April 05, 2019

TRAINING FOR MACHINE LEARNING MODELS TO PERSONALIZE GESTURES IN WEARABLE AND MOBILE DEVICES

Tyler Freeman

TRAINING FOR MACHINE LEARNING MODELS TO PERSONALIZE GESTURES IN WEARABLE AND MOBILE DEVICES

ABSTRACT

A machine learning system is described that enables a device (e.g., a wearable device or a mobile device) to personalize gestures for a user of the device based on movements of the user. Various motion sensors of the device may generate motion data that represents movements of the device. The device may utilize machine learning models to process the motion data to determine whether the user has performed a particular gesture (e.g., a tilt-to-wake gesture, hereinafter "TTW"). If the device determines that the user did preform the particular gesture, the device may perform an action corresponding to the particular gesture (e.g., the device may activate a display of the device in response to the user performing a tilt-to-wake gesture). The device may determine whether the determination of gesture performance was accurate (e.g., did the use interact with the device after the display was activated, which would indicate that the user intended to perform the tilt-to-wake gesture). If the device determines that the determination of gesture performance was accurate, the device may utilize the motion data to further train the machine learning models to process subsequent motion data. In this way, the device may personalize machine learning models to a user.

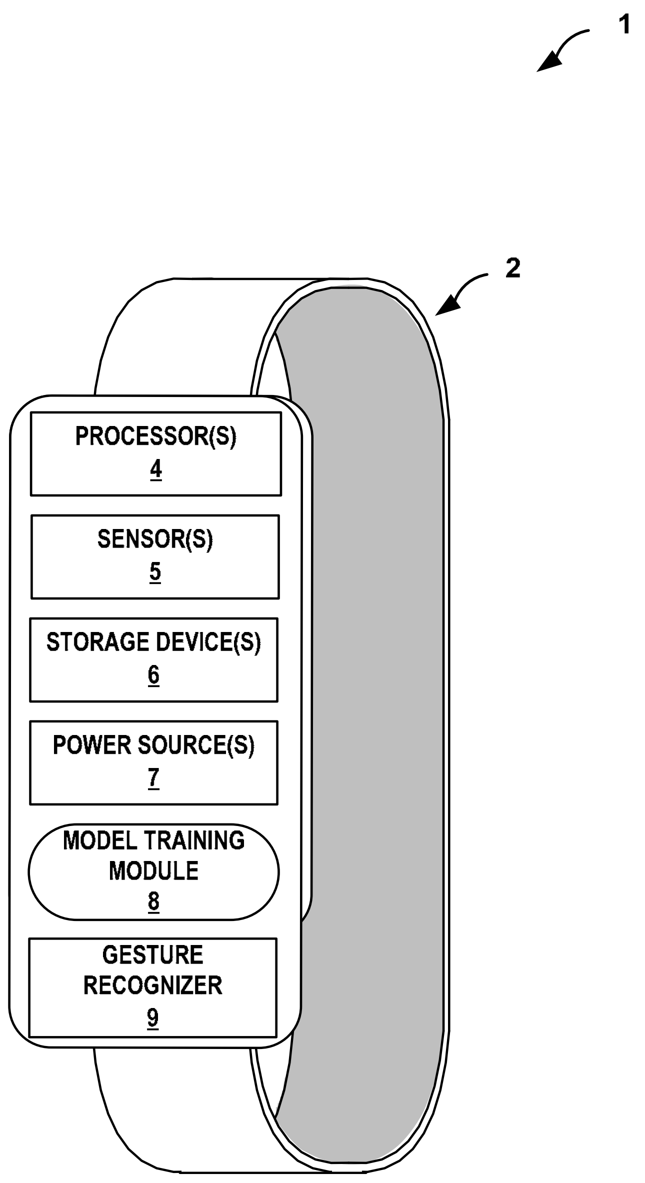
BACKGROUND

Currently, smartwatches and other wearable devices use a pre-defined algorithm in order to detect tilt gestures and other gestures. When the user lifts and tilts their wrist towards their face to look at the display (e.g., performs a TTW gesture), the accelerometer and other motion sensors in the device detect this movement and turn on the display so that the user can view the time, notifications, or other information being displayed. Typically, the display and processor (e.g., CPU) defaults to being turned off or in a low-power "ambient mode" when the user is not looking at the display. In order to save battery life in these extremely power-limited wearable devices, the display and processor may turn on to full power mode when the user performs the TTW gesture. Unfortunately, since TTW is typically a pre-defined, hardcoded algorithm, the gesture may be hard to perform. There is a tradeoff to making the algorithm more permissive (and thus increasing false-positives and thus decreasing battery life because the screen/CPU will wake up even when the user did not intend to look at it), and too strict (where even a deliberate gesture by the user may not actually be recognized as the "proper" TTW gesture, thus resulting in user frustration and poor user experience). Devices often provide little room for error or variation in the gesture which may be caused by body type, clothing, activity (like biking or running or sitting or standing), tightness of the watch band, or other situations that are unique to a single user.

