YOGA MASTER

By: Noa Wengrowicz & Barr Assenheimer

Supervisor: Ron Slossberg
PROBLEM STATEMENT

YOGA MATSER is a platform for yoga practicing anywhere you want. With the Jetson Nano you can take the yoga teacher to the park, to the beach, or just stay at home. Although the teacher is not near you can still get the feedback you need to improve your poses.
PROJECT SCOPE

BACKGROUND

- **NVIDIA® Jetson Nano™** Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing.

- **TensorFlow** is an open-source library for numerical computation and large-scale machine-learning.

- **TensorFlow Posenet** is a real-time pose estimation model.

- **TensorFlow Object Detection API** is an open-source framework built on top of TensorFlow for constructing, training and deploying object detection models.
PROJECT SCOPE

BACKGROUND
In the last few years there is an increasing interest in fitness apps and at home training. The main downside of these apps is the lack of real time feedback.

This need was emphasized in the last few months, during the COVID-19 outbreak, forcing people to find alternatives for live fitness classes.
PROJECT SCOPE

HYPOTHESIS

- The real-time yoga poses detection on top of the posenet skeleton tracker can give the trainers the feedback they need to practice from home and improve their yoga poses.
PROJECT OBJECTIVES

- **YOGA MASTER** will detect yoga poses in real-time.
- **YOGA MASTER** will display the pose detection frames with the skeleton on top of the live video stream.
- **YOGA MASTER** will let the user know whether he did the pose correctly.
- **YOGA MASTER** will keep track of the poses the user did correctly.
YOGA MASTER

USES DOUBLE INFERENCE

Input:

Inference the image through posenet to get the skeleton

Inference the skeleton image via SSDLite to get the pose detection frames

Output:

Display the pose detection frames on top of the skeleton
SOLUTION PROCEDURE

1. Setup the Jetson Nano
2. Collect data
3. Label the data
4. Add instructions and score board
5. Train the network
6. Become a YOGA MASTER
The Jetson Nano environment is challenging. Setting up the required packages for the project took longer than we anticipated.
SOLUTION PROCEDURE

Setup the Jetson Nano → Collect data → Label the data

Become a YOGA MASTER → Add instructions and score board → Train the network
SOLUTION PROCEDURE

- We collected 130 images for each of the 4 poses.
- We tried to create a variety of images:
  - Different people (different body shapes)
  - Different locations in the image
  - Different angles

We collected 130 images for each of the 4 poses.
We tried to create a variety of images:
- Different people (different body shapes)
- Different locations in the image
- Different angles
SOLUTION PROCEDURE

- We used ‘labellmg’ to label all the images
SOLUTION PROCEDURE

- We discovered the training process is too slow on the Jetson Nano, so we needed to find an alternative.

- We decided to do training using the Google Colab GPU. It was much faster, but not so easy to work with.
xml files (labeled images)
xml_to_csv.py

csv files (train and test)
generate_tfrecord.py

train.record
test.record
+ labelmap.pbtxt
ssdlite_mobilenet.config

inference_graph

model.ckpt

model_main.py
SOLUTION PROCEDURE

TRAINING PROCESS

```xml
<annotation>
  <folder>Images</folder>
  <filename>mountain01.jpg</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>640</width>
    <height>480</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>mountain1</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>240</xmin>
      <ymin>36</ymin>
      <xmax>410</xmax>
      <ymax>410</ymax>
    </bndbox>
  </object>
</annotation>
```

```python
model {
  inplace_batchnorm_update: true
  freeze_batchnorm: False
  num_classes: 4
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_modulation: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  encode_background_as_zeros: true
  anchor_generator {
    ssd_anchor_generator {
      num_layers: 6
      min_scale: 0.2
      max_scale: 0.95
      aspect_ratios: 1.0
    }
  }
}
```
SOLUTION PROCEDURE

- We used **TensorBoard** to visualize the training process (model_main.py).
- To get the best results we tried several options of:
  - Batch Sizes
  - Partitions of the train and test data.
  - Training duration.

![](Image)
SOLUTION PROCEDURE

Try to do one of these YOGA poses:

TREE
WARRIOR
TRIANGLE
MOUNTAIN

GO!

Setup the Jetson Nano ➔ Collect data ➔ Label the data ➔ Add instructions and score board ➔ Train the network ➔ Become a YOGA MASTER

SCORE BOARD

TREE  WARRIOR  TRIANGLE  MOUNTAIN
0      0        2        0
SOLUTION PROCEDURE

- Setup the Jetson Nano
- Collect data
- Label the data
- Add instructions and score board
- Train the network
- Become a YOGA MASTER
CONCLUSION

- The video streaming gives better result when running YOGA MASTER directly on the Jetson Nano and not via SSH.
- The Jetson Nano environment is challenging and setting it up takes long time.
- Training the model on the Jetson Nano is too slow. It is better to use a powerful GPU.
- Best training results were obtained when:
  - The batch size was 8.
  - We ended the training is when the ‘DetectionBoxes_Precision mAP’ is very close to 1.
  - We used 80% of the data for training, and the other 20% for testing.
FUTURE WORK

- Add more poses.
- Add different fitness activities.
- Augment exciting images.
THANK YOU

Them: The AI takeover is incoming.
The AI: