aws machine learning

# TVM TensorRT Integration

## NVIDIA GTC 2021

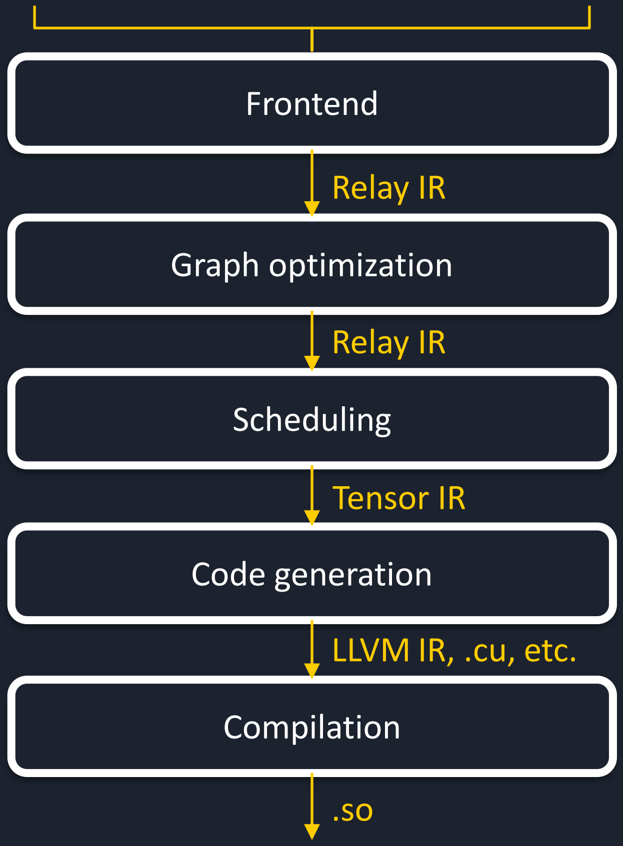
Trevor Morris – AWS SageMaker ML (Deep Learning Compilers)

# Problem

- Fast inference
- Optimize inference fast
- Using TVM alone has limitations
- Using TensorRT alone has limitations

