AI CASE STUDY

Georg Zitzlsberger

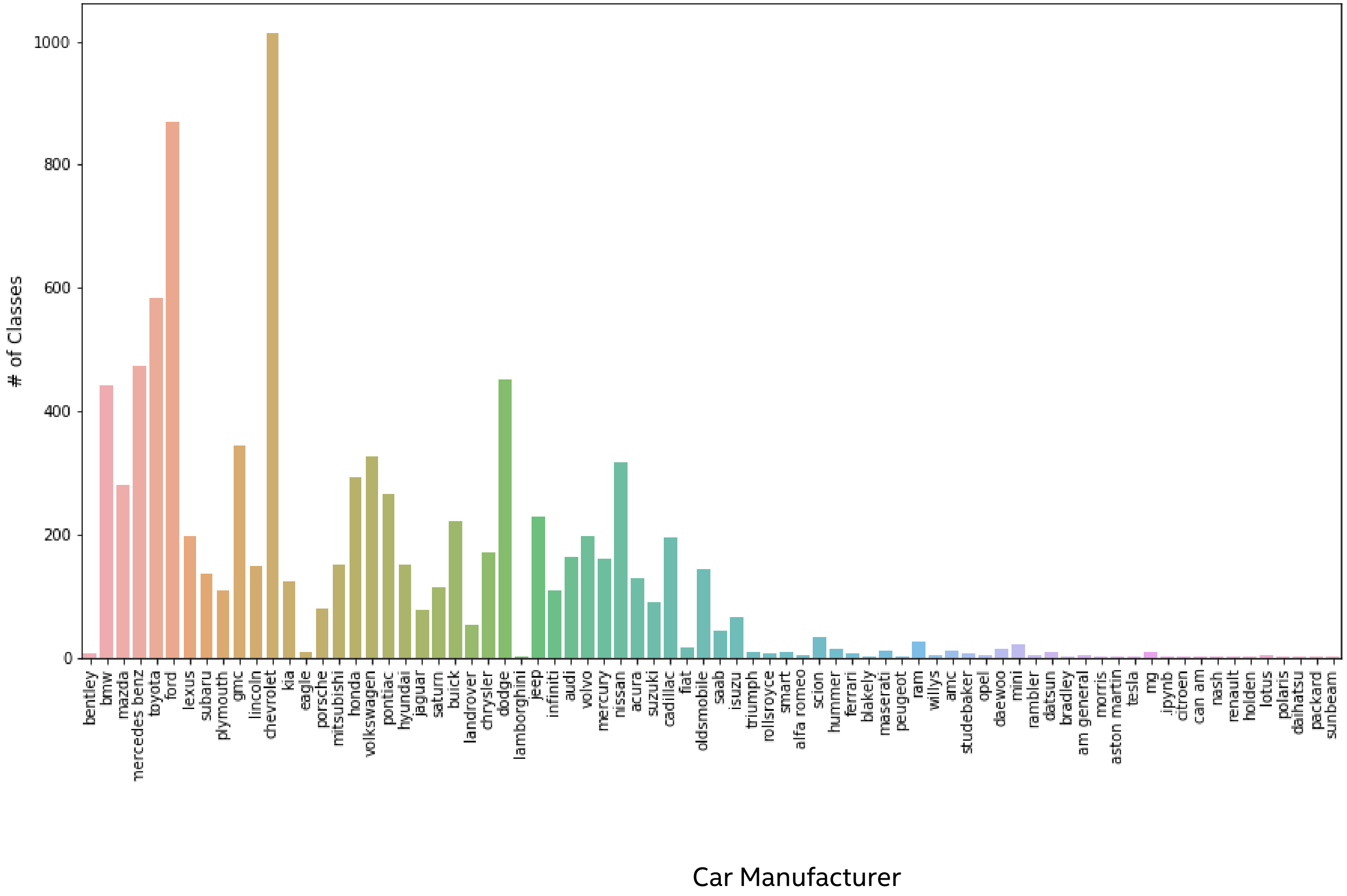
Bayncore

EXPLORATORY DATA ANALYSIS

Initial assessment of the dataset

The Vehicle Make and Model Recognition dataset (VMMRdb):

- Large in scale and diversity
- Images are collected from Craigslist
- Contains 9170 classes
- Identified 76 Car Manufacturers
- 291,752 images in total
- Manufactured between1950-2016



Dataset for the stolen cars challenge

[Hottest Wheels: The Most Stolen New And Used Cars In The U.S.]

Choose the 10 classes in this problem – shortens training time

- Honda Civic (1998): 45,062
- Honda Accord (1997): 43,764
- Ford F-150 (2006): 35,105
- Chevrolet Silverado (2004): 30.056
- Toyota Camry (2017): 17,276
- Nissan Altima (2016): 13,358
- Toyota Corolla (2016): 12,337
- Dodge/Ram Pickup (2001): 12,004
- GMC Sierra (2017): 10,865
- Chevrolet Impala (2008): 9,487

# indicates number of stolen cars in each model in 2017

Prepare dataset for the stolen cars challenge

- **Map multiple year vehicles to the stolen car category (based on exterior similarity)**
  - **Provides more samples to work with**

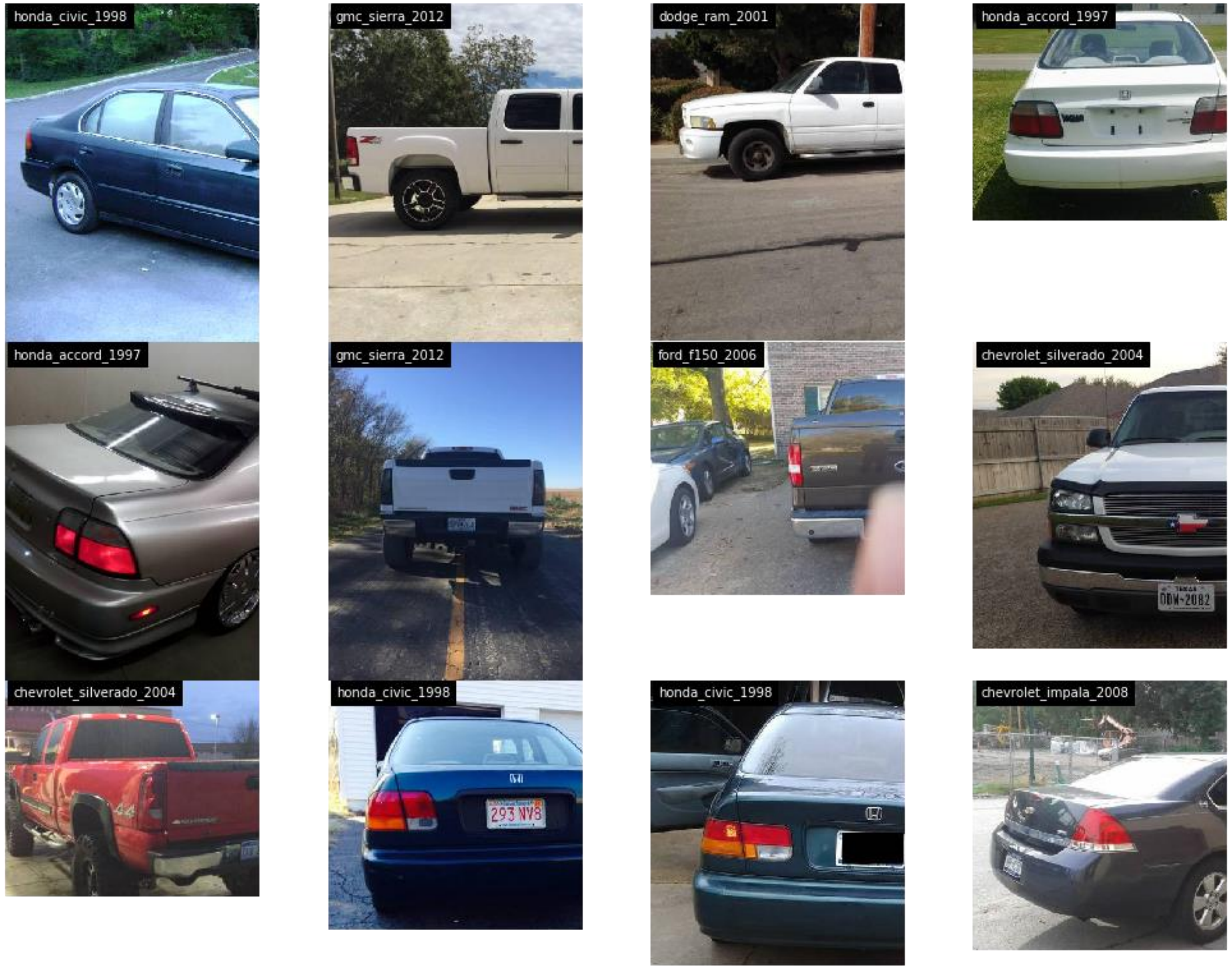
- Honda Civic (1998)	→ Honda Civic (1997 - 1998)
- Honda Accord (1997)	→ Honda Accord (1996 - 1997)
- Ford F-150 (2006)	→ Ford F150 (2005 - 2007)
- Chevrolet Silverado (2004)	→ Chevrolet Silverado (2003 - 2004)
- Toyota Camry (2017)	→Toyota Camry (2012 - 2014)
- Nissan Altima (2016)	→Nissan Altima (2013 - 2015)
- Toyota Corolla (2016)	→Toyota Corolla (2011 - 2013)
- Dodge/Ram Pickup (2001)	→Dodge Ram 1500 (1995 - 2001)
- GMC Sierra (2017)	→GMC Sierra 1500 (2007 - 2013)
- Chevrolet Impala (2008)	→Chevrolet Impala (2007 - 2009)

Preprocess the dataset

- Fetch and visually inspect a dataset
- Image Preprocessing
  - Address Imbalanced Dataset Problem
  - Organize a dataset into training, validation and testing groups
  - Augment training data
    - Limit overlap between training and testing data (!)
    - Sufficient testing and validation datasets
- **Complete Notebook: Part1-Exploratory_Data_Analysis.ipynb**

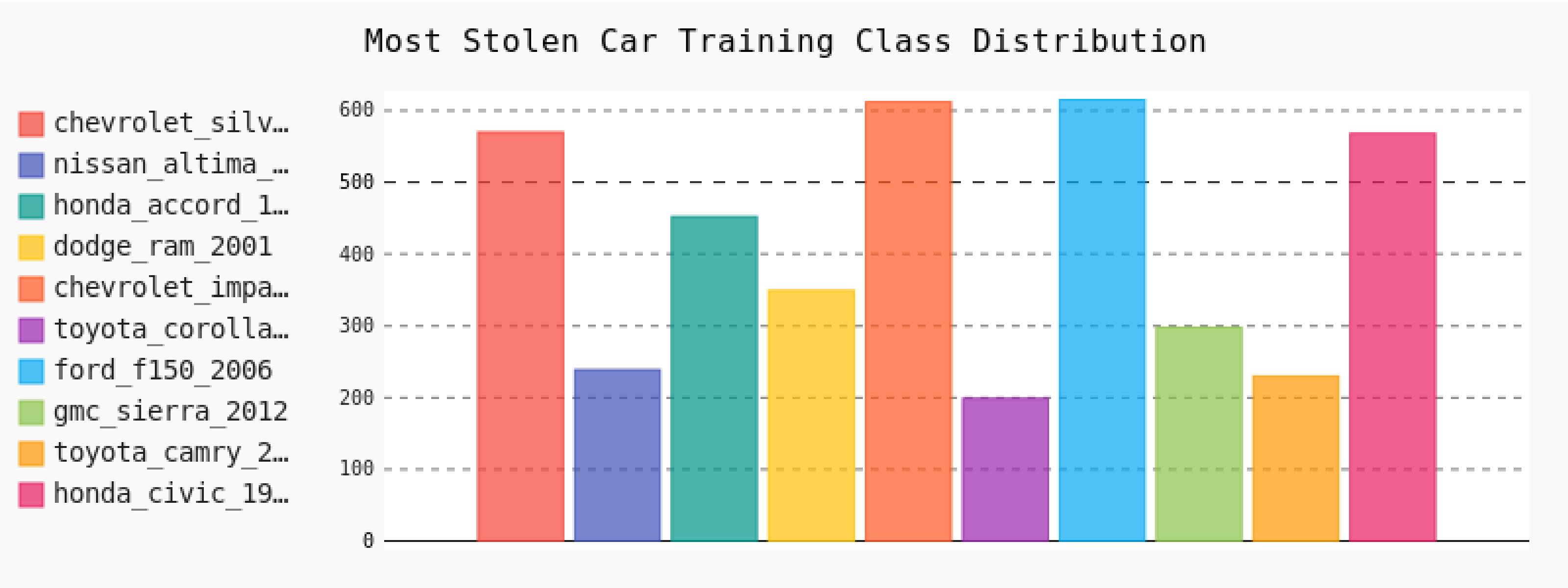
Inspect the dataset

- **Visually Inspecting the Dataset**
  - Taking note of variances
    - ¾ view
    - Front view
    - Back view
    - Side View, etc.
    - Sizes of images differ
    - Image aspect ratio differs
- **Sample Class name:**
  - Manufacturer
  - Model
  - Year



Data creation

- Honda Civic (1998)
- Honda Accord (1997)
- Ford F-150 (2006)
- Chevrolet Silverado (2004)
- Toyota Camry (2014)
- Nissan Altima (2014)
- Toyota Corolla (2013)
- Dodge/Ram Pickup (2001)
- GMC Sierra (2012)
- Chevrolet Impala (2008)



Preprocessing & Augmentation

PREPROCESSING

- **Removes inconsistencies and incompleteness in the raw data and cleans it up for model consumption**
- **Techniques:**
  - Black background
  - Rescaling, gray scaling
  - Sample wise centering, standard normalization
  - Feature wise centering, standard normalization
  - RGB → BGR

DATA AUGMENTATION

- **Improves the quantity and quality of the dataset**
- **Helpful when dataset is small or some classes have less data than others**
- **Techniques:**
  - Rotation
  - Horizontal & Vertical Shift, Flip
  - Zooming & Shearing

Learn more about the preprocessing and augmentation methods in Optional-VMMR_ImageProcessing_DataAugmentation.ipynb

Preprocessing & Augmentation

GRAY SCALING

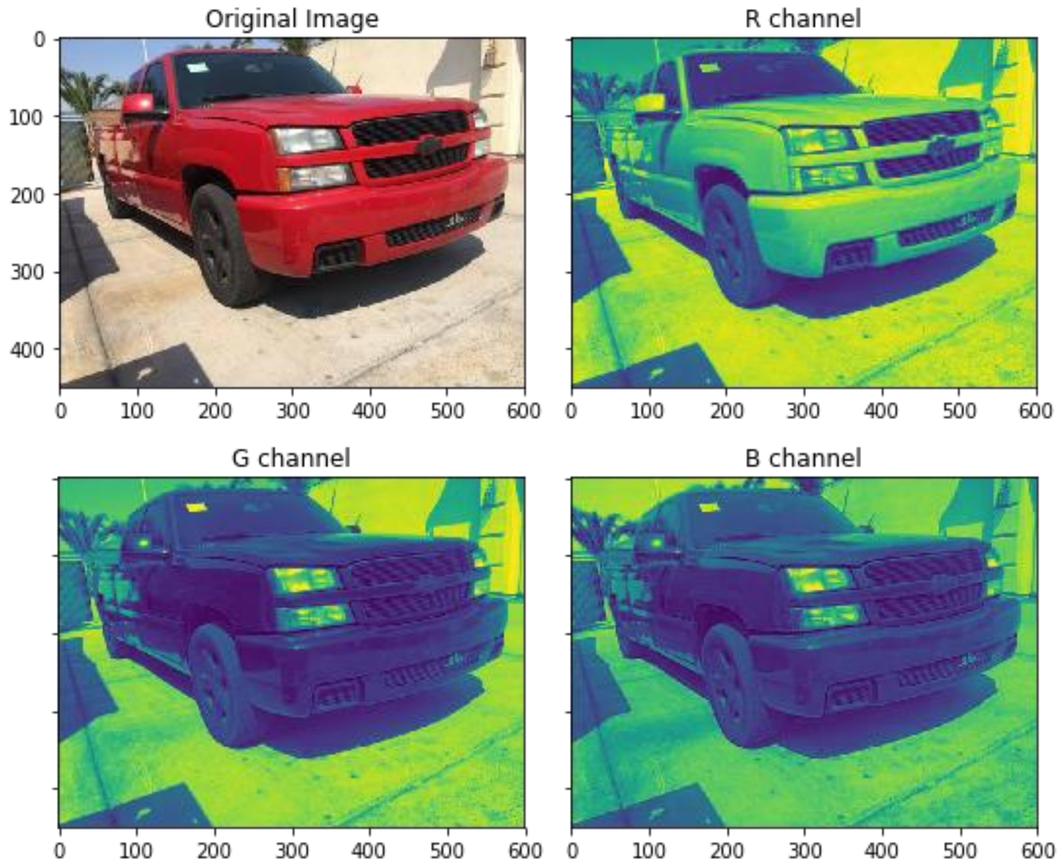
SAMPLE-WISE CENTERING

SAMPLE STD NORMALIZATION

ROTATED

RGB channels

- **Images are made of pixels**
- **Pixels are made of combinations of Red, Green, Blue, channels.**



EXPLORATION JUPYTER NOTEBOOK EXERCISE

RGB – BGR

- **Depending on the network choice RGB-BGR conversion is required.**
- **One way to achieve this task is to use Keras*** `preprocess_input`

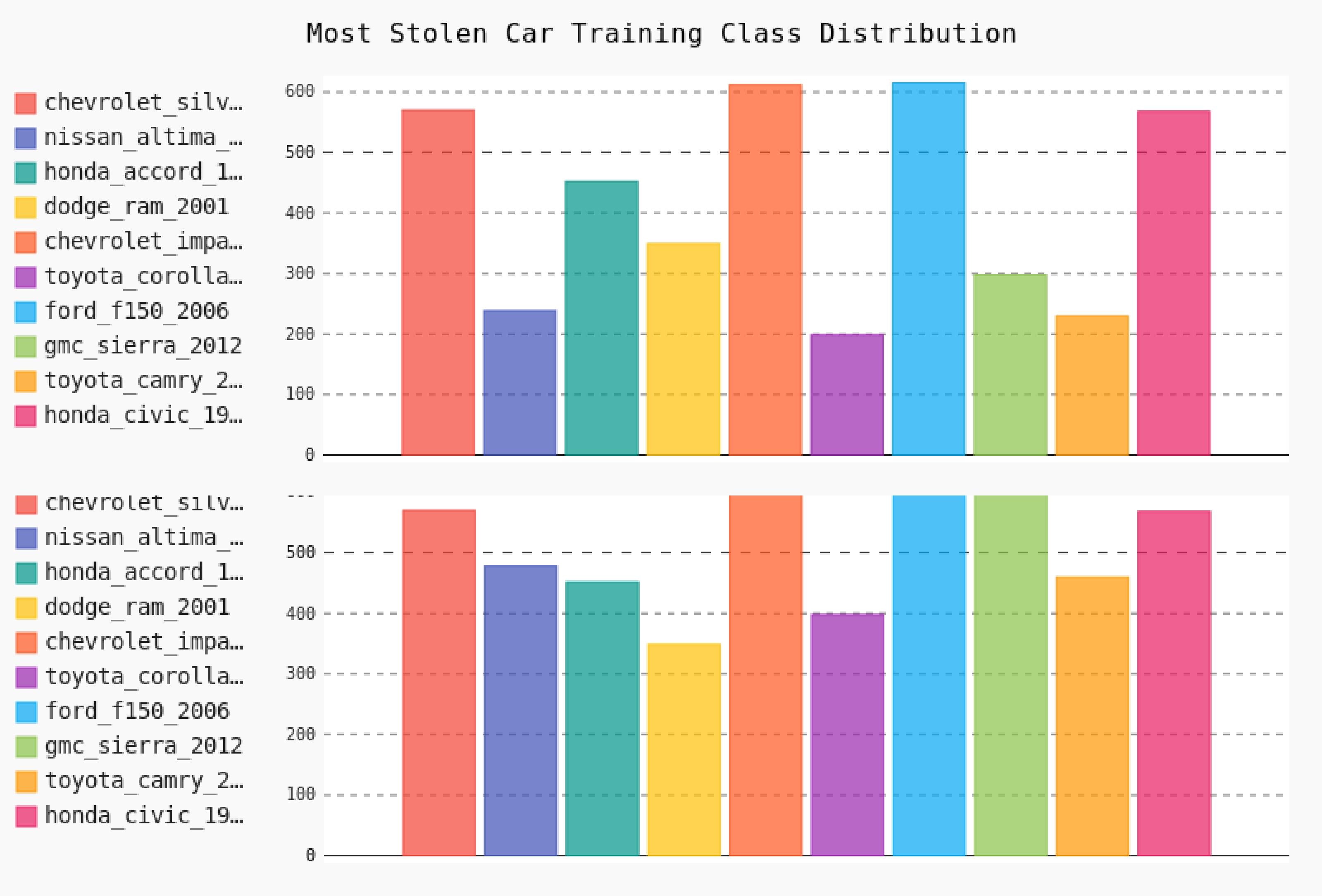
```
>> keras.preprocessing.image.ImageDataGenerator(preprocessing_function=preprocess_input)
```

Complete Notebook : Part1-Exploratory_Data_Analysis.ipynb

Summary

Before Preprocessing

After Preprocessing



THE TRAINING PHASE

SELECTING A FRAMEWORK

Decision metrics for choosing a framework

WHICH FRAMEWORKS IS INTEL OPTIMIZING?

