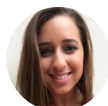




ML.NET September Updates



Bri

September 25th, 2020

[ML.NET](#) is an open-source, cross-platform machine learning framework for .NET developers. It enables integrating machine learning into your .NET apps without requiring you to leave the .NET ecosystem or even have a background in ML or data science. ML.NET provides tooling (Model Builder UI in Visual Studio and the cross platform ML.NET CLI) that automatically trains custom machine learning models for you based on your scenario and data.

This release of ML.NET (1.5.2) brings numerous bug fixes and enhancements, while tooling updates include the ability to train object detection models using Azure ML via Model Builder. You can now also locally train image classification models with the ML.NET CLI.

In this post, we'll cover the following items:

1. [Object detection in Model Builder](#)
2. [Image classification in ML.NET CLI](#)
3. [ML.NET 1.5.2](#)
4. [Feedback](#)
5. [Get started and resources](#)

Object detection in Model Builder

Object detection is a computer vision problem. While closely related to image classification, object

detection performs image classification at a more granular scale. Object detection both locates and

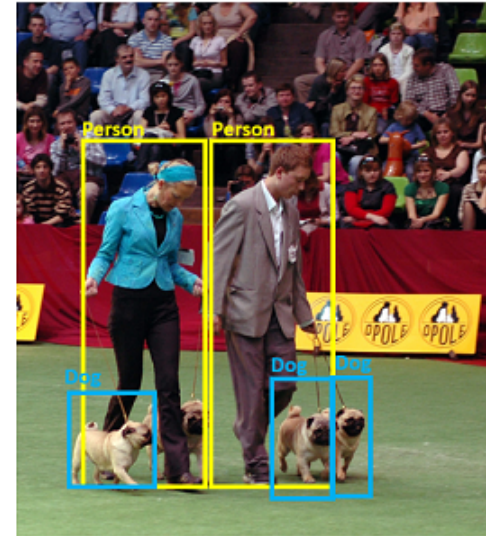
categorizes entities within images. Use object detection when images contain multiple objects of different types.

