

zant

October 4-5

TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Focus AI

cinela

Team 4, Participant names

with the support of:



TOOLBOX

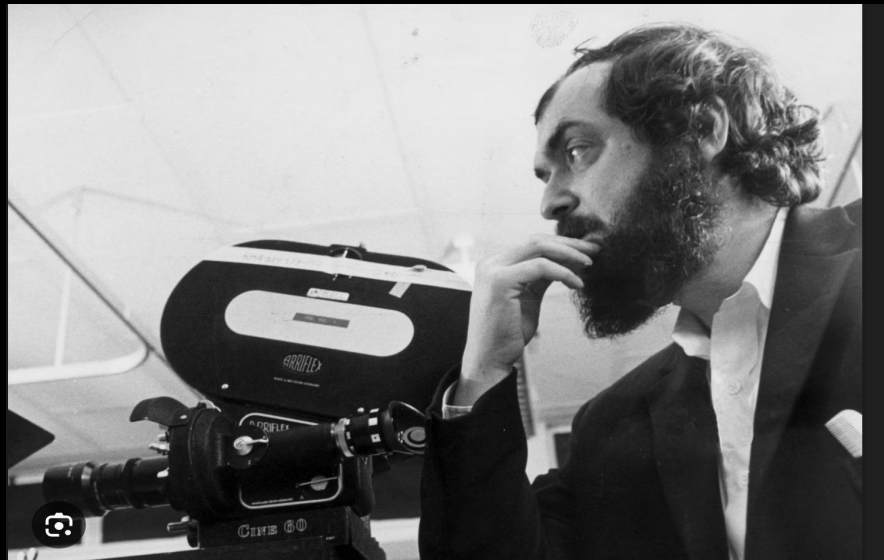
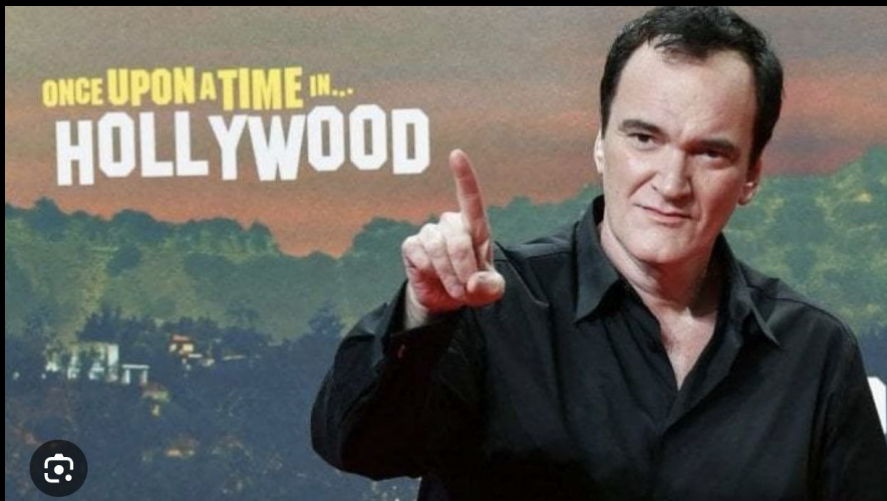
Datapizza

# TINY\_HACK

Turin

The first Embedded AI Vision hackathon

## Imagine being a film director in Hollywood



with the support of:



TOOLBOX

The Datapizza logo, featuring a stylized blue and white icon resembling a pizza slice or a fan, followed by the word 'Datapizza'.

## TINY\_HACK

Turin

The first Embedded AI Vision hackathon

You are disrupting the industry  
by learning what **emotions people  
feel** while watching your movie  
BASED ON DATA

with the support of:



TOOLBOX



zant

October 4-5

# TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Focus AI



You want to put a  
**camera** in front of  
every single **seat** and  
record the  
**impressions of the**  
**watchers** while  
watching your movie  
live

with the support of:



TOOLBOX

Datapizza

zant

October 4-5

# TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Focus AI



This is UCI  
Cinema in Turin

with the support of:



TOOLBOX

Datapizza



## INFORMAZIONI



< Informazioni Servizi Prezzi Come arrivarci Carta del docente >

	POLTRONE	POSTI DISABILI	POSTI TOTALI
SALA 1	139	1	140
SALA 2	139	1	149
SALA 3	136	1	137
SALA 4	139	1	140
SALA 5	276	2	278
SALA 6	698	4	702
SALA 7	276	2	278
SALA 8	139	1	140
SALA 9	136	1	137
SALA 10	139	1	140
SALA 11	139	1	140

## INFORMAZIONI AGGIUNTIVE

Se sei alla ricerca di un cinema a Torino? Vieni da UCI Cinemas Torino Lingotto in via Nizza 262.

Nelle 11 sale troverai sistema audio Dolby Surround, aria condizionata, poltrone comode e posti riservati per disabili.

Oltre al grande cinema, UCI ti offre anche area relax, bar e angolo caramelle, più un'area per organizzare feste di compleanno.

Oltre alle ultime novità cinematografiche nazionali e internazionali avrai la possibilità di assistere a matinée domenicali e enquire eventi live via satellite come concerti e manifestazioni sportive.

It has 11  
rooms with  
140 seats  
each.

Total: 1540  
seats.

Filtra

1 &gt;

zant

October 4-5

# TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Focus AI

# 1540 SEATS!!!!

with the support of:



TOOLBOX

Datapizza

## TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Imagine 1540 heavy  
online ML processes at  
the same time!!!

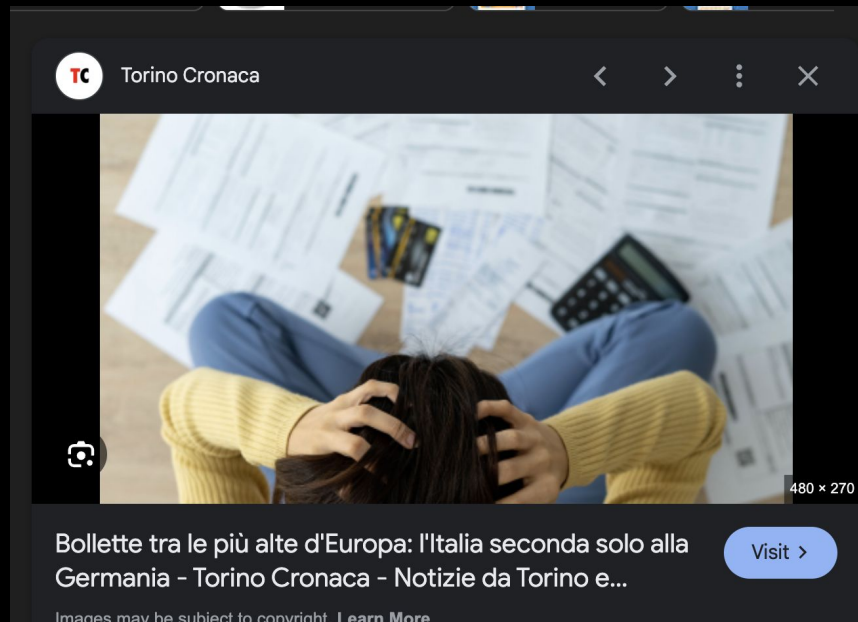
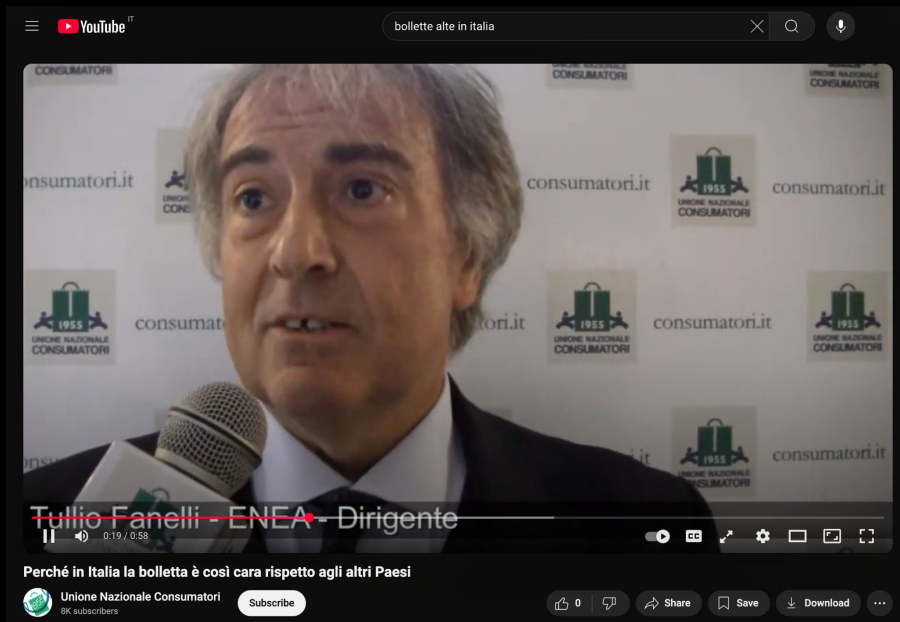
with the support of:



