Bring Your ML Models to the Edge with the DeGirum DeLight Cloud Platform

Shashi Chilappagari

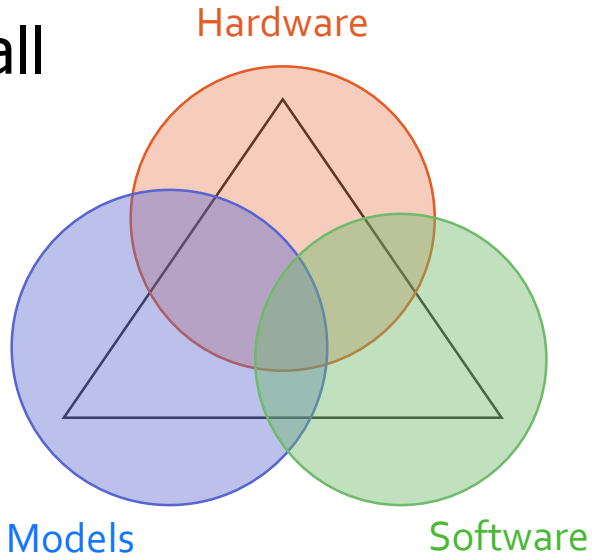
Co-Founder and Chief Architect

DeGirum Corp.

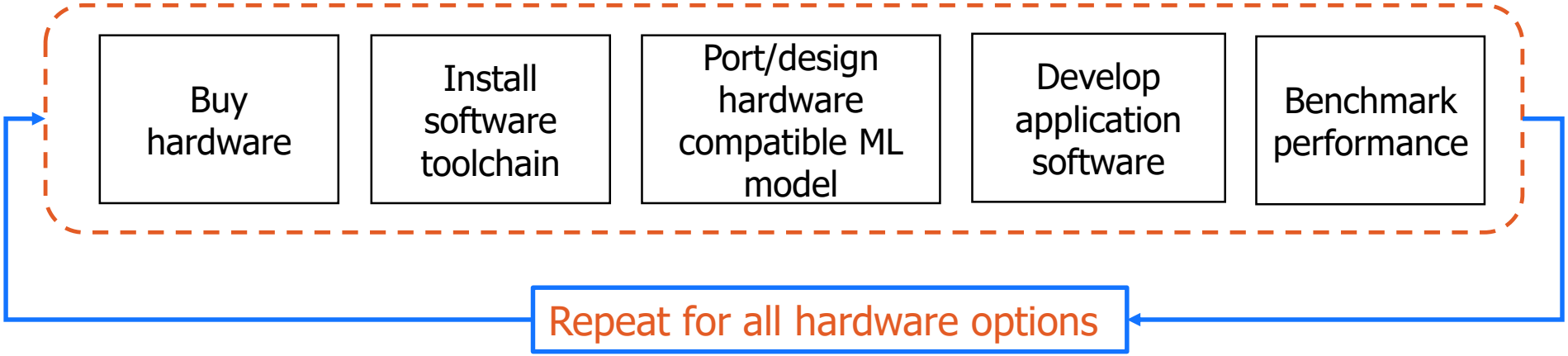
DeGirum®

Developing AI Applications for the Edge

- Not all ML models are compatible with all hardware
- Runtime software can be hardware specific
- Different types of ML models need different software



Evaluating AI Hardware Accelerators is Hard



Result: Separate application software & models for each hardware option

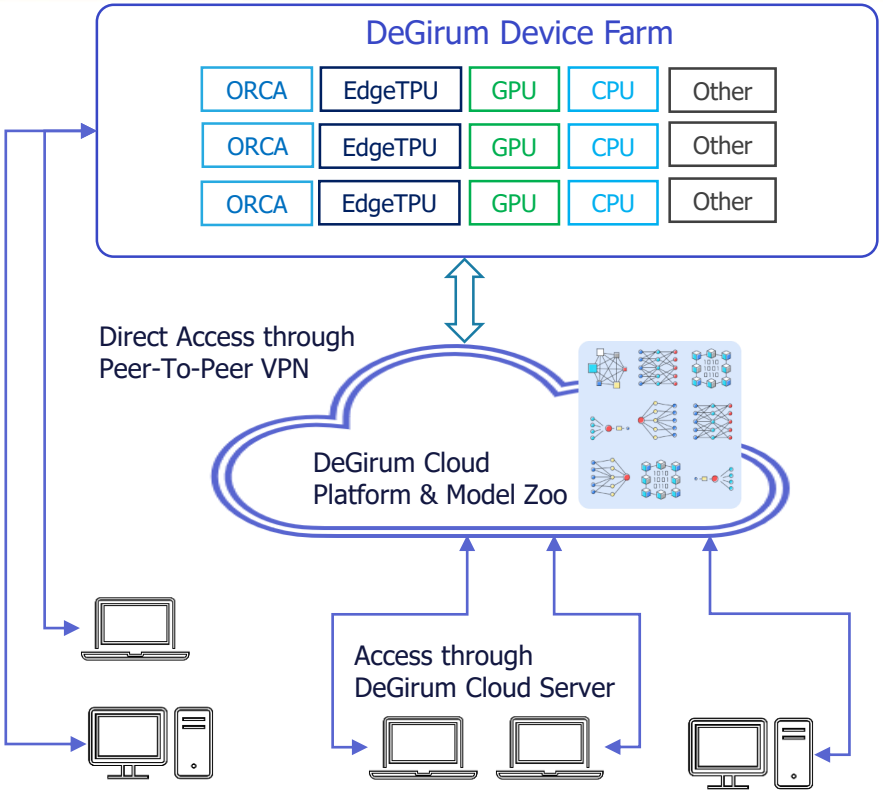
Wasted time, money, and effort

Problems with the Hardware Evaluation Process

Impossible to judge a hardware option based on datasheet

No unified application software that makes evaluation easy

DeGirum Solution: DeLight Cloud Platform



Highlights

1. Cloud access to AI hardware
2. Software that supports multiple hardware options
3. Tools to choose/port/design ML models

Supported Hardware Options

- ORCA-NNX (DeGirum): Available now
- CPU (Intel, AMD, ARM): Available now
- Edge TPU (Google): Coming soon
- GPU (Nvidia): Coming soon (Jetson and Orin series)
- Others can be added on demand

DeGirum PySDK: Sophisticated Applications with Simple APIs

```
import degirum as dg
zoo=dg.connect(ai_server, model_zoo, token)
model=zoo.load_model(model_name)
res=model(image)
res.image_overlay
```

- DeGirum cloud farm
- AI server IP address
- Local hardware

- Cloud model zoo
- Local model zoo
- Single model file

Each line in the above code solves a different problem faced by developers

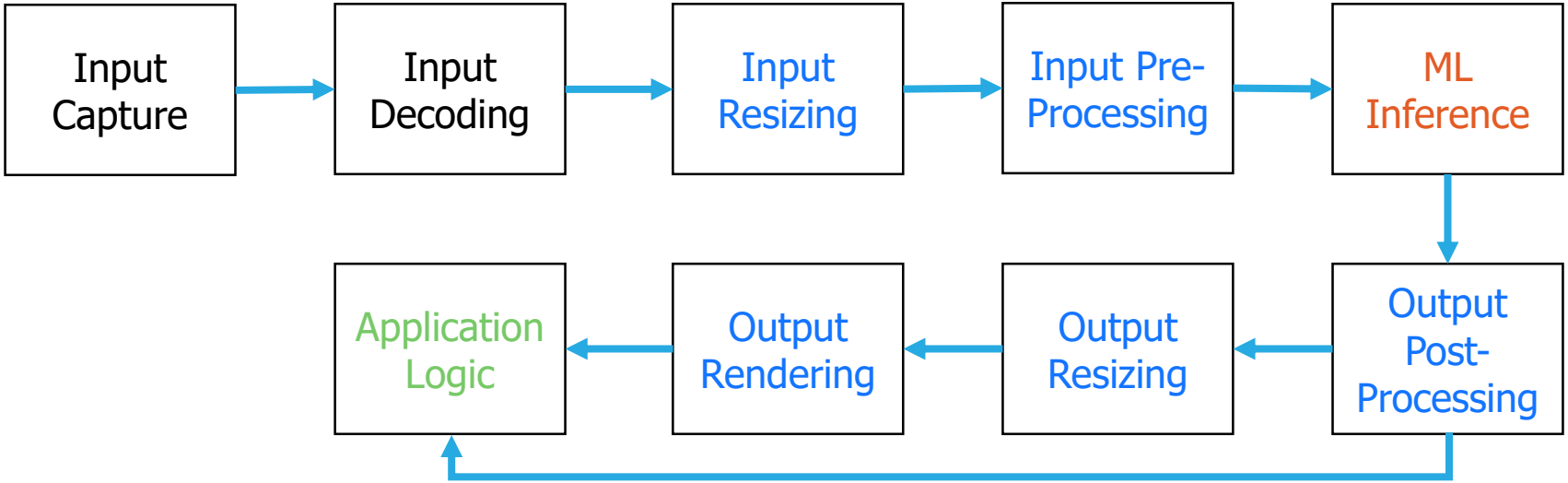
What Do Our Customers Want?

