Carnegie Mellon University
Language Technologies Institute

Lecture 2: Practical Applications, Projects, …

Yonatan Bisk & Emma Strubell

Carnegie Mellon University
Language Technologies Institute

Recognizing People in Photos Through Private On-Device Machine Learning

Floris Chabert, Jingwen Zhu, Brett Keating, and Vinay Sharma

Apple Research Blog July 2021

https://machinelearning.apple.com/research/recognizing-people-photos

Yonatan Bisk & Emma Strubell

Task

Find faces of contacts

Why is this hard?

Lighting, perspective,
skin color, age, gender, …

Why is this hard On-Device?

Motivation

On-Device face recognition is privacy preserving

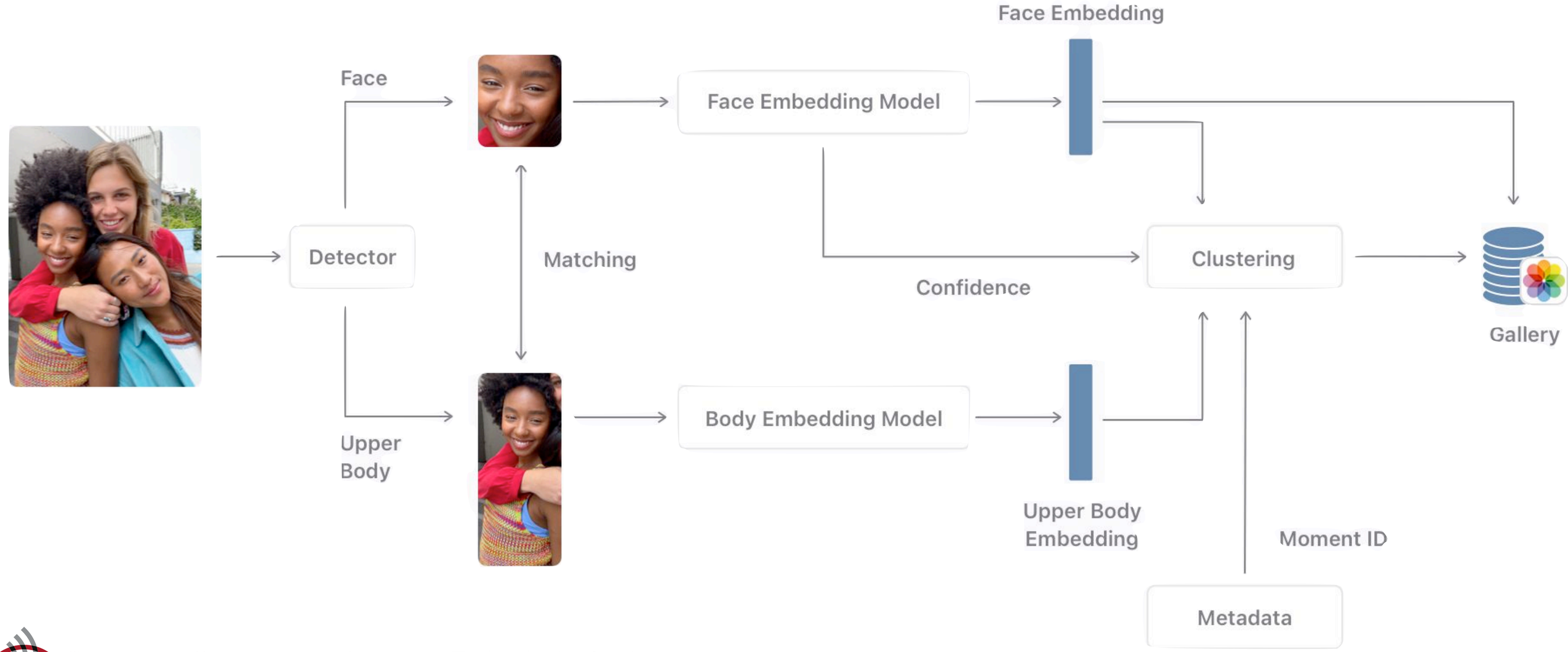
Context:

- Competitors in the market (e.g. Google) use cloud based services and so your data is shared
- Apple has their own neural engine for acceleration
- Quality vs Battery

What is a naive algorithm we might use?

