Mobile Sensing: Deep Learning on Mobiles

Master studies, Winter 2020/2021

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si
Deep Learning in Mobile Sensing

• Mobile computer vision
  – Object recognition with CNN
• Activity recognition
  – CNNs and RNNs
Deep Learning in Mobile Sensing

• Depression detection
  – Autoencoder for relevant mobility feature identification
  – “Using autoencoders to automatically extract mobility features for predicting depressive states” by Mehrotra et al.

• Cognitive load inference
  – LSTM processing wireless radar signals reflected off a person’s chest (i.e. breathing, heartbeats)

T. Matkovic and V. Pejovic, Wi-Mind: Wireless Mental Effort Inference Ubittention workshop with UbiComp'18, Singapore, October 2018.
Deep Learning in Mobile Sensing

• Predicting wireless signal quality in vehicular communication
  – LSTM on wireless spectrum sensing data

J. Joo, M.C. Park, D.S. Han, and V. Pejovic
Deep Learning in Mobile Sensing

- Healthcare
  - Predict an onset of a disease

Deep Learning in Mobile Sensing

- Key benefits of deep learning on mobile devices
  - **Reduced delay** – inferences can happen faster, if we don’t need to send the data back and forth to the server
  - **Reduction in bandwidth usage** – spectrum is a limited resource
  - **Operation when the connectivity is not available**
  - **Privacy** – keeping the data (your photos, voice recordings) on the device
Challenges with Deep Learning on Mobiles

• A neural network requires:
  – A lot of storage space/mem – millions of parameters
  – A lot of training data – NNs tend to work well only when there is a lot of (labelled) data available
  – A lot of computation – matrix multiplication, backpropagation, batch training, etc.

• Solving the problems:
  – Training data – background sensing, semi-supervised learning, crowdsourced efforts (Google)
  – Computation and storage space/mem – we cannot rely on the next generation of devices to solve everything
Reducing the Cost of Deep Learning

• Quantization:
  – Instead of 32b floats, use 16b floats, 8b/4b/2b integers, or binary (1/0) weights and activations
  – Example (from float32 to int8):
    • suppose weights and activations are in the range \([-a, a)\)
    • scale the output to \([-128, 128)\): \(x \mapsto \left[ 128 \frac{x}{a} \right] \).
    • e.g.
      \[
      \begin{pmatrix}
        -0.18120981 & -0.29043840 \\
        0.49722983 & 0.22141714
      \end{pmatrix}
      \begin{pmatrix}
        0.77412377 \\
        0.49299395
      \end{pmatrix}
      =
      \begin{pmatrix}
        -0.28346319 \\
        0.49407474
      \end{pmatrix}
      \]
      \[
      \begin{pmatrix}
        -24 & -38 \\
        63 & 28
      \end{pmatrix}
      \begin{pmatrix}
        99 \\
        63
      \end{pmatrix}
      =
      \begin{pmatrix}
        4770 \\
        8001
      \end{pmatrix}
      .
      \]
    • quantize:
      \[
      \begin{pmatrix}
        -24 & -38 \\
        63 & 28
      \end{pmatrix}
      \begin{pmatrix}
        99 \\
        63
      \end{pmatrix}
      \]
    • dequantize: \(x \mapsto \frac{ax}{16384}\) we get: \(\begin{pmatrix}
      -0.2911377 \\
      0.48834229
    \end{pmatrix}\).
Reducing the Cost of Deep Learning

• Weight sharing/virtualisation:
  – Represent multiple similar weights with a single value (often in combination with quantization)

Reducing the Cost of Deep Learning

• Pruning:
  – Reduce the number of weights after the training – inspired by human biology
  – Unstructured pruning:
    • Prune weights or whole neurons
  – Structured pruning:
    • Prune CNN filters or channels
  – How to select what to prune?
    • Weights lower than a certain threshold
    • Use constrained optimization algorithms
Reducing the Cost of Deep Learning

• Matrix decomposition:
  – Use Singular Value Decomposition (SVD) to replace an \( m \times n \) matrix with two smaller matrices of sizes \( m \times c \) and \( c \times n \)
    • Total calculation drops from \( O(m \times n) \) to \( O(c \times (m+n)) \)
    • \( c \ll m, n \)
Reducing the Cost of Deep Learning

• Other, more advanced approaches
  – Knowledge distillation
    • A larger Teacher network “transfers” knowledge to a smaller Student network

  – Slimmable neural networks
    • The same model can run at different widths allowing adaptive accuracy-efficiency trade-off
Mobile-Ready Deep Learning Networks

• SqueezeNet
  – Same accuracy as AlexNet with 50x fewer parameters
  – Replaces 3x3 filters with 1x1 filters
  – Squeeze layer

• MobileNet
  – Depthwise separable convolutions
  – Probably the best first choice for your deep learning applications
Programming Support for Deep Learning on Mobiles

• Core ML for iOS
• Caffe2 for iOS and Android
• TensorFlow Lite for Android and iOS
• PyTorch Mobile for Android and iOS
• Other players:
  – Fritz AI
  – Snapdragon SDK
  – ...

ML Kit and Firebase ML

- Firebase – a framework supporting mobile app development
  - Messaging, authentication, database, monitoring, etc.
- Firebase ML – (mostly) cloud-based ML
  - Cloud-based inference, model training, sending models to phones
  - Prebuilt models available
- ML Kit – on-device ML
  - Prebuilt models for text recognition, face detection, object detection and tracking, barcode scanning, etc.
TensorFlow Lite

- **TensorFlow** – a framework for NN programming
  - Build and train your NN (on a powerful computer)
  - Validate/test your NN
  - Keras – higher-level API for building and training NNs
- **TensorFlow Lite** – a mobile NN support library
  - Interpret a TF NN model on a mobile device
  - Firebase ML and ML Kit models use TensorFlow Lite under the hood
TensorFlow Lite

- Supported platforms
  - Android, iOS, Raspberry Pi, microcontrollers

- Means of operation
  - Pre-train a model in TensorFlow (Keras)
  - Convert the model to TensorFlow Lite, save to a file, and ship with your app
  - At runtime, an Interpreter runs a model on device

You can also dynamically change the model remotely via the Firebase ML console!
TensorFlow Lite – Achieving Speedup

• Running platform optimization
  – The model can be ran on CPU or GPU

• Faster loading
  – Memory mapped files in Android

• Quantization
  – To 16b floats or integers
  – To 8b dynamic range
  – Weights only, or weights and activations can be quantized

Don’t compress the model file!
TensorFlow Lite – Bootstrapping

• A number of models are already available:
  – https://www.tensorflow.org/lite/models
  – https://github.com/tensorflow/models

• Transfer learning
  – The higher the layer is in the hierarchy, the more specific its inference is
  – Take the first $N-k$ layers of the pre-built model and re-train the last $k$ layers with your data

This is what you do in this week’s lab!
TensorFlow Lite – Bootstrapping

• A shortcut:
  – AutoML model re-trained(?) with your dataset
  – Deployed to Android or pulled from the server on demand
  – To start go to: https://console.firebase.google.com
  – Prepare your dataset with labels
  – Train the model and load the file in your app or provide a link through the AutoML API
TODO

- Read “DeepX: A software accelerator for low-power deep learning inference on mobile devices” by Lane et al. for Thursday!
- Complete the deep learning on mobiles lab by Friday night
- Keep working on your projects!