What is AI at the Edge?

The edge means local (or near local) processing, as opposed to just anywhere in the cloud. This can be an actual local device like a smart refrigerator, or servers located as close as possible to the source (i.e. servers located in a nearby area instead of on the other side of the world).

The edge can be used where low latency is necessary, or where the network itself may not always be available. The use of it can come from a desire for real-time decision-making in certain applications.

Many applications with the cloud get data locally, send the data to the cloud, process it, and send it back. The edge means there's no need to send to the cloud; it can often be more secure (depending on edge device security) and have less impact on a network. Edge AI algorithms can still be trained in the cloud, but get run at the edge.

Why is AI at the Edge Important?

- Network communication can be expensive (bandwidth, power consumption, etc.) and sometimes impossible (think remote locations or during natural disasters).
- Real-time processing is necessary for applications, like self-driving cars, that can't handle latency in making important decisions.
- Edge applications could be using personal data (like health data) that could be sensitive if sent to the cloud.
- Optimization software, specially made for specific hardware, can help achieve great efficiency with edge AI models.

Applications of AI at the Edge

- There are nearly endless possibilities with the edge.
- IoT devices are a big use of the edge.
- Not every single app needs it - you can likely wait for a second while your voice app goes to ask the server a question, or such as when NASA engineers are processing the latest black hole data.

Historical Context

- Cloud computing has gotten a lot of the news in recent years, but the edge is also growing in importance.
- Per Intel®, IoT growth has gone from 2 billion devices in 2006 to a projected 200 billion by 2020.
- From the first network ATMs in the 1970s to the World Wide Web in the '90s, and on up to smart meters in the early 2000s, we've come a long way.
- From the constant use devices like phones to smart speakers, smart refrigerators, locks, warehouse applications and more, the IoT pool keeps expanding.

QUIZ QUESTION

Reasons for the development of the Edge?

1. The proliferation of devices.
2. Need for low-latency compute.
3. Need for disconnected devices.

Course Structure

- In this course, we'll largely focus on AI at the Edge using the Intel® Distribution of OpenVINO™ Toolkit.
- First, we'll start off with pre-trained models available in the OpenVINO™ Open Model Zoo. Even without needing huge amounts of your own data and costly training, you can deploy powerful models already created for many applications.
- Next, you'll learn about the Model Optimizer, which can take a model you trained in frameworks such as TensorFlow, PyTorch, Caffe and more, and create an Intermediate Representation (IR) optimized for inference with OpenVINO™ and Intel® hardware.
- Third, you'll learn about the Inference Engine, where the actual inference is performed on the IR model.
- Lastly, we'll hit some more topics on deploying at the edge, including things like handling input streams, processing model outputs, and the lightweight MQTT architecture used to publish data from your edge models to the web.

Why Are the Topics Distinct?

- Pre-trained models can be used to explore your options without the need to train a model. This pre-trained model can then be used with the Inference Engine, as it will already be in IR format. This can be integrated into your app and deployed at the edge.
- If you created your own model, or are leveraging a model not already in IR format (TensorFlow, PyTorch, Caffe, MXNet, etc), use the Model Optimizer first. This will then feed to the Inference Engine, which can be integrated into your app and deployed at the edge.
- While you'll be able to perform some amazingly efficient inference after feeding into the Inference Engine, you'll still want to appropriately handle the output for the edge application, and that's what we'll hit in the final lesson.

Relevant Tools and Prerequisites

Summary

- **Prerequisites**:
  - Understand some of the basics of computer vision and how AI models are created.
  - Basic Python or C++ experience. This first course is mainly in Python, although C++ can be used with the Intel® Distribution of OpenVINO™ Toolkit easily as well (and can be faster in a completed app!).
- We will not be training models in this course, as our focus is on optimization & deployment at the edge.
- Classroom workspaces will be available for exercises, so no set-up required if you plan to use them.

Local Set-up

- Make sure to make note of the hardware requirements for the Intel® Distribution of OpenVINO™ Toolkit if you want to work locally.
- If you do want to do the exercises on your local machine (or perhaps even on a set-up like a Raspberry Pi with an Intel® Neural Compute Stick 2), you can follow the instructions here for your operating system.

Intel® DevCloud - Edge

There is also the new Intel® DevCloud platform for testing out edge environments. This allows you to have access to a range of Intel® hardware such as CPUs, GPUs, FPGAs, Neural Compute Stick, and more. Later courses will get more into the hardware side of things, but this another option for working with an edge environment.

What You Will Build

In the project at the end of the course, you'll build and deploy a People Counter App at the Edge. In the project, you will:

- Convert a model to an Intermediate Representation (IR)
- Use the IR with the Inference Engine
- Process the output of the model to gather relevant statistics
- Send those statistics to a server, and
- Perform analysis on both the performance and further use cases of your model.