

# 初めてのPythonプログラミングと インテル® OpenVINO™

2021年4月23日 (金)

菱洋エレクトロ株式会社

Ver2.0 2021/4/9  
Ver4.0 2021/4/14  
Ver6.0 2021/4/15  
Ver6.4 2021/4/19  
Ver10.0 2021/4/21

## 経歴

メーカーで通信系システム（固定電話網からイーサーネット）の研究&開発

- ASIC、FPGAでのハードウェア開発経験は20年以上（RTL言語が得意）

外資系に転職

- FPGAのFAEとして携帯電話LTEの基地局開発（ハードウェア）をサポート

上記を経て、商社、菱洋エレクトロに入社。通信系デバイスのサポート。現在はIntelのテクノロジーを習得中

## Pythonを学習しようと思ったきっかけ

自分で作ったPythonコードで、初めてOpenVinoが動いた時はものすごく感動しました。

もっとプログラムを効率的に、もっと見やすくしたいと思ったのが勉強するきっかけ。

## 趣味

コロナ禍で運動不足を解消するためにロードバイクを購入。

半年で5kgのダイエットに成功、さらに禁煙も成功（2020/9月から）



1. OpenVinoとは
2. 初めてのPython
3. Python基本命令
4. Pythonオブジェクト指向
5. Pythonライブラリ
6. OpenVino Python API
7. サンプルコード解説

# OpenVinoとは

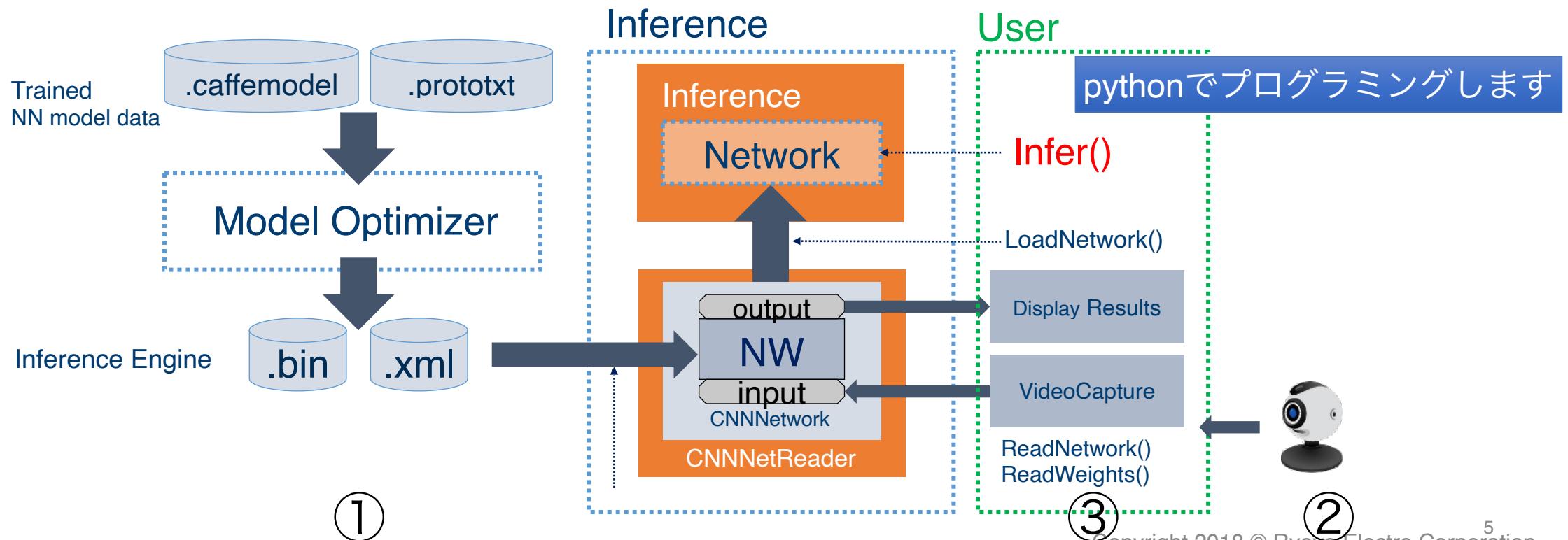
# 1. OpenVINOとは

## OpenVinoとは

Intelが提供するディープラーニング開発ツールです。Intelデバイス上で推論実行する事ができます。

## 推論実行の流れ

- ①推論モデルを用意します。
- ②推論させたいデータ、素材を用意します。（画像、動画、カメラ、音声 等）
- ③pythonで推論できるようにプログラミングします。くーこれについての解説となります。



# 初めてのpython

### Pythonとは

オブジェクト指向のスクリプト言語です。 (コンパイル不要) \*詳しい説明については、別章でも説明します。

### Pythonの特徴

文法がシンプルで読み易い、書きやすい。

ライブラリが豊富。

### Pythonの活躍現場

AI/DL、Webアプリケーション、データ分析、解析、ビックデータ対応、画像処理、画像解析、ブロックチェーンなど幅広い現場で活躍

### 今回使用するソフトウェアのバージョン

OpenVINO 2021.2.185 / Anaconda(Python3.8.5) \*1 / OpenCV 4.5.1-opencvino / Windows10

\*1 Anaconda インストール方法参照  
<https://www.python.jp/install/anaconda/windows/install.html>

### Pythonの動かし方（Anaconda3プロンプト上で実行します）

対話モード：

Anaconda3プロンプト上で”python”と入力します。>>>が出力されると入力待ちとなります。

```
(base) C:\$>
(base) C:\$>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

スクリプトファイルの動かし方：

Anaconda3プロンプト上で”python \*\*\*.py”と入力します。

```
(base) C:\$openvino_2021_2_original>python hello.py
Hello Python
```

```
⑧(base) C:\$openvino_2021_2_original>
```

本セミナではスクリプトファイルでプログラミングします。

# Python 基本構文

## 2. Python基本構文

最低限使用する構文は5つだけ

### 1. プリント表示

```
print("Hello World")
print(5+2)
print(5*2)
print(5/2)
print(5-2)
```

```
Hello World
7
10
2.5
3
```

### 2. 変数定義

```
a = 5
b = 2
print("a + b =",a+b)
```

```
a+b=7
```

### 3. for文

test変数が1から5未満である場合は  
ブロック内を処理します。  
5以上になると処理を抜けます。

```
for test in range(1,5):
    print(test)
```

```
1
2
3
4
```

### 4. if分

old変数は18です。old変数が20未満であれば、  
ブロック内を処理します。  
それ以外の場合はelseのブロックを処理します。

```
old = 18
if old <20:
    print("you can not drink",old)
else:
    print("you can drink",old)
```

```
you can not drink 18
```

### 5. while文

Aは変数です。Aが以内である間は、  
ブロック内を処理します。  
5以上になると処理を抜けます。

