Paris, samedi 14 décembre 2019

Ecole Supérieure de Génie Informatique

Merci à l'ESGI, aux communautés et aux sponsors

AZUGFR
Azure User Group France

ESGI
école supérieure de génie informatique

GUSS

Local sponsors

AZEO
talents & technology

cellenza
DoesITbetter

Global AI Bootcamp

Samedi 14 décembre 2019 – Paris

Custom Vision : projet de reconnaissance de l’équipement de protection

Le projet de contrôle de tenue, qui permet d’améliorer les conditions de travail dans les usines du client. Le projet comprend 2 parties : détection de visage (réalisée avec OpenCV) et classification de tenue (réalisée avec Computer Vision API de Microsoft)

Alibek JAKUPOV (EXPERTIME)

Microsoft MVP (AI)

Global AI Community

15h-16h / Track vert

Use cases

Explanation

Protective Clothing detection

Requirements: Win 10 application connected to USB camera and detecting multiple operators simultaneously

Possible solutions

Deep Learning

- CNTK; TensorFlow; Keras etc…

Cascade Classifier

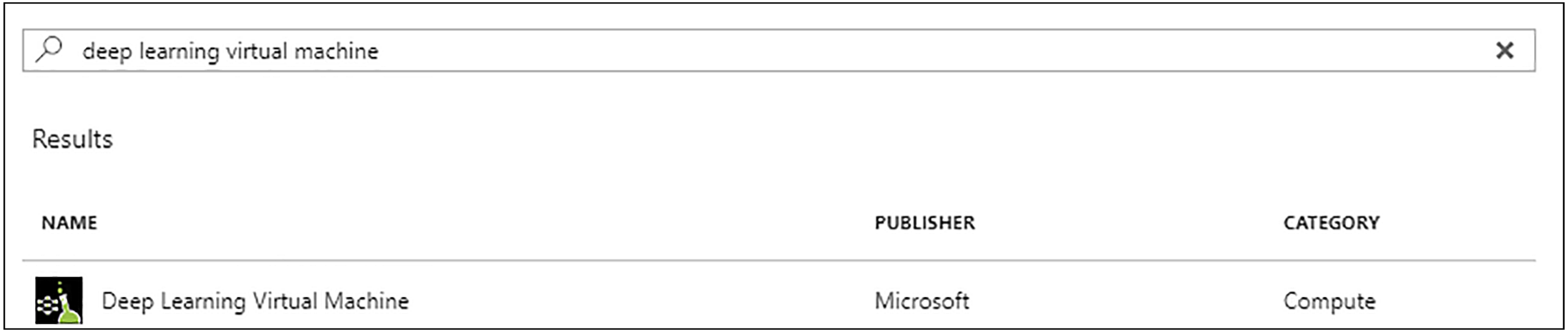
- OpenCV
- Bibliothèque open source spécialisée dans le traitement d'image en temps réel.

Transfer Learning

- Custom Vision API
- API permettant de créer, déployer et améliorer des classificateurs d'image personnalisés.
- Utilise un algorithme d'apprentissage automatique pour classifier les images.

Deep Learning

Use N-series VMs for GPU acceleration



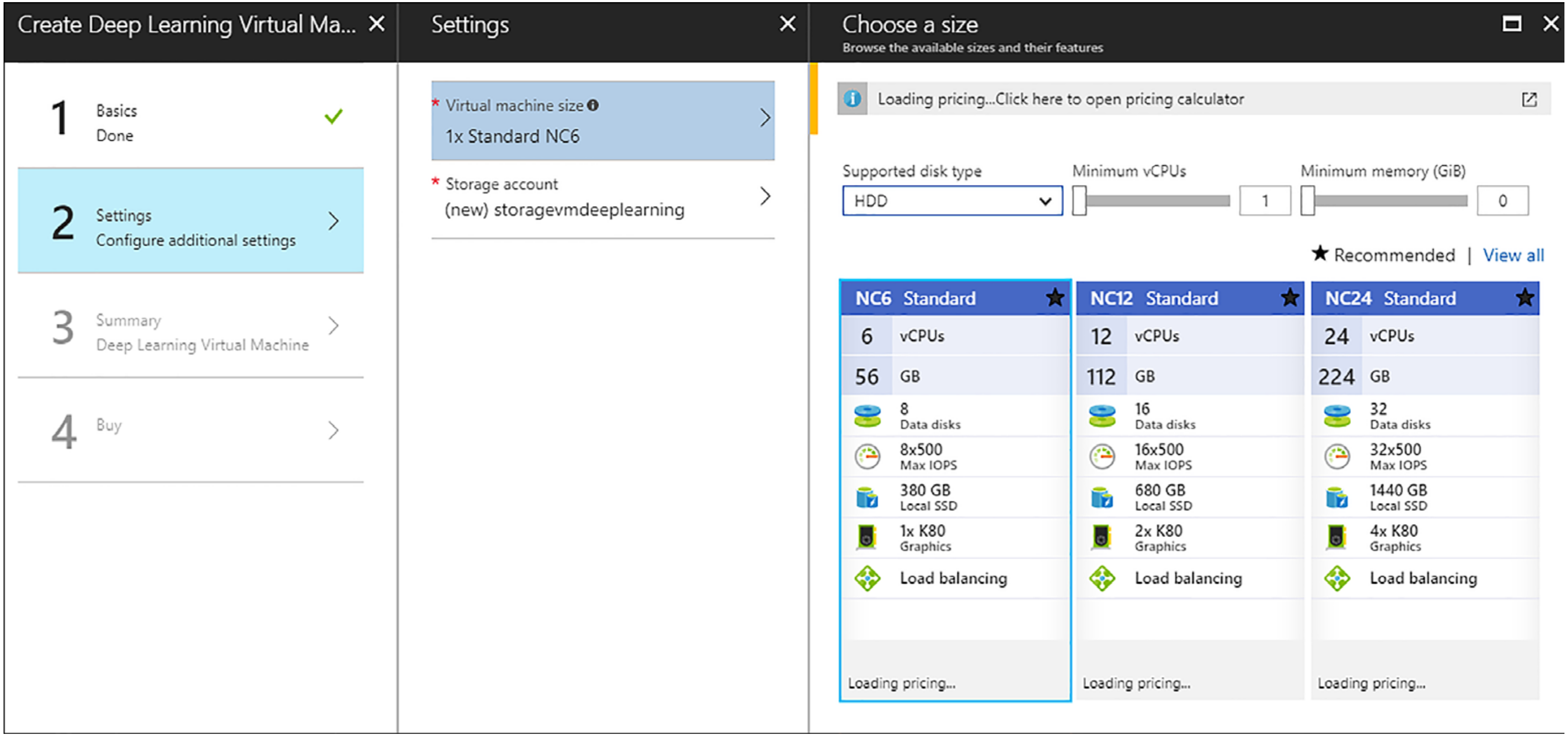
deep learning virtual machine

Results

NAME	PUBLISHER	CATEGORY
Deep Learning Virtual Machine	Microsoft	Compute

Deep Learning

Use N-series VMs for GPU acceleration



Create Deep Learning Virtual Ma... ✕

1 Basics
Done ✓

2 Settings
Configure additional settings >

3 Summary
Deep Learning Virtual Machine >

4 Buy >

Settings ✕

* Virtual machine size
1x Standard NC6 >

* Storage account
(new) storagevmdeeplearning >

Choose a size
Browse the available sizes and their features

Loading pricing...Click here to open pricing calculator

Supported disk type: HDD

Minimum vCPUs: 1

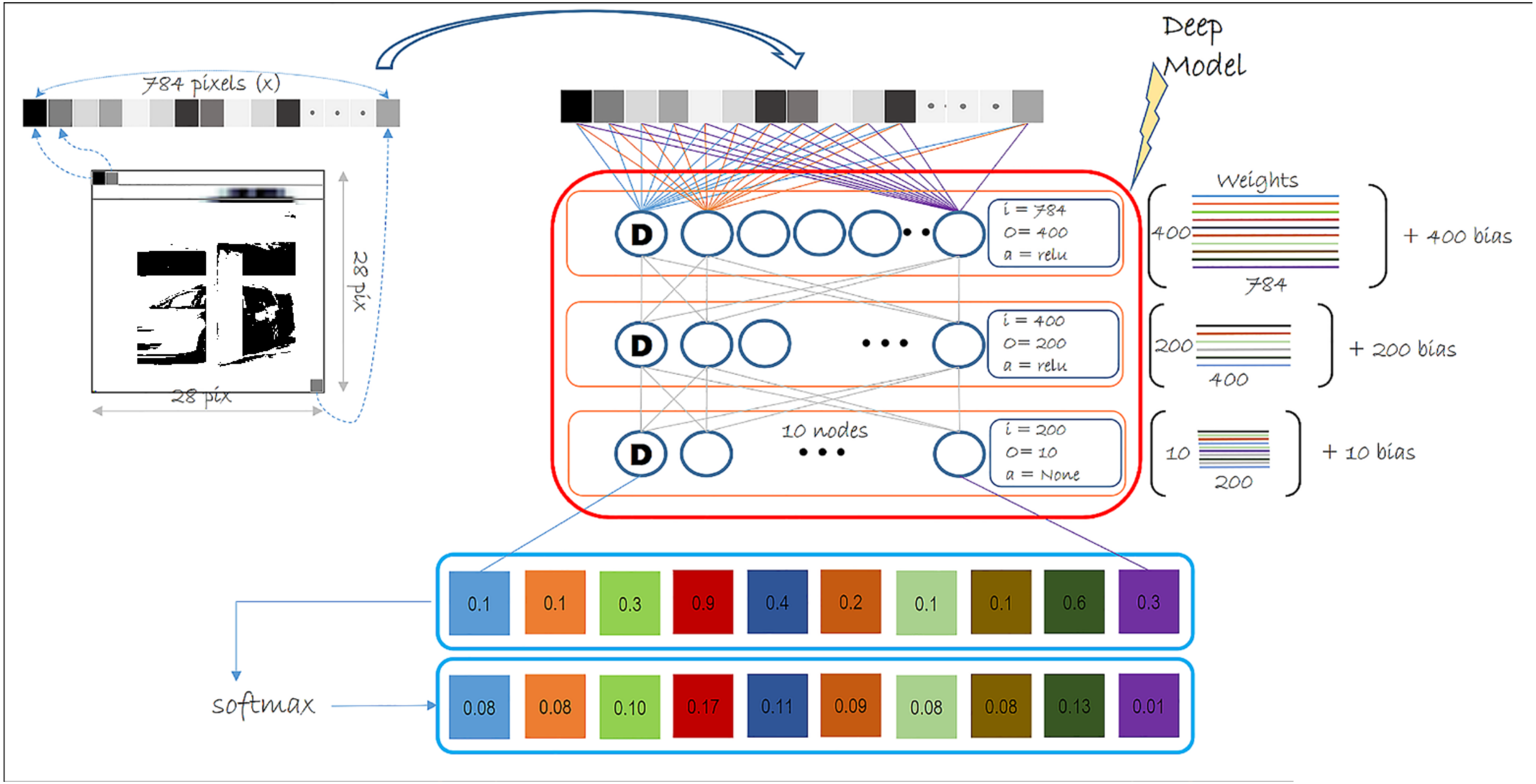
Minimum memory (GiB): 0

★ Recommended | View all

NC6 Standard ★	NC12 Standard ★	NC24 Standard ★
6 vCPUs	12 vCPUs	24 vCPUs
56 GB	112 GB	224 GB
8 Data disks	16 Data disks	32 Data disks
8x500 Max IOPS	16x500 Max IOPS	32x500 Max IOPS
380 GB Local SSD	680 GB Local SSD	1440 GB Local SSD
1x K80 Graphics	2x K80 Graphics	4x K80 Graphics
Load balancing	Load balancing	Load balancing
Loading pricing...	Loading pricing...	Loading pricing...