DESCRIPTION

Techniques are described that enable a low-power and/or embedded device to perform personalized training of machine learning models. The device may include one or more motion sensors that collect motion data. The collected data may be stored in a memory or other storage device of the device. In operation, the device may utilize one or more machine learning models to process the motion data to determine whether the user has performed a gesture. Each time the device determine that the user has performed a gesture, the device may determine whether the determination was accurate. The machine learning models may be updated based on stored motion data corresponding to accurate gesture determinations. The device may utilize the updated machine learning models to process motion data to determine whether the user has performed a gesture at a later time. This process may repeat over time. As such, the machine learning models may become more and more personalized to the user.

Consider system 1 (referred to simply as “system 1”), shown below in FIG. 1 that includes an example wearable or mobile device (referred to simply as “device 2”) configured to facilitate personalization of machine learning models for gesture recognition, in accordance with the techniques described herein. As shown in the example of FIG. 1, system 1 includes device 2. In other examples, system 1 may include one or more devices/systems in addition to device 2. For instance, system 1 may include one or more cloud computing systems.



While illustrated as a wearable computing device, device 2 may be any type of wearable or mobile device. Examples of device 2 include, but are not limited to, smartwatches, visors (e.g., head-mounted computing devices), fitness trackers (e.g., wrist-worn, ankle-worn, waist-worn, or other devices worn by users that track various movement or other athletic activity), mobile computing devices (e.g., smartphones, tablets, and the like), or any other device that uses machine learning models. As shown in FIG. 1, device 2 may include processor(s) 4, sensor(s) 5, storage device(s) 6, power source(s) 7, and model training module 8.

Processors 4 may implement functionality and/or execute instructions within device 2. Examples of processors 4 include, but are not limited to, digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry.

Sensors 5 may generate data representative of various measurands. Examples of sensors 5 include, but are not limited to, microphones, cameras, accelerometers, gyroscopes, thermometers, heart beat sensors, pressure sensors, barometers, ambient light sensors, altimeters, and the like.

Storage devices 6 may store information for processing during operation of device 2. As one example, storage devices 6 may store data generated by sensors 5. As another example, storage devices 6 may store machine learning models used by device 2. As another example, storage devices 6 may store an operating system, applications, or other programs for execution at device 2. Examples of storage devices 6 include, but are not limited to, volatile memories (e.g., random access memories (RAM), dynamic random access memories (DRAM), static random access memories (SRAM), and other forms of volatile memories known in the art) and non-volatile memories (e.g., magnetic hard discs, optical discs, floppy discs, solid-state memories such as flash memories, or forms of electrically programmable memories (EPROM) or electrically erasable and programmable (EEPROM) memories, and other forms of non-volatile memories known in the art).

Power sources 7 may provide power to one or more components of device 2. For instance, power sources 7 may include a battery that provides power to processors 4, sensors 5, and storage devices 6. Power sources 7 may be re-chargeable via a charger. For instance, to recharge power sources 7, device 2 may be connected to a charger via a power link. The power link may be a wired link, such as a charging cable, or may be a wireless link, such as an inductive coupling. Power sources 7 may be sized such that device 2 may be operated all-day without requiring recharging. As such, device 2 may typically be connected to the charger at night, when device 2 is not likely to be used (e.g., as the user of device 2 is likely sleeping).

Device 2 may include gesture recognizer 9, which may be executable by processors 4 to identify the performance of one or more gestures. For purposes of discussion, gesture recognizer 9 will be described being able to identify the performance of a TTW gesture. However, gesture recognizer 9 may be capable of recognizing any gesture or combination of gestures. Gesture recognizer 9 may utilize a machine learning model (“ML model”) to perform the gesture recognition. The ML model may include one or more coefficients, one or more neural networks, or any other type of machine learning model.

As shown in the example of FIG. 1, device 2 may include model training module 8, which may be executable by processors 4 to train one or more machine learning models used by device 2. In other examples, model training module 8 may be located “off-device” in that such that computing resources external to device 2 can be used to perform the model training. Model training module 8 may train the models based on data generated by sensors (e.g., sensors 5), user actions, and context. For instance, model training module 8 may be executable by processors 4 to train a machine learning model used to determine whether or not to activate a display of device 2 using motion data generated by sensors 5 and user actions that indicate whether or not the user viewed or otherwise interacted with the display when it was activated.

When a user first uses device 2, gesture recognizer 9 may implement a default TTW algorithm. For instance, gesture recognizer 9 may implement a neural network machine learning model trained to respond similarly to the hardcoded TTW algorithm discussed above. That is, gesture recognizer 9 may initially utilize a generic algorithm that should perform reasonably in generic cases for average body types and user gestures. Gesture recognizer 9 may lean conservative/strict, in order to save power by filtering out suspected false-positives. However, each time the user tilts device 2 in order to interact with it, device 2 will save that motion data (e.g., accelerometer data from one or more accelerometers of sensors 5) in storage devices 6.