WHAT ARE THE DECISION FACTORS FOR CHOOSING A SPECIFIC FRAMEWORK?

WHY DID WE CHOOSE TENSORFLOW?

OPTIMIZED DEEP LEARNING FRAMEWORKS

INSTALL AN INTEL-OPTIMIZED FRAMEWORK AND FEATURED TOPOLOGY

FRAMEWORKS OPTIMIZED BY INTEL

TensorFlow™ *

mxnet *

Caffe2 *

PyTorch *

BigDL FOR Apache Spark *

More under optimization: PaddlePaddle and more.

GET STARTED TODAY AT AI.INTEL.COM/FRAMEWORK-OPTIMIZATIONS/

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MlLib on Spark, Mahout)
**Limited availability today*
Other names and brands may be claimed as the property of others.

CAFFE / TENSORFLOW / PYTORCH FRAMEWORKS

Developing Deep Neural Network models can be done faster with Machine learning frameworks/libraries. There are a plethora of choices of frameworks and the decision on which to choose is very important. Some of the criteria to consider for the choice are:

1. Opensource and Level of Adoption
2. Optimizations on CPU
3. Graph Visualization
4. Debugging
5. Library Management
6. Inference target (CPU/ Integrated Graphics/ Intel® Movidius™ Neural Compute Stick /FPGA)

Considering all these factors, we have decided to use the Google Deep Learning framework **TensorFlow**

SELECTING A NETWORK

How to SELECT A Network?

We started this project with the plan for inference on an edge device in mind as our ultimate deployment platform. To that end we always considered three things when selecting our topology or network: time to train, size, and inference speed.

- **Time to Train**: Depending on the number of layers and computation required, a network can take a significantly shorter or longer time to train. Computation time and programming time are costly resources, so we wanted a reduced training time.
- **Size**: Since we're targeting edge devices and an Intel® Movidius™ Neural Compute Stick, we must consider the size of the network that is allowed in memory as well as supported networks.
- **Inference Speed**: Typically the deeper and larger the network, the slower the inference speed. In our use case we are working with a live video stream; we want at least 10 frames per second on inference.
- **Accuracy**: It is equally important to have an accurate model. Even though, most pretrained models have their accuracy data published, but we still need to discover how they perform on our dataset.

Inception v3 Network

We decided to train our dataset on the Inception v3 network that is currently supported on our edge devices (CPU, Integrated GPU, Intel® Movidius™ Neural Compute Stick).

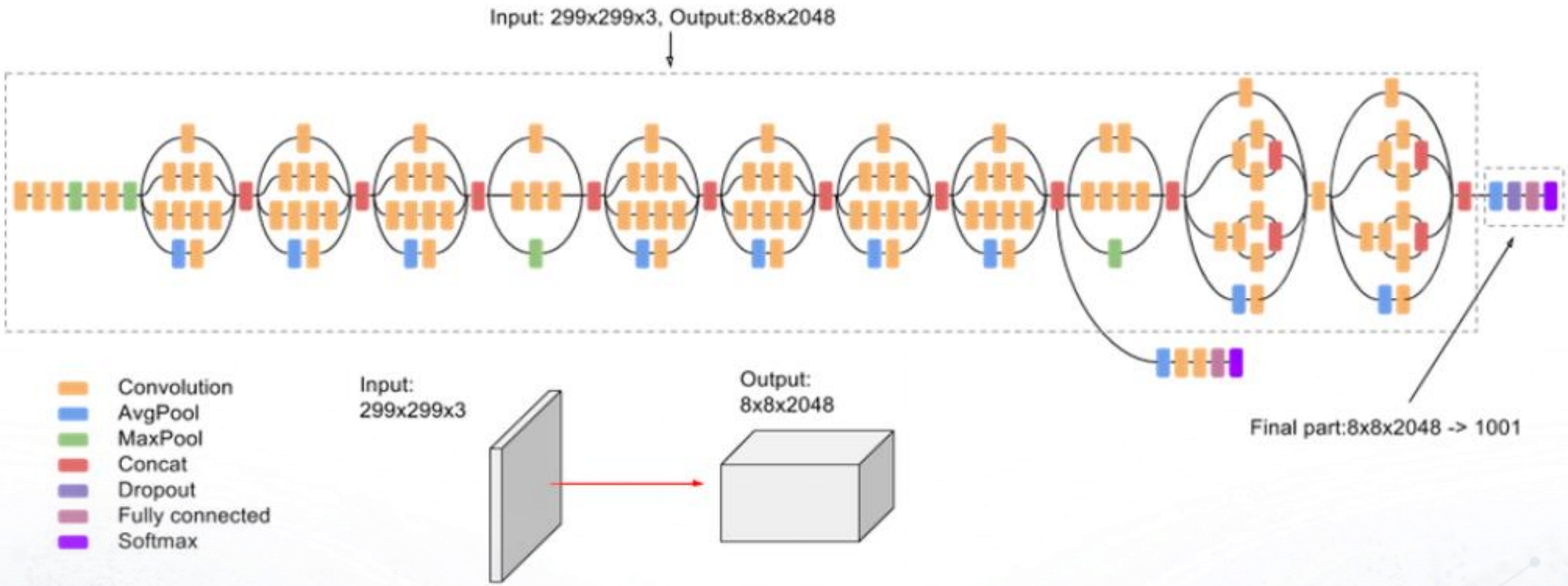
The original paper* trained on ResNet-50. However, it is not supported currently on Intel® Movidius™ Neural Compute Stick.

