Making an Art Critic using Machine Learning

Rob Miles
Author and Raconteur
www.robmiles.com
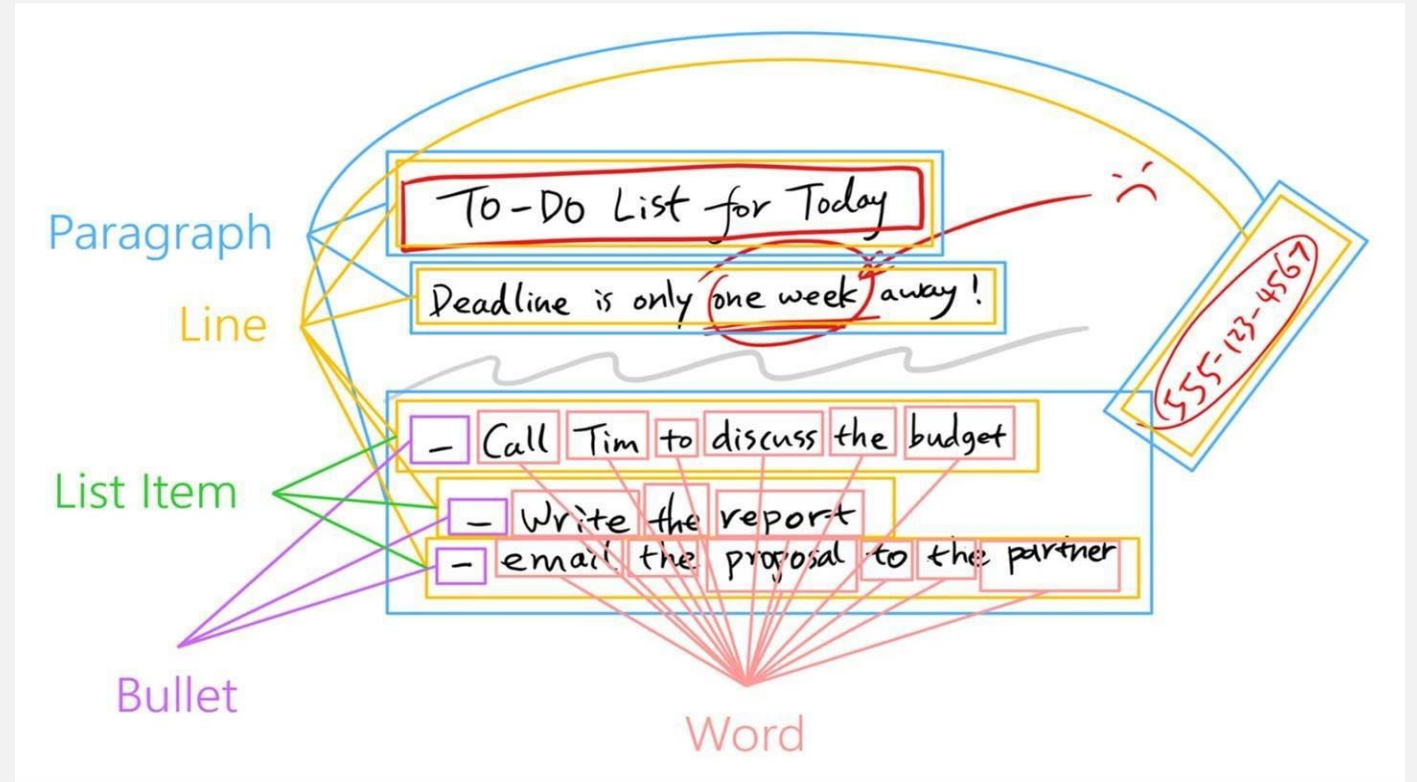
Machine Learning Makes Hard Things Easy

Computers tend to be good at things humans are bad at, and vice versa

- Good at chess
- Bad at recognising people

Machine Learning (ML) can redress the balance and make the computer good at stuff in a useful way

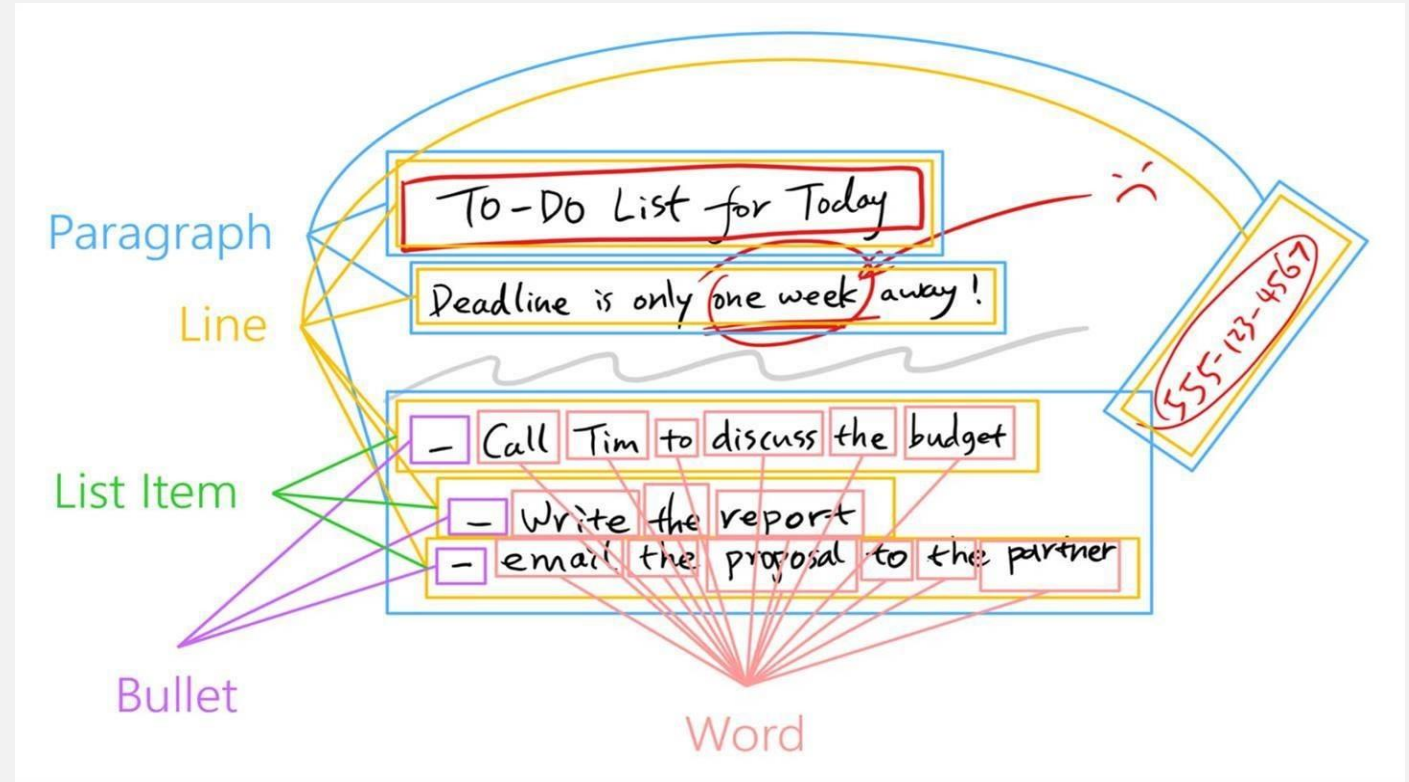# Machine Learning Makes Hard Things Easy

Computers tend to be good at things humans are bad at, and vice versa

- Good at chess
- Bad at recognising people

Machine Learning (ML) can redress the balance and make the computer good at stuff in a useful way

Even if we don't know precisely how it works…..

