[D]: Good Morning everyone, and welcome to the second Workshop of the FAMU-FSU College of Engineering Intel Day. I hope you had some fun yesterday creating those AI generated videos. My name is Diego Abad, I’m a Junior Computer Engineer Undergrad, and I’m part of the oneAPI ambassadors program.

[D]: We’ll cover some general facts about oneAPI, the oneAPI Base Toolkit, the Intel AI Analytics Toolkit, what is OpenVINO, some general facts about OpenVINO, a Kahoot Game to test what you just learned, and finally, an OpenVINO activity.

**MOVE TO NEXT SLIDE**

[D]: Let’s start by asking you the following question: Where do you find Artificial Intelligence?

[D]: Great examples. Now, all of those applications of AI which is short for Artificial Intelligence and I’ll be using that word a lot during the workshop, will require big amounts of resources to compute their output. For example, chatGPT is a generative AI that needs a great amount of computational power, which translates to thousands and thousands of special equipment to maintain its performance. In computer engineering, we called those special equipment hardware accelerators. In general, a hardware accelerator is any kind of hardware designed to accelerate an special task. Some examples of it are here:

[D]: We have Graphics Processing Units or Graphics Cards or GPUs for short, Coprocessors, FPGAs, and other AI Accelerators.

[D]: Now, the issue I’m talking about is related to programming this device to process our AI. Normally, the programmer will have to write different files in different languages to both communicate and make multiple devices to process the heavy workload. However, Intel approached this by [make a pause]

**MOVE TO NEXT SLIDE**

[D]: Introducing oneAPI. oneAPI is an open standard for a unified application programming interface for communication across multiple hardware. This basically means that instead of creating multiple files and using multiple programming languages, you’ll only need to create one file in a single programming language. By doing this, programmers can

[D]: Re-use the code they wrote for specific hardware accelerators, avoid the need to update most of their code when newer version of those hardware come, and make the communication between devices relatively easy.

**MOVE TO NEXT SLIDE**

[D]: Over here we find some possible applications that oneAPI influences. We have Deepfake, Machine Learning, Image processing, Cloud computing, AI assistants, Machine Vision, and others. The one application that we will cover in today’s activity is Image classification.

**MOVE TO NEXT SLIDE**

[D]: Now, oneAPI is a concept that by its definition, will start influencing things at a really low level. What low level means here is things like developing functions and classes that eventually are put together to become libraries. These libraries by itself don’t do much, but they are the base blocks to create complex things like Machine Learning or any sort of computing intensive algorithm.

[D]: The middle level is oriented toward data scientist related topics. For the sake of our workshop, we can say that this is oriented towards people who create and train Machine Learning models. oneAPI is used in this level in a direct way by using special libraries that we will talk about later in the workshop.

[D]: Finally, we have the high level. This level represents an actual implementation of the previously mentioned Machine Learning models, and it targets people who are Application developers. At this level, we will see all the previously mentioned applications of AI and we will use OpenVINO to do so. A remainder here though, is that oneAPI will be implicitly applied to OpenVINO’s usage.

**MOVE TO NEXT SLIDE**

[D]: So [pause] oneAPI is the idea of having a code that allows for unified communication. The “software” where we find the tools to develop oneAPI applications are called Toolkits. Intel offers a variety of these toolkits and for today; **I’ll present you with 6**. Before doing that, I just want to point out that the toolkits you see with the color borders are the toolkits we’ll cover in more detail in my presentation.

[D]: First we have the Base Toolkit, which has the most general usage in the meaning that it is used to create low level applications like functions and classes.

[D]: The second Toolkit is the HPC Toolkit. This toolkit is oriented towards High Performance Computing applications. This means applications that will normally require to run in a cluster computer environment, which is in simple terms is many computers running a program.

[D]: The third Toolkit is the AI Analytics Toolkit. This one is specifically oriented towards Machine Learning developing at the middle level we discussed early.

[D]: The fourth is the IoT Toolkit. This is used for helping in the development of Internet of Things or IoT devices applications and design. For the ones who don’t know what this means, IoT are things like smart lights, or smart watches, or any device that uses internet inside of it.

[D]: The fifth is the Rendering Toolkit which is used to accelerate digital media content creation.

[D]: And finally, the System Bring-Up Toolkit which is used in Industrial systems debugging and analysis.

[D]: Now that we introduced the oneAPI toolkits, we’ll focus now on the Base Toolkit.

