Mobile Computing Lab

Transfer Learning

Olga Saukh
Institute of Technical Informatics, TU Graz
Complexity Science Hub Vienna

saukh@tugraz.at

28.04.2021

Outline

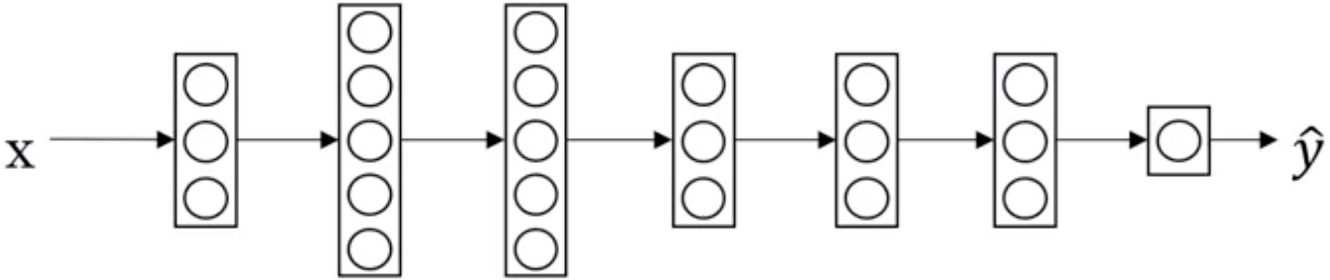
- **Option 1: Activity Monitoring Demos**
- **Workshop 4: On-device Deep Learning**
  - Training a deep activity monitoring model in the cloud
  - TF Lite: Model optimizations for mobile devices
  - Light re-training the model on a mobile device
- **Next week: Option 2: Progress Review**

Option 1: Activity Monitoring Demos

- **"Remote" demos are not easy, especially live!**
  - Video recording
  - Live presentation → mirror your smartphone's screen to your PC:
    - *WebEx on a smartphone and then share screen (?)*
    - *Screen sharing software, e.g., Vysor (free)*
- **When presenting your demo shortly explain**
    - *What is happening (we may not see this)? What do we see?*
    - *What <u>sensors</u> do you use? What <u>activities</u> do you monitor?*
    - *What are the <u>features</u> that you used?*
    - *What <u>method</u> did you use?*
    - *What is your achieved performance? (<u>confusion matrix</u>)*
    - *What was difficult? How much time did it take you for finish the app?*
    - *Are you happy with your results? How can your results be improved?*

Workshop 4: On-device Deep Learning

Transfer Learning



- **Transfer learning = transfer knowledge learned from one task to another task**
  - Knowledge = low-level features
  - Makes sense when you have a lot of data for the task you transfer from and little data for the talk to transfer to
- **Pre-training and fine-tuning**
  - If you have a lot of data → maybe you can re-train all parameters of the network
  - If you have a small data set → maybe re-train only the last 1-2 layers

When Does Transfer Learning Make Sense?

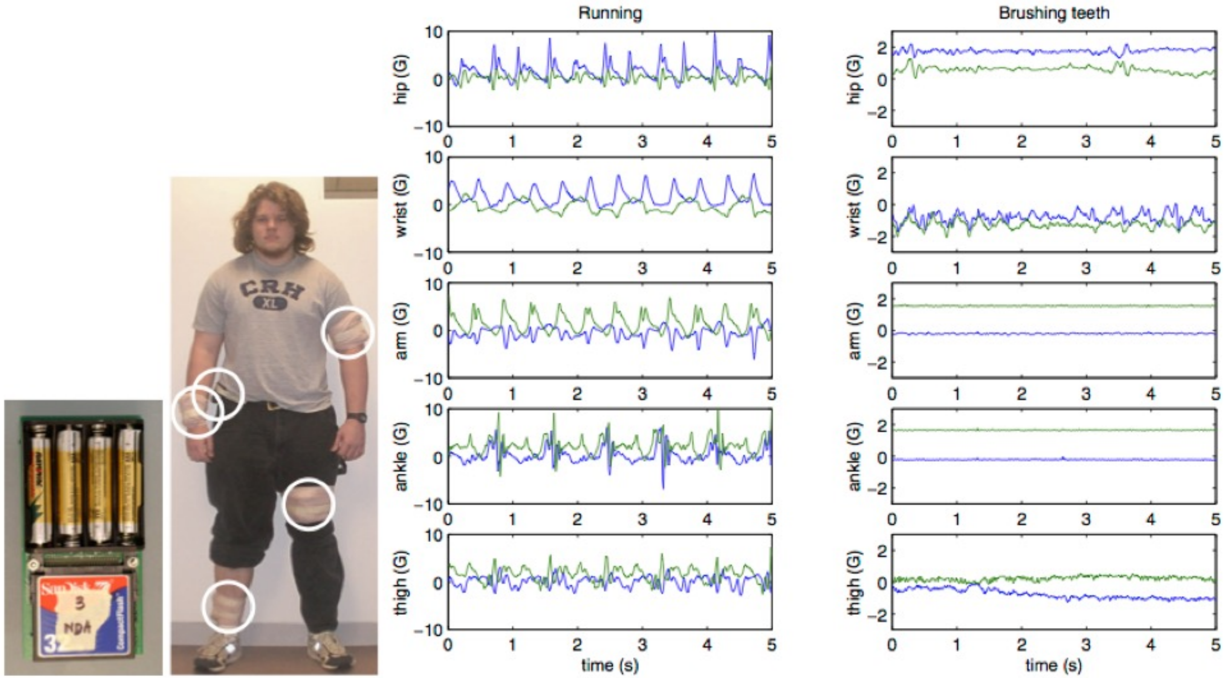
Transfer: Task A → Task B

- **Task A and B have the same input x**
  - E.g., images, audio clips. acceleration measurements
- **You have a lot more data for Task A than Task B**
  - E.g., it’s difficult / inconvenient to gather / label a lot of data for Task B
- **Low-level features from A could be helpful for learning B**
  - E.g., edges, lines, simplexes, state changes

- **Transfer learning = learning tasks sequentially. Multi-task learning = simultaneously**

On-device Activity Recognition

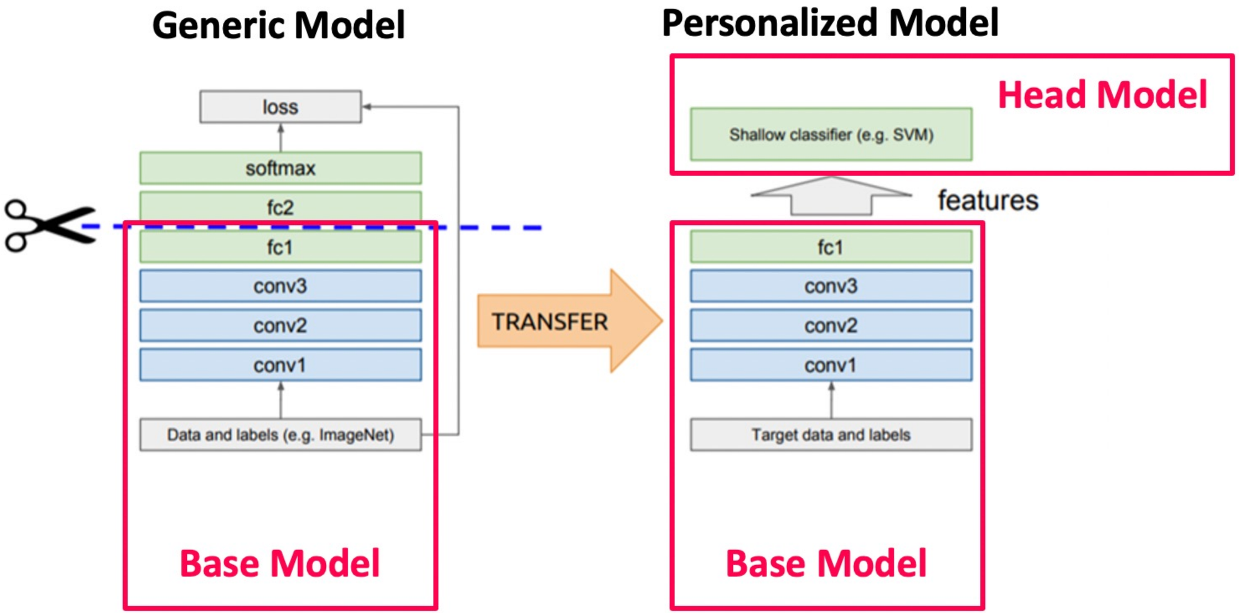
Goal: Learn a generic HAR model and personalize it



