CVPR SEATTLE, WA JUNE 17-21, 2024

OpenVINO™

CVPR 2024

Edge-Optimized Deep Learning: Harnessing Generative AI and Computer Vision with Open-Source Libraries

Module 1

Data Management, Training, and Fine-tuning Computer Vision Tasks

Samet Akcay and Harim Kang

Intel NEX SW

Agenda

By the end of this session, you will learn how to

01

Dataset

Zero-shot to annotate

Automated annotation

Datumaro dataset management.

02

Train / Test

Zero-shot SAM
Classification
Object Detection
Instance Segmentation

03

Export

What are the BKMs to run this model?

04

XAI

Create explainability saliency maps with the IR model

05

Deploy / Demo

Create a demo package ready to be deployed

OTX

Introduction

What is OpenVINO Training eXtensions – OTX?

- One-stop shop of verified algorithms for many vision tasks and learning methods
- Provides simple CLI and API for quick start without hassles
- Full OpenVINO integration for model optimization, inference and deployment.

HK0

What is OpenVINO Training eXtensions – OTX?

One command is all you need

CLI

```
# <train, test, export, explain, deploy>
# $ otx <entrypoint> --arg value
$ otx train \
   --task detection  \
   --data_root /path/to/data …
```

API

```
>>> from otx.engine import Engine
>>> engine = Engine(data_root="data/wgisd")
>>> engine.train()
```

Slide 5

HK0 [@Akcay, Samet] On that page, we can also omit --task from the CLI command to show that the CLI and API are similar. Just for reference.

Kang, Harim, 2024-06-04T13:58:52.080

What is OpenVINO Training eXtensions – OTX?

01

Dataset

How does my data look?

02

Models

Which AI model can I use?

03

BKM

What are the BKMs to run this model?

04

Improve

Is this the highest performance I can achieve?

05

Optimize

How to maximize the model's run-time efficiency?

06

Deploy

How do I create a working inference example?

What is OpenVINO Training eXtensions – OTX?

01	02	03	04	05	06
Dataset	Models	BKM	Improve	Optimize	Deploy
How does my data look?	Which AI model can I use?	What are the BKMs to run this model?	Is this the highest performance I can achieve?	How to maximize the model's run-time efficiency?	How do I create a working inference example?

01 – Leverage Datumaro, the data frontend of OTX

RP0

What is OpenVINO Training eXtensions – OTX?

01	02	03	04	05	06
Dataset	Models	BKM	Improve	Optimize	Deploy
How does my data look like	Which AI model can I use?	What are the BKMs to run this model?	Is this the highest performance I can achieve?	How to maximize the model's run-time efficiency?	How do I create a working inference example?

02 – Use OpenVINO-verified models for best performance and efficiency

```
# CLI
$ otx find
```

```
# API
>>> from otx.models import list_models
>>> list_models(task="DETECTION")
```

Slide 8

RP0	[@Akcay, Samet] Could we show which models are available in the library? Ramos, Paula, 2024-05-29T13:51:42.375
RP0 0	It seems slide 14 has that info. Maybe you could verbalize those models here. Ramos, Paula, 2024-05-29T13:54:12.301

What is OpenVINO Training eXtensions – OTX?

01

Dataset

How does my data look like

02

Models

Which AI model can I use?

03

BKM

What are the BKMs to run this model?

04

Improve

Is this the highest performance I can achieve?

05

Optimize

How to maximize the model's run-time efficiency?

06

Deploy

How do I create a working inference example?

03/04 – Autoconfiguration will find the best task type, model and parameters. Advanced customization is also possible

```
# CLI
$ otx train
```

```
# API
>>> from otx.engine import Engine
>>> engine = Engine(data_root="data/wgisd")
>>> engine.train()
```

What is OpenVINO Training eXtensions – OTX?

01
Dataset
How does my data look like

02
Models
Which AI model can I use?

03
BKM
What are the BKMs to run this model?

04
Improve
Is this the highest performance I can achieve?

05
Optimize
How to maximize the model's run-time efficiency?

06
Deploy
How do I create a working inference example?

05 – Use OpenVINO NNCF-based model optimization and export runnable codes with model

```
# CLI
$ otx export
$ otx optimize
```

```
# API
>>> ir_model_path = engine.export()
>>> engine.optimize(ir_model_path)
```

What is OpenVINO Training eXtensions – OTX?

01	02	03	04	05	06
Dataset	Models	BKM	Improve	Optimize	Deploy
How does my data look like	Which AI model can I use?	What are the BKMs to run this model?	Is this the highest performance I can achieve?	How to maximize the model's run-time efficiency?	How do I create a working inference example?

06 –Evaluate, explain and deploy models with built-in CLI and API

```
# CLI
$ otx explain
$ otx export
$ python demo.py
```

```
# API
>>> engine.explain(checkpoint="<checkpoint-path>",
datamodule=OTXDataModule(...),explain_config=ExplainConfig(postprocess=True),dump=True)
>>> engine.export(export_format='EXPORTABLE_CODE')
```

AS0

intel. OpenVINO

Slide 11

AS0 Also this?
Akcay, Samet, 2024-04-23T10:52:19.604

RC0 0 [@Akcay, Samet]: I reviewed the docs, it looks like there isn't a current solution for using engine.deploy. The only deployment method mentioned is via Python.
Cheruvu, Ria, 2024-05-23T14:58:14.078

RP0 1 [@Akcay, Samet] what is the version of OTX are we using, are the PRs ready for developer's usage?
Ramos, Paula, 2024-05-29T13:53:11.440

AS0 2 I plan to use this branch
https://github.com/openvinotoolkit/training_extensions/tree/tutorials/cvpr24
Akcay, Samet, 2024-05-30T06:21:41.148

OTX

Features

Installation

Lightweight, hardware-agnostic installation

CLI

```
# <train, test, export, explain, deploy>
# Install via PyPI
$ pip install otx

# Multiple installation options
otx install --help
```

Features

End-to-end DL pipeline for all levels – *From beginner to Advanced*

Task Types

- Classification
- Detection, Rotated Detection
- Semantic and Instance Segmentation
- Anomaly Detection
- Action Recognition
- Visual Prompting

Learning Methods

- Fully-supervised
- Semi-supervised
- Self-supervised
- Class Incremental
- Imbalanced

API / CLI Functionality

- Auto-installation
- Auto-learning method
- Integrated Image Tiling
- Hyper-parameter Optimization
- OpenVINO Optimization
- Integrated Explainable AI (XAI)

