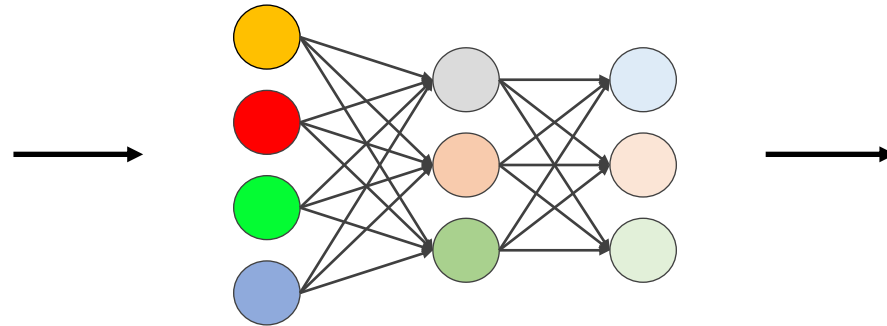


# Image Classification Example



Input Image




Pre-trained Model

**Golden Retriever**

Prediction Output

# Image Classification Tensorflow Code



```
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions

model = InceptionV3(weights='imagenet')

img = image.load_img(inp_file_name, target_size=(299, 299))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x, mode='tf')

preds = model.predict(x)
```

Pre-trained model

Pre-processing

Inference



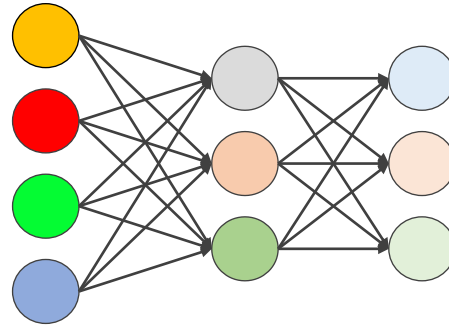
Input Image

[dog.jpg]

Pre-processing



[299,299]



InceptionV3  
imagenet

[frozen.pb]

filtering



**golden\_retriever: 85.93 %**  
Labrador\_retriever: 5.18 %  
kuvasz: 0.94 %

Predictions

[Array]

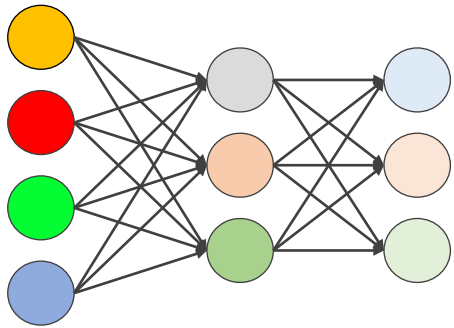
# OpenVINO Model Optimizer

```
# Setup model optimizer command ...
ir_name = "inceptionv3"
ir_data_type = "FP32"
ir_out_dir = f"{model_fname}/IR_models/{ir_data_type}"
ir_input_shape = "[1,299,299,3]"

mo_cmd = f"mo_tf.py \
    --saved_model_dir {model_fname} \
    --input_shape {ir_input_shape} \
    --data_type {ir_data_type} \
    --output_dir {ir_out_dir} \
    --model_name {ir_name}"

#run the Optimizer command
output = subprocess.check_output(mo_cmd, shell=True)
print (output.decode('utf-8'))
```

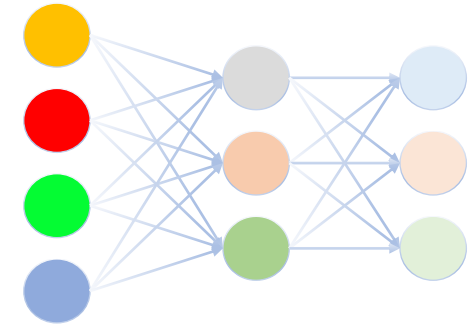
# OpenVINO Model Optimizer



[frozen.pb]



- Linear Operations Fusing
- ResNet optimization (stride optimization)
- Grouped Convolution Fusing
- Model Cutting



[model.bin, model.xml]