Source: Paper "Activity Recognition from User-Annotated Acceleration Data", Pervasive 2004

On-device Transfer Learning Pipeline

- **Generic Model**
  - Dataset
  - Data exploration
  - Model architecture
  - Model training
  - TF Lite Conversion
- **Head Model**
  - Model structure
- **Android Application**
  - Mobile app integration



Learning Generic Model

- **Dataset: HAPT**
  - Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set , 2015
    - *3 static postures (standing, sitting, lying), 3 dynamic activities (walking, walking downstairs and walking upstairs), 6 postural transitions between the static postures*
    - *30 volunteers wearing a smartphone (Samsung Galaxy S II) on the waist*
    - *3-axial accelerations (accelerometer) and 3-axial angular velocities (gyroscope) @ 50Hz*
    - *Manual labelling*

Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set

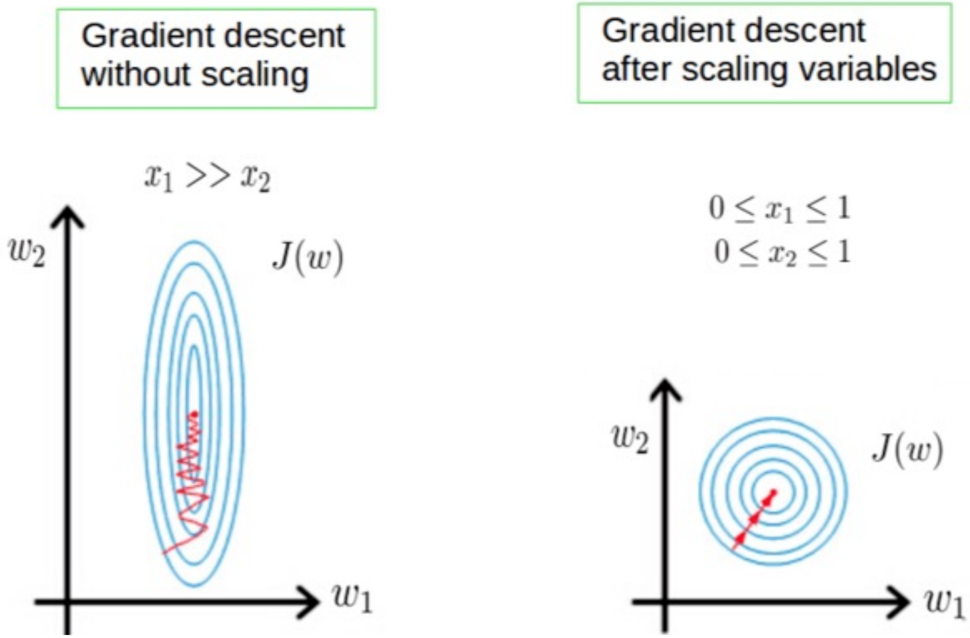
Download: Data Folder, Data Set Description

Abstract: Activity recognition data set built from the recordings of 30 subjects performing basic activities and postural transitions while carrying a waist-mounted smartphone with embedded inertial sensors.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	10929	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	561	**Date Donated**	2015-07-29
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	191721

Learning Generic Model

- **Data pre-processing**
  - **Normalization / Feature scaling**
    - *Why?*
      - Higher ranging numbers have "unwanted" superiority
      - SGD converges faster
  - On-hot-encoding
  - Dataset split: train / dev / test



Read: All about Feature Scaling

Learning Generic Model

- **Data pre-processing**
  - Normalization / Feature scaling
  - **On-hot-encoding**
  - Dataset split: train / dev / test

