

Taller de Intel® oneAPI para IA

Escuela Invierno CAPAP-H 2024

CGS

27 de enero de 2024

- “Intel® oneAPI AI Tools”, <https://www.intel.com/content/www/us/en/developer/tools/oneapi/ai-analytics-toolkit.html>
- “Machine Learning Using oneAPI”, <https://www.intel.com/content/www/us/en/developer/tools/oneapi/training/machine-learning-using-oneapi.html>
- “Intel® Extension for PyTorch”,
<https://github.com/intel/intel-extension-for-pytorch>

Outline

1 Introducción

2 Python en oneAPI

3 Machine Learning

4 Deep Learning

5 Otros



Introducing

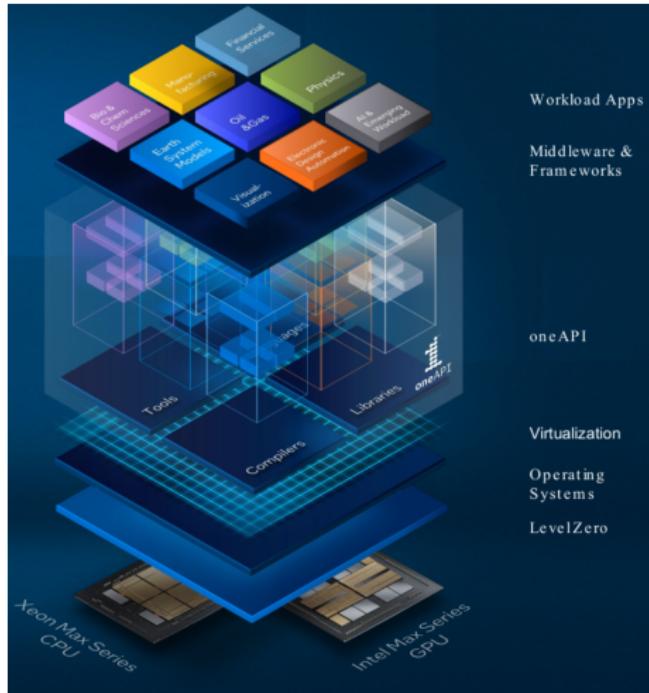
oneAPI

No transistor left behind

Introducción

Introducción

- Modelo de programación unificado: diversas arquitecturas
- Lenguaje y bibliotecas optimizados
- Rendimiento equivalente lenguaje nativo de alto nivel
- Basado en estándares de la industria y especificaciones abiertas
- Compatible con los modelos de programación HPC existentes

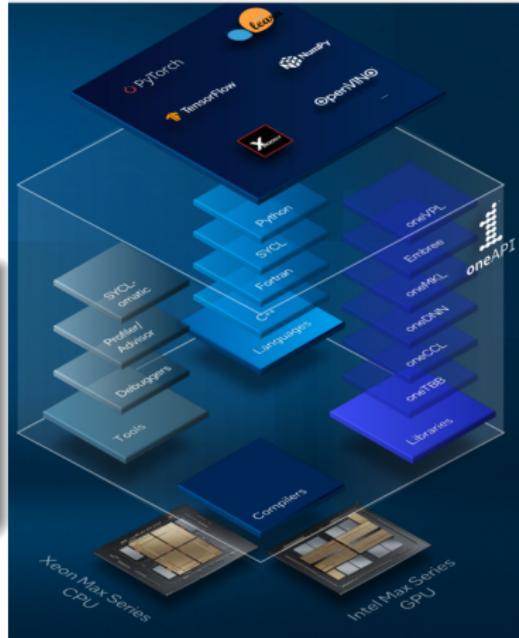


Introducción

- Un lenguaje basado en estándares: C++ y SYCL
- Potentes API para acelerar funciones de dominio específico

Soluciones a proveedor único

- Estándar abierto para promover el apoyo de la comunidad y la industria
- Permite la reutilización de código en diferentes arquitecturas y proveedores



- Un conjunto completo de herramientas de desarrollo testeadas desde CPU a XPU
- Disponible para su instalación en [Intel® oneAPI Toolkits](#)

For most developers

[Intel® oneAPI Base Toolkit](#)

Use Case: Develop performant, data-centric applications across Intel® CPUs, GPUs, and FPGAs with this foundational toolset.

For deep learning inference developers

[Intel® Distribution of OpenVINO™ toolkit \(Powered by oneAPI\)](#)

Use Case: Deploy high-performance inference applications from edge to cloud.

For HPC developers

[Intel® HPC Toolkit](#)

Use Case: Build, analyze, and scale applications across shared- and distributed-memory computing systems.

For visual creators, scientists, and engineers

[Intel® Rendering Toolkit](#)

Use Case: Create high-fidelity, photorealistic experiences that push the boundaries of visualization.

For data scientists and AI developers

[AI Tools](#)

Use Case: Accelerate end-to-end data science and machine learning pipelines using Python* tools and frameworks.

For system engineers

[Intel® System Bring-up Toolkit](#)

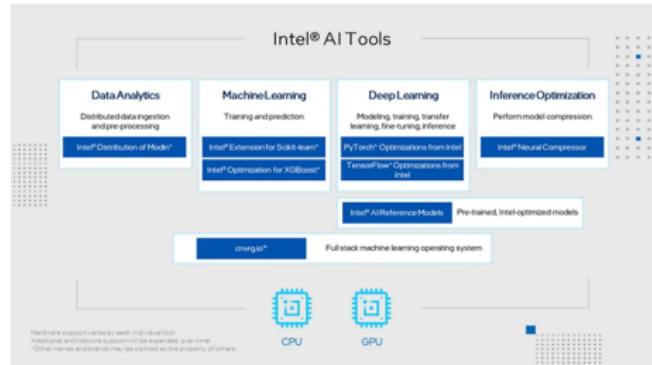
Use Case: Strengthen system reliability with hardware and software insight, and optimize power and performance.

AI Tools Toolkit

- Acelera el flujo de trabajo desde un extremo al otro para aplicaciones IA y analítica de datos mediante librerías optimizadas para arquitecturas Intel
- ¿Para quién es interesante?
 - Desde científicos de datos, investigadores en IA, desarrolladores de aplicaciones IA y ML...
- Beneficios
 - Rendimiento en aplicaciones de DL desde el entrenamiento e inferencia: frameworks optimizados para archs Intel
 - Aceleraciones en aplicaciones de analítica de datos y ML intensivas en cómputo basadas en paquetes **python**

AI Tools Toolkit

- Acelera el flujo de trabajo en desarrollos de IA y analítica de datos desde un extremo al otro mediante librerías optimizadas para arquitecturas Intel
- ¿Para quién es interesante?
 - Desde científicos de datos, investigadores en IA, desarrolladores de aplicaciones IA y ML...
- Beneficios: analítica de datos y ML intensivas en cómputo basadas en paquetes **python**



- Intel Developer Cloud disponible [en la URL](#)



- Varias configuraciones que se adaptan a diversas cargas de trabajo
 - Desde capacitación en IA y inferencia
 - ... creación de prototipos y evaluación del último HW utilizar el entorno que mejor se adapte a sus necesidades comerciales
 - ... hasta aplicaciones para FPGA
- Aprenda con tutoriales prácticos
 - Experimente con ejemplos de código del mundo real
 - Evalúe el rendimiento y la aceleración con múltiples configuraciones de hardware.
 - Cree aplicaciones heterogéneas

Hardware disponible

- Se puede testear y evaluar una variedad de máquinas virtuales
 - Sistemas *bare metal*
 - Dispositivos en el Edge
 - Plataformas para entrenamiento de IA
- Entornos para desarrollo
 - Contenedores
 - JupyterLabs
 - Conexión directa por SSH



Instrucciones de acceso

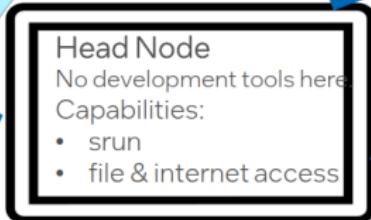
- La documentación y actualizaciones disponible en <https://tinyurl.com/ReadmeIDC> o en el Readme.md

Picture it this way

nodes in queues
pvc-shared
and
pvc
are identically configured
same CPUs, same four PC cards
(single tile PVCs – but four of them!)



ssh

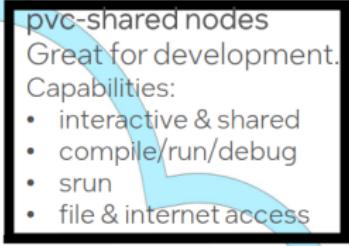


Intel
Developer
Cloud
(IDC)

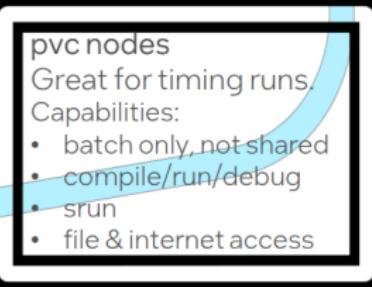
srun



srun



srun



Registro

- Para disponer de cuenta en Intel® Developer Cloud se puede acceder seguir el enlace <http://cloud.intel.com>
- Siguiendo los pasos del proceso de registro
 - ① Selección de **usuario Standard**
 - ② Creación de cuenta
 - ③ Introducción de datos personales
 - ④ Verificación de correo mediante envío de email con un **código**
 - ⑤ Aceptación de **términos y condiciones de uso** del Intel® Developer Cloud

Acceso

- Para acceder a la cuenta en [Intel® Developer Cloud](#)
 - Clicar en **Already a Member? Sign In** e introducir las credenciales creadas anteriormente