```
A=3
while(A<=7):
    A = A+1
    print("A==",A)
```

```
A= 4
A= 5
A= 6
A= 7
A= 8
```

## 2. Python基本構文

Python言語を記述する上で、ブロック文とインデントルールについて

### ブロック文とは

For / IF / While などです。このヘッダは条件式を記述して、コロンで締めます。

1 ライン目に条件定義がされてます。その条件が当てはまった時に処理内容が実施されます。下図参照

### インデントルールとは

ブロック文 (For / IF / While)の場合、処理内容部分をインデント（字下げ）します。

ブロック文内ではインデント量はプログラム内では全て統一しなければなりません。

例 スペースキーで4つ、もしくはTabキー \*どちらかに統一しておく事をお勧めします

ブロック文

For / IF / While 条件式:

処理内容

インデントが必要

ネストした場合もインデント必要

```
4 for test in range(1,5):
5     print("test=",test)
6     for test2 in range(1,3):
7         print("test2=",test2)
8
```

# Python オブジェクト指向プログラミング

### 3. Python オブジェクト指向

#### オブジェクトとは

オブジェクトにはデータ（変数）とメソッド（関数）の両方が含まれたものです。

\*メソッド（関数）：方法、方式

#### オブジェクト指向とは

オブジェクトを作るためのお作法です。

クラスを定義します。

オブジェクトにクラスをインスタンス化します。また変数を定義することもできます。

これにより具体的なオブジェクトが生成されます。

・・・概念図をもとに説明します。

インスタンス化して生成されたものを、本セミナではオブジェクトと呼称するようにいたします。

#### 概念図

例：カレーというオブジェクトを作ります。

方法は・・・？ 調理のクラスを定義します。変数として食材、メソッド（関数）として、定義します。

→ 肉、野菜（変数）などを与え、煮るメソッド（関数）を呼び出すことで、カレーが出来上がります。

変数：データを入れておく箱  
引数：処理をする中で値を代入し、結果を出力する場合に使うパラメータ

#### 調理のクラス

肉	データ（変数）
野菜	データ（変数）
煮る	メソッド（関数）
炒める	メソッド（関数）
茹でる	メソッド（関数）

#### オブジェクトを作る為に（調理のクラスを元に作成）



インスタンス化



つまり。。。オブジェクト=クラス（引数）

### 3. Python オブジェクト指向

pythonでオブジェクト思考プログラミングする上で、重要な6ルールだけ抜き出しました。 **赤文字**

- ① **class** : オブジェクトを生成する上で使われる”型”のようなもの。 ”型”を定義します。 変数とメソッドが含まれます。
- ② **def \_\_init\_\_** : インスタンス生成時、自動的に必ず呼び出される初期化メソッドです。
- ③ **def \*\*\*\*\***: ある処理をまとめた固まりに名前をつけたものです。 メソッドといいます。 幾つでも定義できます。
- ④ **self** : クラスの中のメソッドはデフォルトで引数を最低1つ与えておく必要がある。 → **self**(任意ですが、ないとエラー)  
またクラス変数です。 クラス内で変数として扱えます。
- ⑤ **インスタンス化する** : クラスを元にオブジェクトが生成されます。 オブジェクト=**クラス**.(引数)
- ⑥ **メソッドの呼び出し** : インスタンス化された中のDef (メソッド) を処理します。 変数=オブジェクト.**メソッド** (引数)

Calという型

```
① class Cal:  
②     def __init__(self, a, b):  
③         self.a = a  
④         self.b = b  
⑤     def Sum(self):  
⑥         self.sum = self.a + self.b ④  
⑦         return self.sum
```

⑤ ins = Cal(3,4) オブジェクト=**クラス** (引数)  
⑥ A =ins.Sum() 変数=オブジェクト.**メソッド** (引数)  
print(A)

実行結果

C:> python test.py

7

### 3. Python オブジェクト指向

#### オブジェクト指向プログラミングの流れ

① クラスを定義します

例 Calを定義します。また変数とメソッドを定義します

② オブジェクトにクラスをインスタンス化します（オブジェクトを生成）引数も与えることができます。

例 insというオブジェクト。初期化メソッドが実行

③ 生成されたオブジェクトを扱ってメソッドを呼び出して、プログラミングします。

例 ins.Sumメソッドを実行

①

Calというクラス

オブジェクト=クラス（引数）

②

インスタンス化、insオブジェクト  
にa,b引数を与えます

③ クラス insのSumメソッドを実行し、Print表示

変数=オブジェクト.メソッド（引数）

```
class Cal:  
    def __init__(self, a, b):  
        self.a = a  
        self.b = b  
    def Sum(self):  
        self.sum = self.a + self.b  
        return self.sum
```

初期化メソッド

メソッド

```
ins = Cal(3,4)  
A = ins.Sum()  
print(A)
```

実行結果

C:> python test.py

# Python 一般ライブラリ

## ライブラリとは

ある目的のために**機能をまとめたパッケージ**の総称を示します。

Pythonのライブラリは他の言語に比べて、圧倒的に高機能で実用的なものが準備されています。

## ライブラリのメリット

高度な機能が単純に使用できるようになります。

ライブラリを適切に使用すれば、プログラムの構造、作り自体が似た作りになります。

ここでもオブジェクト指向の考え方について  
**結果=クラス.メソッド（引数）**

## ライブラリの種類

標準ライブラリ：Pythonインストール後、すぐに使用できます。

拡張ライブラリ：追加でインストールが必要となります。 \*(OpenCV, Numpyなど)

\* Windowsの場合、Anaconda3でまとめてインストール可能です。

## ライブラリの読み込み方法

プログラムの冒頭に必ず記述してください。

`import モジュール名 or パッケージ` →モジュールまたはパッケージの全体を利用します。

`from パッケージ import モジュール` →モジュールの一部を利用します。

## cv2とは

cv2 = OpenCV とは、Open Source Computer Vision Libraryの略です。画像や動画を処理するための機能がまとめて実装されています。

## 使用目的

学習済みモデルは、入力できるイメージ条件が定義されます。＊推論モデルで異なります。

本ライブラリを使用することで、**画像、動画を加工**して、学習済みモデルの入力条件に合わせます。

## OpenCVライブラリの機能(メソッド) 一部掲載

画像の読み込み、表示

画像の作成、保存

画像のトリミング、リサイズ、重ね合わせ

画像の回転、上下反転、左右反転

グレースケール変換、色チャネル分析、減色処理

モザイク処理、マスク書き、2枚の画像を合成

**図形の描画、文字の描画**

ノイズ除去、平滑化、ぼかしフィルタ、メディアンフィルタ

物体検出

テンプレートマッチング

\*太字の機能などは本サンプルプログラムでも使用しています。

結果取得=CV2.メソッド (引数)

本ライブラリーの読み込みと使用方法例

```
import cv2
```

```
frame=cv2.resize(file_name, (width, hight))
```

ファイルのリサイズ



file\_name

加工後



frame

## Numpyとは

Numpyとは高度な数値演算を高速に処理できます。多次元配列のデータも扱えます。

## 使用目的

学習済みモデル（IR）から出力される**推論結果は多次元のデータ配列**を持ったものがほとんどです。

本ライブラリを使用することで、**推論結果を人が見やすいように加工**します。 \*推論モデル毎にOutputは異なります

例 多次元配列のデータから、特定の配列のデータ部分だけを取り出す

## numpyライブラリーの機能(メソッド) は以下です。\*一例

ベクトルや行列の演算

次元を持ったデータの操作

乱数生成

合計値、最大値、最小値

平均値、中央値、標準偏差

配列の操作

条件を満たす要素だけ処理

行列の軸 (axis) の入れ替え

\*太字の機能などは本サンプルプログラムでも使用しています。

結果取得=np.メソッド (引数)

### 本ライブラリーの読み込みと使用方法例

