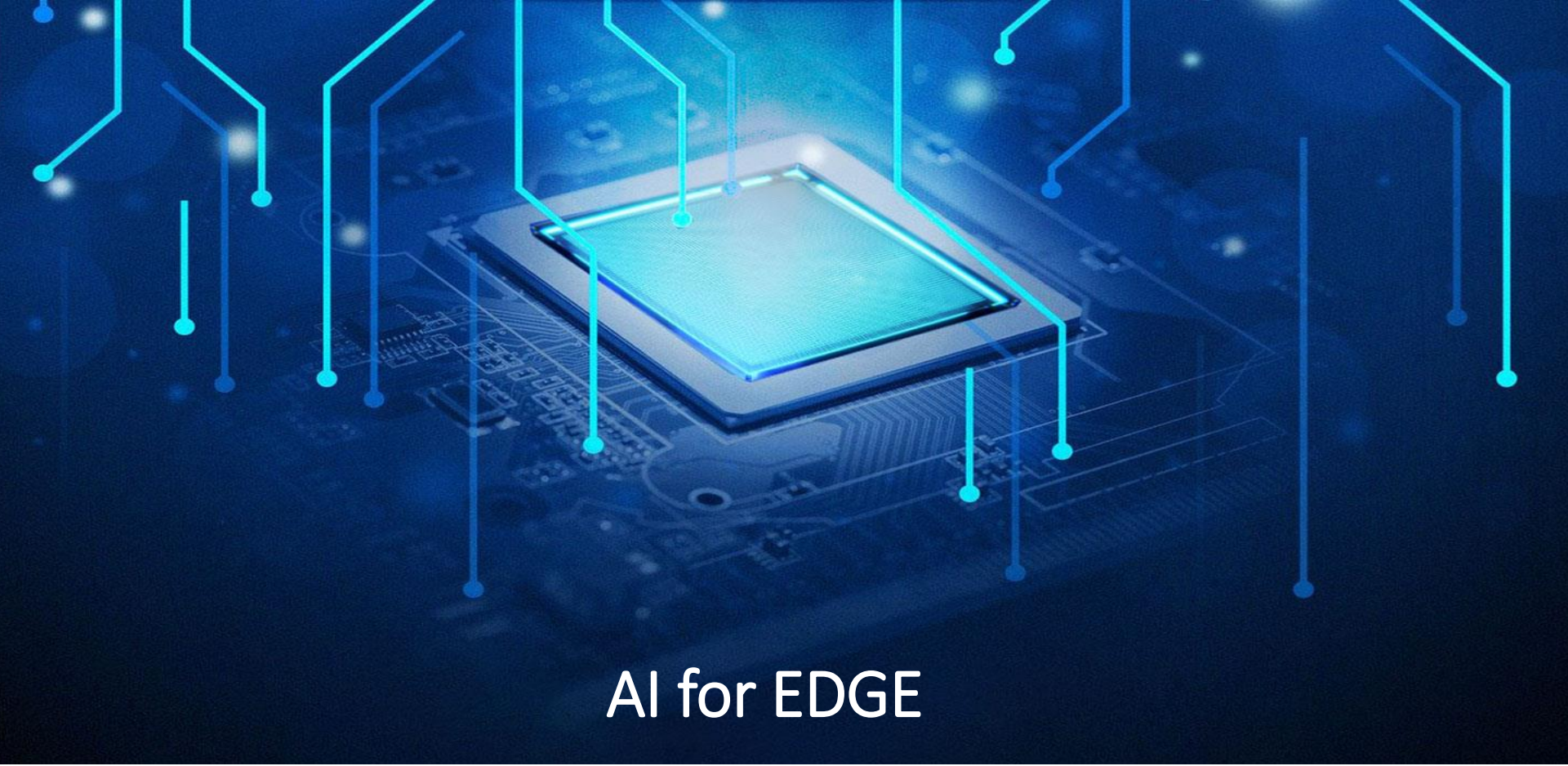


AI for EDGE

Machine learning frameworks

Outline

Machine learning Frameworks

- TensorFlow
- Keras
- scikit-learn
- PyTorch
- OpenVINO

- Machine learning in Python
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

API Reference

This is the class and function reference of scikit-learn. Please refer to the full user guide for further details, as the class and function raw specifications may not be enough to give full guidelines on their uses. For reference on concepts repeated across the API, see Glossary of Common Terms and API Elements.

`sklearn.base`: Base classes and utility functions

Base classes for all estimators.

Base classes

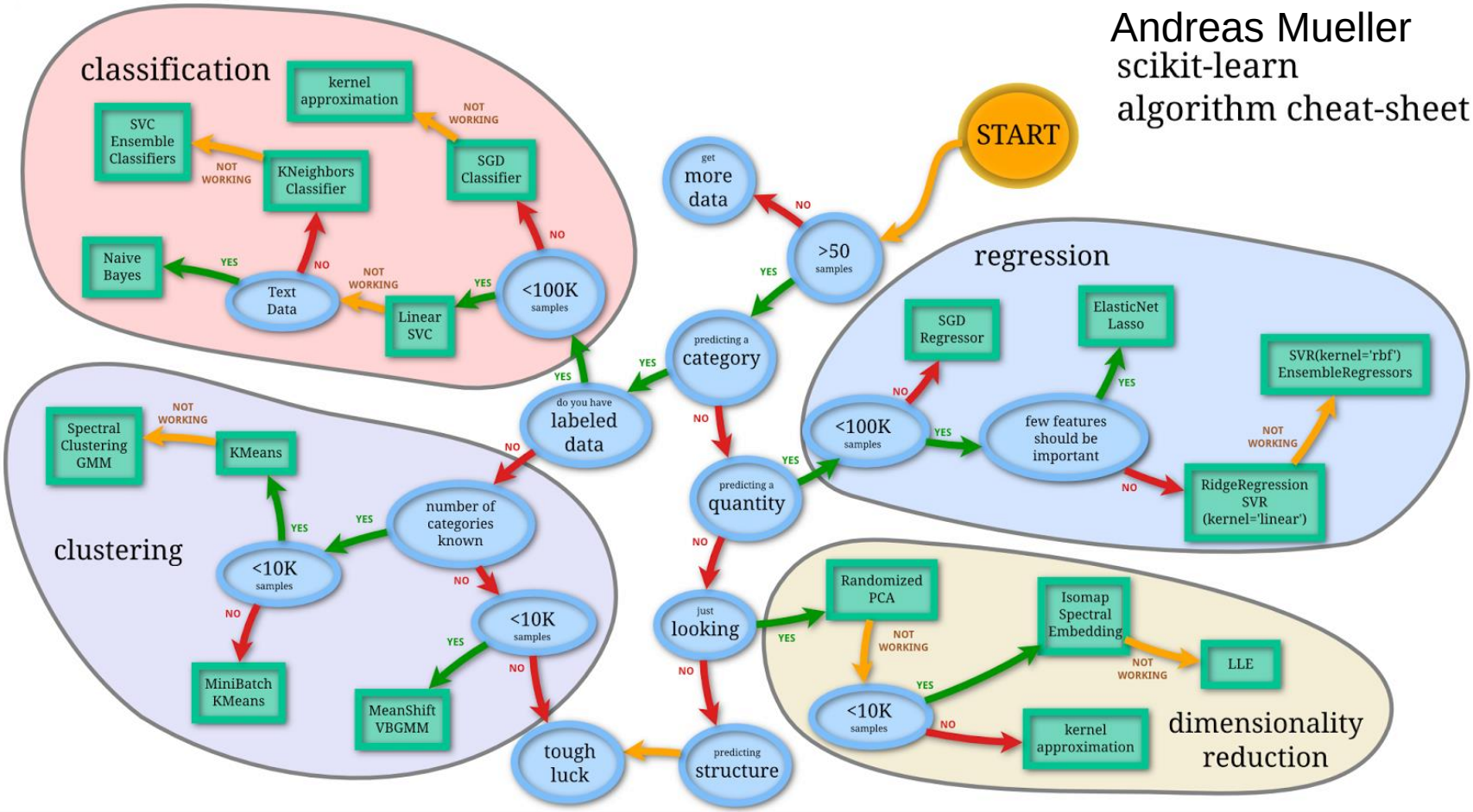
`base.BaseEstimator`	Base class for all estimators in scikit-learn.
`base.BiclusterMixin`	Mixin class for all bicluster estimators in scikit-learn.
`base.ClassifierMixin`	Mixin class for all classifiers in scikit-learn.
`base.ClusterMixin`	Mixin class for all cluster estimators in scikit-learn.
`base.DensityMixin`	Mixin class for all density estimators in scikit-learn.
`base.RegressorMixin`	Mixin class for all regression estimators in scikit-learn.
`base.TransformerMixin`	Mixin class for all transformers in scikit-learn.
`base.OneToOneFeatureMixin`	Provides `get_feature_names_out` for simple transformers.
`base.ClassNamePrefixFeaturesOutMixin`	Mixin class for transformers that generate their own names by prefixing.
`feature_selection.SelectorMixin`	Transformer mixin that performs feature selection given a support mask

