TinyFace: Extreme Edge Face Detection on Embedded Devices

Teodor Fratiloiu

September 23, 2020

Why Do Machine Learning on Microcontrollers

- MCU's are already widely deployed - what if we used them to run *non-deterministic, non-algorithmic* apps
- Compared to ML devices with server-client architectures:
  - Data Processing & Filtering earlier on (right at *metal* level)
  - Better efficiency and privacy
  - Less Network Bottlenecks
- Opens up completely new use scenarios (e.g. smarter consumer IoT, more versatile industrial controllers etc.)

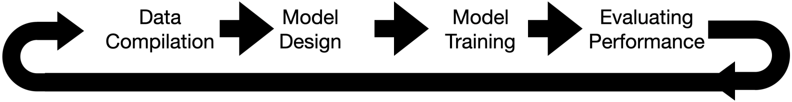
Goals for TinyFace - This Project

1. Face Detection with Machine Learning
2. High Accuracy (Usable for Real-World Applications)
3. Deployable to MCU's [1]
   - Very small model size (less than 1 MB)
   - Low-Power means more efficient apps are *necessary*
   - Less abstraction, more hardware programming
4. Using TensorFlow Lite, our own datasets and custom ML-models with our own training routines

Overview

1. The ML-Powered Face Detection Workflow
2. Compiling and Processing Datasets
3. Implementing and Training ML Models
4. TinyFace Evaluation Demo
5. Future Work

1. The ML-Powered Face Detection Workflow



This means we have a 4-Step (Loop) Process:

1. Generate Custom Datasets
2. Read papers on different model architectures and implement them ourselves
3. Train said models with own data, compare results
4. Develop an app to evaluate real-world accuracy

All steps have been successfully undertaken as part of the Thesis.

2. Compiling and Processing Datasets

- Wrote up an automated tool to generate our own data
  - Can search and categorize images of actors online
  - Can filter and process these images
  - Outputs organized and labelled datasets, with images set to desired sizes and color-spaces (color, b/w)
- Downloaded several datasets, from amongst those popular in research papers
  - Labelled Faces in the Wild [2]
  - Aligned Images [3]
  - YT Faces [4]

Dataset Sizes, Dimensions and Samples

Dimension	n : 1	x : 2	y : 3	c : 4
Content / Dataset	Number of Samples	X Dimension	Y Dimension	Color Channels
LFW	13,633	250	250	3
Own Dataset 1 (GS)	200,000	32	32	1

Dimension	n : 1	x : 2	y : 3	c : 4
Content / Dataset	Number of Samples	X Dimension	Y Dimension	Color Channels
Aligned Images	621,126	340	340	3
YT Faces	5,462,875 videos	1920	1080	3

Dimension	n : 1	x : 2	y : 3	c : 4
Content / Dataset	Number of Samples	X Dimension	Y Dimension	Color Channels
Own Dataset 1 (GS)	200,000	32	32	1
Own Dataset 2 (GS)	50,000	64	64	1
Own Dataset 3 (GS)	30,000	100	100	1
Own Dataset 4 (Color)	15,000	48	48	3

Figure: Training Images from Highest Performing Dataset

What an Image Looks Like to The Model

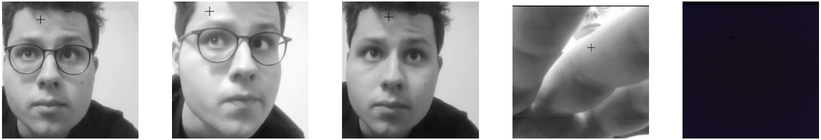


Figure: Original images, as seen on webcam - cropped by hand to similar sizes for the sake of comparison with the images in the following figure and turned to black and white from color for consistency.

Figure: What the interpreter actually *"sees"*. The images have sides of 100*100 pixels, their dynamic range has been downsampled to a value between 0 and 9 grayscale.