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

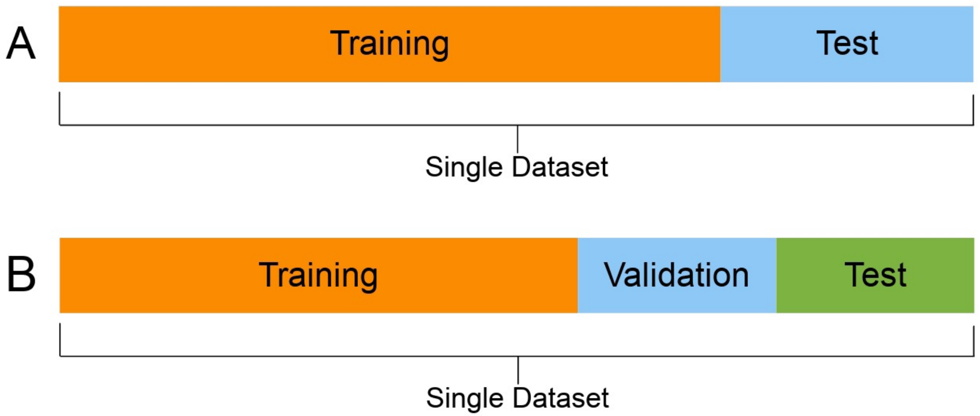
One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Read: What is One Hot Encoding and How to Do It

Learning Generic Model

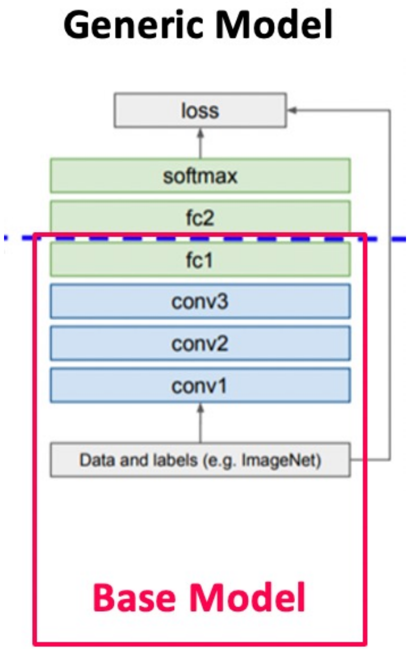
- **Data pre-processing**
  - Normalization / Feature scaling
  - On-hot-encoding
  - **Dataset split: train / validation / test**



Read: Training, validation, and test sets, read: Splitting into train, dev and test sets

