Efficient neural networks

Workflow

Workstation

Frameworks

PYTORCH CNTK ONNX Chainer Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Prop. format

Runtime

Inference engines

- TensorRT
- OpenVino
- ONNX runtime
- -specific APIs (CoreML, Android NN, etc)

- They take advantage of the low-bit precision and oriented on speed and efficiency
- Also take advantage of their prop. Hardware capabilities

Workstation

Frameworks

PYTORCH CNTK Caffe Caffe2 ONNX Chainer mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Frameworks

- **ONNX** – Open neural network exchange (tends to become a standard for representation)
- **Caffe –** very fast for computer vision jobs
- **Tensorflow** – very good scalability, community support, but slow
- **Pytorch** – good scalability
- **Mxnet** – good integration with AWS, fast
- **CNTK –** good integration with Windows

Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

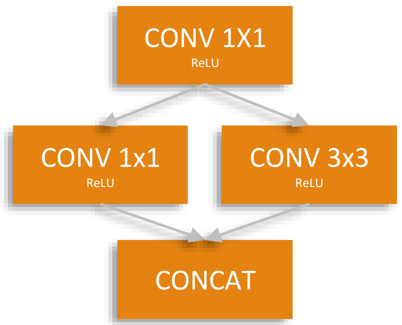
- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

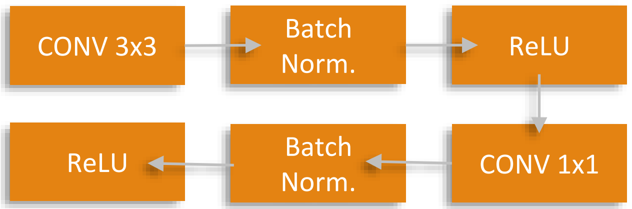
- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Model

- Use mobile architecture such as SqueezeNet and MobilenetV2
- **SqueezeNet Fire Module**:



- **MobileNet Dephtwise Conv Module**



Workstation

Frameworks

ONNX Chainer
PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

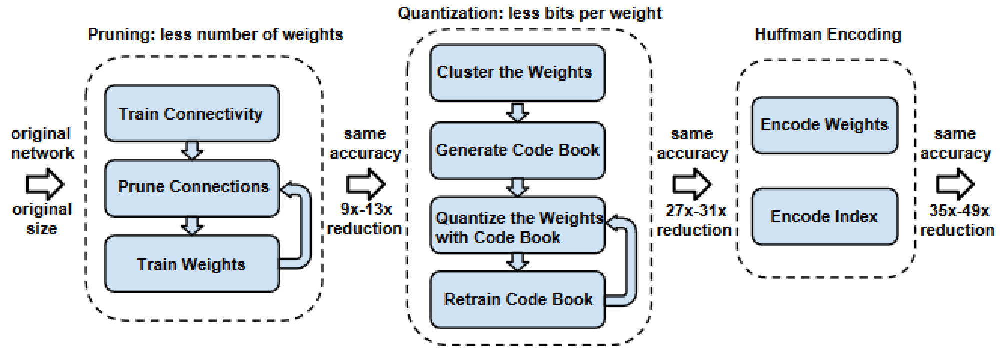
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Workstation

Frameworks

ONNX, Chainer, PYTORCH, CNTK, Caffe, Caffe2, mxnet, TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

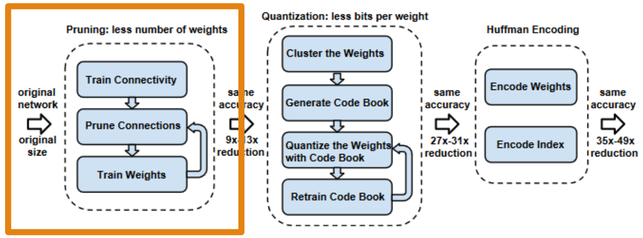
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

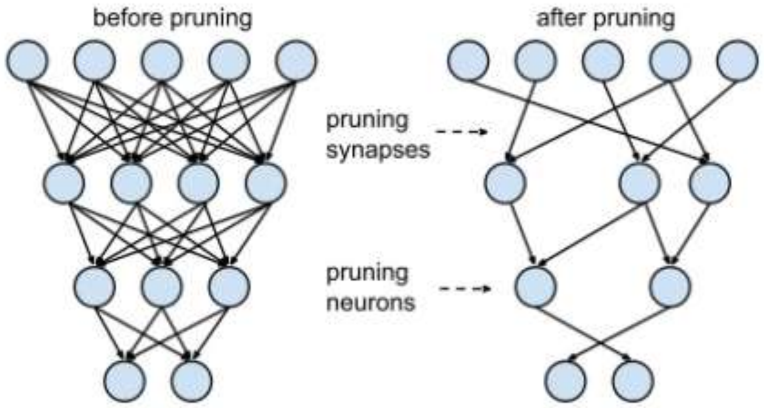
- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Pruning

- During training, once in a while remove the weights/neurons/synapsis that are under a certain threshold
- Redundant information is pruned



Workstation

Frameworks

ONNX, Chainer, PyTorch, CNTK, Caffe, Caffe2, mxnet, TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

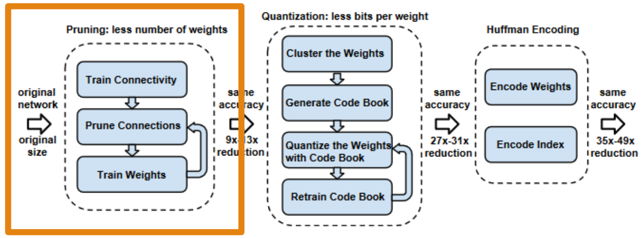
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

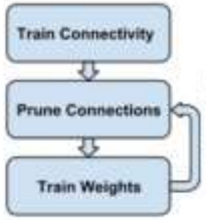
Compiler

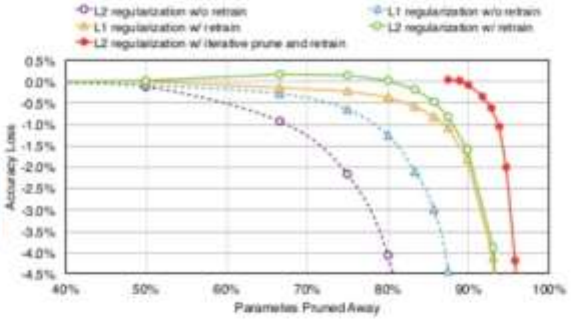
- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Pruning





Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12×**
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12×**
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9×**
VGG16 Ref	31.50%	11.32%	138M	
VGG16 Pruned	31.34%	10.88%	**10.3M**	**13×**

Table 1: Network pruning can save 9× to 13× parameters with no drop in predictive performance

Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

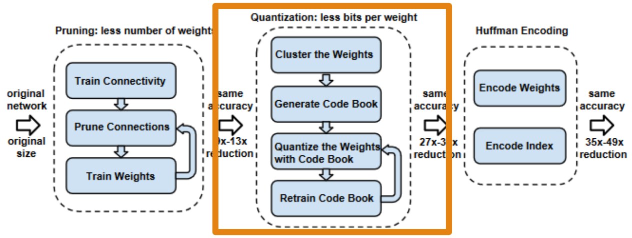
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

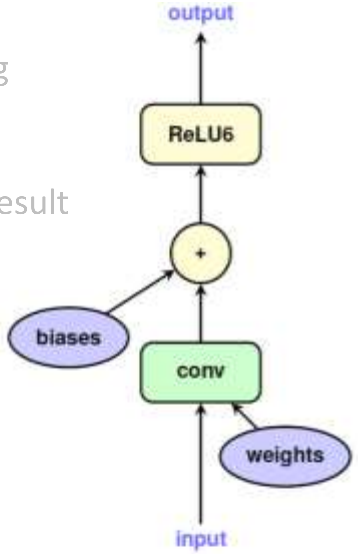
- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

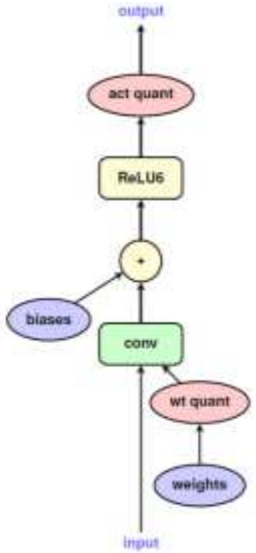
Training



Weight sharing and quantization

- Train the network being aware that it is going to be quantized
- Weight clustering can result in a higher compression rate





Workstation

Frameworks

ONNX, Chainer, PyTorch, CNTK, Caffe, Caffe2, mxnet, TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

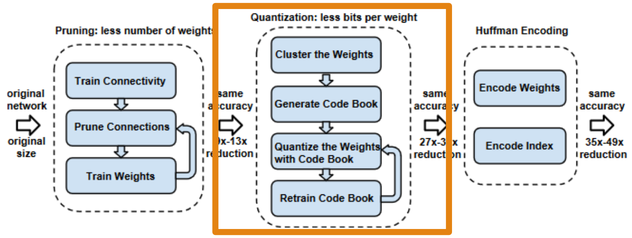
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
    - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

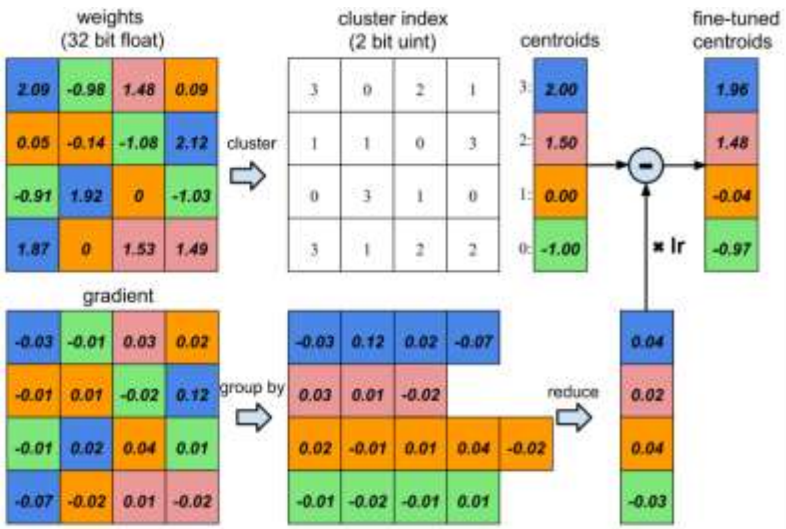
Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Weight sharing and clustering



Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

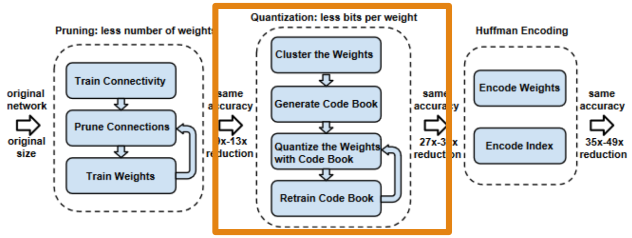
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

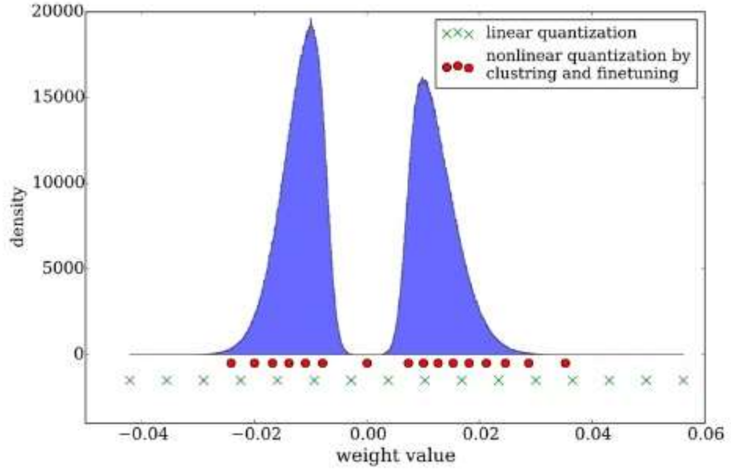
- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Weight sharing and clustering

- Train the model and recording the weight interval and mean value, such that the conversion from FP32 to INT8 will be with as little loss as it can be



Workstation

Frameworks

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

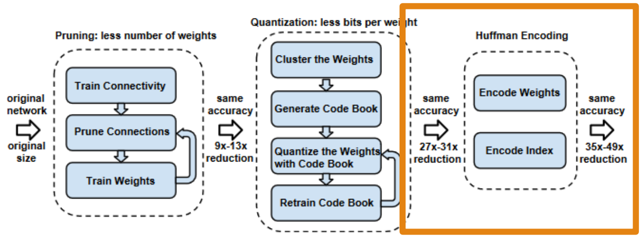
Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Training



Huffman coding

- apply Huffman coding to take advantage of the biased distribution of effective weights
- Huffman coding saves non-uniformly distributed values 20%–30% of network storage

Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Compiler

TensorFlowLite TensorRT OPENVINO™

Model Optimizer is a cross-platform tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices.

Outputs a propietary format that will work with the inference engine.

Can perform the following **optimizations**:

- Layer & Tensor Fusion
- FP16 INT8 Precision Calibration
- Kernel Autotuning

Workstation

Frameworks

ONNX, Chainer, PyTorch, CNTK, Caffe, Caffe2, mxnet, TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

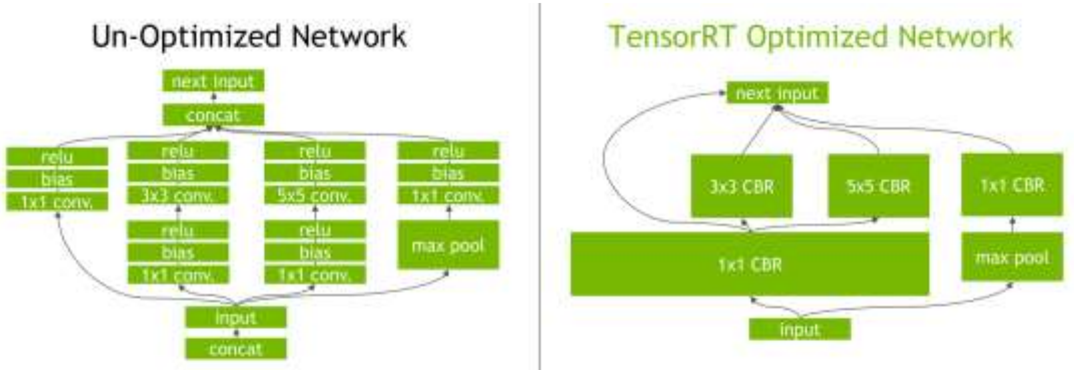
Compiler

TensorFlow Lite TensorRT OpenVINO™

Model Optimizer is a cross-platform tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices.

Layer & Tensor Fusion

- restructure the graph to perform the operations much faster and more efficiently.



Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Compiler

TensorFlowLite TensorRT OpenVINO™

Model Optimizer is a cross-platform tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices.

Precision Calibration

- given a calibration batch, the optimizer will assure the required precision after quantization

Workstation

Frameworks

ONNX Chainer PYTORCH CNTK Caffe Caffe2 mxnet TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
    - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Compiler

TensorFlowLite TensorRT OpenVINO™

Model Optimizer is a cross-platform tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices.

Kernel Autotuning

- pick the implementation from a library of kernels that delivers the best performance for the target GPU, input data size, filter size, tensor layout, batch size and other parameters.
- ex: Conv 1x1 will be done as an matrix multiplication rather than a proper Convolution

Workflow

Workstation

Frameworks

ONNX, Chainer, PYTORCH, CNTK, Caffe, Caffe2, mxnet, TensorFlow

Model

- Use a mobile network architectures such as: SqueezeNet, MobileNet (segmentation, detection, classification)

Training

- Pruning: prune weights that are under a certain threshold and remove isolated neurons or conv filters
- Quantization-aware training:
  - represent weights from FP32 to INT8 or even INT4 and train by keeping that in mind

Compiler

- Output own format for their inference engines
- Optimize graph architecture: fuse layers
- Quantize networks, perform calibration if needed
- TensorRT, OpenVino, ONNX optimizer, TF Lite compiler, Core ML, etc

Prop. format

Mobile Device

Inference engines

- TensorRT
- OpenVino
- ONNX runtime
- -specific APIs (CoreML, Android NN, etc)

- They take advantage of the low-bit precision and oriented on speed and efficiency
- Also take advantage of their prop. Hardware capabilities

Recap

SO FAR WE'VE IMPORTED A TRAINED MODEL INTO THE MODEL OPTIMIZER, AND PERFORMED A NUMBER OF OPTIMIZATIONS TO GENERATE A RUNTIME ENGINE

Workflow

RUNTIME

The runtime is the platform where you would like your model to run. Such platforms can be GPU server, mobile phones, mobile devices, autonomous vehicles, IoT application, etc

This is done through the inference engine. The inference engine just takes care that the optimized graphs are run accordingly and take full advantage of the hardware.

Mobile Device

Inference engines

- TensorRT
- OpenVino
- ONNX runtime
- -specific APIs (CoreML, Android NN, etc)

- They take advantage of the low-bit precision and oriented on speed and efficiency
- Also take advantage of their prop. Hardware capabilities