Other supported networks to train the model on:

- Inception v3
- VGG16
- MobileNet
- others

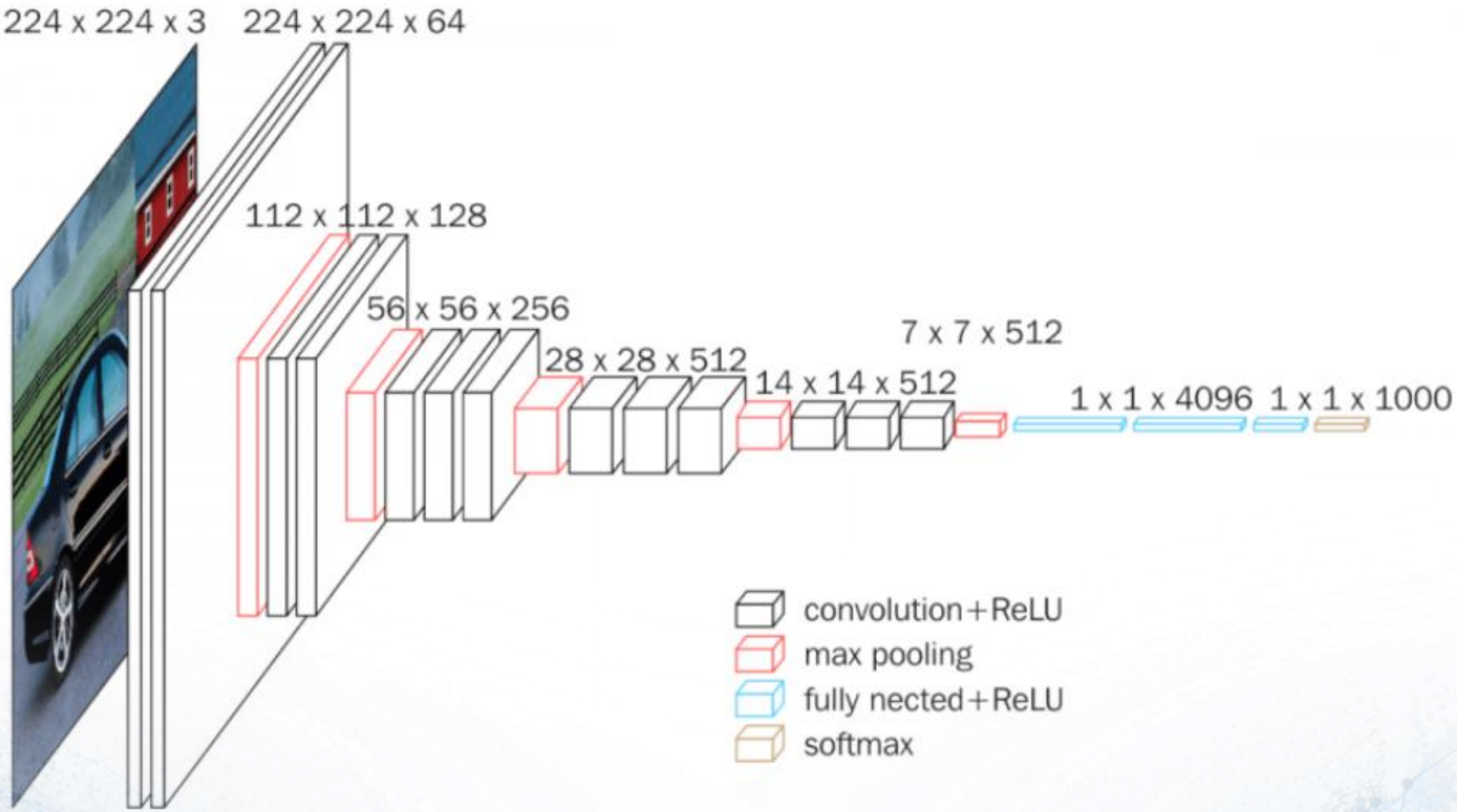
*http://vmmrdb.cecsresearch.org/papers/VMMR_TSWC.pdf

Inception v3



https://arxiv.org/abs/1512.00567

Vgg16



Very Deep Convolutional Networks for Large-Scale Image Recognition
Karen Simonyan and Andrew Zisserman, 2014

MOBILENET

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

https://arxiv.org/pdf/1704.04861.pdf

Inception v3 - VGG16 - MOBILENET

After training and comparing the performance and results based on the previously discussed criteria, our final choice of Network was **Inception V3**.

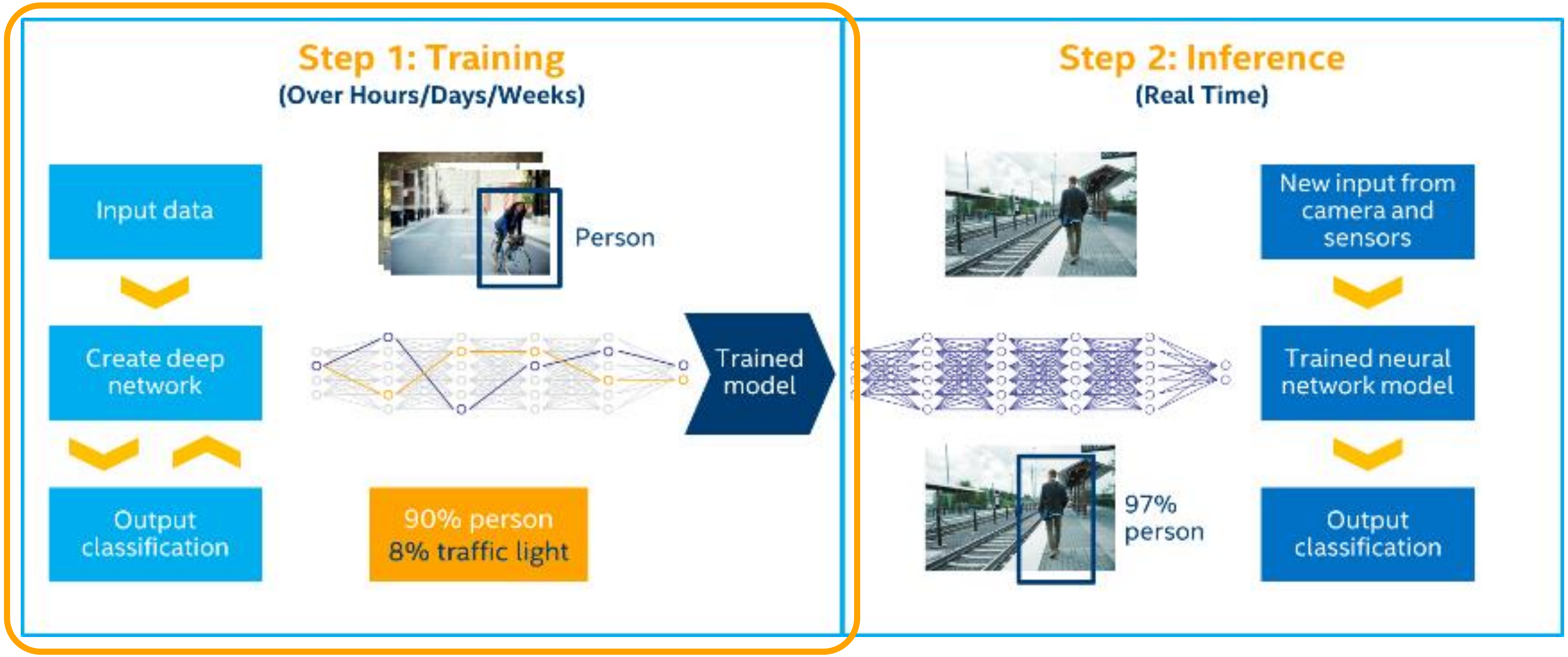
This choice was because, out of the three networks:

- MobileNet was the least accurate model (74%) but had the smallest size (16mb)
- VGG16 was the most accurate (89%) but the largest in size (528mb)
- InceptionV3 had median accuracy (83%) and size (92mb)

There are other network topologies available. This is just an example!