Training

- JupyterLabs: en el menú **Training and Workshops**
 - Clicar en **LaunchJupyterLab**

Instancias en IDC

- Core compute
 - Basada en procesador Xeon 4th gen
 - VMs con 8, 16, 32 cores
 - Bare Metal 112 cores, 256GB y 2TB disco
- Intel Max GPU
 - 4xGPUs 1100 + 2xsockets Xeon 4th gen
- Gaudi Deep Learning Server
 - 8x Gaudi HL + Xeon Platinum 3gen

The screenshot shows the Intel Developer Cloud Hardware Catalog interface. On the left is a sidebar with icons for Home, Catalog, Compute, GPU, Storage, Network, and Security. The main area has a title 'Hardware Catalog' and a sub-section 'Available platforms'. It lists three categories: 'Core compute', 'GPU', and 'AI'. Under 'Core compute', there is a section for '4th Generation Intel® Xeon® Scalable processors' with a 'Select' button and a radio button for 'Released'. Under 'GPU', there is a section for 'Intel® Max Series GPU' with a 'Select' button and a radio button for 'Released'. Under 'AI', there is a section for 'Gaudi™ Deep Learning Server' with a 'Select' button and a radio button for 'Released'. On the right side of the catalog, there is a vertical sidebar with the Intel logo and the text 'Developer Cloud'.



Introducing

oneAPI

No transistor left behind

Python en oneAPI

Distribución de Python

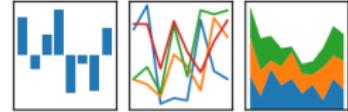
- Distribución optimizada en [plataformas Intel](#)
- Rendimiento cercano al código nativo optimizado para computación científica y HPC
 - Soporte hardware mediante librerías como oneMKL o oneDNN
 - Soporte de las últimas optimizaciones para CPU y GPU
- También disponible en el Toolkit [Intel® AI Tools](#)

Pandas

- Librería para computación con datos estructurados
 - Tipos mixtos en un tabla permitidos
- Se pueden nombrar columnas y filas de datos
- Agregación avanzada de datos y funciones estadísticas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Estructuras de datos

- Vectores de 1D: **Series**
- Arrays 2D...: **DataFrame**

pandas_series.py

```
import pandas as pd
step_data = [3620, 7891, 9761,
             3907, 4338, 5373]
step_counts = pd.Series(step_data, name='steps')
print(step_counts)
```

Terminal #1

```
user@host:~/ $ python panda_creation.py
0    3620
1    7891
2    9761
3    3907
4    4338
5    5373
Name: steps, dtype: int64
```

Hands-on

- ① Conéctate al Intel DevCloud
- ② Vamos a abrir el segundo cuaderno de Jupyter relacionado con el [uso Pandas](#)
- ③ Te animamos a que pongas en práctica el contenido con los [ejercicios](#) y puedes consultar también las [soluciones](#)

- Ventajas

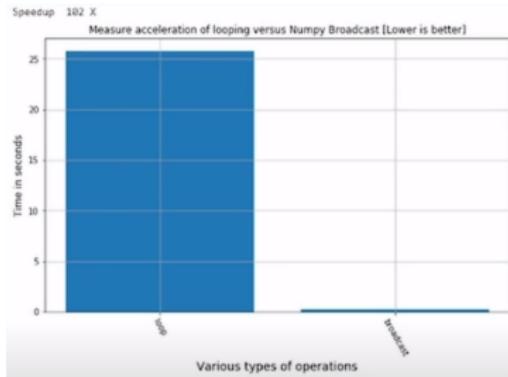
- Es rápido para desarrollar ideas
- Se puede codificar casi todo lo imaginable
- Desarrolla un proyecto rápidamente... se pueden encontrar ejemplos en cualquier sitio
- Fácil: sin tipado (dinámico) lo que hace la programación más sencilla
- Rápido: gran cantidad de librerías disponibles y fáciles de instalar
- Aplicaciones de IA: sencillo de lograr portabilidad de los modelos desarrollados entre arquitecturas

Python

- ... pero es **lento**
 - Tareas de bajo nivel repetitivas
 - Bucles largos
 - Bucles anidados
- Pero hay formas de mitigar los problemas
 - Librerías optimizadas para las arquitecturas Intel
 - NumPy es un ejemplo
 - Otras librerías están también optimizadas

Vectorización

- La explotación de la vectorización **no es una teoría**
 - Grandes aceleraciones son posibles
 - Altamente recomendable utilizar librerías optimizadas en Intel oneAPI como **NumPy**, **SciPy** y otras
 - Explotar la bondad Numpy (paralelismo inherente) mediante el uso de las librerías optimizadas en oneAPI
- Ej: 100x de aceleración empleando Numpy *broadcasting* respecto al código descrito como bucle



Numpy (vectorización)

- Disponible como paquete de [python](#)
- Versión optimizada incluida en oneAPI
 - Última versión disponible en el [Intel® AI Tools](#)

