T, FSL becomes a* ***zero-shot learning*** *problem (ZSL) [78]. As the target class does not contain examples with supervised information, ZSL requires E to contain information from other modalities (such as attributes, WordNet, and word embeddings used in rare object recognition tasks), so as to transfer some supervised information and make learning possible.*

2.2 Relevant Learning Problems

In this section, we discuss some relevant machine learning problems. The relatedness and difference with respect to FSL are clarified.

- *Weakly supervised learning* [163] learns from experience E containing only weak supervision (such as incomplete, inexact, inaccurate or noisy supervised information). The most relevant problem to FSL is *weakly supervised learning with incomplete supervision* where only a small amount of samples have supervised information. According to whether the oracle or human intervention is leveraged, this can be further classified into the following:
  - *Semi-supervised learning* [165], which learns from a small number of labeled samples and (usually a large number of) unlabeled samples in E. Example applications are text and webpage classification. *Positive-unlabeled learning* [81] is a special case of semi-supervised learning, in which only positive and unlabeled samples are given. For example, to recommend friends in social networks, we only know the users' current friends according to the friend list, while their relationships to other people are unknown.
  - *Active learning* [117], which selects informative unlabeled data to query an oracle for output y. This is usually used for applications where annotation labels are costly, such as pedestrian detection.

  By definition, *weakly supervised learning with incomplete supervision* includes only classification and regression, while FSL also includes reinforcement learning problems. Moreover, *weakly supervised learning with incomplete supervision* mainly uses unlabeled data as additional information in E, while FSL leverages various kinds of prior knowledge such as pre-trained models, supervised data from other domains or modalities and does not restrict to using unlabeled data. Therefore, FSL becomes weakly supervised learning problem only when prior knowledge is unlabeled data and the task is classification or regression.
- *Imbalanced learning* [54] learns from experience E with a skewed distribution for y. This happens when some values of y are rarely taken, as in fraud detection and catastrophe anticipation applications. It trains and tests to choose among all possible y's. In contrast, FSL trains and tests for y with a few examples, while possibly taking the other y's as prior knowledge for learning.
- *Transfer learning* [101] transfers knowledge from the source domain/task, where training data is abundant, to the target domain/task, where training data is scarce. It can be used in applications such as cross-domain recommendation, WiFi localization across time periods, space and mobile devices. *Domain adaptation* [11] is a type of transfer learning in which the source/target tasks are the same but the source/target domains are different. For example, in sentiment analysis, the source domain data contains customer comments on movies, while the target domain data contains customer comments on daily goods. Transfer learning methods are popularly used in FSL [7, 82, 85], where the prior knowledge is transferred from the source task to the few-shot task.
- *Meta-learning* [59] improves P of the new task T by the provided data set and the meta-knowledge extracted across tasks by a meta-learner. Specifically, the meta-learner gradually learns generic information (meta-knowledge) across tasks, and the learner generalizes the meta-learner for a new task T using task-specific information. It has been successfully applied in problems such as learning optimizers [5, 80], dealing with the cold-start problem in collaborative filtering [137], and guiding policies by natural language [25]. Meta-learning methods can be used to deal with the FSL problem. As will be shown in Sections 4 and 5, the meta-learner is taken as prior knowledge to guide each specific FSL task. A formal definition of meta-learning and how it is used for the FSL problem are provided in Appendix A.

2.3 Core Issue

In any machine learning problem, usually there are prediction errors and one cannot obtain perfect predictions. In this section, we illustrate the core issue of FSL based on error decomposition in

supervised machine learning [17, 18]. This analysis applies to FSL supervised learning including classification and regression, and can also provide insights for understanding FSL reinforcement learning.

2.3.1 Empirical Risk Minimization. Given a hypothesis h, we want to minimize its *expected risk* R, which is the loss measured with respect to $p(x, y)$. Specifically,

$$R(h) = \int \ell(h(x), y)\, dp(x, y) = \mathbb{E}[\ell(h(x), y)].$$

As $p(x, y)$ is unknown, the *empirical risk* (which is the average of sample losses over the training set D_{train} of I samples)

$$R_I(h) = \frac{1}{I} \sum_{i=1}^{I} \ell(h(x_i), y_i),$$

is usually used as a proxy for $R(h)$, leading to *empirical risk minimization* [94, 136] (with possibly some regularizers). For illustration, let

- $\hat{h} = \arg\min_h R(h)$ be the function that minimizes the expected risk;
- $h^* = \arg\min_{h \in \mathcal{H}} R(h)$ be the function in $\mathcal{H}$ that minimizes the expected risk;
- $h_I = \arg\min_{h \in \mathcal{H}} R_I(h)$ be the function in $\mathcal{H}$ that minimizes the empirical risk.

As $\hat{h}$ is unknown, one has to approximate it by some $h \in \mathcal{H}$. h^* is the best approximation for $\hat{h}$ in $\mathcal{H}$, while h_I is the best hypothesis in $\mathcal{H}$ obtained by empirical risk minimization. For simplicity, we assume that $\hat{h}, h^*$ and h_I are unique. The *total error* can be decomposed as [17, 18]:

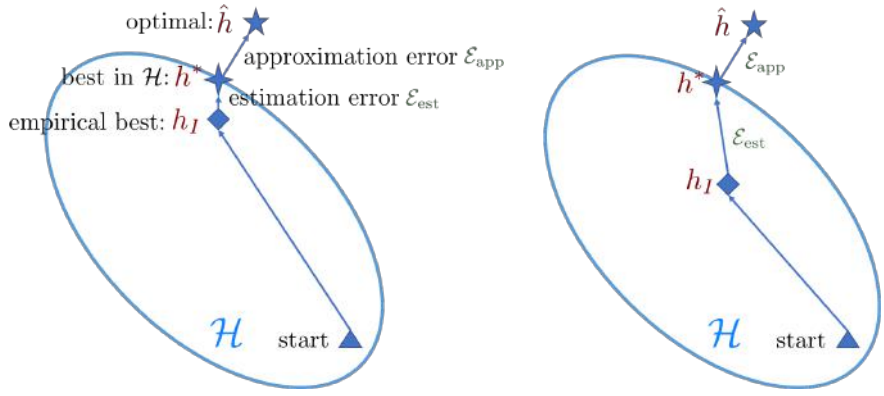
$$\mathbb{E}[R(h_I) - R(\hat{h})] = \underbrace{\mathbb{E}[R(h^*) - R(\hat{h})]}_{\mathcal{E}_{\text{app}}(\mathcal{H})} + \underbrace{\mathbb{E}[R(h_I) - R(h^*)]}_{\mathcal{E}_{\text{est}}(\mathcal{H}, I)}, \tag{1}$$

where the expectation is with respect to the random choice of D_{train}. The *approximation error* $\mathcal{E}_{\text{app}}(\mathcal{H})$ measures how close the functions in $\mathcal{H}$ can approximate the optimal hypothesis $\hat{h}$, and the *estimation error* $\mathcal{E}_{\text{est}}(\mathcal{H}, I)$ measures the effect of minimizing the empirical risk $R_I(h)$ instead of the expected risk $R(h)$ within $\mathcal{H}$.

As shown, the total error is affected by $\mathcal{H}$ (hypothesis space) and I (number of examples in D_{train}). In other words, learning to reduce the total error can be attempted from the perspectives of (i) data, which provides D_{train}; (ii) model, which determines $\mathcal{H}$; and (iii) algorithm, which searches for the optimal $h_I \in \mathcal{H}$ that fits D_{train}.

2.3.2 Unreliable Empirical Risk Minimizer. In general, $\mathcal{E}_{\text{est}}(\mathcal{H}, I)$ can be reduced by having a larger number of examples [17, 18, 41]. Thus, when there is sufficient training data with supervised information (i.e., I is large), the empirical risk minimizer h_I can provide a good approximation $R(h_I)$ to the best possible $R(h^*)$ for h's in $\mathcal{H}$.

However, in FSL, the number of available examples I is small. The empirical risk $R_I(h)$ may then be far from being a good approximation of the expected risk $R(h)$, and the resultant empirical risk minimizer h_I overfits. Indeed, this is the core issue of FSL supervised learning, i.e., *the empirical risk minimizer h_I is no longer reliable.* Therefore, FSL is much harder. A comparison of learning with sufficient and few training samples is shown in Figure 1.

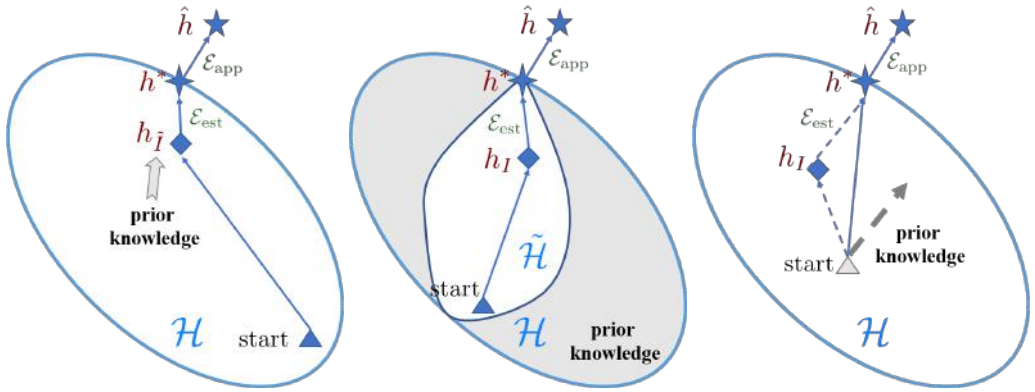


(a) Large I. (b) Small I.

Fig. 1. Comparison of learning with sufficient and few training samples.

2.4 Taxonomy

To alleviate the problem of having an unreliable empirical risk minimizer h_I in FSL supervised learning, prior knowledge must be used. Based on which aspect is enhanced using prior knowledge, existing FSL works can be categorized into the following perspectives (Figure 2).



(a) Data. (b) Model. (c) Algorithm.

Fig. 2. Different perspectives on how FSL methods solve the few-shot problem.

- *Data.* These methods use prior knowledge to augment D_{train}, and increase the number of samples from I to $\tilde{I}$, where $\tilde{I} \gg I$. Standard machine learning models and algorithms can then be used on the augmented data, and a more accurate empirical risk minimizer $h_{\tilde{I}}$ can be obtained (Figure 2(a)).
- *Model.* These methods use prior knowledge to constrain the complexity of $\mathcal{H}$, which results in a much smaller hypothesis space $\tilde{\mathcal{H}}$. As shown in Figure 2(b), the gray area is not considered for optimization as they are known to be unlikely to contain the optimal h^* according to prior knowledge. For this smaller $\tilde{\mathcal{H}}$, D_{train} is sufficient to learn a reliable h_I [43, 86, 99].

- *Algorithm.* These methods use prior knowledge to search for the θ which parameterizes the best hypothesis h^* in $\mathcal{H}$. Prior knowledge alters the search strategy by providing a good initialization (gray triangle in Figure 2(c)), or guiding the search steps (the gray dotted lines in Figure 2(b)). For the latter, the resultant search steps are affected by both prior knowledge and empirical risk minimizer.

Accordingly, existing works can be categorized into a unified taxonomy as shown in Figure 3. We will detail each category in the following sections.

