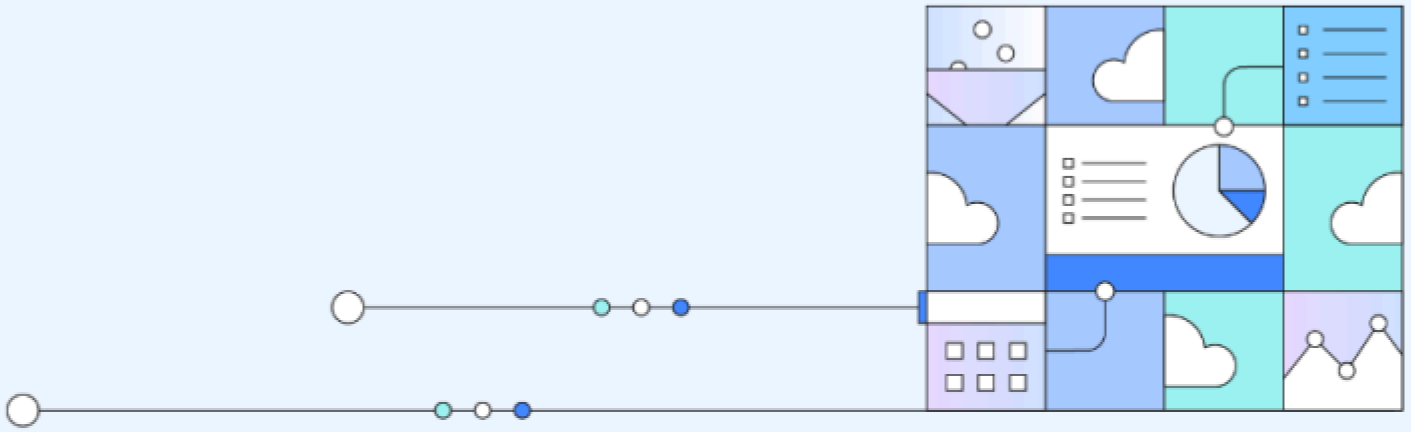
# What is few-shot learning?



Explore our AI platform →

Subscribe for AI updates →

## What is few-shot learning?

Few-shot learning is a machine learning framework in which an AI model learns to make accurate predictions by training on a very small number of labeled examples. It's typically used to train models for classification tasks when suitable training data is scarce.

Few-shot learning (FSL) is a subset of what is sometimes referred to more generally as *n-shot learning*, a category of artificial intelligence that also includes *one-shot learning* (in which there is only one labeled example of each class to be learned) and *zero-shot learning* (in which there are no labeled examples at all). While one-shot learning is essentially just a challenging variant of FSL, zero-shot learning is a distinct learning problem that necessitates its own unique methodologies.

In principle, FSL aims to emulate the human ability to learn from a mere handful of examples. This stands in contrast to conventional supervised learning, which typically uses many hundreds (or thousands) of labeled data points across many rounds of training to teach AI models to recognize classes of data. While powerful, supervised learning is impractical in some real-world settings: obtaining labeled examples is often difficult due to prohibitive costs, the domain-specific expertise needed to annotate data correctly or—in scenarios like unique handwriting, rare diseases or endangered and newly discovered species—the scarcity of existing samples.

While certain specific algorithms and neural network architectures have achieved notable success at FSL tasks, few-shot learning is defined by the nature of the learning problem rather than by the use of any specific method or model structure. Few-shot learning methods range widely, from adapting pre-trained models for use in similar tasks to using generative models to create new samples to *meta learning* methods that train models to generalize well to new classification problems and different classes of data, rather than perform any one specific task.

Related content

Register for the guide on foundation models

# How does few-shot classification work?

Though few-shot learning can utilize a wide variety of algorithms or neural network architectures, most methods are built around *transfer learning* or *meta learning* (or a combination of both).

While few-shot learning can also be applied to regression tasks (or even reinforcement learning), most FSL literature focuses on classification use cases. Some FSL methods can used alongside other solutions that address scarcity of labeled data: for example, in semi-supervised learning methods that incorporate information from large amounts of unlabeled data alongside information from few-shot learning on the limited labeled samples available.[1]

## Transfer learning

Transfer learning-based methods focus on adapting a pre-trained model to learn new tasks or previously unseen classes of data.

When few labeled samples are available, using supervised learning to train a model from scratch—especially one with a large number of parameters, like the convolutional neural networks (CNNs) typically used in computer vision or the transformer-based networks used in natural language processing (NLP)—often leads to overfitting: the model might perform well on test data, but poorly on real-world data. But gathering a sufficiently large amount of data to avoid overfitting is often a bottleneck in model training.

Transfer learning offers a practical solution: leverage useful features and representations that a trained model has already learned. One simple approach is to *fine-tune* a classification model to perform the same task for a new class through supervised learning on a small number of labeled examples. More intricate approaches teach new skills through the design of relevant *downstream tasks*–often meta learning tasks—to a model that been pre-trained via self-supervised *pretext tasks*: this is increasingly common in NLP, particularly in the context of foundation models.

More complex transfer learning approaches adapt a trained neural network via changes to the network architecture: for example, replacing or re-training the outer layers of a neural network, where final classification occurs, while maintaining the internal layers where feature extraction occurs. Freezing (or otherwise regularizing changes to) model weights for all but the outermost layers can ensure that subsequent updates don't result in "catastrophic forgetting" of already-learned knowledge. This allows for greatly expedited learning in a few-shot context.

Transfer learning is most successful when the model's initial training is relevant to the new task. For example, a model trained on certain species of birds will generalize well to unseen species of birds after fine-tuning with only a few labeled samples, because the learned weights of the filters the CNN uses for convolutions are already optimized to capture features relevant to bird classification (like plumage, beaks, wing size, etc.)—but using few-shot learning to teach the same model to recognize vehicles will yield less satisfactory performance.

## Data-level approach

An alternative solution to the problem of limited labeled data samples is to generate additional training samples. This is particularly useful when real-world examples of a given class of data are exceedingly scarce, as may be the case when dealing with rare diseases or exotic species.

*Data generation*, via generative models like Generative Adversarial Networks (GANs) or variational autoencoders (VAEs), can potentially yield enough samples resembling the original labeled samples to perform conventional supervised learning, provided the original samples had sufficient diversity to avoid overfitting.

*Data augmentation*, the process of creating new samples by applying different transformations to original samples, can be combined with other methods: for example, it can be used to create matching samples for use in metric meta learning in a process similar to contrastive self-supervised learning.

## Meta learning

Unlike supervised learning or fine-tuning, in which a classifier is trained on the exact tasks it will be used for and the training set contains the same classes the model will be tested on, *meta learning* takes a broader, more indirect approach. Whereas approaches built upon transfer learning adapt pre-trained models, meta learning methods often train systems end-to-end from scratch.

According to Santoro, et al, “meta learning” refers to scenarios in which multiple tasks are used to train a model at both a short-term and long-term level. Within each task, the model learns rapidly to make predictions relevant to the limited domain of that specific task; across tasks, the model gradually accrues knowledge by capturing the way patterns and task structure vary across target domains. This two-tiered process is often described as the model “learning to learn.” [2]

For example, the goal of many prominent meta learning methods is to train a model function, across multiple training episodes, to output a prediction for the degree of similarity between data points from any classes—including classes the model has not yet seen—to then use learnings from that process to solve downstream tasks (like specifically defined classification problems).

Some meta learning approaches work on a more abstract level, by training models to be easy to train. In traditional supervised learning, a model’s parameters (like weights and biases) are what’s “learned,” while the model’s *hyper*parameters—like the learning rate, or how parameters are initialized—are configured prior to training and not part of the learning process. Meta learning can approximate the benefits of transfer learning by learning ideal starting points: parameter initializations or other hyperparameter choices that will generalize well to different datasets in a minimal amount of training steps.

## N-way-K-shot classification

Though a wide variety of machine learning model architectures can be used for few-shot learning, the structure of FSL training and evaluation generally follows an *N-way-K-shot* framework, in which *N* represents the number of classes and *K* represents the number of examples (or “shots”) provided for each class.

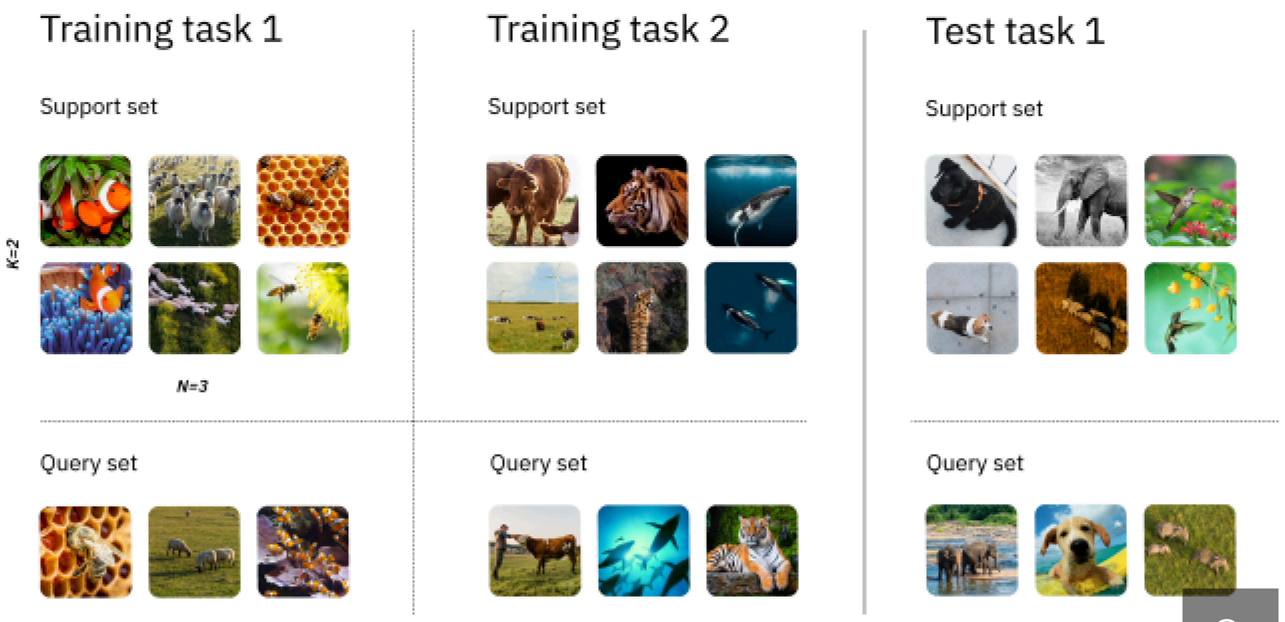
In N-way-K-shot classification, the model undergoes multiple episodes of training. Each training episode consists of one or more *training tasks*. Models are evaluated via *test tasks*, whose structure mirrors that of the training tasks. Each training task (and test task) comprises two datasets:

- The **support set** contains K labeled training samples for each of the N classes. The model uses these support samples to learn generalized representations for each class. For example, the dataset for a *3-way-2-shot* classification task contains 3 classes of images and provides 2 examples of each. When *K*=1, the task is *one-shot learning*. When *K*=0, the problem is *zero-shot learning*—which typically requires unique solutions.

- The **query set** contains one or more new examples for each of the *N* classes. Using representations learned from the support set, the model predicts classification for each example in the query set. A *loss function* measures the divergence ("loss") between the model's predictions and the "correct" predictions; after each training episode, model parameters are adjusted—optimized—to minimize loss.

Because the goal of meta-learning is to train models to generalize well to unseen data, rather than to recognize any specific classes of data, each training task typically includes different data classes than those used in any preceding training tasks.

To test the model's ability to make accurate similarity predictions for heretofore unseen classes, the support set and query set used for testing must contain entirely new classes of data that the model has not yet been exposed to in training tasks.



An example of a 3-way-2-shot classification task for few-shot learning

# Metric-based meta learning

Metric-based meta learning algorithms operate on principle similar to that of *K-nearest neighbors*: rather than predicting classification by directly modeling the decision boundary between classes, metric-based approaches generate a continuous value (like a vector embedding) to represent a given data sample, and make inferences by learning a function that measures some *distance metric* representing the similarity between this value and the value of the different samples or classes it is being compared to.

## Metric-based FSL algorithms

**Siamese networks**
A relatively early development in metric-based algorithms, *Siamese networks* solve binary classification problems by using contrastive learning: shown two samples, Siamese networks predict whether it is positive (matching) or negative (non-matching) pair. The model's loss function is used to minimize the distance between vector embeddings of positive pairs and maximize distance between embeddings of negative pairs. *Triplet loss* models are quite similar: given an "anchor" sample and two additional samples—one matching, one not–the model predicts which is a positive match and which is negative.

In both methods, it is important that training samples be relatively difficult to distinguish from one another—if not, the model will not be forced to learn parameters that yield more effective embeddings. Data augmentation is often used when matching samples are scarce.

Read the paper: "Siamese Neural Networks for One-shot Image Recognition"

**Matching networks**
Whereas Siamese networks can only solve binary classification tasks, *matching networks* can perform multi-way classification. As such, it's considered one of the first dedicated few-shot learning algorithms.

Matching networks output an embedding for each sample in the support and query sets using an appropriate neural network (such as a CNN for image tasks or LLM for natural language tasks) and predict classification by measuring the cosine distance between the embedding of the query sample and that of the available support samples.
Read the paper: "Matching Networks for One Shot Learning"

### Prototypical networks

*Prototypical networks* compute the average features of all samples available for each class in order to calculate a *prototype* for each class. Classification of a given data point is then determined by its relative proximity to the prototypes for each class. Unlike matching networks, Prototypical networks use Euclidian distance rather than cosine distance.

Many refinements to this approach have been proposed: for example, Zhu and Koniusz proposed using label propagation to improve the prototyping process.[3]

Read the paper: "Prototypical Networks for Few-shot Learning"

### Relation networks

A *relation network* (RN) operates on the same general principal as matching and prototypical networks. RNs also utilize an embedding module that learns to compute embeddings for input images and class prototypes—but unlike those two algorithms, which pre-define the distance function used to compare embeddings, RNs add a *relation module* that learns a non-linear distance function that best suits the specific classification problems at hand.

Read the paper: "Learning to Compare: Relation Network for Few-Shot Learning"

## Optimization-based meta learning

Deep learning traditionally requires many iterative updates of model parameters through backpropagation and gradient descent, which in turn depends on a huge quantity of labeled examples to populate training batches. To efficiently train a neural network from scratch for few-shot learning requires a way to optimize model weights in only a few update steps.

## Optimization-based FSL methods

Optimization-based meta learning approaches, also referred to as *gradient-based meta learning* (GMBL), aim to learn initial model parameters or hyperparameters for a neural network that can be efficiently fine-tuned for relevant tasks. They achieve by optimizing the process of gradient descent—that is, by *meta-optimizing* the process of optimization itself.

**Model agnostic meta-learning (MAML)**
MAML is among the most prominent optimization-based approaches, and has served as the foundation for a number of approaches derived from its core methodology. As its name suggests, model agnostic meta-learning doesn't focus on a specific task or AI model architecture: it can be used on any model that learns via gradient descent.

MAML entails two different levels of parameters updates across a set of varied FSL training tasks, $p(T)$. In each training episode, a new task $T_i$ is randomly sampled from $p(T)$; gradient descent, performed in $K$ steps of size $\alpha$, is used to optimize a vector of *task-specific* model parameters ($\vartheta'_i$) after each training task. Across multiple training episodes, a set of *meta-parameters* ($\boldsymbol{\vartheta}$) is optimized by applying gradient descent, in meta-steps of size $\beta$, to those task-specific parameters $\vartheta'_i$. In other words, whereas ordinary gradient descent calculates derivatives in order to optimize a model's parameters for a given task, MAML calculates the *derivatives of the derivatives* (or "second order derivatives") to optimize a model's initial parameters for subsequent task-specific optimization.

Per the original paper, the goal therein is to "find model parameters that are sensitive to changes in the task, such that small changes in the parameters will produce large improvements on the loss function of any task drawn from $p(T)$." This yields benefits similar to those of transfer learning while circumventing the need for large amounts of labeled data for pre-training.

Proposed adaptations to MAML include:

- *First Order MAML (FOMAML)*: MAML's reliance on second-order derivatives is computationally expensive and requires a great deal of memory. FOMAML simplifies the process via a series of assumptions that allow for meta-optimization using only first order derivatives.
- *Reptile*: Reptile presents a midway point between the sophistication of MAML and the simplicity of FOMAML: it uses first order derivatives, but implements unique rules for how parameters are updated.[4]
- *Optimizing step size*: Variants like Meta-SGD[5] and Alpha MAML[6] add the ability to optimize step size and direction for $\alpha$ and $\beta$. Similarly, MAML++[7] introduces a number of modifications to increase stability and computational efficiency.

Read the paper: "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks"

**LTSM meta-learner**

Meta-learning approaches can make use of RNN-based long-short term memory (LSTM) networks to train a meta-learner model to capture both short-term knowledge from each training task and long-term knowledge common to each task. This meta-learner is then used to train a neural network classifier.