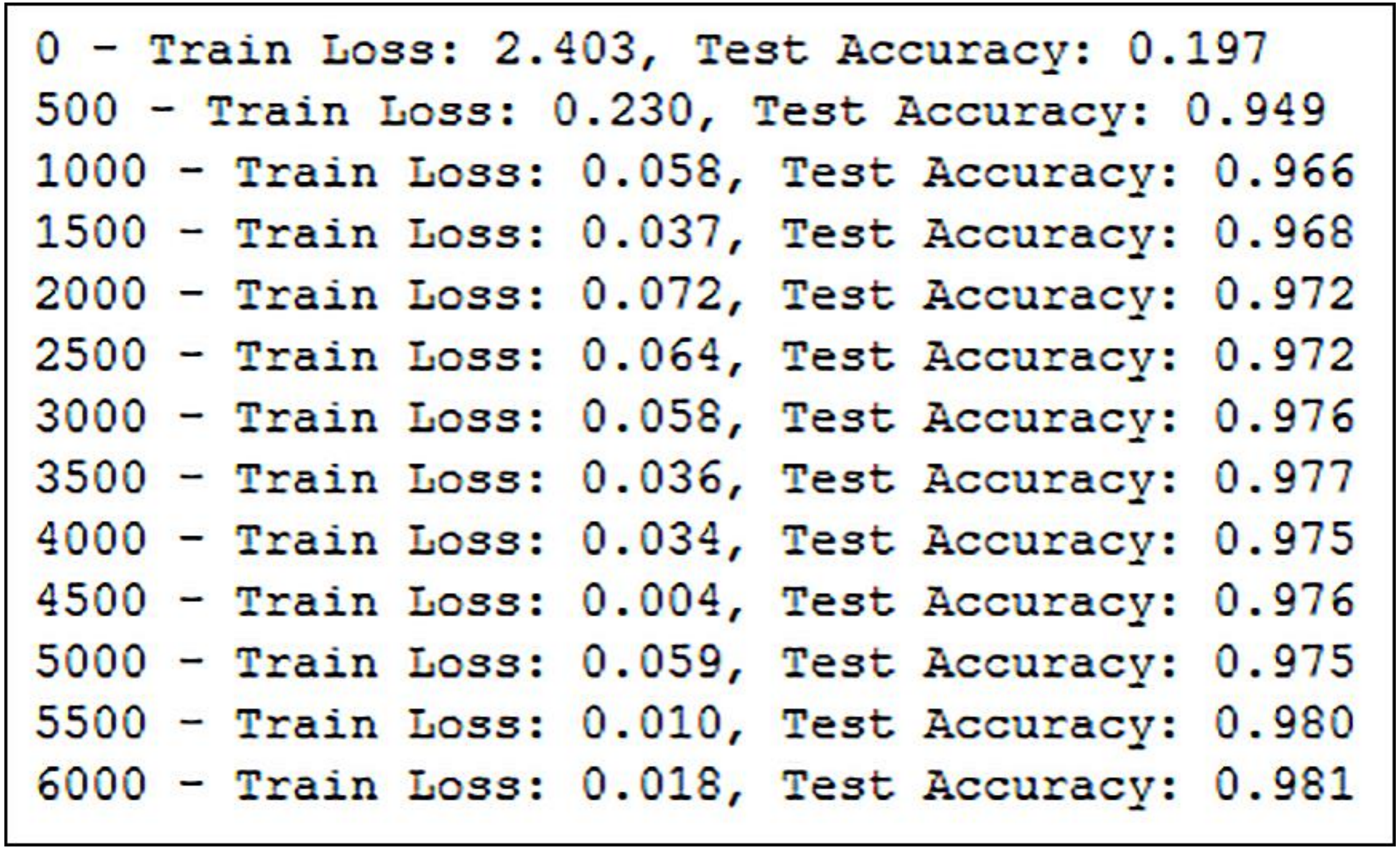
Deep Learning

Use N-series VMs for GPU acceleration



Deep Learning

Use N-series VMs for GPU acceleration



```
0 - Train Loss: 2.403, Test Accuracy: 0.197
500 - Train Loss: 0.230, Test Accuracy: 0.949
1000 - Train Loss: 0.058, Test Accuracy: 0.966
1500 - Train Loss: 0.037, Test Accuracy: 0.968
2000 - Train Loss: 0.072, Test Accuracy: 0.972
2500 - Train Loss: 0.064, Test Accuracy: 0.972
3000 - Train Loss: 0.058, Test Accuracy: 0.976
3500 - Train Loss: 0.036, Test Accuracy: 0.977
4000 - Train Loss: 0.034, Test Accuracy: 0.975
4500 - Train Loss: 0.004, Test Accuracy: 0.976
5000 - Train Loss: 0.059, Test Accuracy: 0.975
5500 - Train Loss: 0.010, Test Accuracy: 0.980
6000 - Train Loss: 0.018, Test Accuracy: 0.981
```

Deep Learning

+

Pros:

Flexibility

Precision

−

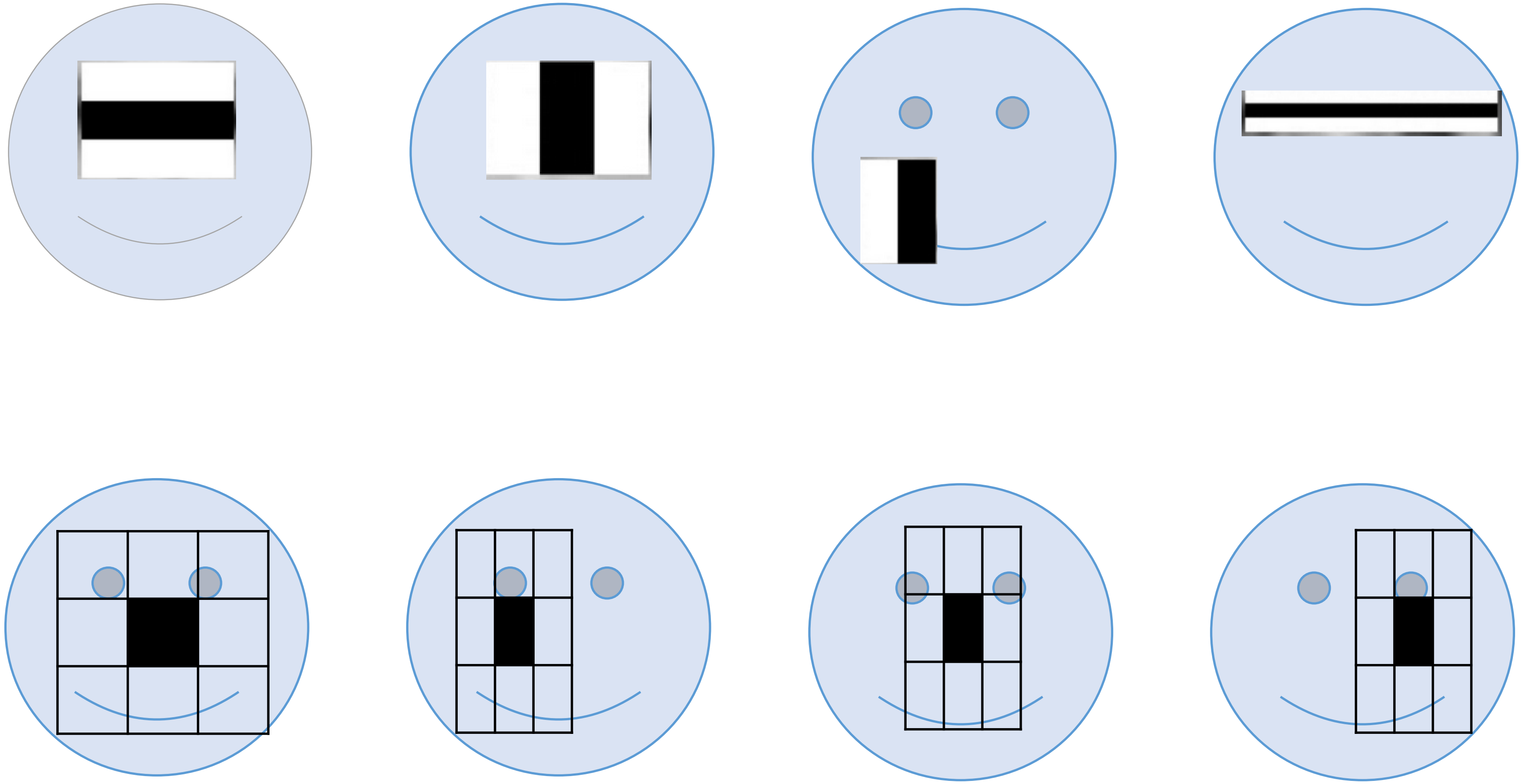
Cons:

Time

Complexity

Large data set

Cascade Classifier



Cascade Classifier

+

Pros:

Speed

Open-source

−

Cons:

Precision

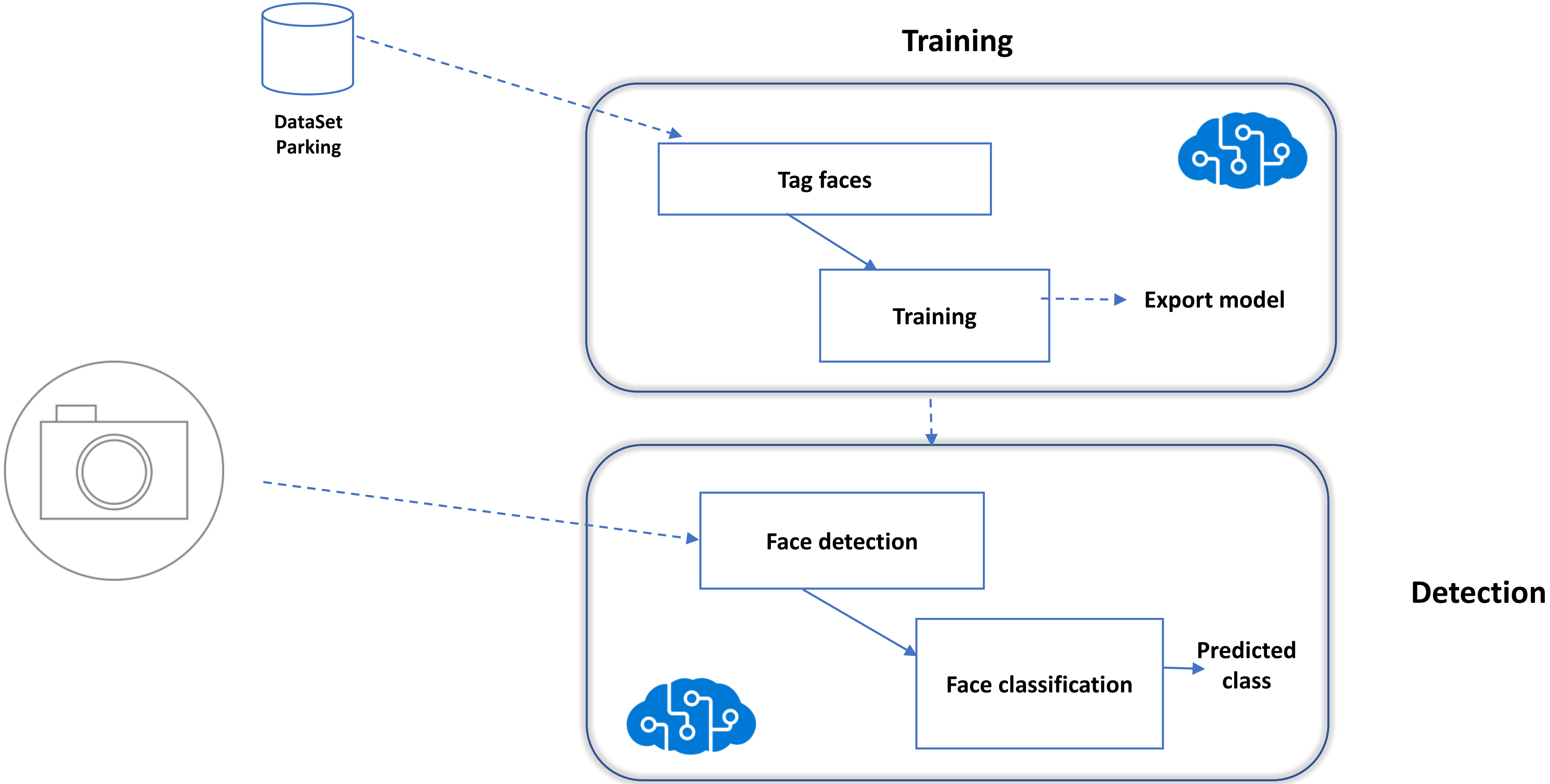
Programming skills required

Image processing skills required

Large training set needed

No user-friendly interface

Transfer Learning



Transfer Learning

+

Pros:

Speed

High precision

Small dataset required

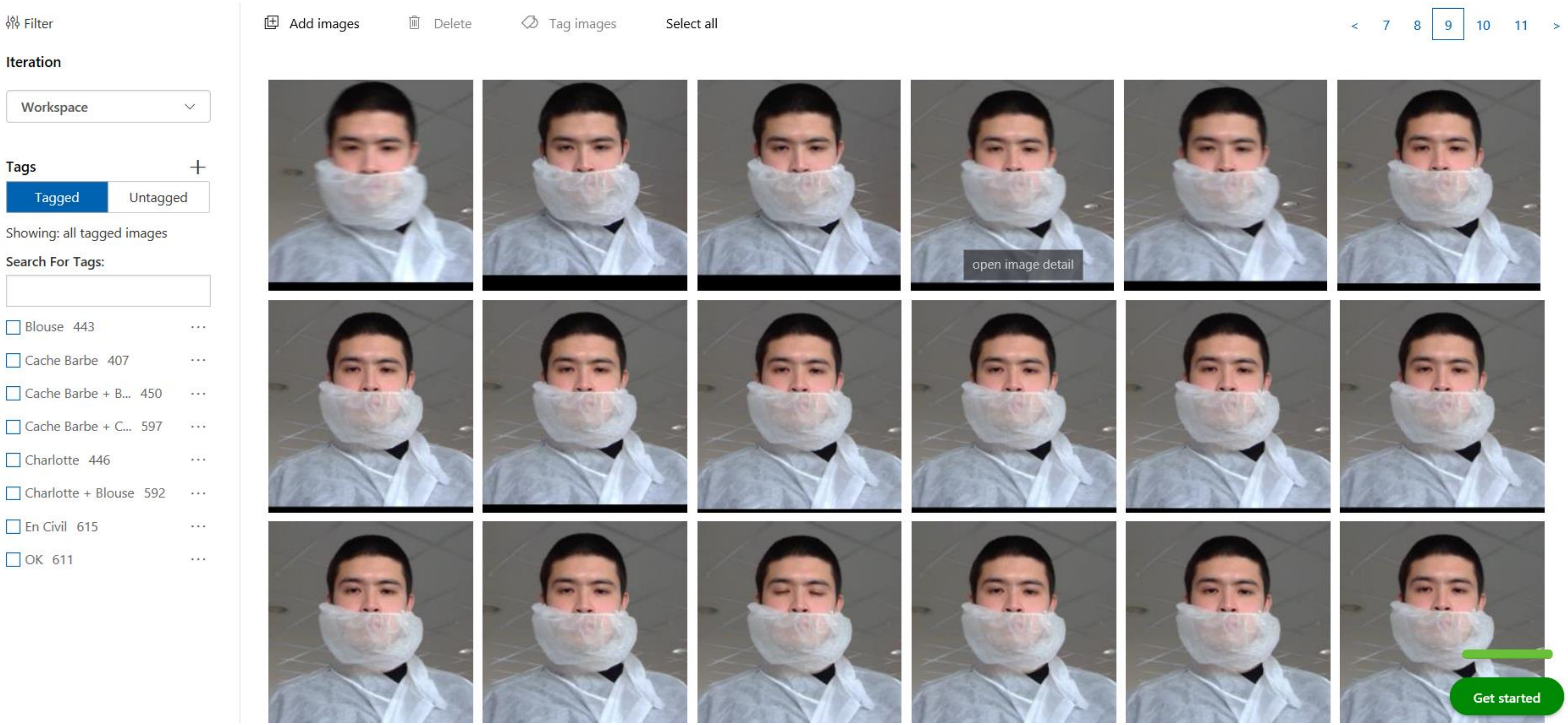
User-friendly interface

−

Cons:

Certain limits in model complexity

Transfer Learning



Iteration

Workspace

Tags +

Showing: all tagged images

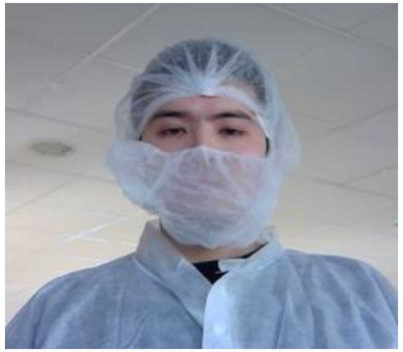
Search For Tags:

- □ Blouse 443
- □ Cache Barbe 407
- □ Cache Barbe + B... 450
- □ Cache Barbe + C... 597
- □ Charlotte 446
- □ Charlotte + Blouse 592
- □ En Civil 615
- □ OK 611

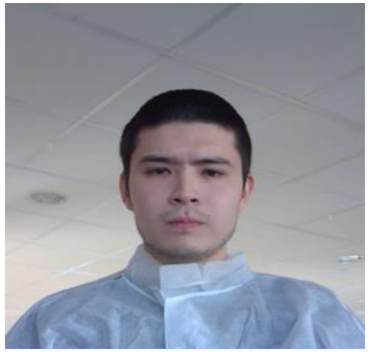
Trainset Generation

Dataset Generation

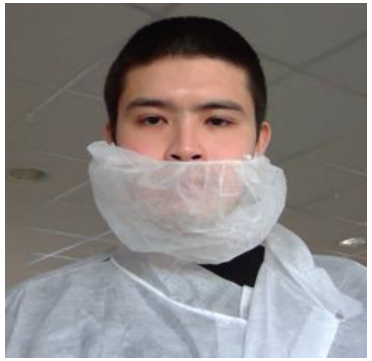
OK



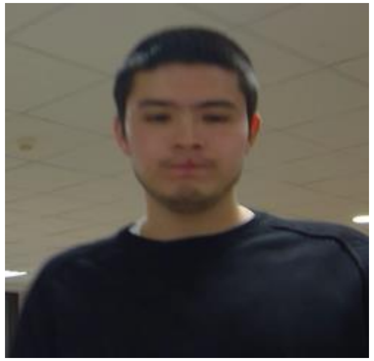
Blouse



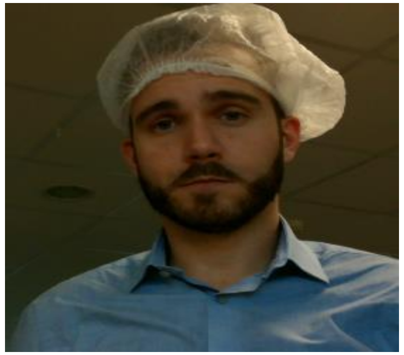
Cache Barbe + Blouse



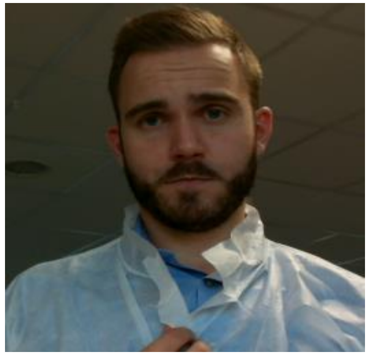
En civil



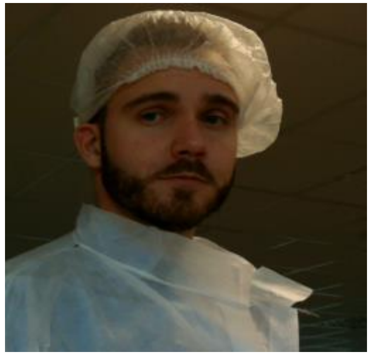
Charlotte



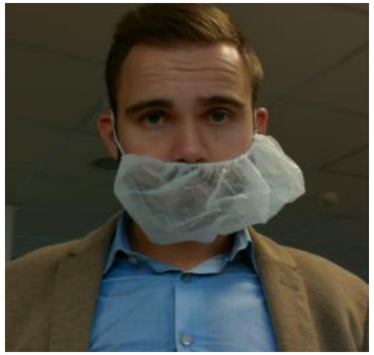
Blouse

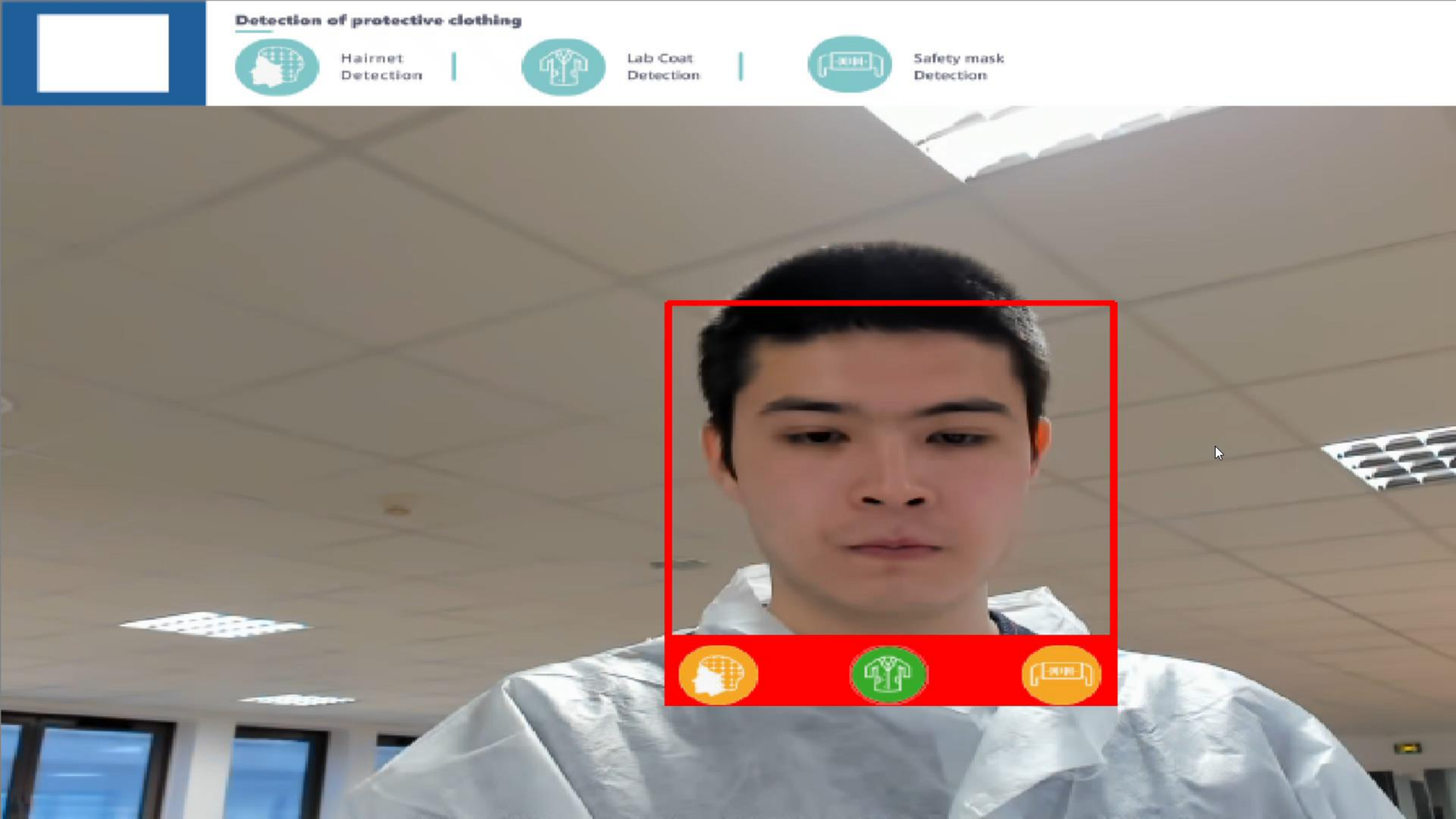


Blouse + Charlotte



Cache Barbe




Detection of protective clothing
Hairnet Detection
Lab Coat Detection
Safety mask Detection

Important points

Problems

- True positives
- Distance/light condition
- Usage scenarios
- Computing speed

Tips and Tricks

Tips and Tricks

- Using Python 3.6 on Windows
- Cold-light
- Define optimal camera distance
- Screen size

Discussions

Alternative and complementary bricks and services

- Improve face detection
- Improve processing speed
- Improve UX
- Improve quality

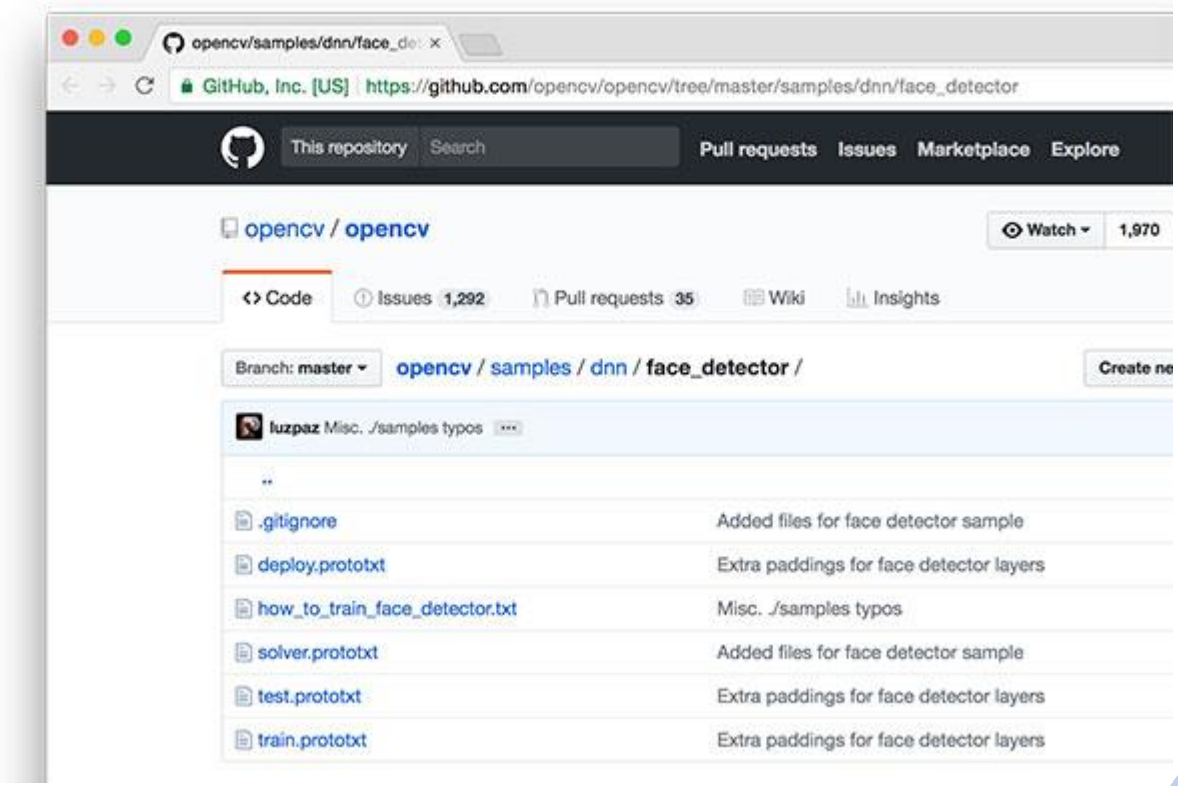
Improve face detection

August 2017, OpenCV 3.3 was officially released, bringing it with it a highly improved “deep neural networks” (dnn) module.

Improve face detection

However, what most OpenCV users do not know is that Rybnikov has included a more accurate, deep learning-based face detector included in the official release of OpenCV

https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/



https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector

opencv / opencv

opencv / samples / dnn / face_detector /

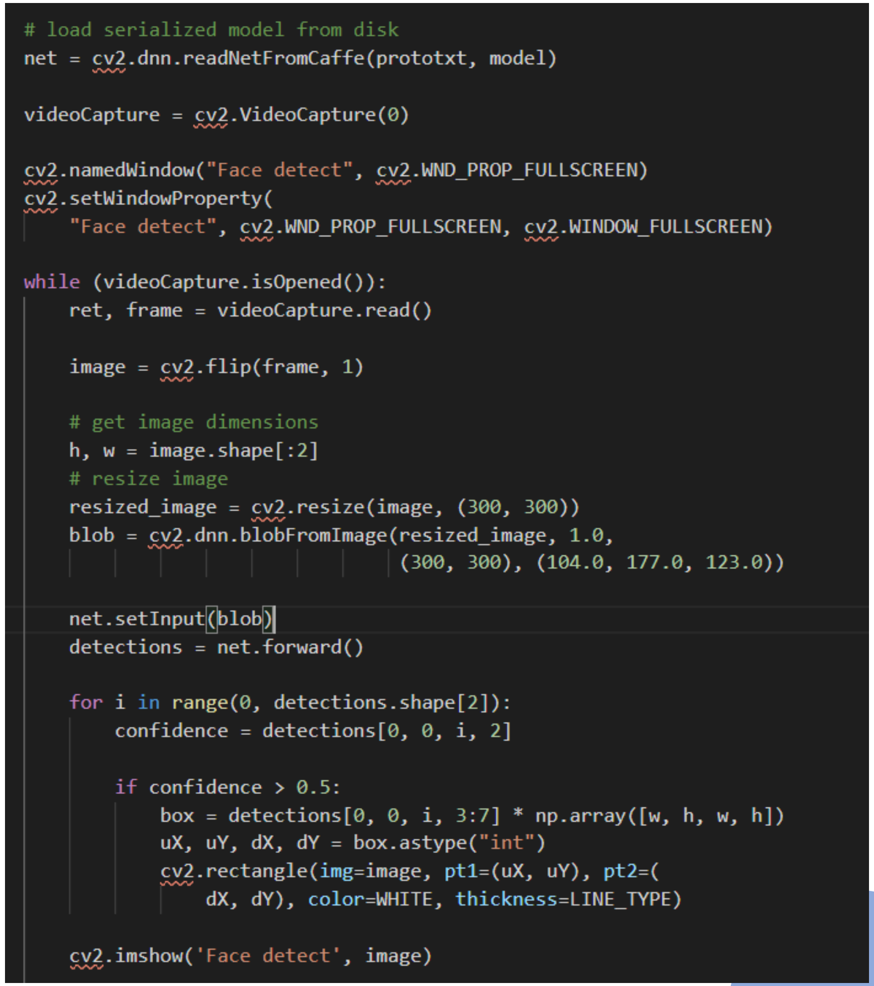
luzpaz Misc. ./samples typos

..

.gitignore	Added files for face detector sample
deploy.prototxt	Extra paddings for face detector layers
how_to_train_face_detector.txt	Misc. ./samples typos
solver.prototxt	Added files for face detector sample
test.prototxt	Extra paddings for face detector layers
train.prototxt	Extra paddings for face detector layers

Improve face detection

- Caffe prototxt files
- Caffe model weight files



```
# load serialized model from disk
net = cv2.dnn.readNetFromCaffe(prototxt, model)

videoCapture = cv2.VideoCapture(0)

cv2.namedWindow("Face detect", cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty(
    "Face detect", cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)

while (videoCapture.isOpened()):
    ret, frame = videoCapture.read()

    image = cv2.flip(frame, 1)

    # get image dimensions
    h, w = image.shape[:2]
    # resize image
    resized_image = cv2.resize(image, (300, 300))
    blob = cv2.dnn.blobFromImage(resized_image, 1.0,
                                 (300, 300), (104.0, 177.0, 123.0))

    net.setInput(blob)
    detections = net.forward()

    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            uX, uY, dX, dY = box.astype("int")
            cv2.rectangle(img=image, pt1=(uX, uY), pt2=(
                dX, dY), color=WHITE, thickness=LINE_TYPE)

    cv2.imshow('Face detect', image)
```

Improve processing speed

- Tensorflow model on Linux run faster
- Full screen works better

Improve UX

- Use Electron JS to launch python scripts
- Create local server for prediction
- Add additional steps for verification