Image Classification



{Dog}

Object Detection

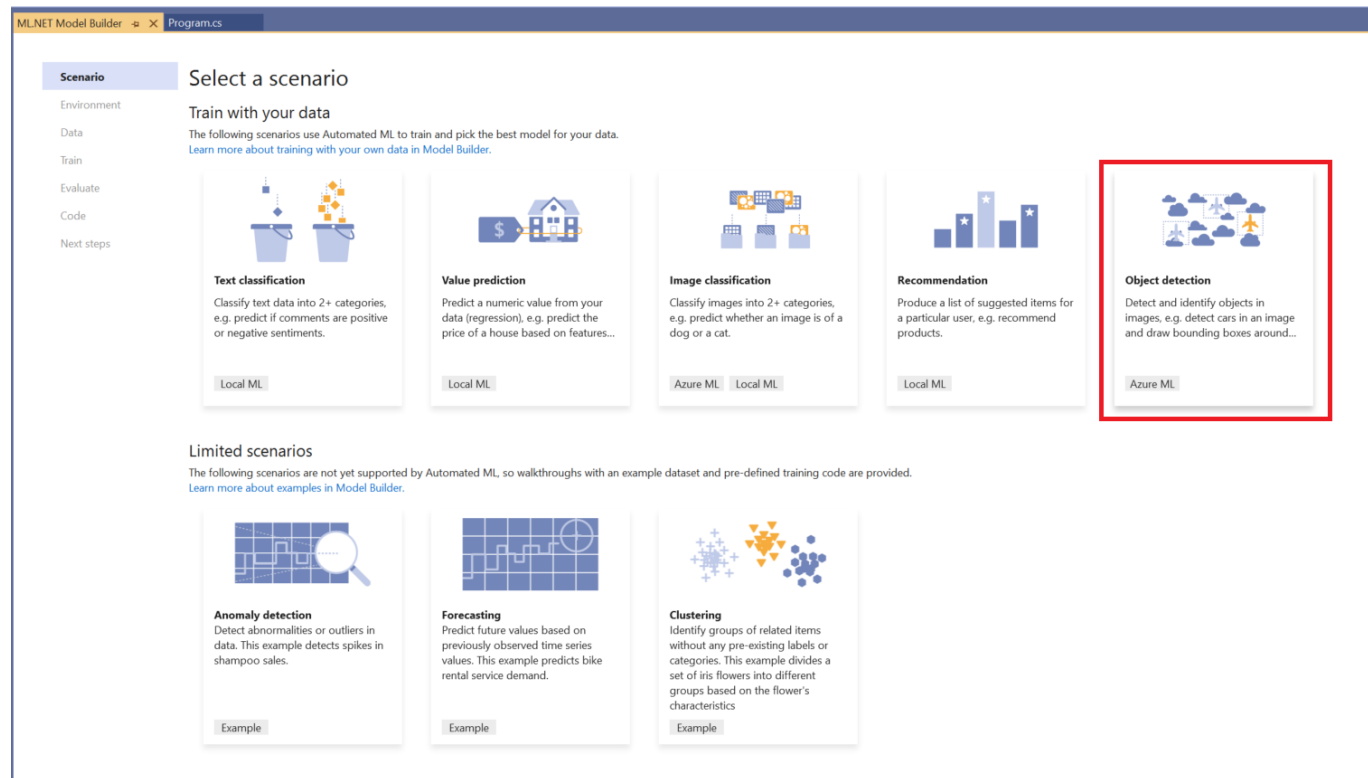


{Dog, Dog, Dog, Person, Person}

Some use cases for object detection include:

- Self-Driving Cars
- Robotics
- Face Detection
- Workplace Safety
- Object Counting
- Activity Recognition

While previously ML.NET offered the ability to consume pre-trained TensorFlow and ONNX models for object detection via the ML.NET API, you can now use Model Builder in Visual Studio to train custom object detection models with the power of Azure and AutoML.



After selecting the object detection scenario and setting up your Azure ML workspace in Model Builder, you must input your data. Currently, Model Builder does not offer a way to annotate images, so you must use an external tool to draw bounding boxes around objects in your training images.

If you need to label your images, we recommend trying out [VoTT](#), an open source annotation and labeling tool for image and video assets.

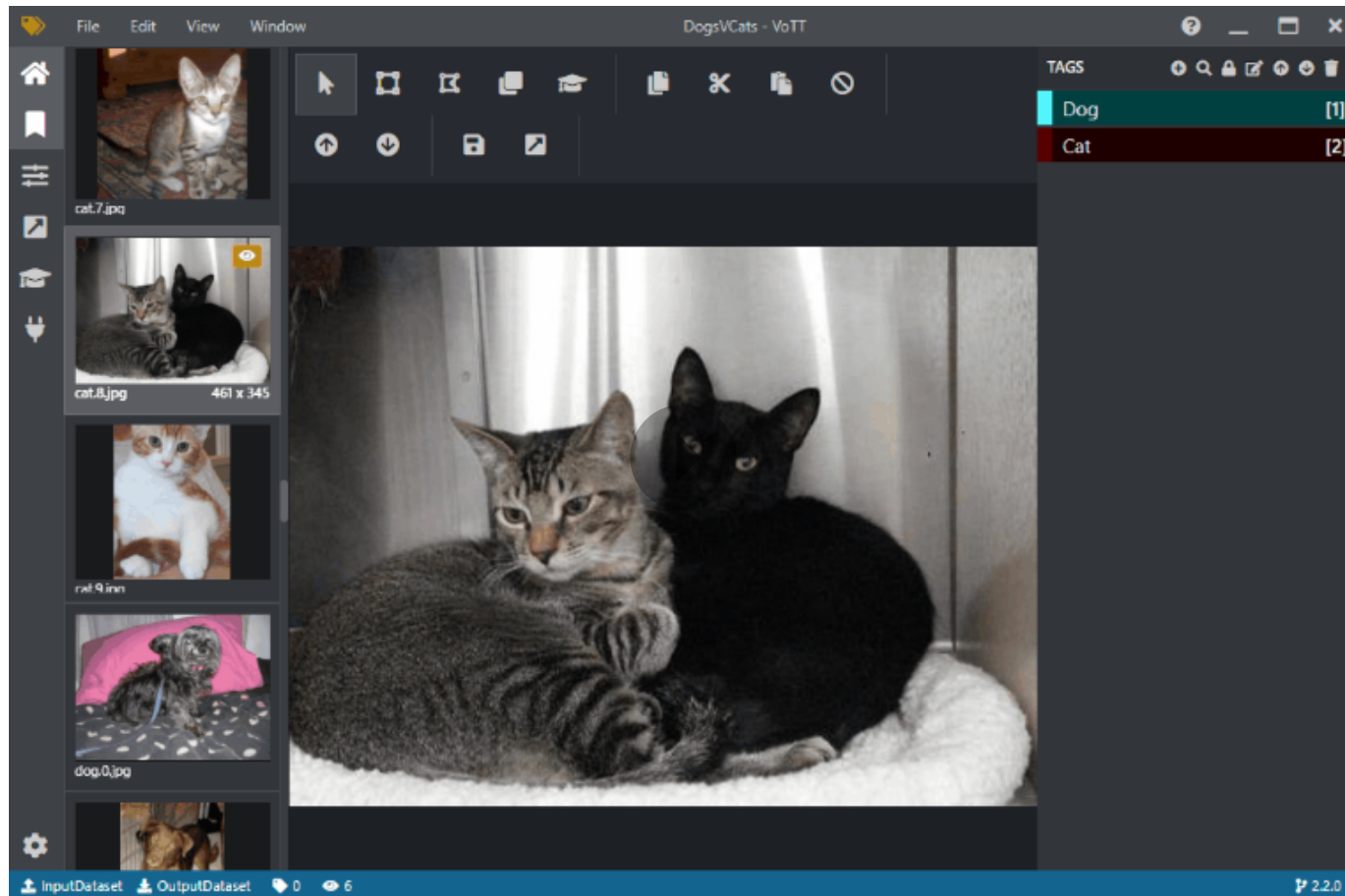
After downloading VoTT, you can create a new VoTT project and choose a dataset from your Local File System (Azure Blob Storage and Bing Image search are also options, but Model Builder currently only supports training from a local dataset). This is simply a folder that contains all the images that you'd like to annotate.