```
import numpy as np
```

#推論結果 (加工前)

```
res = {'age_conv3': array([[[[0.26797116]]], dtype=float32), 'prob': array([[[[0.01605477],  
[[0.9839452 ]]]], dtype=float32)}
```

#次元resデータからage\_conv3次元の要素データを抜き出す

```
age= np.squeeze(res['age_conv3'])  
age<- 0.26797116 (加工後)
```

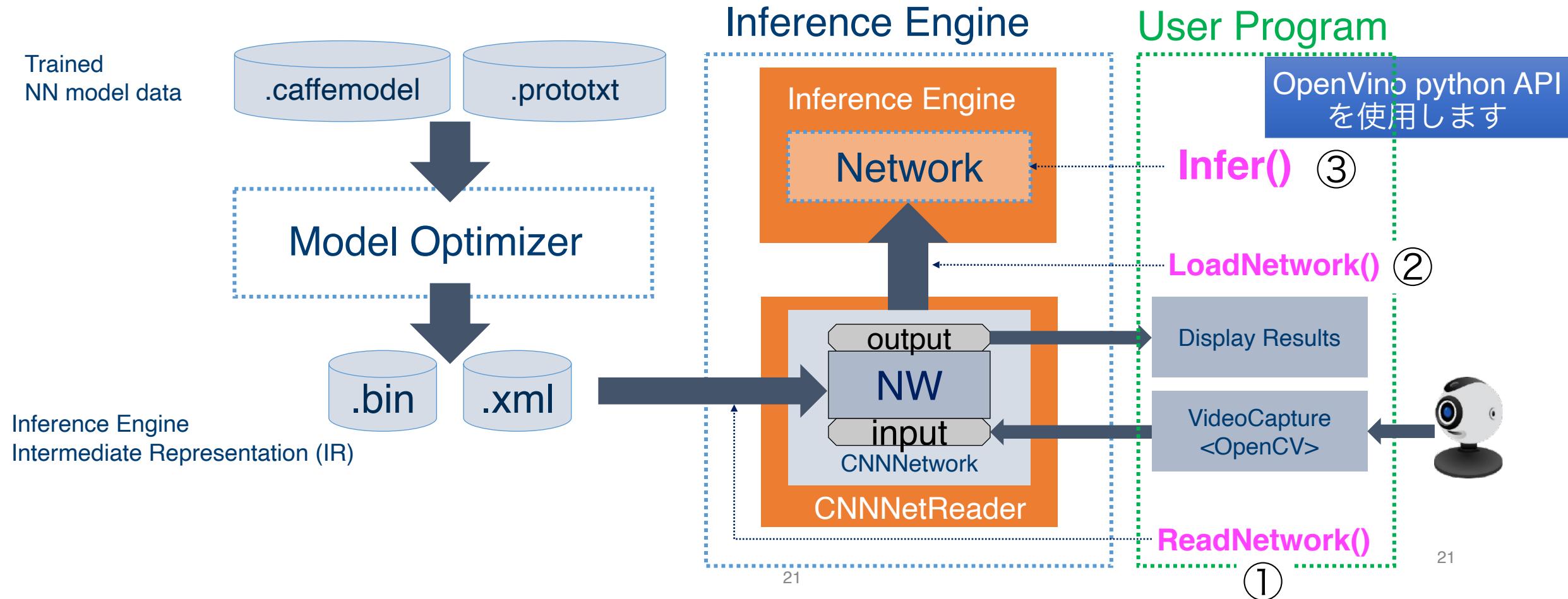
# OpenVino Python API

# 6. OpenVINO推論のプログラム

UserプログラムでOpenVino Python APIを呼び出して、推論モデルを動作させます。

ここでもオブジェクト指向の考え方について  
結果=クラス.メソッド(引数)

- ① ReadNetwork : 推論モデル (IR) のパスを指定してネットワークを構築します。
- ② LoadNetwork : 推論モデル (IR) を使用するプラグインに構築します \*CPU,GPU,MYRIAD,FPGA,HDDL
- ③ Infer : 推論実行します



## 6. OpenVino Python API

OpenVino Pythonで用意されるクラス（API）は以下です。最低限必要なのは下記の2つです。

\*バージョンによりAPIが異なります。最新版は以下のサイトをに参照ください。

[https://docs.openvino-toolkit.org/2021.2/ie\\_python\\_api/annotated.html](https://docs.openvino-toolkit.org/2021.2/ie_python_api/annotated.html) \*OpenVino2021.R2のリンク

▼ N ie_api	
C Blob	This class represents Blob
C CDataPtr	This class is the layer constant data representation
C DataPtr	This class is the layer data representation
C ExecutableNetwork	This class represents a network instance loaded to plugin and ready for inference
C IECore	This class represents an Inference Engine entity and allows you to manipulate with plugins using unified interfaces
C IENetwork	This class contains the information about the network model read from IR and allows you to manipulate with some model parameters such as layers affinity and output layers
C InferRequest	This class provides an interface to infer requests of ExecutableNetwork and serves to handle infer requests execution and to set and get output data
C InputInfoCPtr	This class contains const information about each input of the network
C InputInfoPtr	This class contains information about each input of the network
C PreProcessInfo	This class stores pre-process information for the input
C TensorDesc	This class defines Tensor description

## ie\_api.IECore Class Reference

本クラスで使用しているメソッドは下記です。使用方法は次ページ

### Public Member Functions

def `_init_`(self, xml\_config\_file)

Class constructor. [More...](#)

def `get_versions`(self, device\_name)

Get a `namedtuple` object with versions of the plugin specified. [More...](#)

def `read_network`(self, model, weights, init\_from\_buffer)

Reads a network from Intermediate Representation (IR) or ONNX formats and creates an `IENetwork`.  
[More...](#)

def `load_network`(self, network, device\_name, config=None, num\_requests=1)

Loads a network that was read from the Intermediate Representation (IR) to the plugin with specified device name and creates an `ExecutableNetwork` object of the `IENetwork` class. [More...](#)

def `import_network`(self, model\_file, device\_name, config=None, num\_requests=1)

Creates an executable network from a previously exported network. [More...](#)

def `query_network`(self, network, device\_name, config=None)

Queries the plugin with specified device name what network layers are supported in the current configuration. [More...](#)

IECoreクラスのメソッド一覧

# 6. OpenVino Python API

## IETCoreクラス.メソッド

このメソッドの説明、使用方法例が記述されています。

### • `read_network()`

```
def ie_api.IETCore.read_network
```

推論モデルのパスを指定します

```
( self,  
    model,  
    weights,  
    init_from_buffer  
)
```

#### 説明

Reads a network from Intermediate Representation (IR) or ONNX formats and creates an `IENetwork`.

#### Parameters

`model` A `.xml`, `.onnx` or `.prototxt` model file or string with IR.

`weights` A `.bin` file of the IR. Depending on `init_from_buffer` value, can be a string path or bytes with file content.

`init_from_buffer` Defines the way of how `model` and `weights` attributes are interpreted. If `False`, attributes are interpreted as strings with paths to `.xml` and `.bin` files of IR. If `True`, they are interpreted as Python `bytes` object with `.xml` and `.bin` files content.

#### 使用方法

object

Usage example:

```
1 ie = IETCore()  
2 net = ie.read_network(model=path_to_xml_file, weights=path_to_bin_file)
```

推論モデル (IR).xml/.binのパス

結果取得=ie.メソッド (引数)

### • `load_network()`

```
def ie_api.IETCore.load_network
```

プラグインに推論モデルを構築します

```
( self,  
    network,  
    device_name,  
    config =None,  
    num_requests =1  
)
```

Loads a network that was read from the Intermediate Representation (IR) to the plugin with specified device name and creates an `ExecutableNetwork` object of the `IENetwork` class.

You can create as many networks as you need and use them simultaneously (up to the limitation of the hardware resources).

#### Parameters

`network` A valid `IENetwork` instance

`device_name` A device name of a target plugin

`config` A dictionary of plugin configuration keys and their values

`num_requests` A positive integer value of infer requests to be created. Number of infer requests is limited by device capabilities. Value `0` indicates that optimal number of infer requests will be created.

#### Returns

An `ExecutableNetwork` object

Usage example:

```
1 ie = IETCore()  
2 net = ie.read_network(model=path_to_xml_file, weights=path_to_bin_file)  
3 exec_net = ie.load_network(network=net, device_name="CPU", num_requests=2)
```

## ie api.InferRequest Class Reference

本クラスで使用しているメソッドは下記です。使用方法は次ページ

### Public Member Functions

def `_init_`(self)

There is no explicit class constructor. [More...](#)

def `set_completion_callback`(self, py\_callback, py\_data=None)

Description: Sets a callback function that is called on success or failure of an asynchronous request.  
[More...](#)

def `input_blobs`(self)

Dictionary that maps input layer names to corresponding Blobs.

def `output_blobs`(self)

Dictionary that maps output layer names to corresponding Blobs.

def `preprocess_info`(self)

Dictionary that maps input layer names to corresponding preprocessing information.

def `set_blob`(self, blob\_name, blob, `preprocess_info`)

Sets user defined [Blob](#) for the infer request. [More...](#)

def `infer`(self, `inputs`=None)

Starts synchronous inference of the infer request and fill outputs array. [More...](#)

def `async_infer`(self, `inputs`=None)

Starts asynchronous inference of the infer request and fill outputs array. [More...](#)

def `wait`(self, timeout=None)

Infer requestクラスのメソッド一覧

## ie\_api.InferRequest.メソッド

このメソッドの説明、使用方法例が記述されています。下記のどちらかのAPIで推論を実行します。

### ◆ infer()

```
def ie_api.InferRequest.infer  
    ( self,  
      inputs = None  
    )
```

Starts synchronous inference of the infer request and fill outputs array.

#### Parameters

**inputs** A dictionary that maps input layer names to `numpy.ndarray` objects of proper shape with input data for the layer

#### Returns

None

Usage example:

```
1 exec_net = ie_core.load_network(network=net, device_name="CPU", num_requests=2)  
2 exec_net.requests[0].infer({input_blob: image})  
3 res = exec_net.requests[0].output_blobs['prob']  
4 np.flip(np.sort(np.squeeze(res)),0)  
5 array([4.85416055e-01, 1.70385033e-01, 1.21873841e-01, 1.18894853e-01,  
6      5.45198545e-02, 2.44456064e-02, 5.41366823e-03, 3.42589128e-03,  
7      2.26027006e-03, 2.12283316e-03 ...])
```

### 推論同期実行

### ◆ async\_infer()

```
def ie_api.InferRequest.async_infer  
    ( self,  
      inputs = None  
    )
```

Starts asynchronous inference of the infer request and fill outputs array.

#### Parameters

**inputs** A dictionary that maps input layer names to `numpy.ndarray` objects of proper shape with input data for the layer

#### Returns

:None

Usage example:

```
1 exec_net = ie_core.load_network(network=net, device_name="CPU", num_requests=2)  
2 exec_net.requests[0].async_infer({input_blob: image})  
3 request_status = exec_net.requests[0].wait()  
4 res = exec_net.requests[0].output_blobs['prob']
```

### 推論非同期実行

推論実行時に、推論させたいデータを指定します。

# サンプルコード解説

### ① top\_squeeze.py

画像データを扱って推論します。

### ② top\_road\_detection.py

動画データを扱って推論します。

### ③ top\_head.py

Webカメラデータ（リアルタイム）を扱って推論します。

## Squeeze1.1

学習済みモデル：SqueezeNet1.1の学習済みモデルを使います。

概要：入力された画像データから、物体を検出します。その物体は1000のカテゴリで分類されます。

入力データ：右図上参照（1枚、3原色、サイズは227x227）

出力データ：右図下参照（1-1000の精度を全て）

## モデルの詳細 (Intel Open Model Zoo)

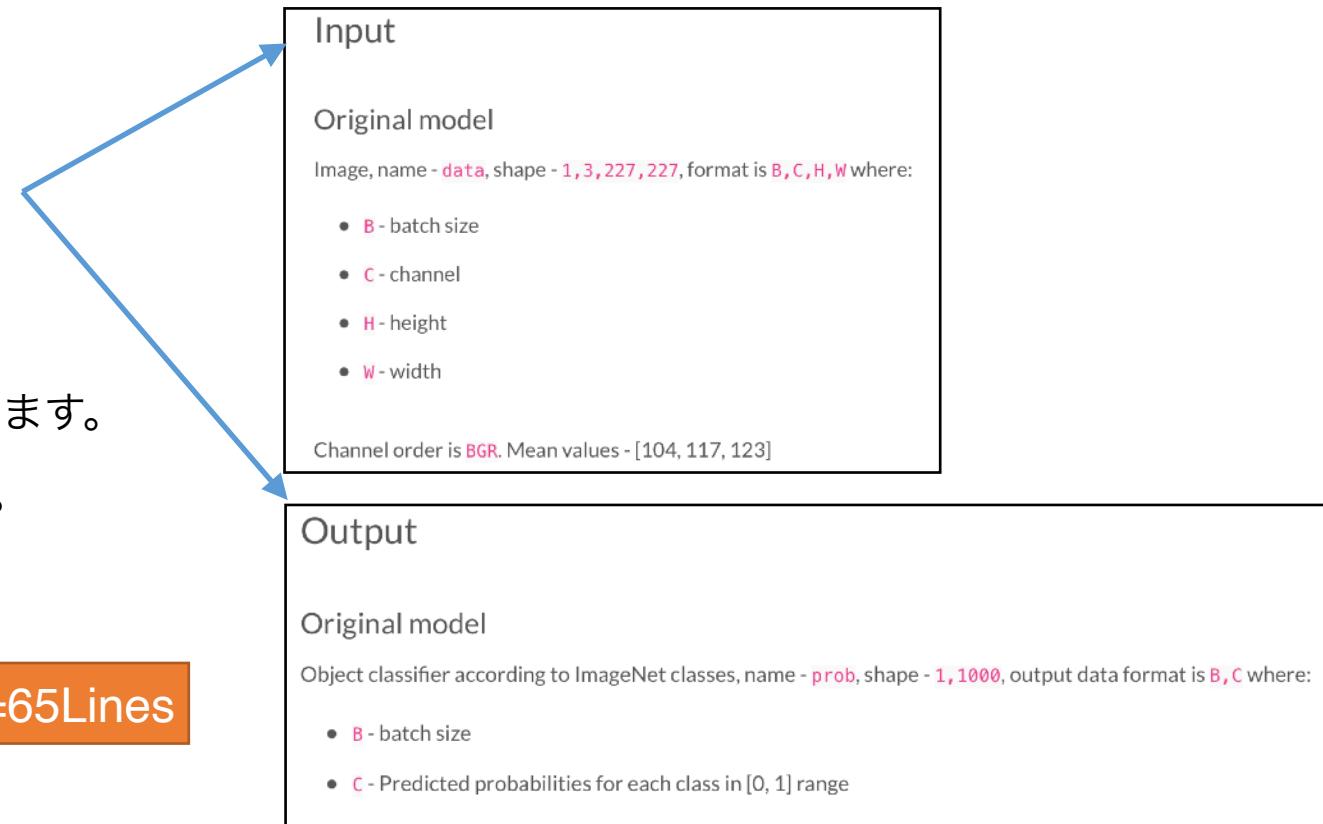
[squeezeNet1.1\\_intel\\_open\\_model\\_zoo](#)

## サンプルプログラム

2枚の画像データ（カップ、車）を順番に推論を実施します。

結果をソートして、精度が高いTop 5をPrint表示します。

プログラム=65Lines



# 7. サンプルコード解説 ① top\_squeeze.py

## サンプルプログラム概要

Import ライブラリー

```
ie = IECORE()
```

使用するライブラリーのimport

```
class squeezenet():
    def __init__(self,__image):
        return
    def squeeze_infer(self):
        return res
```

推論実行する**class** の定義

```
class squeez_label_init():
    def __init__(self):
        return
    def squeeze_label(__label,res):
        return
```

推論結果を見やすく加工する**class** の定義

```
if __name__=='__main__':
    __label = "squeezenet1.1.lables"
```

Run1はcup.pngを推論実行して、結果を表示

```
#_____Run 1_____
__image ="cup.png"
th1   = squeezenet(__image)
__res = th1.squeeze_infer()
th2   = squeeze_label(__label,__res)
th21  = th2.squeeze_label()
```

```
#_____Run 2_____
__image ="car.png"
th3   = squeezenet(__image)
__res = th3.squeeze_infer()
th4   = squeeze_label(__label,__res)
th41  = th4.squeeze_label()
```

Run2はcar.pngを推論実行して、結果を表示

## 7. サンプルコード解説 ① top\_squeeze.py

本プログラムで使用するライブラリをインポートします

```
C: > openvino_2021_2_orginal > squeeze11 > top_squeeze.py
1  # _____ /////////////////////////
2  #__For Openvino by Python written by M.Inoishi
3  #__SqueezeNet1.1 program
4  #__OpenVino2021R2
5  #__Ver3.0 2021/14/Apr
6  #_____ /////////////////////////
7
8 #Step0__ Pythonライブラリー、モジュールのインポート
9 import cv2                                     #OpenCVモジュール（画像、動画ファイルの処理）
10 import numpy as np                            #Pythonでの数値計算を高速、効率化するモジュール
11 import logging as log                         #処理中におけるLogの吐き出しモジュール
12 from time import time                        #日付と時間を扱うモジュール
13 from operator import itemgetter              #リストなどから所望する要素を抽出するモジュール
14 from openvino.inference_engine import IECore   #OpenVinoを動作させるためのモジュール(IECore)
15
16 #Step1__ IECoreをieとしてインスタンス化
17 ie = IECore()                                 #インテラレンスエンジンをieとしてインスタンス化
18
```

使用するライブラリをImport

ieにIECoreクラスをインスタンス化

## 7. サンプルコード解説 ① top\_squeeze.py

クラスSqueezeです。SqueezeNetの推論モデルの準備

```
19 #Step2__ SqueezeNet1.1を使用するためのクラス
20 class squeesenet:
21     #本クラスの初期化メソッド
22     #推論モデルの構築、使用するプラグインを指定して、プラグインへロードし
23     #推論モデルのInput条件も取得
24     def __init__(self,__image):
25         #推論モデルのパスを指定
26         model = './ir_eng/squeezeNet1.1'
27         self.image = __image
28
29         #read_network(OpenVino API)で実行可能なモデルを構築
30         net = ie.read_network(model=model+'.xml', weights=model+'.bin')
31         #使用するプラグインを指定
32         __plug = 'CPU'
33         #__plug='GPU'
34         #__plug='MYRIAD'
35         #__plug='HETERO:FPGA,CPU'
36
37         #使用する推論モデルのInput条件を取得
38         print("Get_batch_size1=",net.batch_size)
39         self.input_layer = next(iter(net.input_info))
40         self.model_n, self.model_c, self.model_h, self.model_w = net.input_info[self.input_layer].input_data.shape
41         print("Input Spec = ",self.model_n,self.model_c,self.model_h,self.model_w)
42
43         #load_network(opennVino API)にて推論モデルをプラグインにロード
44         self.exec_net = ie.load_network(network=net, device_name=__plug)
45         return
```

squeezeNetクラスの定義

推論準備をする初期化メソッド

推論モデルのネットワーク構築

プラグインの指定

推論モデルのInput条件を取得

プラグインに推論モデルのネットワークをロード

## 7. サンプルコード解説 ① top\_squeeze.py

squeeze\_inferのメソッドです。推論実行します

```
48     #推論実行するメソッド
49     #用意した画像データを、推論モデルのInput条件に加工
50     #Squeezezenetの推論モデルを使用して、推論実行
51     def squeeze_infer(self):
52         #読み込んだイメージデータを加工します。cv2ライブラリを使用
53         img_face = cv2.imread(self.image)
54         in_frame = cv2.resize(img_face, (self.model_w, self.model_h))
55         in_frame = cv2.cvtColor(in_frame, cv2.COLOR_BGR2RGB)
56         in_frame = in_frame.transpose((2, 0, 1))
57         cv2.imshow('Image', img_face)
58         cv2.waitKey(2000)
59         cv2.destroyAllWindows()
60
61     #Python上で推論（インファー）同期実行するためのコマンド(OpenVino API)
62     #res = self.exec_net.infer(inputs={self.input_layer:in_frame})
63
64     #Python上で推論（インファー）非同期実行するためのコマンド(OpenVino API)
65     self.exec_net.requests[0].async_infer({self.input_layer: in_frame})
66     request_status = self.exec_net.requests[0].wait(-1)
67     res = self.exec_net.requests[0].outputs[self.output_blobs]
68
69     return res
70
```

推論実行メソッド

入力データを加工

推論非同期実行

推論の結果待ち  
resに推論結果

# 7. サンプルコード解説 ① top\_squeeze.py

クラスSqueeze\_labelです。推論結果を加工します

```
71 #Step3__ squeezenetで推論した結果(res)を、見やすく加工するクラス
72 class squeeze_label:
73     #本クラスの初期化メソッド
74     def __init__(self,__label,__res):
75         self.__label = __label
76         self.__res = __res
77         self.values=[]
78         self.list=[]
79         return
80
81 #推論結果(0-999)をリスト化。精度の高い順に並び替え。Top 5を表示するメソッド
82 def squeeze_label(self):
83     #item変数に推論結果"prob"次元θのデータを追加
84     for item in self.__res['prob'][0]:
85         self.values.append(item[0][0])
86
87     #lines変数に読み込んだラベルファイルの中身を追加
88     with open(self.__label) as file:
89         lines = file.readlines()
90
91     #item変数に0-999
92     for item in range(0,999):
93         self.list.append(["{:.5f}".format(self.values[item]*1),item,lines[item].splitlines()])
94
95     #表示する結果（リスト）を上位5までを表示
96     print("__sort list__")
97     self.list.sort(key=itemgetter(0),reverse=True)
98     for cnt in range(0,5):
99         print(self.list[cnt])
100    print("__End__")
101    return
```

squeeze\_labelクラスの定義

変数の初期化メソッド

推論結果加工メソッド

推論結果を加工して、  
0-999のList作成  
確率が高い順にソート

Top 5 を表示するメソッド

List変数＊イメージです

確率	番号	ラベル
0.00142	000	tench, Tinca, tinca
0.00001	001	goldfish, Caressius
0.08020	002	great white shark
0.00245	003	tiger shark, Galeocerdo
...	...	...
0.00002	999	ear, spike, capitulum

## 7. サンプルコード解説 ① top\_squeeze.py

メインプログラムです。

インスタンス化して、各種クラス、メソッドを実行します。

```
103 #Step4__ main __ メインプログラムを実行するためのスレッド（ここからプログラムの実行が指定される）
104 if __name__=='__main__':
105     __label = "squeezenet1.1.labels"
106
107 # Run 1 #
108 __image ="cup.png" #画像データを用意
109 th1 = squeezenet(__image) #推論準備
110 __res = th1.squeeze_infer() #推論実行
111 th2 = squeeze_label(__label,__res) #推論結果の加工の準備
112 th21 = th2.squeeze_label() #メソッドをインスタンス化。推論結果Top5を表示して終了。戻り値なし
113
114 print("_____") #推論結果のTop5を表示
115
116 # Run 2 #
117 __image ="car.png" #画像データを用意
118 th3 = squeezenet(__image) #推論準備
119 __res = th3.squeeze_infer() #推論実行
120 th4 = squeeze_label(__label,__res) #推論結果の加工の準備
121 th41 = th4.squeeze_label() #メソッドをインスタンス化。推論結果Top5を表示して終了。戻り値なし
122 print("_____") #推論結果のTop5を表示
123
```

メインプログラム

#画像データを用意

推論準備 `squeeze`をインスタンス化。変数として画像データ(`__image`)を渡す

推論実行 `.squeeze_infer`メソッドを実行。推論結果(戻り値)を受け取る

推論結果の加工の準備 `squeeze_label`をインスタンス化。変数としてラベルデータと、推論結果を渡す

推論結果のTop5を表示 `.squeeze_label()`メソッドをインスタンス化。推論結果Top5を表示して終了。戻り値なし

推論結果のTop5を表示

#画像データを用意

推論準備 `squeeze`をインスタンス化。変数として画像データ(`__image`)を渡す

推論実行 `.squeeze_infer`メソッドを実行。推論結果(戻り値)を受け取る

推論結果の加工の準備 `squeeze_label`をインスタンス化。変数としてラベルデータと、推論結果を渡す

推論結果のTop5を表示 `.squeeze_label()`メソッドをインスタンス化。推論結果Top5を表示して終了。戻り値なし

推論結果のTop5を表示

# 7. サンプルコード解説 ② top\_road\_detection.py

## person-vehicle-bike-detection-crossroad-1016

学習済みモデル：person-vehicle-bike-detection-crossroad-1016の学習済みモデルを使います。

概要：入力された動画データから、物体を検出します。その物体は人、車、バイクです。

入力データ：右図上参照（1枚、3原色、サイズは512x512）

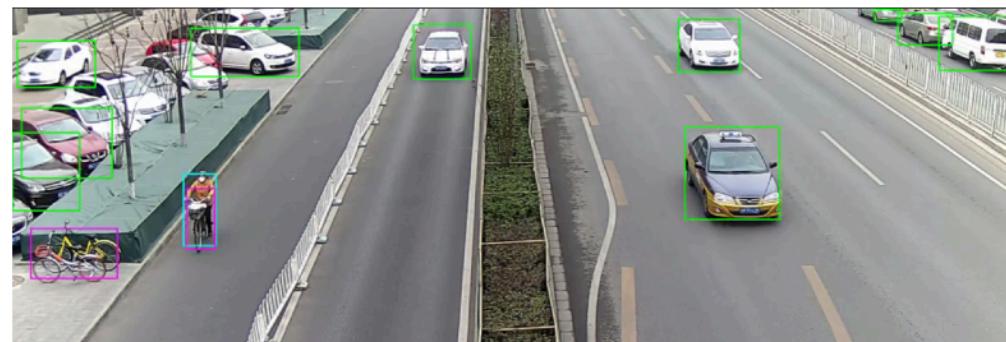
出力データ：右図下参照（ID,label,精度,X/Y座標）

### モデルの詳細 (Intel Open Model Zoo)

[Person-vehicle-bike-detection-crossroad-1016\\_intel\\_open\\_model\\_zoo](#)

### サンプルプログラム

読み込んだ動画データに、検出した物体に青枠をつけます



プログラム=53Lines

#### Inputs

- name: "input.1", shape: [1x3x512x512] - An input image in the format [BxCxHxW], where
  - B - batch size
  - C - number of channels
  - H - image height
  - W - image widthExpected color order: BGR.

#### Outputs

The net outputs blob with shape: [1, 1, N, 7], where N is the number of detected bounding boxes. Each detection has the format [`image_id`, `label`, `conf`, `x_min`, `y_min`, `x_max`, `y_max`], where:

- `image_id` - ID of the image in the batch
- `label` - predicted class ID (0 – non-vehicle, 1 – vehicle, 2 – person)
- `conf` - confidence for the predicted class
- `(x_min, y_min)` - coordinates of the top left bounding box corner
- `(x_max, y_max)` - coordinates of the bottom right bounding box corner.

## 7. サンプルコード解説 ② top\_road\_detection.py

本プログラムで使用するライブラリをインポートします