Learning Generic Model

- Model architecture

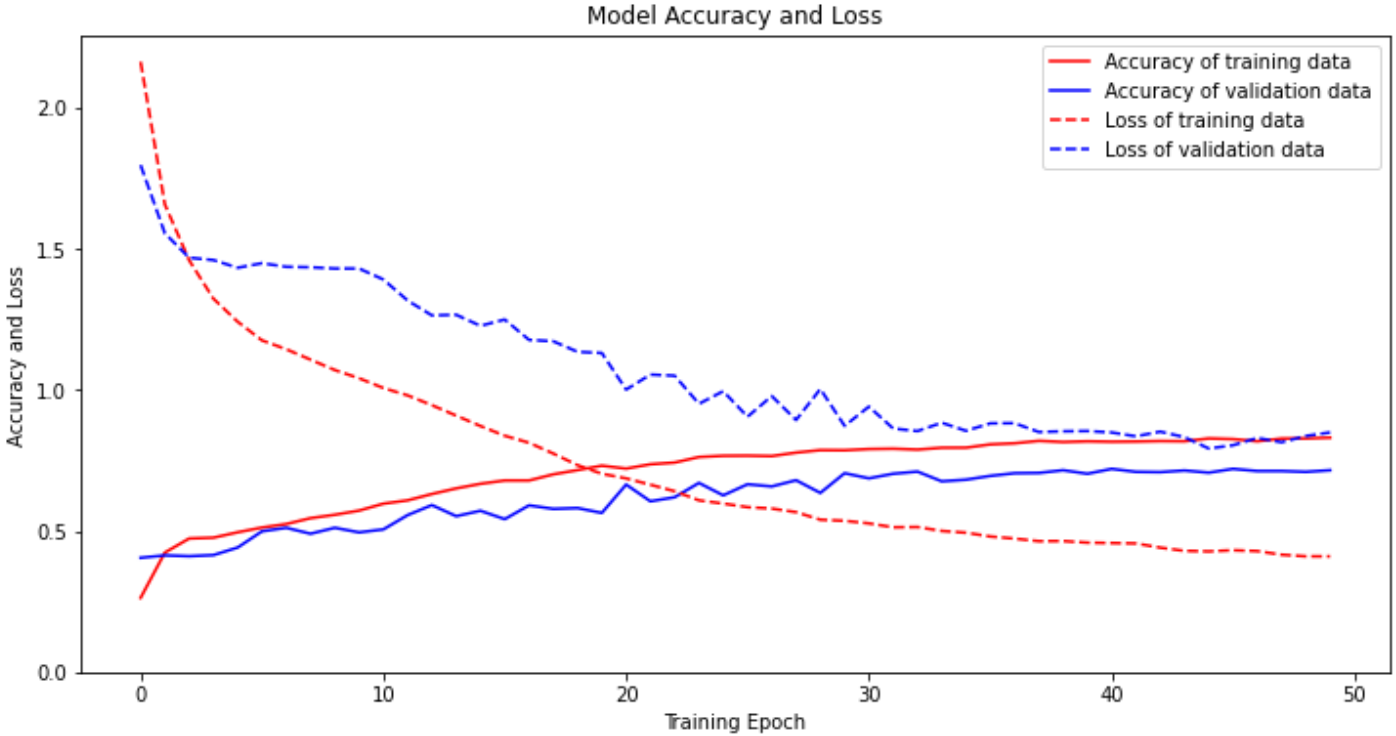


Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 200, 3)	0
dense (Dense)	(None, 200, 32)	128
dropout (Dropout)	(None, 200, 32)	0
dense_1 (Dense)	(None, 200, 16)	528
headlayer (Dense)	(None, 200, 8)	136
flatten (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 12)	19212