+ 7 other frameworks

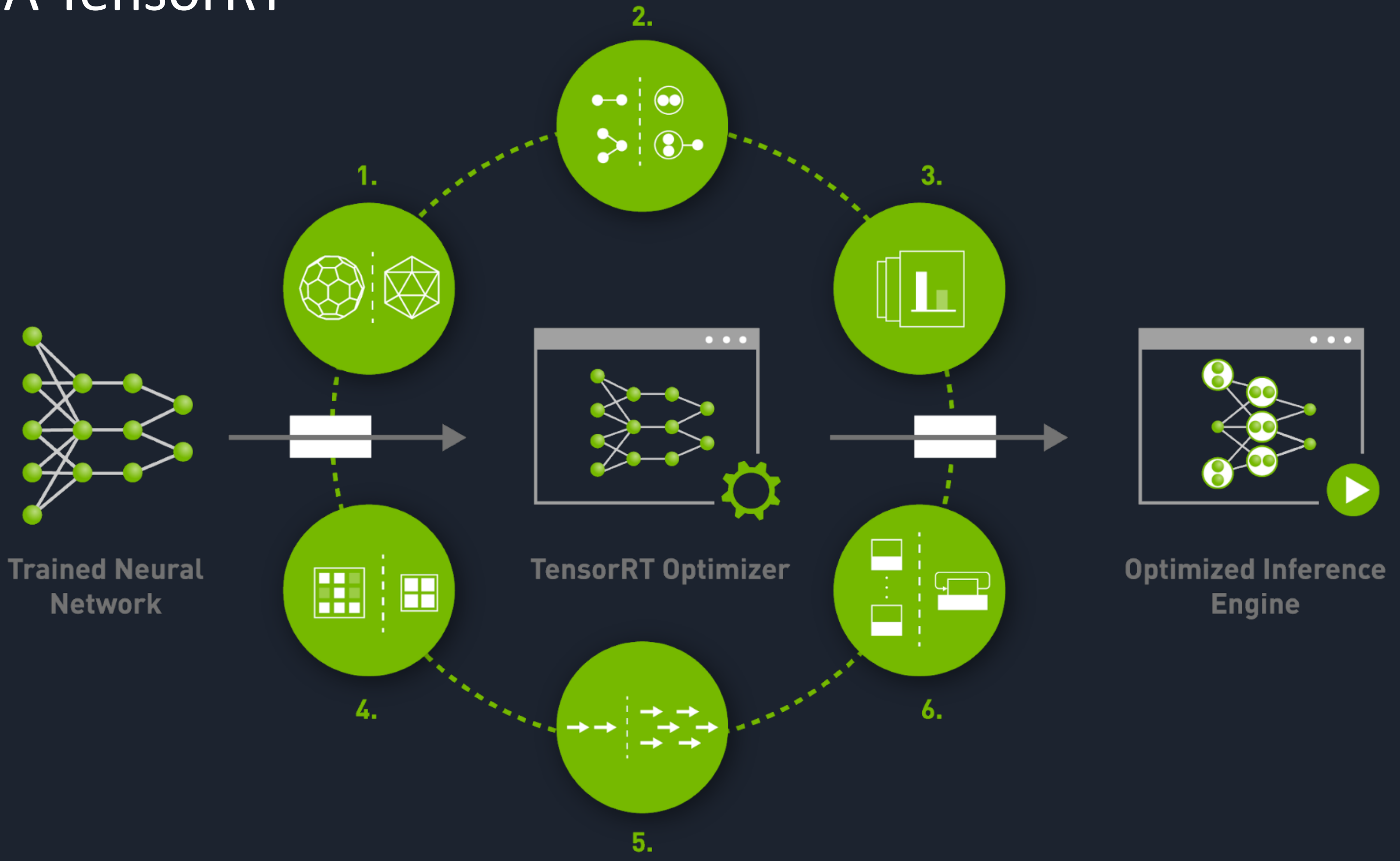



tvm

github.com/apache/tvm

aws machine learning

# NVIDIA TensorRT



# Differences

### TVM Advantages

- Kernels generated automatically by tuning
- Many more ops, frameworks
- Open source, easily extendable

### TensorRT Advantages

- Catalog of kernels handwritten by experts
- Tends to be faster for compute intensive ops
- Automatic lower precision

### TVM Disadvantages

- Tuning can take hours (T4), days (Jetson)
- Tuning must be done for each model

### TensorRT Disadvantages

- Limited operator support
- Difficult to import models
- Closed source

# TVM – TRT Integration

Combine to get best of both worlds

### Better than standalone TensorRT

- Greater model and framework coverage – most models are not fully supported by TRT
- Easier to use

### Better than standalone TVM

- Better performance for ops such as Convolution due to expert written kernels in TRT