3. Implementing and Training ML Models

Layer
2D Convolution
2D MaxPooling
2D Convolution
2D MaxPooling
"Bottleneck Block"
2D MaxPooling
"Bottleneck Block"
2D MaxPooling
"Bottleneck Block"
2D MaxPooling
"Bottleneck Block"
2D Convolution

Layer
2D Convolution
2D Convolution
2D Convolution
ReLU
2D MaxPooling
2D Convolution
ReLU
2D MaxPooling
Dense Layer
ReLU
Dense Layer
Softmax

Layer
2D Convolution
2D Convolution
2D Max-Pooling
Dropout
2D Convolution
2D Convolution
2D Max-Pooling
Dropout
2D Convolution
2D Convolution
2D Max-Pooling
Dropout
Dense Layer
Dropout
Dense Layer

Figure: Architectures of Some Implemented Models, from Literature: DarkNet [5], LeNet [6], VGG [7]

SqueezeNet Model Architecture

2D-Convolution - Window Size: 1 * 1			
Activation - ReLU			
2D-Convolution	Window Size: 1 * 1	2D-Convolution	Window Size: 3 * 3
Activation	ReLU	Activation	ReLU
Concatenation			

A Fire Layer

Layer
2D-Convolution
Activation
2D Max-Pooling
Fire Layer
Fire Layer
Fire Layer
Fire Layer
Dropout
2D-Convolution - Window Size: 1 * 1
Activation
2D Average-Pooling
Activation

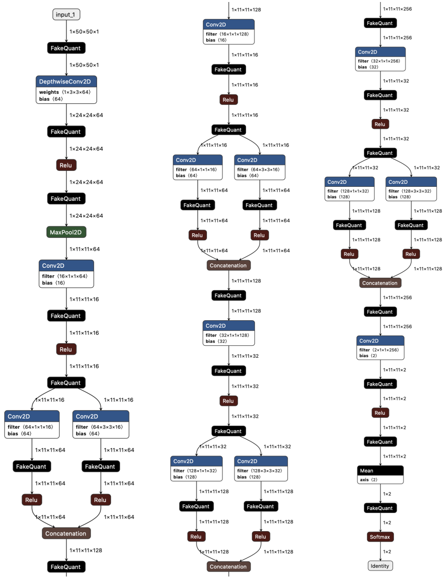
Implemented Models and their respective sizes

Model Architecture	TF-Lite model size	Number of Layers
SqueezeNet	4.8 MB	31
DarkNet	4 MB	26
LeNet	4 MB	11
Optimized SqueezeNet	**139 KB**	31

Figure: Final Sizes of some trained models

SqueezeNet was chosen as the final model architecture.

Trained and Quantized SqueezeNet



Training Performance of Squeezenet with Own Dataset

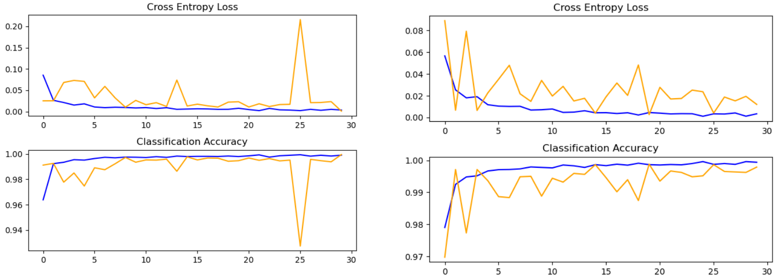


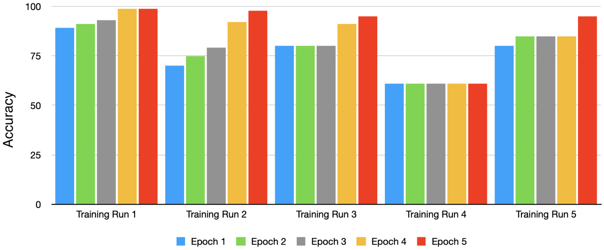
Figure: L: Full Quantization (*uint8*) - R: Full Size Weights (*32 bit*)

