

# Modell mit ML.NET trainieren

Gabriel Beutl & Schwab Alexander – AKT – Teil 1

## Contents

Einleitung.....	1
Installation.....	1
Vorbereitung .....	2
Validierung .....	5
Integration.....	5
Fazit .....	6

## Einleitung

Aufgrund von Interesse und als kleiner Ausblick wird das mit Keras und Python (Tensorflow) trainierte Modell in ML.Net nachgestellt und danach in eine .NET-Anwendung integriert.

## Installation

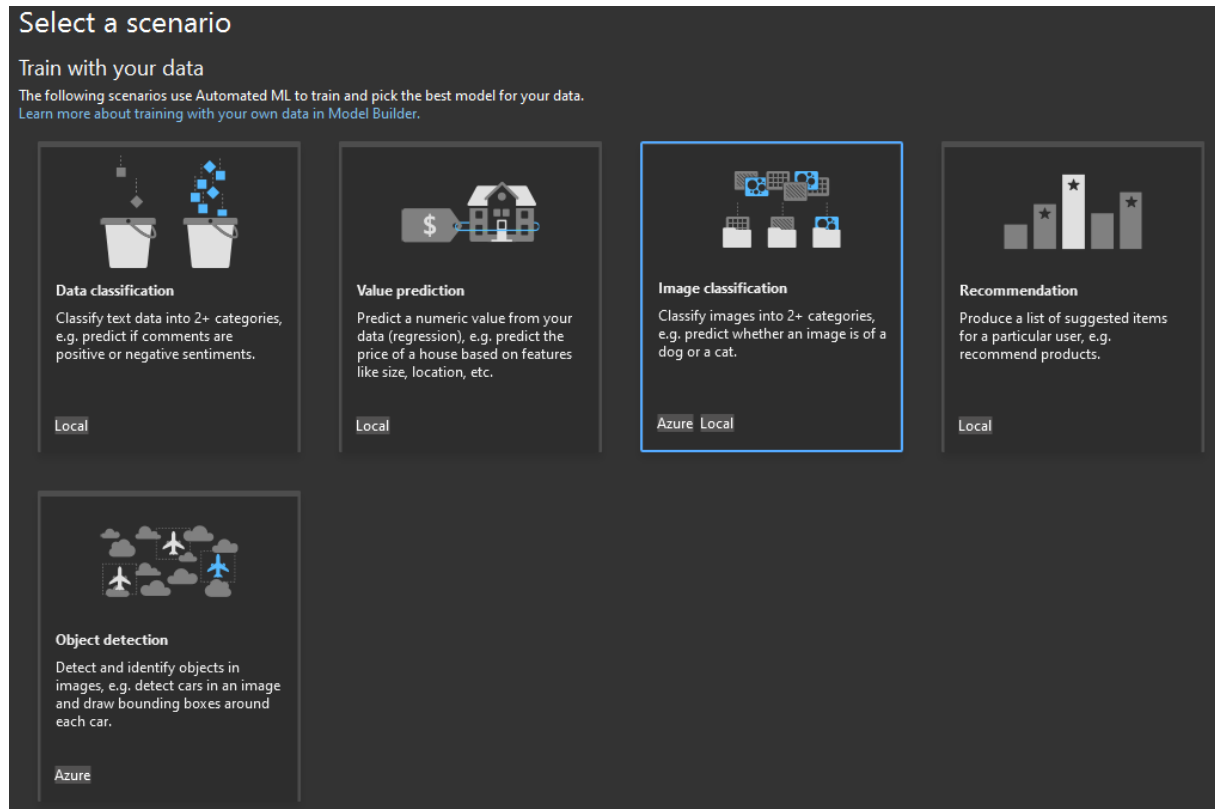
Damit ein Modell mit ML.NET lokal trainiert werden kann, sollte am besten direkt **Visual Studio 2022** installiert werden.

Später, bevor das Modell trainiert wird, können noch Erweiterungen zur GPU-Unterstützung direkt aus dem Visual Studio heraus installiert werden.

## Vorbereitung

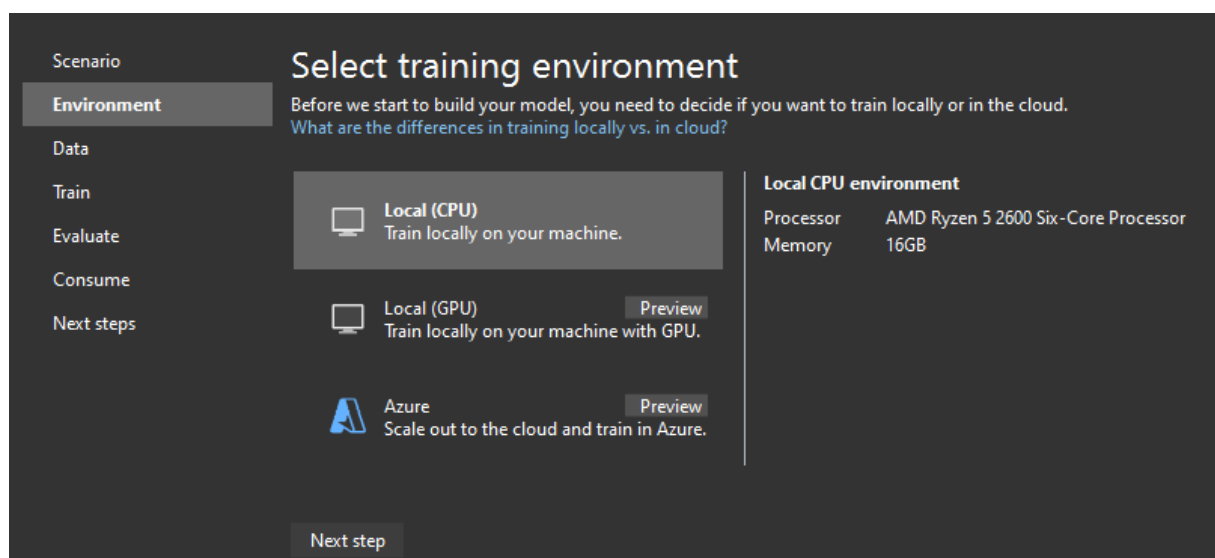
Nachdem ein Projekt erstellt wurde, kann über das Kontext-Menü -> Add -> **Machine Learning Model** ein neues Modell hinzugefügt werden.

Aus einer Vielzahl an Möglichkeiten zum Trainieren wird hier die Image-Classification ausgewählt.



## Training

Zuerst muss ausgewählt werden, welche Plattform für das Training verwendet werden soll, hier wird in unserem Fall CPU ausgewählt.



Mit der nötigen Hardware können Erweiterungen installiert werden, um auch die GPU für das Training verwenden zu können.

## Select training environment

Before we start to build your model, you need to decide if you want to train locally or in the cloud.  
What are the differences in training locally vs. in cloud?



Local (CPU)  
Train locally on your machine.



Local (GPU)  
Train locally on your machine with GPU.

Preview



Azure  
Scale out to the cloud and train in Azure.

Preview

Next step

### Local GPU environment

How do I set up my local GPU for training?

Graphics card 1 NVIDIA GeForce GTX 1060 6GB

Check compatibility

- ❌ GPU extension is missing. [Install extension.](#)
- ✅ GPU is CUDA compatible
- ❌ CUDA 10.1 is missing. Please install CUDA 10.1 and ensure \$env:CUDA\_PATH
- ❌ cuDNN 7.6.4 is missing. [Install cuDNN.](#)

Im nächsten Schritt wird der Ordner mit den Trainings-Bildern ausgewählt, die Bilder, welche den einzelnen Labels zugeordnet sind, müssen sich in Unterordnern mit deren Namen befinden.

## Add data

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

### Input

Select the folder which contains all your images. This folder should organize your photos into separate labeled sub-folders.

Select a folder:

C:\FH\AKT1\plant\_images\train



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

Example folder structure:

- Sample Folder
  - Sub Folder Label 1
    - Image 1
    - Image 2
  - Sub Folder Label 2
  - Sub Folder Label 3

### Data Preview

Total images 1822. Showing 8/910.

train:

Apple\_he... (910)

Apple\_un... (912)



Next step

Danach wird das Training gestartet.

Scenario

Environment

Data

**Train**

Evaluate

Consume

Next steps

## Train

Model Builder automatically sets the training time based on the size of your dataset.

Training setup summary ▾

Train again

✓ Training complete

### Training results

Best accuracy:	98,37%
Best model:	DNN + ResNet50
Training time:	2504,30 seconds
Models explored (total):	1
Generated code-behind:	MLModel.consumption.cs, MLModel.training.cs

Next step

Nach 2500 Sekunden wurde ein Modell mit einer Accuracy von 98,37% trainiert, ML.NET entschied sich hier für ein auf DNN + ResNet50 basierendes Modell.

Scenario

Environment

Data

Train

**Evaluate**

Consume

Next steps

## Evaluate

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

**Best model:**

**Accuracy:** 98,37%

**Model:** ImageClassification

### Try your model

[Browse an image](#)  
(.png, .jpg, .jpeg, .gif only,  
8 MB max size)

Next step


#### Results

Select an image to get its predicted result

## Validierung

Das Modell kann direkt in der UI mit Testbildern getestet werden und liefert die erwarteten Ergebnisse.

### Try your model




Try another image

#### Results

Apple_healthy	98%
Apple_unhealthy	2%

### Try your model



Try another image

#### Results

Apple_unhealthy	100%
Apple_healthy	< 1%

## Integration

Mit folgendem Code wurde dieses Modell nun in eine von uns im 2. Semester entwickelte .NET 6.0 Anwendung integriert:

```
public string SaveImage(ImageUpdated inputImage)
{
```

```

        byte[] bytes = Convert.FromBase64String(inputImage.Content);

        var dir =
Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);
        var path = Path.Combine(dir, "images/" + inputImage.ImageId +
".jpg");

        var imagesDir = Path.Combine(dir, "images");
        if (!Directory.Exists(imagesDir))
        {
            Directory.CreateDirectory(imagesDir);
        }

        File.WriteAllBytes(path, bytes);
        return path;
    }

    public async Task EvaluateHealthinessAsync(ImageUpdated imageUpdatedEvent)
    {
        // load image from imageservice

        if(imageUpdatedEvent != null && imageUpdatedEvent.Content.Any())
        {
            //save image locally
            string localImageUrl = SaveImage(imageUpdatedEvent);

            var predictionInput = new MLModel.ModelInput()
            {
                ImageSource = localImageUrl
            };

            //Load model and predict output
            var result = MLModel.Predict(predictionInput);
            var score = result.Score[0];
            Console.WriteLine(score);

            await this.daprClient.SaveStateAsync(StoreName,
imageUpdatedEvent.ImageId.ToString(), new PlantHealth()
            {
                ImageId = imageUpdatedEvent.ImageId,
                Score = score
            });
        }
    }
}

```

## Fazit

Mit ML.Net kann sehr schnell ein Modell erstellt und im Projekt verwendet werden. Standardprobleme können hier sehr effizient gelöst werden.