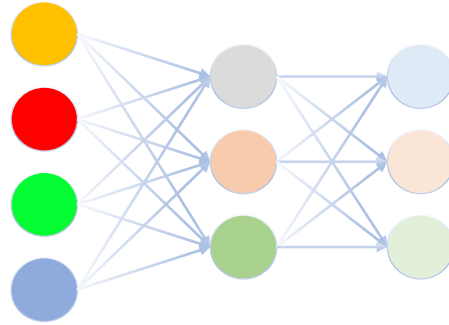
Input Image

[dog.jpg]

Pre-processing



[299,299]



OpenVINO  
Optimized Model

[model.bin, model.xml]

filtering

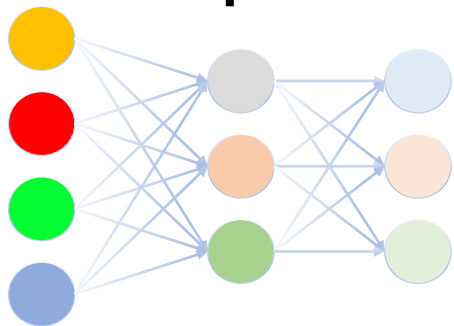
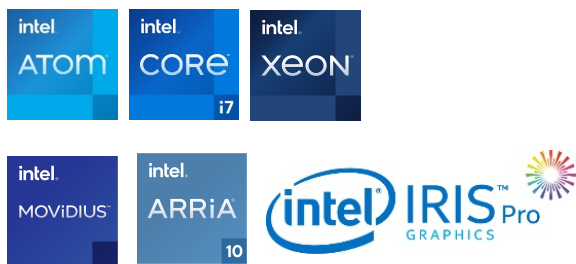


**golden\_retriever: 85.93 %**  
Labrador\_retriever: 5.18 %  
kuvasz: 0.94 %

Predictions

[Array]

# OpenVINO Inference



```
from openvino.inference_engine import IECore
```

```
model_xml = f'{ir_out_dir}/{ir_name}.xml'
```

```
model_bin = f'{ir_out_dir}/{ir_name}.bin'
```

```
# Load network to the plugin
```

```
ie = IECore()
```

```
net = ie.read_network(model=model_xml, weights=model_bin)
```

```
exec_net = ie.load_network(network=net, device_name="CPU")
```

```
del net
```

```
input_layer = next(iter(exec_net.input_info))
```

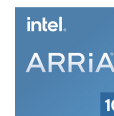
```
output_layer = next(iter(exec_net.outputs))
```

```
# Run the Inference on the Input image...
```

```
res = exec_net.infer(inputs={input_layer: input_image})
```

```
res = res[output_layer]
```

CPU  
GPU  
FPGA  
MYRIAD  
MULTI



GPUs

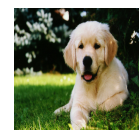
CPUs

FPGAs

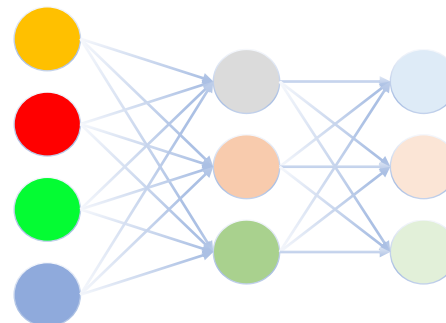
VPUs



Pre-processing



[299,299]



**golden\_retriever: 89.87 %**  
Labrador\_retriever: 1.84 %  
tennis\_ball: 0.79 %

Input Image

[dog.jpg]

OpenVINO Optimized  
Pre-trained Model

[model.bin, model.xml]

Prediction Outcomes

[Array]