```
1  # _____ ///////////////// _____
2  #__For Openvino by Python written by M.Inoishi
3  #__person_vehicle_bike detection program
4  #__OpenVino2021R2
5  #__Ver3.0 2021/14/Apr
6  #_____ ///////////////// _____
7
8 #Step0__ Pythonライブラリー、モジュールのインポート
9 import cv2
10 import numpy as np
11 import logging as log
12 from time import time
13 from operator import itemgetter
14 from openvino.inference_engine import IECore
15
16 #Step1__ IECoreをieとしてインスタンス化
17 ie = IECore()
```

使用するライブラリをImport

#OpenCVモジュール（画像、動画ファイルの処理）  
#Pythonでの数値計算を高速、効率化するモジュール  
#処理中におけるLogの吐き出しモジュール  
#日付と時間を扱うモジュール  
#リストなどから所望する要素を抽出するモジュール  
#OpenVinoを動作させるためのモジュール(IECore)

ieにIECoreクラスをインスタンス化

## 7. サンプルコード解説 ② top\_road\_detection.py

クラスroad\_detectionです。推論モデルの準備

```
19 #Step2__ person-vehicle-bike-detection-crossroad(物体 人、車、バイク) を使用するクラス
20 class road_detection:
21     #本クラスの初期化メソッド
22     #推論モデルの構築、使用するプラグインを指定して、プラグインへロード
23     #推論モデルのInput条件も取得
24     def __init__(self):
25         #推論モデルのパスを指定
26         model = './ir_eng/person-vehicle-bike-detection-crossroad-1016'
27
28         #read_network(OpenVino API)で実行可能なモデルを構築
29         net = ie.read_network(model=model+'.xml', weights=model+'.bin')
30
31         #使用するプラグインを指定
32         __plug="CPU"
33         #__plug="GPU"
34         #__plug="MYRIAD"
35         #__plug="HETERO:FPGA,CPU"
36
37         #使用する推論モデルのInput条件を取得
38         print("Get_batch_size1=",net.batch_size)
39         self.input_layer = next(iter(net.input_info))
40         self.output_blobs = next(iter(net.outputs))
41         self.model_n, self.model_c, self.model_h, self.model_w = net.input_info[self.input_layer].input_data.shape
42         print("Input spec =",self.model_n,self.model_c,self.model_h,self.model_w)
43
44         #load_network(opennVino API)にて推論モデルをプラグインにロード
45         self.exec_net = ie.load_network(network=net, device_name=__plug)
46         return
```

road\_detectionクラスの定義

推論準備をする初期化メソッド

推論モデルの指定

推論モデルのネットワーク構築

プラグインの指定

推論モデルのInput条件を取得

プラグインに推論モデルのネットワークをロード

## 7. サンプルコード解説 ② top\_road\_detection.py

road\_detection\_inferのメソッドです。推論実行します

```
48     #推論実行するメソッド
49     #用意した動画データを、推論モデルのInput条件に加工
50     #person-vehicle-bike-detection-crossroad-1016推論モデルを使用して、推論実行
51     def road_detection_infer(self,__image):
52         self.cap_w = cap.get(3)
53         self.cap_h = cap.get(4)
54         in_frame = cv2.resize(__image, (self.model_w, self.model_h))
55         in_frame = in_frame.transpose((2, 0, 1))
56         in_frame = in_frame.reshape((self.model_n, self.model_c, self.model_h, self.model_w))
57
58         #Python上で推論（インファー）同期実行するためのコマンド(OpenVino API)
59         #res = self.exec_net.infer(inputs={self.input_layer:in_frame})
60
61         #Python上で推論（インファー）非同期実行するためのコマンド(OpenVino API)
62         self.exec_net.requests[0].async_infer({self.input_layer: in_frame})
63         request_status = self.exec_net.requests[0].wait(-1)
64         res = self.exec_net.requests[0].outputs[self.output_blobs]
65
66         return res
```

推論実行メソッド

入力データを加工

推論非同期実行

推論の結果待ち  
resに推論結果

## 7. サンプルコード解説 ② top\_road\_detection.py

road\_detection\_displayメソッドです。推論結果を加工します

推論結果から動画データに青枠で囲んでレンダリングします

```
68     #推論結果をもとに、動画ファイル上に四角枠を表示するメソッド
69     def road_detection_display(self, __res, __num, __image):
70         for obj in __res[0][0]:
71             if obj[2] > 0.08:
72                 __num = __num + 1
73                 xmin = int(obj[3] * self.cap_w)
74                 ymin = int(obj[4] * self.cap_h)
75                 xmax = int(obj[5] * self.cap_w)
76                 ymax = int(obj[6] * self.cap_h)
77                 #class_id = int(obj[1])
78                 color = (255, 0, 0)
79                 cv2.rectangle(__image, (xmin, ymin), (xmax, ymax), color, 2)
80                 cv2.imshow('Video', __image)
81         return __num
82 
```

推論結果加工メソッド

推論結果から、  
検出した物体の座標を取得  
動画データ上に四角枠で表示

## 7. サンプルコード解説 ② top\_road\_detection.py

メインプログラムです。

インスタンス化して、各種クラス、メソッドを実行します。

```
87 #Step4__ main __ メインプログラムを実行するためのスレッド（ここからプログラムの実行が指定される）  
88 if __name__=='__main__':  
89     th1 = road_detection()  
90     cap = cv2.VideoCapture('test1.mp4')  
91  
92     while cap.isOpened():  
93         num = 0  
94         ret, image = cap.read()  
95         key = cv2.waitKey(1)  
96         if key == 27:  
97             break  
98  
99         res = th1.road_detection_infer(image)  
100        num = th1.road_detection_display(res, num, image)  
101        print('Count____', num)
```

メインプログラム

推論準備

動画データ読み込み

動画データがなくなるまで以下の処理を繰り返し

推論実行

推論結果加工、表示

# 7. サンプルコード解説 ③ top\_head.py

## head-pose-estimation-adas-0001

学習済みモデル：head-pose-estimation-adas-0001の学習済みモデルを使います。

概要：入力されたカメラデータ（リアルタイム）から、頭の向きを推定します。

入力データ：右図上参照（1枚、3原色、サイズは60x60）

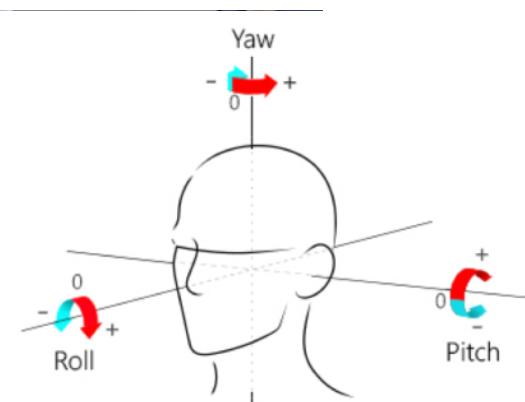
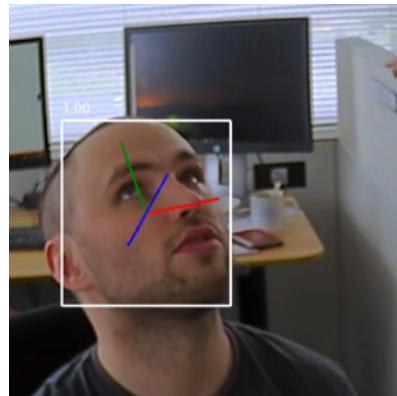
出力データ：右図下参照（Yaw,Pitch,Rollで角度）

### モデルの詳細 (Intel Open Model Zoo)

[head-pose-estimation-adas-0001\\_intel\\_open\\_model\\_zoo](#)

### サンプルプログラム

取り込んだWebカメラ（リアルタイム）から頭のYaw,Roll,PitchでPrint表示します。



#### Inputs

- name: "data" , shape: [1x3x60x60] - An input image in [1xCxHxW] format. Expected color order is BGR.

#### Outputs

Output layer names in Inference Engine format:

- name: "angle\_y\_fc", shape: [1, 1] - Estimated yaw (in degrees).
- name: "angle\_p\_fc", shape: [1, 1] - Estimated pitch (in degrees).
- name: "angle\_r\_fc", shape: [1, 1] - Estimated roll (in degrees).

プログラム=48Lines

## 7. サンプルコード解説 ③ top\_head.py

本プログラムで使用するライブラリをインポートします

```
1 # _____//____//____//____//_____
2 #__For Openvino by Python written by M.Inoishi
3 #__Head pose detection program
4 #__OpenVino2021R2
5 #__Ver3.0 2021/14/Apr
6 #_____//____//____//____//_____
7
8 #Step0__ Pythonライブラリー、モジュールのインポート
9 import cv2                                     #OpenCVモジュール（画像、動画ファイルの処理）
10 import numpy as np                            #Pythonでの数値計算を高速、効率化するモジュール
11 import logging as log                          #処理中におけるLogの吐き出しモジュール
12 from time import time                         #日付と時間を扱うモジュール
13 from operator import itemgetter               #リストなどから所望する要素を抽出するモジュール
14 from openvino.inference_engine import IECore   #OpenVinoを動作させるためのモジュール(IECore)
15
16 #Step1__ IECoreをieとしてインスタンス化
17 ie = IECore()
```

## 使用するライブラリをImport

- #OpenCVモジュール（画像、動画ファイルの処理）
- #Pythonでの数値計算を高速、効率化するモジュール
- #処理中におけるLogの吐き出しモジュール
- #日付と時間を扱うモジュール
- #リストなどから所望する要素を抽出するモジュール
- #OpenVinoを動作させるためのモジュール（IECore）

ieにIECoreクラスをインスタンス化

## 7. サンプルコード解説 ③ top\_head.py

クラスroad\_detectionです。推論モデルを使用できるようにします。

```
19 #Step2__ head-pose-estimation-adas-0001(顔の向き) 推論モデルを使用するクラス
20 class head_pose:
21     #本クラスの初期化メソッド
22     #推論モデルの構築、使用するプラグインを指定して、プラグインへロード
23     #推論モデルのInput条件も取得
24     def __init__(self):
25         #推論モデルのパスを指定
26         model = './ir_eng/head-pose-estimation-adas-0001'
27
28         #read_network(OpenVino API)で実行可能なモデルを構築
29         net = ie.read_network(model=model+'.xml', weights=model+'.bin')
30
31         #使用するプラグインをしてい
32         __plug="CPU"
33         #__plug="GPU"
34         #__plug="MYRIAD"
35         #__plug="HETERO:FPGA,CPU"
36
37         #使用する推論モデルのInput条件を取得
38         print("Get_batch_size1=",net.batch_size)
39         self.input_layer = next(iter(net.input_info))
40         self.output_blobs = next(iter(net.outputs))
41         self.model_n, self.model_c, self.model_h, self.model_w = net.input_info[self.input_layer].input_data.shape
42         print("Input spec =",self.model_n,self.model_c,self.model_h,self.model_w)
43
44
45         #load_network(opennVino API)にて推論モデルをプラグインにロード
46         self.exec_net = ie.load_network(network=net, device_name=__plug,num_requests=1)
47         return
```

head\_poseクラスの定義

推論準備をする初期化メソッド

推論モデルの指定

推論モデルのネットワーク構築

プラグインの指定

推論モデルのInput条件を取得

プラグインに推論モデルのネットワークをロード

## 7. サンプルコード解説 ③ top\_head.py

head\_pose\_inferのメソッドです。推論実行します

```
49     #推論実行するメソッド
50     #用意したカメラのデータを、推論モデルのInput条件に加工
51     #head-pose-estimation-adas-0001推論モデルを使用して、推論実行
52     def head_pose_infer(self, frame):
53         in_frame = cv2.resize(frame, (self.model_w, self.model_h))
54         in_frame = in_frame.transpose((2, 0, 1))
55         in_frame = in_frame.reshape((self.model_n, self.model_c, self.model_h, self.model_w))
56
57         #Python上で推論（インファー）同期実行するためのコマンド(OpenVino API)
58         #res = self.exec_net.infer(inputs={self.input_layer:in_frame})
59
60         #Python上で推論（インファー）非同期実行するためのコマンド(OpenVino API)
61         self.exec_net.requests[0].async_infer({self.input_layer: in_frame})
62         request_status = self.exec_net.requests[0].wait(-1)
63         res = self.exec_net.requests[0].outputs
64
65         return res
```

推論実行するメソッド

入力データを加工

推論非同期実行

推論の結果待ち  
resに推論結果

## 7. サンプルコード解説 ③ top\_head.py

Demoクラスです。推論結果を加工します

推論結果をPrint表示するだけです。値だけではわからないので、Yaw,Pitch ,Roll を補足しました。

```
70 #Step3__ demoのクラス
71 class demo:
72     #初期化メソッド
73     def __init__(self):
74         return
75
76     #結果を表示するメソッド
77     def head_demo_print(self,res):
78         res_y_fc = np.squeeze(res['angle_y_fc'])
79         res_p_fc = np.squeeze(res['angle_p_fc'])
80         res_r_fc = np.squeeze(res['angle_r_fc'])
81         print("yaw=",int(res_y_fc), "pitch=",int(res_p_fc),"roll=",int(res_r_fc))
82         return
```

demoクラスの定義

推論結果を加工して、  
PrintでYaw ,Pitch,Roll表示

## 7. サンプルコード解説 ③ top\_head.py

メインプログラムです。

インスタンス化して、各種クラス、メソッドを実行します。

```
84 #Step4__ main __ メインプログラムを実行するためのスレッド（ここからプログラムの実行が指定される）
85 if __name__=='__main__':
86     th1 = head_pose()
87     th2 = demo()
88
89     cap = cv2.VideoCapture(0)                                メインプログラム
90
91     while cap.isOpened():                                    カメラデータ読み込み
92         ret, __image = cap.read()                           カメラ入力データがなくなるまで以下の処理を繰り返し
93         cv2.imshow("Video", __image)
94         key = cv2.waitKey(1)
95         if key == 27:
96             break
97
98         __res = th1.head_pose_infer(__image)                推論実行
99         th2.head_demo_print(__res)                          推論結果加工、表示
```



# RYOYO

すべてを、つなげよう。技術で、発想で。

Copyright 2019 © Ryoyo Electro Corporation

**Confidential**