No model zoo is large enough: Developers want to bring their own models

Need tools that allow frictionless integration of trained models to application software

Challenges in Integrating ML Models into Applications

ML Application Software Pipeline



Application software is much more than just ML inference

Model Integration Challenge 1

Challenge: Efficient, accurate, and comprehensive implementation of resizing, pre-processing, and post-processing stages

Importance: Model accuracy depends on getting all implementations and parameters right

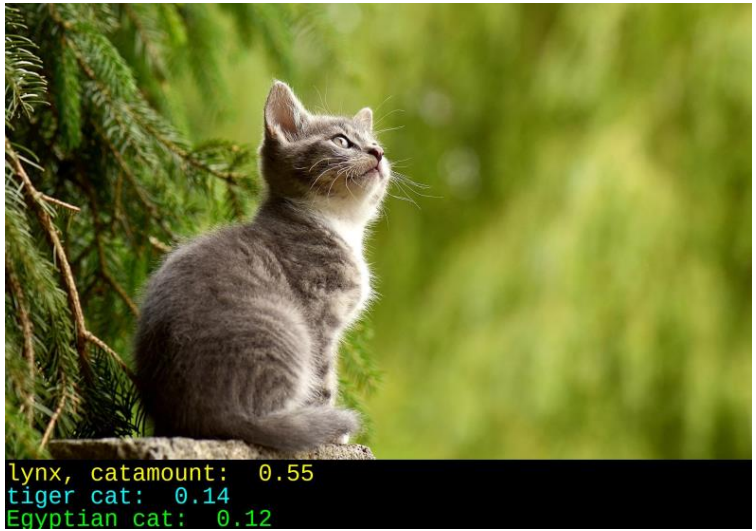
Why is it Hard to Implement these Stages?

- Using the same pre/post-processing parameters during training and inference is crucial to maintaining model accuracy when porting to new hardware
- Software needs to support various options for these parameters
  - Image backend, resize method, interpolation option, input normalization, input quantization, nms threshold, confidence threshold
- Comprehensive and accurate implementation is a lot of work!!!

How Does DeGirum PySDK Solve the Problem?

Model inference API includes decoding, resizing, pre-processing, and post-processing

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('efficientnet')
res=model('cat.jpg')
res.image_overlay
```



Model Configuration JSON File

Allows specifying different parameters