Total params: 20,004
Trainable params: 20,004
Non-trainable params: 0

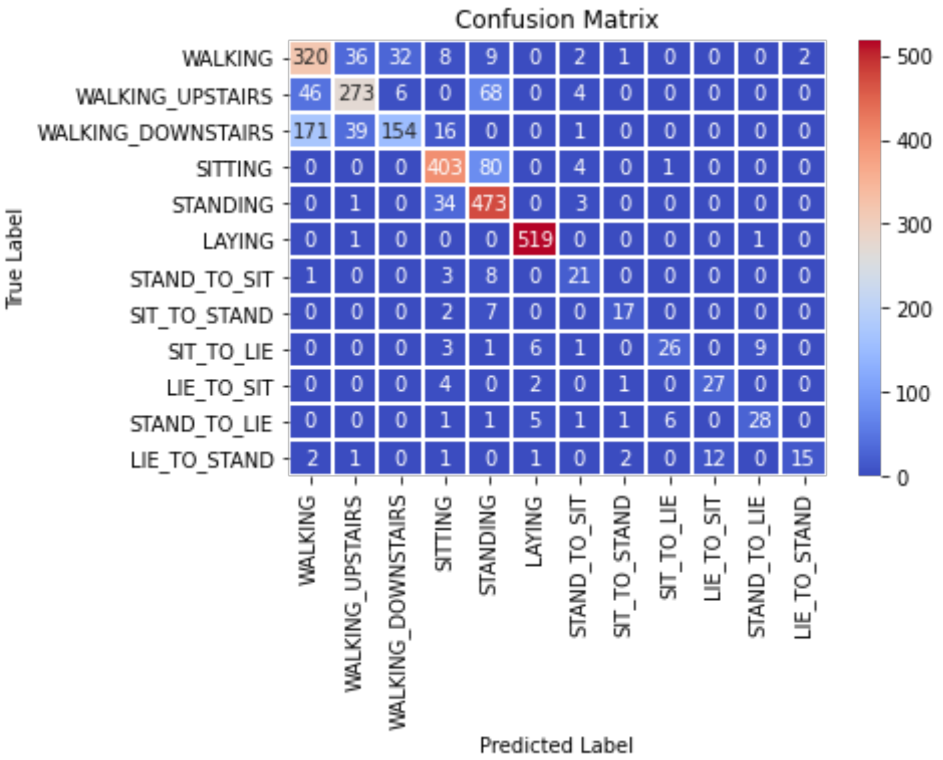
Learning Generic Model

- **Model training and performance (train acc: 81%, validation acc: 78%)**



Learning Generic Model

- **Model Performance (train acc: 81%, test acc: 78%)**

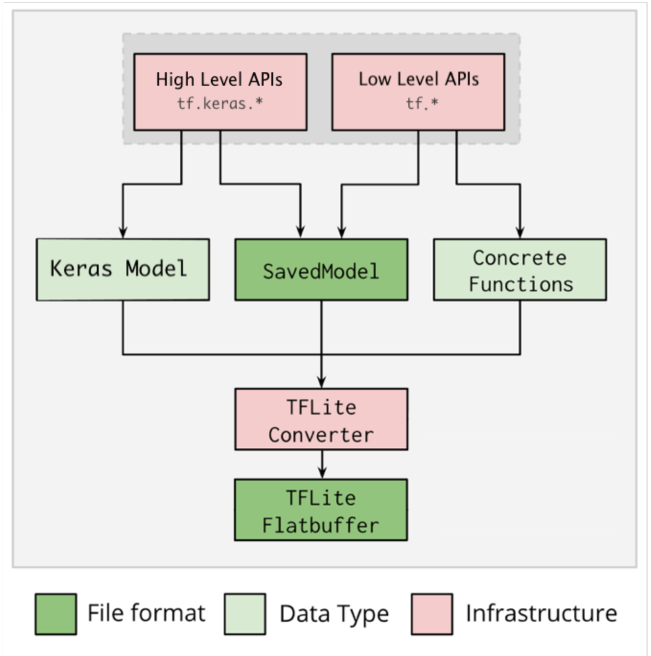


Your Task:

- **Improve generic model performance**
  - For static postures and activities

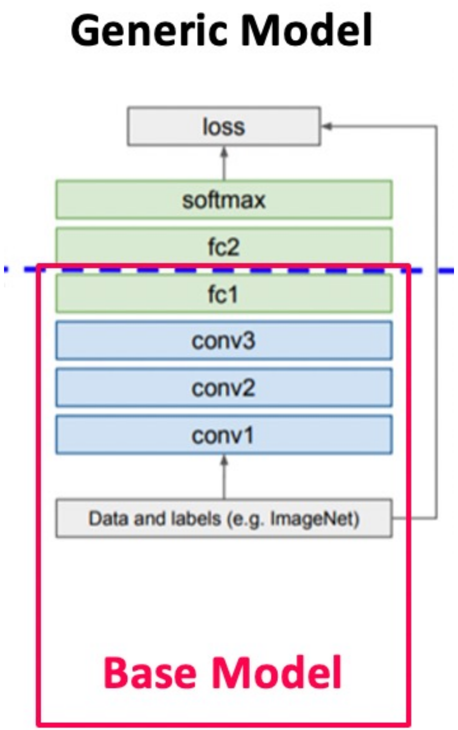
TF Lite Converter

- **Post-training optimizations**
  - Model conversion
    - *TF model → TF Lite model*
    - *Optimized [FlatBuffer] format,* `.tflite` *extension*
    - *CLI or Python API*
  - Optimizations
    - *Quantization = reduce accuracy of model parameters*
    - *Pruning = remove parameter which have minor impact on model accuracy*
    - *Accuracy loss depends on the model being optimized and is difficult to predict ahead of time*