**MOVE TO NEXT SLIDE**

[D]: The first thing we should mention is that this toolkit must be installed first before we could add any other. The reason is because this toolkit contains the programming language that oneAPI offers, which is the Data Parallel C++, or DPC++ for short. This toolkit also includes the compiler for this programming language, a list of libraries made from DPC++ and other Intel optimizations, an Intel Distribution for Python, which we will cover later on, the Intel DPC++ Compatibility Tool, the Intel VTune profiler and Intel Advisor.

[D]: the Base Tool Kit gives you the tools you need to build low level functions and classes that can be used in different applications. For example, I created a Linear Regression algorithm using the Base Tool Kit.

[D]: Now that we covered what the Base Toolkit contained, let’s see in more detailed some of its components.

**MOVE TO NEXT SLIDE**

[D]: The first thing we’ll cover is the DPC++ language. As its name may give away, this programming language is based on C++, and it combines it with the SYCL kernel coding style. So, imagine a computer like your laptop or your desktop at home. Those systems contain a processor, and most likely, a graphics card. We call your processor the “host” where all the non-intensive programs run like background applications, and let’s call your graphics card the “device”, where all our intensive programs run, which in this case are the pictures and graphical information that your computer displays.

[D]: Now, in hardware communication any programs or functions that run in our devices is called a “kernel.” There are different ways **to implement a kernel style coding**, and what SYCL allows is the use of regular C++ and special commands to achieve communication between devices on the same file, and that’s why DPC++ can use both C++ and SYCL. Besides the previous, DPC++ also provides internal parallelization and other hardware optimizations techniques which allows faster workload processing.

[D]: The toolkit also includes the DPC++ compiler, that compiles ISO C++, SYCL, and DPC++ code. ISO means International Organization for Standardization

**MOVE TO NEXT SLIDE**

[D]: Next, let’s look at the libraries that the Base Toolkit includes. There are seven main libraries, and we will quickly mention their use.

[D]: The first one is the Data Parallel Library which contains the DPC++ language. Then we have, the oneDNN and oneCLL library, which we will talk about more in detail later on. After that we have the oneDAL library which is used for data analytics, and the oneTBB or Thread Building Block Library, which we will not cover in detail but basically this is the library that provides the parallelization implementation of oneAPI.

[D]: Finally, we have the oneVPL or the Video Processing Library, which as its names entails, is used in video development, and the oneMKL library, which we will also cover later on the workshop.

**MOVE TO NEXT SLIDE**

[D]: So now, let’s look into the DPC++ Compatibility Tool. Before we start talking about what this tool does, we need be aware of what CUDA is. Imagine it as something similar to what oneAPI does, in the meaning it is an API that also allows for device task communication. The difference here is that the CUDA API is only supported in NVIDIA devices like NVIDIA Graphics Cards and in weird cases, NVIDIA FPGAs.

[D]: Nevertheless, CUDA is a really popular API than has been around for more than a decade, so several researchers and companies across the world use it on a daily basis. To help transition CUDA developers into oneAPI, the Base Toolkit has the DPC++ compatibility tool. This tool uses a CUDA code as its input, and it translates that into a DPC++ code.

[D]: This translated code will be almost runnable by the compiler. Because of differences in formatting and syntax, the programmer will have to lightly modify the translated code so that the DPC++ Compiler runs the code.

[D]: For large projects, it is recommended to have good documentation of each section of your code, since it can be tedious to look for syntax or formatting errors across 20 thousand or more lines of code inside your translated file.

[D]: Additionally, it is recommended to also review the oneAPI kernel programming style since it has some key differences compared to CUDA.

**MOVE TO NEXT SLIDE**

[D]: Now that we know what the Base Tool Kit contains and does, we can move on to our next questions. What if only want to develop AI applications. For this, we have the AI Analytics Toolkit.

[D]: Before we go into explaining what this Toolkit does, we should mention that the concept AI itself has several components. This is probably a reminder of what you saw yesterday before the first workshop. One of those components is Machine Learning. Machine Learning refers to any algorithm that uses data to imitate the way humans learn, which means that it will attempt to solve something until it gets it right. Machine learning contains different kinds of algorithms that range from the simplest Linear Regression to your more advanced K-Nearest Neighbors.

