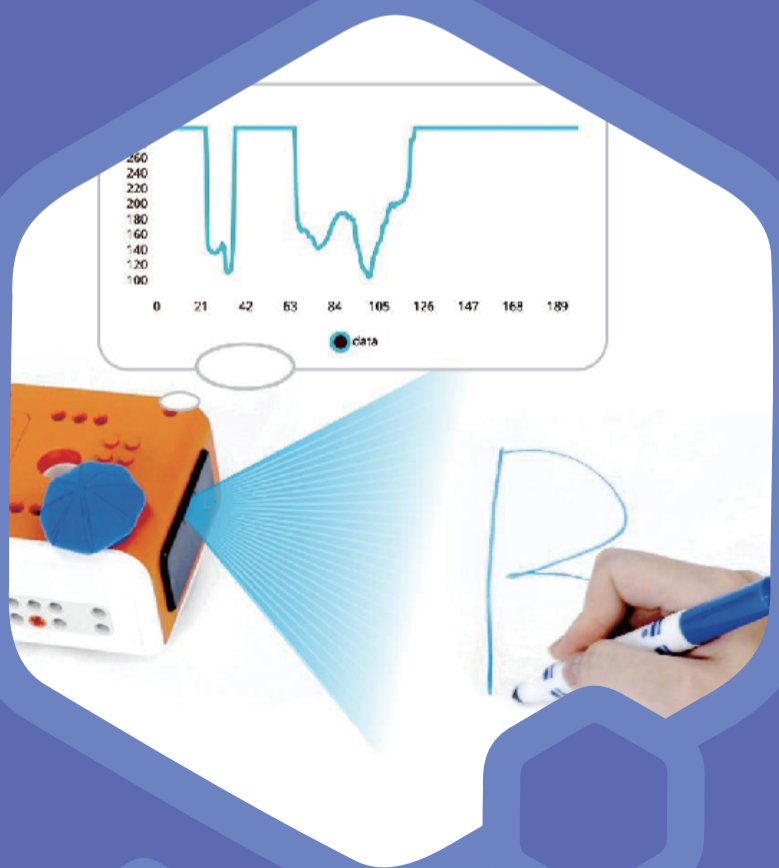
matata studio

Teacher Manual

AI, Coding and Robotics

for VinciBot



Contents

Teacher Guide

Teacher Guide

AI Features of VinciBot: Embedded Machine Learning (Tiny ML) and IoT

What is AI?

AI, or Artificial Intelligence, is like giving brains to machines. Just like how we use our brains to think, learn, and make decisions, AI teaches computers to do the same thing. Imagine having a super-smart friend who can help you with all sorts of tasks: solving puzzles, playing games, even chatting with you! AI is like that friend, but for computers. It helps them understand things, make predictions, and even come up with new ideas. For example, AI can help your phone recognize your voice commands, suggest movies you might like, or beat you at a game of chess. It's really cool because it makes machines smarter and more helpful in our everyday lives. So, learning about AI is like unlocking the secrets of how to make computers act just like humans!

The main aspects of artificial intelligence (AI) encompass various facets of simulating human intelligence in machines. Here are the key aspects: machine learning, natural language processing (NLP), computer vision, expert systems, etc.

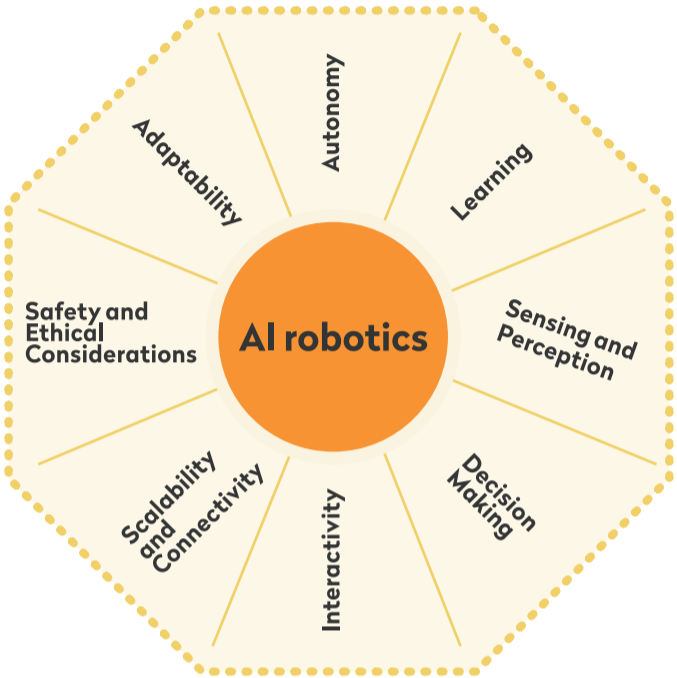
What is AI Robot?

An AI robotic, often referred to simply as a "robot", is a machine equipped with artificial intelligence (AI) capabilities that enable it to perform tasks autonomously or with minimal human intervention.

These robots are equipped with sensors, actuators, control systems, and AI algorithms that enable them to perceive their environment, process information, make decisions, and take actions based on what they sense.

AI robotics is a rapidly evolving field with applications ranging from industrial automation and manufacturing to healthcare, transportation, and even household chores.

Key Characteristics of AI robotics include:



Three Key Components of an AI Robotics

AI robotics typically consists of several key components, including sensors, actuators, and control systems.