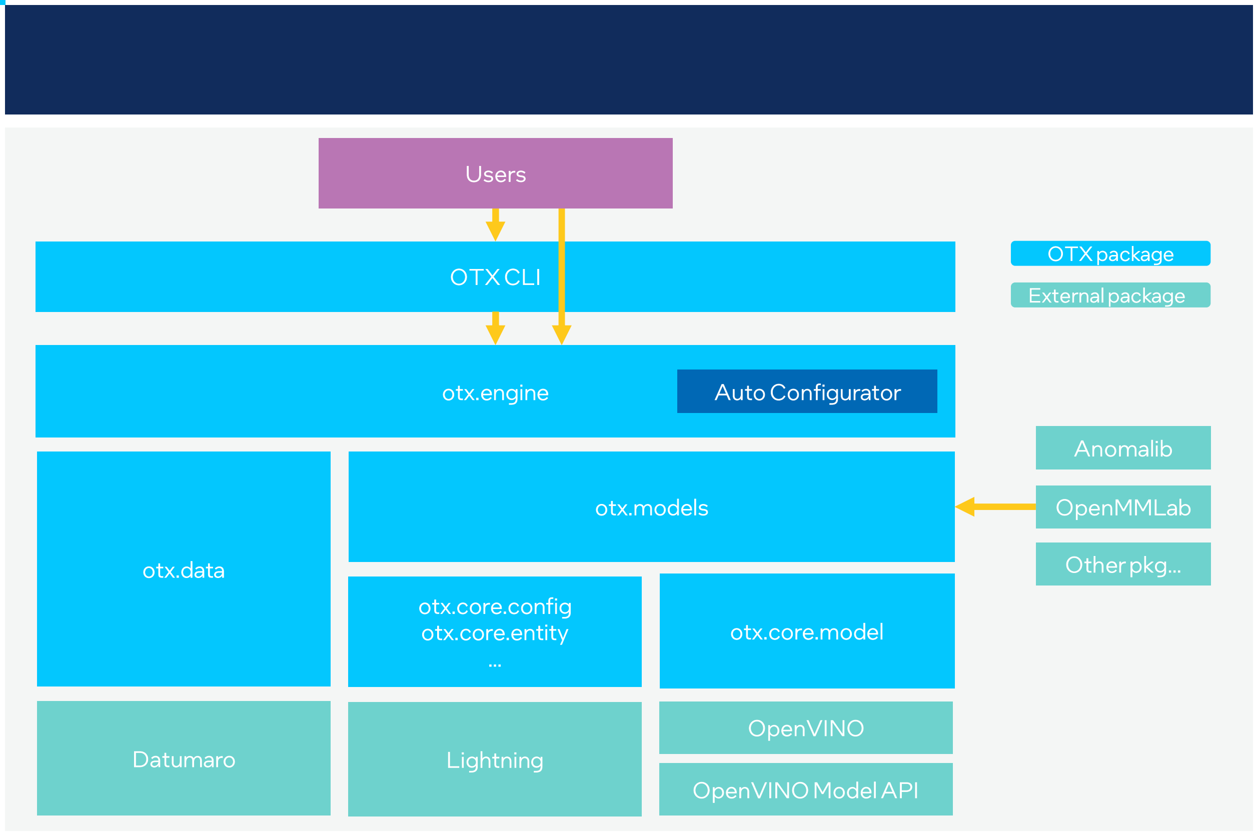
OTX

Architecture

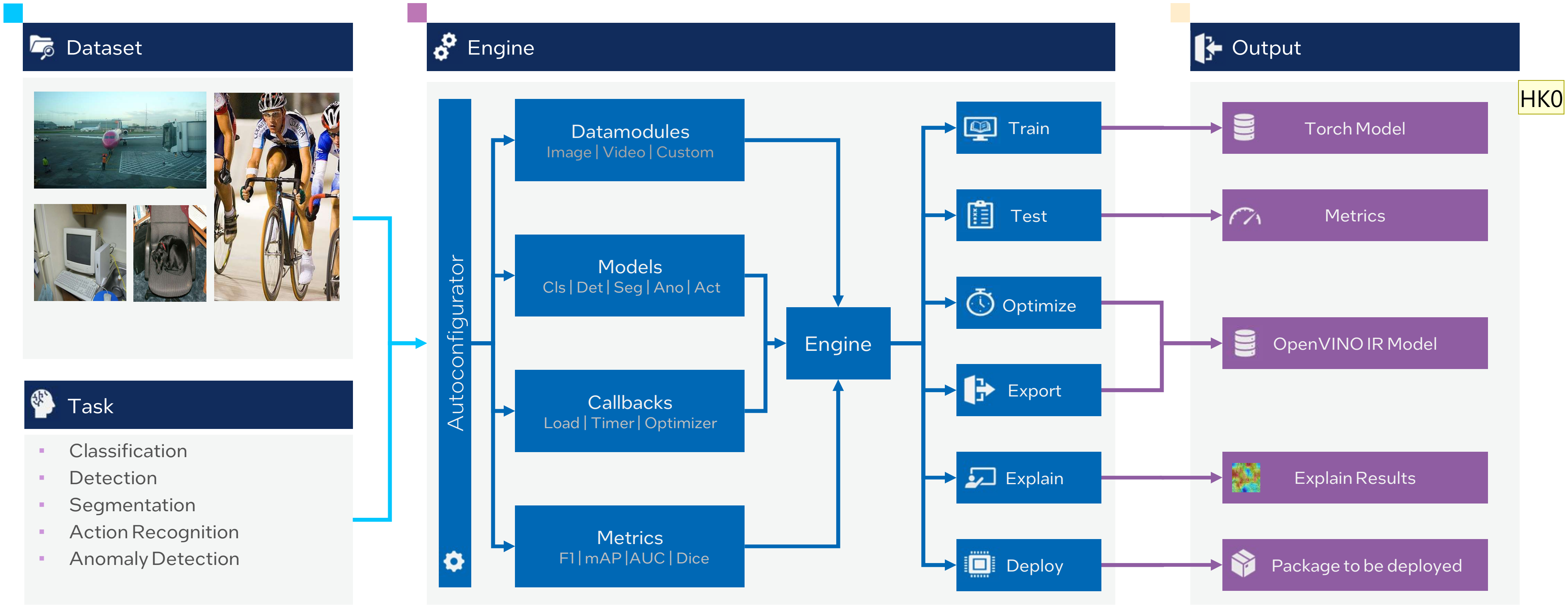
OTX Architecture

API/CLI Parity

- Jsonargparse-based CLI
- Lightning-based Data and Model
- Engine orchestrates the pipeline.