Model training module 8 may apply the gathered data (i.e., the stored motion data) to train ML model utilized by gesture recognizer 9. In particular, model training module 8 may refine the model so that the model will respond to the user’s unique method of lifting and tilting their wrist, automatically accounting for things like body type, wrist size, arm length, and how tight/loose they wear a band of device 2 (e.g., a watch band). In this way, model training module 8 may enable gesture recognizer 9 to speed up recognition of gestures, being able to wake up earlier/faster, and also react to gestures that would not have normally triggered a wakeup by the default algorithm. As discussed above, model training module 8 may be included in device 2, or may be included in another device (e.g., a remote server or the “cloud”). As such, the retraining of the ML model may be performed on-device or in the cloud.

In accordance with one or more techniques of this disclosure, model training module 8 may utilize tagged motion data to perform the model retraining. For instance, each time the user interacts with device 2 (by some type of user input, be it a tap on the touchscreen, a hardware button press, or voice activation) after gesture recognizer 9 determines that a TTW gesture has been performed, gesture recognizer 9 will mark/tag the accelerometer data used to determine that the TTW gesture has been performed (e.g., from the past 3-6 seconds) as a “true positive” TTW intention. However, the user does not interact with device 2 after gesture recognizer 9 determines that a TTW gesture has been performed, gesture recognizer 9 will mark/tag the accelerometer data used to determine that the TTW gesture has been performed (e.g., from the past 3-6 seconds) as a “false positive” TTW intention.

Model training module 8 may be configured to utilize motion data tagged as “true positive” to train the ML model (e.g., to train a neural network classifier so it may classify future inputs as a genuine TTW intention). As such, gesture recognizer 9 may utilize the retrained ML model to recognize similar gestures as TTW gestures, even if the original “default” TTW algorithm would have filtered them out. In some examples, model training module 8 may further train the ML model using motion data tagged as “false positive.” However, in other examples, model training module 8 may only train based on motion data tagged as “true positive”, and may refrain from training based on “false positives” or “true negatives” (i.e. where the accelerometer picks up significant motion but the user does not interact with the watch), as there may not be sufficient signal to determine the user’s intent (e.g. they might tilt the watch just to glance at the time, without actually interacting with the touchscreen).

In some examples, the window of time to gather training data after a user interaction may further be refined by various DSP algorithms, including low-pass filters, only using the most recent significant motion on a certain axis (y-axis), detecting the nearest local extrema on the specific axis, or only where the derivative of the curve of the motion is similar (i.e., does not change polarity, i.e., only use the last consistent motion in a single direction).

In some examples, model training module 8 may defer training of the model(s) based on the collected data until the device is charging. For instance, model training module 8 may monitor a status of device 2 (e.g., actively monitor, receive an interrupt, or the like), to determine when device 2 has been connected to a charger. In some examples, machine learning module 8 may immediately begin training model(s) in response to determining that device 2 has been connected to the charger. In other examples, machine learning module 8 may evaluate one or more other parameters to determine when to initiate training of the models. As one example, model training module 8 may wait a period of time (e.g., 1 minute, 5 minutes, 30 minutes, 1 hour, 2 hours, etc.) after determining that device 2 has been connected to the charger before initiating training of the models. As another example, after determining that device 2 has been connected to the charger, model training module 8 may wait until no user input is being received and/or no user input has been received for a period of time (e.g., 1 minute, 5 minutes, 30 minutes, 1 hour, 2 hours, etc.) before initiating training of the models. As another example, model training module 8 may use a machine learning model to predict when device 2 is likely to charged (e.g., be connected to the charger) for periods of time long enough to perform model training. In this way, if the user just charges device 2 in the middle of the day for a quick refresh (e.g., 10–15 minute), model training module 8 may avoid having to halt a model training operation when device 2 is disconnected from the charger.

By deferring model training for when device 2 is not being used, these techniques may enable device 2 to locally train machine learning models without degrading a user experience. For instance, as training machine learning models may consume more power than normal operation, training the models when device 2 is charging allows premature depletion of power sources 7 to be avoided. Additionally, as training machine learning models requires cycles of processors 4 and/or other system resources, deferring model training for when device 2 is not being used allows the cycles of processors 4 and the other system resources to be used for normal operation and/or user interactions. Similarly, as the model training is performed when the user is not interacting with device 2, device 2 may utilize higher amounts of processing power and/or other system resources than would be available for use if model training was not differed.

Furthermore, by generally enabling devices to locally train machine learning models, these techniques enable the power and benefits of machine learning to be brought to smaller, underpowered devices, and devices owned by users who do not have reliable or cheap connections to the internet for static model training on a remote server.