Me Not Me

Presentazione progetto di Sistemi Digitali

Concept

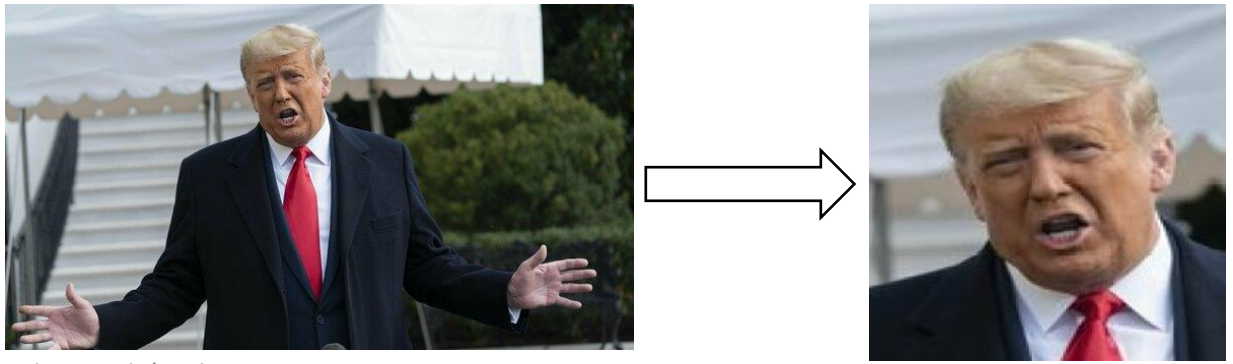
- Realizzazione di un app in **Android** in grado di riconoscere il volto dell'utente
- Rete neurale convoluzionale (CNN)
- Due tipi di filtraggio delle immagini
  - *Statico* → selezionando foto dalla galleria
  - *Real-time* → aprendo la fotocamera

Suddivisone in sotto-progetti

- **Rete**
  Addestramento modello
- **Detector statico con Python**
  Realizzazione in Android di un detector statico implementato in Python e Java
- **Detector live**
  Realizzazione di un riconoscitore di volti in real-time in Android
- **Detector statico senza Python**
  Revisione più efficiente del detector statico
- **Realizzazione dell'app**
  Unione dei vari prodotti per realizzare un'applicazione Android completa

Step 1: Rete

- **Obiettivo**: data una foto con più persone, riconoscere se l'utente è presente
- Creazione **dataset**
  - Due classi: *me* e *not me*
  - **Not me**: Recupero dataset open-source di soggetti eterogenei (Flickr-Faces-HQ Dataset)
  - **Me**: Ricezione foto da parte dell'utente
  - **Creazione algoritmo** ad hoc per estrapolare singoli volti da una foto e *resize* dell'immagine ottenuta



Step 1: Rete

- **Addestramento** rete
  - Utilizzo di Tensorflow/Keras
  - **MobileNet** come modello di partenza
    - Validation ratio: 15%
    - Batch size: 16
    - Epoch: 50
    - Model checkpoint
    - Early stopping: patience=3

```
# ModelCheckpoint to save model in case of interrupting the learning process
checkpoint = ModelCheckpoint("models/face_classifier.h5",
                             monitor="val_loss",
                             mode="min",
                             save_best_only=True,
                             verbose=1)

# EarlyStopping to find best model with a large number of epochs
earlystop = EarlyStopping(monitor='val_loss',
                          restore_best_weights=True,
                          patience=3,  # number of epochs with no improvement after which training will be stopped
                          verbose=1)

callbacks = [earlystop, checkpoint]
```

Step 1: Rete

- **Utilizzo** della rete
  - Input: tensore 4D (shape 1x250x250x3)
    - Scelta di usare framework già pronti per isolare volti all'interno della foto
      - **MTCNN** (Multi-task Cascaded Convolutional Networks)