Functions

`base.clone`(estimator, *[, safe])	Construct a new unfitted estimator with the same parameters.
`base.is_classifier`(estimator)	Return True if the given estimator is (probably) a classifier.
`base.is_regressor`(estimator)	Return True if the given estimator is (probably) a regressor.
`config_context`(*[, assume_finite, ...])	Context manager for global scikit-learn configuration.
`get_config`()	Retrieve current values for configuration set by `set_config`.
`set_config`([assume_finite, working_memory, ...])	Set global scikit-learn configuration
`show_versions`()	Print useful debugging information"

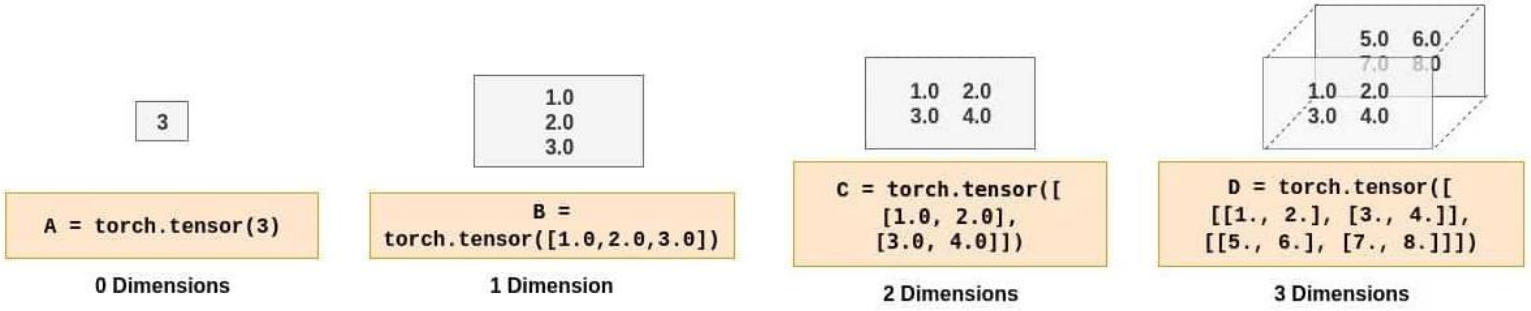
Components of scikit-learn

- Supervised learning algorithms
  (the spread of machine learning algorithms: Generalized linear models, Support Vector Machines, Decision Trees, Bayesian methods, Feature selection…)
- Cross-validation
  (various methods to check the accuracy of supervised models)
- Unsupervised learning algorithms
  (starting from clustering, factor analysis, principal component analysis to unsupervised neural networks)
- Various datasets
  (IRIS dataset, Boston House prices dataset)
- Feature extraction
  (extracting features from images and text, e.g. Bag of words)


Andreas Mueller
scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
classification
kernel approximation
NOT WORKING
SVC Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
Linear SVC
YES
<100K samples
NO
YES
predicting a category
YES
NO
do you have labeled data
NO
YES
regression
SGD Regressor
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
NO
YES
<100K samples
YES
few features should be important
NOT WORKING
NO
RidgeRegression SVR (kernel='linear')
predicting a quantity
YES
NO
clustering
Spectral Clustering GMM
NOT WORKING
KMeans
YES
YES
number of categories known
<10K samples
NO
NO
<10K samples
YES
MiniBatch KMeans
MeanShift VBGMM
NO
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure

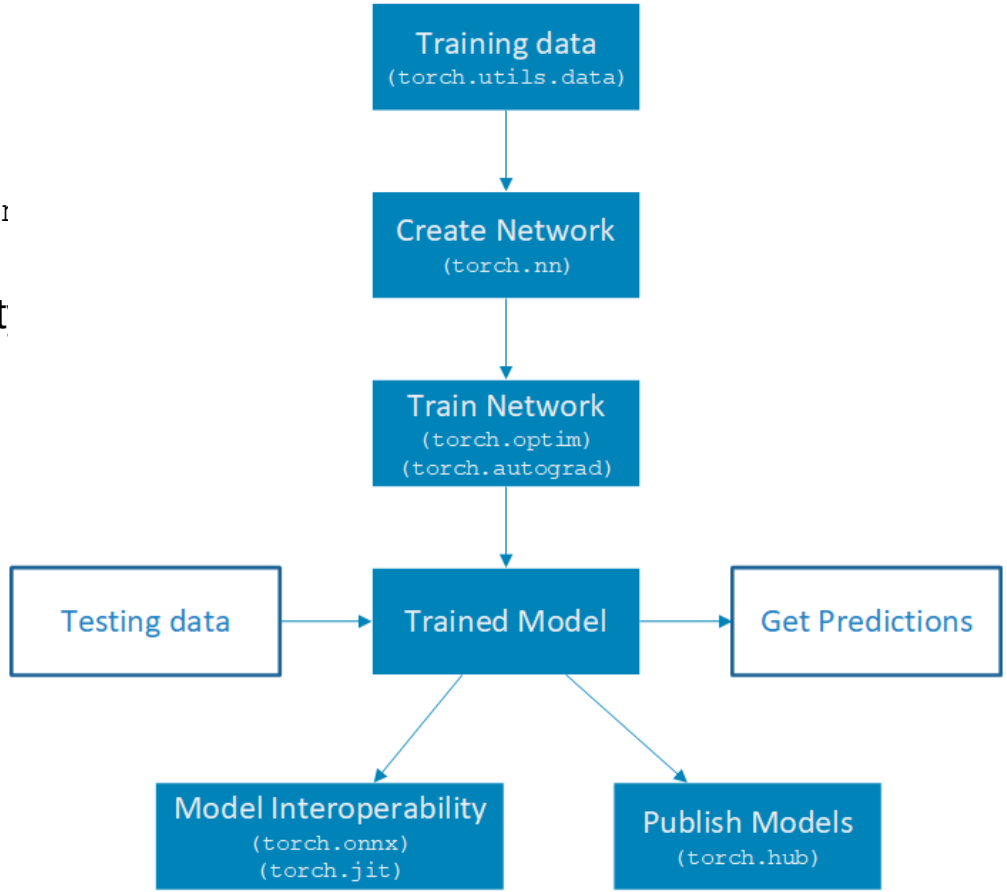
PyTorch

- Machine learning framework based on the Torch library
- Originally developed by Meta AI, now part of the Linux Foundation
- It is free and open-source software released under the modified BSD license https://pytorch.org/
- Defines a class called Tensor (`torch.Tensor`) to store and operate on homogeneous multidimensional rectangular arrays of numbers