Como funciona

- Reemplazando los bucles explícitos asociados a las operaciones entre vectores/arrays: SIMD
- En general operaciones entre arrays pueden lograrse entre 1 o 2 órdenes de magnitud de velocidad que las equivalentes *python puras*

Hands-on

- ① Conéctate al Intel Developer Cloud
- ② Vamos a abrir el tercer cuaderno de Jupyter relacionado con el [uso Numpy](#)



Introducing

oneAPI

No transistor left behind

Machine Learning

- Machine Learning o **Aprendizaje automático**: ciencia que hace que los computadores “aprendan”
 - A partir de unos datos de entrada
 - El computador extra patrones de los diferentes tipos de datos
 - Aprende las reglas del problema

Python scikit-learn

- Scikit-learn es una librería de aprendizaje automático para el lenguaje de programación Python
 - Integración con otras bibliotecas de Python para el procesamiento de datos eficiente NumPy y Pandas o SciPy
 - Numpy: para el procesamiento y operaciones eficiente sobre forma vectorial o matricial
 - Pandas: para manipulación de datos en forma de tablas o series
 - SciPy: algoritmos de optimización, integración, interpolación...

- Desarrollada originalmente como parte del proyecto Google Summer of Code en el año 2007
 - Mantenido y desarrollado por una comunidad de desarrolladores
- Amplia variedad de herramientas para la minería y el análisis de datos
 - Algoritmos para clasificación, regresión, clustering y reducción de dimensionalidad

Características

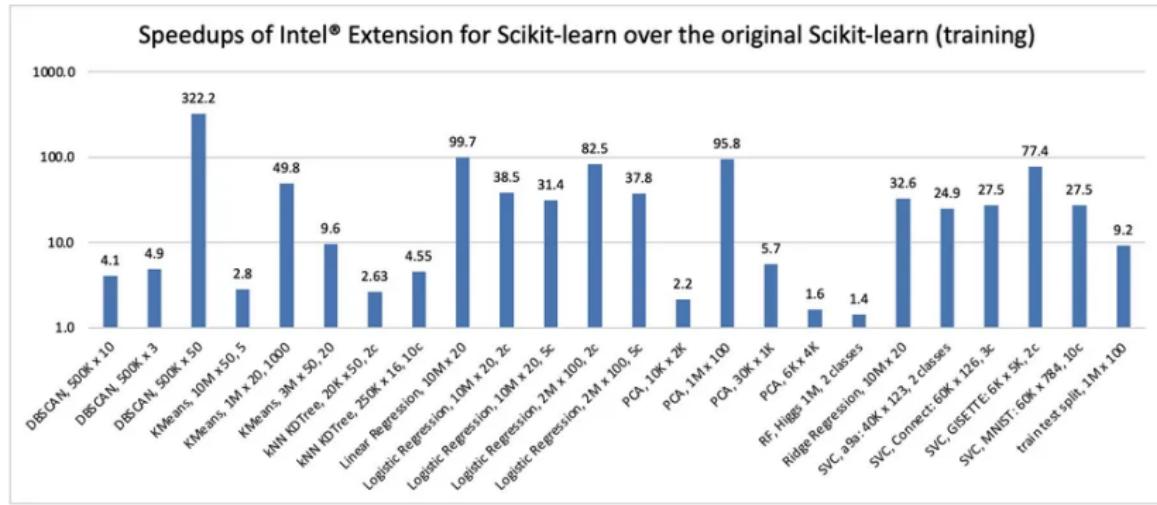
- Implementación de algoritmos de ML: árboles de decisión, SVM, regresión logística, k-means y otros
- Integración con otras bibliotecas como NumPy, Pandas y SciPy
- Herramientas para la extracción de características y reducción de dimensionalidad
- Soporte de aprendizaje supervisado y no supervisado

Tipo	Algoritmos
Supervisado	Regresión lineal, Regresión logística, Árboles de decisión, Bosques aleatorios, SVM, KNN, Gradient Boosting, Redes neuronales, Naive Bayes
No Supervisado	Clustering (K-means, DBSCAN), PCA, ICA, LDA

- Otros:
 - Basados en árboles de decisión: Árboles de decisión, Gradient Boosting...
 - Basados en redes neuronales: Redes neuronales convolucionales (CNN), Redes neuronales recurrentes (RNN), Long short-term memory (LSTM)

Extensión de Intel para Scikit-learn

- Intel ha creado una librería equivalente a Scikit-learn con mejor rendimiento que contiene versiones parcheadas de [32 algoritmos scikit-learn](#)



Introducción

- La **Intel Extension para Scikit-learn** proporciona una forma de acelerar el código scikit-learn
 - En un código existente, importaremos **sklearnex**: nombre de la biblioteca de python para el Extensión Intel
 - Usando “**patching**”: los algoritmos de scikit-learn son intercambiadas por la versión optimizada de Intel
 - Varias maneras:
 - Sin editar el código: invocando *python* con un flag adicional
 - Modificando el código: importando e invocando a las funciones de `sklearnex`
 - Añadiendo parches: sin modificar el resto del código

Alternativas

- Línea de comando

```
python -m sklearnex my_application.py
```

- Dentro del script o el cuaderno de Jupyter, **parcheando** el código

```
from sklearnex import patch_sklearn  
patch_sklearn()
```

- Inhabilitando el parche

```
from sklearnex import unpatch_sklearn  
unpatch_sklearn()
```

Hands-on

- ① Conéctate al Intel DevCloud
- ② Vamos a abrir el cuarto cuaderno de Jupyter relacionado con el ML con la extensión Intel(R) Extension for Scikit-learn



Introducing

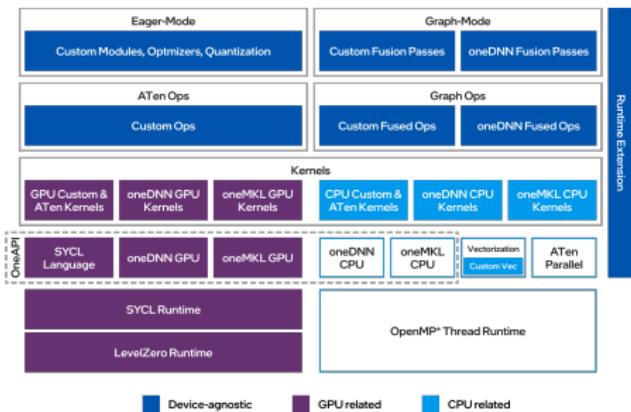
oneAPI

No transistor left behind

Deep Learning

Intel Extension para PyTorch (IPEX)

- Amplía PyTorch con optimizaciones para incremento de rendimiento en la CPU Intel® y GPU
- Código abierto en GitHub
 - CPU: Código y documentación
 - GPU: Código y documentación
- Operadores y kernels optimizados



Optimizaciones soportadas en IPLEX

3-Pillar Framework Optimization Techniques



Op

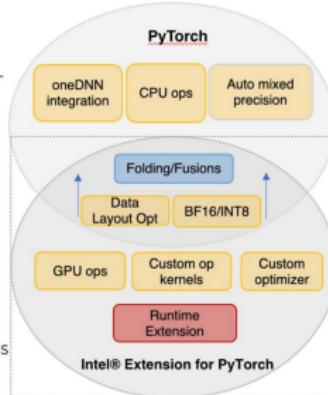
- Vectorization and Multi-threading
- Low-precision BF16/INT8 compute
- Ease-of-use BF16 compute with Auto-Mixed-Precision (AMP)
- Data layout optimization for better cache locality

Graph

- Constant folding to reduce compute
- Op fusion for better cache locality

Runtime

- Thread affinization and multi-streams
- Memory buffer pooling
- GPU runtime
- Launcher



PyTorch Upstream

General Intel new feature enabling and performance optimizations

Intel Extension for PyTorch

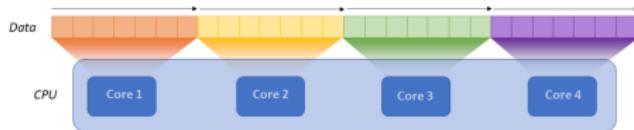
Early access/adoption of aggressive optimizations & GPU support

Optimización de operadores

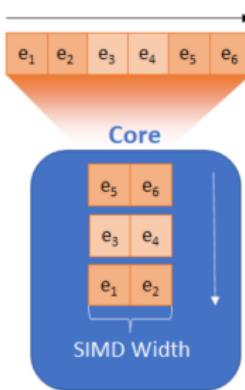
- Vectorización
- Paralelización
- Memoria
- Reducción de precisión

Niveles de paralelismo en un Multicore

- **Multiples cores:** multiples unidades de ejecución permiten procesar datos concurrentemente

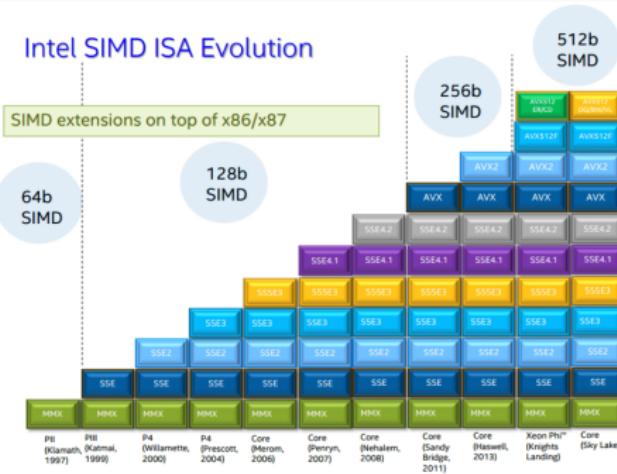


- **SIMD:** procesamiento en pequeños vectores
 - En ej. un SIMD=2 tardará la mitad



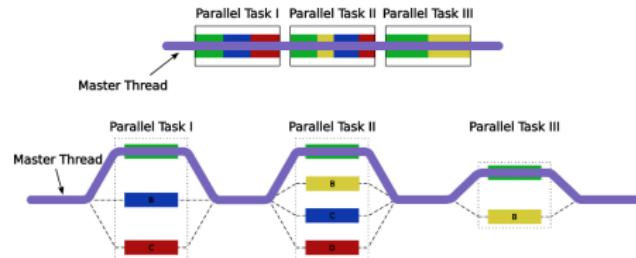
Vectorización

- La explotación de la vectorización **no es una teoría**
 - Grandes aceleraciones son posibles
 - SSE: 4xfp32; AVX: 8xfp32; AVX512: 16xfp32



Paralelización

- Los Intel® Xeon® 5^agen Emerald Rapids hasta 64 cores



Memory Layout

- Utilizado en cargas de trabajo con imágenes
- NCHW
 - Formato predeterminado
 - *torch.contiguous_format*
- NHWC
 - *torch.channels_last*
 - NHWC ofrece mayor rendimiento



Memory Layout

- Por defecto el formato es **NCHW**
 - En el caso de hacer uso de la GPU:

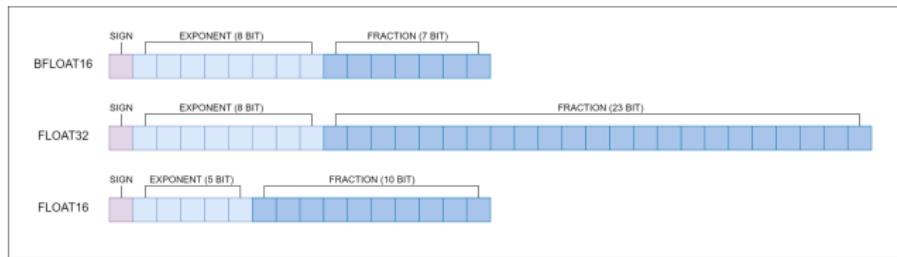
```
import torch
from torch import nn

input = torch.randn(1, 3, 16, 16) # 1 batch, 3 channels, image of 16x16
model = torch.nn.Conv2d(3, 10, 1, 1) # 3 input layers, 10 output channels,
    kernel 1x1
output = model(input)

# Execution in XPU
import intel_extension_for_pytorch as ipex
input_xpu = input.to("xpu")
model_xpu = model.to("xpu")
output_xpu = model_xpu(input_xpu)
```

BF16

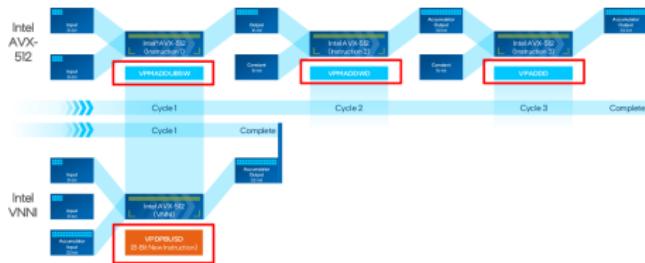
- FP32 proporciona un gran precisión para representar un número real
- En IA no se necesita tal capacidad de precisión
 - La conversión a BF16 es más sencilla y rápida que a FP16
- Incremento de rendimiento en fase de entrenamiento e inferencia: 2x vs fp32



```
import intel_extension_for_pytorch as ipex
with torch.xpu.amp.autocast(dtype=torch.bfloat16):
    input_bfp16 = model(input_xpu)
```

Cuantización a INT8

- ¿Qué es la cuantización?
 - Un método de aproximación
 - El proceso de mapear valores de un conjunto grande (por ejemplo, FP64/FP32) a aquellos con un conjunto más pequeño (por ejemplo BF16, INT8)
- El repertorio AVX512 tiene instrucciones VNNI para acelerar la inferencia AI/DL
- Combinando tres instrucciones en una maximiza el uso del hw, mejora la utilización de la caché y limitando los cuellos de botella
 - 1.74× mejor con INT8 en inferencia para procesamiento de lenguaje natural