Autoconfiguration



Slide 17

HK0 [@Akcay, Samet] The output is a bit weird, I'm expecting something like torch model, IR Model, explain result. What do you think?
Kang, Harim, 2024-05-23T05:59:33.709

AS0 0 Yes, great point! Thanks!
Akcay, Samet, 2024-05-23T06:20:26.486

RC0 1 Hi [@Kang, Harim] and [@Akcay, Samet] : I modified the diagram to focus on the output deliverables. I agree that this would be much clearer and highlight the value of OTX better. Please let me know what you think.
Cheruvu, Ria, 2024-05-23T14:59:23.591

Autoconfiguration

CLI

```
# TASKS:
# - MULTI_LABEL_CLS, MULTI_CLASS_CLS, DETECTION,
# - INSTANCE_SEGMENTATION, SEMANTIC_SEGMENTATION,
# - ACTION_RECOGNITION
$ otx train \
    --task <TASK> \
    --data_root data/VOCdevkit/VOC2012 \
    --data.config.data_format voc
```

AS1

AS0

API

```
# API via config
from otx.engine import Engine

engine = Engine(
    data_root=data_root,
    task="INSTANCE_SEGMENTATION",
    work_dir="otx-workspace-api-ins-seg-auto",
)

engine.train(max_epochs=3)
```

Slide 18

AS0 Do we always need to provide `—work_dir`?
Akcay, Samet, 2024-04-22T18:48:31.389

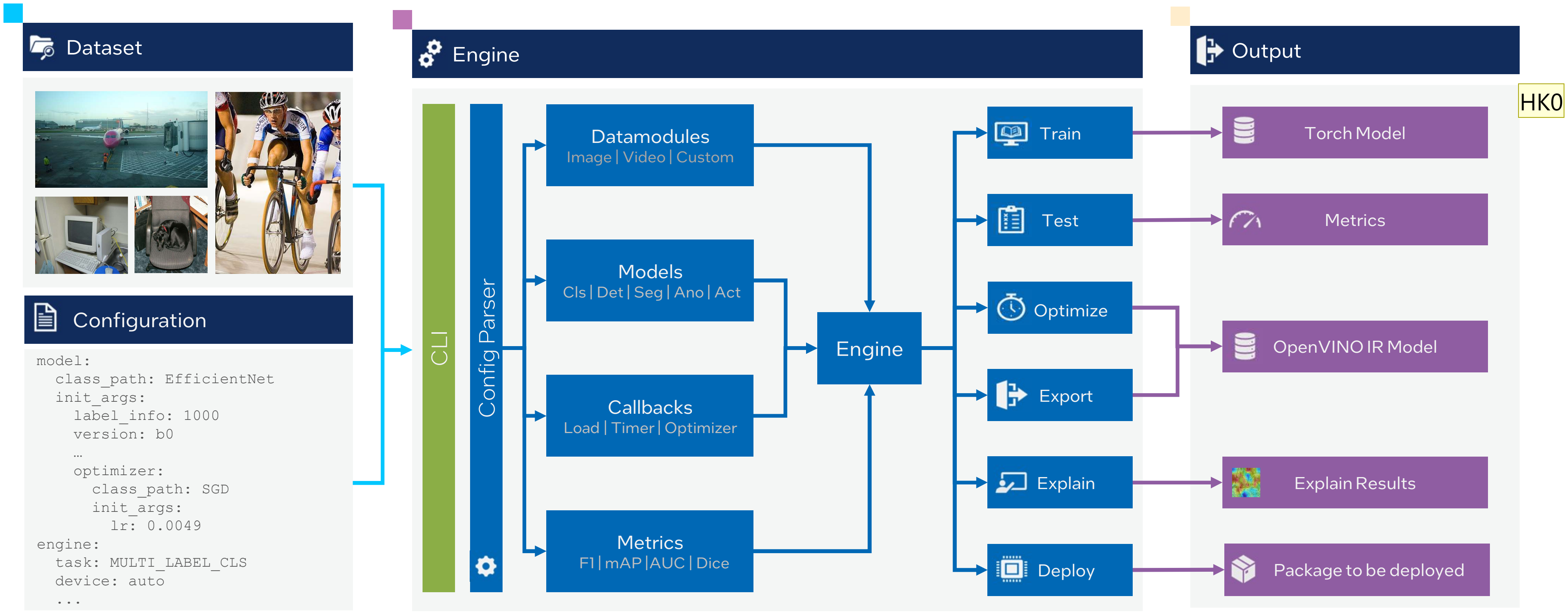
HK0 0 --work_dir is optional. Default value: ./otx-workspace
Kang, Harim, 2024-05-23T06:00:02.443

AS1 Ideally, this should be —data.format

Why do we need to define this as data.config.data_format
Akcay, Samet, 2024-04-22T18:49:14.902

HK1 0 We still need to specify the data_format.
https://github.com/openvinotoolkit/training_extensions/issues/3227
Kang, Harim, 2024-05-23T06:00:43.374

via Config File



```
model:
  class_path: EfficientNet
  init_args:
    label_info: 1000
    version: b0
    …
    optimizer:
      class_path: SGD
      init_args:
        lr: 0.0049
engine:
  task: MULTI_LABEL_CLS
  device: auto
  ...
```

Slide 19

HK0	[@Akcay, Samet] The output is a bit weird, I'm expecting something like torch model, IR Model, explain result. What do you think? Kang, Harim, 2024-05-23T05:59:33.709
AS0 0	Yes, great point! Thanks! Akcay, Samet, 2024-05-23T06:20:26.486
RC0 1	Hi [@Kang, Harim] and [@Akcay, Samet] : I modified the diagram to focus on the output deliverables. I agree that this would be much clearer and highlight the value of OTX better. Please let me know what you think. Cheruvu, Ria, 2024-05-23T14:59:23.591