After selecting the folder path to your dataset of images for your Source Connection, you can then

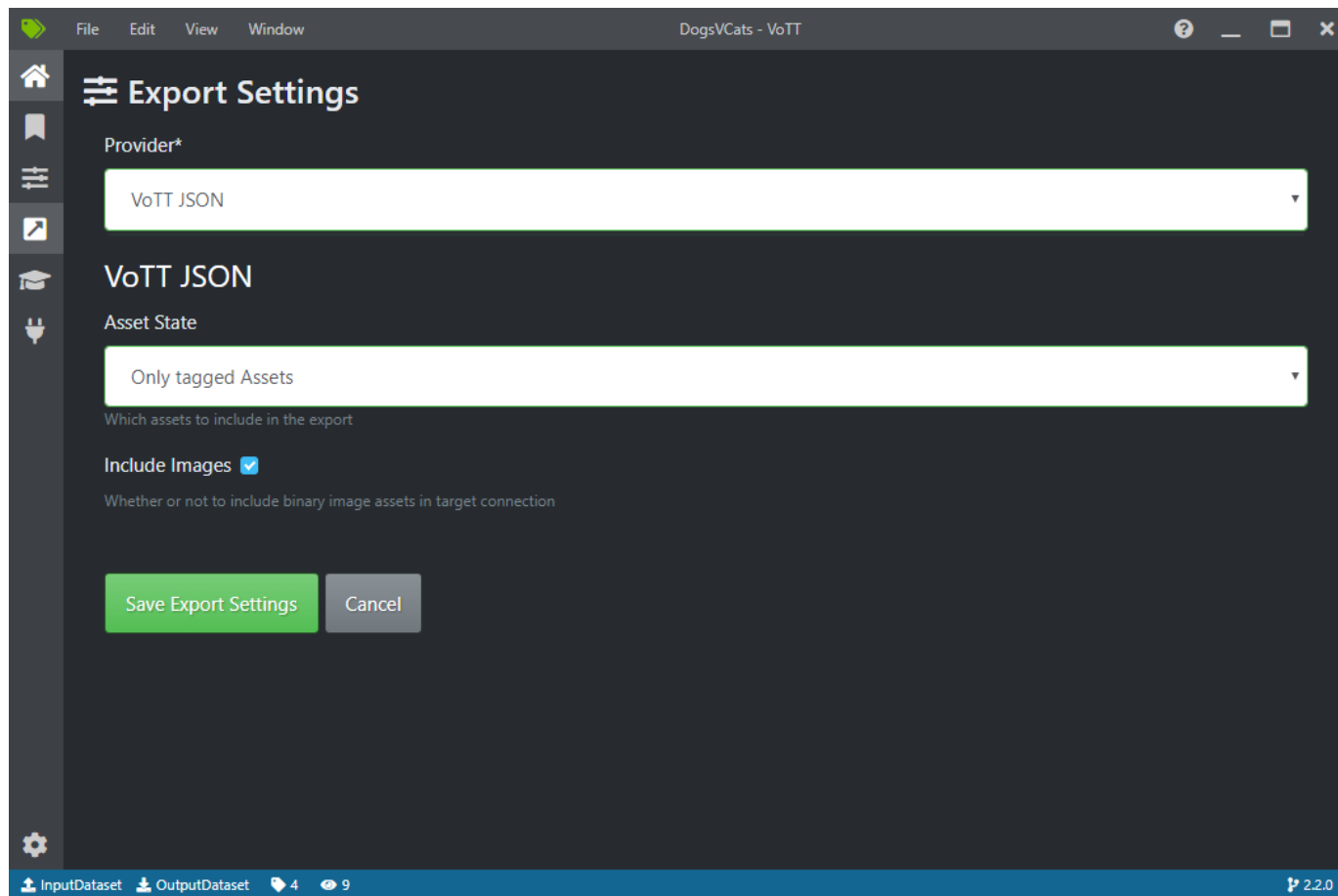
choose an output folder for the target Connection.

You can skip the Video Settings, and then add in your Tags (any objects that you want to be detected in your images):

After saving your project, all images found in the Source Connection folder will appear, and you can start labeling your images like this:



When you're done labeling, you can go to the Export section (4th icon down on the left toolbar), select "VoTT JSON" as the Provider and "Only tagged Assets" as the Asset State (Include images is optional). When you hit Save Export Settings, the VoTT JSON will be generated in the Target Connection folder that you specified during VoTT project creation.



The VoTT JSON looks like this:

```
{
  "name": "CatDog",
  "securityToken": "CatDog Token",
  "videoSettings": {
    "frameExtractionRate": 15
  },
  "tags": [
    {
      "name": "cat",
      "color": "#5db300"
    },
    {
      "name": "dog",
      "color": "#e81123"
    }
  ],
  "id": "QhzB19NrT",
  "actualSpanningSettings": f
```

```

    "autoDetect": false,
    "predictTag": true,
    "modelPathType": "coco"
  },
  "version": "2.2.0",
  "lastVisitedAssetId": "676f5652fe5e62120cb0c4a3554023",
  "assets": {
    "676f5652fe5e62120cb0c4a3554023": {
      "asset": {
        "format": "jpg",
        "id": "676f5652fe5e62120cb0c4a3554023",
        "name": "dog.9.jpg",
        "path": "file:C:/Users/brachtma/Desktop/Datasets/Object Detection/training/vott-json-export/dog.9.jpg",
        "size": {
          "width": 368,
          "height": 500
        },
        "state": 2,
        "type": 1
      },
      "regions": [
        {
          "id": "a6NNYwxN6",
          "type": "RECTANGLE",
          "tags": [
            "dog"
          ],
          "boundingBox": {
            "height": 468.6147186147186,
            "width": 350.16231617647054,
            "left": 0,
            "top": 29.22077922077922
          },
          "points": [
            {
              "x": 0,
              "y": 29.22077922077922
            },
            {
              "x": 350.16231617647054,
              "y": 29.22077922077922
            },
            {
              "x": 350.16231617647054,
              "y": 497.83549783549785
            },
            {
              "x": 0,
              "y": 497.83549783549785
            }
          ]
        }
      ]
    }
  },
  "version": "2.2.0",
  "986d0a21bcec0dc97e4da359c3efcd21": {
    "asset": {

```

You can then use this VoTT JSON as the dataset input for the Data step in Model Builder.

Model Builder currently only accepts the format of JSON generated by VoTT, but we plan to add support for more formats in the future. If there is a dataset format for object detection that you'd like to see supported in Model Builder, leave your feedback on [GitHub](#).

ML.NET Model Builder Program.cs

Scenario
Environment
Data
Train
Evaluate
Code
Next steps

Add data

In order to build a model, you must add image data.
[How do I get sample datasets and learn more?](#)


Input

Select a JSON file with image labeling. [How do I format my JSON?](#)

ction\training\vott-json-export\CatDog-export.json ...





Supported file formats: .jpg, .jpeg, .png

Data Preview (Showing 4 of 110 images)



dog

List of objects
■ dog
Show undetected objects ♥



Next step

After inputting your data and moving to the Train step in Model Builder, you can hit Start training, which uploads your data to Azure and begins the training with Azure ML. When training is finished, your trained ML.NET model is downloaded so that you can test it out locally.

In the Evaluate step, you can see your model's accuracy as well as make predictions on test images:

ML.NET Model Builder Program.cs

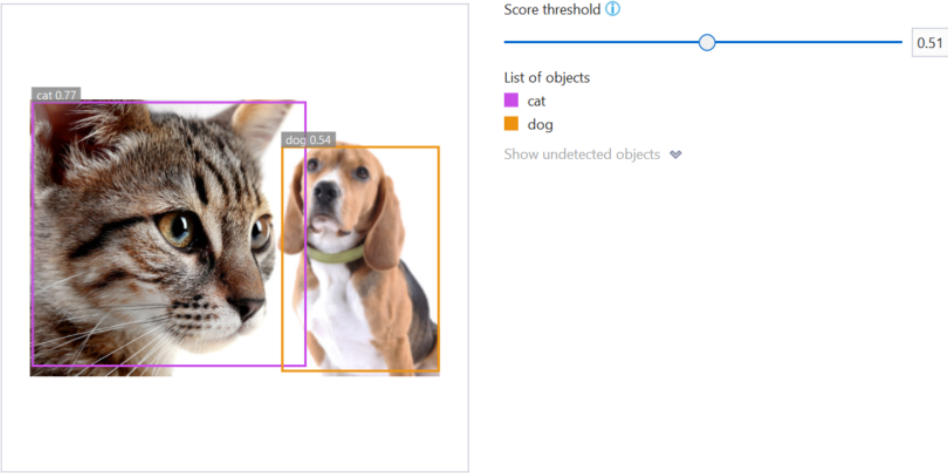
Scenario
Environment
Data
Train
Evaluate
Code
Next steps

Evaluate

Results of training for your model can be found below.
[How do I understand my model performance?](#)

Best model:
Accuracy: 81%
Model: Azure AutoML Algorithm

Try your model



Score threshold ⓘ

0.51

List of objects

- cat
- dog

Show undetected objects ♥

[Browse a different image \(.png, .jpg, .jpeg\)](#)

Next step

The score shown on each detected bounding box indicates the confidence of the detected object. For instance, in the screenshot above, the score on the bounding box around the cat indicates that the model is 77% sure that the detected object is a cat.

The score threshold, which can be increased or decreased with the threshold slider, will add and remove detected objects based on their scores. For instance, if the threshold is .51, then the model will only show objects that have a score / confidence of .51 or above. As you increase the threshold, you will see less detected objects, and as you decrease the threshold, you will see more detected objects.

Once you are satisfied with your model's performance, you can generate the model and consumption code in the Code step in Model Builder and integrate your model into your end-user application.

Image classification in ML.NET CLI

In addition to classification, regression, and recommendation, you can use the cross-platform [ML.NET CLI](#) to locally train custom image classification models.

All you need for this scenario is dataset of images that you'd like to use for training. For instance, let's look at the weather example, where you want to classify an image as rainy, cloudy, or sunny.

First you need to make sure you have your images in the correct format, which is a folder that organizes the photos into separated labeled sub-folders like this:

↓ ↑ 📁 > Datasets > Image Classification > Weather-Small		
<input type="checkbox"/> Name	Date modified	Type
📁 Cloudy	8/18/2020 1:08 PM	File folder
📁 Rainy	8/18/2020 1:08 PM	File folder
📁 Sunny	8/18/2020 1:08 PM	File folder

In this case, each folder contains 30 images of the corresponding weather.

Once you have your dataset, you can use the following command in the ML.NET CLI to start training:

```
mlnet image-classification --dataset "Weather-Small"
```

When training is done, the CLI will output the accuracy of your model and will generate the necessary projects for model consumption and re-training:

```
Command Prompt

=====Experiment Results=====
|
|-----Summary-----|
|ML Task: image-classification|
|Dataset: C:\Users\brachtma\Desktop\Datasets\Image Classification\Weather-Small\ImageLabel.tsv|
|Label : Label|
|Total experiment time : 867.929111 Secs|
|Total number of models explored: 1|
|-----|
|
|-----Top 1 models explored-----|
|
|Trainer          MicroAccuracy  MacroAccuracy  Duration #Iteration|
|1  ImageClassification  0.9181        0.9378        867.9        1|
|-----|
|
Code Generated
Generated C# code for model consumption: C:\Users\brachtma\Desktop\Datasets\Image Classification\SampleImageClassification\SampleImageClassification.ConsoleApp
Check out log file for more information: C:\Users\brachtma\AppData\Local\Temp\mlnet\log.txt
Exiting ...
```

ML.NET 1.5.2

Last month we announced ML.NET 1.5.1, which had a regression that is fixed with ML.NET 1.5.2. Thus, we recommend that you skip 1.5.1 and update to 1.5.2.

This release also closed over 30 reported bugs and added ONNX enhancements to support more types for ONNX export.

You can see more in the [1.5.2 release notes](#).

Feedback

We would love to hear your feedback!

If you run into any issues, please let us know by creating an issue in our GitHub repos (or use the new Feedback button in Model Builder!):

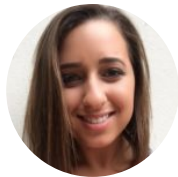
- [ML.NET API](#)
- [ML.NET Tooling \(Model Builder & ML.NET CLI\)](#)

Get started and resources

Get started with ML.NET in this [tutorial](#).

Learn more about ML.NET and Model Builder in [Microsoft Docs](#).

Tune in to the [Machine Learning .NET Community Standup](#) every other Wednesday at 10am Pacific Time.



[Bri Achtman](#)

Program Manager, .NET

Follow   

Posted in [.NET](#) [Machine Learning](#) [ML.NET](#)

Read next

Repo experience survey results

A couple of months ago we ran a survey of our github community to understand our users satisfaction and experience with the mechanics of how the projects are open-sourced with the community. This post shares the results.



[Sam Spencer](#) September 29, 2020

 11 comments

.NET Interactive Preview 3: VS Code Insiders and .NET Polyglot Notebooks

With .NET Interactive Preview 3, we've added a VS Code Insiders experience and a number of new features to our existing .NET notebooks support for Jupyter.



[Maria Naggaga](#) September 30, 2020

 3 comments

2 comments

[Log in](#)

to join the discussion.



Deepak September 28, 2020 8:26 pm



Very nice information. It's very helpful to me Thanks A lot.

Best Regards

[Sachin Patil](#)

↩ [Log in to Reply.](#)



sanme 98 September 29, 2020 7:49 am



Object recognition is a helpful feature, but next in the future is face recognition :)?

↩ [Log in to Reply.](#)

Relevant Links

- [.NET Download](#)
- [.NET Hello World](#)
- [.NET Meetup Events](#)
- [.NET Documentation](#)

Archive

- [October 2020](#)
- [September 2020](#)
- [August 2020](#)
- [July 2020](#)

Topics

- [Dot.Net](#)
- [.NET](#)
- [.NET Core](#)
- [.NET Framework](#)

.NET API Browser

.NET SDKs

.NET Application Architecture Guides

Web apps with ASP.NET Core

Mobile apps with Xamarin.Forms

Microservices with Docker Containers

Modernizing existing .NET apps to the

June 2020

May 2020

April 2020

March 2020

February 2020

January 2020

December 2019

Entity Framework

C#

ML.NET

Visual Studio

F#

WPF

Performance

Stay informed



What's new

Surface Duo

Surface Laptop Go

Surface Pro X

Surface Go 2

Surface Book 3

Microsoft 365

Windows 10 apps

Microsoft Store

Account profile

Download Center

Microsoft Store support

Returns

Order tracking

Virtual workshops and training

Microsoft Store Promise

Education

Microsoft in education

Office for students

Office 365 for schools

Deals for students & parents

Microsoft Azure in education

Enterprise

Azure

AppSource

Automotive

Government

Healthcare

Manufacturing

Financial services

Retail

Developer

Microsoft Visual Studio

Windows Dev Center

Developer Center

Microsoft developer program

Channel 9

Office Dev Center

Microsoft Garage

Company

Careers

About Microsoft

Company news

Privacy at Microsoft

Investors

Diversity and inclusion

Accessibility

Security



English (United States)

[Sitemap](#)

[Contact Microsoft](#)

[Privacy](#)

[Terms of use](#)

[Trademarks](#)

[Safety & eco](#)

[About our ads](#)

© Microsoft 2020