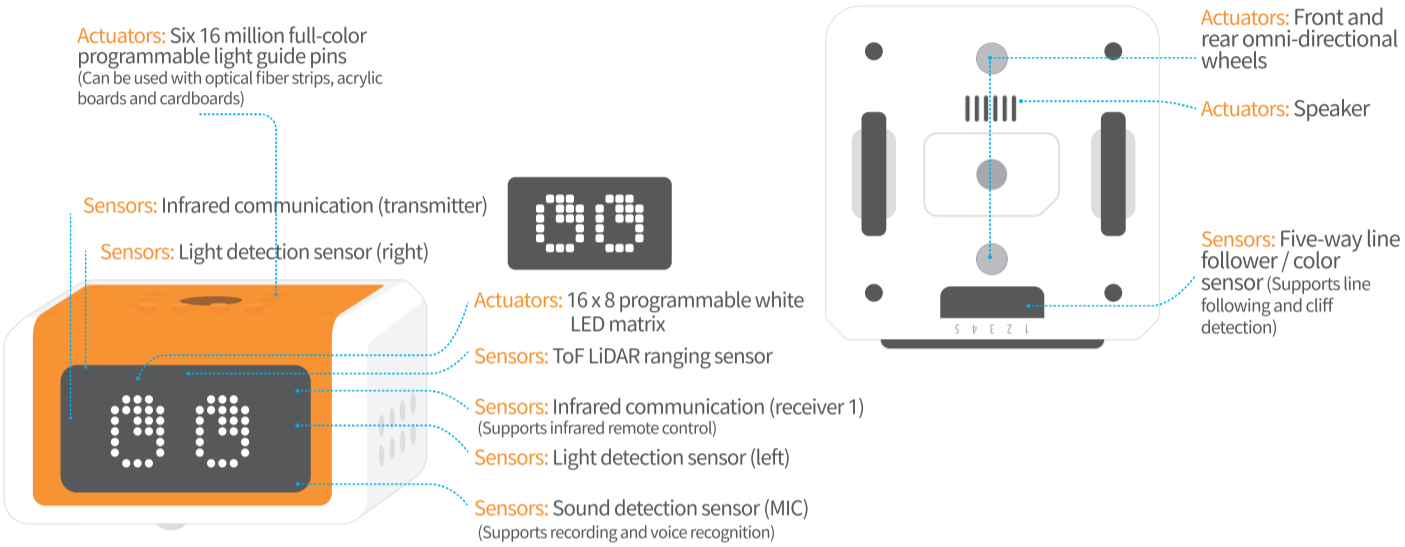
Sensors: These are devices that detect and measure physical properties or environmental conditions. They provide data to the robot about its surroundings, enabling it to perceive and interact with the environment. Common types of sensors used in AI robots include: cameras, ToF ranging sensor, LiDAR (Light Detection and Ranging), infrared sensors, sound sensors, etc.

Actuators: These are mechanisms responsible for physically manipulating the robot's environment based on the instructions received from the control system. Actuators convert electrical or pneumatic signals into mechanical motion. Common types of actuators includes motors, servos, pneumatic actuators, and hydraulic actuators.

Control System: This is the brain of the robot, responsible for processing sensor data, making decisions, and sending commands to actuators to achieve desired tasks or behaviors. The control system may include: microcontrollers or microprocessors, HUB, etc.

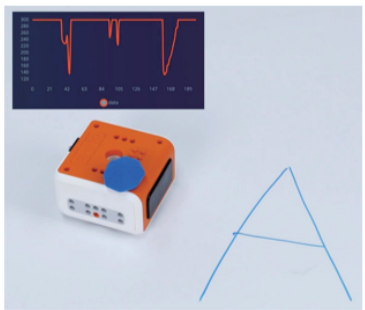
VinciBot: an AI robotics solution

As an AI robotics solution, VinciBot possesses characteristics such as autonomy, learning, sensing and perception, decision making, interactivity, and adaptability. Looking at the product itself, it incorporates a variety of sensors, multiple actuators, and it is equipped with an embedded microcontroller system.



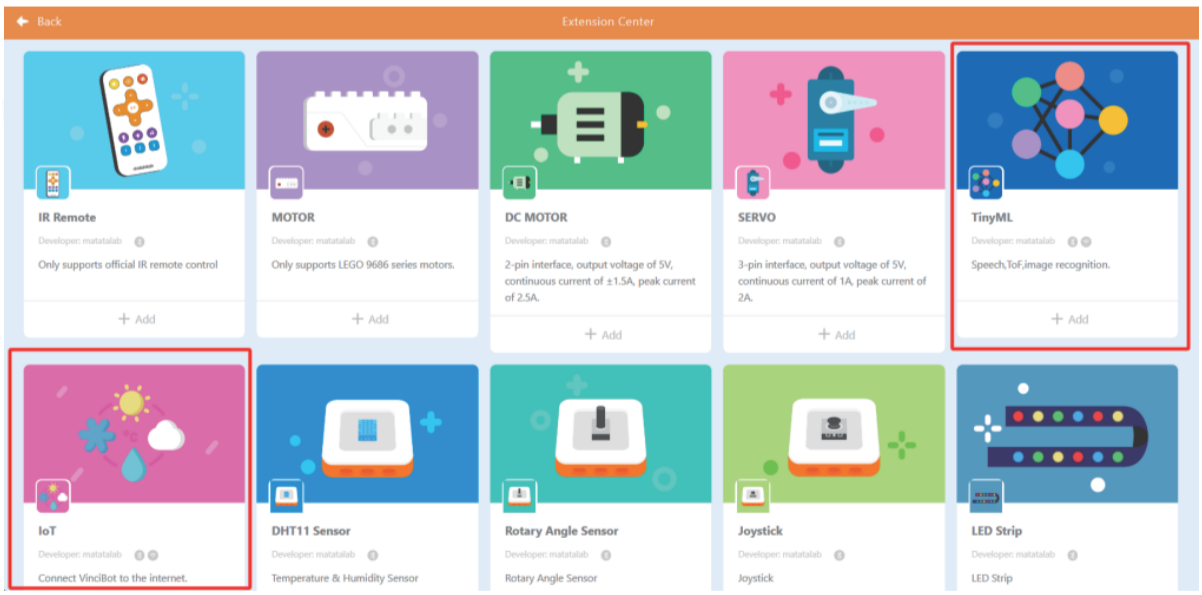
Typical AI Features of VinciBot: Tiny ML and IoT

Tiny ML, or embedded machine learning, is one type of Machine Learning. Through model creation, data acquisition, training and development, and programming, VinciBot can recognize speech commands, hand gestures, objects, etc, with no Internet required!



The Internet of Things (IoT) describes the network of physical objects — “things”— that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Smart Home is a great case for using IoT.

To use the Tiny ML and IoT of VinciBot, please check in the extension center.



60min

Lesson 1

Listen,VinciBot!

Overview

In this lesson, students will learn the four steps of Tiny ML by creating and using a speech model. There are four steps of Tiny ML: model creation, data acquisition, training and development, and programming.

Data Model:

Speech Model

Learning Objectives

- Know Tiny ML, and be familiar with the 4 steps for Tiny ML: model creation, data acquisition, model training and deployment, and programming.
- Create one speech model and know the tips for acquiring speech data.
- Use the created model to program and turn the VinciBot into an obedient robot pet.