End-to-End Training via Config File

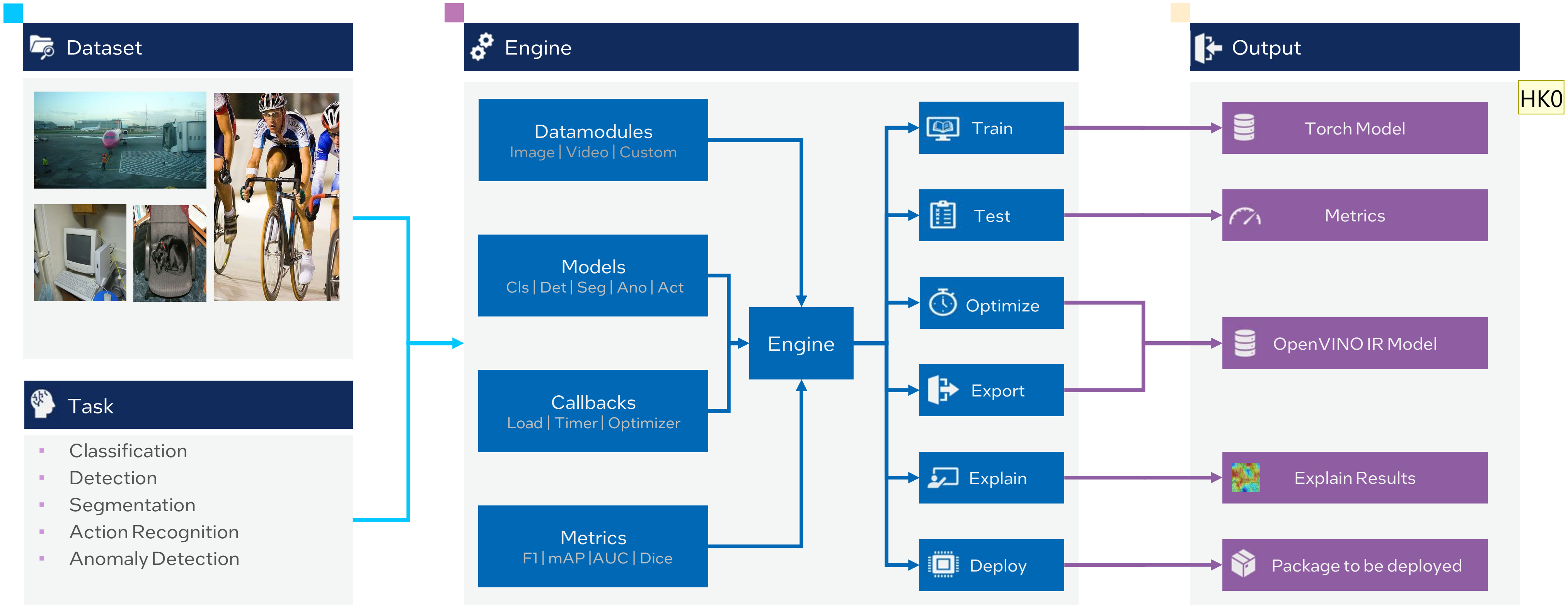
CLI

```
# Train via config file
$ otx train \
    --config efficientnet_b0_light.yaml \
    # Overwrite some arguments (Optional)
    --data_root data/VOCdevkit/VOC2012 \
    --data.config.data_format voc \
    --work_dir otx-workspace-api-multi-label-cls
```

API

```
# API via config
from otx.engine import Engine

data_root = "data/VOCdevkit/VOC2012"
recipe = "src/otx/recipe/classification/multi_label_cls/
              "efficientnet_b0_light.yaml"
override_kwargs = {"data.config.data_format": "voc"}
engine = Engine.from_config(
  config_path=recipe,
  data_root=data_root,
  work_dir="otx-workspace-api-multi-label-cls",
  **override_kwargs,
)

engine.train(max_epochs=2, precision=16)
```

via API Modules



- Classification
- Detection
- Segmentation
- Action Recognition
- Anomaly Detection

Slide 21

HK0 [@Akcay, Samet] The output is a bit weird, I'm expecting something like torch model, IR Model, explain result. What do you think?
Kang, Harim, 2024-05-23T05:59:33.709

AS0 0 Yes, great point! Thanks!
Akcay, Samet, 2024-05-23T06:20:26.486

RC0 1 Hi [@Kang, Harim] and [@Akcay, Samet] : I modified the diagram to focus on the output deliverables. I agree that this would be much clearer and highlight the value of OTX better. Please let me know what you think.
Cheruvu, Ria, 2024-05-23T14:59:23.591

via API Modules

CLI

```
# CLI via API modules
$ otx train
    --config src/otx/recipe/classification/
          multi_label_cls/efficientnet_b0_light.yaml \
    --data_root data/VOCdevkit/VOC2012 \
    --data.config.data_format voc \
```

API

```
# API via modules
>>> datamodule = OTXDataModule(
...     task="MULTI_LABEL_CLS",
...     config=DataModuleConfig(
...         data_format="voc",
...         data_root=data_root,
...           train_subset=SubsetConfig(
...           subset_name="train",
...           batch_size=8,
...           num_workers=2,
...           transform_lib_type="MMPRETRAIN",
...           transforms=simple_transforms,
...         val_subset=SubsetConfig(...),
...         test_subset=SubsetConfig(...),
...     ),
... ),

>>> model = EfficientNetB0ForMultilabelCls(num_classes)

>>> engine = Engine(datamodule=datamodule,model=model)
>>> engine.train(max_epochs=2)
```

OTX

Additional Features

Additional OTX Features

01

Custom Data and Models

It is possible to create custom data and models.

02

Image Tiling

Improve performance for small object detection

03

XPU Support

Train/Test models using Intel XPUs

04

HPO

Hyper-parameter optimization to tune the model performance

05

Export to Different Precision

Possibility to export to different precision such as int8

06

Optimization

Optimization support including post-training quantization

AS0

Custom Data

API

```
# Add Image Tiling API Example Here.
my_transforms = [
   Resize(size=[224, 224]),
   # Your list of custom transforms here.
]

datamodule = OTXDataModule(task="MULTI_LABEL_CLS",
   config=DataModuleConfig(
      data_format="voc",
      data_root=data_root,
      train_subset=SubsetConfig(
            subset_name="train",
            batch_size=32,
            num_workers=2,
            transform_lib_type="TORCHVISION",
            transforms= my_transforms,
        ),
        val_subset=SubsetConfig(...),
        test_subset=SubsetConfig(...
        ),
        ),
)
```

Slide 25

AS0 We will need to add an image tiling example here. [@Kang, Harim]
Akcay, Samet, 2024-05-30T10:55:18.750

HK0 0 You can refer
https://openvinotoolkit.github.io/training_extensions/latest/guide/explanation/additional_features/tiling.html
Or maybe you can ask Eugene for help.
Kang, Harim, 2024-05-31T01:05:08.987