TRAINING JUPYTER NOTEBOOK EXERCISE

Training and Inference Workflow



Complete Notebook : Part2-Training_InceptionV3-Student.ipynb

Summary

Based on your projects requirements the choice of framework and topology will differ.

- Time to train
- Size of the model
- Inference speed
- Acceptable accuracy

There is no one size fits all approach to these choices and there is trial and error to finding your optimal solution.

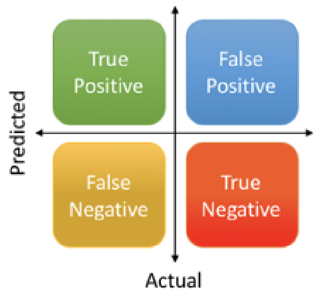
ANALYSIS JUPYTER NOTEBOOK EXERCISE

Model analysis

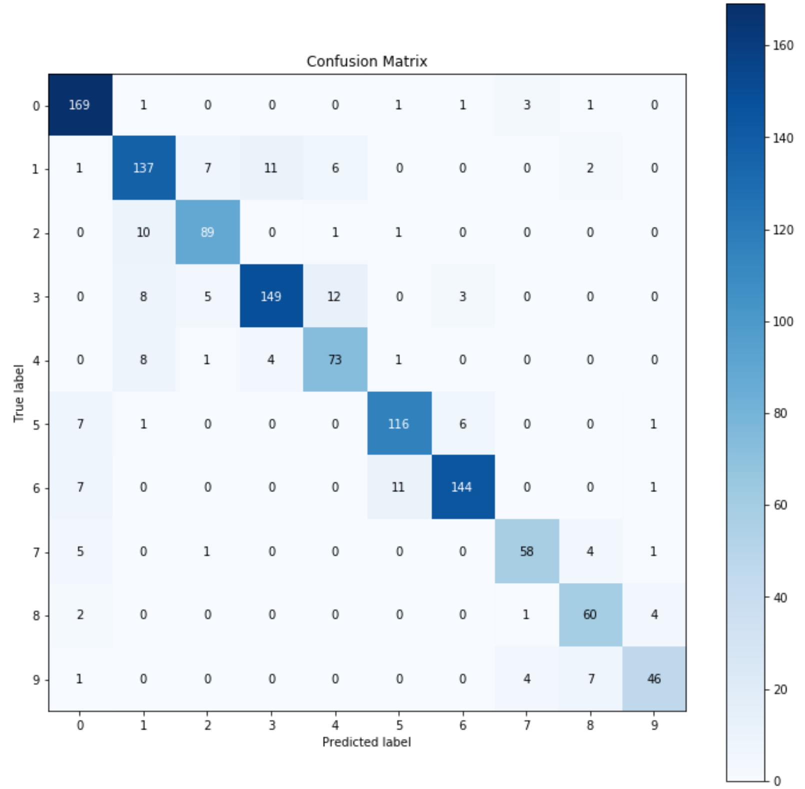
$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



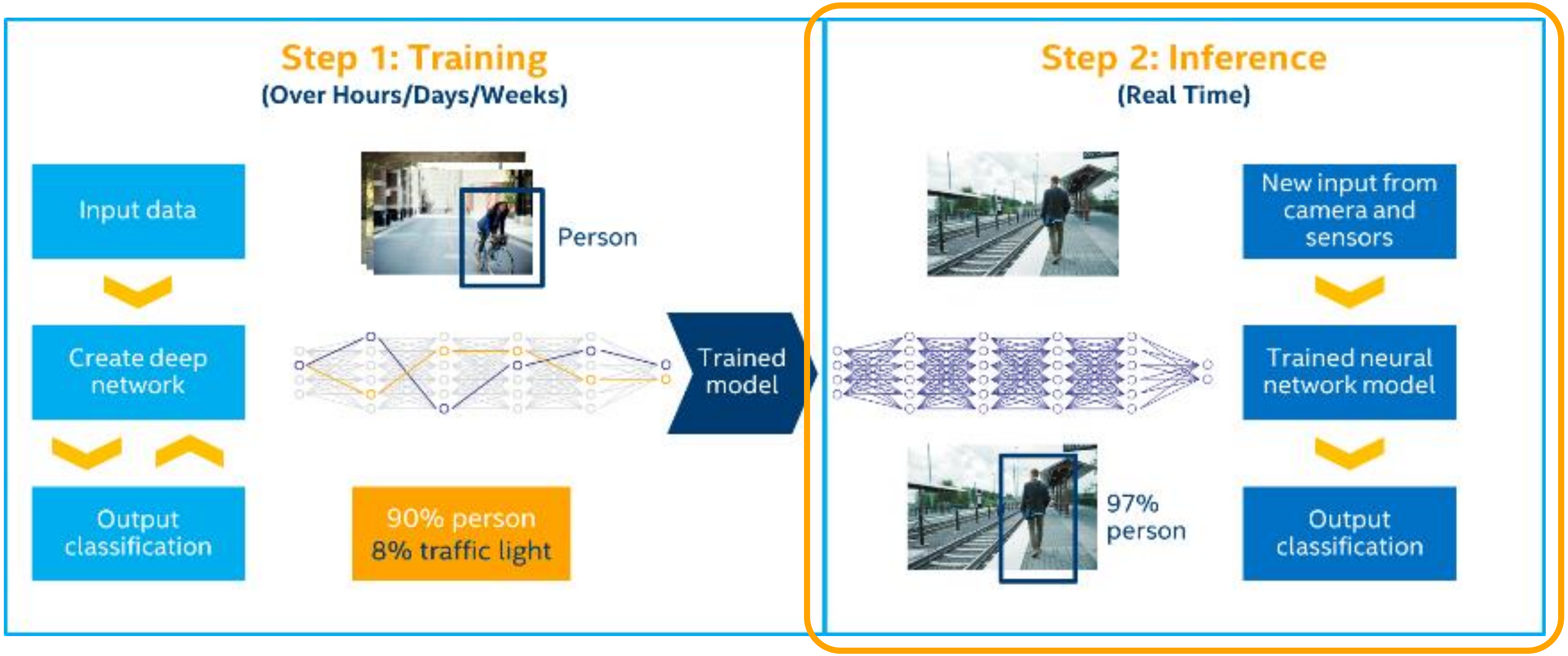
- Understand how to interpret the results of the training by analyzing our model with different metrics and graphs
  - Confusion Matrix
  - Classification Report
  - Precision-Recall Plot
  - ROC (Receiver Operating Characteristic) Plot



Complete Notebook : Part3-Model_Analysis.ipynb

THE DEPLOYMENT PHASE

What does deployment/inference mean?