Standards

ISTE: 1a, 1d, 2d, 4a, 5a, 5b, 6c

CSTA (Grade 3-5) : 1B-CS-01, 1B-CS-03, 1B-DA-06, 1B-DA-07, 1B-AP-10, 1B-AP-11, 1B-AP-12, 1B-AP-15

Materials

VinciBot, PC/Pad

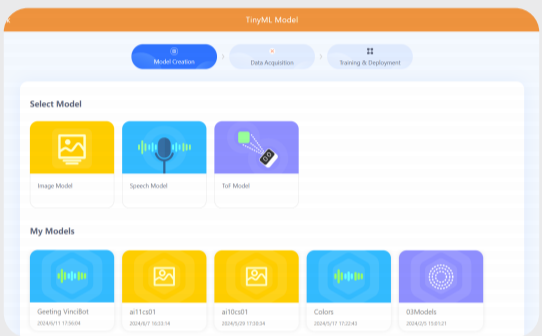
Instructions for Students

AI and Machine Learning for AI

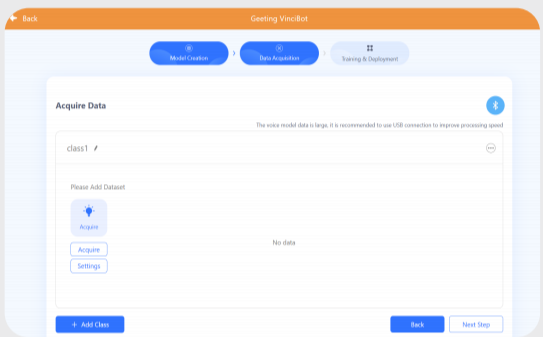
VinciBot as an Artificial Intelligence (AI) robotic solution, in addition to basic programming, it also supports AI functions such as embedded machine learning (Tiny ML), Internet of Things (IoT), etc. In this lesson, we will first become familiar with the the Tiny ML.

Embedded machine learning (Tiny ML) is a typical type of Machine Learning. It is also a typical function for VinciBot. There is one microphone in VinciBot, which helps the VinciBot can "hear" according to its AI function. In this lesson, we are going to create a speech model, and use the created model to program.