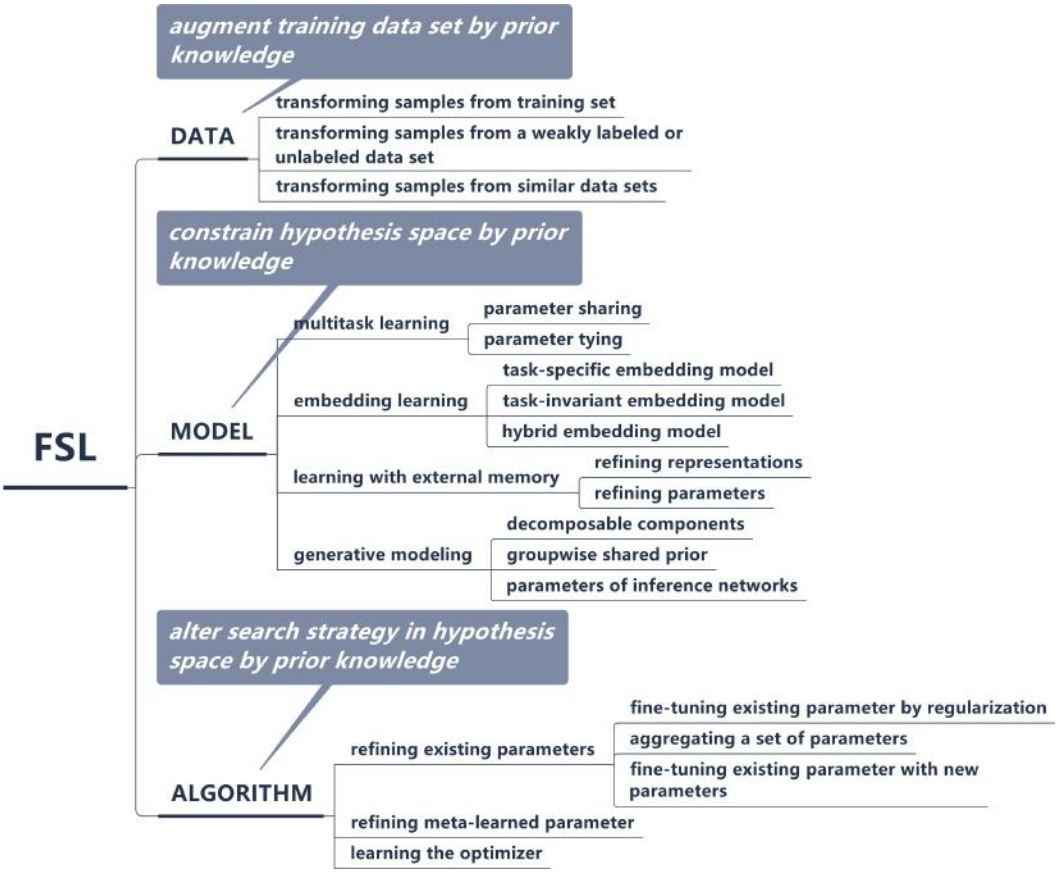


Fig. 3. A taxonomy of FSL methods based on the focus of each method.

3 DATA

FSL methods in this section use prior knowledge to augment data D_{train}, such that the supervised information in E is enriched. With the augmented sample set, the data is sufficient enough to obtain a reliable h_I (Figure 4).

Data augmentation via hand-crafted rules is usually used as pre-processing in FSL methods. They can introduce different kinds of invariance for the model to capture. For example, on images, one can use translation [12, 76, 114, 119], flipping [103, 119], shearing [119], scaling [76, 160], reflection [34, 72], cropping [103, 160] and rotation [114, 138]. However, designing these rules depends heavily on domain knowledge and requires expensive labor cost. Moreover, the augmentation rules can be

specific to the data set, making them hard to be applied to other data sets. Moreover, it is unlikely that human can enumerate all possible invariance. Therefore, manual data augmentation cannot solve the FSL problem completely [12, 34, 72, 76, 114, 119].

Besides these hand-crafted rules, we review in the following more advanced data augmentation methods. Depending on what samples are transformed and added to D_{train}, we categorize these methods as shown in Table 3.

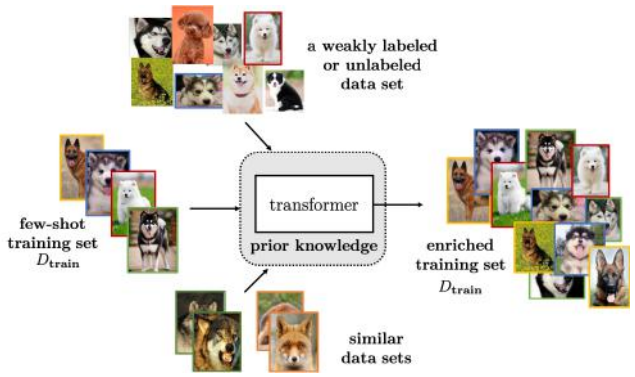


Fig. 4. Solving the FSL problem by data augmentation.

Table 3. Characteristics for FSL methods focusing on the data perspective. The transformer $t(\cdot)$ takes input (x, y) and returns synthesized sample $(\tilde{x}, \tilde{y})$ to augment the few-shot D_{train}.

category	input (x, y)	transformer t	output $(\tilde{x}, \tilde{y})$
transforming samples from D_{train}	original (x_i, y_i)	learned transformation function on x_i	$(t(x_i), y_i)$
transforming samples from a weakly labeled or unlabeled data set	weakly labeled or unlabeled $(\bar{x}, -)$	a predictor trained from D_{train}	$(\bar{x}, t(\bar{x}))$
transforming samples from similar data sets	samples $\{(\hat{x}_j, \hat{y}_j)\}$ from similar data sets	an aggregator to combine $\{(\hat{x}_j, \hat{y}_j)\}$	$(t(\{\hat{x}_j\}), t(\{\hat{y}_j\}))$

3.1 Transforming Samples from D_{train}

This strategy augments D_{train} by transforming each $(x_i, y_i) \in D_{\text{train}}$ into several samples with variations. The transformation procedure is included in experience E as prior knowledge so as to generate additional samples. An early FSL paper [90] learns a set of geometric transformations from a similar class by iteratively aligning each sample with the other samples. The learned transformation is applied to each (x_i, y_i) to form a large data set, which can then be learned by standard machine learning methods. Similarly, a set of auto-encoders, each representing one intra-class variability, are learned from similar classes in [116]. New samples are generated by adding the learned variations to x_i. In [53], by assuming that all categories share some transformable variability across samples, a single transformation function is learned to transfer variation between sample pairs learned from the other classes to (x_i, y_i). In [74], instead of enumerating the variabilities within pairs, it transforms each x_i to several samples using a set of independent attribute strength regressors learned from a large set of scene images, and assigns the label of the original x_i to these new samples. Improved upon [74], a continuous attribute subspace is used to add attribute variations to x in [82].

3.2 Transforming Samples from a Weakly Labeled or Unlabeled Data Set

This strategy augments D_{train} by selecting samples with the target label from a large data set which is weakly labeled or unlabeled. For example, in photos taken by a surveillance camera, there are people, cars and roads but none of them are labeled. Another example is a video for a long presentation. This contains a series of gestures of the speaker, but none of them are annotated explicitly. As such a data set contains large variations of samples, augmenting them to D_{train} helps depict a clearer $p(x, y)$. Moreover, collecting such a data set is easier as human effort is not needed for labeling. However, though the collection cost is low, a major issue is how to select samples with the target label to be augmented to D_{train}. In [102], an exemplar SVM is learned for each target label in D_{train}, which is then used to predict labels for samples from a weakly labeled data set. Samples having the target labels are then added to D_{train}. In [32], instead of learning a classifier, label propagation is directly used to label an unlabeled data set. In [148], a progressive strategy is used to select informative unlabeled samples. The selected samples are then assigned pseudo-labels and used to update the CNN.

3.3 Transforming Samples from Similar Data Sets

This strategy augments D_{train} by aggregating and adapting input-output pairs from a similar but larger data sets. The aggregation weight is usually based on some similarity measure between samples. In [133], it extracts the aggregation weight from an auxiliary text corpus [133]. As these samples may not come from the target FSL class, directly augmenting the aggregated samples to D_{train} may be misleading. Therefore, a generative adversarial network (GAN) [46] is designed to generate indistinguishable synthetic $\tilde{x}$ aggregated from a data set of many samples [42]. It has two generators, one maps samples of the few-shot class to the large-scale class, and the other maps samples of the large-scale class to the few-shot class (to compensate for the lack of samples in GAN training).

3.4 Discussion and Summary

The choice of which augmentation strategy to use depends on the application. Sometimes, a large number of weakly supervised or unlabeled samples exist for the target task (or class), but few-shot learning is preferred because of the high cost of gathering annotated data and/or computational cost (which corresponds to the third scenario introduced in Section 2.1). In this case, one can perform augmentation by transforming samples from a weakly labeled or unlabeled data set. When a large-scale unlabeled data set is hard to collect, but the few-shot class has some similar classes, one can transform samples from these similar classes. If only some learned transformers rather than raw samples are available, augmentation can be done by transforming the original samples from D_{train}.

In general, solving a FSL problem by augmenting D_{train} is straightforward and easy to understand. The data is augmented by taking advantage of the prior information for the target task. On the other hand, the weakness of solving the FSL problem by data augmentation is that the augmentation policy is often tailor-made for each data set in an adhoc manner, and cannot be used easily on other data sets (especially for data sets from other domains). Recently, AutoAugment [27], which automatically learns the augmentation policy for deep network training, is proposed to address this issue. Apart from that, existing methods are mainly designed for images, as the generated images can be visually evaluated by humans easily. In contrast, text and audio involve syntax and structures, and are harder to generate. A recent attempt on using data augmentation for text is reported in [144].

4 MODEL

In order to approximate the ground-truth hypothesis $\hat{h}$, the model has to determine a hypothesis space $\mathcal{H}$ containing a family of hypotheses h's, such that the distance between the optimal $h^* \in \mathcal{H}$ and $\hat{h}$ is small.

Given the few-shot D_{train} with limited samples, one can choose a small $\mathcal{H}$ with only simple models (such as linear classifiers) [92, 94]. However, real-world problems are typically complicated, and cannot be well represented by an hypothesis h from a small $\mathcal{H}$ (which can lead to a large $\mathcal{E}_{\text{app}}(\mathcal{H})$ in (1)) [45]. Therefore, a large enough $\mathcal{H}$ is preferred in FSL, which makes standard machine learning models infeasible. FSL methods in this section manage to learn by constraining $\mathcal{H}$ to a smaller hypothesis space $\tilde{\mathcal{H}}$ via prior knowledge in E (Figure 2(b)). The empirical risk minimizer is then more reliable, and the risk of overfitting is reduced.

In terms of what prior knowledge is used, methods belonging to this category can be further classified into four types (Table 4).

Table 4. Characteristics for FSL methods focusing on the model perspective.

strategy	prior knowledge	how to constrain $\mathcal{H}$
multitask learning	other T's with their data sets D's	share/tie parameter
embedding learning	embedding learned from/together with other T's	project samples to a smaller embedding space in which similar and dissimilar samples can be easily discriminated
learning with external memory	embedding learned from other T's to interact with memory	refine samples using key-value pairs stored in memory
generative modeling	prior model learned from other T's	restrict the form of distribution

4.1 Multitask Learning

In the presence of multiple related tasks, *multitask learning* [23, 161] learns these tasks simultaneously by exploiting both task-generic and task-specific information. Hence, they can be naturally used for FSL. Here, we present some instantiations of using multitask learning in FSL .

We are given C related tasks $T_1, \ldots, T_C$, in which some of them have very few samples while some have a larger number of samples. Each task T_c has a data set $D_c = \{D^c_{\text{train}}, D^c_{\text{test}}\}$, in which D^c_{train} is the training set and D^c_{test} is the test set. Among these C tasks, we regard the few-shot tasks as *target tasks*, and the rest as *source tasks*. Multitask learning learns from D^c_{train}'s to obtain θ_c for each T_c. As these tasks are jointly learned, the parameter θ_c of h_c learned for task T_c is constrained by the other tasks. According to how the task parameters are constrained, we divide methods in this strategy as (i) parameter sharing; and (ii) parameter tying [45].

4.1.1 Parameter Sharing. This strategy directly shares some parameters among tasks (Figure 5). In [160], the two task networks share the first few layers for the generic information, and learn different final layers to deal with different outputs. In [61], two natural language processing tasks on legal texts are solved together: charge prediction and legal attribute prediction. A single embedding function is used to encode the criminal case description, which is then fed to task-specific embedding functions and classifiers. In [95], a variational auto-encoder is first pre-trained from the source tasks, and then cloned to the target task. Some layers in the two variational auto-encoders are shared in order to capture the generic information, while allowing both tasks to have some task-specific layers. The target task can only update its task-specific layers, while the source task can update both the shared and task-specific layers. In [12], both the original and generated samples are first

mapped to a task-specific space by learning separate embedding functions for the source and target tasks, and are then embedded by a shared variational auto-encoder.

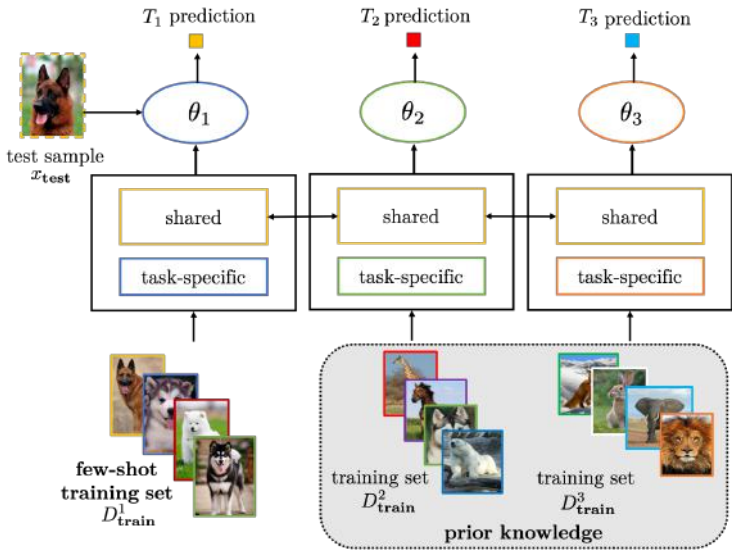


Fig. 5. Solving the FSL problem by multitask learning with parameter sharing.

4.1.2 Parameter Tying. This strategy encourages parameters (θ_c's) of different tasks to be similar (Figure 6) [45]. A popular approach is by regularizing the θ_c's. In [151], all pairwise differences of θ_c's are penalized. In [85], there is a CNN for the source task, and another one for the target task. Layers of these two CNNs are aligned using some specially designed regularization terms.

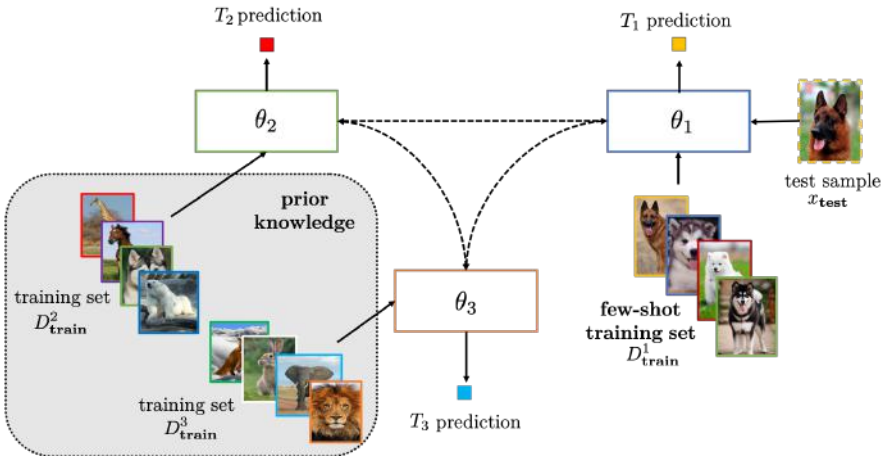


Fig. 6. Solving the FSL problem by multitask learning with parameter tying.

4.2 Embedding Learning

Embedding learning [63, 122] embeds each sample $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$ to a lower-dimensional $z_i \in \mathcal{Z} \subseteq \mathbb{R}^m$, such that similar samples are close together while dissimilar samples can be more easily differentiated. In this lower-dimensional $\mathcal{Z}$, one can then construct a smaller hypothesis space $\tilde{\mathcal{H}}$ which subsequently requires fewer training samples. The embedding function is mainly learned from prior knowledge, and can additionally use task-specific information from D_{train}.

Embedding learning has the following key components: (i) a function f which embeds test sample $x_{\text{test}} \in D_{\text{test}}$ to $\mathcal{Z}$, (ii) a function g which embeds training sample $x_i \in D_{\text{train}}$ to $\mathcal{Z}$, and (iii) a similarity function $s(\cdot, \cdot)$ which measures the similarity between $f(x_{\text{test}})$ and $g(x_i)$ in $\mathcal{Z}$. The test sample x_{test} is assigned to the class of x_i, whose embedding $g(x_i)$ is most similar to $f(x_{\text{test}})$ in $\mathcal{Z}$ according to s. Although one can use a common embedding function for both x_i and x_{test}, using two separate embedding functions may obtain better accuracy [14, 138]. A summary of existing embedding learning methods is shown in Table 5.

According to whether the parameters of embedding functions f and g vary across tasks, we classify these FSL methods as using a (i) task-specific embedding model; (ii) task-invariant (i.e., general) embedding model; and (iii) hybrid embedding model, which encodes both task-specific and task-invariant information.

Table 5. Characteristics of embedding learning methods.

category	method	embedding function f for x_{test}	embedding function g for D_{train}	similarity measure s
task-specific	mAP-DLM/SSVM[130]	CNN	the same as f	cosine similarity
task-invariant	class relevance pseudo-metric [36]	kernel	the same as f	squared ℓ_2 distance
	convolutional siamese net [70]	CNN	the same as f	weighted ℓ_1 distance
	Micro-Set[127]	logistic projection	the same as f	ℓ_2 distance
	Matching Nets [138]	CNN, LSTM	CNN, biLSTM	cosine similarity
	resLSTM [4]	GNN, LSTM	GNN, LSTM	cosine similarity
	Active MN [8]	CNN	biLSTM	cosine similarity
	SSMN [24]	CNN	another CNN	learned distance
	ProtoNet [121]	CNN	the same as f	squared ℓ_2 distance
	semi-supervised ProtoNet[108]	CNN	the same as f	squared ℓ_2 distance
	PMN [141]	CNN, LSTM	CNN, biLSTM	cosine similarity
	ARC [119]	LSTM, biLSTM	the same as f	-
	Relation Net [126]	CNN	the same as f	-
	GNN [115]	CNN, GNN	the same as f	learned distance
	TPN [84]	CNN	the same as f	Gaussian similarity
	SNAIL [91]	CNN	the same as f	-
hybrid	Learnet [14]	adaptive CNN	CNN	weighted ℓ_1 distance
	DCCN [162]	adaptive CNN	CNN	-
	R2-D2 [13]	adaptive CNN	CNN	-
	TADAM [100]	adaptive CNN	the same as f	squared ℓ_2 distance

4.2.1 Task-Specific Embedding Model. Task-specific embedding methods learn an embedding function tailored for each task, by using only information from that task. For example, using the few-shot data D^c_{train} of task T_c, all pairwise rankings among samples in D^c_{train} are enumerated as sample pairs in [130]. The number of training samples is thus increased, and an embedding function can be learned even though only the task-specific information is used.

4.2.2 Task-Invariant Embedding Model. Task-invariant embedding methods learn a general embedding function from a large-scale data set containing sufficient samples with various outputs, and then directly use this on the new few-shot D_{train} without retraining (Figure 7). The first FSL embedding model [36] embeds the samples using a kernel. Recently, more complicated embeddings are learned [70, 150] by a convolutional siamese net [20].

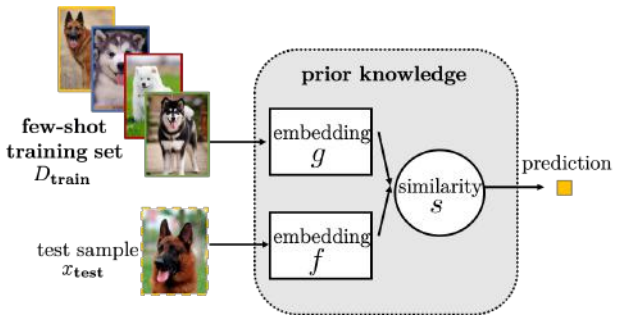


Fig. 7. Solving the FSL problem by task-invariant embedding model.

Although task-invariant embedding does not update the embedding model parameter using the few-shot D_{train}, many methods in this category [121, 126, 138] simulate the few-shot scenario while training the embedding model. Assume that we have training sets $\{D_c\}$, each has N classes. In each D_c, samples from only U out of its N classes are used for training. The embedding model is optimized by maximizing the performance on the remaining $N - U$ classes. Thus, the learned model will have good generalization for few-shot tasks. An early attempt [127] learns a linear embedding from $\{D_c\}$. Recently, more complicated task-invariant embedding models are learned via meta-learning[2] methods:

(1) *Matching Nets* [138] and its variants [4, 8, 24]: Matching Nets [138] meta-learns different embedding functions (f and g) for the training sample x_i and test sample x_{test}. The residual LSTM (resLSTM) [4] proposes better designs for f and g. An active learning variant of Matching Nets [8] adds a sample selection step, which labels the most beneficial unlabeled sample and uses it to augment D_{train}. The Matching Nets is also extended to set-to-set matching [24], which is useful in labeling multiple parts of a sample.

(2) *Prototypical Networks (ProtoNet)* [121] and its variants [100, 108, 141]: Instead of comparing $f(x_{\text{test}})$ with each $g(x_i)$ where $x_i \in D_{\text{train}}$, ProtoNet [121] only compares $f(x_{\text{test}})$ with the class prototypes in D_{train}. For class n, its prototype is simply $c_n = \frac{1}{K}\sum_{i=1}^{K} g(x_i)$, where the K x_i's are from class n. Empirically, this leads to more stable results and reduces the computation cost. The idea of using prototypes is introduced to the Matching Nets in [141]. A semi-supervised variant of ProtoNet assigns unlabeled samples to augment D_{train} via soft-assignment during learning [108].

(3) *Other methods.* Examples include Attentive Recurrent Comparators (ARC) [119], which uses a LSTM with attention [9] to compare different regions of x_{test} with prototype c_n, and then embeds the comparison results as an intermediate embedding. Additionally, it uses a bidirectional LSTM (biLSTM) to embed all comparisons as the final embedding. The Relation Net [126] uses a CNN to embed x_{test} and x_i to $\mathcal{Z}$, then concatenates them as the embedding, which is fed to another CNN to output a similarity score. The graph neural network (GNN) is

[2]A brief introduction on meta-learning is provided in Appendix A. The few-shot task is actually one of the meta-testing task T_t with $D_t = \{D^t_{\text{train}}, D^t_{\text{test}}\}$. For illustration simplicity, we drop the subscript and superscript t.

used in [84, 115] to leverage information from local neighborhoods. In few-shot reinforcement learning applications (as in continuous control and visual navigation), temporal information is important. The Simple Neural AttentIve Learner (SNAIL) [91] is an embedding network with interleaved temporal convolution layers and attention layers. The temporal convolution layer aggregates information from past time steps, while the attention layer selectively attends to specific time steps relevant to the current input.

4.2.3 Hybrid Embedding Model. Although task-invariant embedding methods can be applied to new tasks with a low computation cost, they do not leverage specific knowledge of the current task. When task specialty is the reason that D_{train} has only a few examples (such as learning for rare cases), simply applying a task-invariant embedding function may not be suitable. To alleviate this problem, hybrid embedding models adapt the generic task-invariant embedding model learned from prior knowledge by the task-specific information in D_{train} This is done by learning a function which takes information extracted from D_{train} as input and returns an embedding which acts as the parameter for $f(\cdot)$ (Figure 8).

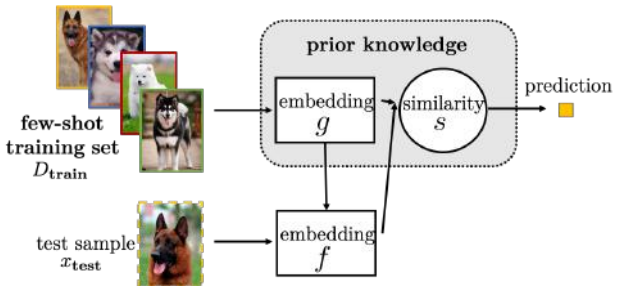


Fig. 8. Solving the FSL problem by hybrid embedding model.