      → Nel detector statico basato su Python
      - API Vision di Google **ML Kit**

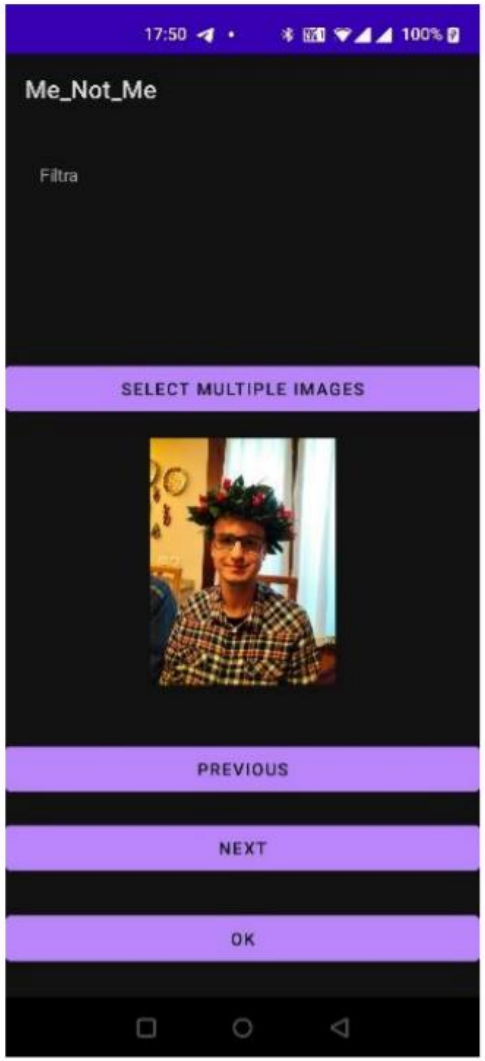
      → Nei detector basati solo su Java
  - Output: tensore 2D (shape 1x2)
- Conversione in TF Lite per uso su sistemi embedded

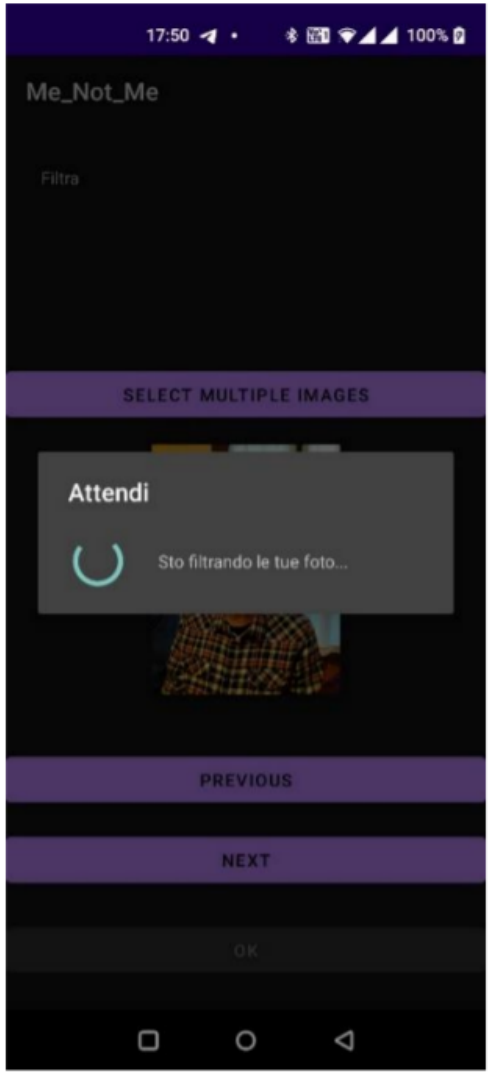
Step 2: Detector statico

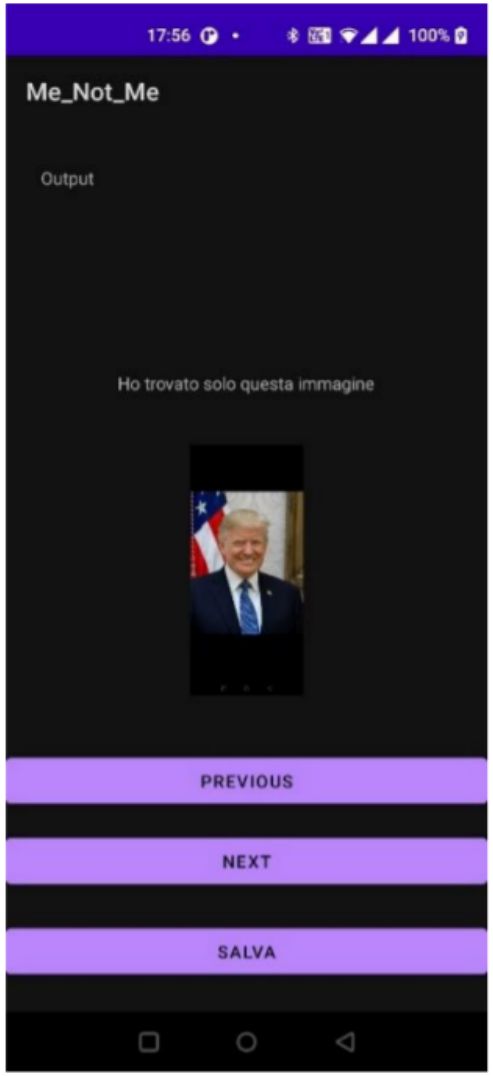
- Guscio esterno realizzato in Java
- Business logic in **Python**
  - *Individuazione volti* → MTCNN
  - *Riconoscimento volto* → Modello TFLite
- **Uso di *Chaquopy* ←**
  - SDK per integrare Python in app Android
  - API per chiamare codice Python da Java

```
//ottenimento di un PyObject che rappresenta la funzione python da chiamare
//all'interno dello script detector.py
PyObject obj = Python.getInstance().getModule( name: "detector").callAttr( key: "main");
```

