Creating smaller, faster, production-worthy mobile machine learning models

Jameson Toole  

O’Reilly AI London, 2019
“We showcase this approach by training an 8.3 billion parameter transformer language model with 8-way model parallelism and 64-way data parallelism on 512 GPUs, making it the largest transformer based language model ever trained at 24x the size of BERT and 5.6x the size of GPT-2.” - MegatronLM, 2019
Are we going in the right direction?
Training Megatron-ML from scratch: $0.3\text{ kW} \times 220\text{ hours} \times 512\text{ GPUs} = 33,914\text{ kw}$

3X yearly energy consumption of the average American

**Common carbon footprint benchmarks**

in lbs of CO2 equivalent

- Roundtrip flight b/w NY and SF (1 passenger): 1,984
- Human life (avg. 1 year): 11,023
- American life (avg. 1 year): 36,156
- US car including fuel (avg. 1 lifetime): 126,000
- Transformer (213M parameters) w/ neural architecture search: 626,155

Does my model enable the largest number of people to iterate as fast as possible using the fewest amount resources on the most devices?
How do you teach a microwave its name?
How do you teach a microwave its name?

AmazonBasics Microwave
Voice-controlled microwave

Edge intelligence: small, efficient neural networks that run directly on-device.
How do you teach a ____ to ____?
Edge Intelligence is necessary and inevitable.

**Latency:** too much data, too fast

**Power:** radios use too much energy

**Connectivity:** internet access isn’t guaranteed

**Cost:** compute and bandwidth aren’t free

**Privacy:** some data should stay in the hands of users
Most intelligence will be at the edge.

- <100M servers
- 3B phones
- 12B IoT
- 150B embedded devices

= 1 billion devices
The Edge Intelligence lifecycle.
Model selection

75MB: Avg size of Top-100 app

348KB: SRAM SparkFun Edge Development Board
Model selection: macro-architecture

Design Principles

- Keep activation maps large by downsampling later or using atrous (dilated) convolutions
- Use more channels, but fewer layers
- Spend more time optimizing expensive input and output blocks, they are usually 15-25% of your computation cost
Model selection: macro-architecture

Layers

- Depthwise Separable Convolutions
- Bilinear upsampling

Backbones

- MobileNet (20mb)
- SqueezeNet (5mb)

8-9X reduction in computation cost

https://arxiv.org/abs/1704.04861
Model selection: micro-architecture

Design Principles

- Add a width multiplier to control the number of parameters with a hyperparameter: kernel x kernel x channel x w
- Use 1x1 convolutions instead of 3x3 convolutions where possible
- Arrange layers so they can be fused before inference (e.g. bias + batch norm)
Training small, fast models

Most neural networks are massively over-parameterized.
Knowledge distillation: a smaller “student” network learns from a larger “teacher”

Results:

1. ResNet on CIFAR10:
   a. 46X smaller,
   b. 10% less accurate
2. ResNet on ImageNet:
   a. 2X smaller
   b. 2% less accurate
3. TinyBert on Squad:
   a. 7.5X smaller,
   b. 3% less accurate

https://nervanasystems.github.io/distiller/knowledge_distillation.html
https://arxiv.org/abs/1802.05668v1
Training small, fast models: pruning

**Iterative pruning**: periodically removing unimportant weights and / or filters during training.

Results:

1. AlexNet and VGG on ImageNet:
   a. Weight Level: 9-11X smaller
   b. Filter Level: 2-3X smaller
   c. No accuracy loss
2. No clear consensus on whether pruning is required vs training smaller networks from scratch.

https://arxiv.org/abs/1506.02626
https://arxiv.org/abs/1608.08710
https://arxiv.org/abs/1810.05270v2
https://arxiv.org/abs/1510.00149v5
Compressing models via quantization

32-bit floating point precision is (usually) unnecessary. Quantizing weights to fixed precision integers decreases size and (sometimes) increases speed.
Compressing models via quantization

**Post-training quantization**: train networks normally, quantize once after training.

**Training aware quantization**: periodically removing unimportant weights and / or filters during training.

**Weights and activations**: quantize both weights and activations to increase speed

**Results:**

1. **Post-training 8-bit quantization**: 4X smaller with <2% accuracy loss
2. **Training aware quantization**: 8-16X smaller with minimal accuracy loss
3. **Quantizing weights and activations**: can result in a 2-3X speed increase on CPUs

https://arxiv.org/abs/1806.08342
Deployment: embracing combinatorics
Deployment: embracing combinatorics

Design Principles

- Train multiple models targeting different devices: OS x device
- Use native formats and frameworks
- Leverage available DSPs
- Monitor performance across devices
Putting it all together
Putting it all together

Edge Intelligence Lifecycle

- Model selection: use efficient layers, parameterize model size
- Training: distill / prune for 2-10X smaller models, little accuracy loss
- Quantization: 8-bit models 4X smaller, 2-3X faster, no accuracy loss
- Deployment: use native formats that leverage available DSPs
- Improvement: put the right model on the right device at the right time
Putting it all together

1.6 million parameters

6327 kb / 7 fps iPhone X

225x smaller

6,300 parameters

28kb / +50 fps iPhone X
Putting it all together

“TinyBERT is empirically effective and achieves comparable results with BERT in GLUE datasets, while being 7.5x smaller and 9.4x faster on inference.” - Jiao et al

“Our method reduced the size of VGG-16 by 49x from 552MB to 11.3MB, again with no loss of accuracy.” - Han et al

“The model itself takes up less than 20KB of Flash storage space ... and it only needs 30KB of RAM to operate.” - Peter Warden at TensorFlow Dev Summit 2019
Open questions and future work

Need better support for quantized operations.

Need more rigorous study of model optimization vs task complexity.

Will platform-aware architecture search be helpful?

Can MLIR solve the combinatorics problem?
Complete Platform for Edge Intelligence

Deploy ML/AI models on all your mobile devices

- Build
  - Train your own model
  - Use one of ours

- Release
  - Optimize
  - Cross-platform portability
  - Protect

- Manage
  - Analytics
  - Monitoring
  - OTA Update

Native Model
Developer API
Monitoring
iOS & Android
Complete Platform for Edge Intelligence

Add ML-powered features to iOS and Android apps with only a few lines of code. Use our APIs with ready-to-use ML models baked right in. What will you build with Fritz?

- **Object Detection**: Identify and track objects within an image or video.
- **Image Labeling**: Recognize people, places, and things with hundreds of labels.
- **Style Transfer**: Turn photos and videos into unique masterpieces based on real art.
- **Image Segmentation**: Recognize people and objects to separate them from the background.
- **Pose Estimation**: Track movement and orientation of key body parts.
Benefits of using Fritz

Mobile Developers
- Prepared + Pretrained
- Simple APIs
- Fast, Secure, On-device

Machine Learning Engineers
- Iterate on Mobile
- Benchmark + Optimize
- Analytics

Try yourself:
Fritz AI Studio

App Store
Google Play
Working at the edge?

https://www.fritz.ai

Join the community!

heartbeat.fritz.ai

Questions?
@jamesonthecrow
jameson@fritz.ai