Microsoft

Windows Machine Learning: Implement AI in your Windows apps

Contents

Introduction

At Microsoft, artificial intelligence (AI) is all about amplifying human ingenuity with technology, which is why we're continuing to make huge investments in AI across the company. Today, AI in Office 365 helps increase productivity at work, AI in the Windows 10 photo app makes it easier to create videos and search photo collections, and AI helps Windows Hello recognize your face and quickly log you in. We're also using AI to help advertisers build deeper connections with customers, answer your questions when using Bing Search or Cortana, and continually assess the behavior of Azure services like SQL Database to automatically tune performance and improve security and reliability.

All these examples of AI are powered by machine learning (ML), which lets computers predict future results based on historical relationships and trends within a set of data. With our release of Windows Machine Learning (Windows ML) in the October 2018 update for Windows 10, we're further democratizing AI for developers by delivering the machine learning advances that we've been building into our own apps, services, and cloud platform as part of Windows 10—so that every developer can use them to deliver more powerful, engaging experiences. More specifically, in technical terms, with the October 2018 update, developers can now train their AI models in the cloud, and then evaluate those models on a Windows 10 device running "AI at the edge."

Here's an example: Let's assume you have maintenance history for a multimillion-dollar machine that's connected to an industrial controller running Windows 10 IoT on the shop floor. Let's also assume you have historical telemetry from that machine from before it failed, such as vibration, operating temperature, and power draw. With machine learning, you can ask the question "What changed in the historical telemetry data before each failure?" (i.e., training your ML model in the cloud) and then push those insights to the Windows 10 IoT device that's controlling or monitoring the machine. It can watch for similar behavior in real time (i.e., evaluating your ML model at the edge) to *predict* when the machine might fail before it does—and brings production to a halt. By sending the new telemetry data collected during that event back to the cloud to further refine your ML model, and then pushing that new ML model down to the local Windows 10 IoT device, you can create a virtuous cycle of continuous improvement that will further enhance your ability to predict failures over time.

While this example is specific to manufacturing, the ways you can harness the power of Windows ML are virtually endless. Some examples include video processing to better understand in-store traffic and behavior, facial recognition for security cameras, and natural language processing and contextual understanding in e-commerce scenarios. Other examples are improved intelligence for robotics and drones, and a multitude of human-computer interaction scenarios where machine learning can help computers interact with people in novel, more meaningful and productive ways.

At this point, you may be thinking, "Why not just do it all in the cloud?" By enabling the evaluation of pretrained ML models on Windows devices, Windows ML provides several advantages:

- **Low latency, real-time results.** ML models can be evaluated using the local processing capabilities of the Windows device, enabling local, real-time analysis of large data volumes, such as images and video. Results are available quickly and efficiently for use in performance-intensive workloads like game engines, or background tasks such as indexing for search.

- **Increased flexibility.** The option to evaluate ML models on Windows devices lets you address a broader range of scenarios. For example, evaluation of ML models can run while the device is offline, or when faced with intermittent connectivity. This also lets you address scenarios where not all data can be sent to the cloud due to privacy or data sovereignty issues.
- **Reduced operational costs.** Training ML models in the cloud and then evaluating them locally on Windows devices can deliver significant savings in bandwidth costs, with only minimal data sent to the cloud—as might be needed for continual improvement of your ML models.
- **Ease of development.** With Windows ML built into the latest version of Windows 10, all you need is Visual Studio and a trained Open Neural Network Exchange (ONNX) model, which can be distributed along with the Windows app.

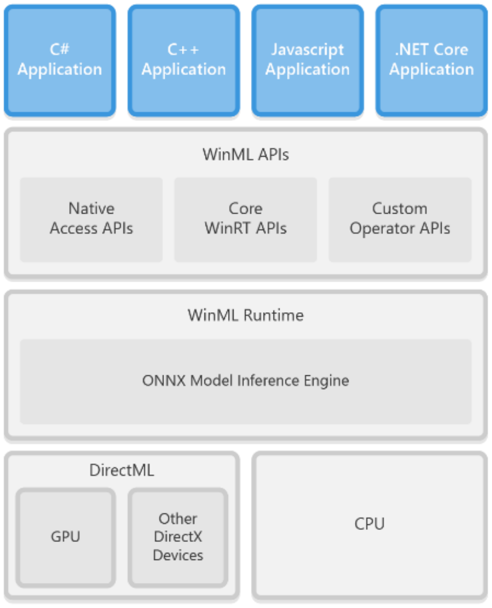
In the rest of this paper, we'll take a closer look at how Windows ML works—including training a machine learning model, integrating it into your Windows app, and running it on a local Windows device. We'll also briefly examine how Windows ML fits into the Microsoft IoT platform, and some resources that you can use to get started with Windows ML today.

Windows ML: A brief technical overview

With Windows ML, developers can use trained ML models in Windows apps that are written in C#, C++, or JavaScript. And with Windows ML now baked into the latest version of Windows 10, the process is simple, requiring just a few straightforward steps:

1. Get a trained ONNX model, or convert models trained in other ML frameworks into ONNX.
2. Add the ONNX model file to your app, or make it available in some other way on the target device.
3. Integrate the model into your application code, then build and deploy the application.

When your app runs, the Windows ML runtime (which contains the ONNX Model Inference Engine) evaluates the trained model locally, on the Windows 10 device, without requiring any connectivity. Windows ML handles the hardware abstraction, allowing developers to target a broad range of silicon—including CPUs, GPUs, and, in the future, AI-specific hardware from a range of hardware vendors—while delivering strong performance and reliability.



