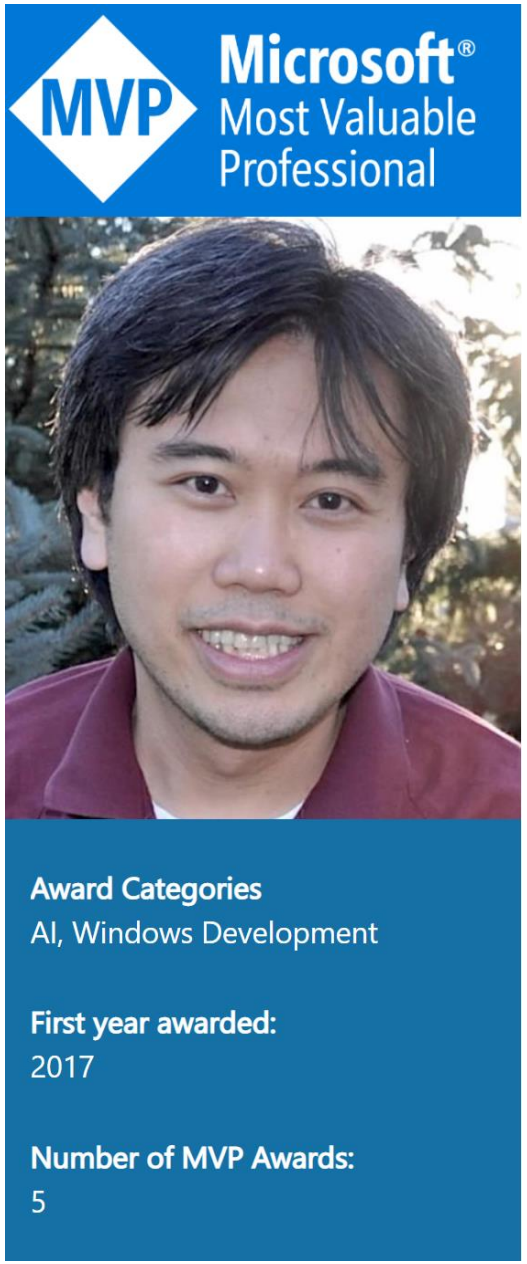
# Build Intelligent applications with ML.NET and Windows Machine Learning

