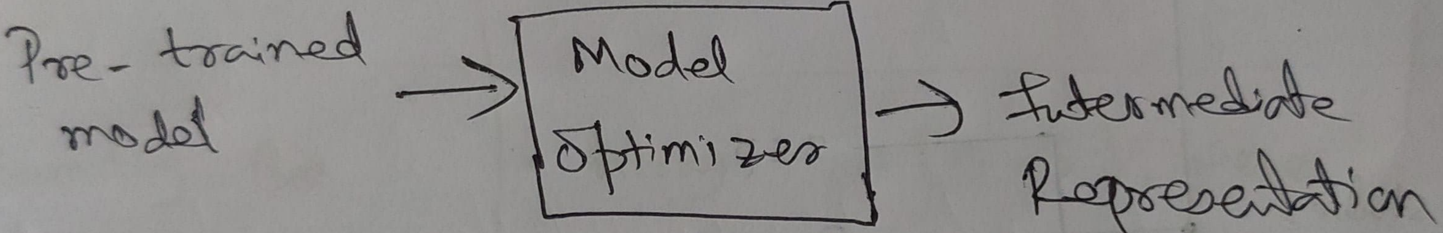
③ THE MODEL OPTIMIZER

- Pre-trained models
- The Model Optimizer
- The Inference Engine
- Deploying the app

Basically, a model optimizer is a converter that converts models into an intermediate representation (IR), that can than be fed to an inference engine.



Improvements:

- Model size
- Speed

Trade-offs

- Accuracy (minimized)

Note: Using a pre-trained model is mandatory.

Optimization Techniques

① Quantization

- Refers to the number of bits used to represent the weights and biases of the model.

Model before optimizatn	Model after optimizatn
↑ accuracy	↓ accuracy (not substantial)
↑ size	↓ size
↓ compute ~~time~~ speed	↑ compute ~~time~~ speed

- Model precisions

~~FP32~~	Default precision ~~FP32~~	Present in OpenVINO	Supported by Model Optimizer
FP32	✓	✓	✓
FP16	X	✓	✓
INT8	X	✓	X

② Freezing

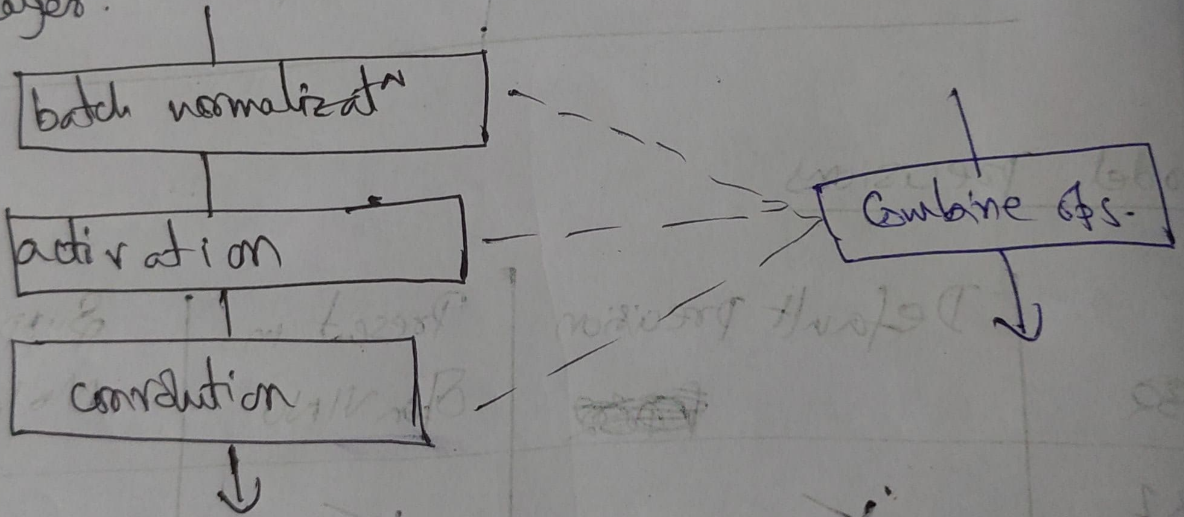
- Used in context of Tensorflow models.
- Removes operations and meta data only needed for training and not inference

Eg: backpropagation.

③ fusion

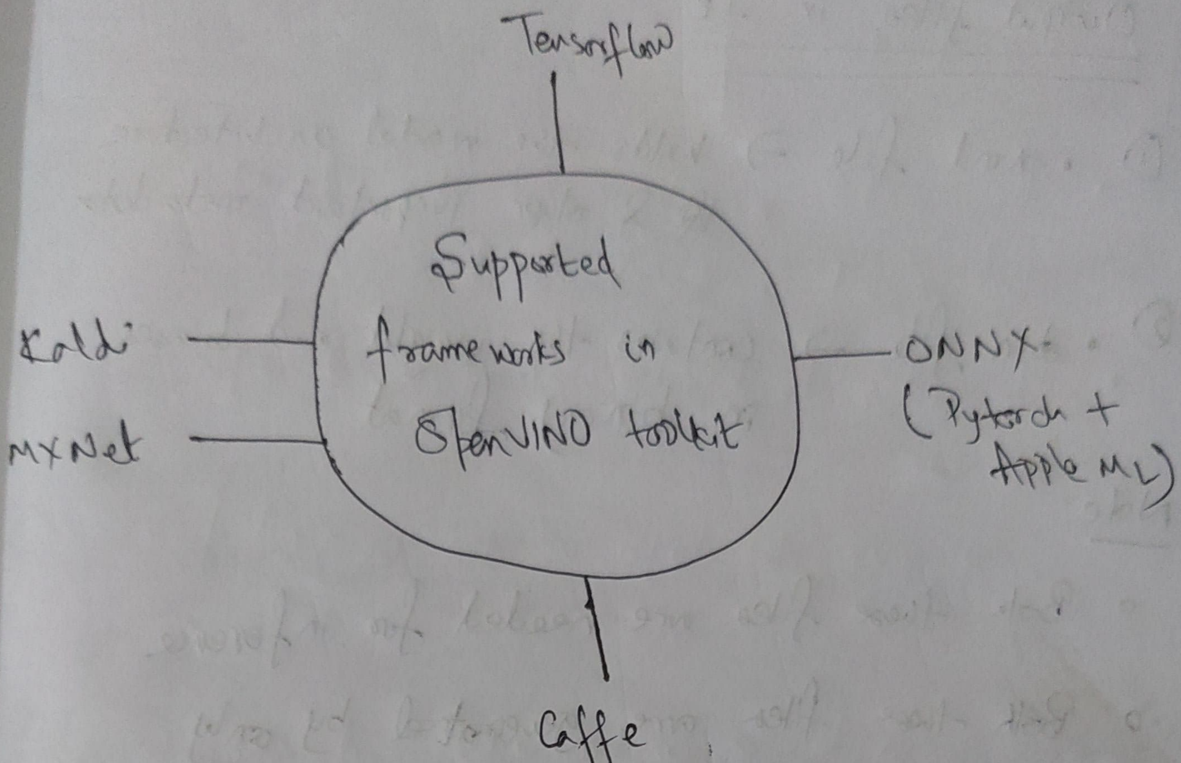
- Combines multiple layers operations into a single layer.

Eg.



- Useful when diff. operations perform in diff. kernels, but fused operation performs in one kernel

⟺ Overhead decreases while switching from one kernel to next


Tensorflow
Supported frameworks in OpenVINO toolkit
Kaldi
MXNet
ONNX (Pytorch + Apple ML)
Caffe

Intermediate Representations

- Refers to the standard structure and naming of neural network architectures in OpenVINO toolkit.

⇒ like "shared dialect" for neural network layers across all the supported frameworks.

Eg: ~~[illegible]~~ Conv2D (tensorflow) | Convolution (IR)

Convolution (caffe)

Conv (ONNX)

- The model Optimizer does this translation into a shared dialect that the Inference Engine can understand.

Output files in IR

① .xml file → holds the model architecture & other important metadata

② .bin file → contains the weight and biases in binary format

Note:

- Both these files are needed for inference
- Both these files are generated by only specifying the precision to Model Optimizer using -- data-type flag.

  (By default, the data type has FP32 precision)

- For unfrozen model, use --mean_values and --scale (in case of tensorflow).
- for tensorflow models, remember to:

  i) reverse i/p channels (BGR → RGB)

  ii) use custom operatns config

  iii) apply object detection API pipeline conf

Cutting Parts of a Model

why

- Model has pre- / post-processing parts that don't translate to existing IR layers.
- Model has training part which is not needed during inference.
- Model is too complex w/ many unsupported applications.
- Model is one of the SSD models
  ⇒ cut-off the post-processing part
- Cutting the model helps localize any issue in model conversion

How

CLI args:

i) --input: if cutting from beginning

ii) --output: if cutting from end

Custom Layers

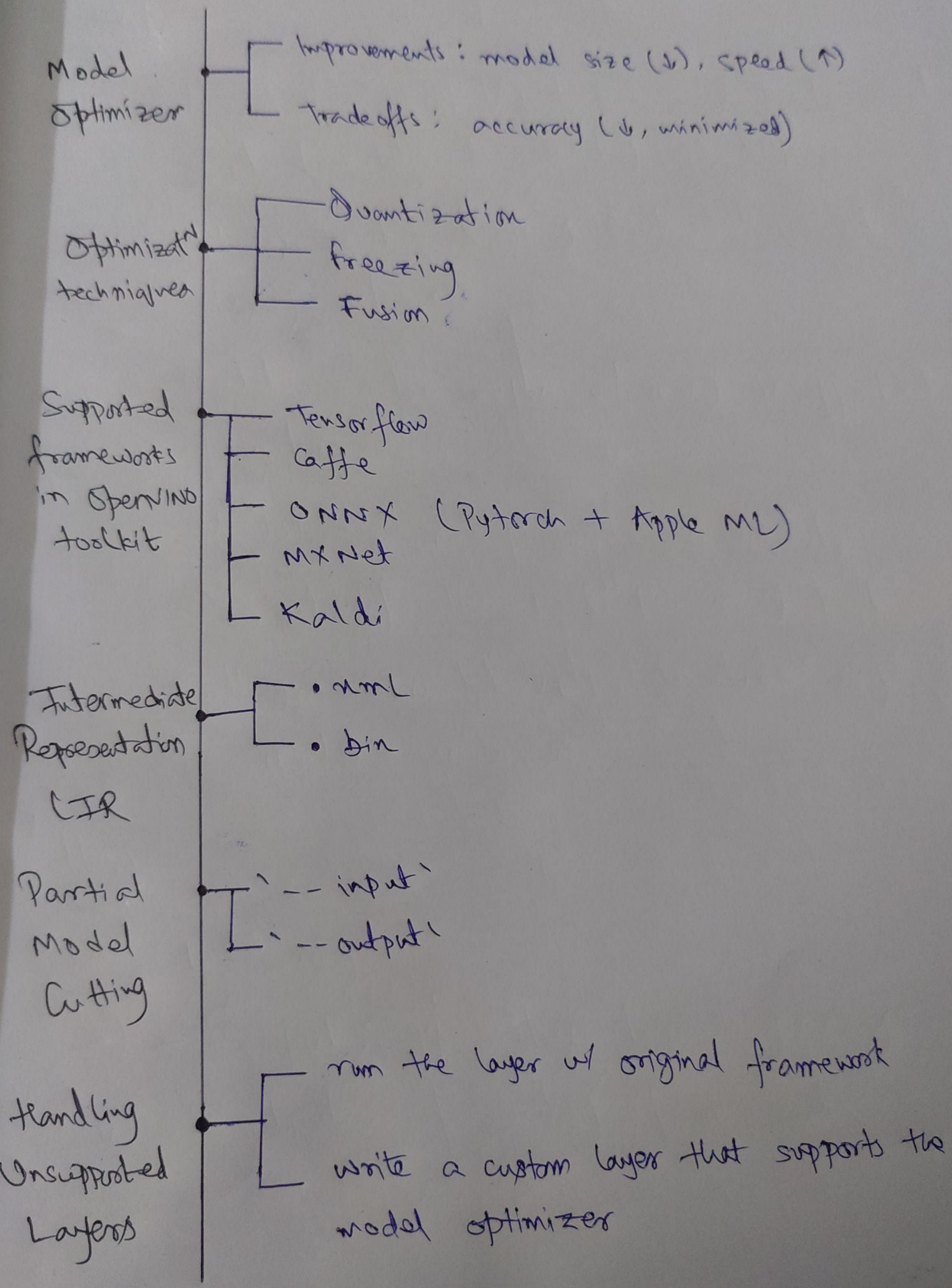
- The model may have unsupported layers that need to be handled.
- In order to handle unsupported layers, we can:
  - run the layer w/ the original framework,

    OR
  - write a custom layer that supports the model optimizer

Adding Custom Layers

Tensorflow	Caffe
1. Register the custom layers as extensions to the model optimizer	Register the custom layers as extensions to the mod optimizer
2. Replace the unsupported subgraph w/ a different subgraph	Register the layers as Custom
Offload the computation of the subgraph back to TF during inference.	Use Caffe to calculate t o/p of the Shape of the layer.

Summary



- Model Optimizer
  - Improvements: model size (↓), speed (↑)
  - Tradeoffs: accuracy (↓, minimized)
- Optimizat^{n} techniques
  - Quantization
  - Freezing
  - Fusion
- Supported frameworks in OpenVINO toolkit
  - Tensorflow
  - Caffe
  - ONNX (Pytorch + Apple ML)
  - MXNet
  - Kaldi
- Intermediate Representation (IR
  - .xml
  - .bin
- Partial Model Cutting
  - '--input'
  - '--output'
- Handling Unsupported Layers
  - run the layer w/ original framework
  - write a custom layer that supports the model optimizer