WINDOWS MACHINE LEARNING (WINML*)

LEGAL NOTICES AND DISCLAIMERS

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo, neon, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

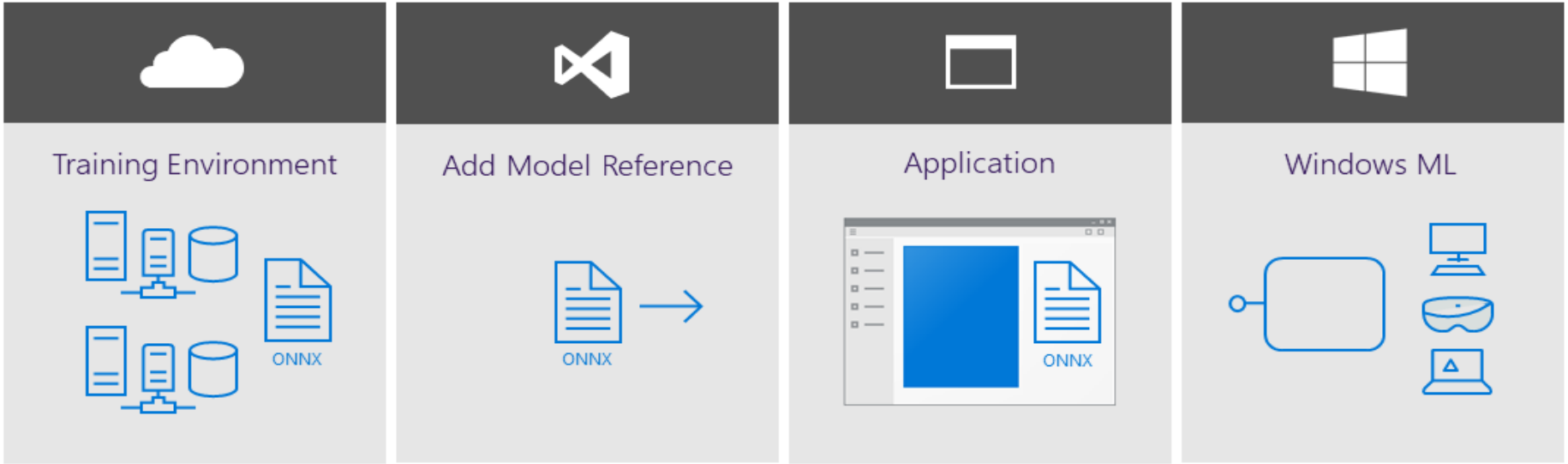
*Other names and brands may be claimed as the property of others.

OUTLINE: WHAT STUDENTS ARE EXPECTED TO LEARN

- Introduction to WinML*
- Steps in WinML* App Development
- Intel® Support for WinML*
- Summary

WINDOWS MACHINE LEARNING

- Run AI locally on Windows* 10 devices
- API provided with Windows* 10 for on-device evaluation of pre trained ML and DL models



Windows ML Architecture

WinML API

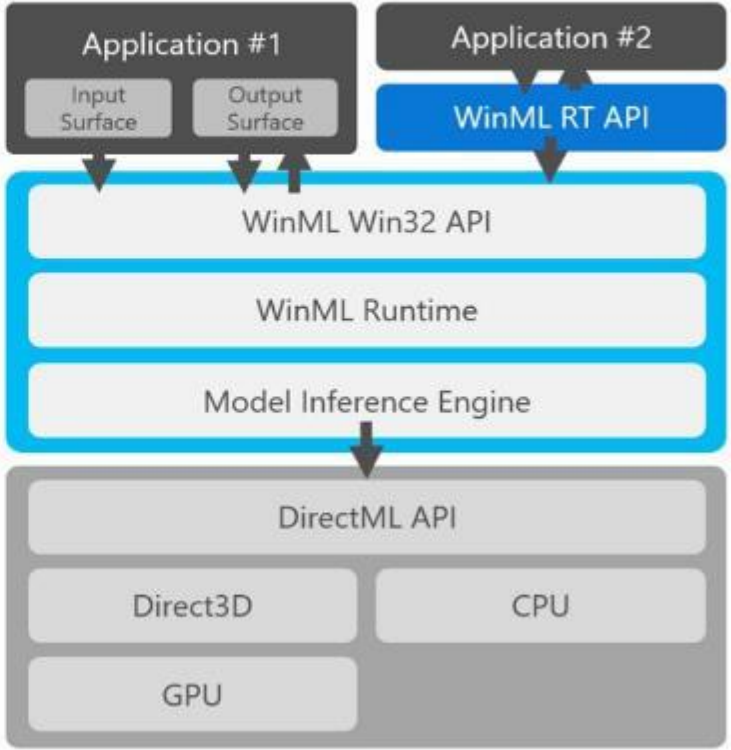
- Win32 & WinRT Layers
- Converts images to Tensor Resources
- Available on all Windows editions in 2018