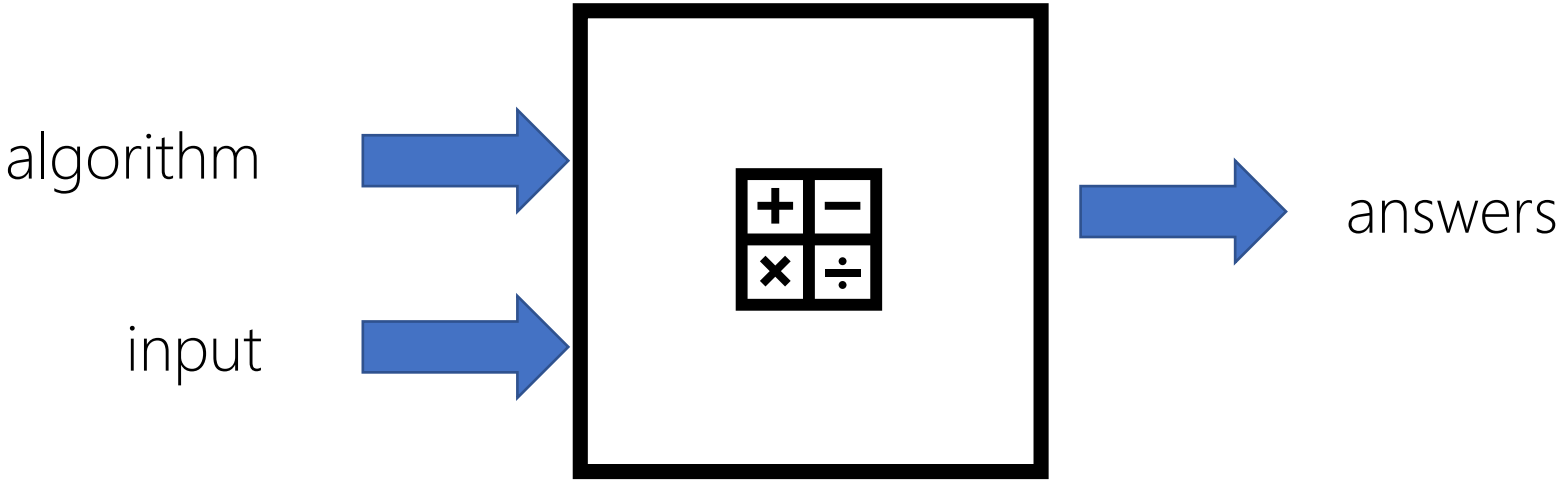
Ron Dagdag



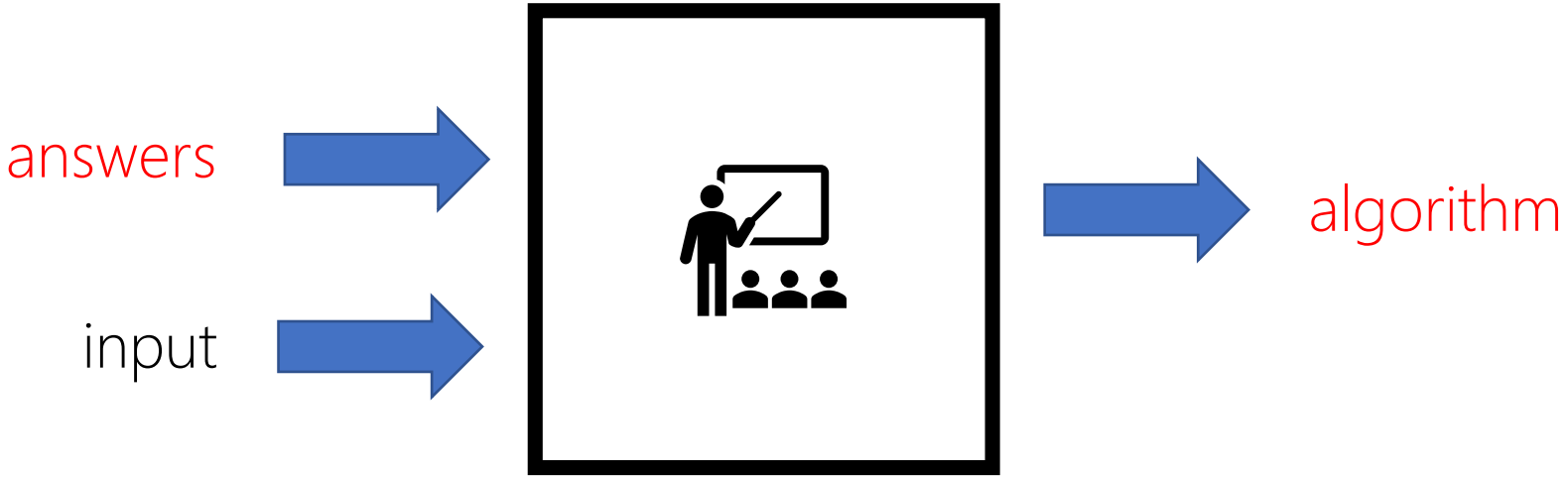
# Agenda

- What is Machine Learning?
- Windows Machine Learning
- Open Neural Network Exchange (ONNX)
- ONNX Runtime
- Community Toolkit - Intelligent API
- ML.NET Model Builder
- Demo