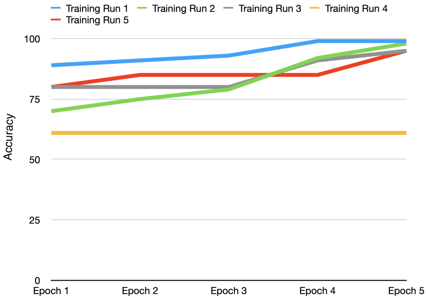
4. TinyFace Evaluation Demo

Properties of Used Model for Inference:

- 5 Training Epochs
- A Batch Size of 32 Samples
- A 75%-25% Divide between Training and Test Data
- A Dropout Rate of 10%

Training Performance of Said Model





Inference Performance of Said Model

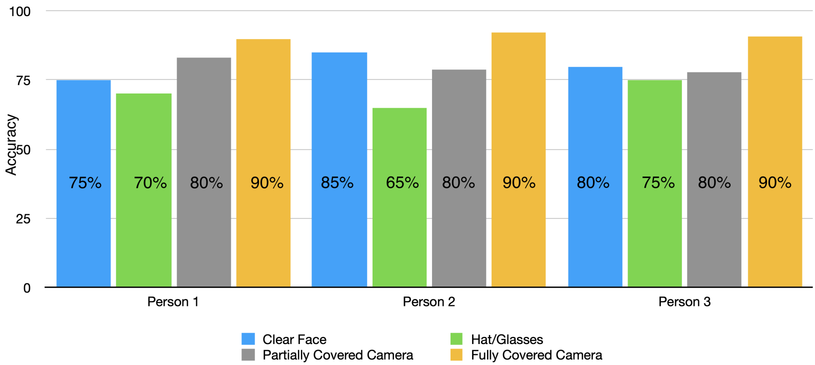


Figure: Real-Life Test

What Have We Accomplished

- This thesis demonstrated the possibility of creating accurate, very small-size ML face-recognition models
- Proof-of-Concept in a local interpreter, running completely custom trained model, and ready to be deployed to other devices

5. Future Work

- Implementing Face Detection on an embedded device is the next big step
  - Converting TF-Lite Model into compiled C/C++ code
  - Deploying above code as embedded-board-specific code modules
  - Building an app to leverage above ML-modules
- Automating model training is the other big topic to look into

BSc. Thesis Summary

- **Topic**: Machine Learning and Embedded ML
  - Dataset Compilation and Processing
  - Getting to know & Designing Model Architectures
  - Training said Models
- **Project 1**: Model Design and Training
  - Implemented several ML Models from Research Literature
  - Trained and Evaluated Performance of said Models
  - Compared Accuracy using Dataset Pictures and Trained Size
- **Project 2**: Model Performance Evaluation and Benchmark Application
  - Built demo-ed Interpreter App
  - Evaluated Real-World Performance

Thank you.

References

- TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers
  Daniel Situnayake and Pete Warden (2020)
- Labelled Faces in the Wild Dataset
  http://vis-www.cs.umass.edu/lfw/
- Aligned Faces Dataset
  http://www.cslab.openu.ac.il/download/wolftau/aligned$_{i}mages_{D}B.tar.gz$
- YouTube Faces dataset
  https://www.cs.tau.ac.il/ wolf/ytfaces/
- Tiny Darknet
  https://pjreddie.com/darknet/tiny-darknet/
- Backpropagation Applied to Handwritten Zip Code Recognition
  LeCun, Y. and Boser, B. and Denker, J. S. and Henderson, D. (1989)
- Very Deep Convolutional Networks for Large-Scale Image Recognition
  Karen Simonyan and Andrew Zisserman (2015)
  https://arxiv.org/pdf/1409.1556.pdf