### Better than framework-integrated TRT (TF-TRT)

- TRT performance is same, but TVM generated CUDA code is faster than framework

aws machine learning

# TVM “Bring Your Own Codegen”

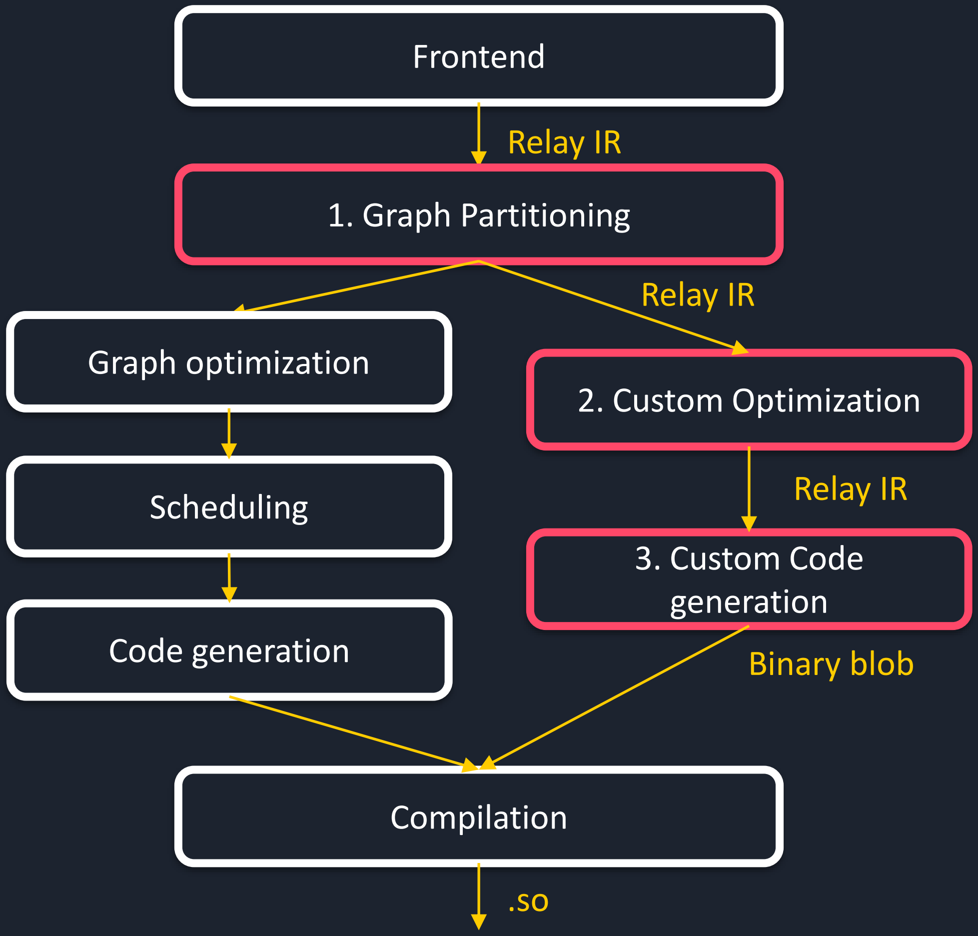
tvm.apache.org/docs/dev/relay_bring_your_own_codegen.html

- Interface for TVM developers to integrate proprietary accelerator libraries

- Must implement 4 things:
  1. Partitioning
  2. Optimizations
  3. Custom codegen
  4. Custom runtime

aws machine learning

# Standard Compilation vs BYOC


aws machine learning

 |

# TVM Relay IR

- Supports traditional computational dataflow graph (DAG)
- Supports let-binding, scopes, functions, control flow (Expression)

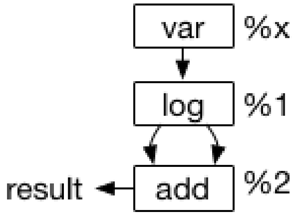
Python Code

```
x = relay.var(“x”)
v1 = relay.log(x)
v2 = relay.add(v1, v1)
f = relay.Function([x], v2)
```