Learnet [14] improves the task-invariant convolutional siamese net [70] by incorporating the specific information of D_{train}. It learns a meta-learner from multiple meta-training sets, and maps each training example $x_i \in D_{\text{train}}$ to the parameter of the learner (a convolutional siamese net). In this way, the parameter of $f(\cdot)$ changes with the given x_i, resulting in a hybrid embedding. Improved upon Learnet, the classification layer of the learner is replaced by ridge regression in [13], such that parameters can be efficiently obtained in closed-form. The following two works [100, 162] take D_{train} as a whole to output the task-specific parameter for $f(\cdot)$. Task dependent adaptive metric (TADAM) [100] averages class prototypes into the task embedding, and uses a meta-learned function to map it to the ProtoNet parameters. Dynamic Conditional Convolutional Network (DCCN) [162] uses a fixed set of filters, and learns the combination coefficients using D_{train}.

4.3 Learning with External Memory

Learning with external memory [49, 89, 124, 145] extracts knowledge from D_{train}, and stores it in an external memory (Figure 9). Each new sample x_{test} is then represented by a weighted average of contents extracted from the memory. This limits x_{test} to be represented by contents in the memory, and thus essentially reduces the size of $\mathcal{H}$.

A key-value memory [89] is usually used in FSL. Let the memory be $M \in \mathbb{R}^{b\times m}$, with each of its b memory slots $M(i) \in \mathbb{R}^m$ consisting of a key-value pair $M(i) = (M_{\text{key}}(i), M_{\text{value}}(i))$. A test sample x_{test} is first embedded by an embedding function f. However, unlike embedding methods, $f(x_{\text{test}})$ is not used directly as the representation of x_{test}. Instead, it is only used to query for the

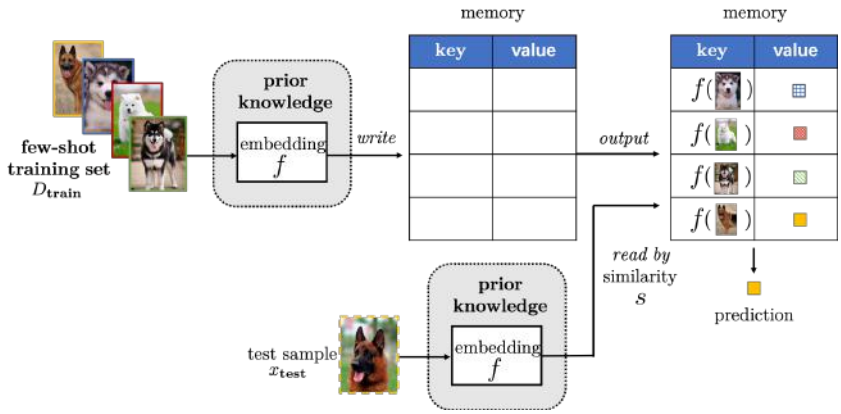


Fig. 9. Solving the FSL problem by learning with external memory. This figure illustrates a simplified example where the embedding function f is used for representation learning and the memory takes $f(x_i)$ as the key and output y_i as the label.

most similar memory slots, based on the similarity $s(f(x_{\text{test}}), M_{\text{key}}(i))$ between $f(x_{\text{test}})$ and each key $M_{\text{key}}(i)$. The values of the most similar memory slots ($M_{\text{value}}(i)$'s) are extracted and combined to form the representation of x_{test}. This is then used as input to a simple classifier (such as a softmax function) to make prediction. As manipulating M is expensive, M usually has a small size. When M is not full, new samples can be written to vacant memory slots. When M is full, one has to decide which memory slots to be replaced. Table 6 introduces the characteristics for methods with external memory.

Table 6. Characteristics of FSL methods based on learning with external memory. Here, f is an embedding function usually pre-trained by CNN or LSTM.

category	method	memory M		similarity s
		key M_{key}	value M_{value}	
refining representations	MANN [114]	$f(x_i, y_{i-1})$	$f(x_i, y_{i-1})$	cosine similarity
	APL [104]	$f(x_i)$	y_i	squared ℓ_2 distance
	abstraction memory[149]	$f(x_i)$	word embedding of y_i	dot product
	CMN [164]	$f(x_i)$	y_i, age	dot product
	life-long memory [65]	$f(x_i)$	y_i, age	cosine similarity
	Mem2Vec[125]	$f(x_i)$	word embedding of y_i, age	dot product
refining parameters	MetaNet [96]	$f(x_i)$	fast weight	cosine similarity
	CSNs [97]	$f(x_i)$	fast weight	cosine similarity
	MN-Net [22]	$f(x_i)$	y_i	dot product

As each x_{test} is represented as a weighted average of values extracted from the memory, the quality of key-value pairs in the memory is important. According to the functionality of the memory, FSL methods in this category can be subdivided into two types.

4.3.1 Refining Representations. The following methods carefully put D_{train} into the memory, such that the stored key-value pairs can represent x_{test} more accurately. Memory-Augmented Neural Networks (MANN) [114] meta-learns the embedding f, and maps samples of the same class to the same value. Samples of the same class then refine their class representations in the memory together. This class representation can be viewed as a refined class prototype in ProtoNet [121].

The surprise-based memory module [104] updates M only when it cannot represent an x_i well. Hence, updating M using this x_i makes M more expressive, and also reduces the computation cost. The abstract memory [149] uses two memories. One extracts relevant key-value pairs from a fixed memory containing large-scale machine annotated data set, and the other refines the extracted values and abstracts out the most useful information for few-shot (image) classification. This idea is extended to few-shot video classification in [164].

Along this line, some methods pay special attention to protecting the few-shot classes in the memory. Note that few-shot classes are small, and so have a lower chance of being kept in M. Each few-shot sample in M can also be easily replaced by samples from the more abundant classes. To alleviate this problem, lifelong memory [65] is proposed. Unlike previous memories [104, 114, 149, 164] which wipe out the memory content across tasks, the lifelong memory erases the "oldest" memory value when the memory is full. The ages of all the memory slots are then reset to zero. For a new sample, when the returned $M_{\text{value}}(i)$ value matches its ground-truth output, it is merged with the current $M_{\text{key}}(i)$ instead of being written to a new memory slot. Hence, it is more likely that all classes occupy an equal number of memory slots, and rare classes are protected. Recently, this lifelong memory is adapted to learn word representations in [125].

However, even with the use of a lifelong memory, rare samples can still be forgotten. After each update, the lifelong memory resets the age of the selected $M(i)$ to zero, and increases the ages of the other non-empty memory slots by one. When the memory is full and the returned value is wrong, the oldest memory slot is replaced. As the rare class samples seldom update their $M(i)$'s, they have a higher chance of being erased.

4.3.2 Refining Parameters. Recall that the Learnet [14] and its variants (Section 4.2.3) map information from D_{train} to parameterize the embedding function $g(\cdot)$ for a new x_{test}. This parameter can be refined using a memory. Meta Networks (MetaNet) [96] parameterizes a classification model using a "slow" weight which is meta-learned from multiple data sets, and a "fast" weight which is a task-specific embedding of D_{train}. As shown in [97], the computation cost of MetaNet can be reduced by learning to modify each neuron rather the complete parameter. MN-Net [22] uses a memory to refine the embedding learned in the Matching Nets, whose output is used to parameterize a CNN as in Learnet.

4.4 Generative Modeling

Generative modeling methods estimate the probability distribution $p(x)$ from the observed x_i's with the help of prior knowledge (Figure 10). Estimation of $p(x)$ usually involves estimations of $p(x|y)$ and $p(y)$. Methods in this class can deal with many tasks, such as generation [34, 76, 107, 109], recognition [34, 35, 47, 76, 113, 129, 159], reconstruction [47], and image flipping [107].

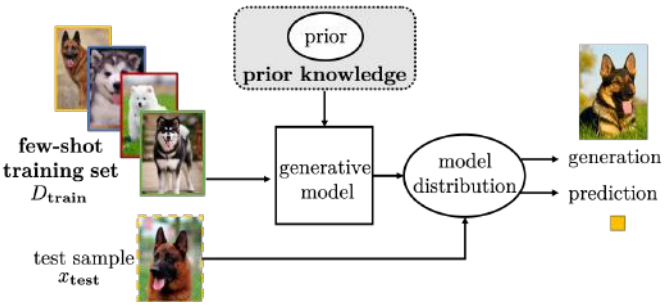


Fig. 10. Solving the FSL problem by generative modeling.

In generative modeling, the observed x is assumed to be drawn from some distribution $p(x;\theta)$ parameterized by θ. Usually, there exists a latent variable $z \sim p(z;\gamma)$, so that $x \sim \int p(x|z;\theta)p(z;\gamma)dz$. The prior distribution $p(z;\gamma)$, which is learned from other data sets, brings in prior knowledge that is vital to FSL. By combining the provided training set D_{train} with this $p(z;\gamma)$, the resultant posterior probability distribution is constrained. In other words, $\mathcal{H}$ is constrained to a much smaller $\tilde{\mathcal{H}}$.

According to what the latent variable z represents, we group these FSL generative modeling methods into three types.

4.4.1 Decomposable Components. Although samples with supervised information are scarce in a FSL problem, they may share some smaller decomposable components with samples from the other tasks. For example, consider the recognition of a person using only a few face photos provided. Although similar faces may be hard to find, one can easily find photos with similar eyes, noses or mouths. With a larger number of samples, models for these decomposable components can be easily learned. One then only needs to find the correct combination of these decomposable components, and decides which target class this combination belongs to. As the decomposable components are chosen by human, this strategy is more interpretable. Bayesian One-Shot [35] uses a generative model to capture the interactions between decomposable components (i.e., shapes and appearances of objects) and target class (i.e., objects to be recognized). Bayesian Program Learning (BPL) [76] models characters by separating it into types, tokens and further templates, parts, primitives. To generate a new character, one needs to search a large combination space containing theses components. In [76], this inference cost is reduced by only considering the top possible combinations. In natural language processing, a recent work [64] models spans instead of the complete parse tree, and adapts parsers between syntactically distant domains by training individual classifiers for spans.

4.4.2 Groupwise Shared Prior. Often, similar tasks have similar prior probabilities, and this can be utilized in FSL. For example, consider the three-class classification of “orange cat", “leopard" and “Bengal tiger". These three species are similar, but Bengal tiger is endangered, while orange cats and leopards are abundant. Hence, one can learn a prior probability from “orange cats" and “leopards", and use this as the prior for the few-shot class “Bengal tiger".

In [113], a set of data sets $\{D_c\}$ are grouped into a hierarchy via unsupervised learning. Data sets in each group together learn the class prior probabilities. For a new few-shot class, one first finds the group this new class belongs to, and then models it by the class prior drawn from the groupwise shared prior probability. In [129], the feature learning step in [113] is further improved with the use of deep Boltzmann machines [112].

4.4.3 Parameters of Inference Networks. To find the best θ, one has to maximize the posterior

$$p(z|x;\theta,\gamma) = \frac{p(x,z;\theta,\gamma)}{p(x;\gamma)} = \frac{p(x|z;\theta)p(z;\gamma)}{\int p(x|z;\theta)p(z;\gamma)dz}. \tag{2}$$

Due to the integral in the denominator, it is intractable to solve (2). A variational distribution $q(z;\delta)$, which is learned from data, is often used to approximate $p(z|x;\theta,\gamma)$. Recently, this $q(z;\delta)$ is approximated via amortized variational inference with the inference network [158]. Although z no longer has semantic meaning, the powerful representation learned by these deep models can lead to better performance. Once learned, the inference network can be applied to a new task directly, which is more efficient and requires less human knowledge. As the inference network has a large number of parameters, it is usually trained using some auxiliary large-scale data sets. Many classic inference networks are adapted to the FSL problem. For example, the variational auto-encoder

(VAE) [68] is used in [34, 57, 109], autoregressive model [135] is used in [107], generative adversarial networks (GAN) [46] is used in [159], and a combination of VAE and GAN is proposed in [47].

4.5 Discussion and Summary

When there exist similar tasks or auxiliary tasks, multitask learning can be used to constrain the $\mathcal{H}$ of the few-shot task. However, note that joint training of all the tasks together is required. Thus, when a new few-shot task arrives, the whole multitask model has to be trained again, which can be costly and slow. Moreover, the sizes of D and D_c should not comparable, otherwise, the few-shot task may be overwhelmed by tasks with many samples.

When there exist a large-scale data set containing sufficient samples of various classes, one can use embedding learning methods. These methods map samples to a good embedding space in which samples from different classes can be well-separated, and so a smaller $\tilde{\mathcal{H}}$ is needed. However, they may not work well when the few-shot task is not closely related to the other tasks. Moreover, more exploration on how to mix the invariant and task-specific information of tasks is helpful.

When a memory network is available, it can be readily used for FSL by training a simple model (e.g., classifier) on top of the memory. By using carefully-designed update rule, one can selectively protect memory slots. The weakness of this strategy is that it incurs additional space and computational costs, which increase with memory size. Therefore, current external memory has a limited size.

Finally, when one wants to perform tasks such as generation and reconstruction besides FSL, generative models can be used. They learn prior probability $p(z;\gamma)$ from the other data sets, which reduces $\mathcal{H}$ to a smaller $\tilde{\mathcal{H}}$. The learned generative models can also be used to generate samples for data augmentation. However, generative modeling methods have high inference cost, and are more difficult to derive than deterministic models.

5 ALGORITHM

The algorithm is the strategy to search in the hypothesis space $\mathcal{H}$ for the parameter θ of the best hypothesis h^* [17, 18]. At the tth iteration, $\theta_t = \theta_{t-1} + \Delta\theta_{t-1}$, where $\Delta\theta_{t-1}$ is the update. For example, for the popular stochastic gradient descent (SGD) and its variants [17, 18], θ is updated as

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \ell(h(x_t;\theta_{t-1}), y_t), \tag{3}$$

where α_t is the stepsize. With θ initialized at θ_0, θ_t can be written as

$$\theta_t = \theta_0 + \sum_{i=1}^{t} \Delta\theta_{i-1}. \tag{4}$$

When supervised information is rich, there are enough training samples to update θ, and to find an appropriate stepsize α by cross-validation. However, in FSL, the provided few-shot D_{train} is not large enough, and the obtained empirical risk minimizer is unreliable.

Methods in this section use prior knowledge to influence how θ is obtained, either by (i) providing a good initialized parameter θ_0, or (ii) directly learning an optimizer to output search steps. In terms of how the search strategy is affected by prior knowledge, we classify methods in this section into three groups (Table 7):

(1) *Refining existing parameters.* An initial θ_0 learned from other tasks, and is then refined using D_{train}.
(2) *Refining meta-learned parameters.* An initial θ_0 is meta-learned from a set of tasks, which are drawn from the same task distribution as the few-shot task, and then further refined by the learner using D_{train}.

(3) *Learning the optimizer.* This strategy learns a meta-learner as optimizer to output search steps for each learner directly, such as changing the search direction or stepsize.

Table 7. Characteristics for FSL methods focusing on the algorithm perspective.

strategy	prior knowledge	how to search θ of the h^* in $\mathcal{H}$
refining existing parameters	learned θ_0	refine θ_0 by D_{train}
refining meta-learned parameters	meta-learner	refine θ_0 by D_{train}
learning the optimizer	meta-learner	use search steps provided by the meta-learner

5.1 Refining Existing Parameters

This strategy takes θ_0 of a pre-trained model learned from related tasks as a good initialization, and adapts it to θ by D_{train}. The assumption is that θ_0 captures some general structures of the large-scale data. Therefore, it can be adapted to D with a few iterations.

5.1.1 Fine-Tuning Existing Parameter by Regularization. This strategy fine-tunes the pre-trained θ_0 for the few-shot task by regularization (Figure 11), and is popularly used in practice. In [21], a CNN pre-trained on the ImageNet for image classification is tuned using a large data set for foreground segmentation, then further fine-tuned using a single shot of segmented object for object segmentation. Given the few-shot D_{train}, simply fine-tuning θ_0 by gradient descent may lead to overfitting. Hence, how to adapt θ_0 without overfitting to D_{train} is a key design issue.

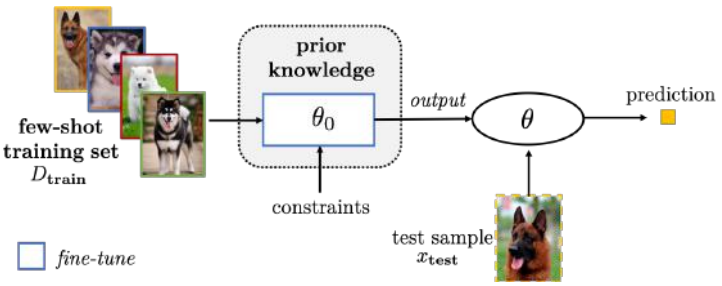


Fig. 11. Solving the FSL problem by fine-tuning existing parameter θ_0 by regularization.

In this section, methods fine-tune θ_0 by regularization to prevent overfitting. They can be grouped as follows:

(1) *Early-stopping.* It requires separating a validation set from D_{train} to monitor the training procedure. Learning is stopped when there is no performance improvement on the validation set [6].
(2) *Selectively updating* θ_0. Only a portion of θ_0 is updated in order to avoid overfitting. For example, in [67], given a set of pre-trained filters, it only learns a strength parameter that is multiplied with the filters.
(3) *Updating related parts of* θ_0 *together.* One can group elements of θ_0 (such as the neurons in a deep neural network), and update each group jointly with the same update information. In [155], the filters of a pre-trained CNN are clustered together according to some auxiliary information, and then fine-tuned by groupwise back-propagation using D_{train}.
(4) *Using a model regression network.* A model regression network [143] captures the task-agnostic transformation which maps the parameter values obtained by training on a few examples

to the parameter values that will be obtained by training on a lot of samples. Similarly, in [72], the transformation function that maps the embedding of x_i to a classification decision boundary is learned.

5.1.2 Aggregating a Set of Parameters. Sometimes, we do not have a suitable θ_0 to start with. Instead, we have many models that are learned from related tasks. For example, in face recognition, we may already have recognition models for the eye, nose, and ear. Therefore, one can aggregate these model parameters to a suitable model, which is then either directly used or refined by D_{train} (Figure 12).

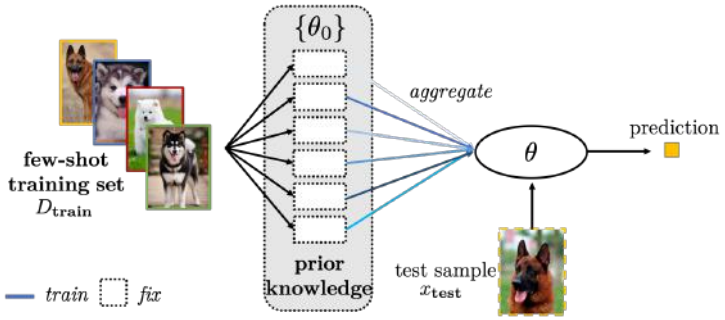


Fig. 12. Solving the FSL problem by aggregating a set of parameters θ_0's into θ. Provided with a set of pre-trained θ_0's, one only needs to learn the combination weights (blue lines).

As discussed in Section 3, samples from unlabeled data sets (Section 3.2) and similar labeled data sets (Section 3.3) can be used to augment the few-shot D_{train}. Instead of using the samples directly, the following methods use models (with parameters θ_0's) pre-trained from these data sets. The problem is then how to adapt them efficiently to the new task using D_{train}.

(1) *Unlabeled data set.* Although there is no supervised information, similar samples can be grouped together. Therefore, one can pre-train functions from the unlabeled data to cluster and separate samples well. A neural network is then used to adapt them to the new task with the few-shot D_{train} [142, 143].
(2) *Similar data sets.* In [10], few-shot object classification is performed by leveraging samples and classifiers from similar classes. First, it replaces the features of samples from these similar classes by features from the new class. The learned classifier is then reused, and only the classification threshold is adjusted for the new class. In [44, 157], they learn to combine existing parameters learned from similar data sets using D_{train}.

5.1.3 Fine-Tuning Existing Parameter with New Parameters. The pre-trained θ_0 may not be enough to encode the new FSL task completely. Hence, an additional parameter(s) δ is used to take the specialty of D_{train} into account (Figure 13). Specifically, this strategy expands the model parameter to become $\theta = \{\theta_0, \delta\}$, and fine-tunes θ_0 while learning δ. In [60], it uses the lower layers of a pre-trained CNN for feature embedding, and learns a linear classifier on the embedded features using D_{train}. In font style transfer [7], a network is first pre-trained to capture the fonts in gray images. To generate stylish colored fonts, this is fine-tuned together with the training of an additional network.

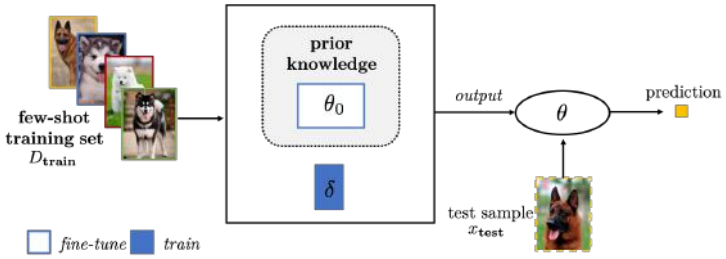


Fig. 13. Solving the FSL problem by fine-tuning existing parameter θ_0 with new parameters.

5.2 Refining Meta-Learned Parameter

Methods in this section use meta-learning to refine the meta-learned parameter θ_0 (Figure 14). The θ_0 is continuously optimized by the meta-learner according to performance of the learner. This is different from Section 5.1 in which θ_0 is fixed.

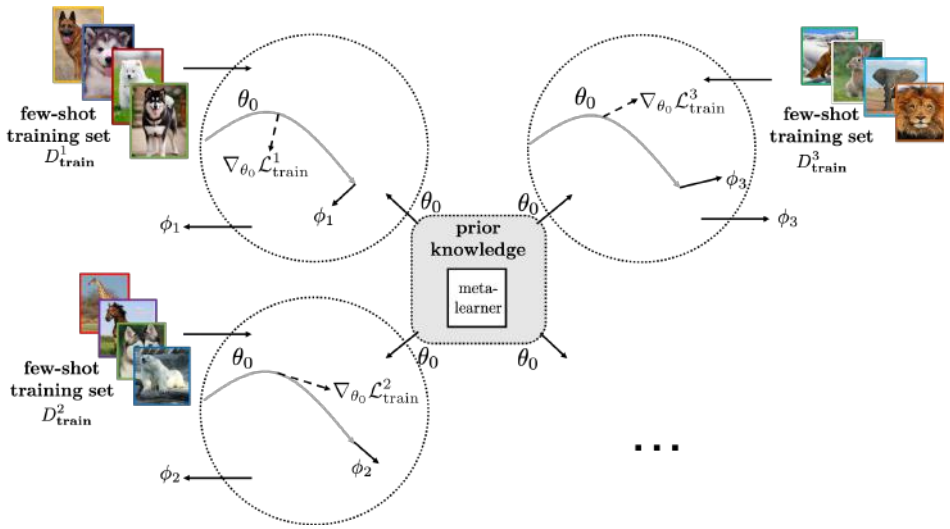


Fig. 14. Solving the FSL problem by refining the meta-learned parameter θ_0.

The meta-learned θ_0 is often refined by gradient descent. A representative method is the Model-Agnostic Meta-Learning (MAML) [37]. It meta-learns θ_0, which is then adjusted to obtain a good task-specific parameter ϕ_s for some $T_s \sim P(T)$ via a few effective gradient descent steps, as: $\phi_s = \theta_0 - \alpha\nabla_{\theta_0}\mathcal{L}^s_{\text{train}}(\theta_0)$. Here, $\mathcal{L}^s_{\text{train}}(\theta_0)$ is the sum of losses over the training samples in D_{train}, and α is the stepsize. Note that ϕ_s is invariant to permutation of the samples. The meta-learned parameter θ_0 is updated by feedbacks from multiple meta-training tasks as $\theta_0 \leftarrow \theta_0 - \beta\nabla_{\theta_0}\sum_{T_s\sim P(T)}\mathcal{L}^s_{\text{test}}(\theta_0)$, where $\mathcal{L}^s_{\text{test}}(\theta_0)$ is the sum of losses over the test samples in D_{test} and β is another stepsize. By continuously refining θ_0 using the few-shot samples in D_{train}, the meta-learner improves its θ_0 to quickly adapt to the few-shot training set.

Recently, many improvements have been proposed for MAML, mainly along the following three aspects:

(1) *Incorporating task-specific information.* MAML provides the same initialization for all tasks. However, this neglects task-specific information, and is appropriate only when the set of tasks are all very similar. To address this problem, in [79], it learns to choose $\{\theta_0\}$ from a subset of a good initialization parameter for a new task.
(2) *Modeling the uncertainty of using a meta-learned θ_0.* Learning with a few examples inevitably results in a model with higher uncertainty [39]. Hence, the learned model may not be able to perform prediction on the new task with high confidence. The ability to measure this uncertainty provides hints for active learning and further data collection [39]. There are works that consider uncertainty for the meta-learned θ_0 [39, 156], uncertainty for the task-specific ϕ_s [48, 105], and uncertainty for class n's class-specific parameter $\phi_{s,n}$ [111].
(3) *Improving the refining procedure.* Refinement by a few gradient descent steps may not be reliable. Regularization can be used to correct the descent direction. In [50], the model regression network [143] is used to regularize task T_s's ϕ_s to be close to the model trained with large-scale samples.

5.3 Learning the Optimizer

In Section 5.2, the meta-learner θ_0 acts as a good initialization for $T \sim P(T)$ with data D, and it is adjusted to a task-specific parameter ϕ via a few effective gradient descent steps. In contrast, instead of using gradient descent, methods in this section learns an optimizer which can directly output the update ($\sum_{i=1}^{t} \Delta\theta^{i-1}$ in (4)) (Figure 15). There is then no need to tune the stepsize α or find the search direction, as the learning algorithm does that automatically.

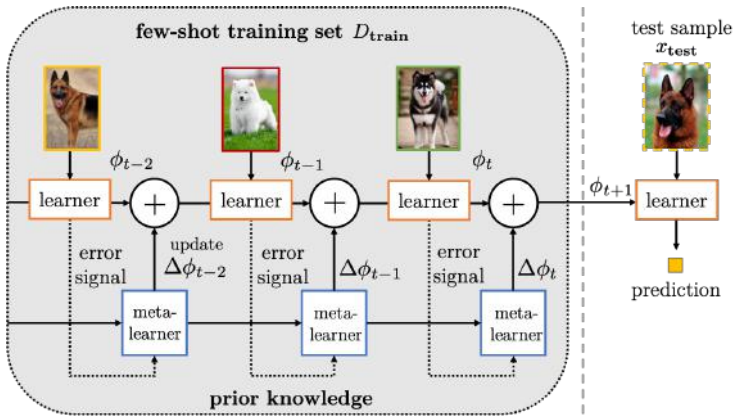


Fig. 15. Solving the FSL problem by learning the optimizer.

At the tth iteration, this line of works [5, 106] learn an meta-learner which takes the error signal computed at the $(t-1)$th iteration, and directly outputs update $\Delta\phi_{t-1}$ to update the task-specific parameter ϕ_{t-1} of the learner as $\phi_t = \phi_{t-1} + \Delta\phi_{t-1}$. Therefore, in contrast to strategies mentioned in Sections 5.1 and 5.2, this strategy provides an optimizer for the new task (which is in turn optimized by the learner). This ϕ_t is then used to compute the loss $\ell_t(\phi_t) = \ell(h(x_t; \phi_t), y_t)$ using the tth sample $(x_t, y_t) \in D_{\text{train}}$, which acts as the error signal to be fed into the meta-learner at the next iteration. After learning a task, the meta-learner is improved by gradient descent on the loss on the test set D_{test}. By learning from a set of T_s's drawn from $P(T)$, the meta-learner improves on proposing efficient algorithms for FSL. Recently, [106] obtains ϕ_t by instantiating (3) with the cell state update in the LSTM (where ϕ is set to the cell state of the LSTM).

5.4 Discussion and Summary

Refining existing parameters can reduce the search effort in $\mathcal{H}$. By using an existing θ_0 as initialization, these methods usually need a lower computation cost to obtain a good hypothesis $h \in \mathcal{H}$. Learning focuses on refining these existing parameters. However, as θ_0 is learned from tasks different from the current task, this strategy may sacrifice precision for speed.

The other two strategies rely on meta-learning. By learning from a set of related tasks, the meta-learned θ_0 can be closer to the task-specific parameter ϕ_t for a new task T_t. Learning search steps by a meta-learner can directly guide the learning algorithm. In other words, the meta-learner acts as an optimizer. However, important issues such as how to meta-learn across different granularities (such as coarse-grained classifications of animals versus fine-grained classification of dog species) or different data sources (such as images versus texts) [131] are still open. From this perspective, meta-learning and multi-tasks are similar, and so there is also a concern on how to avoid negative transfer [28]

6 FUTURE WORKS

In this section, we discuss four key directions for the further development of FSL, namely, (i) problem setups, (ii) techniques, (iii) applications and (iv) theories.

6.1 Problem Setups

Existing FSL methods often use prior knowledge from one single modality (such as images, texts, or videos). However, though D_{train} has a few examples for the modality currently used, there may exist another modality in which supervised samples are abundant. An example is in the study of extinct animals. While this animal species may only have a limited number of visual examples, there might be a lot of information about it in the textual domain (such as textbooks or web pages), as people tend to pay special attention to the rare class. Therefore, prior knowledge from multiple modalities can provide prior knowledge for complementary views. In zero-shot learning (ZSL), multi-modality data has been frequently used. Example prior information are attributes [2, 62], WordNet [2, 62], word embeddings [133, 139], co-occurrence statistics [88], and knowledge graphs [140].

Recently, there have been efforts in borrowing techniques from ZSL methods to FSL problems. For example, one can use the few-shot D_{train} to fine-tune the parameters learned by ZSL methods [2, 62]. However, fine-tuning using a small number of samples may lead to overfitting. Another possibility is to force the embedding learned by multiple modalities to match in a shared space [133, 139]. A recent work [110] exploits the structured relationships among labels and utilizes a GNN to align the embedding for FSL. As different modalities may contain different structures, this should be carefully handled. For example, texts need to obey syntactic structures while images do not. In the future, a promising direction is to consider the use of multi-modality information in designing FSL methods.

6.2 Techniques

In previous sections, according to how the prior knowledge in FSL is used, we categorize FSL methods from the perspectives of data (Section 3), model (Section 4), and algorithm (Section 5). Each of these components can be improved. For example, using state-of-the-art ResNet [55] as the embedding function can be better than using the VGG [123].

Meta-learning-based FSL methods, as reviewed in Sections 4 and 5, are particularly interesting. By learning across tasks, meta-learning can adapt to new tasks rapidly with a small inference cost. However, the tasks considered in meta-learning are often assumed to be drawn from a single

task distribution $p(T)$. In practice, we can have a large number of tasks whose task relatedness is unknown or expensive to determine. In this case, directly learning from all these tasks can lead to negative transfer [28]. Besides, current FSL methods often consider a static and fixed $P(T)$ [37, 106]. However, in streaming applications, $p(T)$ is dynamic [38] and new tasks are continually arriving. Hence, this should also be incorporated into $p(T)$. An important issue is how to avoid catastrophic forgetting [69] in a dynamic setting, which means that information on the old tasks should not be forgotten.

As discussed in previous sections, different FSL methods have pros and cons, and there is no absolute winner in all settings. Moreover, both the hypothesis space $\mathcal{H}$ and search strategies in $\mathcal{H}$ often rely on human design. *Automated machine learning* (AutoML) [153], by constructing task-aware machine learning models, has achieved state-of-the-art on many applications. Recently, AutoML has been used on data augmentation [27]. Another direction is to extend the AutoML methods of automated feature engineering [66], model selection [71] and neural architecture search [166] to FSL. One can then obtain better algorithm designs whose components are learned by AutoML in an economic, efficient and effective manner.

6.3 Applications

Recall that FSL is needed due to rareness of samples, endeavor to reduce data gathering effort and computational cost, or as a stepping stone to mimic human-like learning. Hence, many real-world applications involve FSL. Computer vision is one of very first testbed for FSL algorithms. FSL has also attracted a lot of recent attention in many other applications, such as robotics, natural language processing, and acoustic signal processing. In summary, there are many interesting fields and applications for FSL to explore.

6.3.1 Computer Vision. Most existing works target FSL problems in computer vision. The two most popular applications are character recognition [14, 36, 37, 65, 70, 96, 113, 114, 119, 121, 130, 138, 146] and image classification [37, 70, 96, 106, 119, 121, 127, 130, 132, 138, 142, 143, 149]. Very high accuracies have already been obtained on the standard benchmark data sets (such as Ominiglot and miniImageNet), leaving little space for further improvement [131]. Recently, a large and diverse benchmark data set, constructed from multiple image data sources, is presented in [131]. Besides character recognition and image classification, other image applications have also been considered. These include object recognition [35, 36, 82], font style transfer [7], phrase grounding [162], image retrieval [130], object tracking [14], specific object counting in images [162], scene location recognition [74], gesture recognition [102], part labeling [24], image generation [34, 76, 107, 109], image translation across domains [12], shape view reconstruction for 3D objects [47], and image captioning and visual question answering [31].

FSL has also been successfully used in video applications, including motion prediction [50], video classification [164], action localization [152], person re-identification [148], event detection [151], and object segmentation [21].

6.3.2 Robotics. In order for robots to behave more like human, they should be able to generalize from a few demonstrations. Hence, FSL has played an important role in robotics. For example, learning of robot arm movement using imitating learning from a single demonstration [147], and learning manipulation actions from a few demonstrations with the help of a teacher who corrects the false actions [1].

Apart from imitating users, robots can improve their behavior through interacting with users. Recently, assistive strategies are learned from a few interactions through FSL reinforcement learning [51]. Other examples of FSL in robotics include multi-armed bandits [33], visual navigation [33, 37],

and continuous control [37, 91, 156]. Recently, these applications are further extended to dynamic environments [3, 98].

6.3.3 Natural Language Processing. Recently, the use of FSL has drawn attention in natural language processing. Example applications include parsing [64], translation [65], sentence completion (which fills in the blanks using a word chosen from a provided set) [97, 138], sentiment classification from short reviews [150, 157], user intent classification for dialog systems [157], criminal charge prediction [61], word similarity tasks such as nonce definition [56, 125], and multi-label text classification [110]. Recently, a new relation classification data set called FewRel [52] is released. This compensates for the lack of benchmark data set for FSL tasks in natural language processing.

6.3.4 Acoustic Signal Processing. Apart from the early efforts on using FSL to recognize spoken words from one example [75], recent endeavors are on voice synthesis. A popular task is voice cloning from a few audio samples of the user [6]. This can be useful in generating personal voice navigation in map applications, or mimicking the parents' voice in story-telling to kids in a smart home toolkit. Recently, it is possible to perform voice conversion from one user to another using one-shot voice or text sample [128] or even across different languages [93].

6.3.5 Others. For example, a recent attempt in the context of medical applications is few-shot drug discovery [4]. For learning of deep networks, one-shot architecture search (OAS) is studied in [19, 83, 154]. Unlike random search and grid search which require multiple runs to find the best architecture, OAS methods can find good architectures by training the supernet once. FSL has also been used in curve fitting [39, 48, 114, 156] and understanding number analogy by logic reasoning to perform calculations [104].