Improve quality

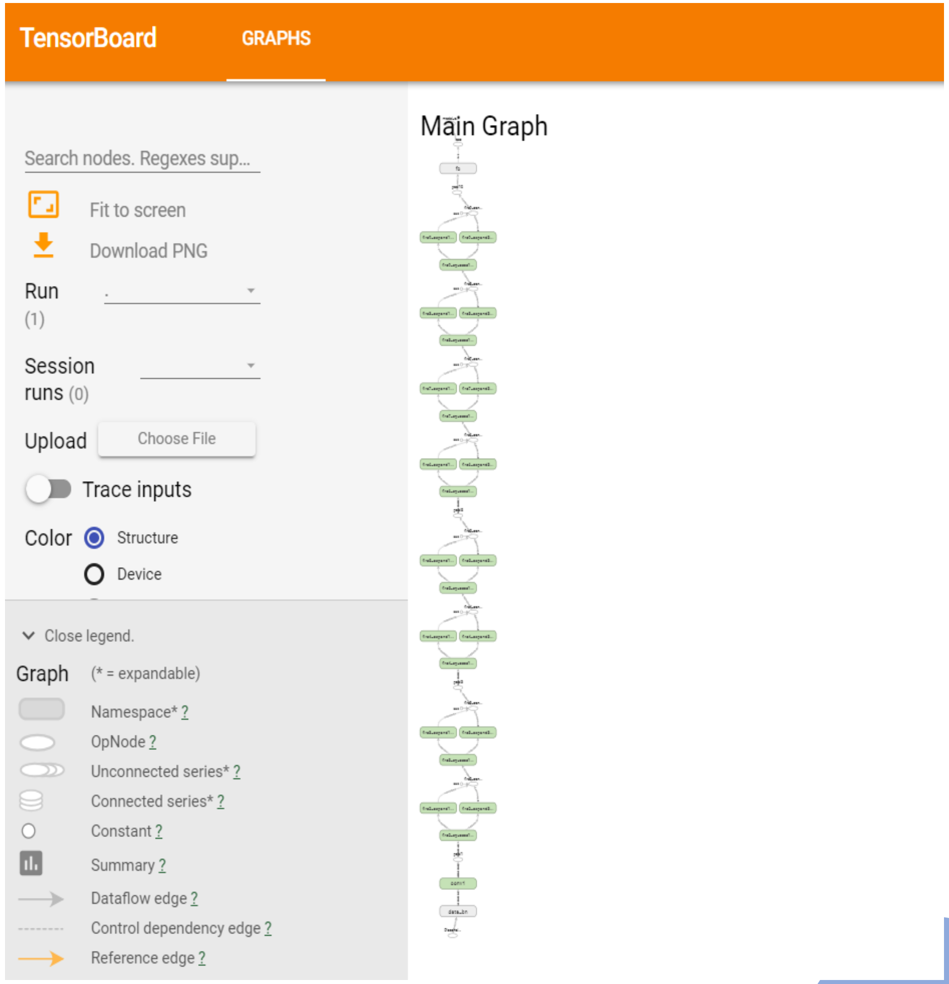
- Split model into two for better focus
- Add more data

Visualize Graph

pip install tensorboard

tensorboard --logdir=log

TensorBoard 1.11.0 at http://<computer-name>:6006 (Press CTRL+C to quit)



Edge Computing

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	1x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I2C, I2S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

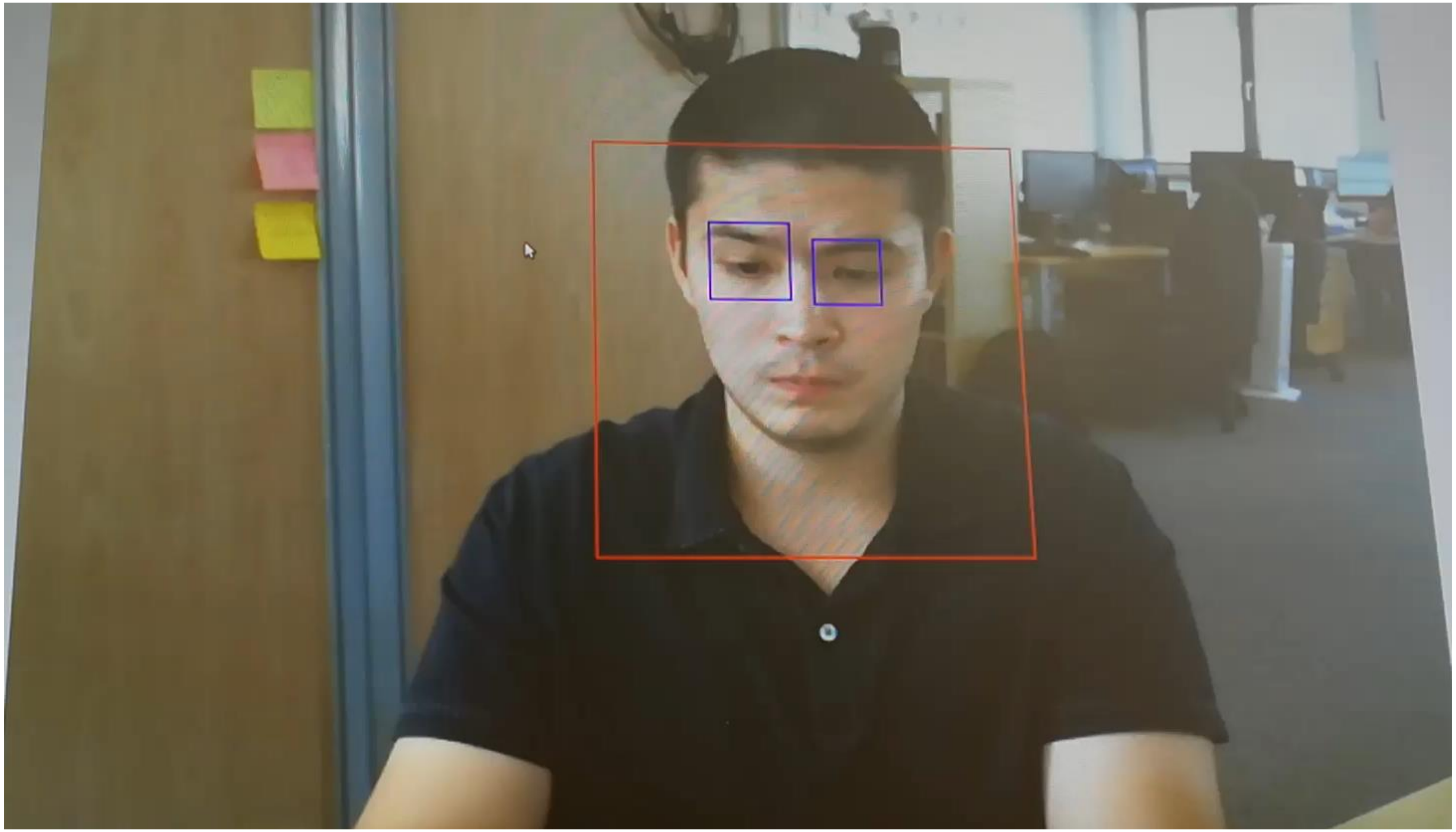
Edge Computing

Rewrite code on c++ using libjetson-inference library

Edge Computing

Edge Computing

Edge Computing



Paris, samedi 14 décembre 2019

Ecole Supérieure de Génie Informatique