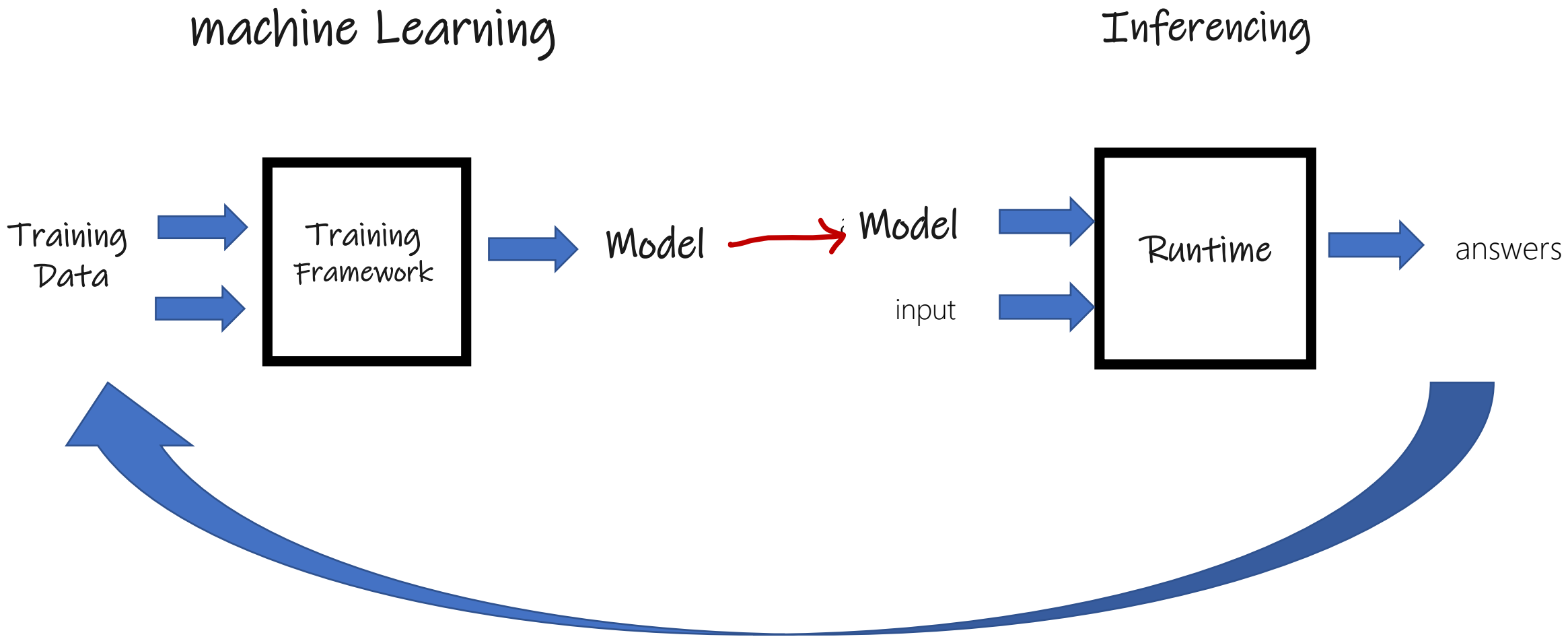
# programming



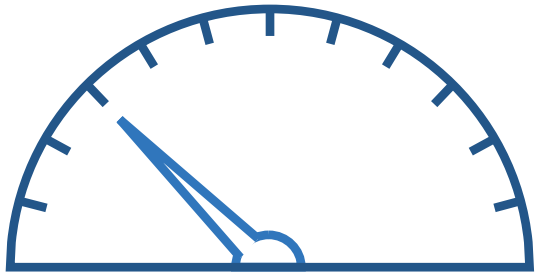
# machine learning



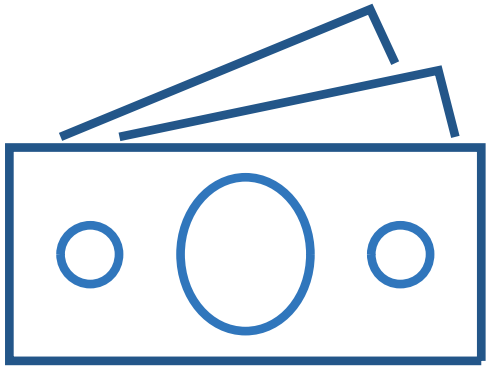
# ML Primer


machine Learning
Inferencing
Training Data
Training Framework
Model
Model
input
Runtime
answers

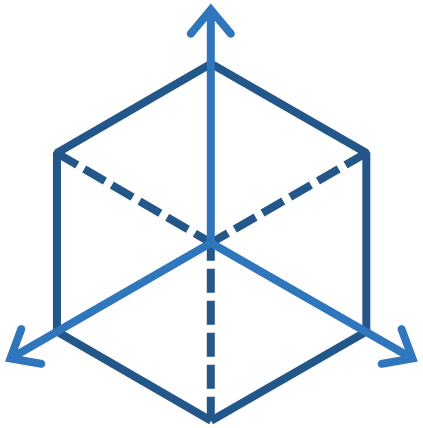
# AI on the edge



Low latency

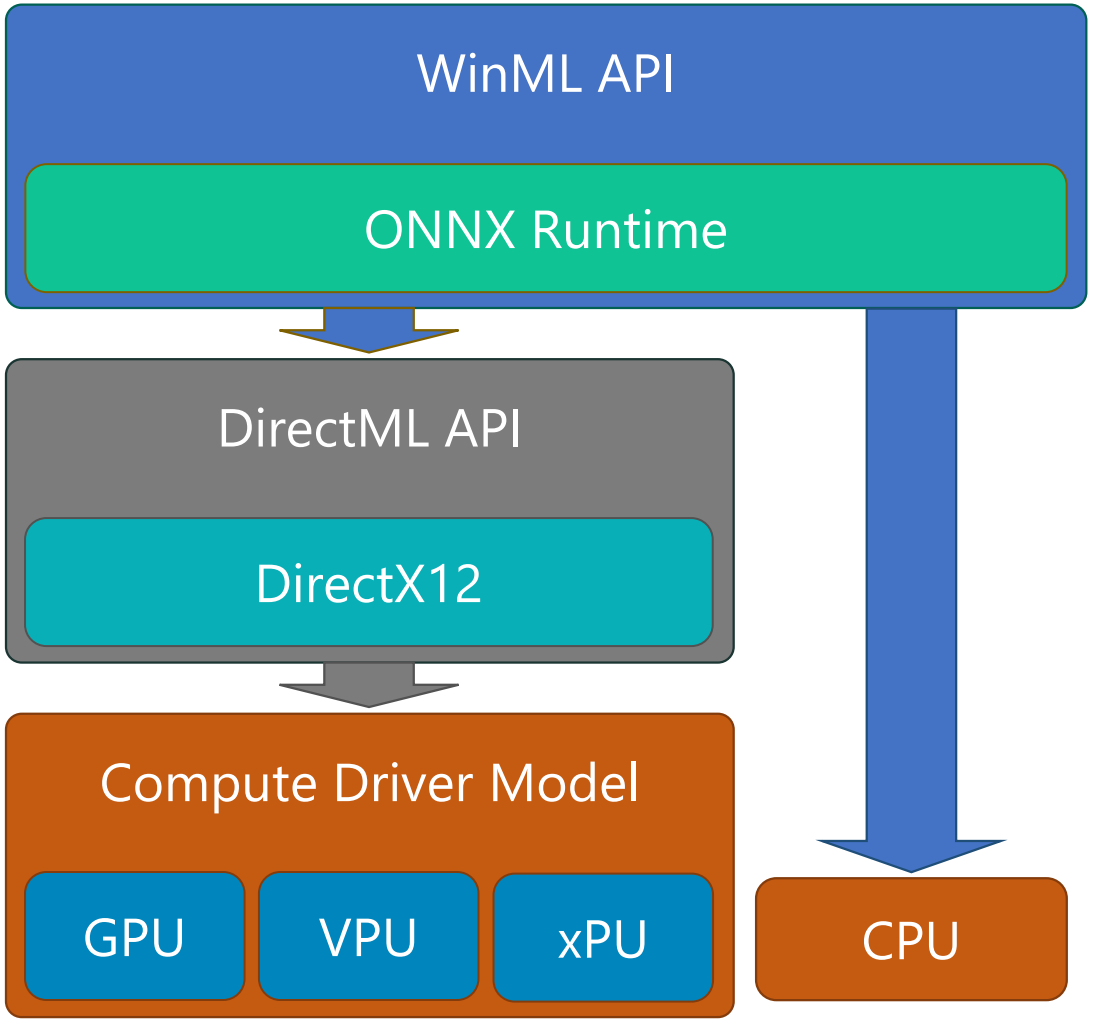


Scalability



Flexibility

# Windows AI platform




- WinML
  - **Practical**, simple model-based API for ML inferencing on Windows
- DirectML
  - **Realtime, high control** ML operator API; part of DirectX family
- Compute Driver Model
  - Robust **hardware reach**/abstraction layer for compute and graphics silicon

# Windows Machine Learning (WinML)

- Ease of development
- Abstract model-specific code away
- Broad hardware support
- Performs hardware optimizations
- Implement Machine Learning in Windows apps using Windows ML

# Windows Machine Learning (WinML)

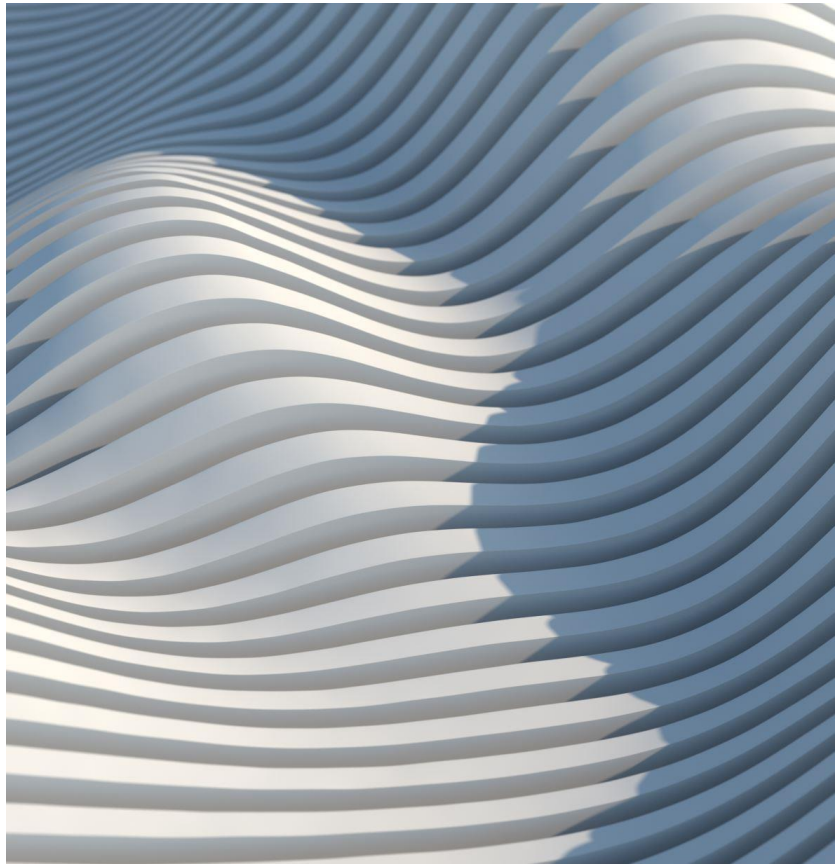
- Improve performance significantly on Windows
- high-performance
- Low latency, real-time results
- Increased flexibility
- Reduced operational costs
- Reliable API for deploying hardware-accelerated ML inferences on Windows devices

# WinForm


# DEMO

# Intelligent API

- Machine learning tasks easier for devs
- No ML expertise need
- Reuse existing ML models
- Add Nuget package and calling a function
- Inferencing machine learning models on Windows
- Each APIs employs WinML



# Intelligent API

https://github.com/CommunityToolkit/Labs-IntelligentAPIs

- Add a new nuget source with the feed URL

  https://pkgs.dev.azure.com/dotnet/CommunityToolkit/_packaging/CommunityToolkit-Labs/nuget/v3/index.json

- Add nuget package to application

  CommunityToolkit.Labs.Intelligent.ImageClassification

  CommunityToolkit.Labs.Intelligent.ObjectDetection

  CommunityToolkit.Labs.Intelligent.EmotionRecognition

# Intelligent API

- Reference Library

```
using CommunityToolkit.Labs.Intelligent.ImageClassification;
```

- Call Classify Image

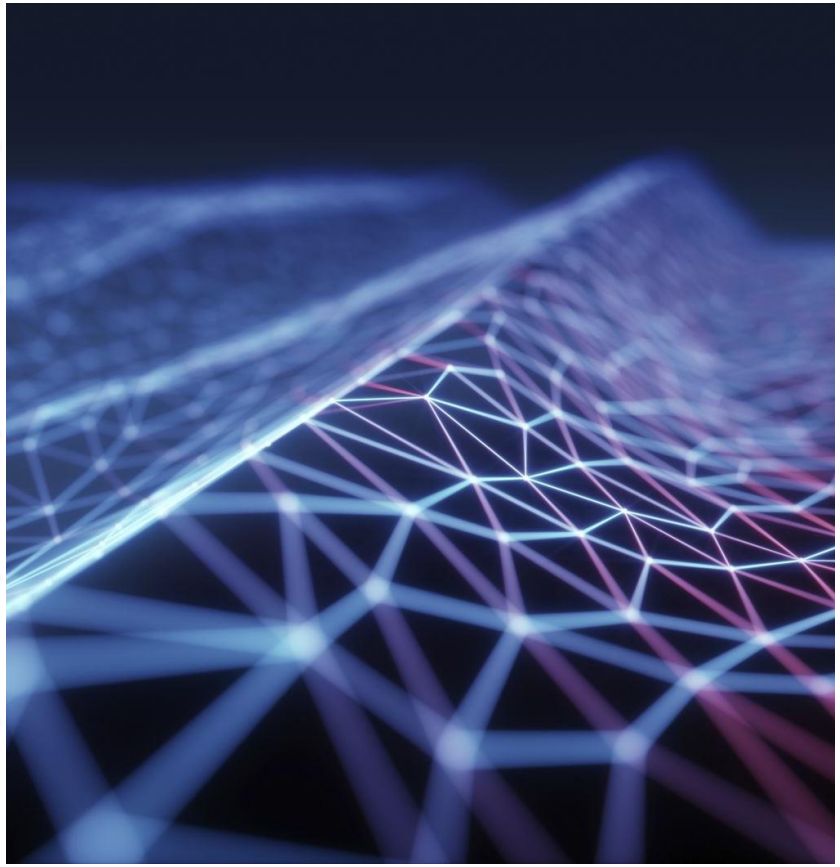
```
List<ClassificationResult> list = await
SqueezeNetImageClassifier.ClassifyImage(selectedStorageFile, 3);
```

# Intelligent API DEMO

# Open and Interoperable AI

# When to use ONNX?

- Trained in Python - deploy into a C#/Java/Javascript app
- High Inferencing latency for production use
- Model to run resource on IoT/edge devices
- Model to run on different OS or Hardware
- Combine models created from different frameworks
- Training takes too long (transformer models)



# ONNX Runtime

onnxruntime.ai

**Optimize Inferencing** | Optimize Training

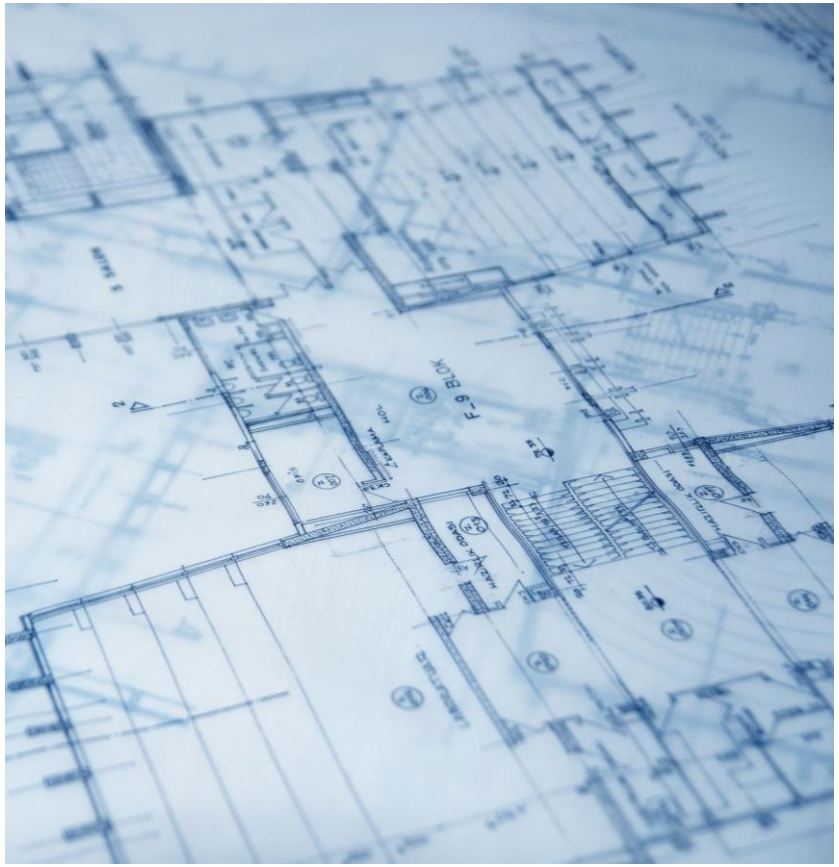
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Platform** | Windows | Linux | Mac | Android | iOS | Web Browser (Preview) | |
| **API** | Python | C++ | C# | C | Java | JS | Obj-C | WinRT |
| **Architecture** | X64 | X86 | ARM64 | ARM32 | IBM Power | | |
| **Hardware Acceleration** | Default CPU | CoreML | CUDA | DirectML | oneDNN | | |
| | OpenVINO | TensorRT | NNAPI | ACL (Preview) | ArmNN (Preview) | | |
| | MIGraphX (Preview) | TVM (Preview) | Rockchip NPU (Preview) | Vitis AI (Preview) | | | |
| **Installation Instructions** | Install Nuget package **Microsoft.ML.OnnxRuntime.Gpu**<br>Refer to **docs** for requirements. | | | | | | |

# ML.NET

- add machine learning to .NET applications
- online or offline
- Add automatic predictions to apps
- ML.NET can generate machine learning **model**.
- model - steps to transform input data into a prediction
- import pre-trained TensorFlow and ONNX models
- Supports Windows, Linux, and macOS

# ML.NET Model Builder

- Simple UI tool in Visual Studio
- Runs locally to build, train and ship ML projects
- build/train in Azure
- Generates Custom ML models



# Model Builder

Model Builder supports the following environment options:

| Scenario | Local CPU | Local GPU | Azure GPU |
|---|---|---|---|
| Data classification | ✓ | ✗ | ✗ |
| Value prediction | ✓ | ✗ | ✗ |
| Image classification | ✓ | ✓ | ✓ |
| Recommendation | ✓ | ✗ | ✗ |
| Object detection | ✗ | ✗ | ✓ |

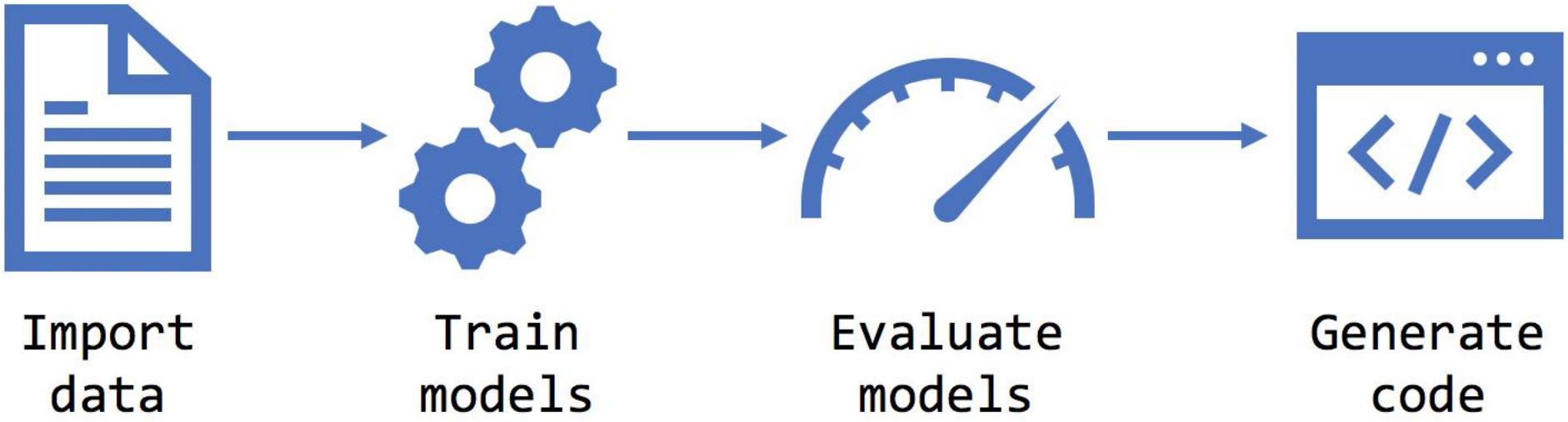
# Model Builder

| Dataset size | Average time to train |
|---|---|
| 0 - 10 MB | 10 sec |
| 10 - 100 MB | 10 min |
| 100 - 500 MB | 30 min |
| 500 - 1 GB | 60 min |
| 1 GB+ | 3+ hours |

These numbers are a guide only. The exact length of training is dependent on:

- the number of features (columns) being used to as input to the model
- the type of columns
- the ML task
- the CPU, disk, and memory performance of the machine used for training

# Model Builder



# Model Builder DEMO

# About Me

## Ron Dagdag

**Award Categories**
AI, Windows Development

**First year awarded:**
2017

**Number of MVP Awards:**
5

https://linktr.ee/rondagdag

- Lead Software Engineer at Spacee
- 5th year Microsoft MVP awardee
- www.dagdag.net
- ron@dagdag.net
  @rondagdag
- Linked In
  www.linkedin.com/in/rondagdag/
- Thanks for geeking out with me about ML.NET