Read: TensorFlow Lite Converter
Read: Model Optimization

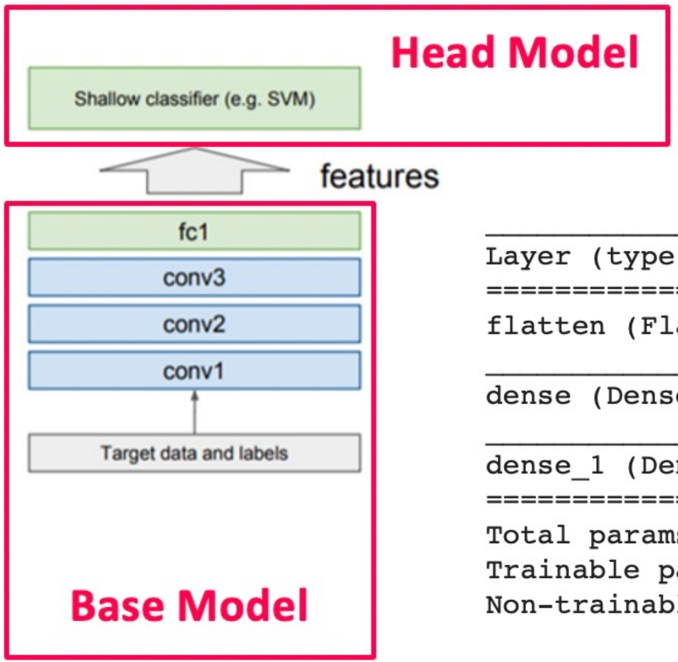
Base Model




```
Layer (type)                   Output Shape          Param #
=============================================================
reshape_input (InputLayer)    [(None, 600)]          0
_____________________________________________________________
reshape (Reshape)              (None, 200, 3)        0
_____________________________________________________________
dense (Dense)                  (None, 200, 32)       128
_____________________________________________________________
dropout (Dropout)              (None, 200, 32)       0
_____________________________________________________________
dense_1 (Dense)                (None, 200, 16)       528
_____________________________________________________________
headlayer (Dense)              (None, 200, 8)        136
=============================================================
Total params: 792
Trainable params: 792
Non-trainable params: 0
```

Head Model

Personalized Model




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 9606
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 14
=================================================================
Total params: 9,620
Trainable params: 9,620
Non-trainable params: 0
```

Transfer Learning Model

- **Transfer Learning Converter**

Read: [Example on-device model personalization with TensorFlow Lite]()

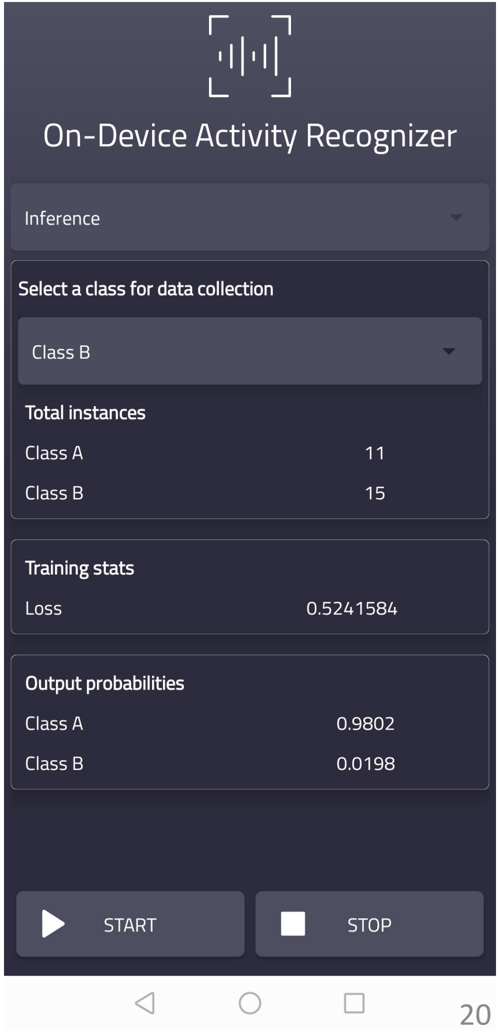
Android Application

- **2 Classes A and B**
- **Functions**
  - Gather data, training, inference
    - *Vibration if the probability of class B is above a threshold*
- **Code to store and load the trained model**

Your Task:

- **Extend to 4 static postures or activities**
- **Compare to your kNN implementation for two smartphone positions**

Read: On-device Learning of Activity Recognition Networks



Where We Are?

Option 1: Activity Monitoring

- **Activity Monitoring + Transfer Learning**
  1. Activity Monitoring: accelerometer + kNN (or other)
  2. **Transfer Learning (WS4)**
- **Evaluation criteria:**
  - Performance and creativity

Demos: 28.04.2021
Progress Review: 02.06.2021
Final demos: 23.06.2021

Option 2: Free Choice

- **Your own project** (requires approval)
  1. Pitch your idea
  2. **Implementation**
- **Evaluation criteria:**
  - Technical depth and creativity

Progress Review: 05.05.2021
Progress Review: 02.06.2021
Final demos: 23.06.2021

Questions?