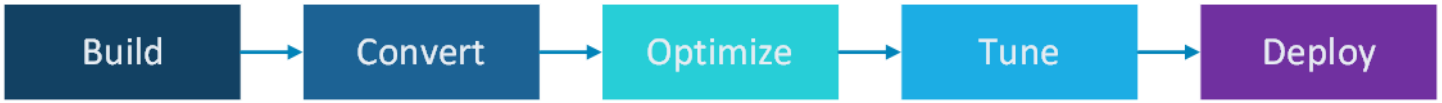
The basic pipeline of a PyTorch project

- Data Loading and Handling (`torch.utils.data`)
- Building Neural Network (`torch.n`, `torch.optim`, `torch.autograd`)
- Model Inference and Compatibilit (`torch.onnx`, `torch.hub`)



OpenVINO

- Open Visual Inferencing and Neural Network Optimization
- Open-source toolkit for optimizing and deploying AI inference
- Created and developed by Intel
- Can be used on Intel CPU, GPU, VPU or FPGA



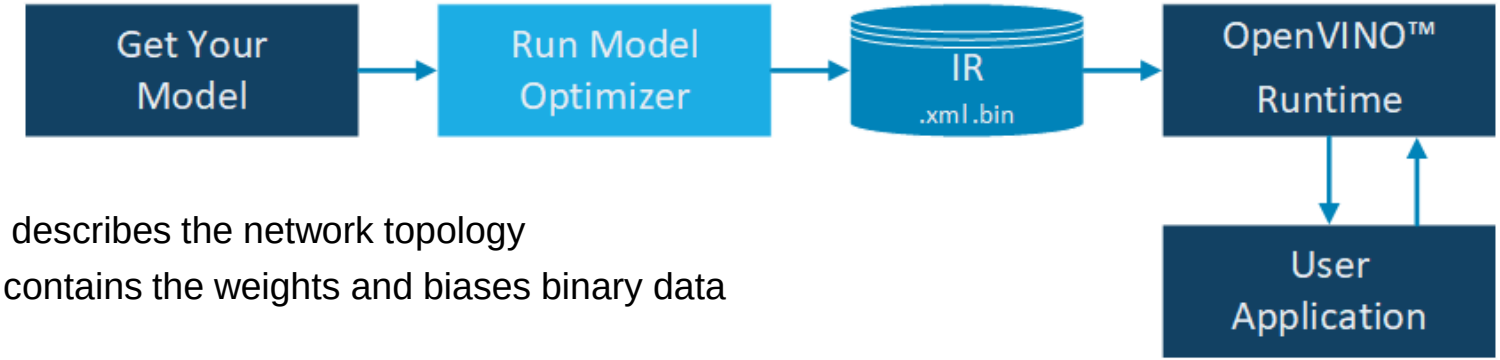
OpenVINO™

- Model preparation
  - Prepare a custom model
  - Adjust a ready-made solution
  - Run a pre-trained network from Model Zoo
- Supported frameworks
  - ONNX
  - PaddlePaddle
  - TensorFlow*
  - PyTorch*
  - MXNet*
  - Caffe*
  - Kladi*

* formats supported indirectly, need to be converted to OpenVINO IR before inference

Workflow for deploying a trained deep learning model

- Configure Model Optimizer for the specific framework (used to train your model)
- Run Model Optimizer to produce an Intermediate Representation (IR) of the model based on the trained network topology, weights and biases values, and other optional parameters
- Test the model in the IR format using the Inference Engine in the target environment
- Integrate Inference Engine in your application to deploy the model in the target environment



- .xml - describes the network topology
- .bin - contains the weights and biases binary data

Model Optimizer

- Cross-platform command line tool
- Facilitates the transition between the training and deployment environment
- Converts the model to the OpenVINO Intermediate Representation format (IR)
- Operations:
  - Reshaping – reshape input
  - Modifying the Network Structure – for example remove Dropout layers
  - Standardizing and Scaling – perform operations like normalization
  - Quantization – change datatypes used by model

