Distributed LSTM training - Predicting Human Activities on Edge Devices

KONSTANTIN SOZINOV KIM HAMMAR
sozinov | kimham @kth.se

KTH Royal Institute of Technology

January 9, 2018

The prediction task

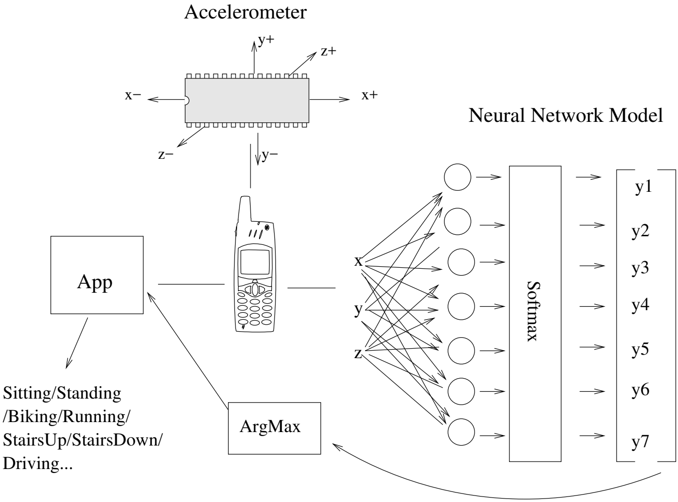


Figure: The mobile app will read data from the phone's accelerometer and feed that into a neural network model for predicting human activities

Background

- Open source Heterogeneity Activity Recognition Data Set[1]
- Data collected by Stisen et al.[2]
- Magnitude $\sim$ 10GB with sliding windows
- LSTM model inspiration from Venelin Valkov[3]
- Distributed training with TensorFlowOnSpark [4] running on `hops.site`

---

[1] https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition

[2] Allan Stisen et al. "Smart Devices Are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. SenSys '15. Seoul, South Korea: ACM, 2015, pp. 127–140. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809718. URL: http://doi.acm.org/10.1145/2809695.2809718.

[3] Venelin Valkov. *Human Activity Recognition using LSTMs on AndroidTensorFlow for Hackers*. https://medium.com/@curiousily/human-activity-recognition-using-lstms-on-android-tensorflow-for-hackers-part-vi-492da5adef64. 2017.

[4] https://github.com/yahoo/TensorFlowOnSpark

Data Exploration

Output classes

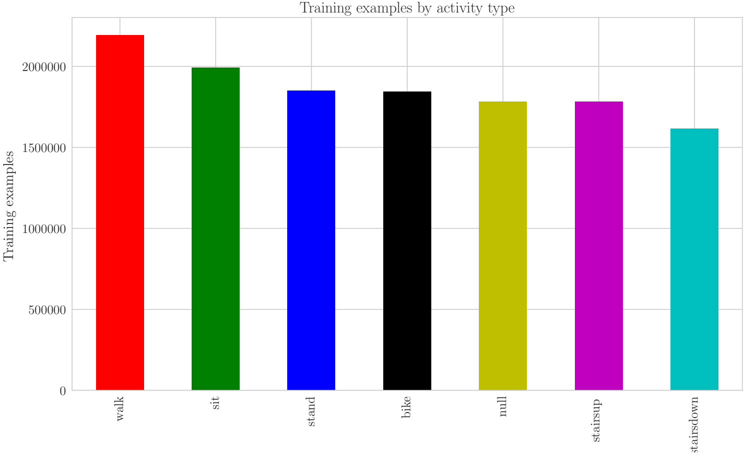
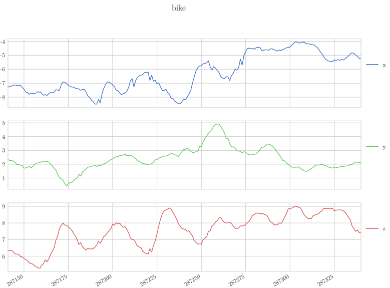


Figure: Distribution of classes among training examples

Data Exploration

Sensor data



(a) Sensor input for the activity "bike"

(b) Sensor input for the activity "sit"

Figure: Sensor inputs for the activities "bike" and "sit". x = blue, y = green, z = red

The Model

Neural Network Model

- Two FCC Layers
- Two LSTM layers
- Size[5]= $N_{bits}(3 \cdot 64 + 2(4(64^2 + 64^2)) + 64 \cdot 7 = 32b \cdot 66176 = 2117632b \approx 260KB$

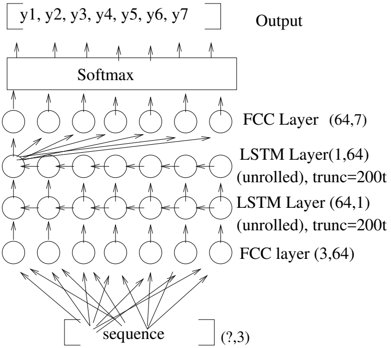


Figure: Neural Network Model

[5]Bias terms are excluded from this calculation. Each LSTM cell has 4 input gates, each one having one input weight matrix W of size 64^2 and one input recurrent weight matrix U of size 64^2.

Distributed Training & On-edge Inference

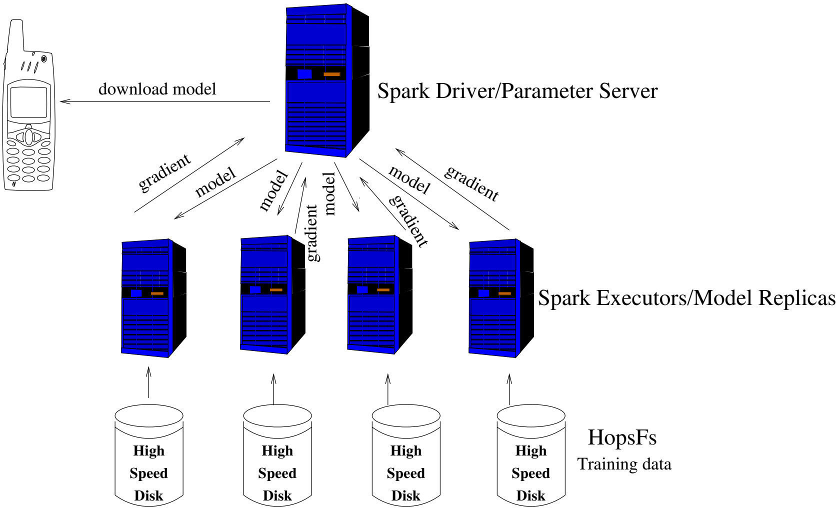


Figure: Distributed synchronous/asynchronous SGD on hops. Final model is frozen and downloaded to mobile client.

Results

Accuracy for single node training

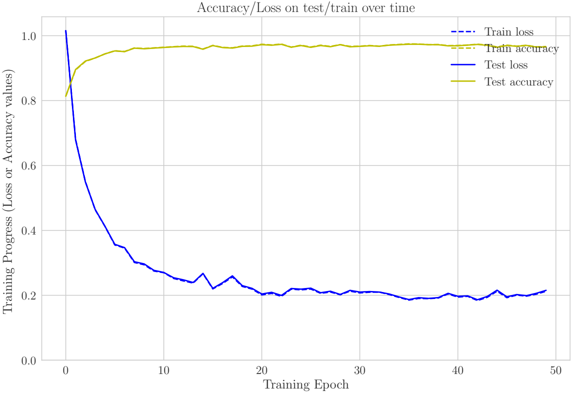
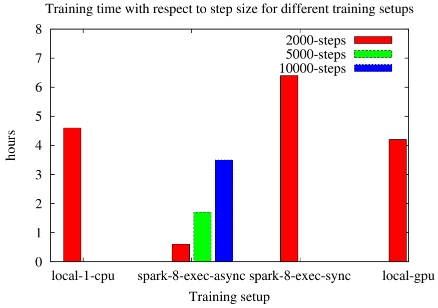


Figure: Accuracy and Loss over time during training for 50 epochs. **Best result: 97% accuracy (24h training on single machine)** Each epoch accounts for $\approx$ 510 batches. Batch size = 1000 sequences of length 200

Results

Training Time



(a) Training time benchmarks

(b) Accuracy benchmarks

Figure: Asynchronous SGD yielded **near linear training speedup but slower convergence**. Synchronous SGD was the slowest and suffered from exploding gradients (NaN loss). This indicates that the learning rate need to be fine-tuned for distributed training. TF-GPUs don't support full LSTM operations, did only give small boost over CPU.

App Facts and Demo

- Model size - $\approx$ 1.5 MB. Huge models, for example, Google's Inception V3 up to $\approx$ 100 MB.
- TensorFlow libraries for Android (if you want to target all CPU architectures) $\approx$ 50 MB. Can be reduced in Android 8 using Neural Network API.
- Core ML in iOS 11, support for frozen TensorFlow models.
- Emulator and live demo!

Questions?

- Try the code: project **har_2** on `hops.site`.
- Public Dataset folder : **HAR_Dataset/** The folder contains:
  - **README.md** with details about the data, preprocessing, the notebooks, and evaluation.
  - **The dataset** in its original form and its cleaned form
  - **Notebooks** for preprocessing the data, training with gpu, training with cpu, training with spark-sync-SGD, and training with spark-async-SGD
  - **Android App** for inference with a frozen version of the model on android phones.