Getting an ONNX model

Windows ML evaluates models in the ONNX format, an open format for ML models that allows you to interchange them between various ML frameworks and tools. ONNX is being driven by Microsoft, Facebook, and Amazon Web Services, and supported by independent hardware vendors including NVIDIA, Intel, Qualcomm, and AMD.

Commented [SB1]: FYI to CELA: Taken from https://blogs.windows.com/buildingapps/2018/03/07/ai-platform-windows-developers/

To get an ONNX model for use with Windows ML, you can:

- Download a pretrained ONNX model from the ONNX Model Zoo.
- Train your own model with services like Custom Vision Service, Azure Machine Learning, or Visual Studio Tools for AI—and then export to ONNX format.
- Convert models trained in other ML frameworks into ONNX format with WinMLTools converters, or by following the ONNX tutorials on GitHub.

Training an ONNX model

ONNX models are an open format, so you can train them in any number of ways, using a broad range of tools. To help developers get started training a model, or at least to develop a better understanding of how it's done, we've created a few tutorials:

- Train a model with CNTK describes how to use Visual Studio Tools for AI to train a ML model for use with Windows ML. You'll train the model using the Microsoft Cognitive Toolkit and the MNIST dataset, then save the model in the ONNX format.
- Train a model with PyTorch describes how to use the PyTorch framework to train a model in the cloud, and then download it for use with Windows ML. You can also train the model locally.

Converting models trained in other ML frameworks

You can use WinMLTools—an extension of ONNXMLTools and TF2ONNX—to convert ML models created with different training frameworks into ONNX for use with Windows ML. WinMLTools currently supports conversion from Apple CoreML, Keras, scikit-learn, lightgbm, xgboost, libSVM, and TensorFlow (experimental). Again, we've created a handy tutorial to help you understand how it's done.

Integrating an ONNX model into your Windows app

After you have a trained ONNX model, you'll need to integrate it into your Windows app—and distribute the .onnx model files with your app. You can integrate the model into your app by using the Windows ML APIs, or by using mlgen, the Windows ML automatic code generator. Again, we've created a few tutorials to show how it works and help you get started:

- Integrate a model into your app with Windows ML shows how to use the Windows ML APIs to integrate an ONNX model into your Windows app. You'll learn how to load a model, bind the model to a device, bind inputs and outputs, and evaluate the model. Our online documentation provides detailed information on the Windows ML APIs.
- Automatic code generation with mlgen shows how to use mlgen to create wrapper classes that call the Windows ML API for you—allowing you to easily load, bind, and evaluate a model in your project.

We've invested heavily in delivering a great developer experience that makes all this as easy as possible. For developers creating Windows ML applications using Visual Studio 2017 or later, mlgen is provided as a Visual Studio extension. (In Windows 10 version 1903 and later, mlgen is no longer included in the Windows 10 SDK, so you must download and install the extension. There's one for Visual Studio 2017 and one for Visual Studio 2019.

Device compatibility and performance

As part of Windows 10, Windows ML works on any device that can run the Windows 10 October 2018 update—including IoT edge devices running Windows 10 IoT, 2-in-1s and desktop PCs, workstations, and servers.[1] The Windows ML runtime, including its ONNX model inference engine, is tuned for maximum efficiency across that same broad range of devices. We're delivering this performance by using instruction set optimizations for modern CPUs, hardware acceleration for any GPUs with a DirectX 12-certified driver, and a driver model that's designed for purpose-built AI processors in the future.

Our online documentation provides some tips for optimizing performance and memory utilization.

It's also worth noting that Windows ML supports high-performance load and execution of model chains by carefully optimizing its GPU path. Model chains are defined by two or more models that execute sequentially, where the outputs of one model become the inputs to the next model down the chain. Our article on executing multiple models in a chain gets into the details.

An extension of the Microsoft Azure ecosystem

Windows ML is designed to work with the Microsoft Azure AI platform, and, by extension, the broader Azure platform as a whole, so you'll have all the cloud services you need to deliver a comprehensive solution. This includes extensive resources for building and training your ML models, such as Azure Batch AI, Azure Machine Learning service, Data Science Virtual Machines, and Machine Learning Studio. Even our hardware is optimized for AI, including the latest in GPU technology and FPGA-accelerated AI models and networks, giving you the power and flexibility to train your ML models at the highest scale.

Windows ML works with the Microsoft IoT ecosystem, as well, including Azure IoT services. For example, you can use Azure IoT Edge to deploy your Windows ML apps and ONNX models to edge devices running Windows 10 IoT, as a means of supporting complex event processing, machine learning, image recognition, and other high-value AI functions to those devices. They're designed to work together, with minimal effort and complexity, making it easy to train your ML models in the cloud and evaluate them locally "on the edge"—even in offline situations or with intermittent connectivity.

Other useful resources

In the above technical overview, we've provided some links to tutorials that developers can use to get started with Windows ML. Here are some other useful resources:

- Full Windows ML documentation, which covers all of the above concepts and more
- Tools and sample apps

[1] Windows ML is only available as part of the Desktop installation option with Windows Server 2019.

- Release notes on the latest Windows ML features and fixes
- Frequently asked questions about Windows ML

Conclusion

With Windows ML, you can integrate trained machine learning models in your Windows apps, making them more powerful, engaging, and useful. Whether the scenario is video recognition, facial recognition, language processing, contextual understanding, robotic intelligence, or any number of others, you'll have all that you need to train your ML models in the cloud and run them on Windows, where they make the most sense. Extensive prebuilt functionality and interoperation with Visual Studio makes this easy for developers, so they can quickly and easily build and deploy ML-enabled apps.

Learn more about Windows Machine Learning or start developing with it today.