Inference Engine

- Model & Device resource management
- Loads and compiles operator kernels
- Execute dataflow graph

Device Layer

- CPU instruction optimizations up to AVX-512
- DirectML generating DX12 Compute shaders
- ~80% of Windows 10 MAD are on DX12



WHY WINML*?

➢ Low latency, real-time results

Windows can perform AI tasks using the local processing capabilities of the PC

➢ Reduced operational costs

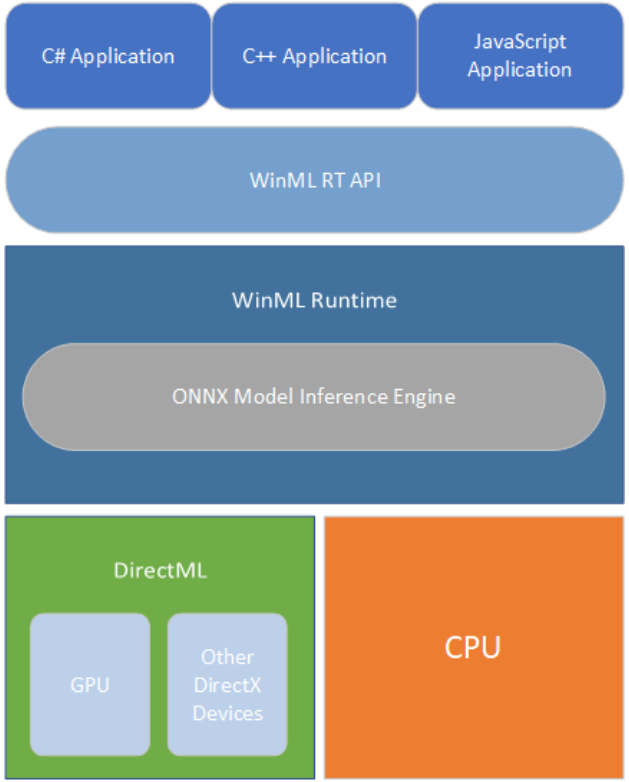
Developers can build affordable end-to-end AI solutions

➢ Flexibility

Developers can choose to perform AI tasks on device or in the cloud

➢ Hardware acceleration

WinML* will leverage device CPUs and GPUs



THE WINML* ECOSYSTEM

- WinML* is compatible with models in ONNX format
- Visual Studio* IDE performs automatic interface code generation that processes ONNX* file and creates wrapper classes allowing easy interaction with AI models.
- WinML* allows hardware to accelerate model evaluation on DirectX* 12 GPU and will handle the communication with the evaluation hardware (CPU or GPU) on its behalf

- Open Neural Network Exchange Format (ONNX*) is a common Intermediate Representation (IR) for models
- Move easily between models, state-of-the-art tools and choose the best combination
- Enables interoperability between different frameworks, e.g. Caffe2*, MS CNTK*, MXNet*, PyTorch, and Tensorflow*
- ONNX-compatible runtimes and libraries designed to maximize performance on some of the best hardware in the industry.