[D]: Normally, Machine Learning algorithms require the use of several libraries that take care of different aspects of its development, like multi-dimensional array creation and graphical display of results. Since it gets tedious to manage different libraries at once, several entities created different “Frameworks.” A framework uses several libraries and create their own functions that simplify complex processes like Machine Learning.

[D]: Some popular frameworks are Pytorch, Caffe, and TensorFlow.

[D]: Now, **while** frameworks are important, it still needs to be installed somewhere. The place where we store frameworks is called environments. There are several ways to create environments depending on your Operating System, but one of the most popular software to do so, and the one most people use is called Anaconda.

[D]: In any case, we need to know this information so that we can understand what the AI Analytics Toolkit does.

**MOVE TO NEXT SLIDE**

[D]: In simple words, it helps us in the development of Machine Learning models, as well as improving the performance by accelerating the training of the models.

[D]: It greatly impacts some aspects of Machine Learning like Data acquisition and pre-processing. This refers to the data you input inside your Machine Learning model, so it gets trained by one part of that data, and it validates itself with the other part of that data. A common library used for data formatting is Pandas, which is used to create Data Frames. For the sake of simplicity, data frames are just data structures that store information, which in this case is the input for our Machine Learning Model.

[D]: The AI Analytics Toolkit provides their own version of Pandas to work specifically with the oneAPI environment.

[D]: Another aspect of Machine Learning that the AI Analytics toolkit helps improve is the processing and training part. It is done by providing their own version of the NumPy and SciPy python libraries, with the first being used when multi-dimensional array manipulation, and the second one used for integration, interpolation, eigenvalue problems, algebraic equations, differential equations and statistics.

[D]: Now, let’s look at some of the components that the AI Analytics Toolkit provides.

**MOVE TO NEXT SLIDE**

[D]: It offers optimization for the previously mentioned frameworks: TensorFlow and PyTorch respectively. This will be in the form of drop-in replacements of commonly used functions, which translates to just having to add the Intel libraries version in their original environments, which in turn will allow the use of those optimized functions without the need to modifying your code.

[D]: It also offers Model Zoo for Intel Architectures and the Intel Neural Compressor. Model Zoo is a platform where users can find a variety of pre-trained models, and we’ll cover this in more detail later in the presentation.

[D]: The Intel Neural Compressor on the other hand compresses the model size and increases the speed of the deep learning inference on CPUs and GPUs. This is accomplished by using some techniques like quantization and pruning, which reduces the precision of some parts of the Machine Learning model with minimum loss in accuracy.

[D]: Then we have the Intel Extension for Scikit-learn, which improves upon the simple Scikit-learn by being optimized for Intel CPU and GPU architecture. It can improve the performance up to a 100x more than using the regular scikit-learn. This means whatever application you created using scikit-learn in the past, will benefit from using this extension if operated in Intel devices.

[D]: Coming next, we have the Intel Optimization of XGBoost, which by itself basically is a gradient boosting machine learning library, or in simple words a library that boost the training of some Machine Learning algorithms, but now optimized to run faster and efficiently in Intel hardware.

[D]: And finally, we got the Intel Distribution of Modin. So remember how we talked about how Pandas help us in Machine Learning because it allows us to use Data Frames, well regular Pandas is optimized only for single thread processing. Because of this, Modin was born. It’s basically built on top of Pandas to allow faster processing by allowing multi-core and cluster processing. This concept will be elaborated more in the HPC workshop so be ready for that. In any case, this Intel Distribution of Modin, as we could predict from before, will be faster in Intel hardware.

**MOVE TO NEXT SLIDE**

[D]: So, let’s get into general Machine Learning development. I’ll just go over the libraries that make all the previous possible.

**MOVE TO NEXT SLIDE**

[D]: oneMKL is the Math Kernel Library, which is the library that most of the other libraries and applications of oneAPI are based on. This library provides the fundamental blocks for complex math applications used in Machine Learning and other complex applications.

[D]: Now, oneDAL is the Data Analysis Library we mentioned before. oneDAL has the particularity of being used to both analysis and display of data of any kinds of high-level process. Both the oneMKL and oneDAL library are used to create the Python libraries we talked about before and the ones that you see on the screen.

**MOVE TO NEXT SLIDE**

[D]: Now, as we mentioned before, Machine Learning has several subsets. One of those subsets is Deep Learning. Deep learning is the most famous type of Machine Learning, mainly because of the wide range of applications it reaches. A deep learning model will contain several neurons, which as the name indicates, work similarly as the ones you have in your brain. They get activated by specific stimulus, which in this case is our input.

[D]: So, neurons are arranged in what we call layers, which is just a vertical arrangement of neurons. Almost all neural networks will be formed as this. An input layer where the data is first introduced to the network, an output layer that provides the end results of the neural network, and a middle layer that will be in charge of activating or deactivating neurons in such a way that it gives the end result. That combination of input, output, and middle layer is what we call a neural network.

[D]: Long story short, neural networks have different ways to both make your predictions more accurate, and to make it work for a specific kind of job. For example, Convolutional Neural Networks are used for image classification, and Recurrent Neural Networks are used for data sequence predictions.

[D]: However, as we mentioned before, we don’t want to re-invent the wheel so that’s why the following libraries exist.

**MOVE TO NEXT SLIDE**

[D]: oneDNN is a Deep Neural Network library that is used for Deep Neural Network development. However, what I like to emphasize here is that it can still be used for low level development in the meaning you can in theory create a simple Neural Network using this library. But normally, as we mentioned before, it will be used implicitly inside the optimized Python functions for Deep Learning frameworks.

**MOVE TO NEXT SLIDE**

[D]: And finally, we get into oneCCL, or the Collective Communications Library. This provides the protocols necessary for device communication in both the “host”-“device” approach we mentioned before, as well as for larger scale operation, like computer clusters, which goes more into HPC topics.

**MOVE TO NEXT SLIDE**

[D]: After looking at what oneAPI is, and what does the Base Toolkit and the AI Analytics Toolkit provides, we’ll move into applying our already trained model

**MOVE TO NEXT SLIDE**

[D]: OpenVINO is a software that aims to accelerate Deep Learning Inferences. OpenVINO also offers a streamlined development and ease of use from various internal components such as the model optimizer and the inference engine.

[D]: The most important feature is the ability to develop your deep learning inference applications once and deployed it across all the supported hardware platforms without having to rewrite your code base.

[D]: The main components in OpenVINO are the model optimizer and the inference engine.

[D]: Before we go into the two components, let’s mention the different utilities that helps OpenVINO to function.

**MOVE TO NEXT SLIDE**

[D]: We have the Deployment Manager that enables an optimal and minimize the runtime package for your hardware target device, The post training optimization toolkit which enables you to quantize your model from a Floating Point 32 to an Integer 8 data type.

[D]: The Benchmark App which gives you a performance estimate of your model on the inference engine across different devices. The DL Workbench which offers various performance profiling utilities, and the DL Streamer which enables the creation and deployment of optimized streaming data analytics pipelines.

[D]: So now let’s talk about the Model Optimizer.

**MOVE TO NEXT SLIDE**

[D]: Model Optimizer is a Python-based Command Line Interface that converts your pre-trained model into an intermediate representation of that model.

[D]: The Model Optimizer will apply hardware-agnostic optimizations like layer fusion and remove layers used in training but not inferencing, such as drop-out layers.

[D]: It will also perform quantization depending on which type is needed to support the target hardware platforms. Simply put, the Model Optimizer will analyze the model for opportunities for optimization and apply the necessary utilities we discussed earlier. Ultimately, it will generate an Intermediate Representation(IR) file.

[D]: This intermediate representation is made of two main things: a .xml file containing the model's topology and a .bin file containing the model's "data," like weights and biases.

[D]: Now the other main component of OpenVINO is called the Inference Engine.

**MOVE TO NEXT SLIDE**

[D]: The Inference Engine is a simplified and unified API for Optimized inference across all Intel Hardware platforms. So the inference application will sit at the top of the Inference Engine API, and it will export the available hardware platforms to you through the corresponding hardware plugins.

[D]: This plugin ranges from the DNNL plugin for CPUs to FPGA plugins.

[D]: The following demonstrates the workflow that OpenVINO uses to deploy your model and use it for real world applications.

[D]: The activity we’re going to do will go in more detail regarding how you use the Inference Engine to deploy your model.

**MOVE TO NEXT SLIDE**

[D]: Last but not least important, we have the Open Model Zoo. As I mentioned previously, the Open Model Zoo is large recollection of both Intel and Public models. It supports a C++ and Python languages, as well as the OpenCV Graph API. It consists of more than 200 Neural Networks, with applications like Object detection, speech recognition, image segmentation, and others.