Text Form

```
fn (%x) {
  %1 = log(%x)
  %2 = add(%1, %1)
  %2
}
```

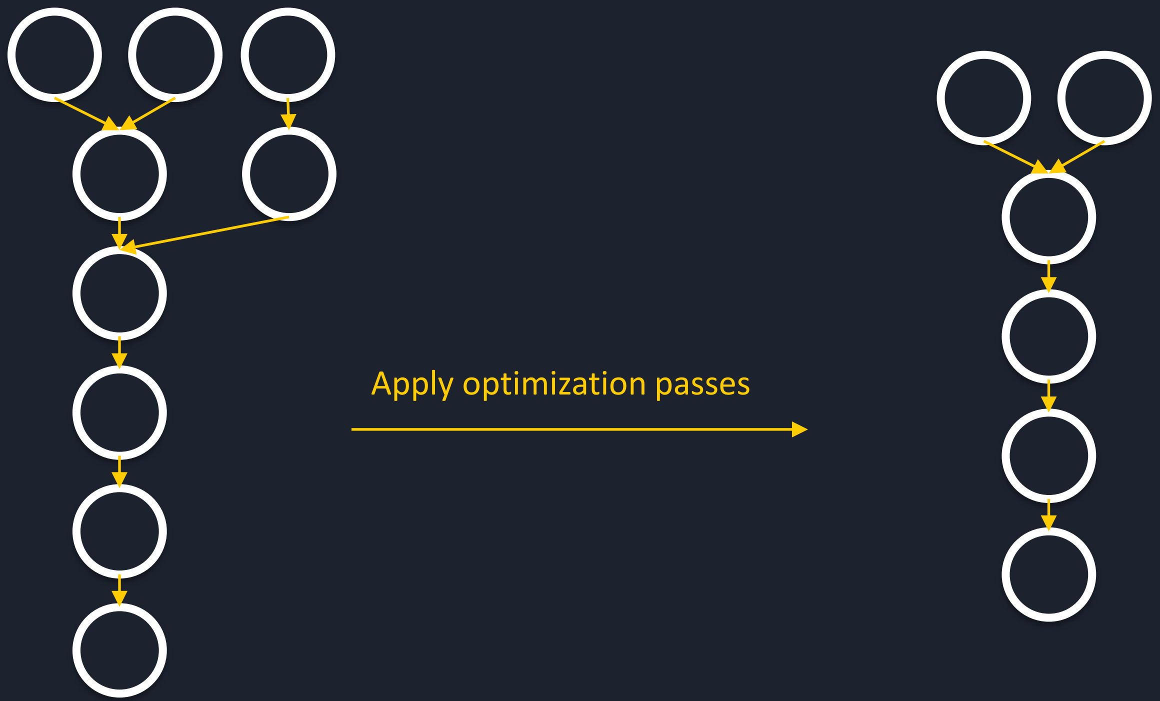
AST Structure





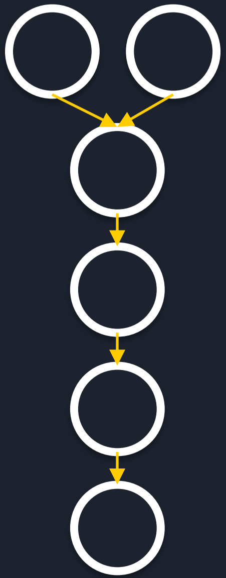
# Graph Optimization



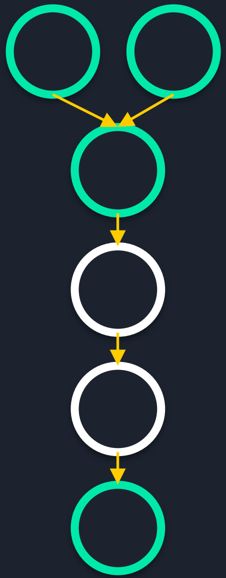