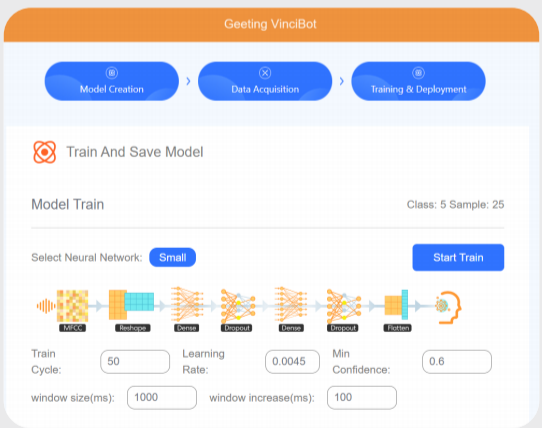
The process of Tiny ML



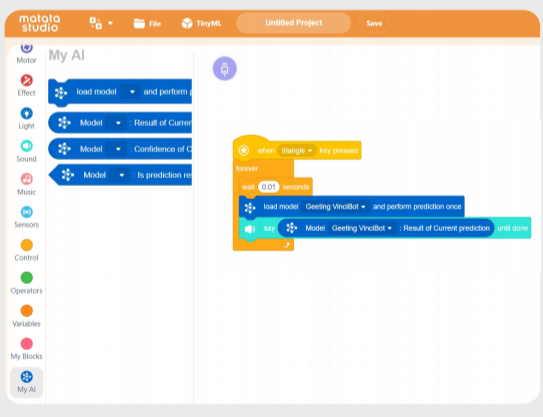
Step 1: Choose and create an ML model



Step 2: Acquire relevant data

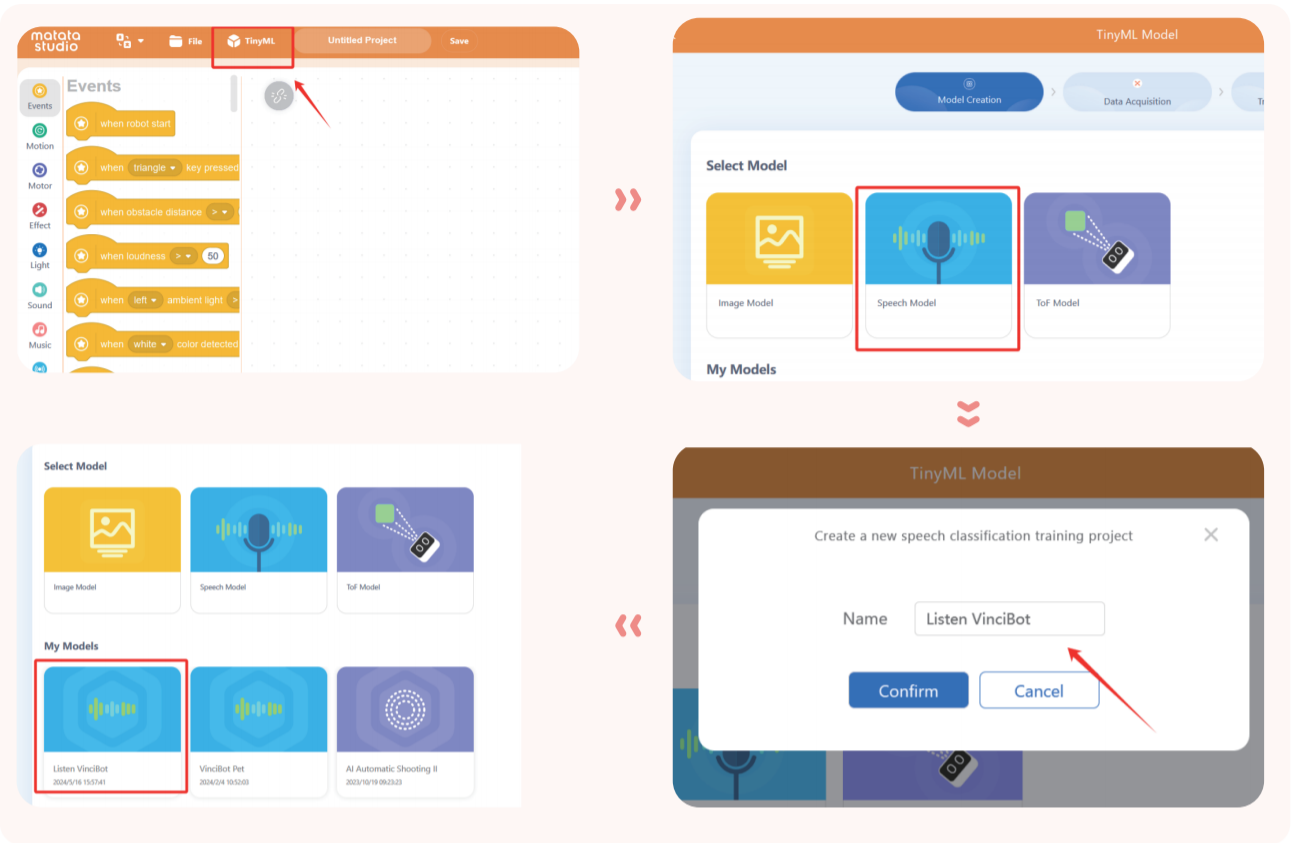


Step 3: Train and deploy the model



Step 4: Use the model to program

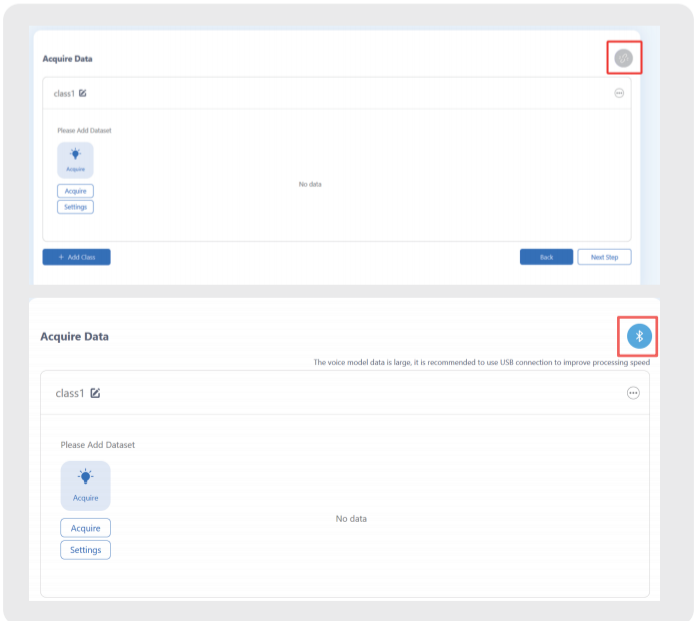
How can you choose and create a speech model?



How can you acquire relevant data?

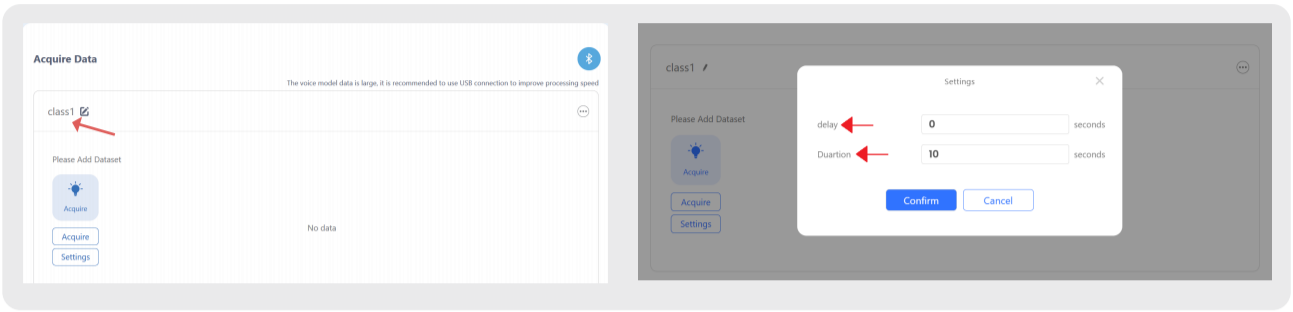
First, design the speech commands/wake-up words. For example, “Hello” “Dance” , and “Sing a song” .

Then, Before acquiring the data, you need to make sure the VinciBot is connected to the coding platform.



Next, change the class name, and check the “Setting” .

In “Settings” , "delay" means the delay of several seconds in starting to collect data after the the "Acquire" button is pressed. “Duration” shows how long it takes to acquire one sample. Usually, we need to repeat the commands/wake-up words several times in this duration. In most cases, we can use the default settings, which include no delay and a duration of 10 seconds.



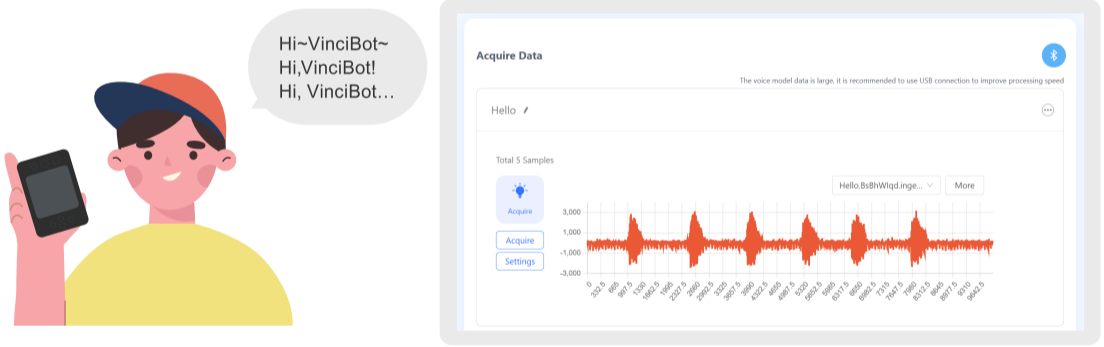
Finally, press the "Acquire" button to acquire data samples. There are several tips:

1. In the “duration” of acquiring data samples (which is 10 seconds in this case), you need to keep repeating the command/wake-up word.

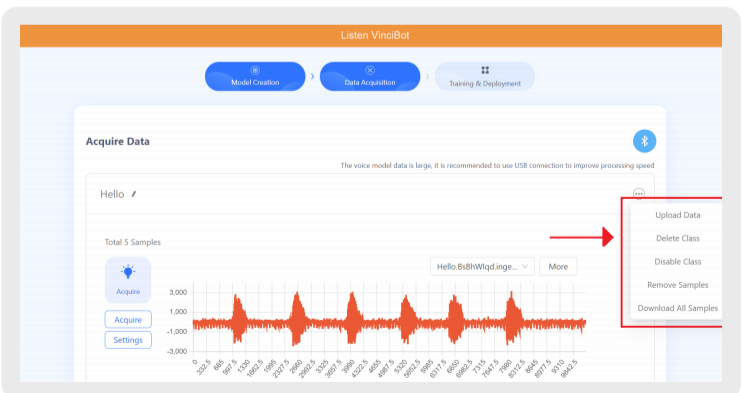
2. Data diversity: The more diverse the data, the better the model. So, we can acquire speech data with different speech speeds, timbres, and accents.