AS0 1 Thanks Harim. My feedback for the data module stuff is valid here for the tiler as well.
Akcay, Samet, 2024-05-31T05:16:53.235

AS0 2 Ideally, we should get rid of the `config` stuff. This flag could just be `data.tile.enable True`. The rest is just extra boiler plate and verbosity
Akcay, Samet, 2024-05-31T05:17:37.141

HK0 3 This is woven into the OTXDataModuleConfig, so as we continue to talk about OTXDataModules with the team, I think this will be included.
Kang, Harim, 2024-06-03T05:08:49.945

AS0 4 [@Kang, Harim], should we add the cli example here as well?
Akcay, Samet, 2024-06-04T20:30:07.155

HK0 5 For simplicity, we could add an example like the one below.
Change Batch size of train subset:

```
otx train ... --data.config.train_subset.batch_size <batch-size>
```

Kang, Harim, 2024-06-05T02:45:55.515

AS0

Custom Model

Create torchvision models

CLI

```
otx train \
    --model otx.algo.classification.OTXTVModel \
    --model.backbone convnext_small \
    --data_root otx_v2_dataset/multiclass_CUB_small/1 \
    --work_dir otx-workspace-convnext \
    --max_epochs 2
```

API

```
# Imports
from otx.algo.classification import OTXTVModel
from otx.engine import Engine

# Create a torchvision model
tv_model = OTXTVModel(backbone="convnext_small", label_info=2)

# Multi-Class Classification
engine = Engine(
   data_root="otx_v2_dataset/multiclass_CUB_small/1",
   model=tv_model,
   work_dir="otx-workspace-tv-model",
)
engine.train(max_epochs=2)
```

Slide 26

AS0	We will need to add an image tiling example here. [@Kang, Harim] Akcay, Samet, 2024-05-30T10:55:18.750
HK0 0	You can refer https://openvinotoolkit.github.io/training_extensions/latest/guide/explanation/additional_features/tiling.html Or maybe you can ask Eugene for help. Kang, Harim, 2024-05-31T01:05:08.987
AS0 1	Thanks Harim. My feedback for the data module stuff is valid here for the tiler as well. Akcay, Samet, 2024-05-31T05:16:53.235
AS0 2	Ideally, we should get rid of the \`config\` stuff. This flag could just be \`data.tile.enable True\`. The rest is just extra boiler plate and verbosity Akcay, Samet, 2024-05-31T05:17:37.141
HK0 3	This is woven into the OTXDataModuleConfig, so as we continue to talk about OTXDataModules with the team, I think this will be included. Kang, Harim, 2024-06-03T05:08:49.945

AS0

Custom Model

Custom objective functions

API

```
from otx.algo.classification.efficientnet import EfficientNetForMultilabelCls
from otx.algo.classification.losses. import AsymmetricAngularLossWithIgnore

model = EfficientNetForMultilabelCls(
   label_info=datamodule.label_info,
   loss_callable=AsymmetricAngularLossWithIgnore(),
)

# Multi-Label Classification
engine = Engine(
   datamodule=datamodule,
   model=model,
   work_dir="otx-workspace-api-multi-label-cls",
)

engine.train(max_epochs=2)
```

Slide 27

AS0 [@Kang, Harim] would it be an idea here to include the CLI example for this one ? Thoughts?
Akcay, Samet, 2024-06-03T10:57:50.010

HK0 0 I think we can add examples using the torchvision model. I'll write one and share it with you.
Kang, Harim, 2024-06-03T11:46:52.192

AS0

Image Tiling

CLI

```
# Add Image Tiling CLI Example Here.
$ otx train
    ...
    --data.config.tile_config.enable_tiler True
```

API

```
# Add Image Tiling API Example Here.
>>> datamodule = OTXDataModule(
...    task="DETECTION",
...    config=DataModuleConfig(
...          …
...          tile_config=TileConfig(enable_tiler=True),
...     ),

... ),

>>> engine = Engine(datamodule=datamodule,model=model)
>>> engine.train(max_epochs=2)
```

Slide 28

AS0 We will need to add an image tiling example here. [@Kang, Harim]
Akcay, Samet, 2024-05-30T10:55:18.750

HK0 0 You can refer
https://openvinotoolkit.github.io/training_extensions/latest/guide/explanation/additional_features/tiling.html
Or maybe you can ask Eugene for help.
Kang, Harim, 2024-05-31T01:05:08.987

AS0 1 Thanks Harim. My feedback for the data module stuff is valid here for the tiler as well.
Akcay, Samet, 2024-05-31T05:16:53.235

AS0 2 Ideally, we should get rid of the `config` stuff. This flag could just be `data.tile.enable True`. The rest is just extra boiler plate and verbosity
Akcay, Samet, 2024-05-31T05:17:37.141

HK0 3 This is woven into the OTXDataModuleConfig, so as we continue to talk about OTXDataModules with the team, I think this will be included.
Kang, Harim, 2024-06-03T05:08:49.945

XPU Support

CLI

```
# CLI – Installation for XPU support
$ pip install '.[xpu]'
>     --extra-index-url https://pytorch-extension.intel.com/release-
whl/stable/xpu/us/
>    --data.config.tile_config.enable_tiler True

$ source /path/to/intel/oneapi/setvars.sh
$ export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
$ export IPEX_FP32_MATH_MODE=TF32

# CLI – XPU Support
$ otx train
    ...
    --engine.device xpu
```

API

```
# API – XPU Support
>>> from otx.engine import Engine
>>> engine = Engine(…, device='xpu')
>>> engine.train()
```

AS0

Hyper-parameter Optimization

CLI

```
# CLI – HPO
$ otx train
    ...
    --run_hpo True
```

API

```
# API – HPO
>>> from otx.engine import Engine
>>> engine = Engine(..)
>>> engine.train(run_hpo=True)
```

Slide 30

AS0	HPO Akcay, Samet, 2024-05-30T10:56:45.380
AS0 0	[@Kang, Harim] Akcay, Samet, 2024-05-30T10:56:51.906
AS0 1	Is this supported properly? Akcay, Samet, 2024-05-30T14:35:34.311
HK0 2	Yes, it works fine. Since Engine is the core entry point to the API and CLI, both behaviors provide roughly the same functionality. Kang, Harim, 2024-05-31T01:08:18

Export to Different Precision

CLI

```
# CLI – Precision Example
$ otx export
     ...
     --export_precision FP16
```

API

```
# API – Precision Example
>>> from otx.engine import Engine
>>> engine = Engine(..)
>>> engine.export(export_precision='FP16')
```