STEPS IN WINML* APP DEVELOPMENT

1. GET THE ONNX MODELS

- Download a pre-trained model from ONNX* Model zoo https://github.com/onnx/models
- Train your own model with services like Visual Studio* Tools for AI, and export to ONNX* format.
- Convert models trained in other ML frameworks into ONNX* format with WinML* Tools converters or ONNX* tutorials

2. INTEGRATE THE MODEL INTO YOUR APP

There are two methods to integrate the model to your app

I. Using WinML* API

II. Using MLGEN*

USING WINML* API

- **Load models :** Using methods in LearningModel
- **Create a session :** Binds the model to a device that runs and evaluates the model using LearningModelSession
- **Choose a device :** choose the device as you create a session LearningModelDeviceKind, By default its CPU
- **Bind inputs and outputs :** Use LearningModelBinding to bind values to a feature, referencing the ILearningModelFeatureDescriptor by its Name property.
- **Call evaluate :** To run the model

USING MLGEN

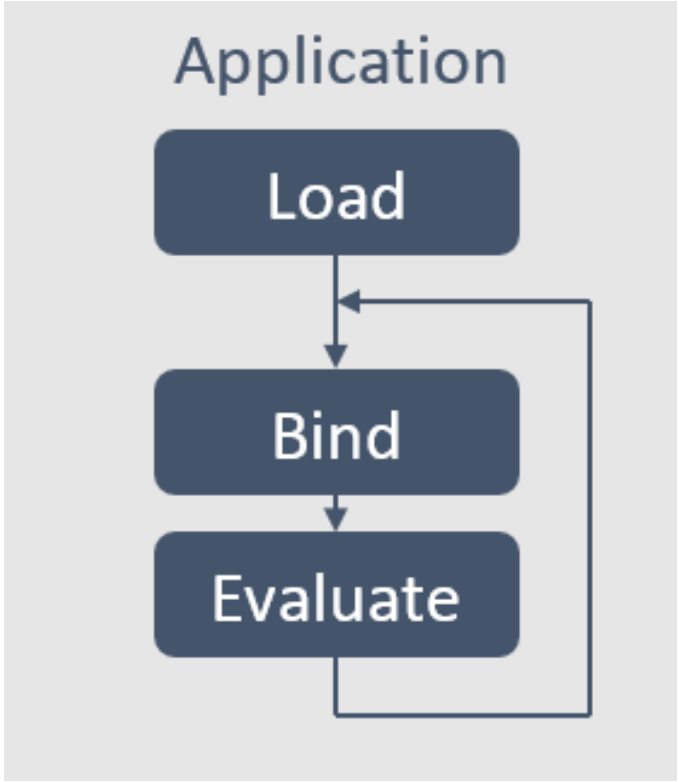
Using windows code generator MLGEN creates an interface with wrapper classes that call the WinML* API

Easily load, bind, and evaluate a model in your project

- For UWP developers, MLGEN* is integrated with Visual Studio* version 15.8.4
- Inside your Visual Studio* project, simply add your ONNX* file to your project's Assets, and Visual Studio* will generate Windows ML wrapper classes in a new interface file.
- For other workflows, or older versions of Visual Studio*, one can use the command line tool MLGEN.exe, that comes with Windows* SDK

WINML* AND ONNX IN UWP APPLICATION

- All we need to do, is to load the pre-trained model, bind data to the model, and evaluate the model against the data
- Linking of application with the model is seamless



WRAPPER CLASSES

An interface to easily interact with your machine learning model through Windows ML APIs.

- *Input class* – This class is to represent the input data which will be bound to the model.
- *Output class* – This class is to represent the output data which will be bound to the model.
- *Model class* – This class is to represent the model object to load and evaluate.