3. Please collect at least four audio data samples for each class.

4. Each model should have at least two classes of data, with one class being "Nothing." (In a speech model, "Nothing" refers to background noise when no commands/wake-up words is detected; in an image model, "Nothing" refers to the background scene when no image is present.)

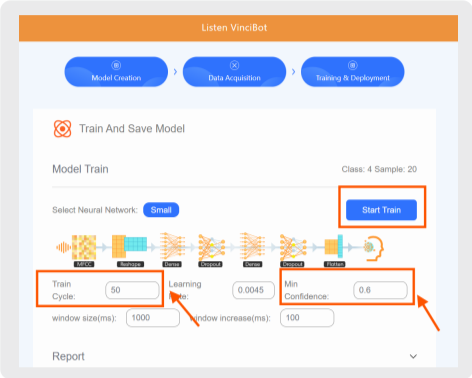


One important tip is we can download data of one class by pressing the “download all samples” button and uploading them to another model.

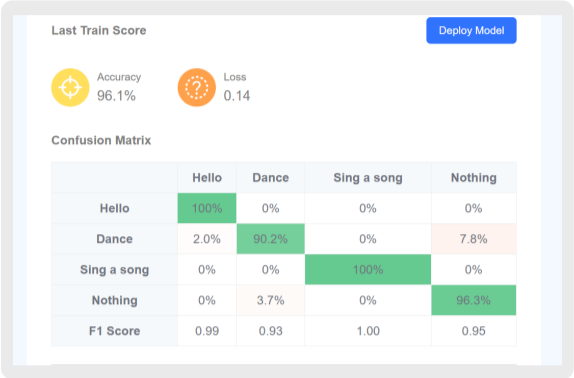


How can you train and deploy the model?

- Press " Start Train" to train the model. Pay attention to the "Train Cycle", the default Train Cycle is 50.

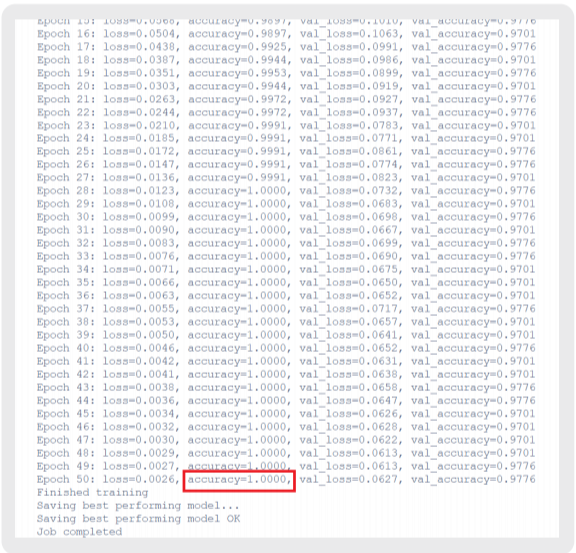


- After finishing training the model, please read the report. In this case, the report shows that the trained model is pretty good.



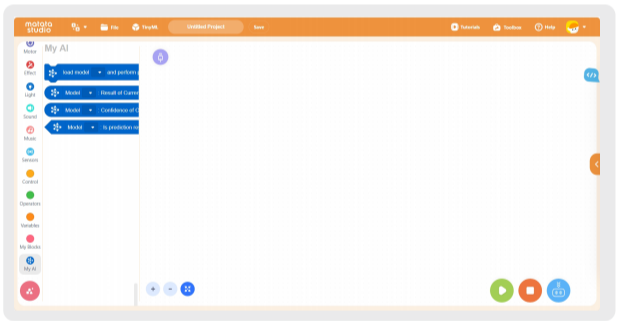
	Hello	Dance	Sing a song	Nothing
Hello	100%	0%	0%	0%
Dance	2.0%	90.2%	0%	7.8%
Sing a song	0%	0%	100%	0%
Nothing	0%	3.7%	0%	96.3%
F1 Score	0.99	0.93	1.00	0.95

- We should also read the Logout. Usually, the accuracy from Epoch 1 to Epoch 50 should be getting closer to 1.