Cuantización a INT8

- Cuantizar un modelo a INT8 en el Intel Extension para PyTorch

```
...
import intel_extension_for_pytorch as ipex

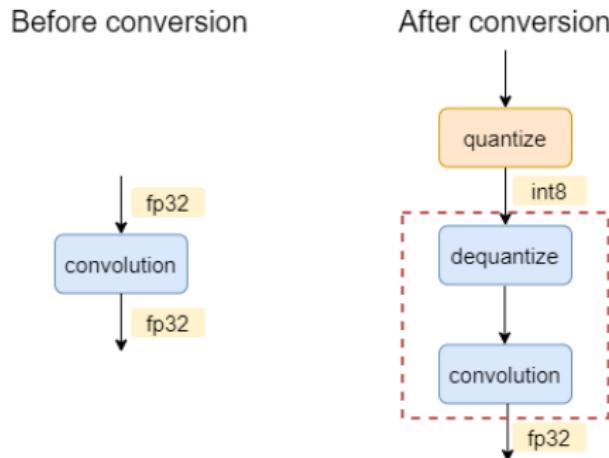
conf = ipex.QuantConf(dtype=torch.int8)

model, conf = ipex.quantization.prepare(model, conf)
for d in calibration_data_loader():
    # conf will be updated with observed statistics during calibrating with the
    # dataset
    with ipex.quantization.calibrate(conf):
        model(d)

conf.save('int8_conf.json', default_recipe=True)
model = ipex.quantization.convert(model, conf, torch.rand(<shape>))
```

Optimización del gráfico

- La mayoría de los modelos de aprendizaje profundo podrían describirse como DAG (gráfico acíclico dirigido), pudiendo conocer las dependencias entre capas
- La optimización del gráfico conlleva la fusión del operadores
 - Fusión de capas: $Conv(2, 3)D + ReLU\dots$



Optimización del gráfico

- La mayoría de los modelos de aprendizaje profundo podrían describirse como DAG (gráfico acíclico dirigido), pudiendo conocer las dependencias entre capas
- La optimización del gráfico conlleva la fusión del operadores
 - Fusión de capas: $Conv(2, 3)D + ReLU\dots$
 - Fusión de operaciones aritméticas:

```
for (i in 1:n)
  tmp[i,1] = A[i,1] * B[i,1]
for (i in 1:n)
  R[i,1] = tmp[i,1] + C[i,1]

# Equivalent
for (i in 1:n)
  R[i,1] = (A[i,1]* B[i,1]) + C[i,1]
```

Entrenamiento con el Intel Extension para PyTorch

```
import torch
import torchvision
#####
# code changes #####
import intel_extension_for_pytorch as ipex
#####

#####
# code changes #####
LR = 0.001
DOWNLOAD = True
DATA = 'datasets/cifar10/'

transform = torchvision.transforms.Compose([
    torchvision.transforms.Resize((224, 224)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
train_dataset = torchvision.datasets.CIFAR10(
    root=DATA,
    train=True,
    transform=transform,
    download=DOWNLOAD,
)
train_loader = torch.utils.data.DataLoader(
    dataset=train_dataset,
    batch_size=128
)
model = torchvision.models.resnet50()
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = LR, momentum=0.9)
model.train()
```

*The .to("xpu") is needed for GPU only
**Use torch.cpu.amp.autocast() for CPU
***Channels last format is automatic

```
#####
# code changes #####
model = model.to("xpu")
criterion = criterion.to("xpu")
model, optimizer = ipex.optimize(model, optimizer=optimizer, dtype=torch.bfloat16)
#####

#####
# code changes #####
for batch_idx, (data, target) in enumerate(train_loader):
    optimizer.zero_grad()
    #####
    # code changes #####
    data = data.to("xpu")
    target = target.to("xpu")
    with torch.xpu.amp.autocast(enabled=True, dtype=torch.bfloat16):
        #####
        # code changes #####
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
        print(batch_idx)
    torch.save({
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
    }, 'checkpoint.pth')
```

Inferencia con el Intel Extension para PyTorch

Resnet50

```
import torch
import torchvision.models as models
#####
# code changes #####
import intel_extension_for_pytorch as ipex
#####

model = models.resnet50(pretrained=True)
model.eval()
data = torch.rand(1, 3, 224, 224)

#####
# code changes #####
model = model.to("xpu")
data = data.to("xpu")
model = ipex.optimize(model, dtype=torch.bfloat16)
#####

with torch.no_grad():
    d = torch.rand(1, 3, 224, 224)
#####
# code changes #####
    d = d.to("xpu")
    with torch.xpu.amp.autocast(enabled=True, dtype=torch.bfloat16):
#####
# code changes #####
        model = torch.jit.trace(model, d)
        model = torch.jit.freeze(model)
        model(data)
```

BERT

```
*The .to("xpu") is needed for GPU only
**Use torch.cpu.amp.autocast() for CPU
*** Channels last format is automatic

import torch
from transformers import BertModel
#####
# code changes #####
import intel_extension_for_pytorch as ipex
#####

model = BertModel.from_pretrained(args.model_name)
model.eval()

vocab_size = model.config.vocab_size
batch_size = 1
seq_length = 512
data = torch.randint(vocab_size, size=[batch_size, seq_length])

#####
# code changes #####
model = model.to("xpu")
data = data.to("xpu")
model = ipex.optimize(model, dtype=torch.bfloat16)
#####

with torch.no_grad():
    d = torch.randint(vocab_size, size=[batch_size, seq_length])
#####
# code changes #####
    d = d.to("xpu")
    with torch.xpu.amp.autocast(enabled=True, dtype=torch.bfloat16):
#####
# code changes #####
        model = torch.jit.trace(model, (d,), strict=False)
        model = torch.jit.freeze(model)

        model(data)
```