Inference Pipeline

Notes:
1. Two feature representations (2 * model)
2. Agglomerative Clustering (naively expensive)
3. Use of external metadata (can correct for weak model)

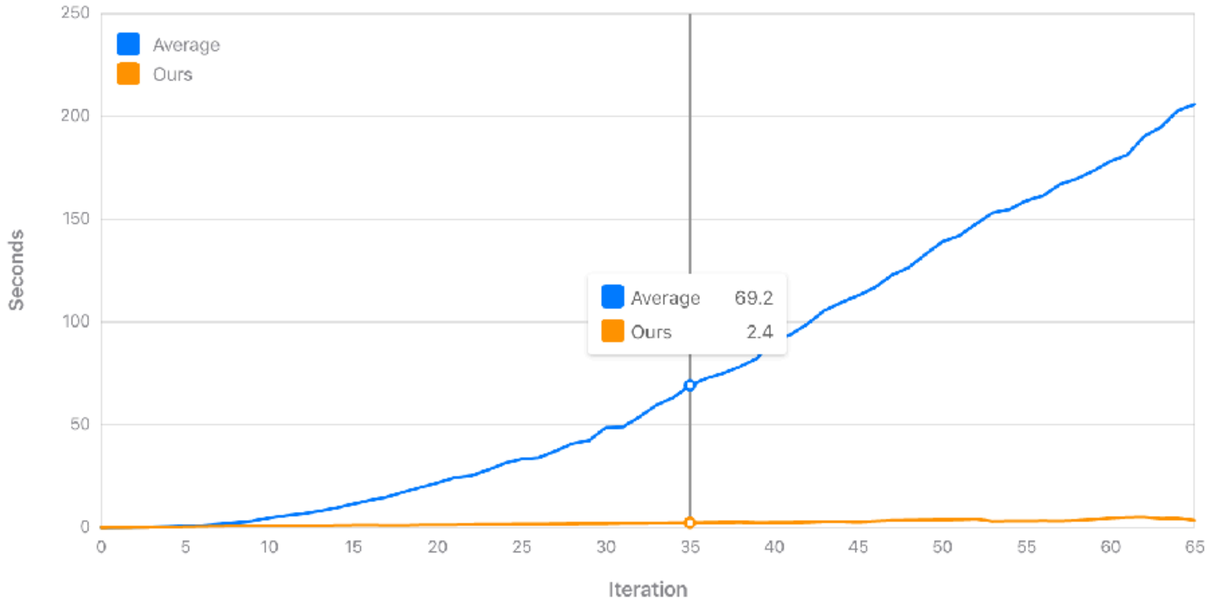


Clustering

1. Conservative Embedding clusters (very few merges - within moments?)
   Relies on hand-tuned weighting for Face (vs mean face) and body
   $D_{ij} = min(F_{ij}, \alpha \cdot F_{ij} + \beta \cdot T_{ij})$ where F & T are face and body, respectively

2. Agglomerative Clustering (Faces only)
   First pass (ideal): “median distance between the members of two HAC clusters”
   After threshold: “random sampling” ← Maintain linear runtime (no guarantees)