6.4 Theories

FSL uses prior knowledge to compensate for the lack of supervised information. This is related to the theoretical study of sample complexity, which is the number of training samples needed to obtain a model with small empirical risk $R_I(h)$ having high probability [92, 94]. $\mathcal{H}$ needs to be less complicated to make the provided I samples enough. Recall that FSL methods use prior knowledge to augment more samples (i.e., increase I), constrain $\mathcal{H}$ (i.e., reduce the complexity of $\mathcal{H}$) and alter the search strategy (i.e., increase the probability of finding a good h). This suggests that FSL methods can reduce the required sample complexity using prior knowledge. A detailed analysis on this aspect will be useful.

Besides, recall that FSL is related to domain adaptation [85, 95, 102], and existing theoretical bounds on domain adaptation may be inspiring [11, 16]. For example, recent analysis shows that better risk bounds can be obtained by fine-tuning feedforward neural networks [87]. By considering a specific meta-learning method, the risk of transferring a model trained on one task to another task is examined in [29]. However, only a small number of methods have been studied so far. There are still a lot of theoretical issues to explore.

Finally, convergence of the FSL algorithms is not fully understood. In particular, meta-learning methods optimize θ over a task distribution instead of over a single task. Recent analysis in [40] provides sufficient conditions for convergence of one meta-learning method. The meta-learner learns the lower layers of a deep network, while the learner learns the last layer, all using gradient descent. A more general analysis on the convergence of meta-learning methods will be highly useful.

7 CONCLUSION

Few-Shot Learning (FSL) targets at bridging the gap between AI and human learning. It can learn new tasks containing only a few examples with supervised information by incorporating prior knowledge. FSL acts as a test-bed for AI, makes the learning of rare cases possible, or helps to relieve the burden of collecting large-scale supervised date in industrial applications. In this survey, we provide a comprehensive and systematic review of FSL. We first formally define FSL, and discuss the relatedness and differences of FSL with relevant learning problems such as weakly supervised learning, imbalanced learning, transfer learning and meta-learning. We then point out the core issue of FSL is the unreliable empirical risk minimizer that makes FSL hard to learn. Understanding the core issue helps categorize different works into data, model and algorithm according to how they solve the core issue using prior knowledge: data augments the supervised experience of FSL, model constrains the hypothesis space of FSL to be smaller, and algorithm alters the search strategy for the best hypothesis in the given hypothesis space. In each category, the pros and cons are thoroughly discussed and some summary and insights are presented. To inspire future research in FSL, we also provide possible directions on problem setups, techniques, applications and theories to explore.

A APPENDIX: META-LEARNING

Meta-learning [59] improves P of the new task T by the provided data set and the meta-knowledge extracted across tasks by a meta-learner (Figure 16). Let $p(T)$ be the distribution of task T. In mete-training, it learns from a set of tasks $T_s \sim p(T)$. Each task T_s operates on data set D_s of N classes, where $D_s = \{D^s_{\text{train}}, D^s_{\text{test}}\}$ consists of a training set D^s_{train} and a test set D^s_{test}. Each learner learns from D^s_{train} and measures the test error on D^s_{test}. The parameter θ_0 of meta-learner is optimized to minimize the error across all learners, as:

$$\theta_0 = \arg\min_{\theta_0} \mathbb{E}_{T_s \sim p(T)} \sum\nolimits_{((x_i, y_i)) \in D_s} \ell(h(x_i; \theta_0), y_i).$$

In meta-testing, another disjoint set of tasks $T_t \sim p(T)$ is used to test the generalization ability of the meta-learner. Each T_t works on a data set D_t of N' classes, where $D_t = \{D^t_{\text{train}}, D^t_{\text{test}}\}$. The learner learns from the training set D^t_{train} and evaluates on the test set D^t_{test}. The loss averaged across T_t's is taken as the meta-learning testing error.

ACKNOWLEDGMENTS

This research is partially done in 4Paradigm Inc. when Yaqing Wang took the internship.

REFERENCES

[1] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss. 2013. Learning manipulation actions from a few demonstrations. In *International Conference on Robotics and Automation*. 1268–1275.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. 2013. Label-embedding for attribute-based classification. In *Conference on Computer Vision and Pattern Recognition*. 819–826.

[3] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. 2018. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*.

[4] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande. 2017. Low data drug discovery with one-shot learning. *ACS Central Science* 3, 4 (2017), 283–293.

[5] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*. 3981–3989.

[6] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou. 2018. Neural voice cloning with a few samples. In *Advances in Neural Information Processing Systems*. 10019–10029.

[7] S. Azadi, M. Fisher, V. G. Kim, Z. Wang, E. Shechtman, and T. Darrell. 2018. Multi-content GAN for few-shot font style transfer. In *Conference on Computer Vision and Pattern Recognition*. 7564–7573.

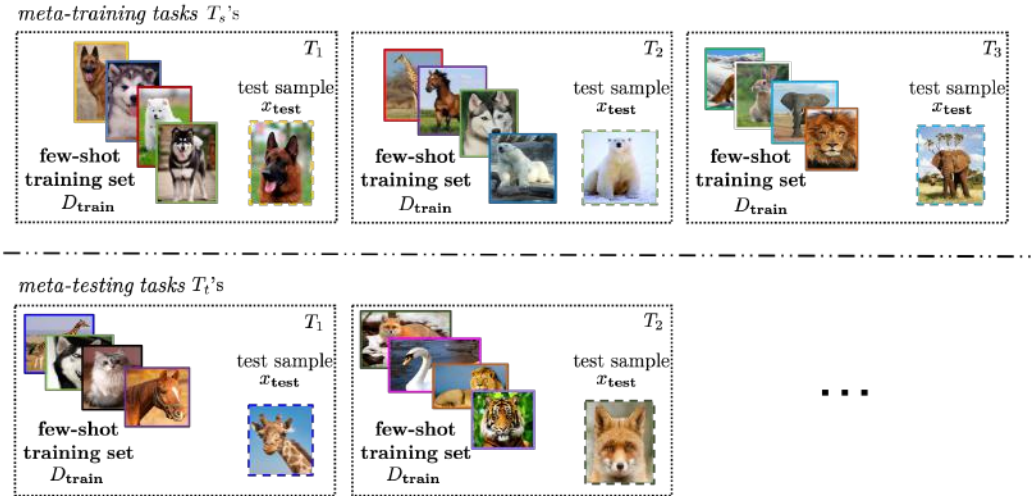


Fig. 16. Solving the FSL problem by meta-learning.

[8] P. Bachman, A. Sordoni, and A. Trischler. 2017. Learning algorithms for active learning. In *International Conference on Machine Learning*. 301–310.

[9] Bengio Y. Bahdanau D, Cho K. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

[10] E. Bart and S. Ullman. 2005. Cross-generalization: Learning novel classes from a single example by feature replacement. In *Conference on Computer Vision and Pattern Recognition*, Vol. 1. 672–679.

[11] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*. 137–144.

[12] S. Benaim and L. Wolf. 2018. One-shot unsupervised cross domain translation. In *Advances in Neural Information Processing Systems*. 2104–2114.

[13] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. 2019. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*.

[14] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. 2016. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*. 523–531.

[15] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[16] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. 2008. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems*. 129–136.

[17] L. Bottou and O. Bousquet. 2008. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*. 161–168.

[18] L. Bottou, F. E. Curtis, and J. Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Rev.* 60, 2 (2018), 223–311.

[19] A. Brock, T. Lim, J.M. Ritchie, and N. Weston. 2018. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*.

[20] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. 1994. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*. 737–744.

[21] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. 2017. One-shot video object segmentation. In *Conference on Computer Vision and Pattern Recognition*. 221–230.

[22] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei. 2018. Memory matching networks for one-shot image recognition. In *Conference on Computer Vision and Pattern Recognition*. 4080–4088.

[23] R. Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[24] J. Choi, J. Krishnamurthy, A. Kembhavi, and A. Farhadi. 2018. Structured set matching networks for one-shot part labeling. In *Conference on Computer Vision and Pattern Recognition*. 3627–3636.

[25] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine. 2019. Meta-learning language-guided policy learning. In *International Conference on Learning Representations*.

[26] J. J. Craig. 2009. *Introduction to Robotics: Mechanics and Control.* Pearson Education India.

[27] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. 2019. AutoAugment: Learning augmentation policies from data. In *Conference on Computer Vision and Pattern Recognition*. 113–123.

[28] T. Deleu and Y. Bengio. 2018. The effects of negative adaptation in Model-Agnostic Meta-Learning. *arXiv preprint arXiv:1812.02159* (2018).

[29] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. 2018. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems*. 10190–10200.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*. 248–255.

[31] X. Dong, L. Zhu, D. Zhang, Y. Yang, and F. Wu. 2018. Fast parameter adaptation for few-shot image captioning and visual question answering. In *ACM International Conference on Multimedia*. 54–62.

[32] M. Douze, A. Szlam, B. Hariharan, and H. Jégou. 2018. Low-shot learning with large-scale diffusion. In *Conference on Computer Vision and Pattern Recognition*. 3349–3358.

[33] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. 2017. One-shot imitation learning. In *Advances in Neural Information Processing Systems*. 1087–1098.

[34] H. Edwards and A. Storkey. 2017. Towards a neural statistician. In *International Conference on Learning Representations*.

[35] L. Fei-Fei, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 4 (2006), 594–611.

[36] M. Fink. 2005. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems*. 449–456.

[37] C. Finn, P. Abbeel, and S. Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. 1126–1135.

[38] C. Finn and S. Levine. 2018. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*.

[39] C. Finn, K. Xu, and S. Levine. 2018. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*. 9537–9548.

[40] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*. 1563–1572.

[41] J. Friedman, T. Hastie, and R. Tibshirani. 2001. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics New York.

[42] H. Gao, Z. Shou, A. Zareian, H. Zhang, and S. Chang. 2018. Low-shot learning via covariance-preserving adversarial augmentation networks. In *Advances in Neural Information Processing Systems*. 983–993.

[43] P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien. 2016. PAC-Bayesian theory meets Bayesian inference. In *Advances in Neural Information Processing Systems*. 1884–1892.

[44] S. Gidaris and N. Komodakis. 2018. Dynamic few-shot visual learning without forgetting. In *Conference on Computer Vision and Pattern Recognition*. 4367–4375.

[45] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

[47] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner. 2019. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*.

[48] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations*.

[49] A. Graves, G. Wayne, and I. Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401* (2014).

[50] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. Moura. 2018. Few-shot human motion prediction via meta-learning. In *European Conference on Computer Vision*. 432–450.

[51] M. Hamaya, T. Matsubara, T. Noda, T. Teramae, and J. Morimoto. 2016. Learning assistive strategies from a few user-robot interactions: Model-based reinforcement learning approach. In *International Conference on Robotics and Automation*. 3346–3351.

[52] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Conference on Empirical Methods in Natural Language Processing*. 4803–4809.

[53] B. Hariharan and R. Girshick. 2017. Low-shot visual recognition by shrinking and hallucinating features. In *International Conference on Computer Vision*.

[54] H. He and E. A. Garcia. 2008. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 9 (2008), 1263–1284.