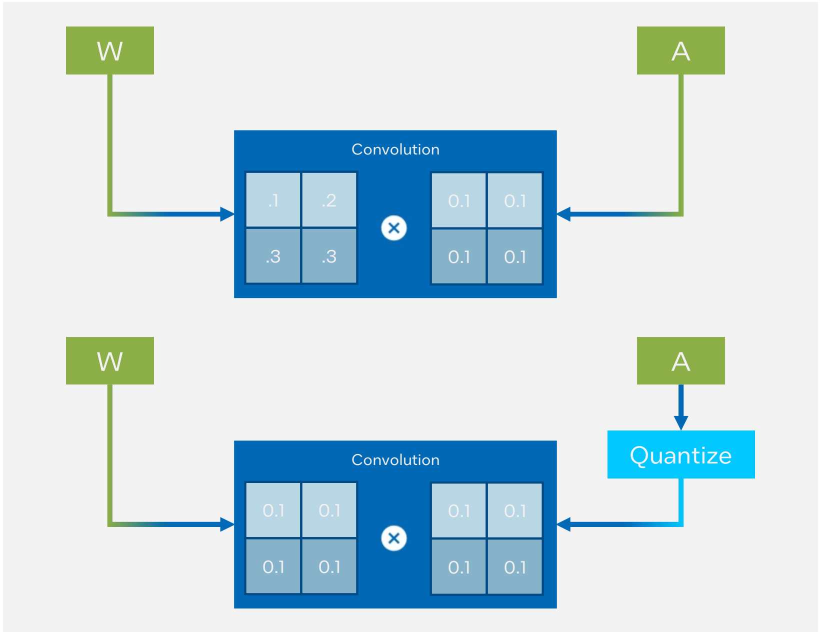
AS0

Optimization via NNCF

API

```
import nncf
import openvino.runtime as ov
from torch.utils.data import DataLoader

from torchvision.datasets import ImageFolder
from torchvision.transforms import Compose, ToTensor
from nncf.Dataset import NNCFDataset

# Instantiate your uncompressed model
model = ov.Core().read_model("/path/to/model.xml")

# Provide validation part of the dataset to collect statistics needed
# for the compression algorithm
transforms = Compose([ToTensor()])
val_dataset = ImageFolder("/path/to/dataset", transform=transforms)
val_dataloader = DataLoader(val_dataset, batch_size=1, shuffle=False)

# Step 1: Initialize the transform function
def transform_fn(data_item):
    images, _ = data_item
    return images

# Step 2: Initialize the NNCF dataset
calibration_dataset = NNCFDataset(val_dataloader, transform_fn)

# Step 3: Runt the quantization pipeline
quantized_model = nncf.quantize(model, calibration_dataset)
```

High-Level Diagram



Slide 32

AS0	An example showing different precision support Akcay, Samet, 2024-05-30T10:57:41.494
AS0 0	[@Kang, Harim] Akcay, Samet, 2024-05-30T10:57:46.272

RP0

Practical Implementation

intel. OpenVINO

Slide 33

RP0	[@Akcay, Samet] Please add QR Code for demo - Notebook will be ideal Ramos, Paula, 2024-05-29T14:01:53.938

Get Started

Installation

Installation

PyPI Install

```
python -m venv .otx
source .otx/bin/activate

# Install OTX CLI
pip install otx
# Install the full functionality via OTX CLI
otx install -v
```

Notebook

Documentation

https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/000_install.ipynb

Installation

Source Installation

```
git clone https://github.com/openvinotoolkit/training_extensions.git
cd training_extensions

python -m venv .otx && source .otx/bin/activate

pip install -e .
otx install -v
```

Notebook

Documentation

https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/000_install.ipynb

Use Case

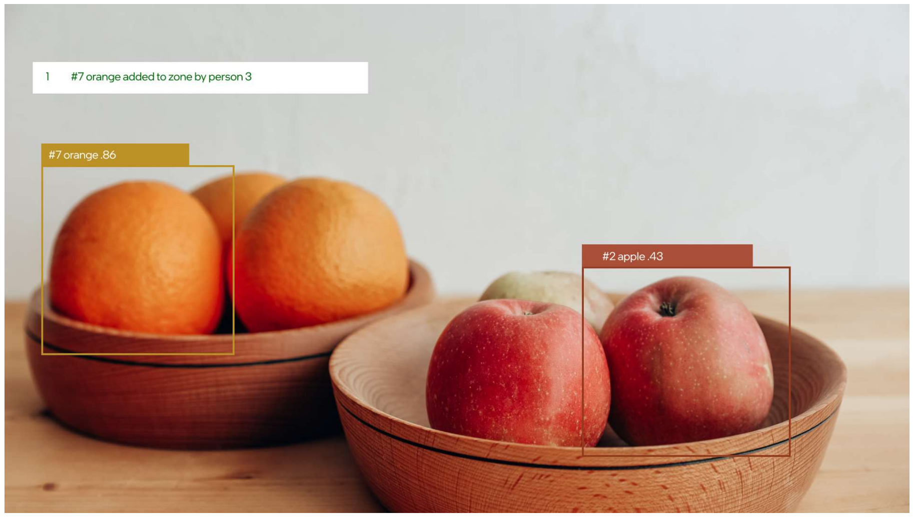
Problem Definition

Self Checkout Dataset – *Representative example*

Classification | Detection | Segmentation | VLMs

Self-Checkout in Retail: Challenges

- Real-time scalability
- Model performance
- Memory-efficiency and low-power to run on edge