Read the paper: "Optimization as a Model for Few-Shot Learning"

**Latent embedding optimization (LEO)**

Rather than explicitly instantiating and updating a unique set of model meta-parameters $\vartheta$, *latent embedding optimization* learns a generative distribution of task-specific model parameters in a manner similar to variational autoencoders (VAEs), which serves the same purpose. Gradient optimization can then be performed within that learned, low-dimensional embedding space.

Read the paper: "Meta-Learning with Latent Embedding Optimization"

## Few-shot learning use cases

Few-shot learning techniques have a wide variety of applications, as many industries and research fields stand to benefit from the ability to learn quickly and effectively from relatively few examples.

**Computer vision**
While many prominent FSL algorithms were originally developed for (or proven on) image classification tasks, FSL can also be used for more complex computer vision problems.

While object detection is a significantly more complex problem than image classification, as objects must be not only classified but also accurate localized, it generally takes image classification as a prerequisite. As such, many ideas used for classification can be adopted for few-shot object detection.[8]

Likewise, a number of model architectures have been proposed for few-shot semantic segmentation.[9]

**Robotics**
FSL can enable robots to quickly adapt to new environments and new tasks through both few-shot classification tasks[10] and reinforcement learning.[11]

**Natural language processing**
FSL has shown promising results for natural language processing (NLP), particularly through transfer learning: it's an intuitive way to adapt large language models (LLMs), pre-trained on a massive corpus of unlabeled data, to specific tasks like text classification and sentiment analysis that may require specific contextual understanding.

**Healthcare**
FSL's potential to quickly acclimate a model to rare and unseen data classes is particularly promising for medical fields, in which the rarity of certain conditions or the expertise required to accurately annotate medical data (like MRIs or

echocardiography) can make the acquisition of a large number of labeled samples prohibitively difficult.

## Related solutions

### AI consulting services

Reimagine how you work with AI: our diverse, global team of more than 20,000 AI experts can help you quickly and confidently design and scale AI and automation across your business, working across our own IBM watsonx technology and an open ecosystem of partners to deliver any AI model, on any cloud, guided by ethics and trust.

Explore IBM AI consulting services

### IBM watsonx Assistant

Deliver consistent and intelligent customer care across all channels and touchpoints with conversational AI.

Explore IBM watsonx Assistant

### IBM watsonx.ai

Train, validate, tune and deploy generative AI, foundation models and machine learning capabilities with ease and build AI applications in a fraction of the time with a fraction of the data.

Explore watsonx.ai

# Few-shot learning resources

Blog

### Few-shot learning and generative AI

Learn more about the rise and future of generative AI—and how few-shot learning fits into the bigger picture.

Research

### Few-shot learning with multiple semantics

Building upon recent advances in few-shot learning with additional semantic information, we demonstrate that further improvements are possible by combining multiple and richer semantics (category labels, attributes, and natural language descriptions).

Tutorial

### Tips for writing foundation model prompts

A quick overview of prompt engineering tips for few-shot and one-shot prompting for foundation models.

# Take the next step

Train, validate, tune and deploy generative AI, foundation models and machine learning capabilities with IBM watsonx.ai, a next-generation enterprise studio for AI builders. Build AI applications in a fraction of the time with a fraction of the data.

**Explore watsonx.ai** →

**Book a live demo** →

**Footnotes**

*All links reside outside ibm.com*

[1] "An Embarrassingly Simple Approach to Semi-Supervised Few-Shot Learning," arXiv, 28 September 2022
[2] "Meta-Learning with Memory-Augmented Neural Networks," Proceedings of the 33rd International Conference on Machine Learning, 19 June 2016
[3] "Transductive Few-shot Learning with Prototype-based Label Propagation by Iterative Graph Refinement," Computer Vision Foundation, 23 April 2023
[4] "Reptile: A scalable meta-learning algorithm," OpenAI, 7 March 2018
[5] "Meta-SGD: Learning to Learn Quickly for Few-Shot Learning," arXiv, 28 September 2017
[6] "Alpha MAML: Adaptive Model-Agnostic Meta-Learning," arXiv, 17 May 2019
[7] "How to train your MAML," arXiv, 5 March 2019
[8] "Few-Shot Object Detection: A Comprehensive Survey," arXiv, 15 September 2022
[9] "Leaderboard: Few-Shot Semantic Segmentation on PASCAL-5i (1-Shot)," PapersWithCode, accessed 3 January 2024
[10] "Few-Shot Continual Active Learning by a Robot," arXiv, 12 October 2022
[11] "Few-Shot Preference Learning for Human-in-the-Loop RL," Proceedings of the 6th Conference on Robot Learning, 2023