Relay Expression

Equivalent Relay Expression

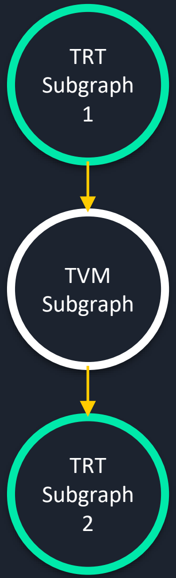
# Graph Partitioning



Relay Expression



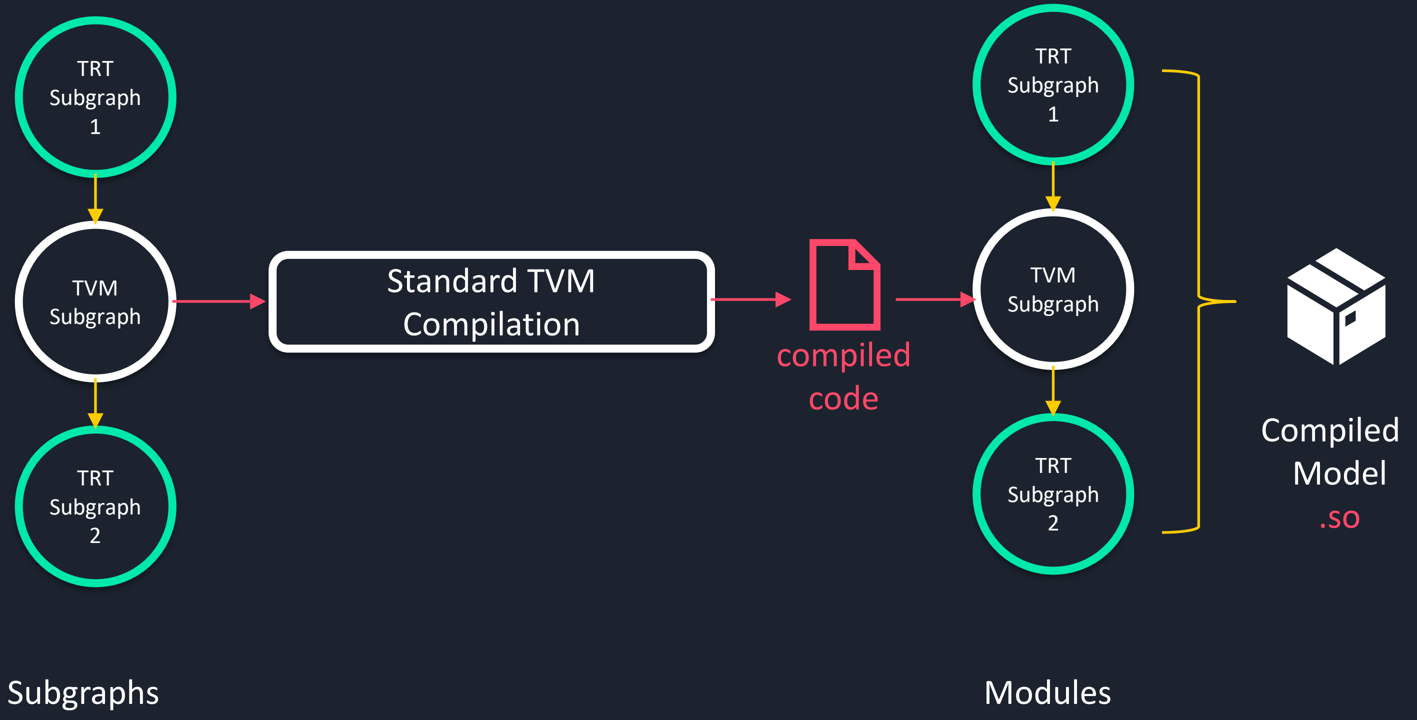
Annotation
Supported by TRT




Partitioning