https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/000_install.ipynb

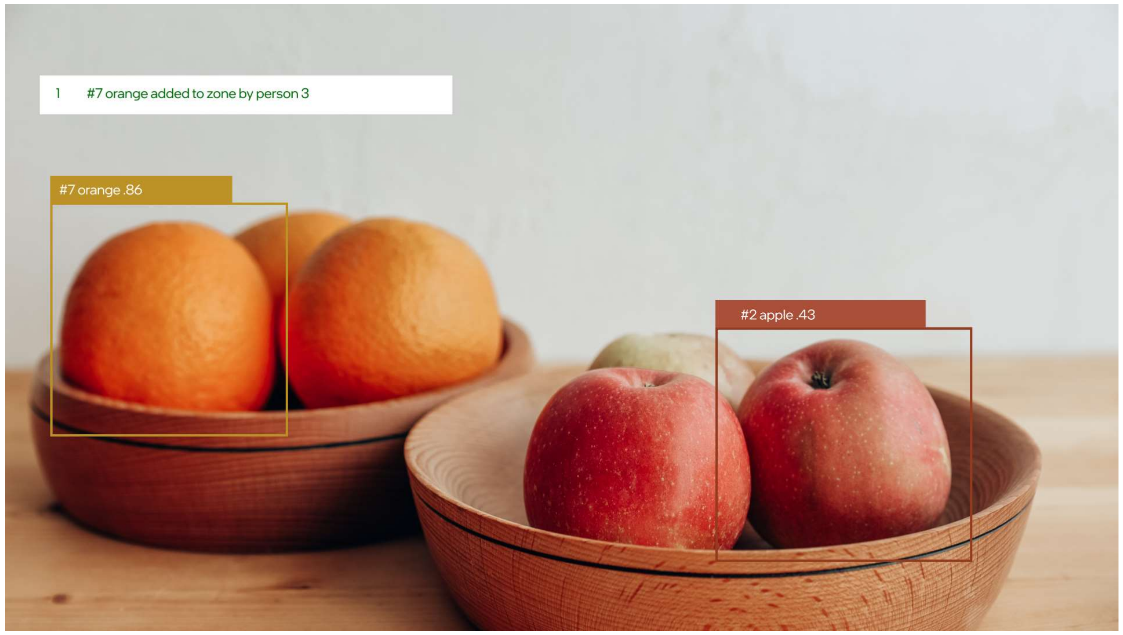
Self-Checkout in Retail: Solutions

01 Zero-shot Visual Prompting

02 Classification

03 Detection

04 Segmentation



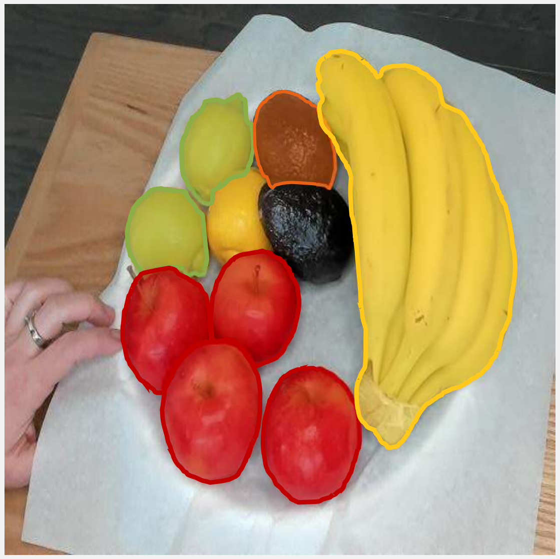
https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/000_install.ipynb

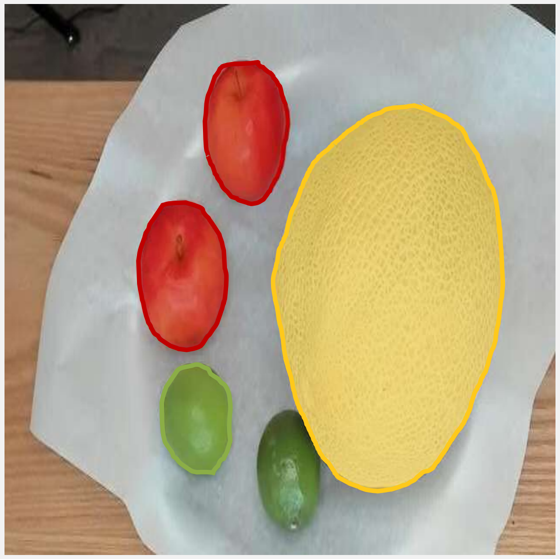
Task Types

Zero-Shot Visual Prompting

Zero-shot via SAM

No annotation, no problem! Create annotations via zero-shot SAM





Notebook

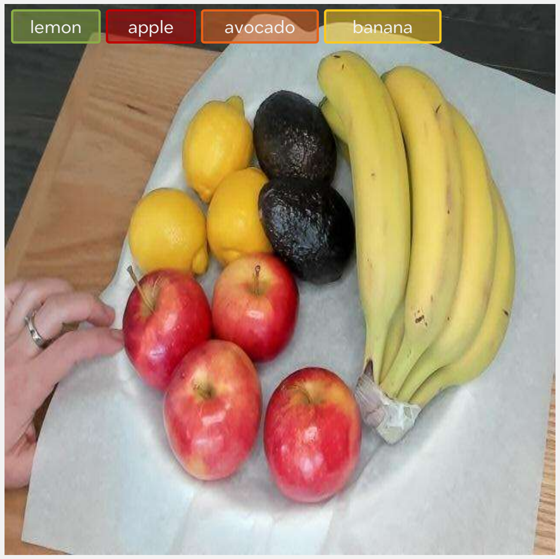
https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/001_zero_shot_visual_prompting.ipynb

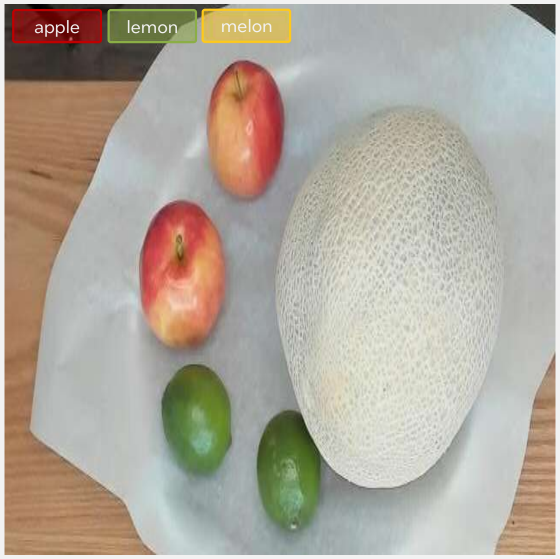
Task Types

Classification

Multi-Label Classification

Classify multi-labels





Notebook

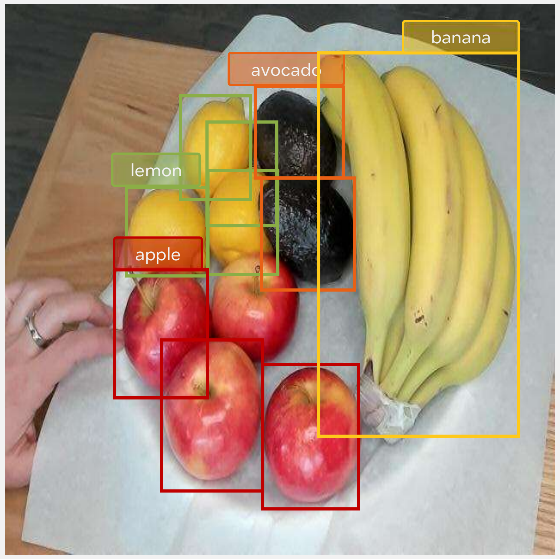
https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/002_multi_label_classification.ipynb

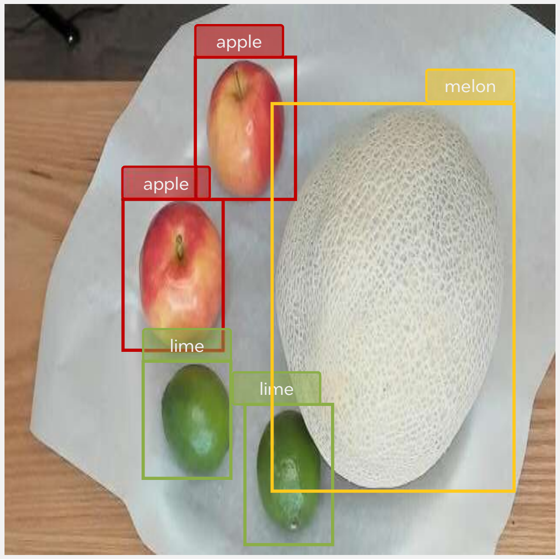
Task Types

Detection

Object Detection

Localize objects with bounding boxes





Notebook

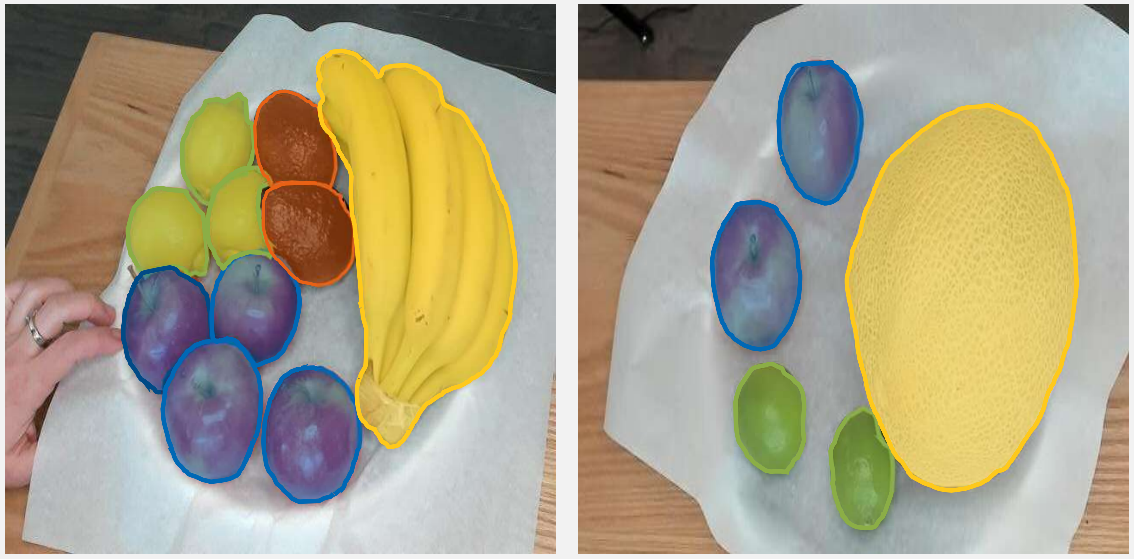
https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/003_detection.ipynb

Task Types

Segmentation

Semantic Segmentation

Pixel-wise semantic segmentation

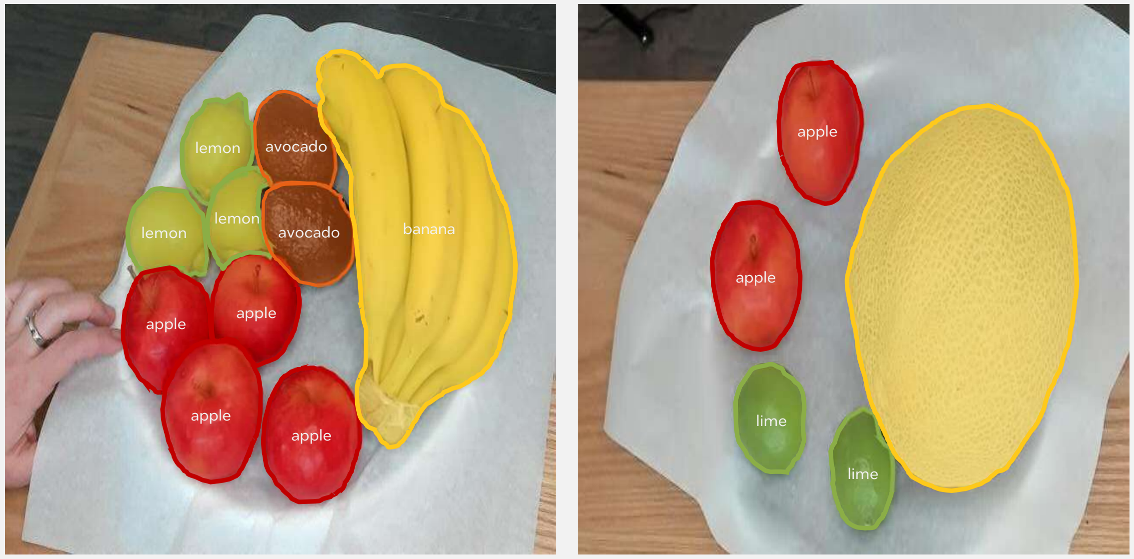


Notebook

https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/004_segmentation.ipynb

Instance Segmentation

Assign unique label to each detected object



Notebook

https://github.com/openvinotoolkit/training_extensions/blob/tutorials/cvpr24/notebooks/005_instance_segmentation.ipynb

Next Steps?

Many more features to try!

Data Management

- Image Tiling
- Noisy Label Detection
- Fast Data Loading

Algorithms

- Supervised CV tasks
- Semi and self supervised
- Visual prompting
- Adaptive Training
- HPO

Deploy

- Optimize with OpenVINO
- Deploy at the edge

https://github.com/openvinotoolkit/training_extensions