TOOLBOX







with the support of:



TOOLBOX

Datapizza

zant

October 4-5

TINY\_HACK

Turin

The first Embedded AI Vision hackathon

Focus AI

But luckily we have



The Zant Project

Focus AI



with the support of:



TOOLBOX

Datapizza

# TINY\_HACK

Turin

The first Embedded AI Vision hackathon

So we put 1 **Arduino NICLA VISION** in front of every seat and run **FOC00S** models compiled with **Zant** to do ML and collect the data



with the support of:



TOOLBOX

 Datapizza

# 1. Project Overview

Cinecla connects a Nicla to every spectator in a cinema, to capture emotions and viewers' responses during a movie

Two use cases:

- it helps film makers understand how the audience reacts to their movies (they can understand which scenes resonate, bore, shock, or move the audience)
- each spectator can access a short video showing their own reactions

## 2. Dataset & Model



### Dataset:

we merged relevant face emotion datasets

- collection: face emotion dataset on roboflow
- augmentation: horizontal flip, rotation, zoom out



### Model architecture:

- fai-cls-s-coco: small multiclass classification model



### Trade-offs:

accuracy vs. image and model size

## 2. Dataset & Model



## 2. Dataset & Model

TINY\_HACK

```
quantize.py (~Developer/misc/cinecla/4-cinecla-daniel-manfredi-francesco) - Nvim (nvim)

* quantize.py
from fcoos import ModelManager, ASSETS_DIR, MODELS_DIR, RuntimeType
from fcoos import ModelManager, MODELS_DIR, RuntimeType
from fcoos.infer.quantizer import OnnxQuantizer, QuantizationCfg
from fcoos.infer.infer_model import InferModel
from PIL import Image
import os

ASSETS_DIR = "final_grouped"

image_size = 96
model_name = "hub://863972ce265d4922"
im = os.path.join(ASSETS_DIR, "valid", "neutral", "44.jpg")

model = ModelManager.get(model_name)

exported_model = model.export(runtime_type=RuntimeType.ONNX_CPU, # optimized for edge or cpu
                              image_size=image_size,
                              dynamic_axes=False, # quantization need static axes!
                              simplify_onnx=False, # simplify and optimize onnx model graph
                              onnx_opset=18,
                              out_dir=os.path.join(MODELS_DIR, "cinecla_quant"))

# benchmark onnx model
exported_model.benchmark(iterations=100)

# test onnx model

result = exported_model.infer(im, annotate=True)
Image.fromarray(result.image)

quantization_cfg = QuantizationCfg(
    size = image_size, # input size: must be same as exported model
    calibration_images_folder = "calibration_images", # Calibration images folder: It is strongly recommended
                                                         # to use the dataset validation split on which the model was
                                                         # trained.
                                                         # Here, for example, we will use the assets folder.
    format="QDQ", # QD (QOperator): All the quantized operators have their own ONNX definitions, like
                  # QLinearConv, MatMulInteger etc.
                  # QDQ (Quantize-DeQuantize): inserts DeQuantizeLinear(QuantizeLinear(tensor)) between the
                  # original operators to simulate the quantization and dequantization process.
    per_channel=True, # Per-channel quantization: each channel has its own scale/zero-point -> more accurate,
                     # especially for convolutions, at the cost of extra memory and computation.
    normalize_images=True, # normalize images during preprocessing: some models have normalization outside of
                           # model forward
)

quantizer = OnnxQuantizer(
    input_model_path=exported_model.model_path,
    cfg=quantization_cfg
)

model_path = quantizer.quantize(
    benchmark=True # benchmark bot fp32 and int8 models
)

quantized_model = InferModel(model_path, runtime_type=RuntimeType.ONNX_CPU)

res = quantized_model.infer(im, annotate=True)
Image.fromarray(res.image)
```



