Before we introduce the algorithm, I first introduce a concept called task. In meta-learning algorithm, we don't use datapoints as samples, we use tasks to train the meta model. A task contains n way k shots. N way represents N labels in the data, K shot represents that in each label, it has K datapoints. Meanwhile, K shot also has 2 categories, we divide it to support set and query set, we use support set to train the model and use query set to test the model accuracy.

Then, I'd like to formally introduce the concept of meta learning to you guys. For the classical machine learning, our goal is to train a model on a dataset generated by only one distribution, and test on different dataset generated from the same distribution. However, for meta learning, we actually train a model that can output the specific classification or regression model to classify the task generated from different distributions. For example, a typical task can be datasets of dogs generated by a distribution of dogs, another task can be datasets of cats, we can only use these two tasks to train a meta model to let it know the essential characteristics behind different tasks. Then, when we have datasets of rabbits, we can do fine-tuning on 3-4 datapoints of rabbits and get a good accuracy, this is what meta learning actually do.

Then, I'll introduce the concept of MAML. MAML is a typical type of algorithm in the meta learning field. Its goal is to train a best set of parameters to obtain a fast adaption in different test scenarios, as showing in the ppt slides. Our goal is to find a set of parameters that can rapidly converge to global minimum of a new test task. In this figure, we can find a set of parameters φ, for different new test tasks 1 and 2, which can easily converge on different global minimum in different tasks.

Then, we move to illustrate the concrete algorithm of MAML. It consists of two processes, meta-training and meta-testing stages. For meta-training stage, here is an intuitive figure which illustrates how the model updates its own parameters. Here θ corresponds to our initialization. Here we have M tasks. For each task, MAML first conducts inner gradients for each task to find each task's optimal direction, which corresponds to theta_1' up to theta_M'. And then MAML updates the final model parameter theta* by integrating each task's optimal direction and finds the most potential direction for all the tasks. In the meta-testing stage, we can just use a few samples in the specific task to do fine-tuning, in order to get a considerable accuracy. We use MAML in Widar since it has multiple domains, also regarded as multiple tasks. Finally, we can achieve a relatively better result.