EXAMPLE OF INPUT AND OUTPUT CLASSES

```
public sealed class JacketModelInput
{
    public VideoFrame data { get; set; }
}

public sealed class JacketModelOutput
{
    public IList<string> classLabel { get; set; }
    public IDictionary<string, float> loss { get; set; }
    public JacketModelOutput()
    {
        this.classLabel = new List<string>();
        this.loss = new Dictionary<string, float>()
        {
            { "hardshell", float.NaN },
            { "insulated", float.NaN },
        };
    }
}
```

EXAMPLE OF MODEL CLASS

```
public sealed class JacketModel
{
    private LearningModelPreview learningModel;
    public static async Task<JacketModel> CreateModel(StorageFile
file)
    {
    . . . . .
    }
    public async Task<JacketModelOutput>
EvaluateAsync(JacketModelInput input)
    {
        . . . . .
    }
}
```

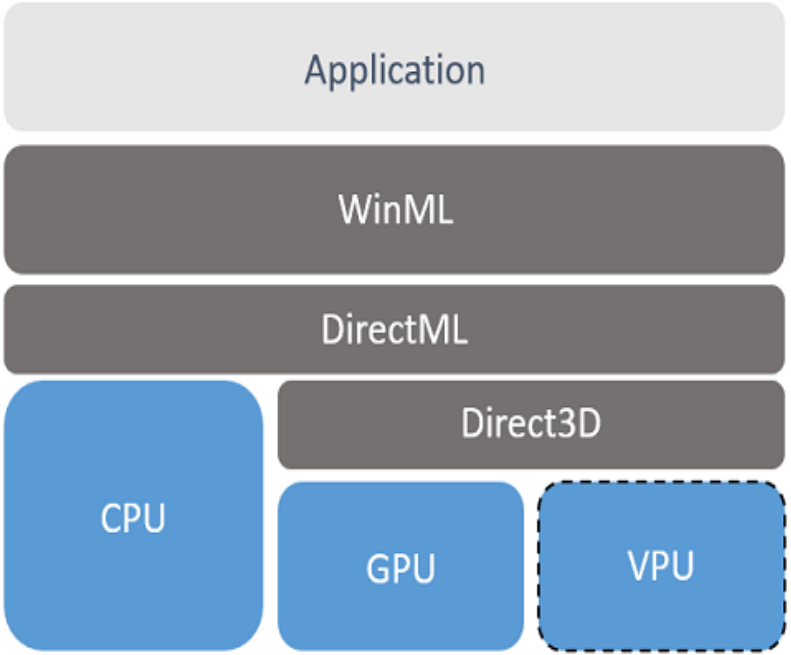
INTEL SUPPORT FOR WINML*

HARDWARE ACCELERATION BY WINML*

- WinML* stack exploits both the power of the Intel® Advanced Vector Extensions 512 (Intel® AVX-512) CPU instruction set and the power of the DirectX* 12 compute pipeline to accelerate execution on the integrated graphics
- Once the model is evaluated, the WinML* stack will query the driver for an optimized version of specific model. If available the same will be executed without much changes to the application.

WINML* STACK

- WinML* inference engine understands the model
- The Direct ML* abstraction layer selects the target hardware for evaluation and executes the model.
- In near future it will offer model-level operator acceleration, by employing a new Meta Command interface in the DirectX* 12 layer.



UPCOMING FEATURES

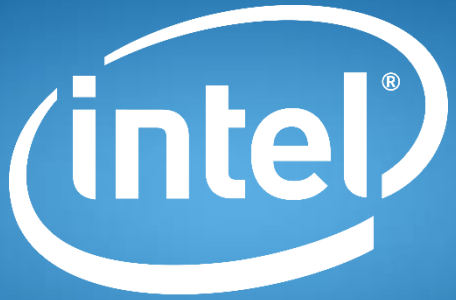
With the next available OS update, and a driver update

- Hand-optimized kernels that make the best use of Intel integrated graphics
- Additional optimizations via meta commands coming later this year
- Gain performance with no application changes

SUMMARY

SUMMARY

- **Windows Machine Learning (WinML)** refers to the Windows ecosystem of that perform model conversions and optimizations from a variety of AI frameworks and into ONNX; and from there into the C# and C++ application APIs.
- **WinML runtime layer** enables applications to perform inference AI computations on Windows 10* devices.
- The **general workflow in WinML applications** is to load the pre-trained model, bind data to the model, and evaluate the model against the data.
- WinML* stack exploits both the power of the **Intel® AVX-512 CPU** instruction set and the power of the **DirectX* 12** compute pipeline to accelerate execution on the integrated graphics.
- WinML is a generic inference solution for AI applications on the PC.



experience
what's inside™