### 3. Deployment Pipeline

We have a **Flask backend** communicating via websocket to **React frontend**, and then we just listen to json streams from the NICLAS on serial ports.

The NICLAS send json objects containing the frame raw data and the inference.

## 4. Demo & User Experience

live demo:

<https://youtu.be/oKET4S8TAfw>

screen recording:

<https://youtu.be/mRasYDhN6X0>

While the viewers  
are watching the  
video, the Nicla  
**captures their  
emotions** and  
streams them to a  
client

TINY\_HACK

## Cinema Impressions

[View Dashboard →](#)

### Device 1

😮 Surprised

😮 Surprised

0:25

😮 Surprised

0:24

😮 Surprised

0:23

😮 Surprised

0:22

### Device 2

😮 Impressed



😮 Impressed

0:25

😮 Impressed



0:24

😮 Impressed

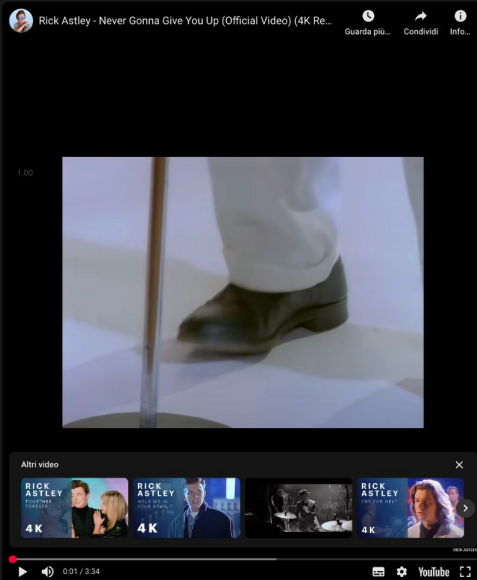
0:23

😮 Impressed

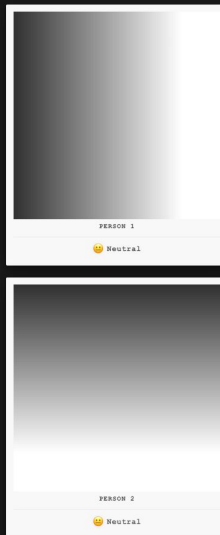


0:22

## Dashboard

[New Session](#)[Emotion Replay](#)[Cinema View](#)

## Camera Feeds

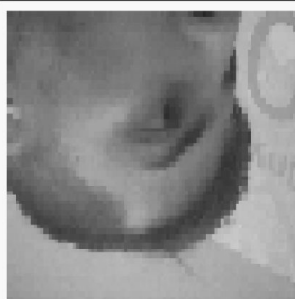


## Emotion Timeline

50 reactions - Video duration: 3:34 - Click any marker to jump

EMOTIONS: Neutral Happy Sad Impressed

There is also a dashboard for the film-maker to replay the faces of the viewers and understand their emotions. The video is synced with the Nicla frames and the predicted emotions.



PERSON 1

😊 Happy



PERSON 2

😮 Impressed

## Emotion Timeline

31 reactions • Video duration: 40:22 • Click any marker to jump



TINY\_HACK





# Dashboard

New Session

Emotion Replay

Cinema View



Finally, there's also  
a **cinema view** !



The seats light up  
when the viewer who's  
seating there is  
**"impressed"**

thank you very much