[55] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*. 770–778.
[56] A. Herbelot and M. Baroni. 2017. High-risk learning: Acquiring new word vectors from tiny data. In *Conference on Empirical Methods in Natural Language Processing*. 304–309.
[57] L. B. Hewitt, M. I. Nye, A. Gane, T. Jaakkola, and J. B. Tenenbaum. 2018. The variational homoencoder: Learning to learn high capacity generative models from few examples. In *Uncertainty in Artificial Intelligence*. 988–997.
[58] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[59] S. Hochreiter, A. S. Younger, and P. R. Conwell. 2001. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*. 87–94.
[60] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. 2013. One-shot adaptation of supervised deep convolutional models. In *International Conference on Learning Representations*.
[61] Z. Hu, X. Li, C. Tu, Z. Liu, and M. Sun. 2018. Few-shot charge prediction with discriminative legal attributes. In *International Conference on Computational Linguistics*. 487–498.
[62] S. J. Hwang and L. Sigal. 2014. A unified semantic embedding: Relating taxonomies and attributes. In *Advances in Neural Information Processing Systems*. 271–279.
[63] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*. 675–678.
[64] V. Joshi, M. Peters, and M. Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Annual Meeting of the Association for Computational Linguistics*. 1190–1199.
[65] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio. 2017. Learning to remember rare events. In *International Conference on Learning Representations*.
[66] J. M. Kanter and K. Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *International Conference on Data Science and Advanced Analytics*. 1–10.
[67] R. Keshari, M. Vatsa, R. Singh, and A. Noore. 2018. Learning structure and strength of CNN filters for small sample size training. In *Conference on Computer Vision and Pattern Recognition*. 9349–9358.
[68] D. P. Kingma and M. Welling. 2014. Auto-encoding variational Bayes. In *International Conference on Learning Representations*.
[69] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences* 114, 13 (2017), 3521–3526.
[70] G. Koch. 2015. *Siamese neural networks for one-shot image recognition*. Ph.D. Dissertation. University of Toronto.
[71] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. 2017. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research* 18, 1 (2017), 826–830.
[72] J. Kozerawski and M. Turk. 2018. CLEAR: Cumulative learning for one-shot one-class image recognition. In *Conference on Computer Vision and Pattern Recognition*. 3446–3455.
[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.
[74] R. Kwitt, S. Hegenbart, and M. Niethammer. 2016. One-shot learning of scene locations via feature trajectory transfer. In *Conference on Computer Vision and Pattern Recognition*. 78–86.
[75] B. Lake, C.-Y. Lee, J. Glass, and J. Tenenbaum. 2014. One-shot learning of generative speech concepts. In *Annual Meeting of the Cognitive Science Society*, Vol. 36.
[76] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 6266 (2015), 1332–1338.
[77] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences* 40 (2017).
[78] C. H. Lampert, H. Nickisch, and S. Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *Conference on Computer Vision and Pattern Recognition*. 951–958.
[79] Y. Lee and S. Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*. 2933–2942.
[80] K. Li and J. Malik. 2017. Learning to optimize. In *International Conference on Learning Representations*.
[81] X.-L. Li, P. S. Yu, B. Liu, and S.-K. Ng. 2009. Positive unlabeled learning for data stream classification. In *SIAM International Conference on Data Mining*. 259–270.
[82] B. Liu, X. Wang, M. Dixit, R. Kwitt, and N. Vasconcelos. 2018. Feature space transfer for data augmentation. In *Conference on Computer Vision and Pattern Recognition*. 9090–9098.
[83] H. Liu, K. Simonyan, and Y. Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

[84] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, and Y Yang. 2019. Learning to propopagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*.
[85] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei. 2017. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*. 165–177.
[86] S. Mahadevan and P. Tadepalli. 1994. Quantifying prior determination knowledge using the PAC learning model. *Machine Learning* 17, 1 (1994), 69–105.
[87] D. McNamara and M.-F. Balcan. 2017. Risk bounds for transferring representations with and without fine-tuning. In *International Conference on Machine Learning*. 2373–2381.
[88] T. Mensink, E. Gavves, and C. Snoek. 2014. Costa: Co-occurrence statistics for zero-shot classification. In *Conference on Computer Vision and Pattern Recognition*. 2441–2448.
[89] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. 2016. Key-value memory networks for directly reading documents. In *Conference on Empirical Methods in Natural Language Processing*. 1400–1409.
[90] E. G. Miller, N. E. Matsakis, and P. A. Viola. 2000. Learning from one example through shared densities on transforms. In *Conference on Computer Vision and Pattern Recognition*, Vol. 1. 464–471.
[91] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.
[92] M. T. Mitchell. 1997. *Machine Learning*. McGraw-Hill.
[93] S. H. Mohammadi and T. Kim. 2018. Investigation of using disentangled and interpretable representations for one-shot cross-lingual voice conversion. In *INTERSPEECH*. 2833–2837.
[94] M. Mohri, A. Rostamizadeh, and A. Talwalkar. 2018. *Foundations of machine learning*. MIT Press.
[95] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto. 2017. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*. 6670–6680.
[96] T. Munkhdalai and H. Yu. 2017. Meta networks. In *International Conference on Machine Learning*. 2554–2563.
[97] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler. 2018. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*. 3661–3670.
[98] A. Nagabandi, C. Finn, and S. Levine. 2018. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*.
[99] H. Nguyen and L. Zakynthinou. 2018. Improved algorithms for collaborative PAC learning. In *Advances in Neural Information Processing Systems*. 7631–7639.
[100] B. Oreshkin, P. R. López, and A. Lacoste. 2018. TADAM: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*. 719–729.
[101] S. J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 10, 22 (2010), 1345–1359.
[102] T. Pfister, J. Charles, and A. Zisserman. 2014. Domain-adaptive discriminative one-shot learning of gestures. In *European Conference on Computer Vision*. 814–829.
[103] H. Qi, M. Brown, and D. G. Lowe. 2018. Low-shot learning with imprinted weights. In *Conference on Computer Vision and Pattern Recognition*. 5822–5830.
[104] T. Ramalho and M. Garnelo. 2019. Adaptive posterior learning: Few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*.
[105] S. Ravi and A. Beatson. 2019. Amortized Bayesian meta-learning. In *International Conference on Learning Representations*.
[106] S. Ravi and H. Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.
[107] S. Reed, Y. Chen, T. Paine, A. van den Oord, S. M. A. Eslami, D. Rezende, O. Vinyals, and N. de Freitas. 2018. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *International Conference on Learning Representations*.
[108] M. Ren, S. Ravi, E. Triantafillou, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. 2018. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*.
[109] D. Rezende, I. Danihelka, K. Gregor, and D. Wierstra. 2016. One-shot generalization in deep generative models. In *International Conference on Machine Learning*. 1521–1529.
[110] A. Rios and R. Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Conference on Empirical Methods in Natural Language Processing*. 3132.
[111] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.
[112] R. Salakhutdinov and G. Hinton. 2009. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*. 448–455.

[113] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. 2012. One-shot learning with a hierarchical nonparametric Bayesian model. In *ICML Workshop on Unsupervised and Transfer Learning*. 195–206.
[114] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*. 1842–1850.
[115] V. G. Satorras and J. B. Estrach. 2018. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*.
[116] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein. 2018. Delta-encoder: An effective sample synthesis method for few-shot object recognition. In *Advances in Neural Information Processing Systems*. 2850–2860.
[117] B. Settles. 2009. *Active learning literature survey.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences.
[118] J. Shu, Z. Xu, and D Meng. 2018. Small sample learning in big data era. *arXiv preprint arXiv:1808.04572* (2018).
[119] P. Shyam, S. Gupta, and A. Dukkipati. 2017. Attentive recurrent comparators. In *International Conference on Machine Learning*. 3173–3181.
[120] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.
[121] J. Snell, K. Swersky, and R. S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. 4077–4087.
[122] M. D. Spivak. 1970. *A Comprehensive Introduction to Differential Geometry.* Publish or Perish.
[123] R. K. Srivastava, K. Greff, and J. Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*. 2377–2385.
[124] S. Sukhbaatar, J. Weston, R. Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. 2440–2448.
[125] J. Sun, S. Wang, and C. Zong. 2018. Memory, show the way: Memory based few shot word representation learning. In *Conference on Empirical Methods in Natural Language Processing*. 1435–1444.
[126] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*. 1199–1208.
[127] K. D. Tang, M. F. Tappen, R. Sukthankar, and C. H. Lampert. 2010. Optimizing one-shot recognition with micro-set learning. In *Conference on Computer Vision and Pattern Recognition*. 3027–3034.
[128] A. Tjandra, S. Sakti, and S. Nakamura. 2018. Machine speech chain with one-shot speaker adaptation. In *INTERSPEECH*. 887–891.
[129] A. Torralba, J. B. Tenenbaum, and R. R. Salakhutdinov. 2011. Learning to learn with compound HD models. In *Advances in Neural Information Processing Systems*. 2061–2069.
[130] E. Triantafillou, R. Zemel, and R. Urtasun. 2017. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*. 2255–2265.
[131] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, et al. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096* (2019).
[132] Y.-H. Tsai, L.-K. Huang, and R. Salakhutdinov. 2017. Learning robust visual-semantic embeddings. In *Conference on Computer Vision and Pattern Recognition*. 3571–3580.
[133] Y. H. Tsai and R. Salakhutdinov. 2017. Improving one-shot learning through fusing side information. *arXiv preprint arXiv:1710.08347* (2017).
[134] M. A. Turing. 1950. Computing machinery and intelligence. *Mind* 59, 236 (1950), 433–433.
[135] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. 2016. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*. 4790–4798.
[136] V. N. Vapnik. 1992. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*. 831–838.
[137] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *Advances in Neural Information Processing Systems*. 6904–6914.
[138] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. 3630–3638.
[139] P. Wang, L. Liu, C. Shen, Z. Huang, A. van den Hengel, and H. Tao Shen. 2017. Multi-attention network for one shot learning. In *Conference on Computer Vision and Pattern Recognition*. 2721–2729.
[140] X. Wang, Y. Ye, and A. Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Conference on Computer Vision and Pattern Recognition*. 6857–6866.

[141] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. 2018. Low-shot learning from imaginary data. In *Conference on Computer Vision and Pattern Recognition*. 7278–7286.
[142] Y.-X. Wang and M. Hebert. 2016. Learning from small sample sets by combining unsupervised meta-training with CNNs. In *Advances in Neural Information Processing Systems*. 244–252.
[143] Y.-X. Wang and M. Hebert. 2016. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*. 616–634.
[144] J. Wei and K. Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*. 6383–6389.
[145] J. Weston, S. Chopra, and A. Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).
[146] M. Woodward and C. Finn. 2017. Active one-shot learning. *arXiv preprint arXiv:1702.06559* (2017).
[147] Y. Wu and Y. Demiris. 2010. Towards one shot learning by imitation for humanoid robots. In *International Conference on Robotics and Automation*. 2889–2894.
[148] Y. Wu, Y. Lin, X. Dong, Y. Yan, W. Ouyang, and Y. Yang. 2018. Exploit the unknown gradually: One-shot video-based person re-identification by stepwise learning. In *Conference on Computer Vision and Pattern Recognition*. 5177–5186.
[149] Z. Xu, L. Zhu, and Y. Yang. 2017. Few-shot object recognition from machine-labeled web images. In *Conference on Computer Vision and Pattern Recognition*. 1164–1172.
[150] L. Yan, Y. Zheng, and J. Cao. 2018. Few-shot learning for short text classification. *Multimedia Tools and Applications* 77, 22 (2018), 29799–29810.
[151] W. Yan, J. Yap, and G. Mori. 2015. Multi-task transfer methods to improve one-shot learning for multimedia event detection. In *British Machine Vision Conference*.
[152] H. Yang, X. He, and F. Porikli. 2018. One-shot action localization by learning sequence matching network. In *Conference on Computer Vision and Pattern Recognition*. 1450–1459.
[153] Q. Yao, M. Wang, E. H. Jair, I. Guyon, Y.-Q. Hu, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu. 2018. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306* (2018).
[154] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu. 2020. Efficient neural architecture search via proximal iterations. In *AAAI Conference on Artificial Intelligence*.
[155] D. Yoo, H. Fan, V. N. Boddeti, and K. M. Kitani. 2018. Efficient k-shot learning with regularized deep networks. In *AAAI Conference on Artificial Intelligence*.
[156] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. 2018. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*. 7343–7353.
[157] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1206–1215.
[158] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. 2019. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 8 (2019), 2008–2026.
[159] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song. 2018. MetaGAN: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*. 2371–2380.
[160] Y. Zhang, H. Tang, and K. Jia. 2018. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *European Conference on Computer Vision*. 233–248.
[161] Y. Zhang and Q. Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* (2017).
[162] F. Zhao, J. Zhao, S. Yan, and J. Feng. 2018. Dynamic conditional networks for few-shot learning. In *European Conference on Computer Vision*.
[163] Z.-H. Zhou. 2017. A brief introduction to weakly supervised learning. *National Science Review* 5, 1 (2017), 44–53.
[164] L. Zhu and Y. Yang. 2018. Compound memory networks for few-shot video classification. In *European Conference on Computer Vision*. 751–766.
[165] X. J. Zhu. 2005. *Semi-supervised learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.
[166] B. Zoph and Q. V. Le. 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*.