Model Zoo for Intel® Architecture

- **Repositorio** con los modelos más comunes preentrenados tanto en TensorFlow como en PyTorch para ser usados
 - Reconocimiento de imágenes
 - Segmentación de imágenes
 - Lenguaje natural
 - Detección de objetos
 - Recomendaciones
 - ...
- Tanto para CPU, como GPU con diferentes preciones (FP32, FP16, BF16, INT8)

Optimización ejecución

- CPU utiliza la herramienta **numactl** de Linux para implementar la afinidad de subprocessos
- PyTorch permite afinidad de hilos

Terminal #1

```
uXXXX@idc-beta-batch-pvc-node-03:~$ lscpu
...
Architecture:           x86_64
CPU(s):                 224
On-line CPU(s) list:   0-223
Model name:             Intel(R) Xeon(R) Platinum 8480+
...
Thread(s) per core:    2
Core(s) per socket:    56
...
NUMA node(s):          2
```

```
model = SimpleNet()
model.eval()
cpu_pool = ipex.cpu.runtime.CPUPool([1, 2, 3, 4])
with ipex.cpu.runtime.pin(cpu_pool):
    for i in range(steps):
        y_runtime = model(x)
```



Configuración en tiempo de ejecución

- *IPEX_VERBOSE=1*: muestra información de ejecución (por defecto a 0)
- *IPEX_SIMPLE_TRACE=ON*: muestra traza de las operaciones IPEX
- *OMP_NUM_THREADS=n*: controla el número de hilos de ejecución en CPU
- *KMP_AFFINITY*: controla el mapeo de hilos “lógicos” a cores físicos
- *IPEX_FP32_MATH_MODE*: modo de aritmética, FP32 por defecto
- *ONEDNN_PRIMITIVE_CACHE_CAPACITY*: configura la capacidad cache para almacenar algunos datos de entrada. Ej para *speech-recognition* el valor de 4096 ofrece mejor rendimiento

- ① Conéctate al Intel Developer Cloud
- ② En el apartado de **Training and Workshops**
 - Lanza el entorno de Jupyter
- ③ Dentro del IDC, abrir el cuaderno de la ruta
oneAPI_IberianTour23/ipex/IntelPyTorch_GPU_InferenceOptimization_with_AMP/IntelPyTorch_GPU_InferenceOptimization_with_AMP.ipynb



Introducing

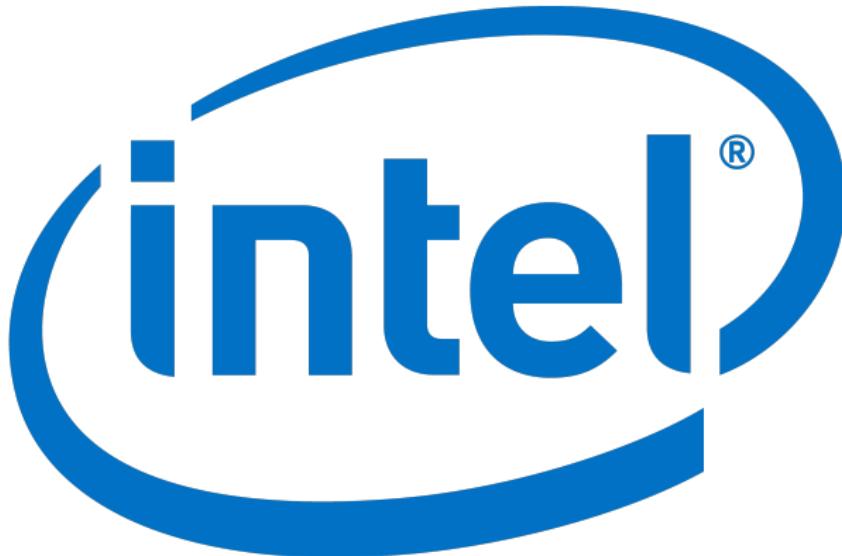
oneAPI

No transistor left behind

Otros

Recursos disponibles

- Curso ML con oneAPI
- Iniciativa oneAPI
- Intel IA Tools
- Documentación de Intel® Extension for Scikit-learn
- Repo Intel® Extension for Scikit-learn en [GitHub](#)
- Documentación de Intel® Extension for PyTorch
- Repo IPEX en [GitHub](#)
- Zoo Model
- Libro **Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL** disponible en el link



Software

¡¡¡Gracias!!!



Dirección

Avda. de la industria 4, edif. 1
28108 Alcobendas | Madrid | España

Correo

info@danysoft.com

Teléfono

[+34] 91 663 8683

Sitio Web

www.danysoft.com/intel