Note: runs periodically,
typically overnight during device charging

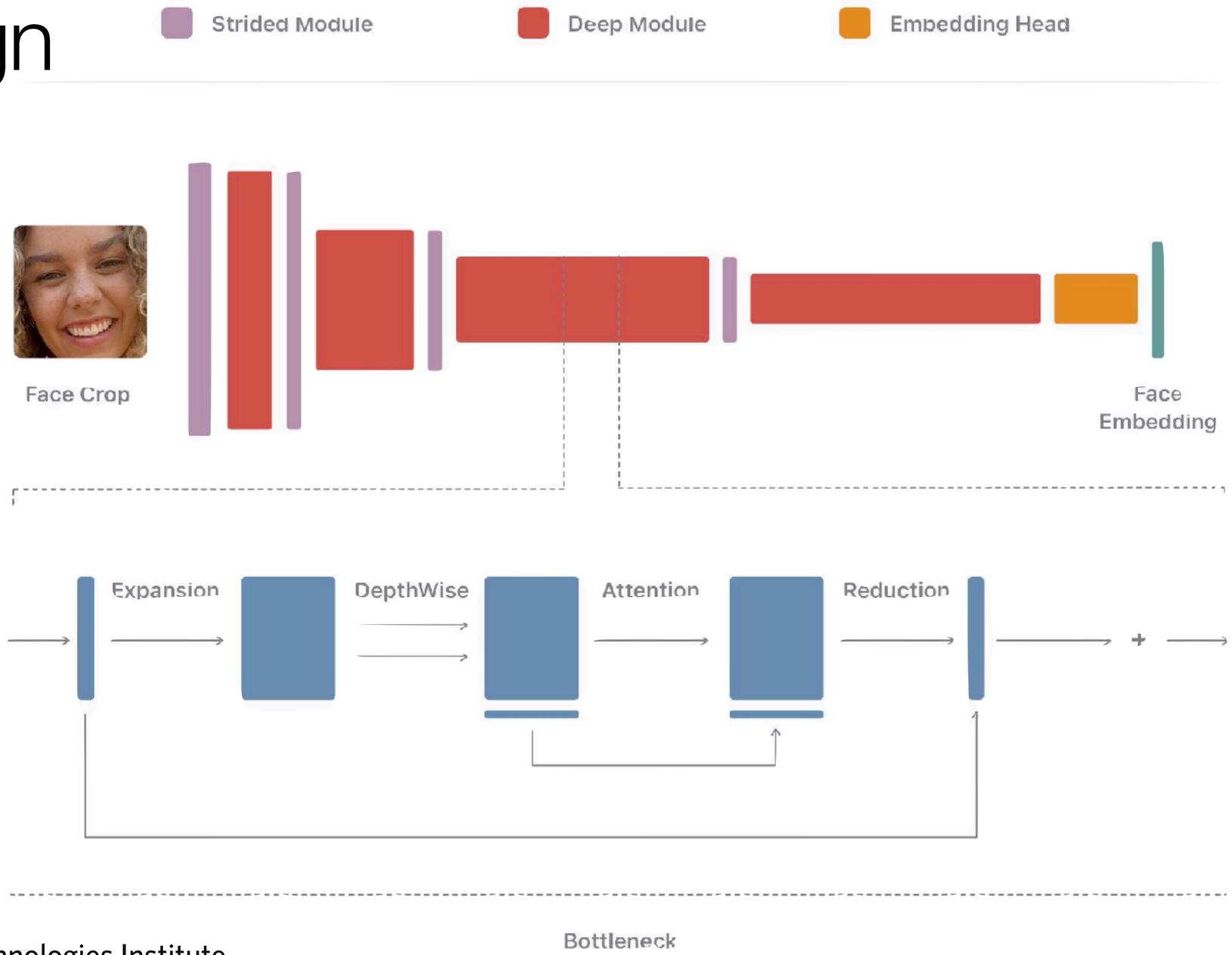


Assigning Identity

- Every cluster has c “canonical” examplars:
  $D = [X_0^1, X_1^1, \ldots X_c^1, X_0^2, X_1^2, \ldots X_c^2, X_1^K, \ldots X_c^K]$
- Construct a representation for the input as a function of the dictionary (existing clusters): $\min_x \|y - D \cdot x\|_2^2 + \lambda \cdot \|x\|_1$

  This reduces to a convex optimization (least squares) for the values in x
- So this is quickly learnable (optimally)
- Now the values in x_j for a given X_*^i define the cluster

Network Design

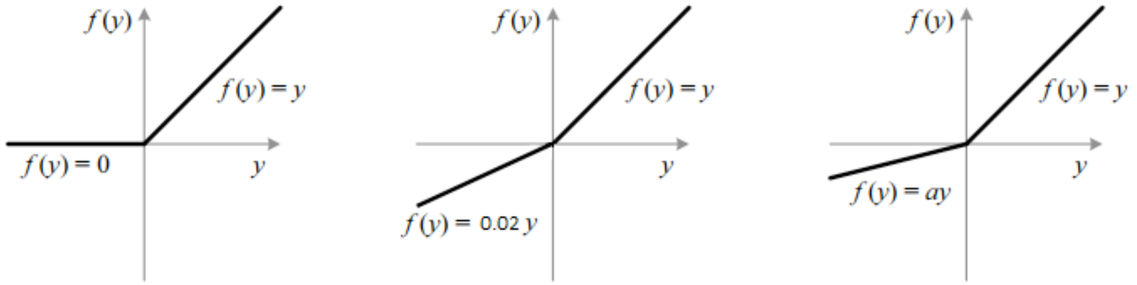


Network Design

"highest accuracy possible while running efficiently on-device, with low latency and a thin memory profile"

- Skipping important details here because the model is largely based around MobileNet which will be discussed later, BUT
- Double channels "within limits of computation"
- Bottleneck expansions are smaller and added attention at every layer
- PReLU

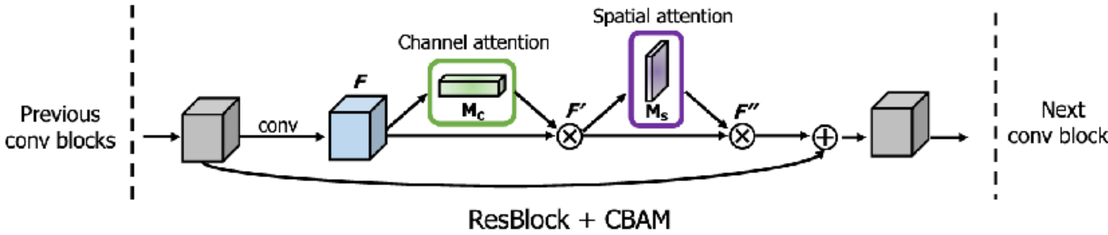
Input	Operator	t	c	n	s
112x112x3	conv 3x3	-	64	1	2
56x56x64	depthwise conv3x3	-	64	1	1
56x56x64	bottleneck	2	64	1	2
28x28x64	bottleneck	2	64	9	1
28x28x64	bottleneck	4	128	1	2
14x14x128	bottleneck	2	128	16	1
14x14x128	bottleneck	8	256	1	2
7x7x256	bottleneck	2	256	6	1
7x7x256	conv1x1	-	1024	1	1
7x7x1024	linear GDConv7x7	-	1024	1	1
1x1x1024	linear conv1x1	-	512	1	1



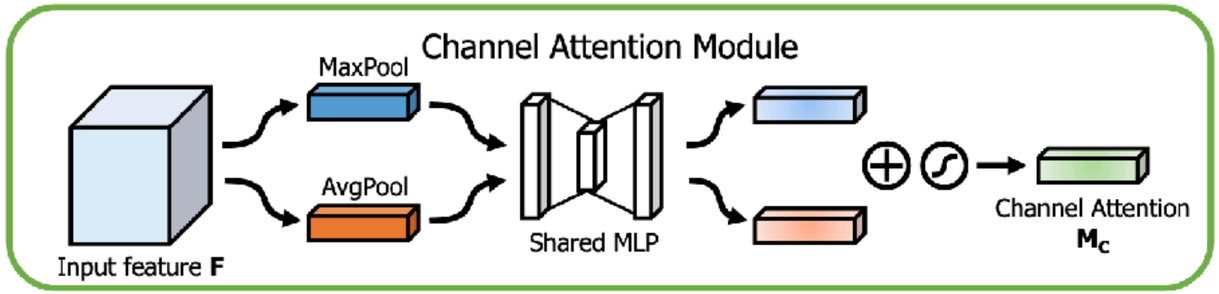
Network Design

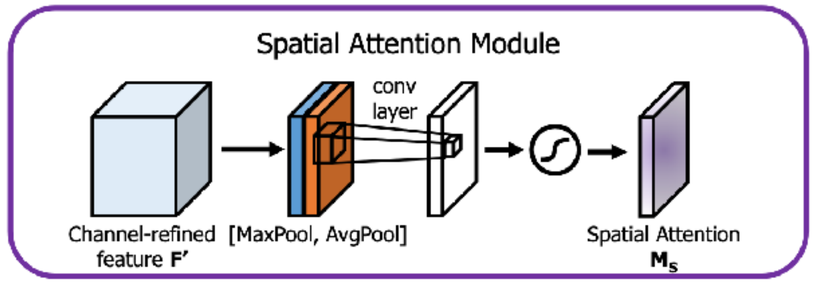
"highest accuracy possible while running efficiently on-device, with low latency and a thin memory profile"

- Wider ~= performance to deeper (but faster)
  Zagoruyko, S., Komodakis, N.: Wide residual networks



- Attention adds performance with little to no new parameters





$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$
$$= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),$$

$$\mathbf{M_s}(\mathbf{F}) = \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})]))$$
$$= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),$$

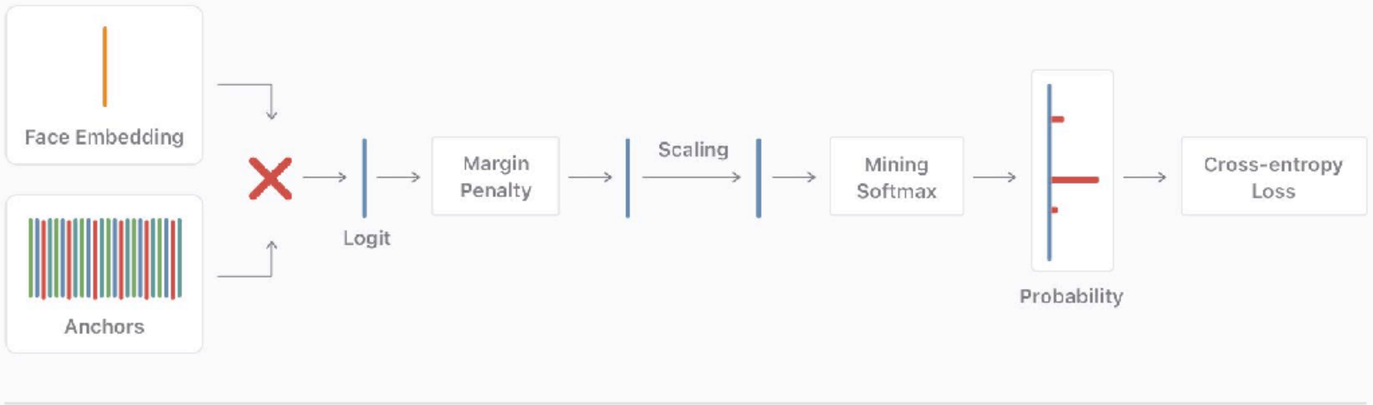
Performance of Attention

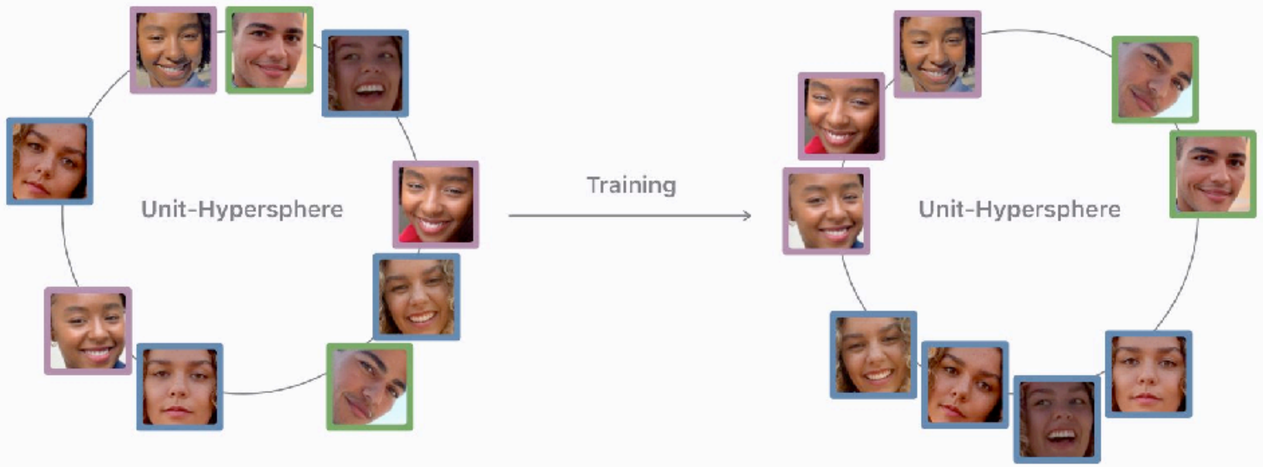
Architecture	Parameters	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
MobileNet [34] $\alpha = 0.7$	2.30M	0.283	34.86	13.69
MobileNet [34] $\alpha = 0.7$ + SE [28]	2.71M	0.283	32.50	12.49
MobileNet [34] $\alpha = 0.7$ + CBAM	2.71M	0.289	**31.51**	**11.48**
MobileNet [34]	4.23M	0.569	31.39	11.51
MobileNet [34] + SE [28]	5.07M	0.570	29.97	10.63
MobileNet [34] + CBAM	5.07M	0.576	**29.01**	**9.99**

* all results are reproduced in the PyTorch framework.

Table 5: **Classification results on ImageNet-1K using the light-weight network, MobileNet [34].** Single-crop validation errors are reported.

Training (Focus on normalization and cos)

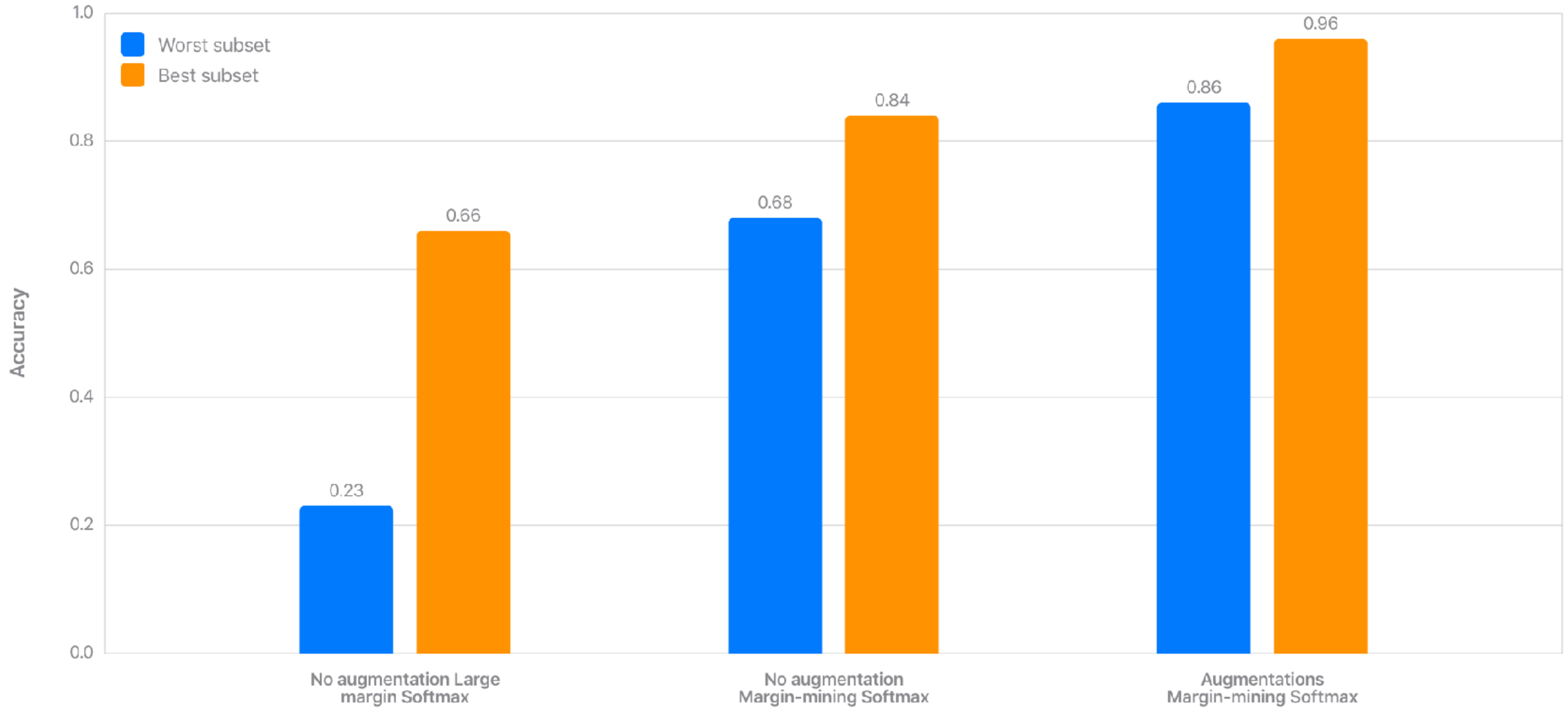




$$L_i = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j\neq y_i}^{K} e^{s\,f(\cos\theta_j)}}$$

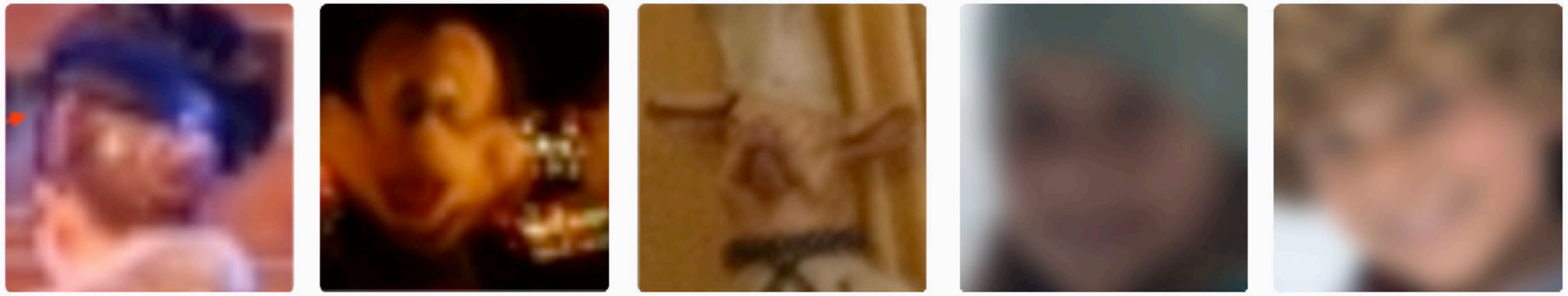
Margin ensures weighting on hard examples

Training (Focus on normalization and cos)



Other Considerations

1. Filtering Unclear Faces (no details)



2. Augmentations: *“pixel-level changes such as color jitter or grayscale conversion, structural changes like left-right flipping or distortion, Gaussian blur, random compression artifacts and cutout regularization”*

3. COVID-19: *“we designed a synthetic mask augmentation. We used face landmarks to generate a realistic shape corresponding to a face mask. We then overlaid random samples from clothing and other textures in the inferred mask area over the input face”*

Qualitative

Key Components

- Optimized clustering (constant time)
- Assignment via Convex coding (minimal updates)
- Wider (shallower) networks

Main Question I have:
1. Was the attention worth it?
2. Was this only possible because of the neural engine

Course Project

Anatomy of the Course Project

We provide:

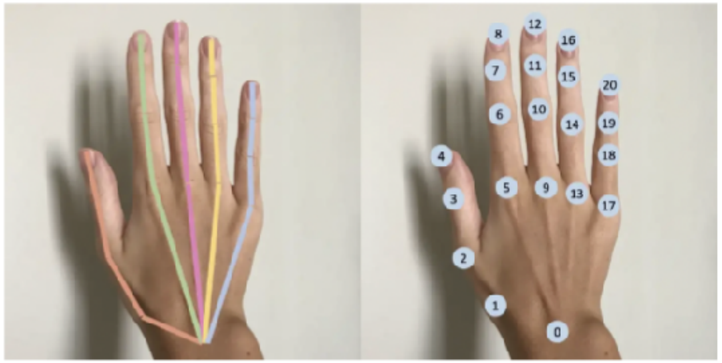
- **Lab 2:** Benchmarking
- **Lab 3:** Quantization
- **Lab 4:** Pruning
- **Lab 5:** Hardware, Energy & Carbon

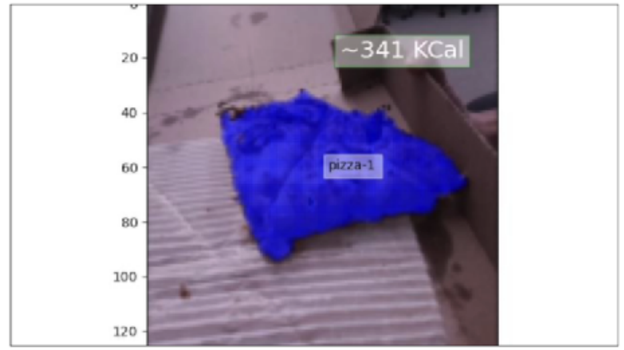
You decide:

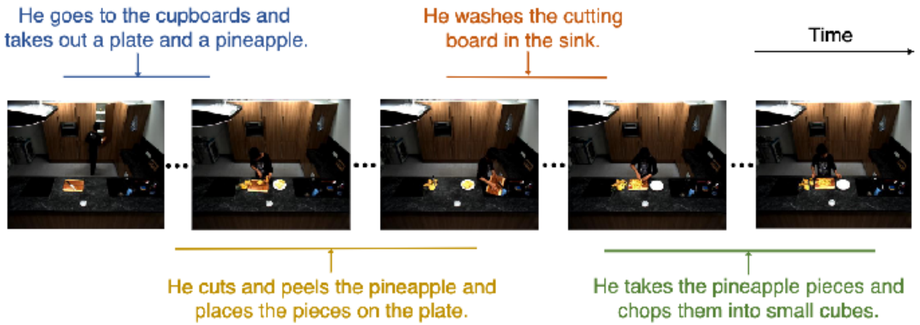
- **Hardware:** Laptop, robot, RPi…
- **Model:** ResNet, Transformer, encoder vs. decoder…
- **Data:** Language, vision, … Same as training data, or transfer/ adaptation setting?

Example Projects

- **AnySurface:** Converting any surface into a controller by compressing UNet, run on RPi4.
- **Speech-to-text translation:** Automatic speech recognition and translation on RPi4.
- **Im2Cal:** Estimating food calories from image by compressing Segformer, on RPi4.
- **Hey where's that thing:** Temporal localization in videos by compressing 2D-TAN on laptop.
- **Shazaam:** On-device music recognition w/ FAISS, separable convolutions.







Axes to Consider

- **Theory or practice?** Resource Optimized vs Resource Constrained
- **Target hardware:**
  CPU + RAM vs GPU/M1 + Shared RAM vs GPU+CPU + Separate RAM
- **Hardware support:** Logic, quantization, sparse ops, batching…
- **Novelty:** Reproduction vs Transfer (new data/hardware) vs Novel
- **In-distribution or transfer:** Fitting to in-distribution data, vs. adapting to a new task or domain?

Efficiency in Theory versus Practice

Resource Optimized

- Magnitude pruning
- Server
- Quantization (3-bit)

Resource Constrained

- Structured pruning / layer pruning
- Edge device
- Quantization (8-bit) if hw supports

Target hardware considerations

- Where do you store model weights, activations, gradients?
  How does this impact latency?
- Trade-off between storage size, speed, and on-the-fly computation
- Do I want on-device training? Fine-tuning?
- How heavy is the OS? How heavy are USB vs GPIO?
- Does your hardware support efficient batched computation? Efficient low-bitwidth computation? Efficient control flow?

Project Ideas

Resource Optimized

- Does efficient method X published in a CV venue apply to NLP, or vice versa?
- Does theoretically proven idea Y published in ML venue apply to larger, more complex models and datasets?

Resource Constrained

- Does “efficient” method Z, evaluated on GPU/TPU, work on CPU/Edge? Under memory constraints? Power constraints?
- Can you further optimize an already-efficient model?
  Can you compress a huge model enough to fit it on device?

All of the above

- Compare existing methods across different metrics: Pareto optimality, generalization, fairness, …

Learning Goals

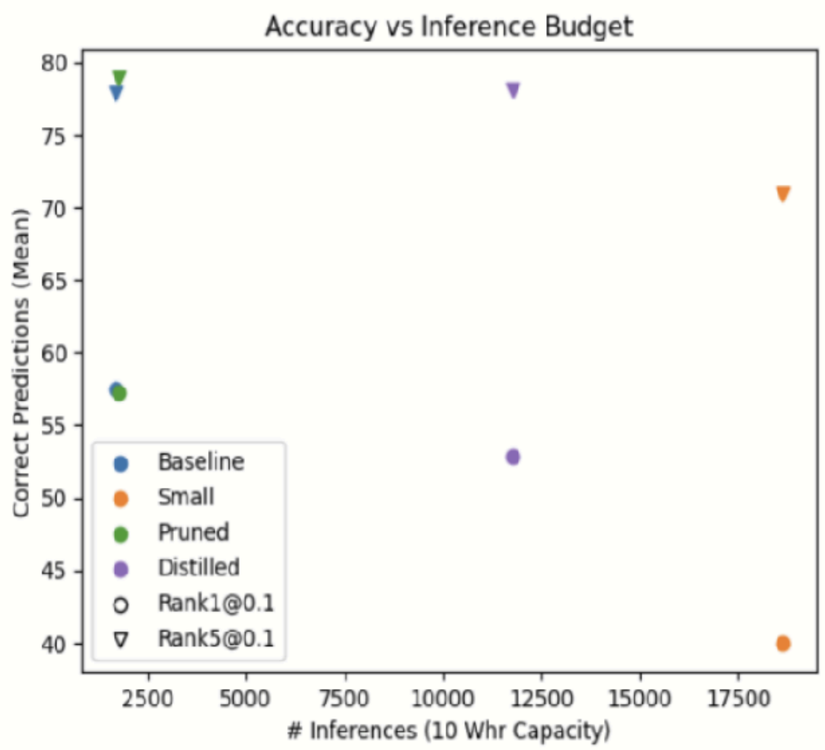
Project is **not**

- Entrepreneurship 101
- Multimodal Machine Learning (amazing class)
- Graded based on model performance
- Real world robotics

Project **is**

- Measuring Efficiency and Power
- Adjusting data for 👆
- Changing architectures for 👆
- Producing Pareto curves for 👆

Plotting Goals



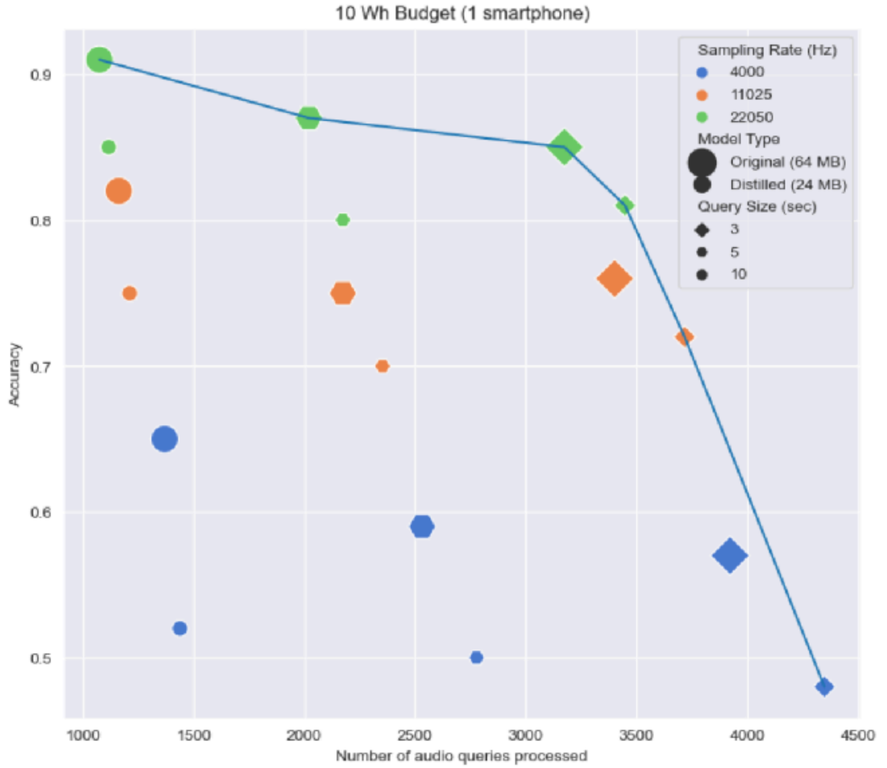


Figure 2: Accuracy vs. Efficiency Trade-off

Where to start?

- What pre-trained models exist for my task?
- What is a baseline I can feasibly train in a few hours?
- How can I sub-sample my data to create a feasible train/test set?
  - Single domain? Limited Label space? Simplified Task?
- Want — Performance that's non-trivial but do not need competitive performance
- **What is unique about my data/task/… that makes me think I can compress my models?**