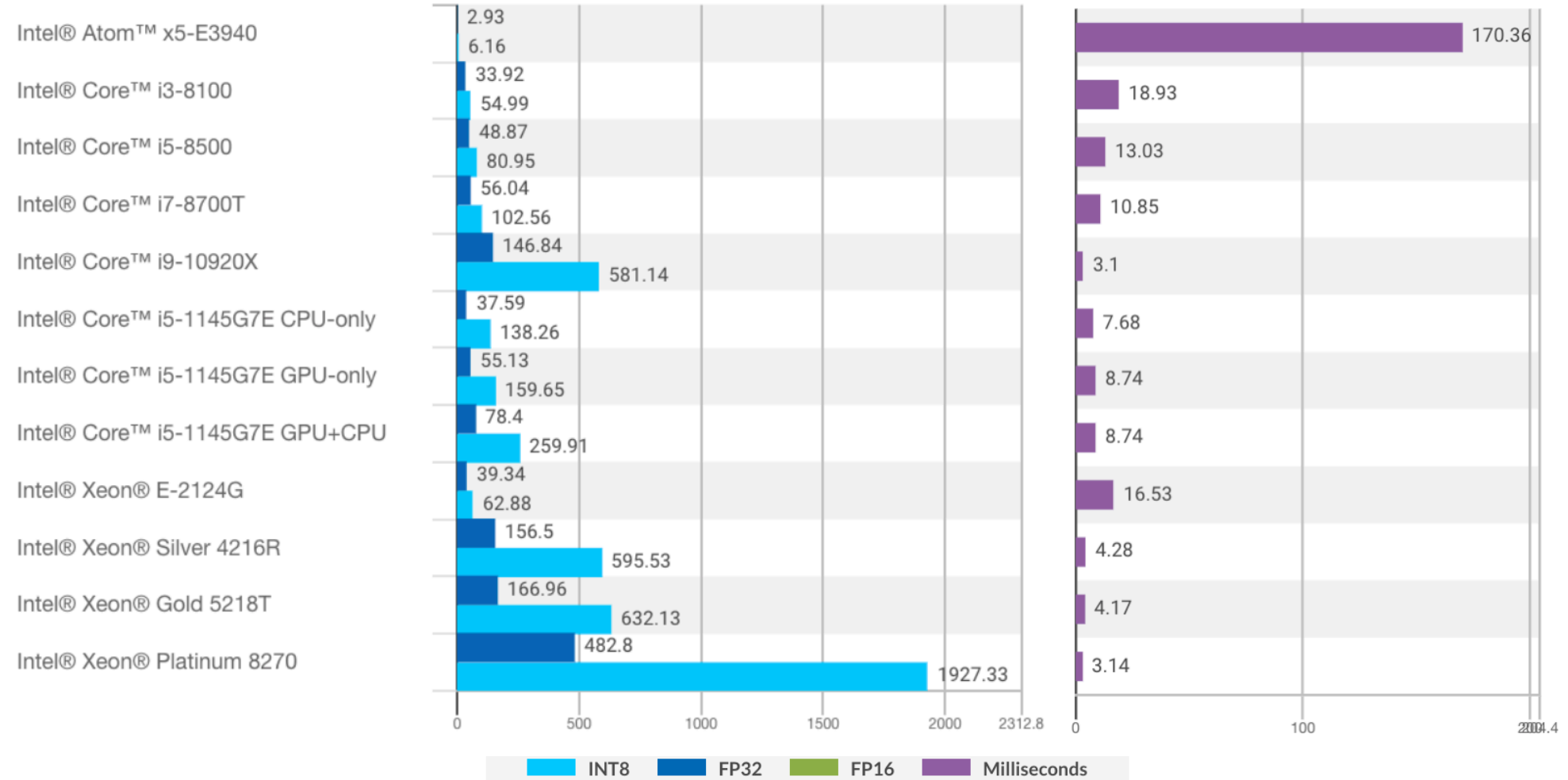


# inception-v3-TF

Throughput (higher is better)

Latency (lower is better)

## CPU INFERENCE ENGINES



Source: [https://docs.openvino toolkit.org/latest/openvino\\_docs\\_performance\\_benchmarks.html](https://docs.openvino toolkit.org/latest/openvino_docs_performance_benchmarks.html)

## Value (higher is better)

## Efficiency (higher is better)

Intel® Atom™ x5-E3940

0.146

0.647

Intel® Core™ i3-8100

0.469

0.845

Intel® Core™ i5-8500

0.421

1.245

Intel® Core™ i7-8700T

0.338

2.93

Intel® Core™ i9-10920X

0.83

3.522

Intel® Core™ i5-1145G7E CPU-only

0.447

4.937

Intel® Core™ i5-1145G7E GPU-only

0.516

5.701

Intel® Core™ i5-1145G7E GPU+CPU

0.841

9.282

Intel® Xeon® E-2124G

0.295

0.885

Intel® Xeon® Silver 4216R

0.594

2.382

Intel® Xeon® Gold 5218T

0.468

3.01

Intel® Xeon® Platinum 8270

0.26

4.7

FPS/\$

FPS/Thermal Design Power