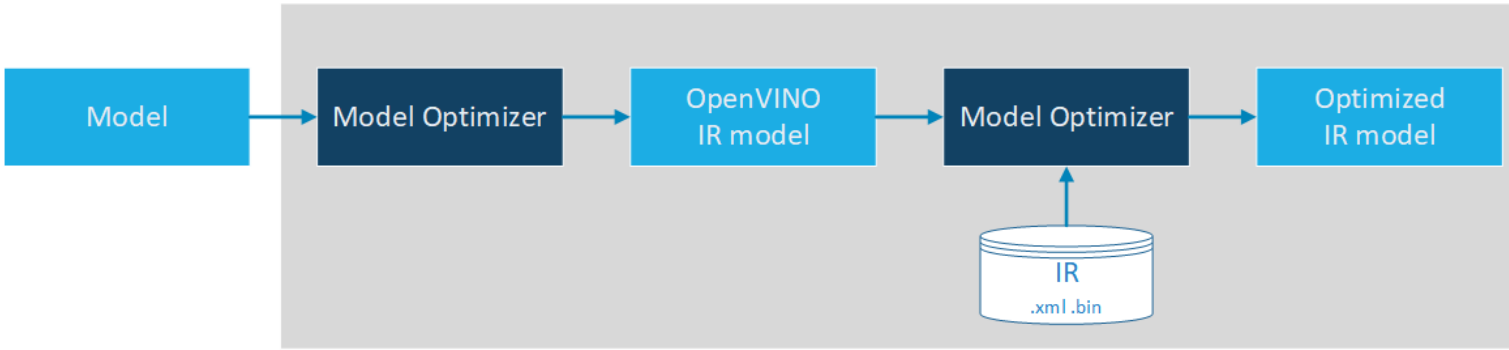
Intermediate Representation (IR)

- Intermediate representation describing a deep learning model plays an important role connecting the OpenVINO toolkit components
- The IR is a pair of files:
  - The topology file - an XML file that describes the network topology
  - The trained data file - a .bin file that contains the weights and biases binary data

- Intermediate Representation (IR) files can be read, loaded and inferred with the Inference Engine. Inference Engine offers a unified API across a number of supported Intel platforms.
- IR can be additionally optimized for inference by Post-training optimization (POT) that applies post-training quantization methods

- Quantizing Models with POT
  - supports the uniform integer quantization method
  - allows moving from floating-point precision to integer precision during the inference time
  - calibrates the network based on the resulting weights and activation values

- After post-training quantization model run faster and take less memory, (it may cause a reduction in accuracy)
- Try using QAT (Quantization-aware Training) to increase accuracy



- Supported hardware (Intel)
  - CPU (Xeon with AVX2, Atom with SSE)
  - GPU (HD Graphics, Iris Graphics)
  - VPU (Movidius Myriad)
  - GNA (Gaussian neural accelerator)

- Supported model format

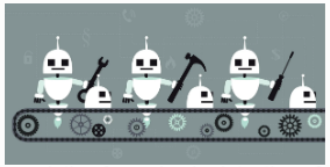
Plugin	FP32	FP16	I8
CPU plugin	Supported and preferred	Supported	Supported
GPU plugin	Supported	Supported and preferred	Supported
VPU plugins	Not supported	Supported	Not supported
GNA plugin	Supported	Supported	Not supported
Arm® CPU plugin	Supported and preferred	Supported	Supported (partially)

https://www.mouser.mx/blog/object-size-detection-openvino-toolkit

OpenVINO - examples

Object Size Detection with the OpenVINO™ Toolkit

En agosto 26, 2019 en All, Open Source por M. Tim Jones



Traditional methods of defect detection faced a number of challenges that reduced the quality of the process. Applying deep learning algorithms to captured video information increases the speed and accuracy of identifying objects that do not meet a predefined standard. Though deep learning is a relatively new solution for defect detection, it can expand the scope of the solution from simple detection of a defect to classification of the type of defect. Training deep learning networks to identify types of defects makes it possible to automatically route objects based upon their severity—such as the size of the flaw. In this example of the Intel® OpenVINO™ toolkit, we will look at a simple example of how video images can be used to determine whether an object is defective based upon its surface area.