Building an Art Critic

I'm not very good with art

But I know a good picture when I see it

With that in mind, the starting point for a computerised art critic should be something that can recognise different objects

So, lets build one using Machine Learning

Building an Art Critic

I'm not very good with art

But I know a good picture when I see it

With that in mind, the starting point for a computerised art critic should be something that can recognise different objects

So, lets build one using Machine Learning

What we are going to do…

Use training data to build model for image recognition

Incorporate the training data in a Universal Windows Application

Deploy the application and run it on the target device

What we are going to do…

Use training data to build model for image recognition

Use Microsoft Azure to build the model in the cloud

Incorporate the training data in a Universal Windows Application

Deploy the application and run it on the target device

What we are going to do…

Use training data to build model for image recognition

Use Microsoft Azure to build the model in the cloud

Incorporate the training data in a Universal Windows Application

Drop the model into Visual Studio and see how our program can bind to it

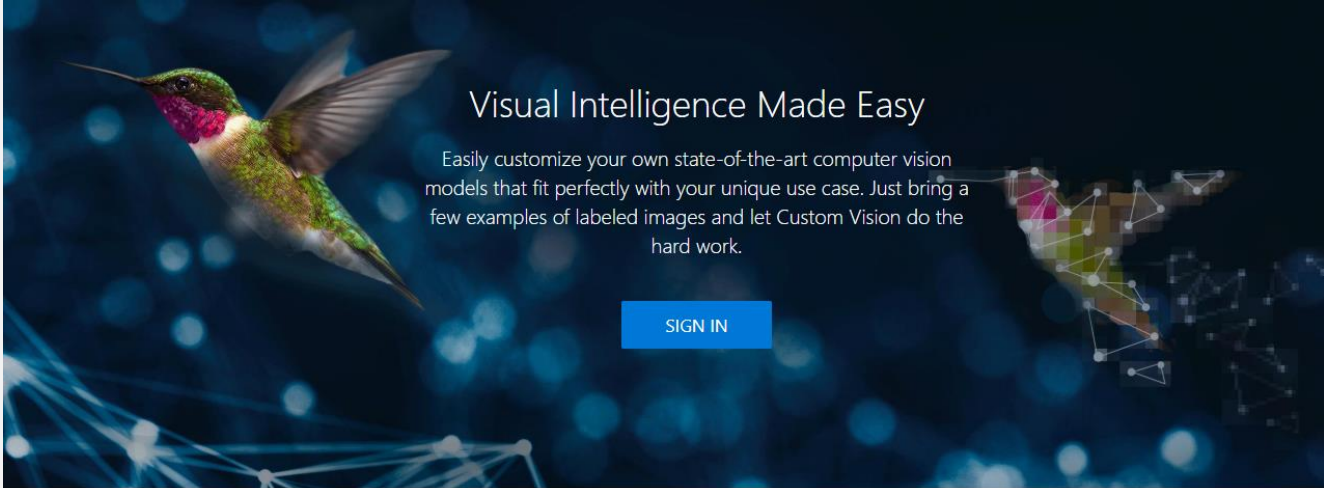Deploy the application and run it on the target device

What we are going to do...

Use training data to build model for image recognition

        Use Microsoft Azure to build the model in the cloud

Incorporate the training data in a Universal Windows Application

        Drop the model into Visual Studio and see how our program can bind to it

Deploy the application and run it on the target device

        Build and run the application

Microsoft Custom Vision

You can create your own models in the cloud using Microsoft Custom Vision

- Sign in using your Azure account

- Upload images

- Build the model

- Download it for use in your application
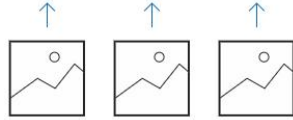
www.customvision.ai

Custom Vision Project

In demo mode you can create two projects with a limited number of figures and classification types

Plenty to get started
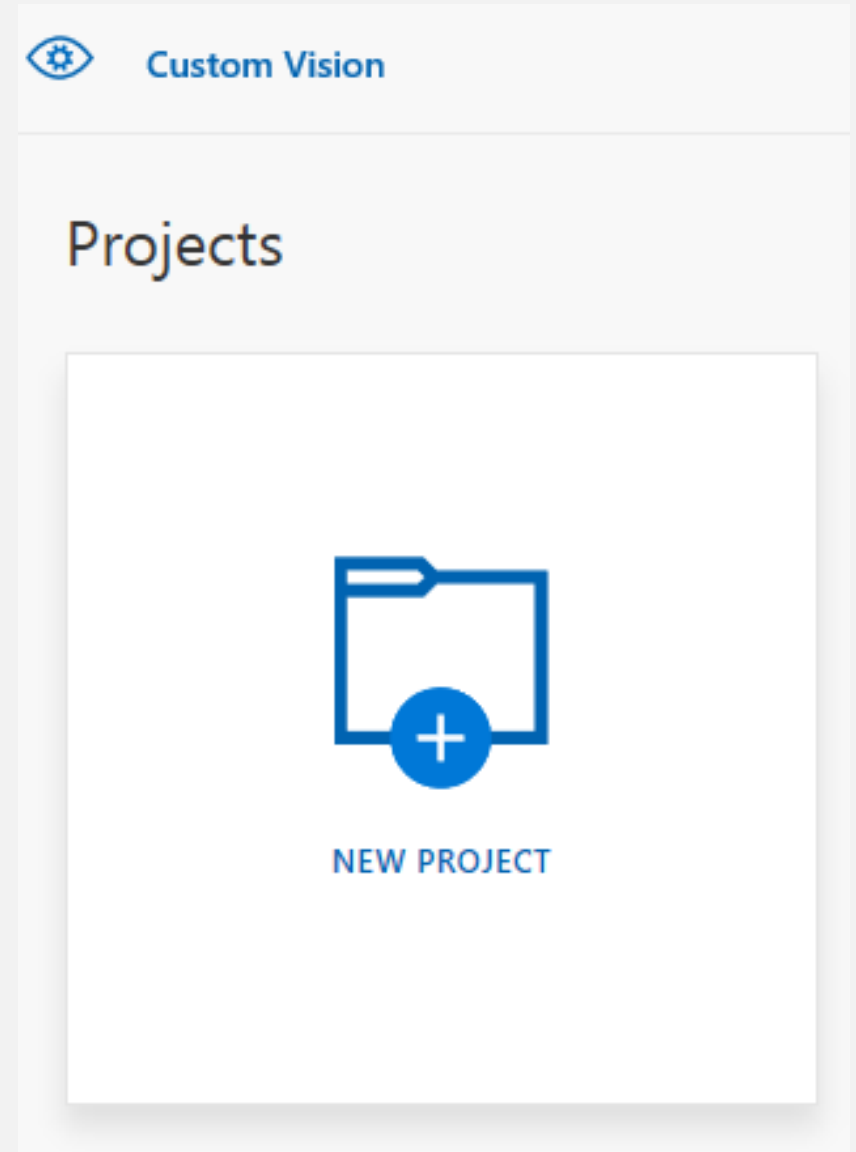
Make a new project

These are the properties of a new project

Create new project

Name*

Art Critic

Description

This project can recognise simple objects

Project Types ⓘ

◉ Classification
○ Object Detection

Classification Types ⓘ

○ Multilabel (Multiple tags per image)
◉ Multiclass (Single tag per image)

Domains ⓘ

○ General
○ Food
○ Landmarks
○ Retail
○ Adult
◉ General (compact)
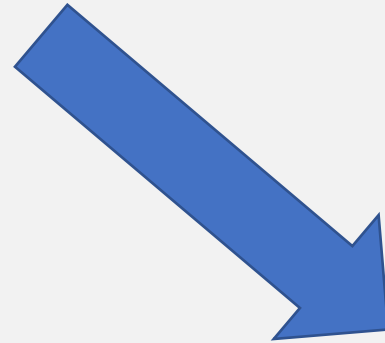○ Food (compact)
○ Landmarks (compact)
○ Retail (compact)

Cancel   Create project

Make a new project

These are the properties of a new project

Only compact models can be exported

Create new project   ✕

Name*

Art Critic

Description

This project can recognise simple objects

Project Types ⓘ

◉ Classification
○ Object Detection

Classification Types ⓘ

○ Multilabel (Multiple tags per image)
◉ Multiclass (Single tag per image)

Domains ⓘ

○ General
○ Food
○ Landmarks
○ Retail
○ Adult
◉ General (compact)
○ Food (compact)
○ Landmarks (compact)
○ Retail (compact)

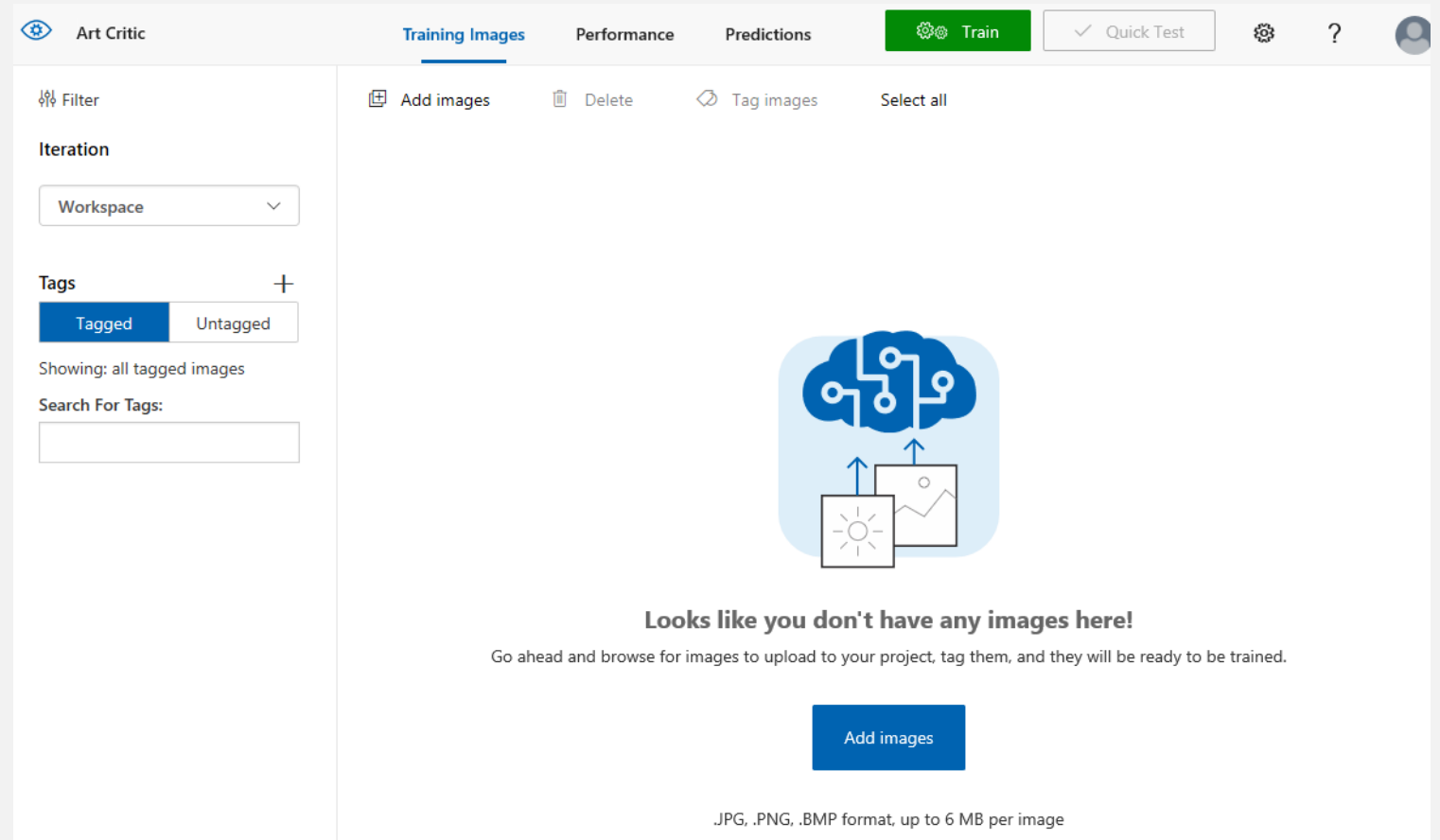Cancel   Create project

Images

Now that we have our project we can add some images

These can be in formats such as png or jpeg

Note that high resolution might not be your friend

The larger the image, the more data will have to be processed
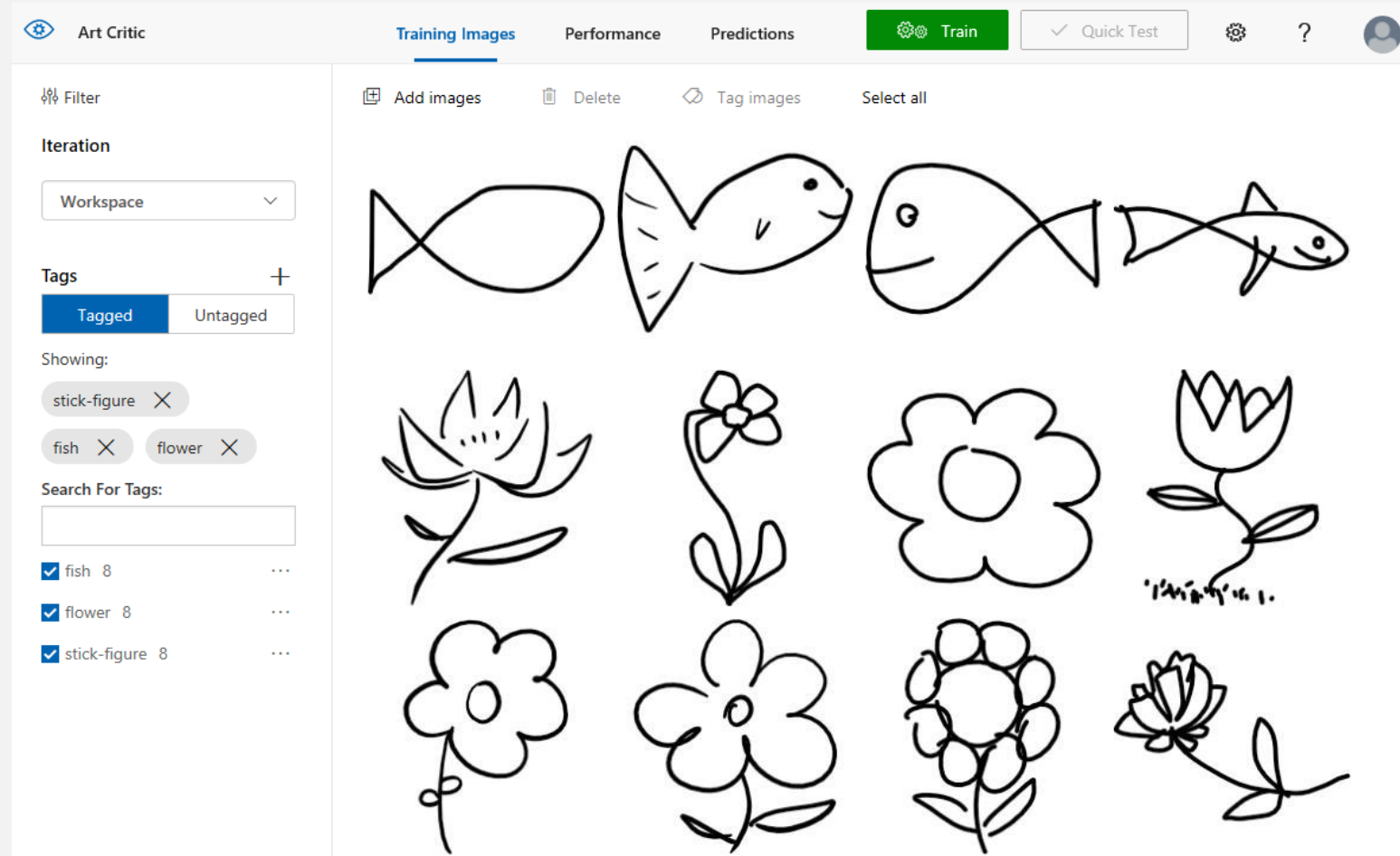
Add some images
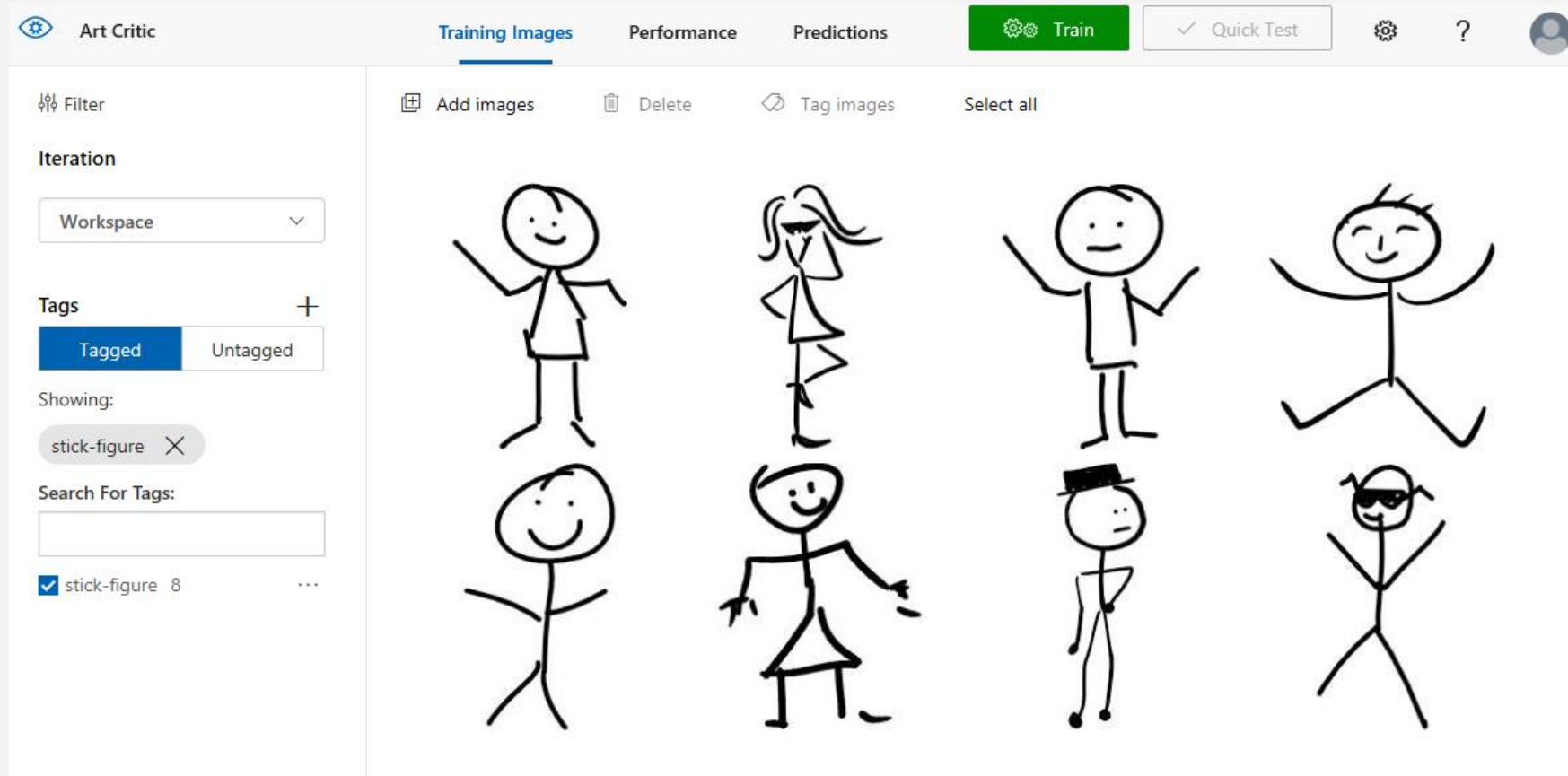
Choose your images carefully....

All images loaded

These are all the images loaded

Now we need to use this data to train a model

All images loaded

These are all the images loaded

Now we need to use this data to train a model

Press the Train button to start
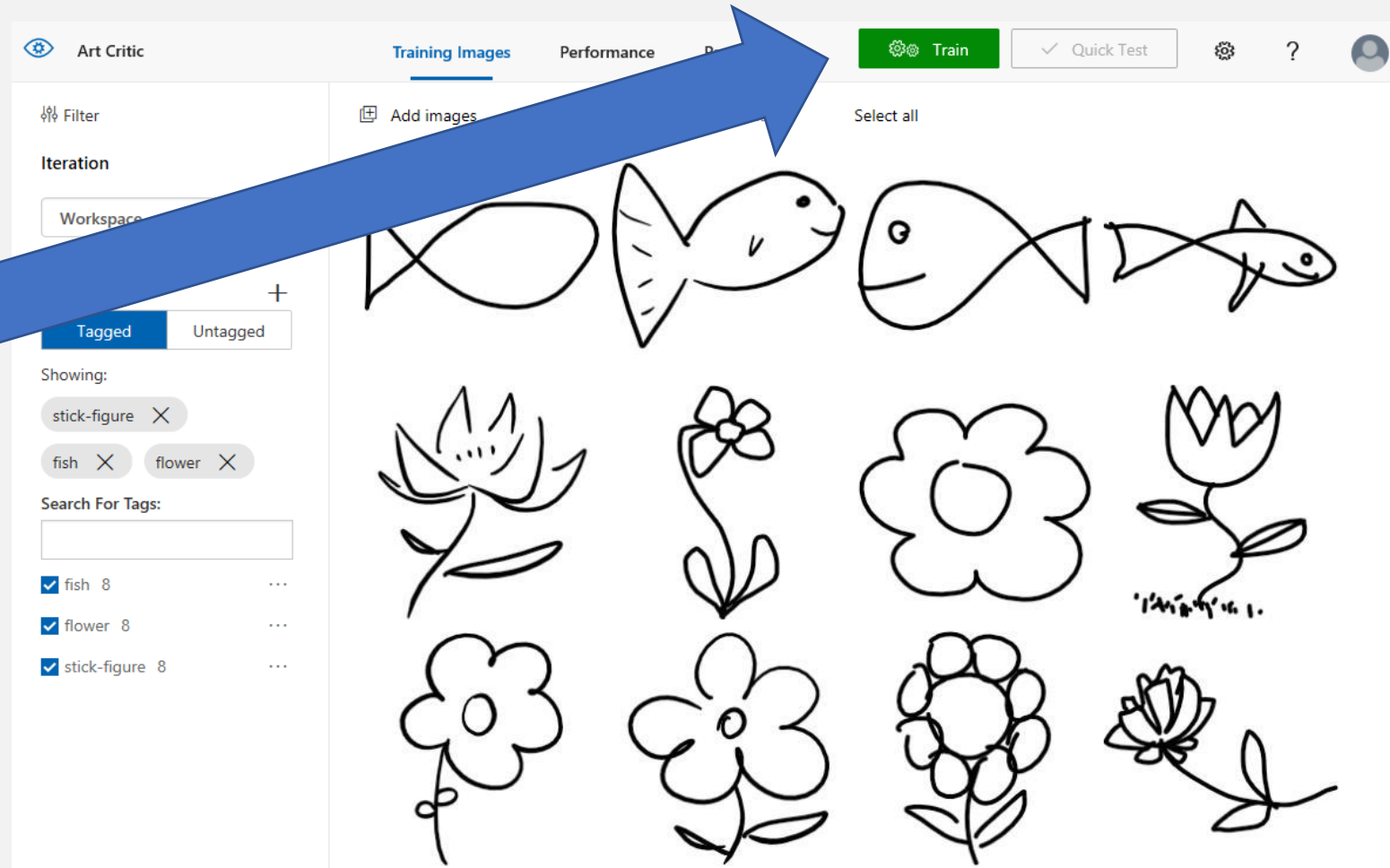
Training results

You get a report of how successful the training was

Precision:

A measure of the likelihood of a tag being predicted correctly

Recall:

Out of the tags that should be predicted correctly, what percentage did the model find

You can use this to tweak your training data

You can also test your model against data with the Quick Test button
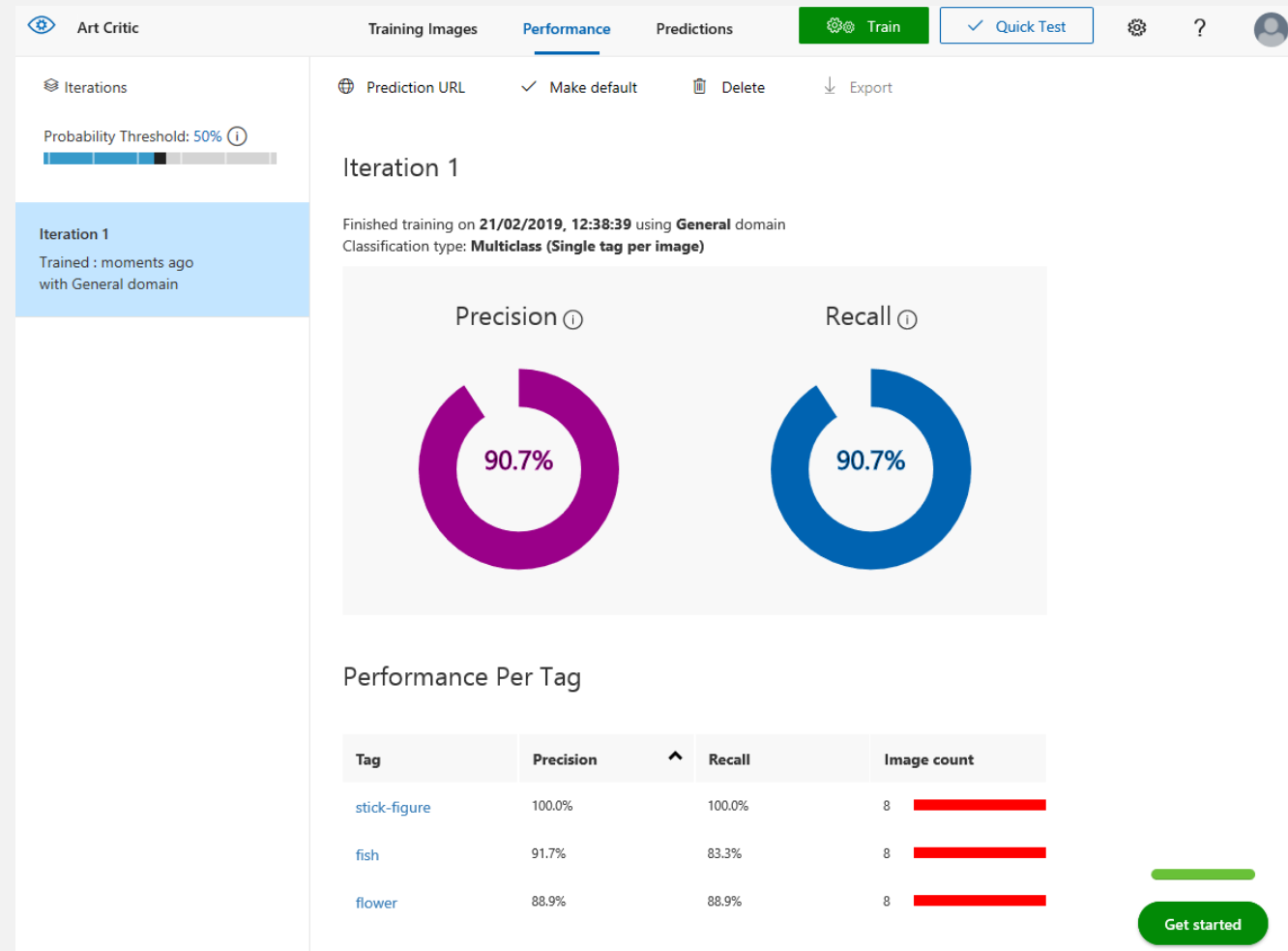
Exporting the model

We now have a model which can be used in applications to allow them to categorise an image based on the training data

We need to export the model to a file that we can incorporate in our application

Exporting to ONNX

We are going to use the ONNX format

Version 1.2

Visual Studio can import files of this format
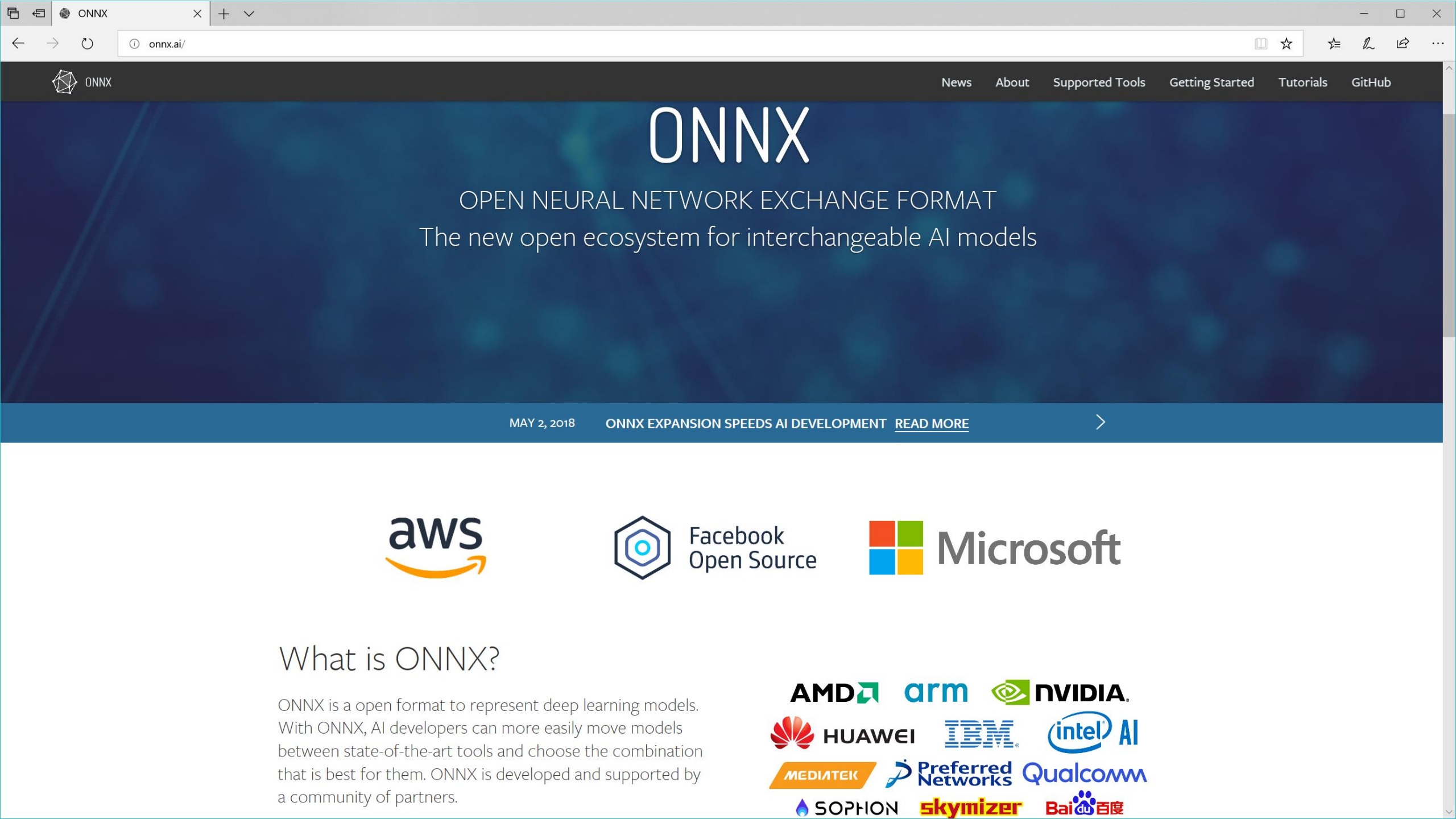
News   About   Supported Tools   Getting Started   Tutorials   GitHub

ONNX
OPEN NEURAL NETWORK EXCHANGE FORMAT
The new open ecosystem for interchangeable AI models

MAY 2, 2018   ONNX EXPANSION SPEEDS AI DEVELOPMENT   READ MORE

What is ONNX?

ONNX is a open format to represent deep learning models. With ONNX, AI developers can more easily move models between state-of-the-art tools and choose the combination that is best for them. ONNX is developed and supported by a community of partners.

Import the model into Visual Studio
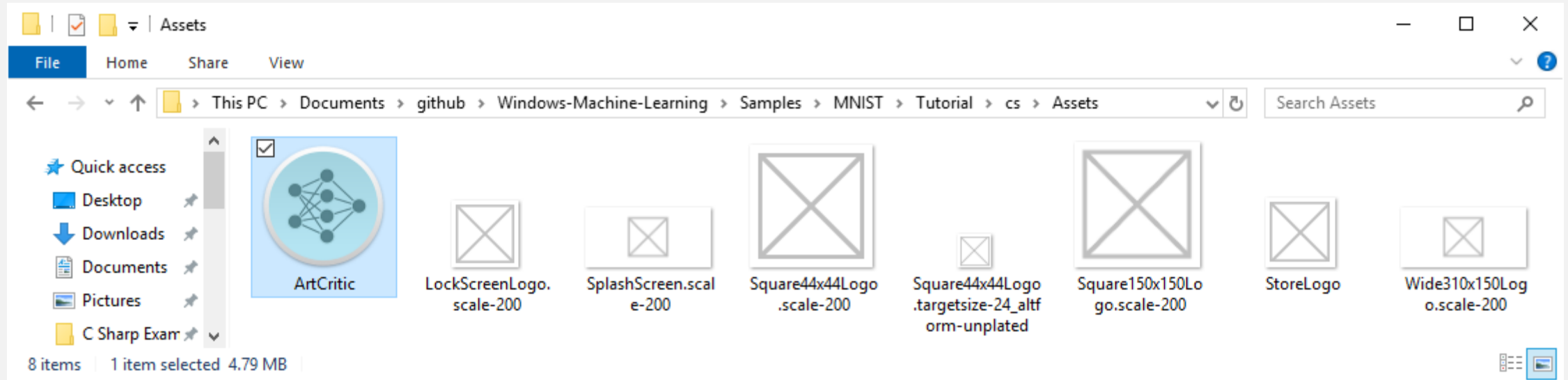
The model can now be imported into Visual Studio

Start by adding it to the Assets file for the application solution

Add the item to the project
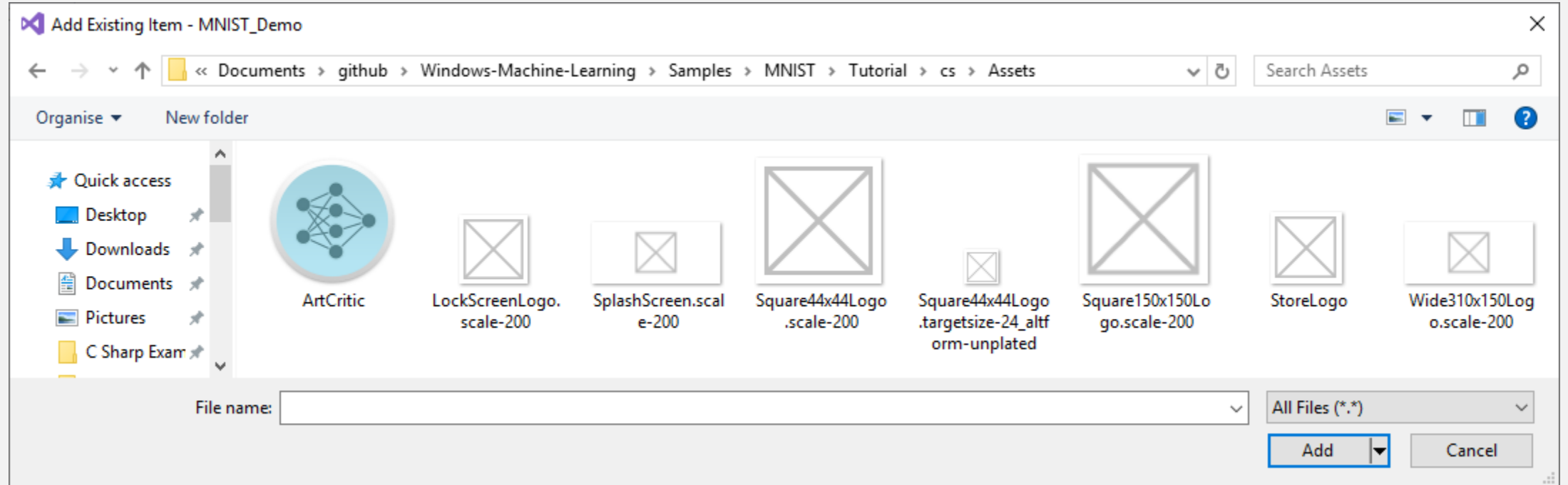
Visual Studio will import the model into the solution

The Model Interface File

When the model is imported into Visual Studio a source file is added to the project that the application can use as an interface to the model

This contains class definitions that describe the model, its input and its outputs
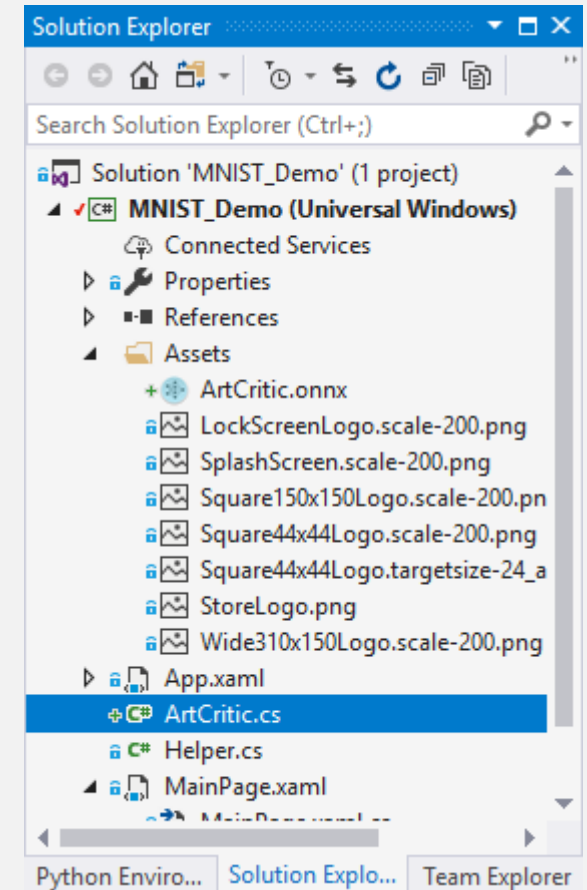
Using the Model

```
private ArtCriticModel ModelGen;
private ArtCriticInput ModelInput = new ArtCriticInput();
private ArtCriticOutput ModelOutput;
```

These variables provide the connection between the model and our application

When we want to recognise an object we need to create a `ModelInput` instance

    This happens when the user asks the program to recognise an object

The output from the recognition is supplied in the `ModelOutput` instance

    The program will display this result for the user

Of course your AI program could do anything it likes with the data

Using the Model

```csharp
private async void recognizeButton_Click(object sender, RoutedEventArgs e)
{
    //Bind model input with contents from InkCanvas
    VideoFrame vf = await helper.GetHandWrittenImage(inkGrid);

    ModelInput.data = ImageFeatureValue.CreateFromVideoFrame(vf);
    // Evaluate the model
    ModelOutput = await ModelGen.EvaluateAsync(ModelInput);

    //Display the results
    numberLabel.Text = ModelOutput.classLabel.GetAsVectorView()[0];
}
```

This is the function that runs when the art critic is asked to "recognise" an image

Using the Model

```csharp
private async void recognizeButton_Click(object sender, RoutedEventArgs e)
{
    //Bind model input with contents from InkCanvas
    VideoFrame vf = await helper.GetHandWrittenImage(inkGrid);

    ModelInput.data = ImageFeatureValue.CreateFromVideoFrame(vf);
    // Evaluate the model
    ModelOutput = await ModelGen.EvaluateAsync(ModelInput);

    //Display the results
    numberLabel.Text = ModelOutput.classLabel.GetAsVectorView()[0];
}
```

This code gets the image that the user has drawn

This turns out to be quite easy

Using the Model

```csharp
private async void recognizeButton_Click(object sender, RoutedEventArgs e)
{
    //Bind model input with contents from InkCanvas
    VideoFrame vf = await helper.GetHandWrittenImage(inkGrid);

    ModelInput.data = ImageFeatureValue.CreateFromVideoFrame(vf);
    // Evaluate the model
    ModelOutput = await ModelGen.EvaluateAsync(ModelInput);

    //Display the results
    numberLabel.Text = ModelOutput.classLabel.GetAsVectorView()[0];
}
```

The `ImageFeatureValue` class is the way that a program passes images into an ML application

This code creates an instance of ImageFeatureValue from the captured video and then sets the data property of the model input to this object

Using the Model

```
private async void recognizeButton_Click(object sender, RoutedEventArgs e)
{
    //Bind model input with contents from InkCanvas
    VideoFrame vf = await helper.GetHandWrittenImage(inkGrid);

    ModelInput.data = ImageFeatureValue.CreateFromVideoFrame(vf);
    // Evaluate the model
    ModelOutput = await ModelGen.EvaluateAsync(ModelInput);

    //Display the results
    numberLabel.Text = ModelOutput.classLabel.GetAsVectorView()[0];
}
```

This method call is where the Model is used to analyse the input

Note that this call is performed asynchronously

It delivers an instance of the result

Using the Model

```csharp
private async void recognizeButton_Click(object sender, RoutedEventArgs e)
{
    //Bind model input with contents from InkCanvas
    VideoFrame vf = await helper.GetHandWrittenImage(inkGrid);

    ModelInput.data = ImageFeatureValue.CreateFromVideoFrame(vf);
    // Evaluate the model
    ModelOutput = await ModelGen.EvaluateAsync(ModelInput);

    //Display the results
    numberLabel.Text = ModelOutput.classLabel.GetAsVectorView()[0];
}
```
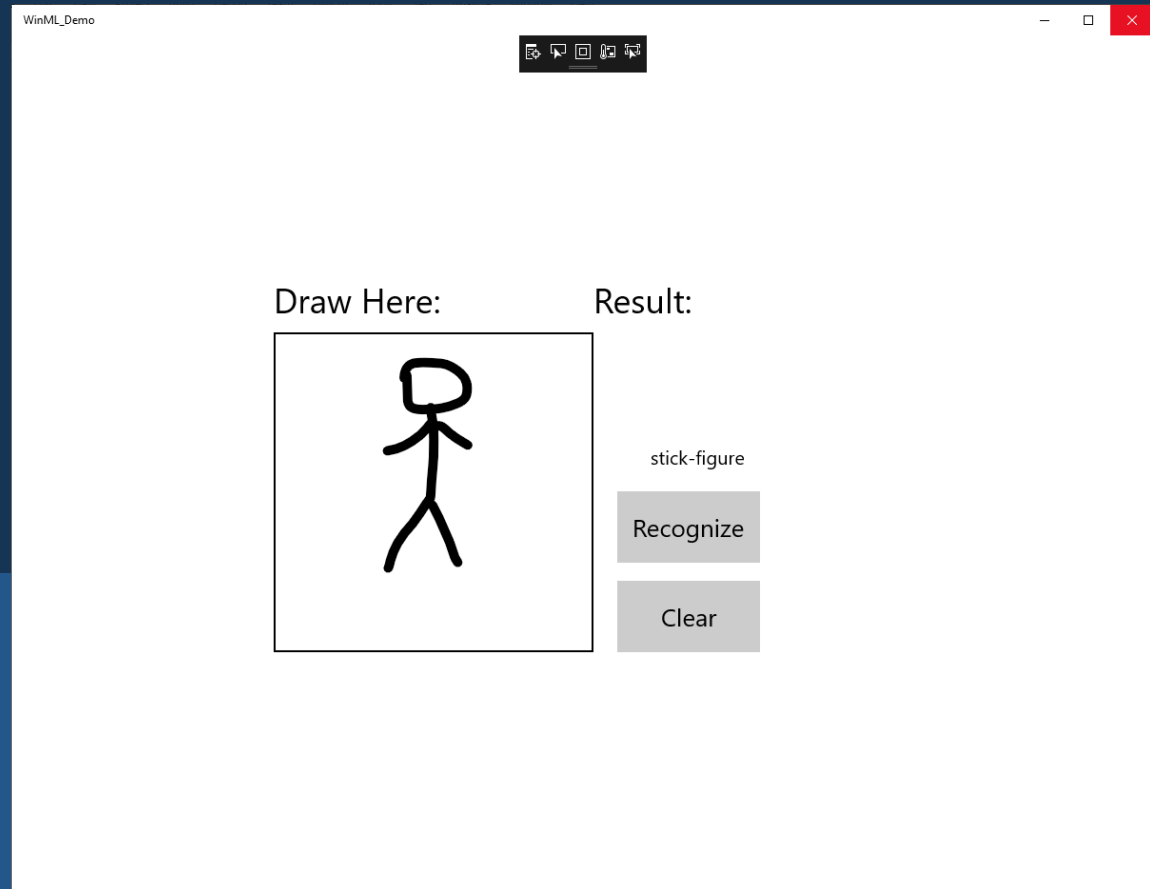
The output for this model is in the form of a `TensorString`

   a `TensorString` is how the Machine Learning framework represents collections of strings

We just have to pull out one string from this, which is the name of the category recognised category

DEMO

Running the Art Critic

Summary

It is very easy to create models and use them

The model is created in the cloud but used in the device

    This is "edge" ai – no need for even a network connection

    The device can even be a Raspberry Pi

    The ai model can run on a GPU if one is available

Visual Studio can import and use ONNX models directly and will make the wrapper classes for you

What you do with the model is entirely up to you

You can download the code and this presentation from:

        https://github.com/CrazyRobMiles/MLArt-Critic

www.robmiles.com