```
{
 "PRE_PROCESS": [
  {
   "InputType": "Image",
   "InputN": 1,
   "InputH": 224,
   "InputW": 224,
   "InputC": 3,
   "InputQuantEn": true,
   "InputQuantOffset": 114,
   "InputQuantScale": 0.018658,
   "InputPadMethod": "crop-last",
   "InputImgNormEn": true,
   "InputImgMean": [ 0.485, 0.456, 0.406 ],
   "InputImgStd": [  0.229, 0.224, 0.225 ],
   "InputResizeMethod": "bicubic",
   "InputCropPercentage": 0.875
  }
 ],
 "MODEL_PARAMETERS": [
  {
   "ModelPath": "efficientnet_es_orca_2.n2x"
  }
 ],
 "POST_PROCESS": [
  {
   "OutputPostprocessType": "Classification",
   "OutputTopK": 5,
   "OutputConfThreshold": 0.1,
   "LabelsPath": "labels_ILSVRC2012_1000.json"
  }
 ],
 "DEVICE": [
  {
   "DeviceType": "ORCA",
   "RuntimeAgent": "N2X"
  }
 ],
}
```

Model Integration Challenge 2

Challenge: Visualizing output of ML model

Importance: Developers need to visualize ML model output to get a qualitative sense of model performance

Why is Model Output Visualization Challenging?

- Different types of models provide different types of information
  - Image classification: Top labels and their probabilities
  - Object detection: Object categories and their locations (bounding boxes)
  - Semantic segmentation: Category label for every pixel
  - Pose detection: Key point locations
- Visualizing output for different types of models requires developing custom output rendering logic for each type

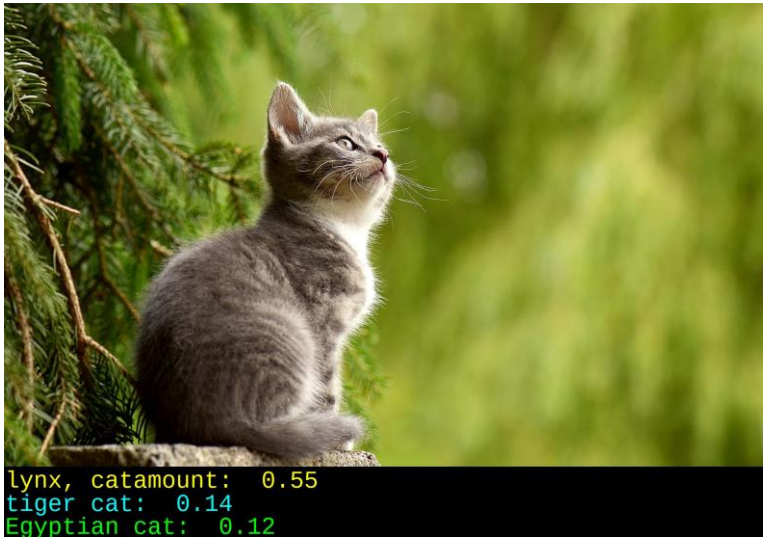
How Does DeGirum PySDK Solve the Problem?

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('yolov5s')
res=model(image)
res.image_overlay
```

Model inference results object helps visualize the output

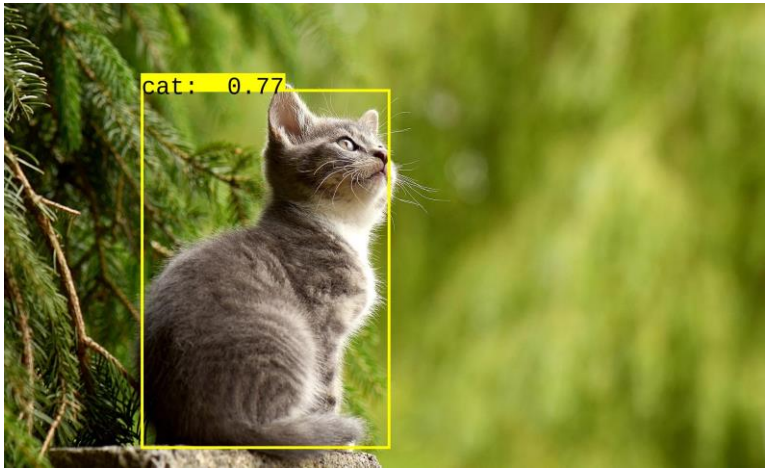
Output Visualization Using PySDK

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('efficientnet')
res=model(image)
res.image_overlay
```



Output Visualization Using PySDK

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('yolov5s')
res=model(image)
res.image_overlay
```