Step 2: Detector statico





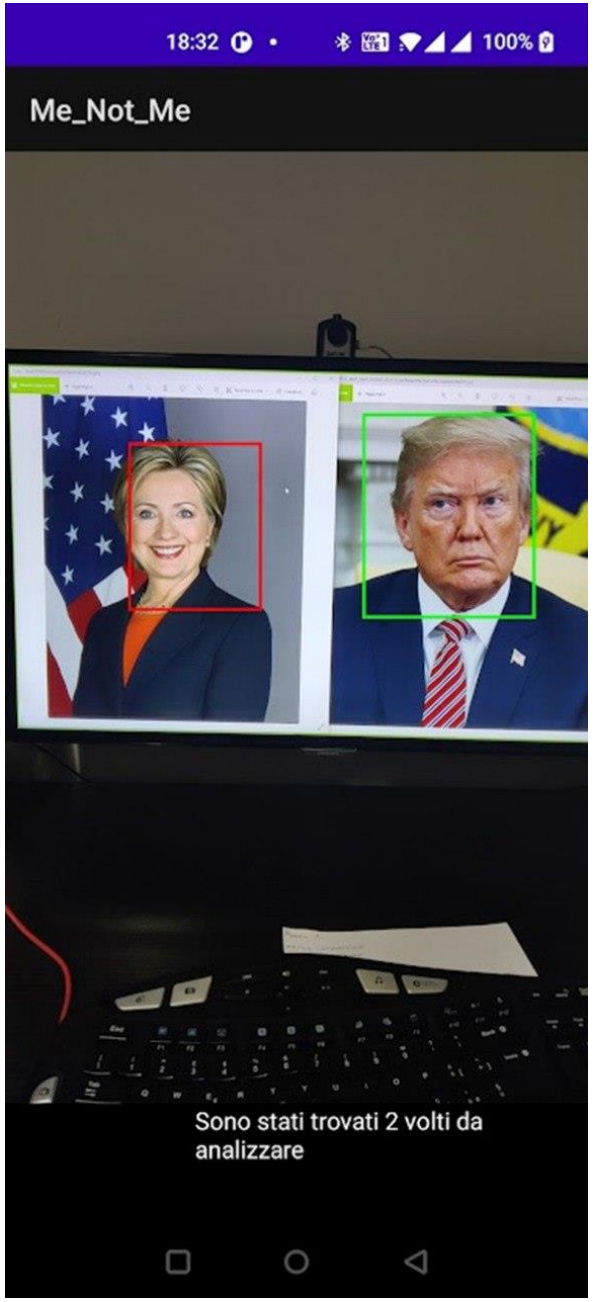


Chaquopy: considerazioni

- **Prodotto sotto licenza**
  - Impossibilità legale e materiale di distribuire l'app
  - Limitazioni sul tempo di utilizzo del codice Python
- **Impossibilità di accedere a ottimizzazioni hw**
  - Degradazione delle prestazioni
- Uso di risorse
  - Consumo di memoria eccessivo
  - Tempi di building onerosi
- Impossibile da usare per il detector live

Step 3: Detector live
(attività progettuale)

- No Python (e Chaquopy)
- Librerie di Tensorflow per Java
- API Vision di Google ML Kit
- API di CameraX
- Image Analysis
- Disegno quadrati su layer trasparente



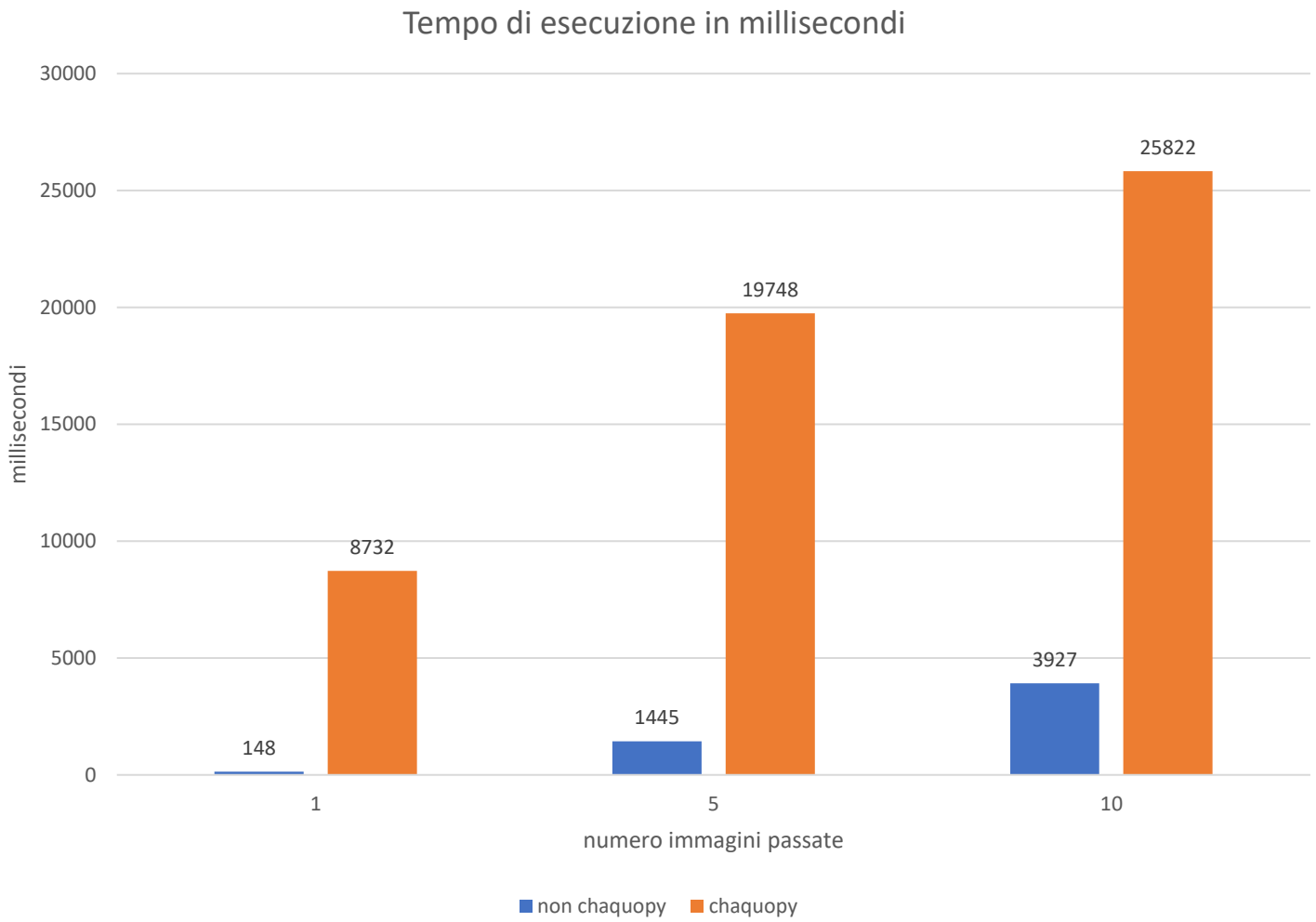
Step 4: revisione detector statico
(attività progettuale)

- No Python (e Chaquopy)
- Librerie di Tensorflow per Java
- API Vision di Google ML Kit

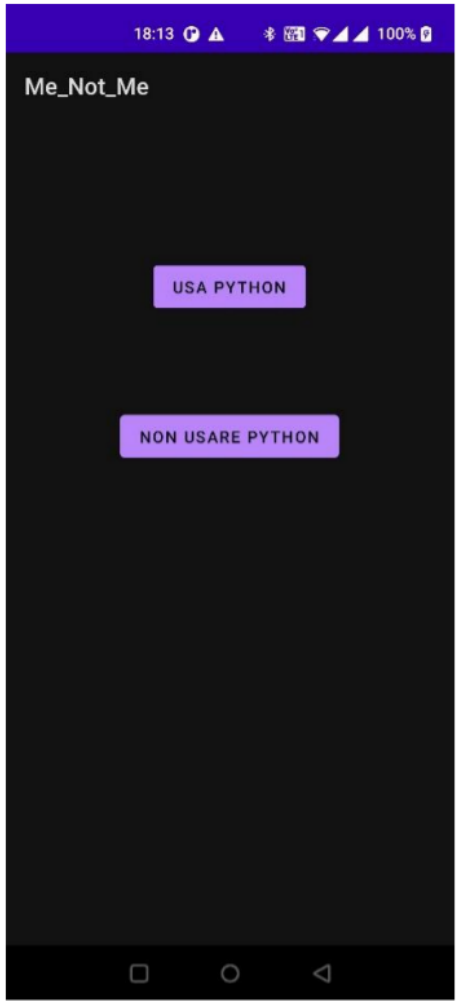
```
//caricamento del modello tflite
MappedByteBuffer tfliteModel = FileUtil.loadMappedFile(getApplicationContext(), filePath: "model.tflite");
//inizializzazione dell'interprete
Interpreter tflite = new Interpreter(tfliteModel);
```

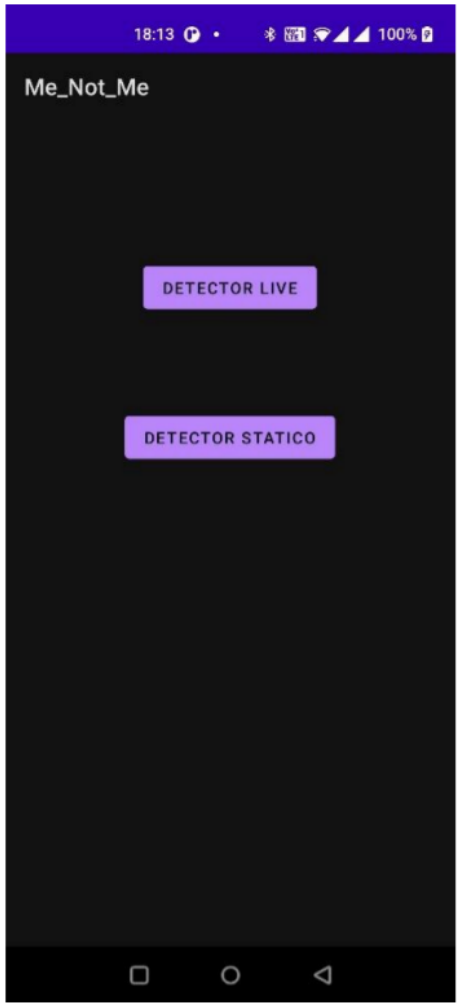
```
//per ogni foto selezionata si ricavano tutte le facce presenti in essa
resultFace = detector.process(InputImage.fromBitmap(bitmapFOTO, rotationDegrees: 0));
```

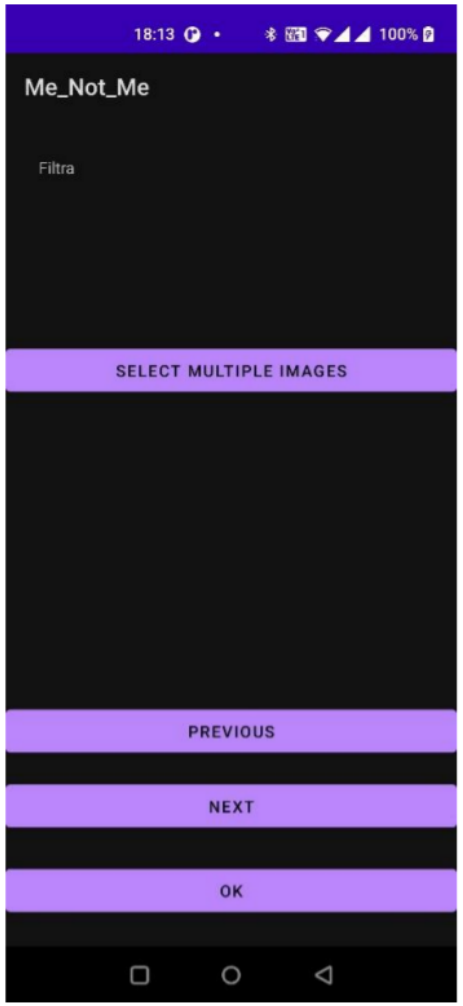
Confronto detector statici



Step 5: App


18:13
100%
Me_Not_Me
USA PYTHON
NON USARE PYTHON


18:13
100%
Me_Not_Me
DETECTOR LIVE
DETECTOR STATICO


18:13
100%
Me_Not_Me
Filtra
SELECT MULTIPLE IMAGES
PREVIOUS
NEXT
OK

Conclusioni

- Risultati soddisfacenti
- Applicativo conforme alle aspettative
- Chaquopy inadeguato
- Android adatto e prestante
- Sviluppi futuri:
  - Aggiungere possibilità addestramento
  - Rimuovere parte con Python