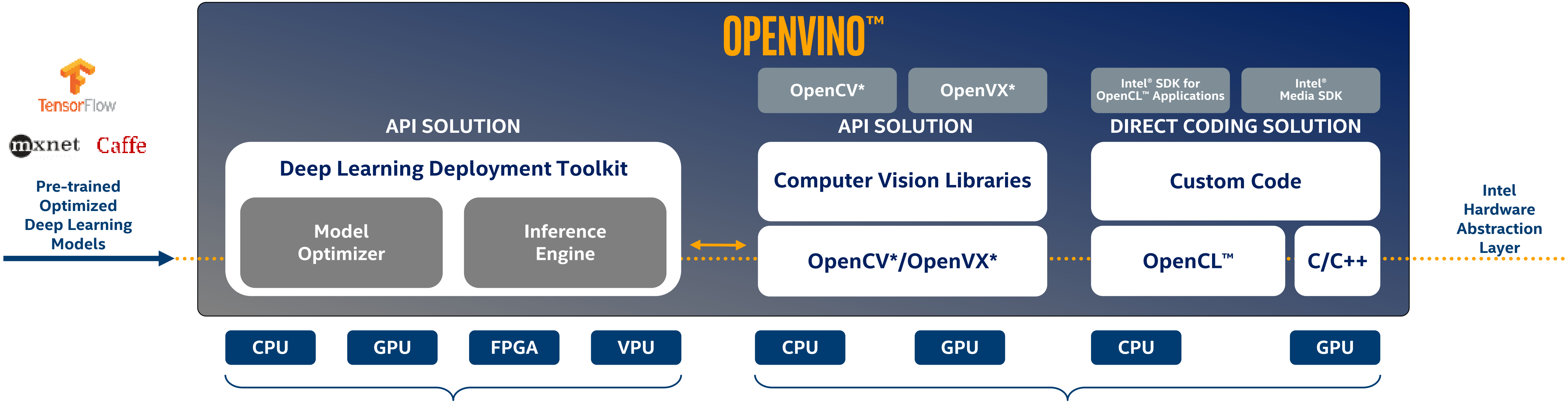
What is inference on the Edge?

Real-time evaluation of a model subject to the constraints of power, latency and memory

Requires AI models that are specially tuned to the above-mentioned constraints

Models such SqueezeNet, for example, are tuned for image inferencing on PCs and embedded devices

INTEL OPEN VISUAL INTERFACE & NEURAL NETWORK OPTIMIZATION (OPENVINO) TOOLKIT



DEEP LEARNING COMPUTER VISION

- Based on application of a large number of filters to an image to extract features.
- Features in the object(s) are analyzed with the goal of associating each input image with an output node for each type of object.
- Values are assigned to output node representing the probability that the image is the object associated with the output node.

TRADITIONAL COMPUTER VISION

- Based on selection and connections of computational filters to abstract key features and correlating them to an object
- Works well with well defined objects and controlled scene
- Difficult to predict critical features in larger number of objects or varying scenes

Intel® Deep Learning Deployment Toolkit

TRAIN

Train a DL model.
Currently supports:

- Caffe*
- Mxnet*
- TensorFlow*

TensorFlow
Caffe
mxnet

.prototxt
.caffemodel

PREPARE OPTIMIZE

Model optimizer:

- Converting
- Optimizing
- Preparing to inference

(device agnostic, generic optimization)

Run Model Optimizer

IR
.xml
.bin

INFERENCE

Inference engine lightweight API to use in applications for inference.

User Application

Inference Engine

OPTIMIZE/ HETEROGENEOUS

Inference engine supports multiple devices for heterogeneous flows.

(device-level optimization)

MKL-DNN	CPU: Intel® Xeon®/Intel® Core™/Intel Atom®
cl-DNN	GPU
DLA	ALTERA FPGA
Intel® Movidius™ API	Movidius Myriad™ 2/X

EXTEND

Inference engine supports extensibility and allows custom kernels for various devices.

Extensibility **C++**

Extensibility **OpenCL™**

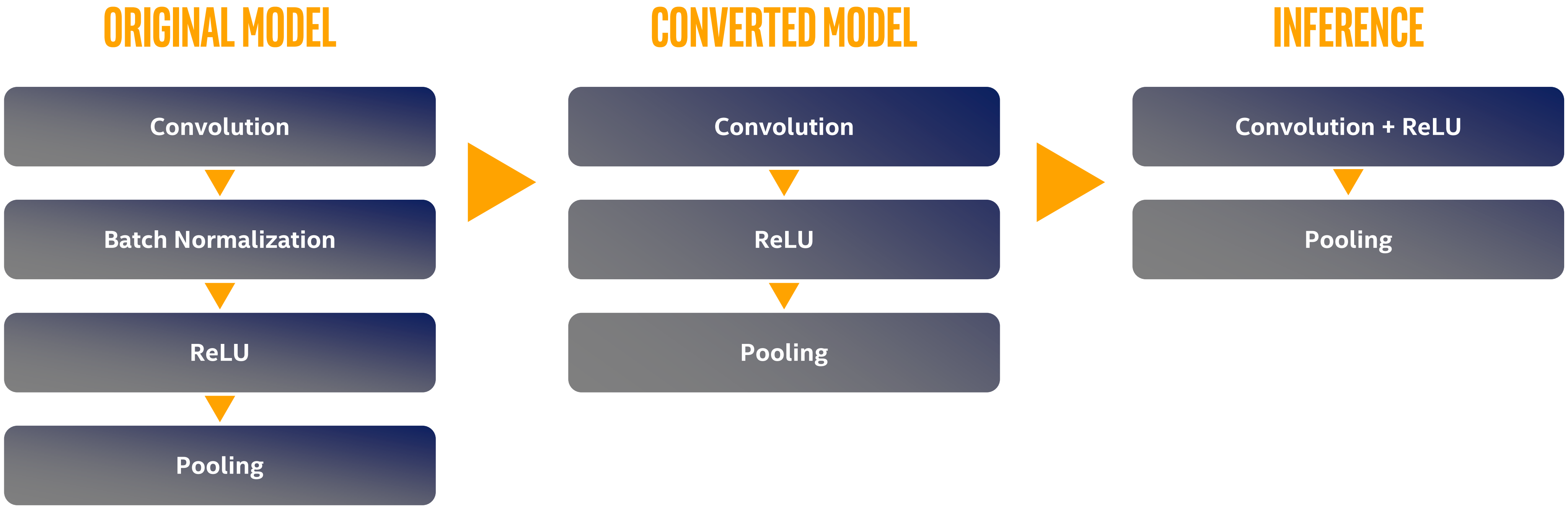
Extensibility **OpenCL™/TBD**

Extensibility **TBD**

Improve Performance with Model Optimizer

EXAMPLE

1. Remove Batch normalization stage.
2. Recalculate the weights to ‘include’ the operation.
3. Merge Convolution and ReLU into one optimized kernel.