Output Visualization Using PySDK

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('deeplab_seg')
res=model(image)
res.image_overlay
```

Postprocessors Supported by PySDK

- Image classification
- Object detection
- Semantic segmentation
- Pose detection
- Hand landmark detection
- License plate OCR
- Custom logic via python

Model Integration Challenge 3

Challenge: Developing software that works for multiple hardware options

Importance:

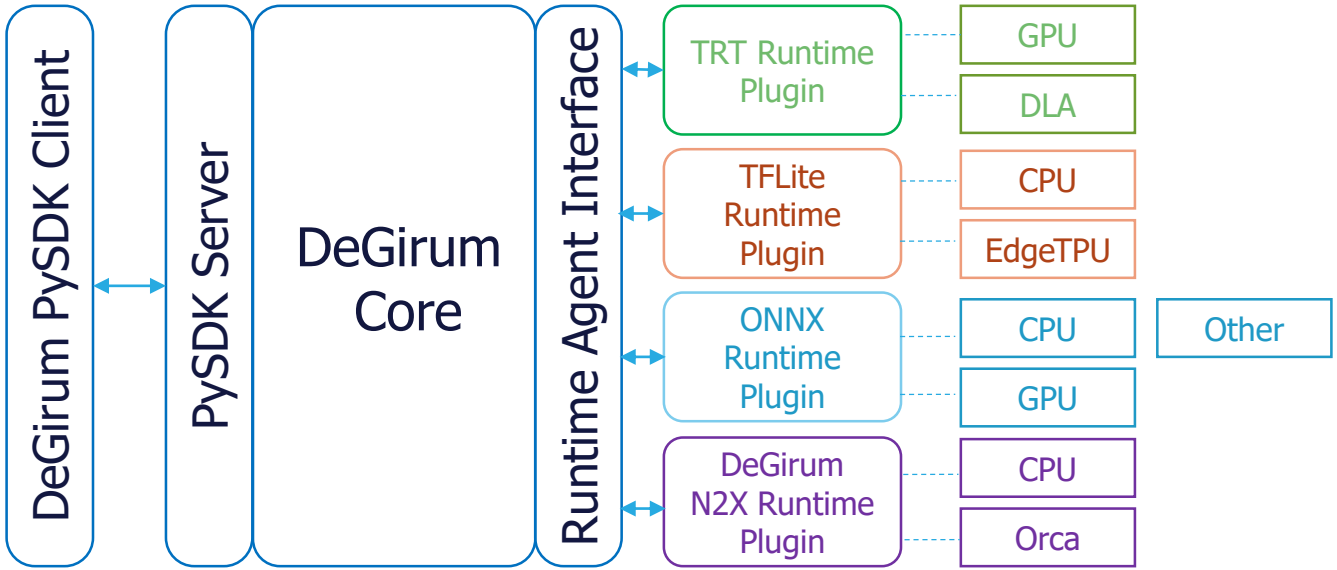
(1) Enables easy and fair comparison of hardware options

(2) Enables multiple product lines with different hardware but single software

Challenge in Developing Unified Software

- Different hardware use different runtimes
  - N2X: DeGirum
  - OpenVINO: Intel
  - TensorRT: Nvidia
  - TFLite: Google
  - ONNXRT: Microsoft
- Unifying multiple hardware options into one software requires significant amount of work

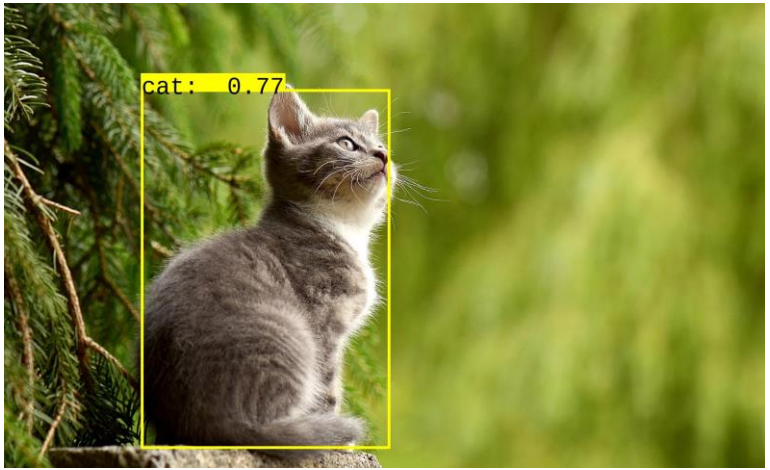
How Does DeGirum PySDK Solve the Problem?



Common runtime agents are integrated as plugins

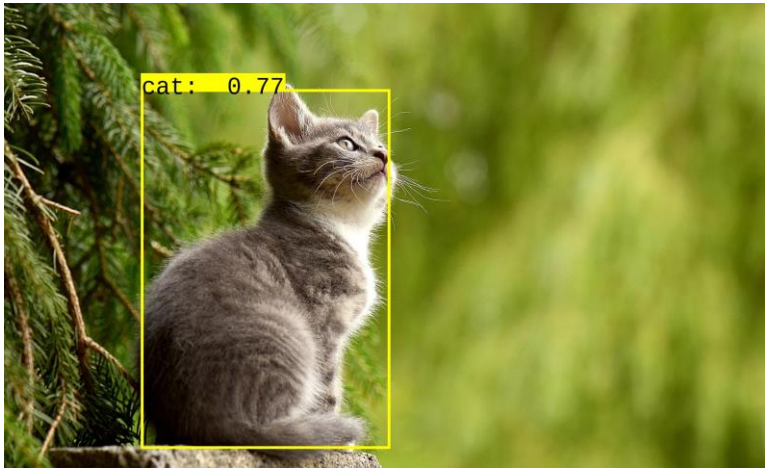
Software Supporting Multiple Hardware

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('yolov5_orca')
res=model(image)
res.image_overlay
```



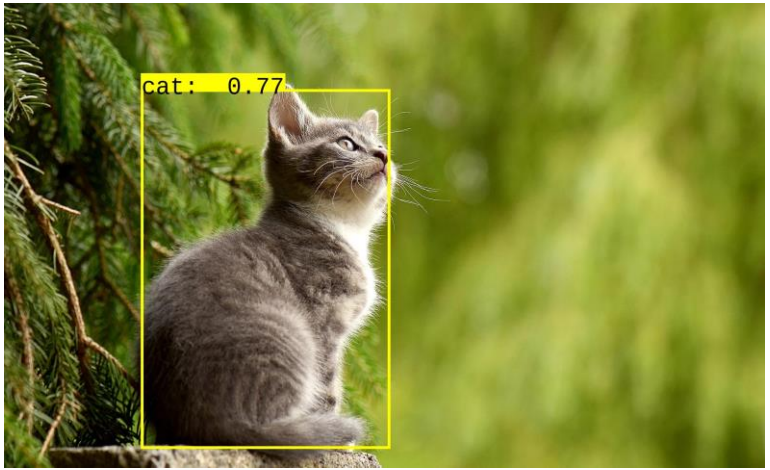
Software Supporting Multiple Hardware

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('yolov5_cpu')
res=model(image)
res.image_overlay
```



Software Supporting Multiple Hardware

```
import degirum as dg
zoo=dg.connect()
model=zoo.load_model('yolov5_gpu')
res=model(image)
res.image_overlay
```



Model Integration Challenge 4

Challenge: Efficient pipelining of all stages in the ML application

Importance:
(1) Needed for optimizing application performance
(2) Needed to ensure hardware is utilized efficiently

Challenges in Pipelining

- Software contains multiple stages with each stage running (possibly) on a different hardware resource
  - ML inference on AI hardware accelerator, application logic on host CPU, input decoding on special decoders
- Scheduling workloads for different hardware components efficiently is a difficult problem
- Balancing trade-offs between frames per second (FPS) and latency adds extra layer of complexity
- Managing multiple applications requires lot of system level testing

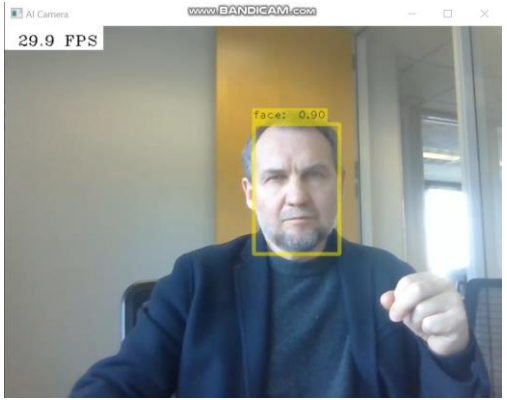
How Does DeGirum PySDK Solve the Problem?

```
import degirum as dg
import mytools

camera_id = 0
zoo=dg.connect()
model=zoo.load_model('yolo_v5s_face')

with mytools.Display("AI Camera") as display, \
     mytools.open_video_stream(camera_id) as stream:

    for res in model.predict_batch(mytools.video_source(stream)):
        display.show(res.image_overlay)
```



Summary

- DeGirum DeLight platform speeds your work in selecting processors and models, and building portable edge ML applications by providing
  - Cloud access to edge hardware
  - Software that supports multiple hardware options
  - Simple to use inference APIs
  - Tools that enable visualizing model outputs
  - Methods to optimize application performance

Save time, money, and effort

Additional Information

- Request access to DeGirum DeLight Cloud Platform at https://cs.degirum.com/
- See code examples of PySDK at https://github.com/DeGirum/PySDKExamples
- Download compiler, AI server, and AI client dockers at https://hub.docker.com/u/degirum
- Watch demos at https://www.youtube.com/@degirumcorp.5330

Resources

DeGirum Website
https://www.degirum.com/

DeGirum Cloud Platform
https://cs.degirum.com/

DeGirum GitHub Repo
https://github.com/DeGirum/PySDKExamples

2023 Embedded Vision Summit

Visit our booth!