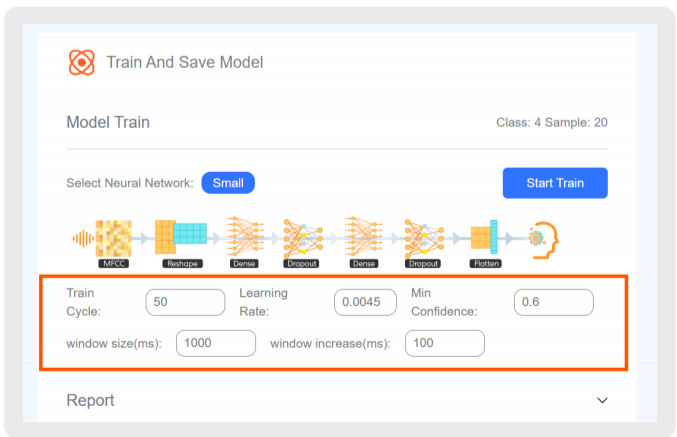


```
Epoch 16: loss=0.0504, accuracy=0.9897, val_loss=0.1063, val_accuracy=0.9701
Epoch 17: loss=0.0438, accuracy=0.9925, val_loss=0.0991, val_accuracy=0.9776
Epoch 18: loss=0.0387, accuracy=0.9944, val_loss=0.0986, val_accuracy=0.9701
Epoch 19: loss=0.0351, accuracy=0.9953, val_loss=0.0899, val_accuracy=0.9776
Epoch 20: loss=0.0303, accuracy=0.9944, val_loss=0.0919, val_accuracy=0.9701
Epoch 21: loss=0.0263, accuracy=0.9972, val_loss=0.0927, val_accuracy=0.9776
Epoch 22: loss=0.0244, accuracy=0.9972, val_loss=0.0937, val_accuracy=0.9776
Epoch 23: loss=0.0210, accuracy=0.9991, val_loss=0.0783, val_accuracy=0.9701
Epoch 24: loss=0.0185, accuracy=0.9991, val_loss=0.0771, val_accuracy=0.9701
Epoch 25: loss=0.0172, accuracy=0.9991, val_loss=0.0861, val_accuracy=0.9776
Epoch 26: loss=0.0147, accuracy=0.9991, val_loss=0.0774, val_accuracy=0.9776
Epoch 27: loss=0.0136, accuracy=0.9991, val_loss=0.0823, val_accuracy=0.9701
Epoch 28: loss=0.0123, accuracy=1.0000, val_loss=0.0732, val_accuracy=0.9776
Epoch 29: loss=0.0108, accuracy=1.0000, val_loss=0.0683, val_accuracy=0.9701
Epoch 30: loss=0.0099, accuracy=1.0000, val_loss=0.0698, val_accuracy=0.9776
Epoch 31: loss=0.0090, accuracy=1.0000, val_loss=0.0667, val_accuracy=0.9701
Epoch 32: loss=0.0083, accuracy=1.0000, val_loss=0.0699, val_accuracy=0.9776
Epoch 33: loss=0.0076, accuracy=1.0000, val_loss=0.0690, val_accuracy=0.9776
Epoch 34: loss=0.0071, accuracy=1.0000, val_loss=0.0675, val_accuracy=0.9701
Epoch 35: loss=0.0066, accuracy=1.0000, val_loss=0.0650, val_accuracy=0.9701
Epoch 36: loss=0.0063, accuracy=1.0000, val_loss=0.0652, val_accuracy=0.9701
Epoch 37: loss=0.0055, accuracy=1.0000, val_loss=0.0717, val_accuracy=0.9776
Epoch 38: loss=0.0053, accuracy=1.0000, val_loss=0.0657, val_accuracy=0.9701
Epoch 39: loss=0.0050, accuracy=1.0000, val_loss=0.0641, val_accuracy=0.9701
Epoch 40: loss=0.0046, accuracy=1.0000, val_loss=0.0652, val_accuracy=0.9776
Epoch 41: loss=0.0042, accuracy=1.0000, val_loss=0.0631, val_accuracy=0.9701
Epoch 42: loss=0.0041, accuracy=1.0000, val_loss=0.0638, val_accuracy=0.9701
Epoch 43: loss=0.0038, accuracy=1.0000, val_loss=0.0658, val_accuracy=0.9776
Epoch 44: loss=0.0036, accuracy=1.0000, val_loss=0.0647, val_accuracy=0.9776
Epoch 45: loss=0.0034, accuracy=1.0000, val_loss=0.0626, val_accuracy=0.9701
Epoch 46: loss=0.0032, accuracy=1.0000, val_loss=0.0628, val_accuracy=0.9701
Epoch 47: loss=0.0030, accuracy=1.0000, val_loss=0.0622, val_accuracy=0.9701
Epoch 48: loss=0.0029, accuracy=1.0000, val_loss=0.0613, val_accuracy=0.9701
Epoch 49: loss=0.0027, accuracy=1.0000, val_loss=0.0613, val_accuracy=0.9776
Epoch 50: loss=0.0026, accuracy=1.0000, val_loss=0.0627, val_accuracy=0.9776
Finished training
Saving best performing model...
Saving best performing model OK
Job completed
```

- If the "Accuracy" in the report and the logout are not very good, you can either acquire more data samples, simply retrain the model, or change the "Train Cycle" to retrain the model.
- Press the "Deploy Model" button, and go to the programming page.



Here are some parameters in “Train” , including “Train Cycle” , “Learning Rate” , "Min Confidence” , “window size(ms)” and “window increase(ms)” . Usually, such terms have their default settings. So, we rarely need to change the settings.



Train Cycle describes the iterative process of training a model using a dataset, evaluating its performance, and adjusting the parameters, all until satisfactory results are achieved.

Learning Rate is a hyperparameter that controls the size of the steps taken during the optimization process, particularly in iterative optimization algorithms, which include gradient descent. It determines the model's parameter adjustment range in each update during training. Finding an appropriate learning rate often involves experimentation and tuning.

Window Size refers to the number of data points or observations considered in a single iteration or processing step.

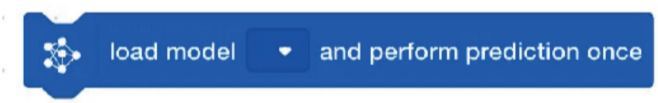
Sometimes, if your training results are not particularly satisfactory, you can achieve better outcomes by adjusting “Train Cycle” or "Learning Rate” and training again. However, how to change the parameter is quite complex. At this stage, we just need to try different parameters.

The parameter in window size is relatively easy to understand. For example, in a speech ML model, the preset parameter for window size is 1000ms, which is 1 second. When our command is "Hello" or similarly short, and it can be repeated more than once within 1 second, this setting requires no change. However, when our command is longer, such as "VinciBot, how are you?” and "VinciBot, how do you do?", increasing the window size appropriately becomes necessary.

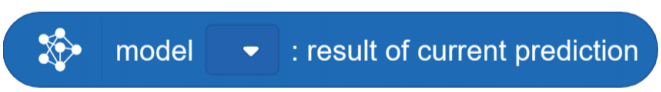
How can you program?

1. Get familiar with My AI coding blocks

- Load the specified embedded machine learning model and make a prediction.

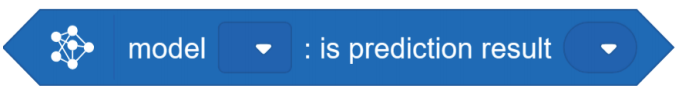


- Obtain a specified model's prediction. The name of the category to which the forecast result belongs.