Improve Performance with Model Optimizer (cont'd)

Model optimizer performs generic optimization:

- Node merging
- Horizontal fusion
- Batch normalization to scale shift
- Fold scale shift with convolution
- Drop unused layers (dropout)
- FP16/FP32 quantization

	FP32	FP16
CPU	YES	NO
GPU	YES	RECOMMENDED
MYRIAD	NO	YES
FPGA/DLA	NO	YES

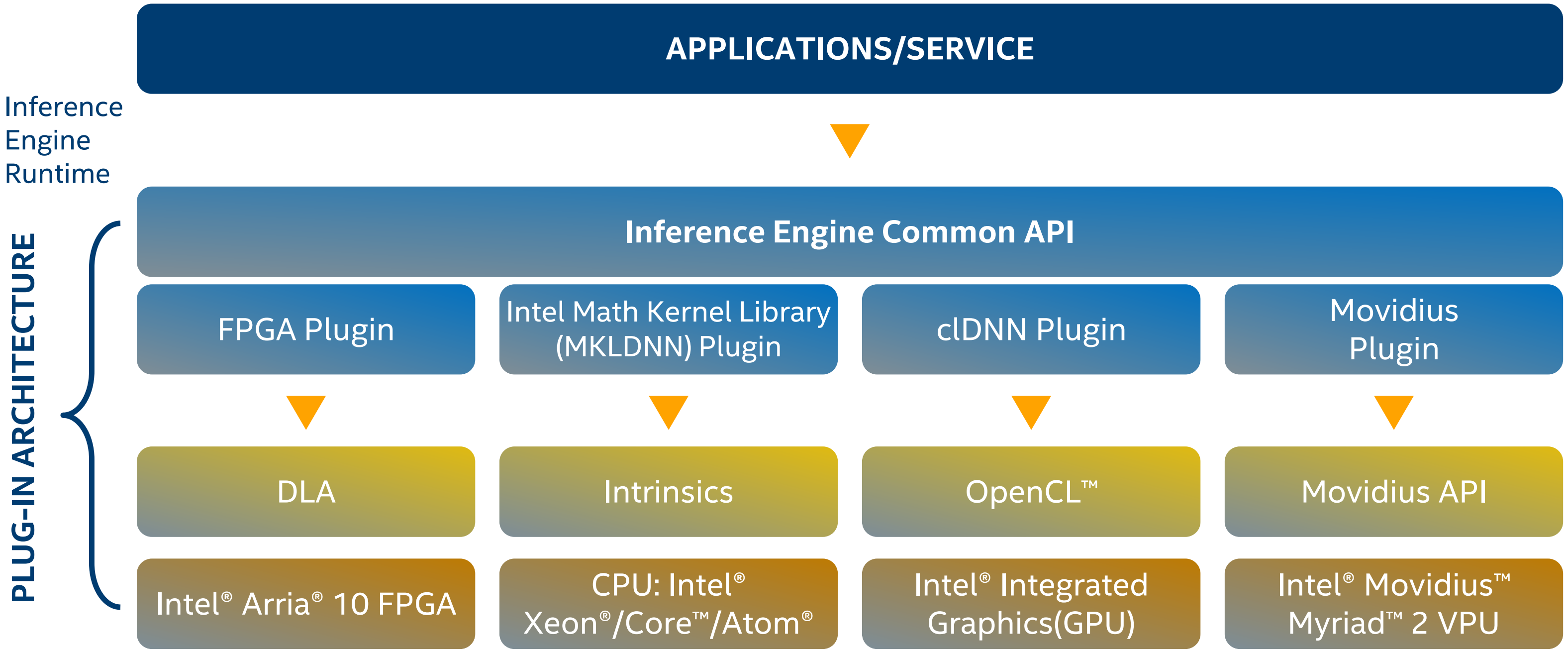
Model optimizer can cut out a portion of the network:

- Model has pre/post-processing parts that cannot be mapped to existing layers.
- Model has a training part that is not used during inference.
- Model is too complex and cannot be converted in one shot.

Optimal Model Performance Using the Inference Engine

- Simple & Unified API for Inference across all Intel® architecture (IA)
- Optimized inference on large IA hardware targets (CPU/iGPU/FPGA)
- Heterogeneity support allows execution of layers across hardware types
- Asynchronous execution improves performance
- Futureproof/scale your development for future Intel® processors

TRANSFORM MODELS & DATA INTO RESULTS & INTELLIGENCE



Layers Supported by Inference Engine Plugins

- **CPU – Intel® MKL-DNN Plugin**
  - Supports FP32, INT8 (planned)
  - Supports Intel® Xeon®/Intel® Core™/Intel Atom® platforms (https://github.com/01org/mkl-dnn)
- **GPU – clDNN Plugin**
  - Supports FP32 and FP16 (recommended for most topologies)
  - Supports Gen9 and above graphics architectures (https://github.com/01org/clDNN)
- **FPGA – DLA Plugin**
  - Supports Intel® Arria® 10
  - FP16 data types, FP11 is coming
- **Intel® Movidius™ Neural Compute Stick– Intel® Movidius™ Myriad™ VPU Plugin**
  - Set of layers are supported on Intel® Movidius™ Myriad™ X (28 layers), non-supported layers must be inferred through other inference engine (IE) plugins . Supports FP16

Layer Type	CPU	FPGA	GPU	MyriadX
Convolution	Yes	Yes	Yes	Yes
Fully Connected	Yes	Yes	Yes	Yes
Deconvolution	Yes	Yes	Yes	Yes
Pooling	Yes	Yes	Yes	Yes
ROI Pooling	Yes		Yes	
ReLU	Yes	Yes	Yes	Yes
PReLU	Yes		Yes	Yes
Sigmoid			Yes	Yes
Tanh			Yes	Yes
Clamp	Yes		Yes	
LRN	Yes	Yes	Yes	Yes
Normalize	Yes		Yes	Yes
Mul & Add	Yes		Yes	Yes
Scale & Bias	Yes	Yes	Yes	Yes
Batch Normalization	Yes		Yes	Yes
SoftMax	Yes		Yes	Yes
Split	Yes		Yes	Yes
Concat	Yes	Yes	Yes	Yes
Flatten	Yes		Yes	Yes
Reshape	Yes		Yes	Yes
Crop	Yes		Yes	Yes
Mul	Yes		Yes	Yes
Add	Yes	Yes	Yes	Yes
Permute	Yes		Yes	Yes
PriorBox	Yes		Yes	Yes
SimplerNMS	Yes		Yes	
Detection Output	Yes		Yes	Yes
Memory / Delay Object	Yes			
Tile	Yes			Yes

ACCELERATED AI/DL INFERENCE (INT8) ON 2ND GEN INTEL® XEON™ SCALABLE PROCESSORS

INTEL® DEEP LEARNING BOOST (DL BOOST)

FEATURING VECTOR NEURAL NETWORK INSTRUCTIONS (VNNI)

Sign ← *Mantissa* →

INT8	07	06	05	04	03	02	01	00

Current AVX-512 instructions to perform INT8 convolutions: **vpmaddubsw, vpmaddwd, vpaddd**

Future AVX-512 (VNNI) instruction to accelerate INT8 convolutions: **vpdpbusd****

Inference

- Use Model Optimizer to create the IR
- Use Inference Engine for video

Complete Notebook : Part4-OpenVINO_Video_Inference.ipynb

intel®

experience
what's inside™

Legal Disclaimer & Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804