Object Size Detection Pipeline

In prior blog posts, we've seen examples of face and vehicle detection using images captured by a video camera. In this application, we'll look at a different type of detection using deep learning to identify an object on a conveyor belt, measure its surface area, and check for defects.

Figure 1 shows the Object Size Detection pipeline. Let's explore this pipeline and the activities that occur.

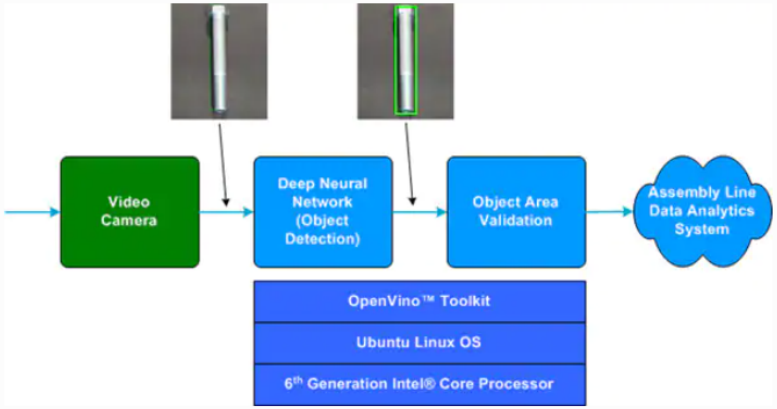


***Figure 1**: The Object Size Detection Pipeline diagram illustrates how this application of the OpenVINO™ toolkit processes an image to determine whether an object has a defect based on its surface area. (Source: Author)*

This image processing application uses images captured by a video camera mounted above a conveyor belt. A Convolutional Neural Network (CNN)—a type of image processing deep neural network—processes the captured images to determine if an object is present. First, the CNN identifies whether an object is in the capture frame. If an object is present, the CNN draws a bounding box and calculates the area that object occupies. Then, this area is checked against the predefined acceptable constraint. If the object is larger or smaller than expected, then a defect indication is communicated.

Figure 2 shows the output of the Object Size Detection application of the OpenVINO™ toolkit. Note that in this example, the CNN found the object and bounded it in order to calculate its area.

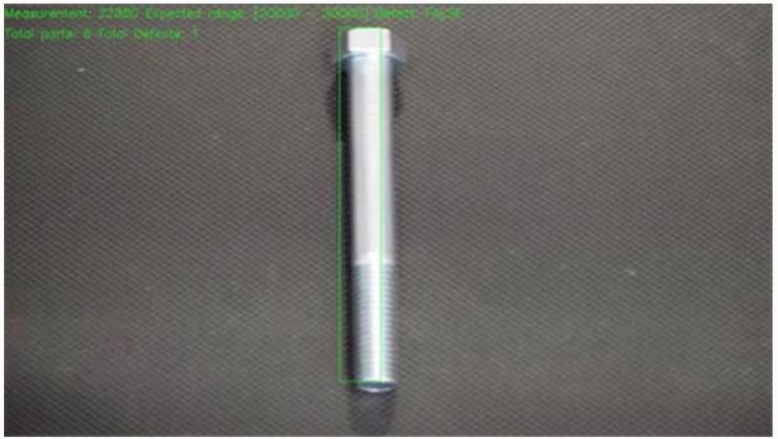


***Figure 2**: The Object Size Detector output screen shows an example of the calculated area of a detected object. (Source: Intel)*

The sample application also illustrates the use of the Message Queue Telemetry Transport (MQTT) protocol, which communicates the zone information to an industrial data analytics system.

Related topics

https://www.tensorflow.org/guide

https://scikit-learn.org/stable/user_guide.html

Post-Training Quantization Best Practices
https://docs.openvino.ai/latest/pot_docs_BestPractices.html#doxid-pot-docs-best-practices

Movidius Myriad X VPU
https://www.aaeon.ai/eu/product/detail/ai-core-x

Introduction to Torch’s tensor
https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html