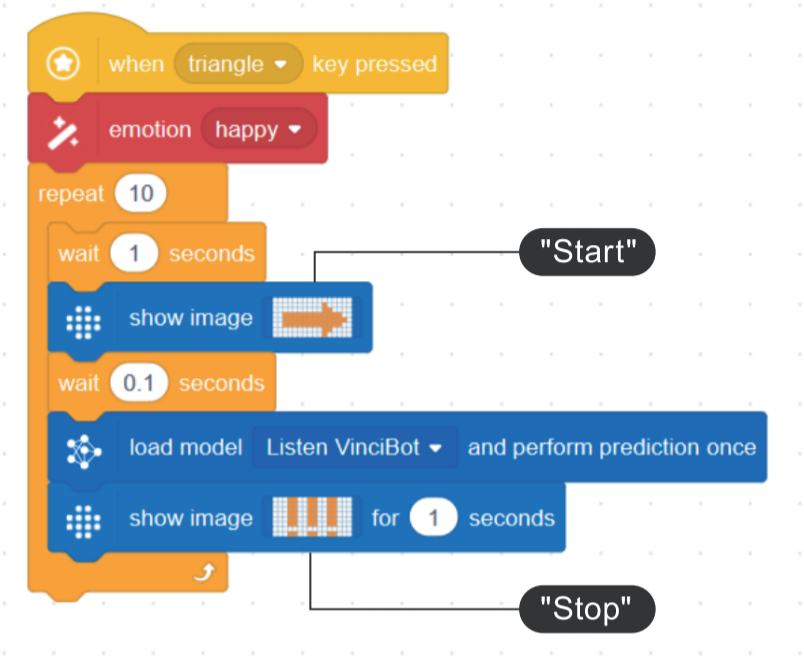
- Obtain the confidence level in a specified category from a specified model's predictions. Return a value in the range of 0-100.

- Determine whether a specified model's predictions belong to the specified category. Return a boolean value. True: The forecast matches the specified category. False: the forecast mismatches it.



2. Increase the accuracy of predictions by indicating the time of each prediction.

To enhance the speech recognition accuracy, we can add prompts for the "Start" and "Stop" of each prediction through programming. One demo sub-program to prompt each prediction time is shown below.



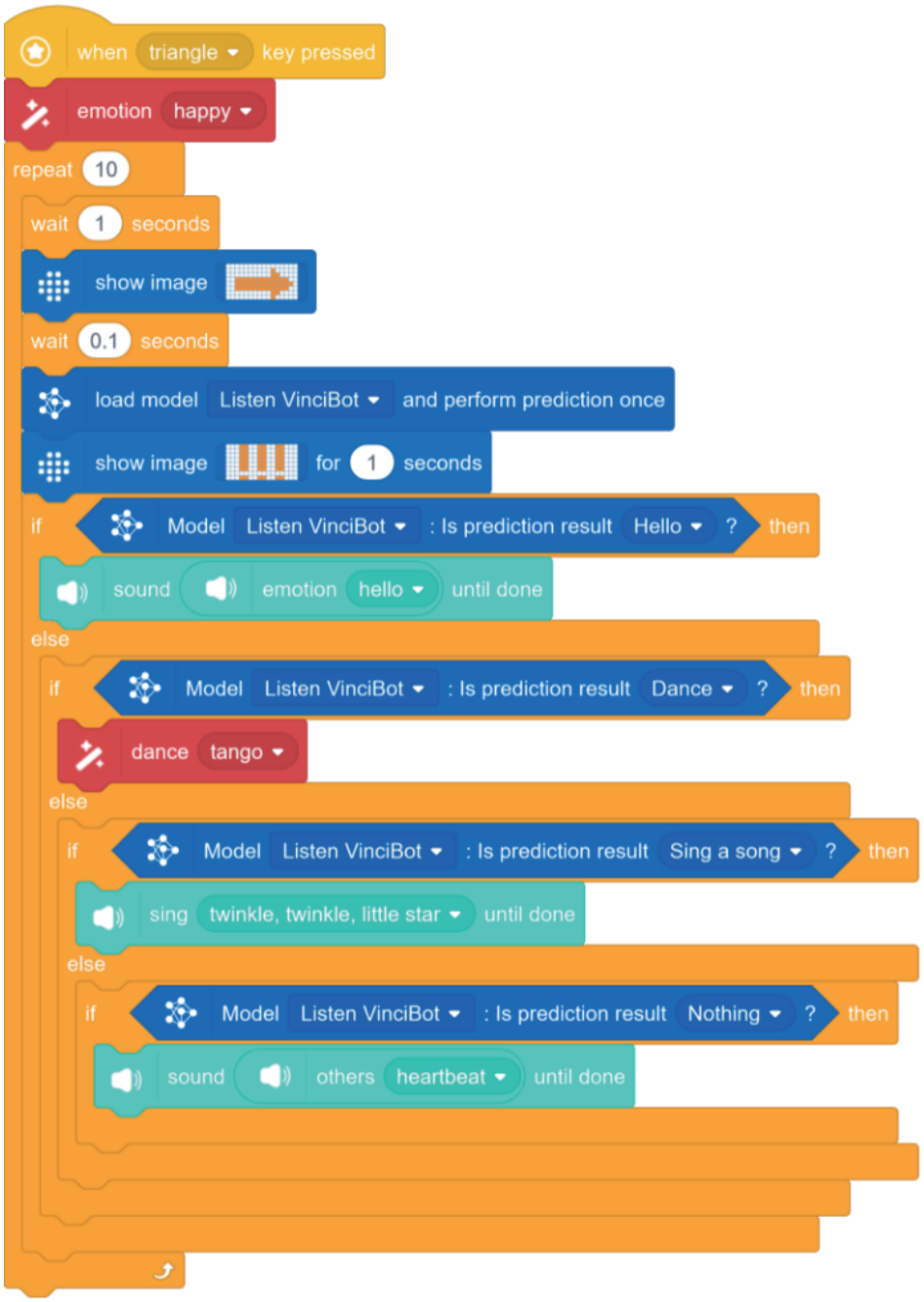
When using the speech commands, we need to start speaking a command continuously as soon as the "Start" signal lights up. We should keep saying the command

Activity

Activity 1： Listen, VinciBot!

The Speech commands	VinciBot will
Hello	Say “Hello”
Dance	Dance
Sing a song	Sing a Song
Nothing	Make a “heartbeat” sound

Demo Program:



Students discuss, create, and share

1. What is the process of Tiny ML?
2. How many Tiny ML model types does VinciBot has? What are they?
3. Talk about the tips of acquiring speech data.

Extension Activity

Vinci Pet

45min

Task

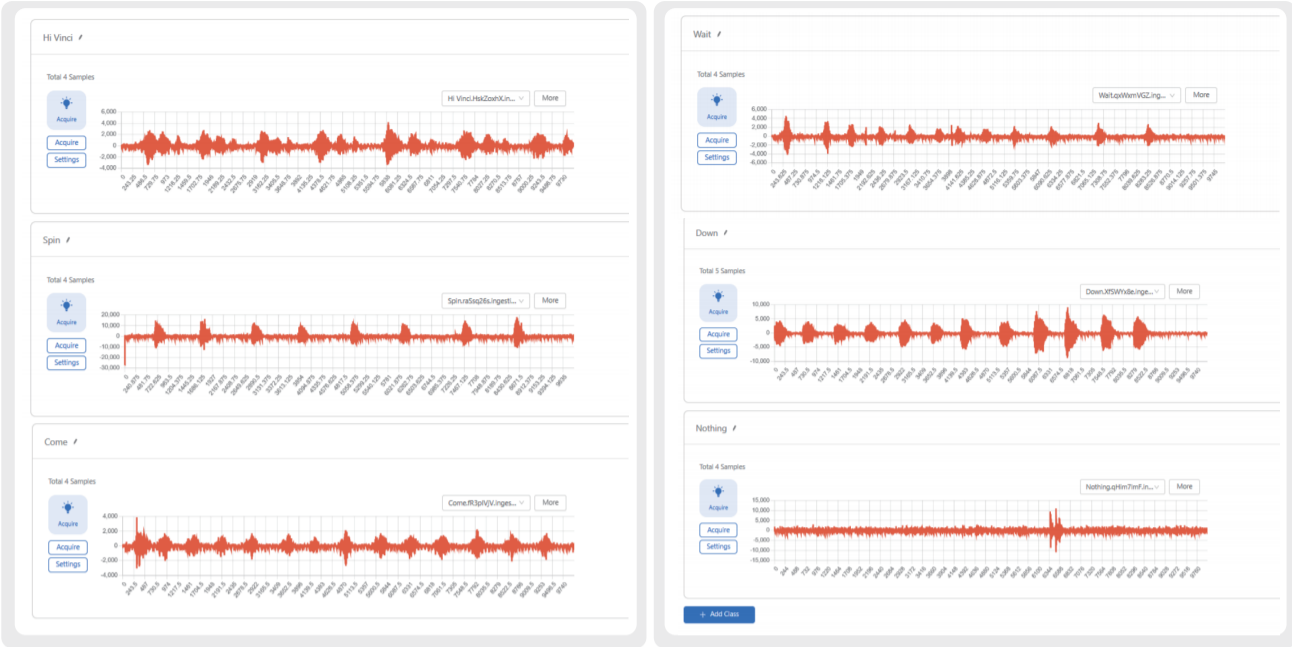
Students are going to create, train and deploy a speech model which includes all the commands/wake up words for your Vinci pet. Then, they will use the created model to program, and turn the VinciBot into an obedient robot pet.

Step 1: Students design the command words in the speech model "Vinci Pet" and their corresponding reactions from the Vinci Pet. For example:

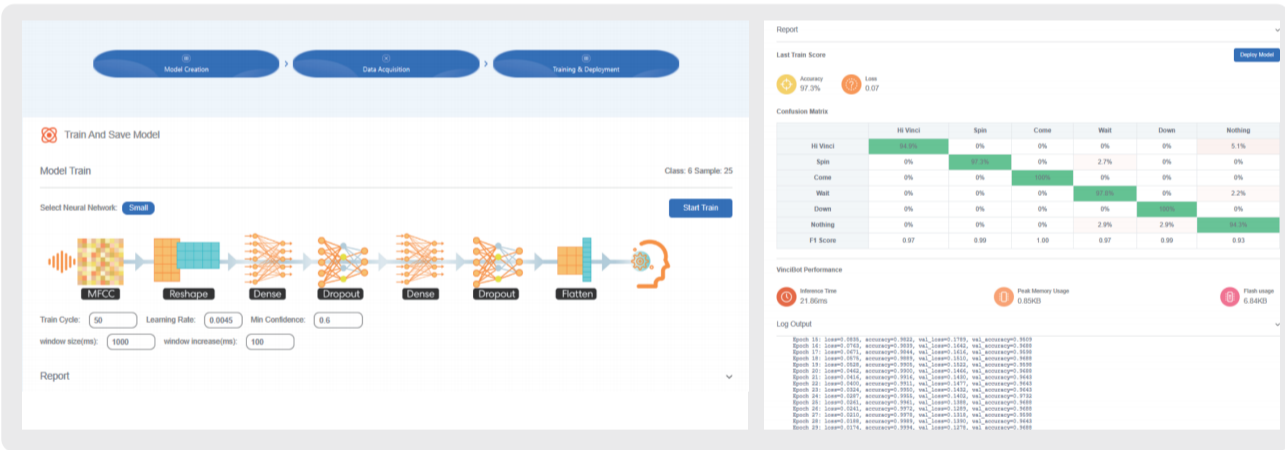
The Speech Commands	VinciBot will
Hi, Vinci	Make a baking sound
Spin	Spin around in place
Come	Do the action “sprint”
Wait	Do the action “shake”
Down	Make an emotion “sleepy”
Nothing	Make a “heartbeat” sound

Step 2: Students create, train, and deploy a speech model named Vinci Pet.

1. Students select and create the voice model "Vinci Pet."
2. Students follow the steps to systematically collect all voice data.



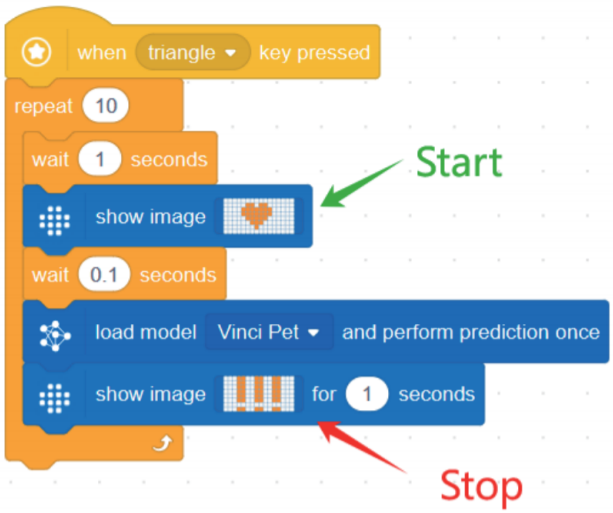
3. Students train the model and decide whether to retrain it based on the Accuracy Report. If the results are still unsatisfactory, students need to adjust some parameters and train the model again.



4. Click the "Deploy Model" button to deploy the model into My Models.

Step 3: Students use the “Vinci Pet” speech model to program.

1. Students should design "Start" and "Stop" signals for each prediction. Then write a sub-program to prompt each prediction time to enhance the speech recognition accuracy.



2. Add the “nested if else” statement to finish the program.
Demo program: