ML-EXray: Visibility into ML Deployment on the Edge

Hang Qiu, Ioanna Vavelidou, Jian Li, Evgenya Pergament,
Pete Warden, Sandeep Chinchali, Zain Asgar, Sachin Katti

Stanford University

Google

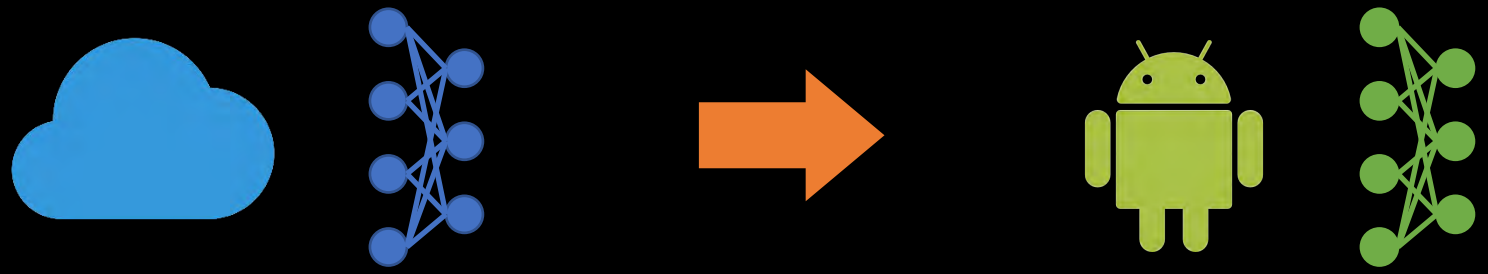
Edge ML

Low Latency

Privacy & Security

Low Power

ML Deployment on the Edge



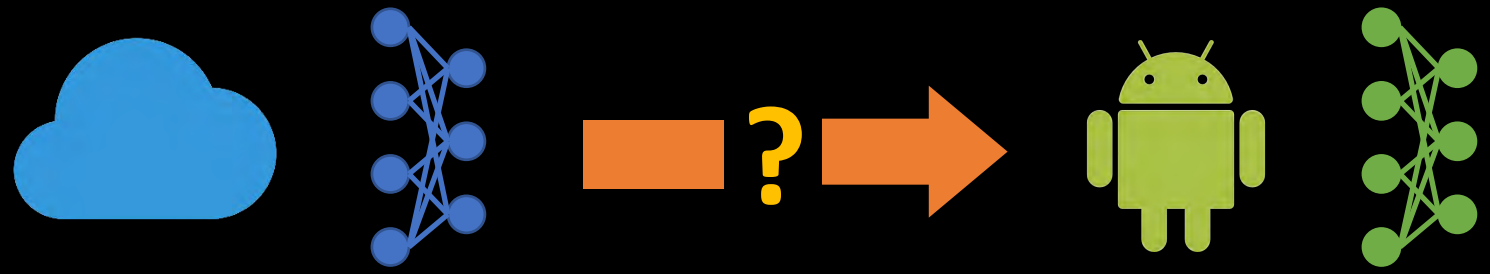
Dog: 0.95

Cat: 0.33

Deployed model can have *mismatching* performance, and often there is *little clue* what this may happen.

There is a *disconnect* between model developers and app developers

The Disconnect



Model Performance

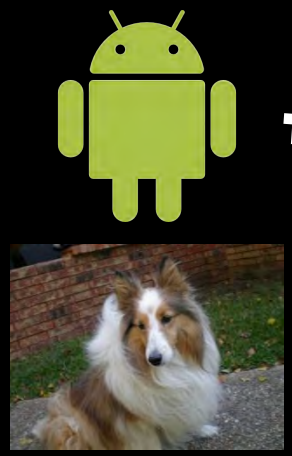
- Accuracy
- Latency & throughput

Deployment Performance

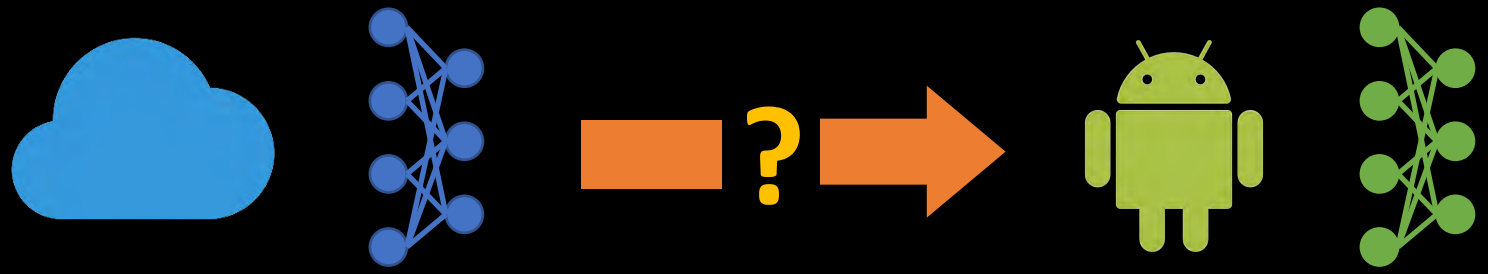
- End-to-end latency
- Power consumption
- Memory footprint

Design choices made for training are often *lost during the handoff*
Design choices may *conflict with heterogenous hardware*

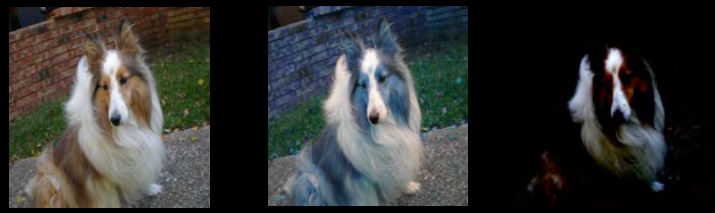
An Image Classifier Example



What could possibly go wrong?



- Preprocessing bugs



- Quantization issues

- Kernel optimization v.s. heterogenous hardware

Challenge: How to debug?

- *Low awareness* of potential issues
- *Little visibility* into the edge black box
- *Tedious reverse engineering* to debug

Dog: 0.95

RGB [0,1]

Dog: 0.45

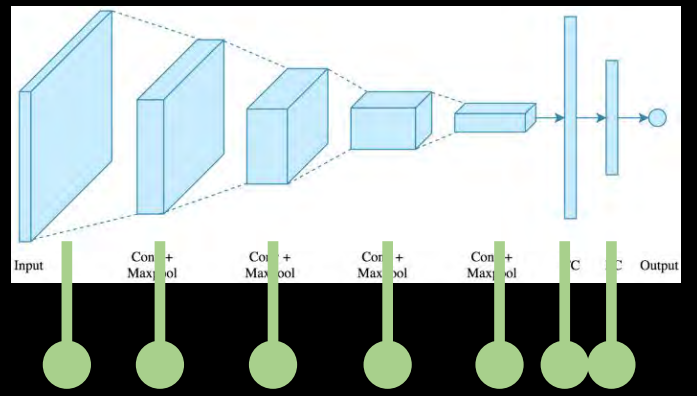
BGR [0,1]

Dog: 0.25

RGB [-1,1]

Coral

ARDUINO

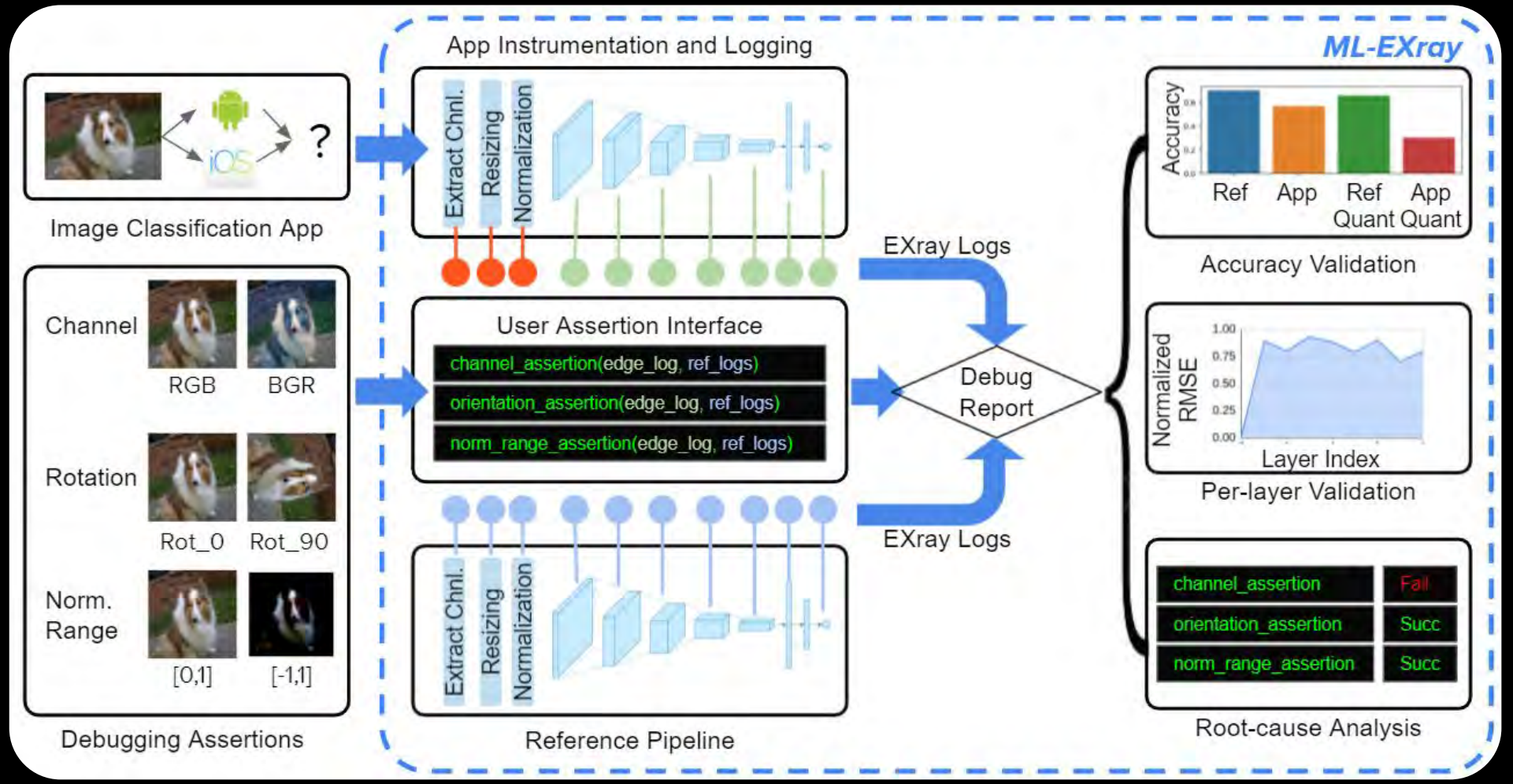


Per-layer weight, bias, output

ML-EXray Contributions

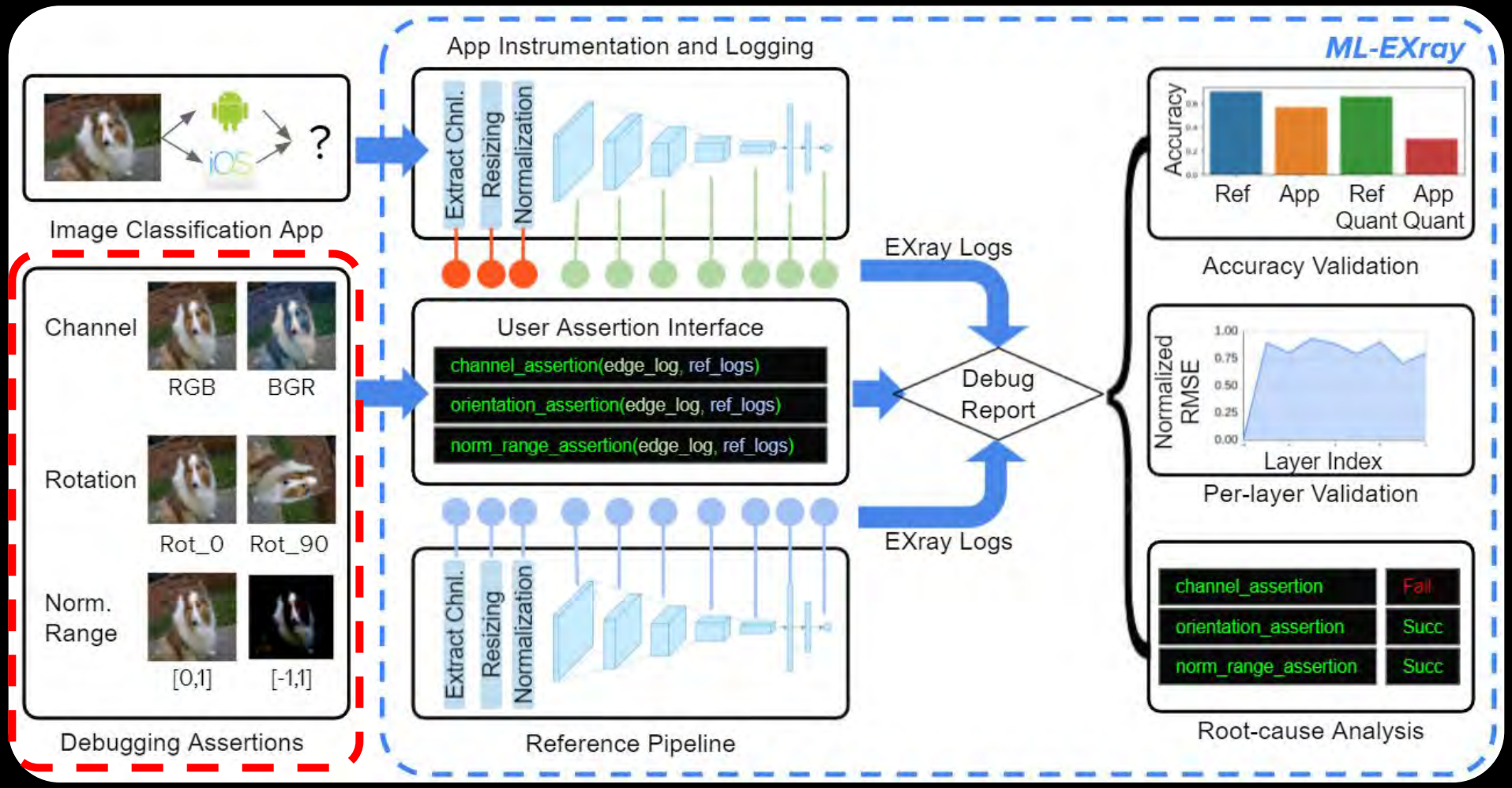
- *Visibility:* Instrumentation APIs for layer-level details
- *Bridging the disconnect:* Reference pipelines and data playback
- *Automated debugging:* Programming model for deployment validation
- *Awareness of deployment issues:* Uncovering deployment issues and their impact

ML-EXray Overview



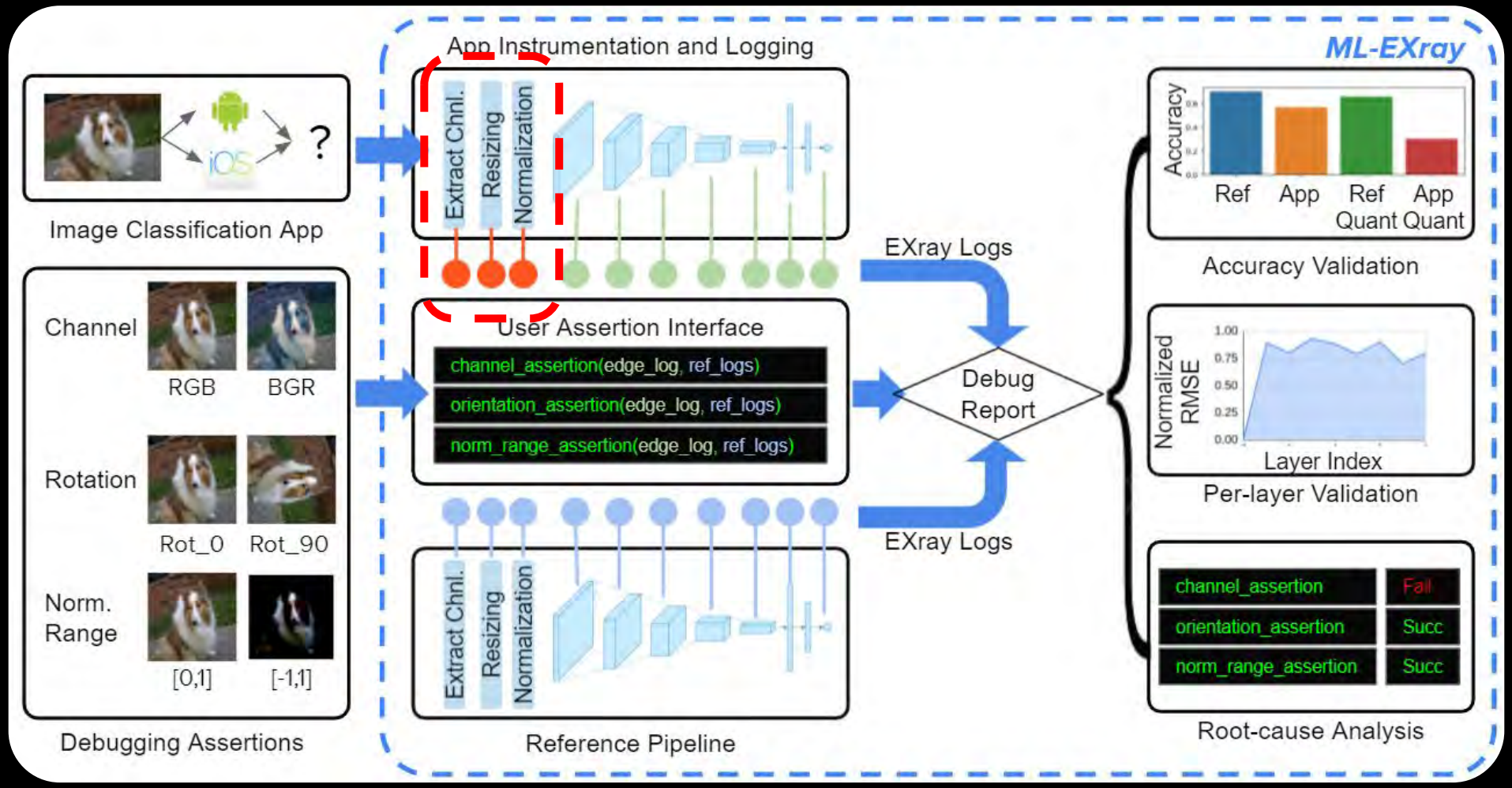
- Easy-to-use API w/ low overhead
- End-to-end performance validation
- Per-layer latency measurement and output validation
- Extensibility: user-defined logging, reference pipeline, and assertions

Customization



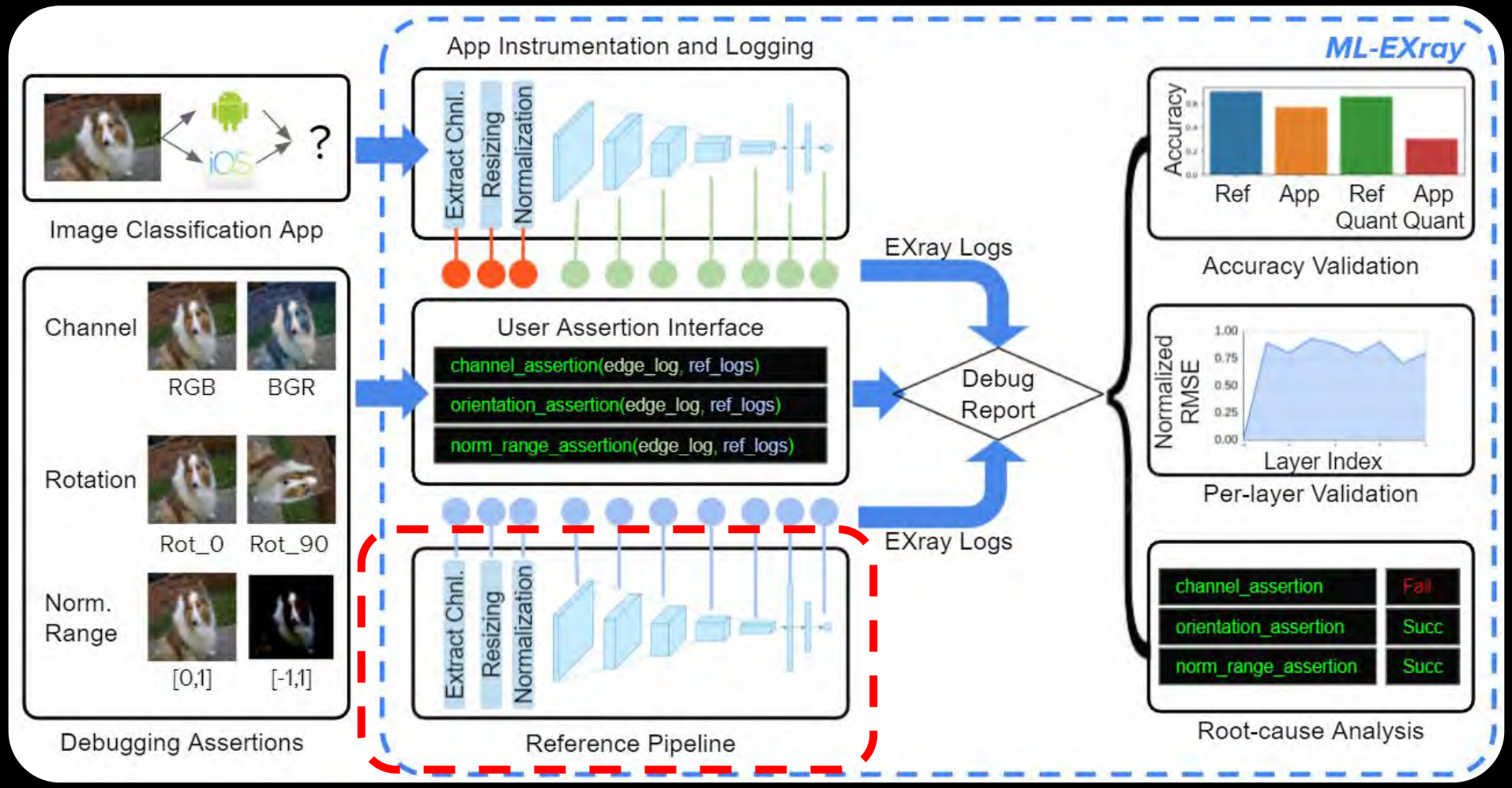
***Assertion functions**: users can define custom debugging assertions to validate suspected issues, such as input channel arrangement, orientation, and normalization range.*

Customization



***Log elements**:* users can instrument the app at any point throughout the pipeline to log custom variables for different debugging purposes.

Customization



***Reference Pipelines**:* users can provide alternative pipelines as references, such as a previous successful deployment pipeline on a different device.

Instrumentation APIs & Data Model

A suite of APIs in C++, Java, Python

```
// (C++)
MLEXray->on_inf_start();
TfLiteStatus s = m_interpreter->Invoke();
MLEXray->on_inf_stop(&m_interpreter);
```

Assertion function

```
def channel_assertion(edge_out, ref_out) {
    if not np.allclose(edge_out, ref_out):
        edge_out = cv2.cvtColor(edge_out,
        cv2.COLOR_BGR2RGB)
        if np.allclose(edge_out, ref_out):
            raise AssertionError('BGR->RGB')}
```

Data Model

Input / Output

- Model I/O
- Per-layer I/O
- Pre-processing I/O
- User-defined I/O

Performance

- End-to-end Latency
- Per-layer Latency
- Memory usage

Peripheral

- Orientation
- Motion
- Lighting

Reference Pipelines and Playback

Reference ML Pipeline

- Cloud model
- Correct pre-processing
- No model optimization
- Default standard kernels

Input / Output

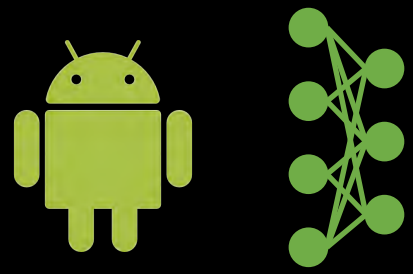
- Model I/O
- Per-layer I/O
- Pre-processing I/O
- User-defined I/O

Performance

- End-to-end Latency
- Per-layer Latency
- Memory usage

Peripheral

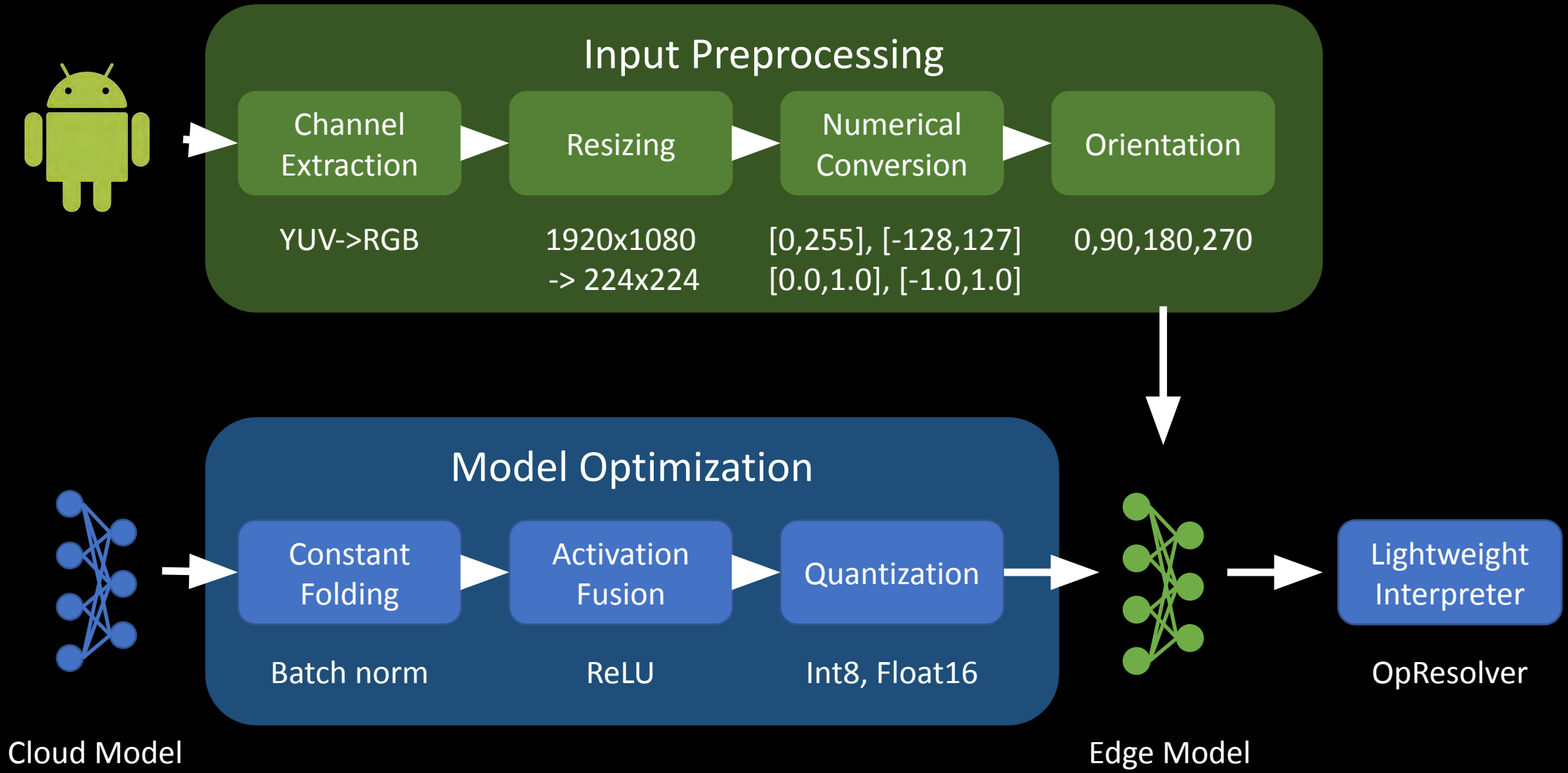
- Orientation
- Motion
- Lighting



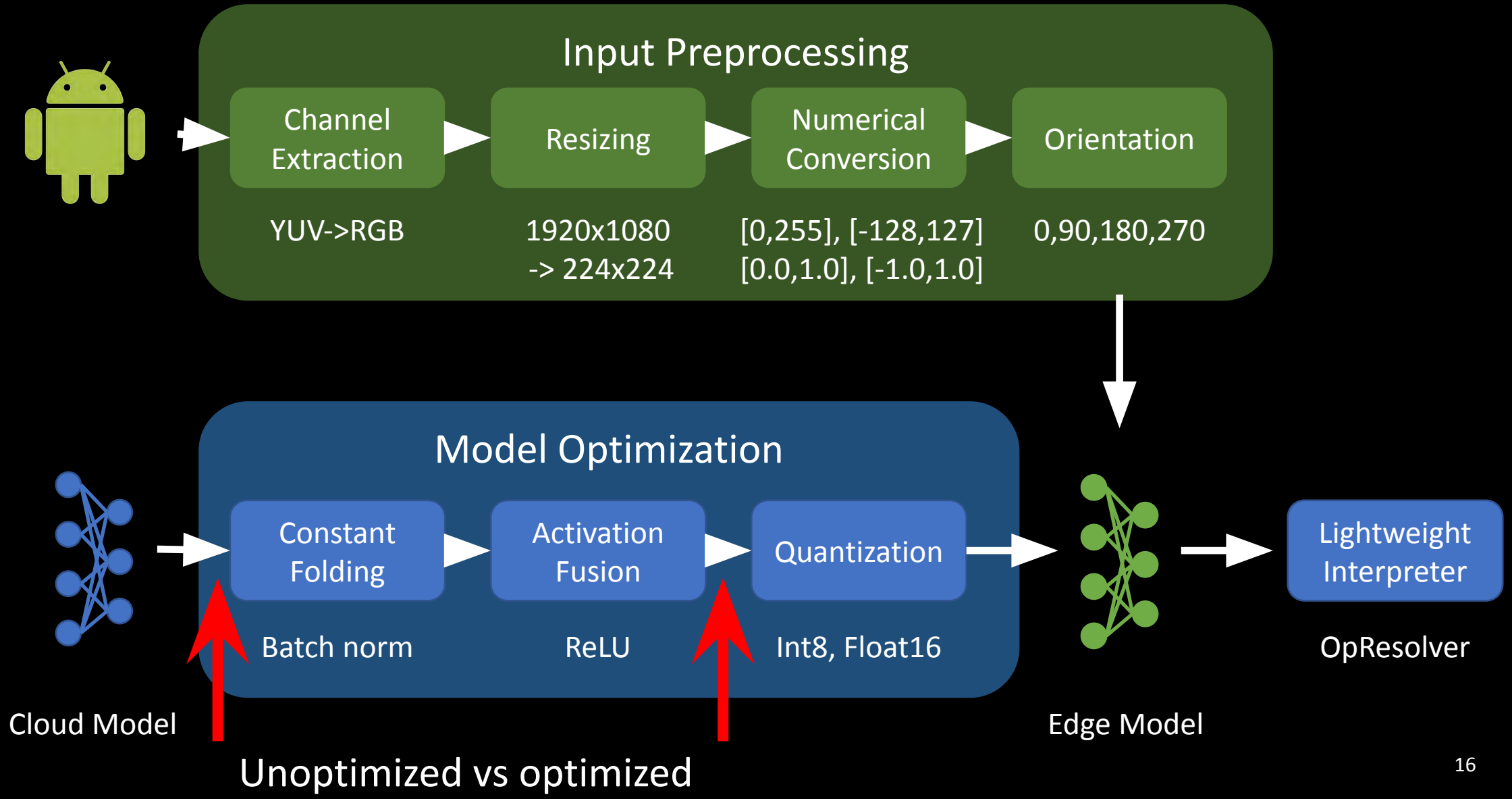
Edge ML Pipeline

- Edge model
- User pre-processing code
- Optimized model
- Edge device specific kernels (e.g. optimized op resolver)

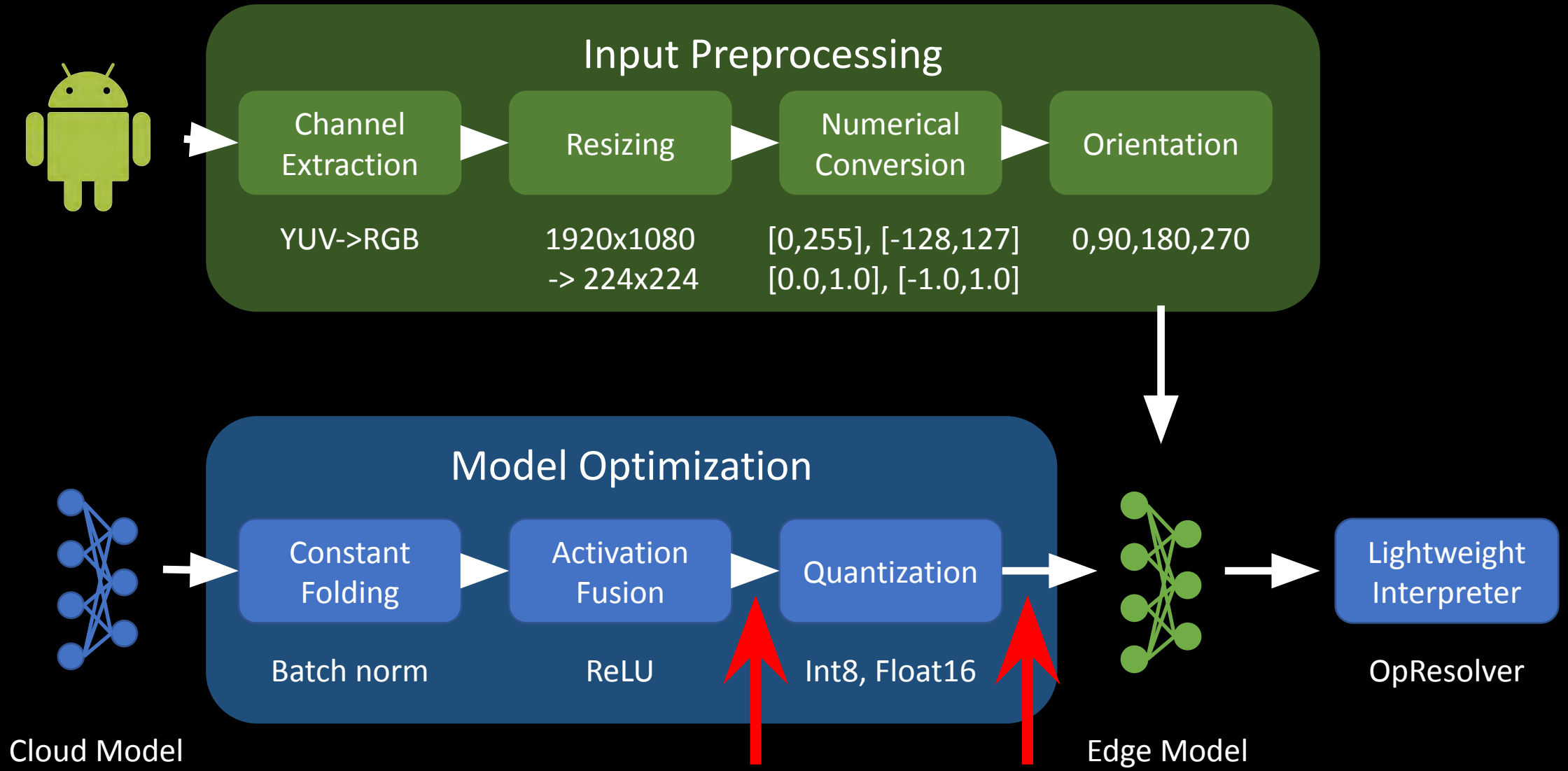
A Basic Reference Pipeline Example



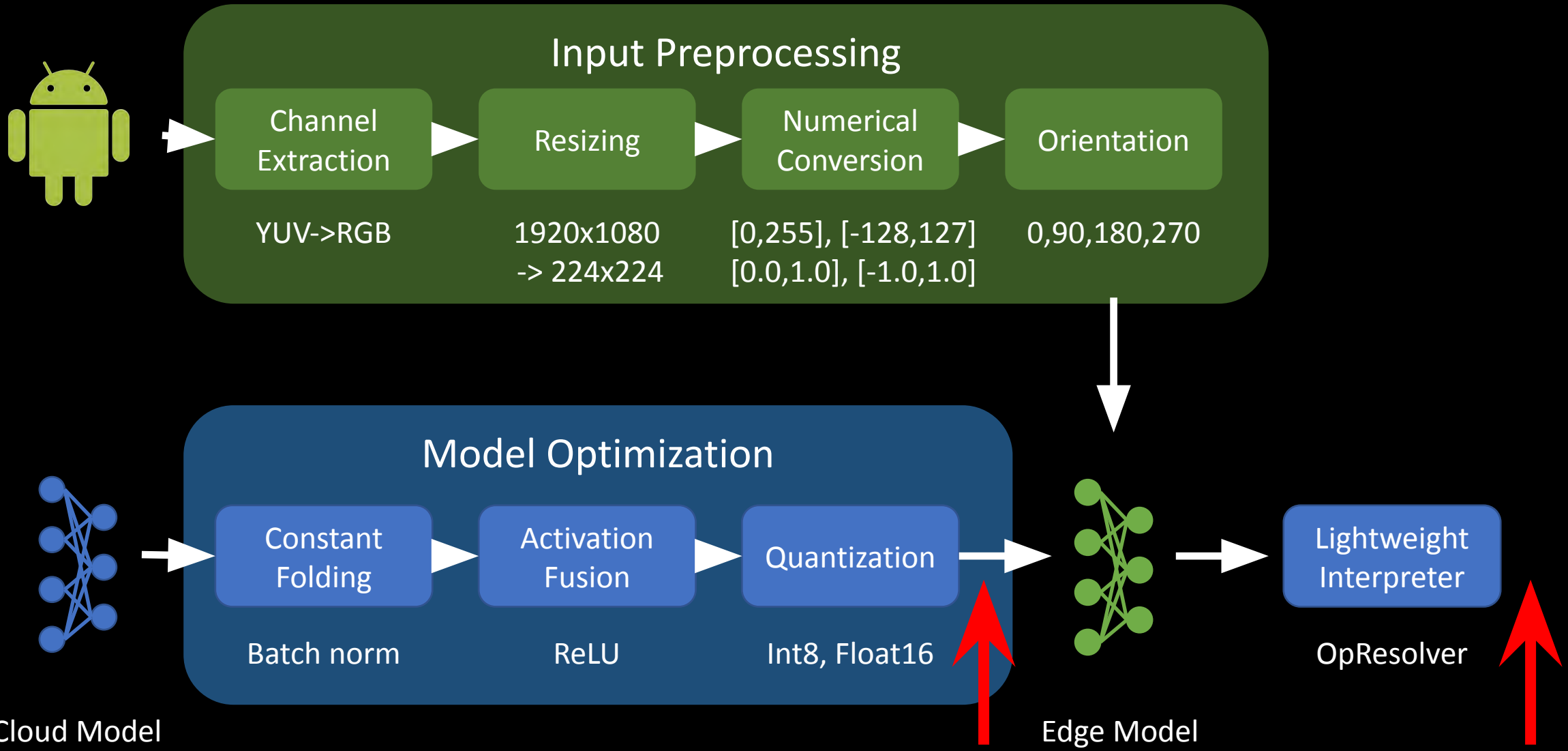
User-defined Reference Pipelines



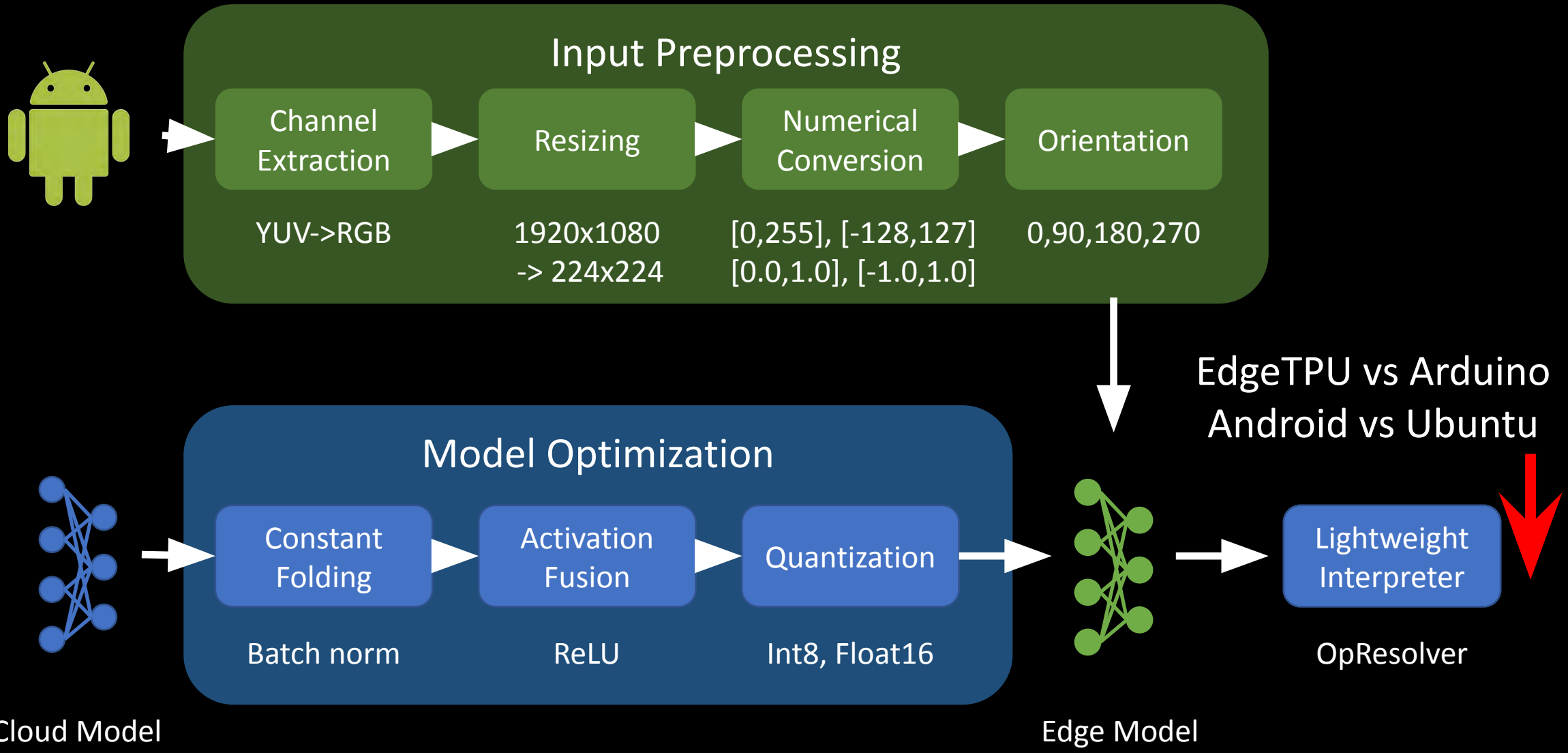
User-defined Reference Pipelines



User-defined Reference Pipelines



User-defined Reference Pipelines



Deployment Validation and Assertions

- *End-to-end Performance Benchmark*

Accuracy Validation

- *Per-layer Validation*

Per-layer Validation

- *Running Assertions for Root-cause*

Root-cause Analysis

Evaluation

- Wide applicability
- System overhead
- Pre-processing bugs and impact
- Quantization issues and impact
- Sub-optimal kernels and latency

Wide Applicability

Task	Models	Assertions					
		Channel	Resizing	Normalization	Rotation	Quantization	Latency
Image classification	Mobilenet v1, v2, v3, Inception v3, Densenet_121, Resnet_50 v2	✔	✔	✔	✔	✔	✔
Object detection	SSD_Mobilenet_v1, FasterRCNN	✔	✔	✔	✔	✔	✔
Image segmentation	Deeplab v3	✔	✔	✔	✔	✔	✔
Speech recognition	Plain CNN, Yamnet		✔	✔		✔	✔
Text classification	NNLM, mobilebert					✔	✔

ML-EXray catches a wide range of deployment issues across different tasks

System Overhead

Lightweight

- Latency: a few ms increase
- Memory: < 5 MB

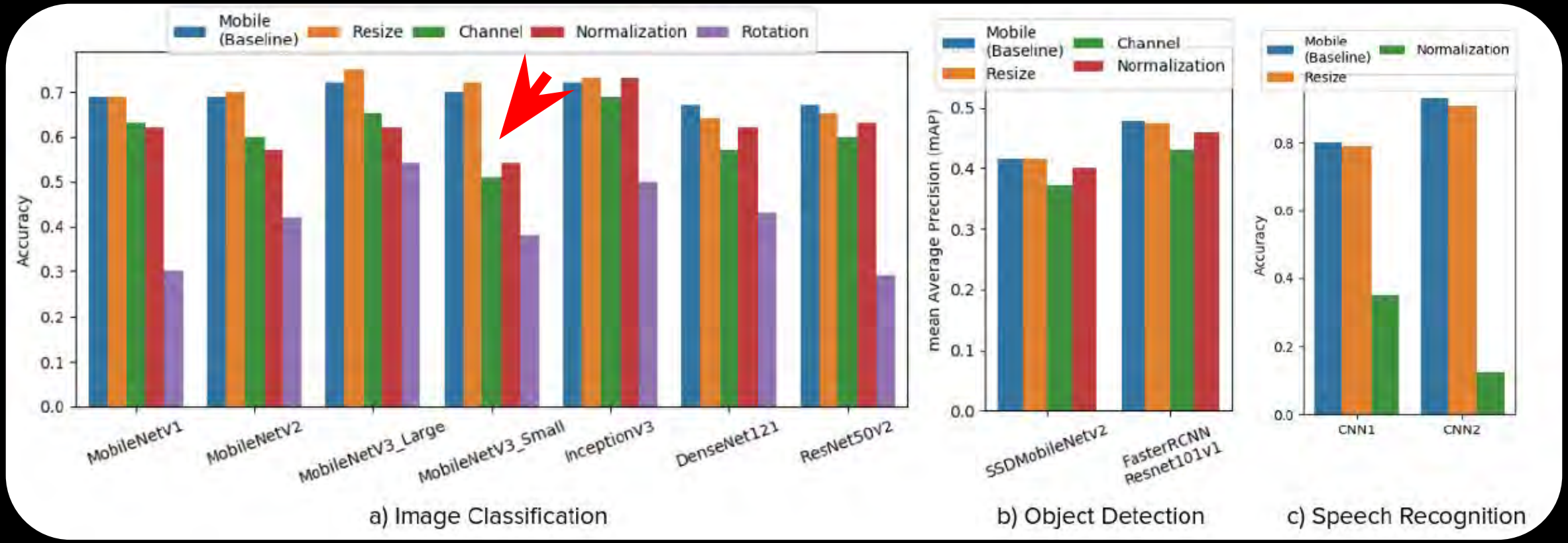
	Lat (ms) CPU only	Lat (ms) GPU enabled	Mem (MB)	Disk (KB/Frm)
Pixel 4	128.2±6.1	16.7±0.3	6.42	-
P4(Inst)	129.6±5.0	19.1±0.6	10.12	0.41
Pixel 3	157.0±6.7	28.4±0.4	9.26	-
P3(Inst)	158.3±7.3	30.0±0.5	12.37	0.41

Easy to use

- a few LoC for different validation target

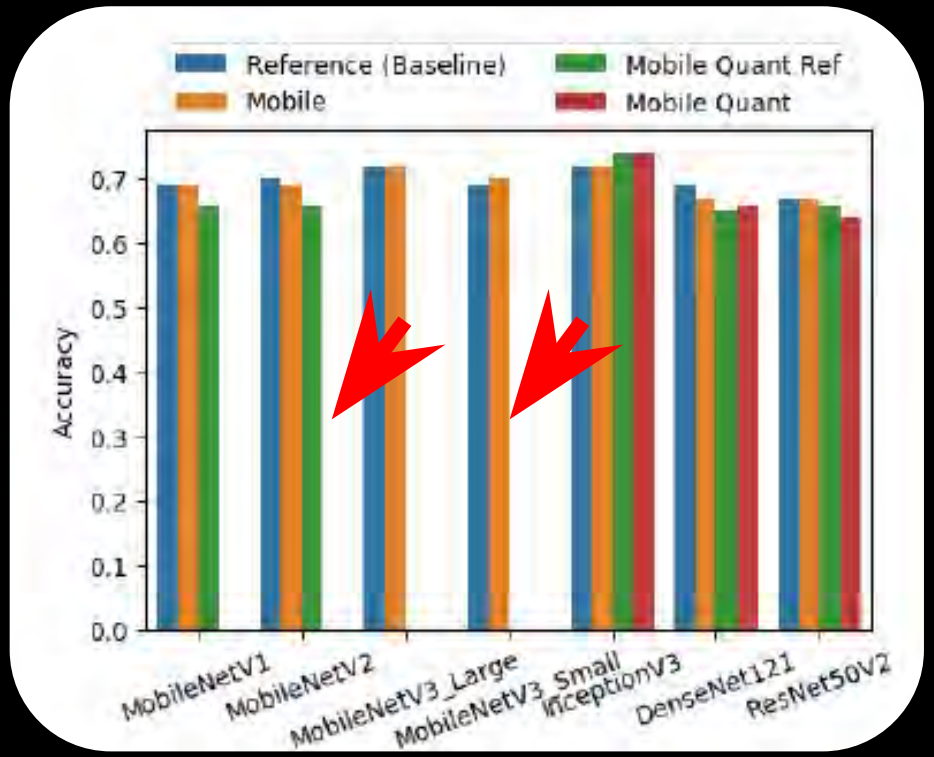
Debugging Target	Line of Code					
	W/ ML-EXRAY			W/O ML-EXRAY		
	Inst	Asrt	**Total**	Inst	Asrt	**Total**
Preprocessing	1	3	**4**	18	7	**25**
Quantization	4	9	**13**	82	183	**265**
Lat. & Mem.	4	4	**8**	14	8	**22**
Per-layer Lat.	2	6	**8**	14	90	**104**

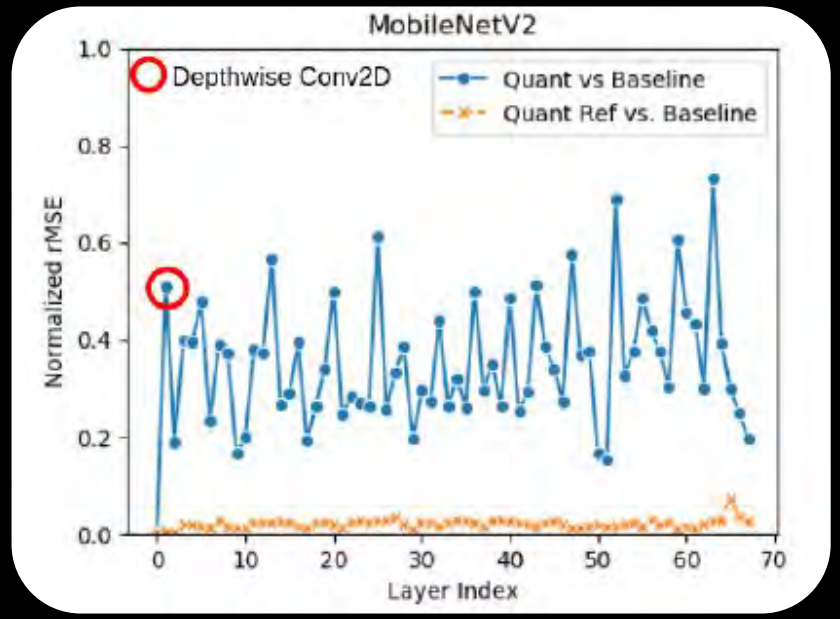
Pre-processing Bugs and Impact

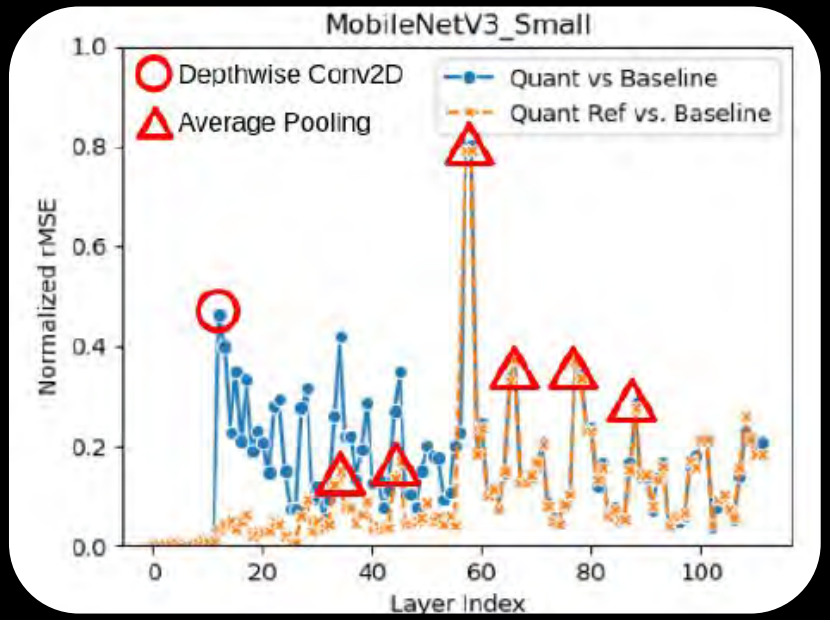


Eradicating these issues, ML-EXray can correct model performance by 5 - 40%

Localizing Quantization Issues







Quantized MobileNet not working well

Per-layer validation can easily localize the issue

Sub-optimal Kernels and Latency

Latency by layer type (MobileNet v2)

Layer Type (Count)	Mobile (ms)	Mobile Quant (ms)	Mobile Quant Ref (ms)	Emulator(x86) Mobile (ms)
D-Conv(17)	**95.4**	22.7	2885.2	120.0
Conv(35)	23.5	**32.3**	**18662.3**	**1409.8**
FC(1)	7.4	7.1	7.0	71.2
Mean(1)	6.1	5.6	5.0	2.5
Pad(4)	1.6	18.7	60.8	104.8
Add(10)	1.5	7.7	99.8	7.0
Softmax(1)	0.4	0.0	0.0	0.2
Quantize(1)	-	3.3	0.7	-
Total	136.26	97.816	21721.2	1715.7

Reference kernels may be more stable but much slower

Emulator is slow on convolution layers

Debug Your Edge ML

https://github.com/hangqiu/ML-EXray

https://arxiv.org/abs/2111.04779

MLSys | 2022

ML-EXray: Visibility into ML Execution on the Edge

Thank you!

ML-EXray: Visibility into ML Deployment on the Edge

Hang Qiu, Ioanna Vavelidou, Jian Li, Evgenya Pergament,
Pete Warden, Sandeep Chinchali, Zain Asgar, Sachin Katti

Stanford University

Google