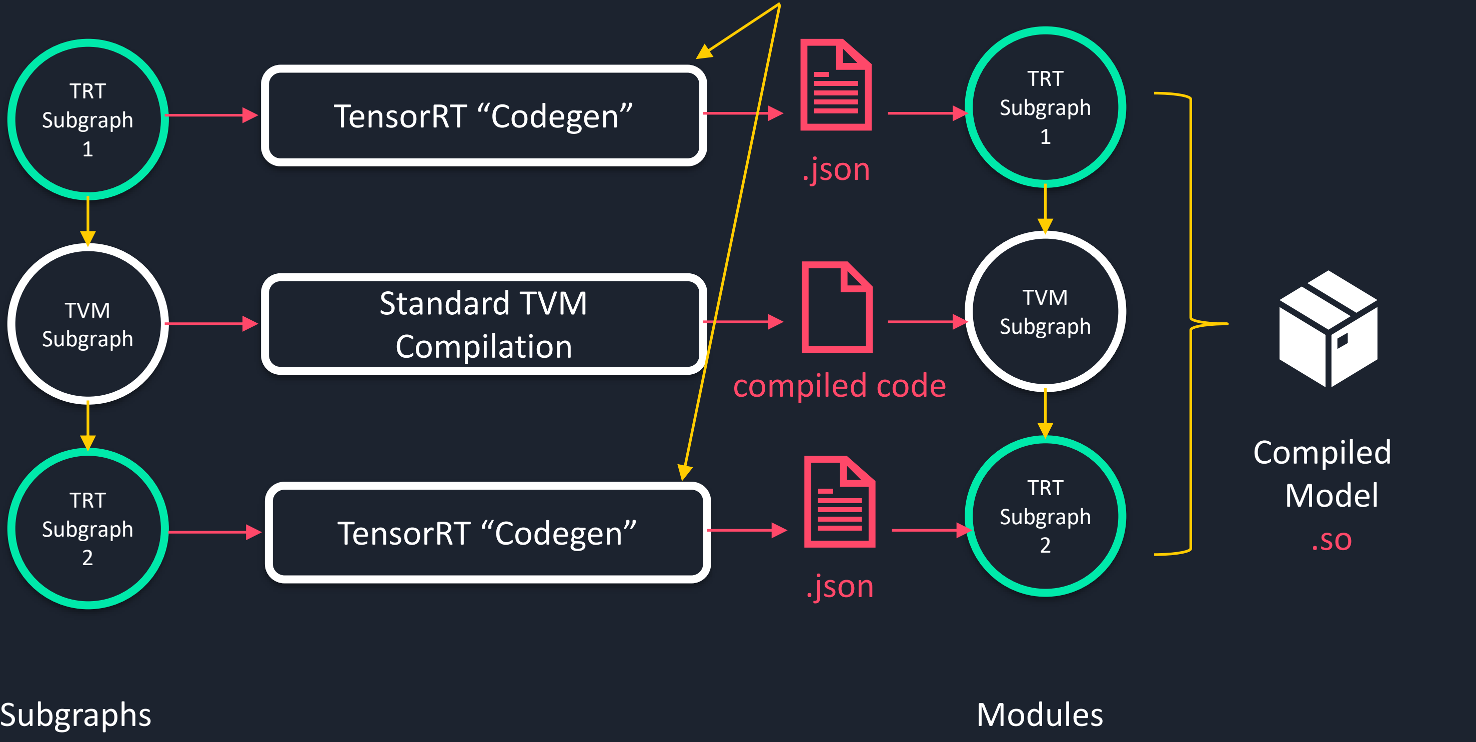
# Code Generation



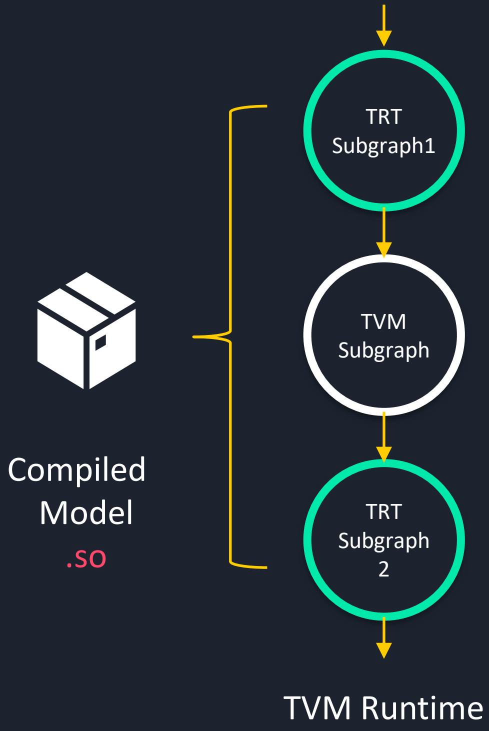

aws machine learning

# Code Generation

Only serialize Relay IR to JSON.
Will defer TensorRT usage to runtime - TRT is platform specific



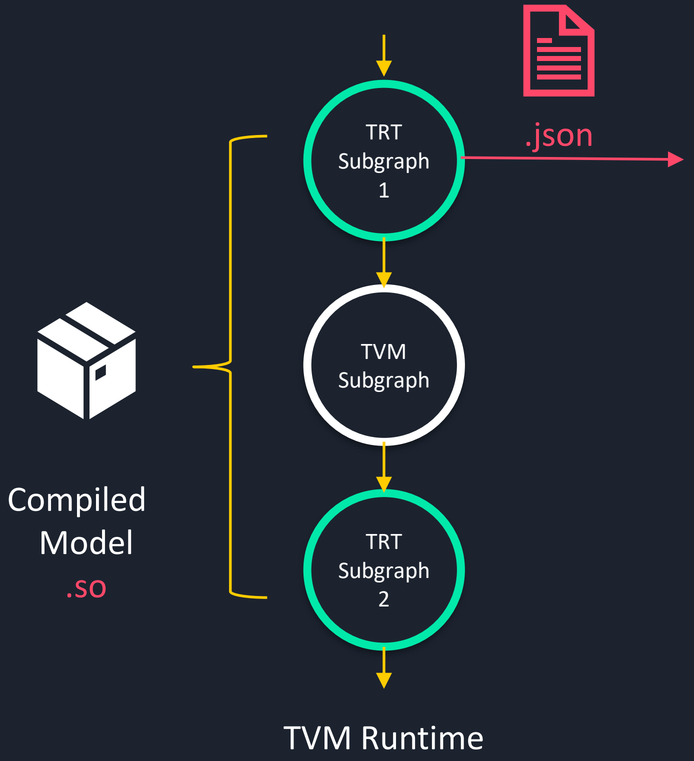
# Runtime


TRT
Subgraph1
TVM
Subgraph
TRT
Subgraph
2
Compiled
Model
.so
TVM Runtime



# Runtime



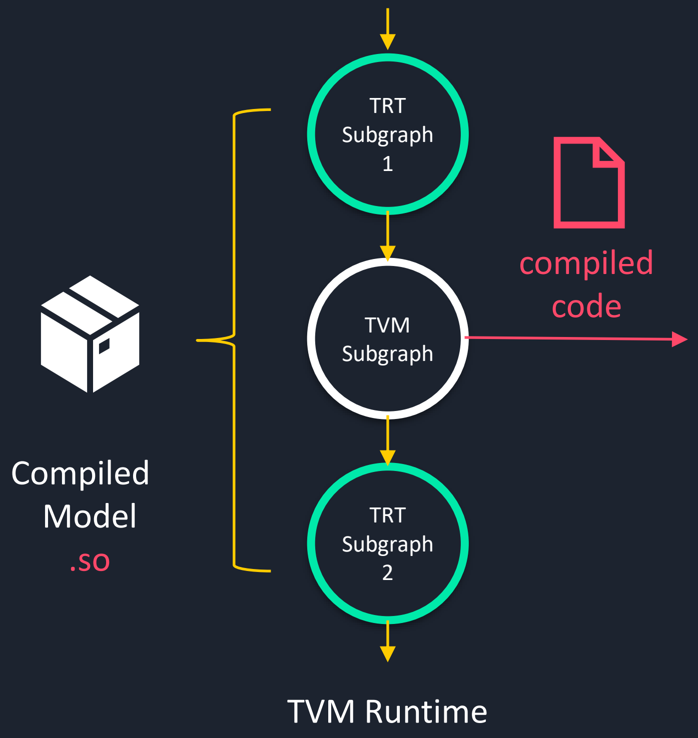

## TensorRT Runtime Module

Initialization
- Convert Relay json to TRT INetwork
- Build TRT engine and cache it

Execution
- Map TVM inputs and outputs to TRT bindings
- Invoke cached TRT engine

aws machine learning

# Runtime




TVM CUDA Runtime Module

Initialization
- Load compiled CUDA function from fatbin or ptx

Execution
- Launch CUDA kernel

# TVM

tvm.apache.org/docs/deploy/tensorrt.html

Install TVM dependencies

Install CUDA, CUDNN, TensorRT

Build TVM from source

```
import tvm
from tvm import relay
import mxnet
from mxnet.gluon.model_zoo.vision import get_model
block = get_model('resnet18_v1', pretrained=True)
input_shape = (1, 3, 224, 224)
mod, params = relay.frontend.from_mxnet(
    block, shape={'data': input_shape}, dtype="float32"
)
from tvm.relay.op.contrib.tensorrt import
     partition_for_tensorrt
mod, config = partition_for_tensorrt(mod, params)
with tvm.transform.PassContext(
    opt_level=3,
    config={'relay.ext.tensorrt.options': config}
):
    lib = relay.build(mod, target="cuda", params=params)
lib.export_library('compiled.so')
ctx = tvm.gpu(0)
lib = tvm.runtime.load_module('compiled.so')
runtime = tvm.contrib.graph_runtime.GraphModule(
    lib['default'](ctx)
)
input_data = np.random.uniform(0, 1, input_shape)
runtime.run(data=input_data)
output = runtime.get_output(0)
```

# AWS SageMaker Neo

aws.amazon.com/sagemaker/neo/

```
sm.create_compilation_job(
  CompilationJobName=compilation_job_name,
  RoleArn=role_arn ,
  InputConfig={
    'S3Uri': 's3://bucket/model',
    'DataInputConfig': data_shape,
    'Framework': 'MXNET'
  },
  OutputConfig={
    'S3OutputLocation': 's3://bucket/',
    'TargetDevice': 'jetson_xavier'
  }
)
```

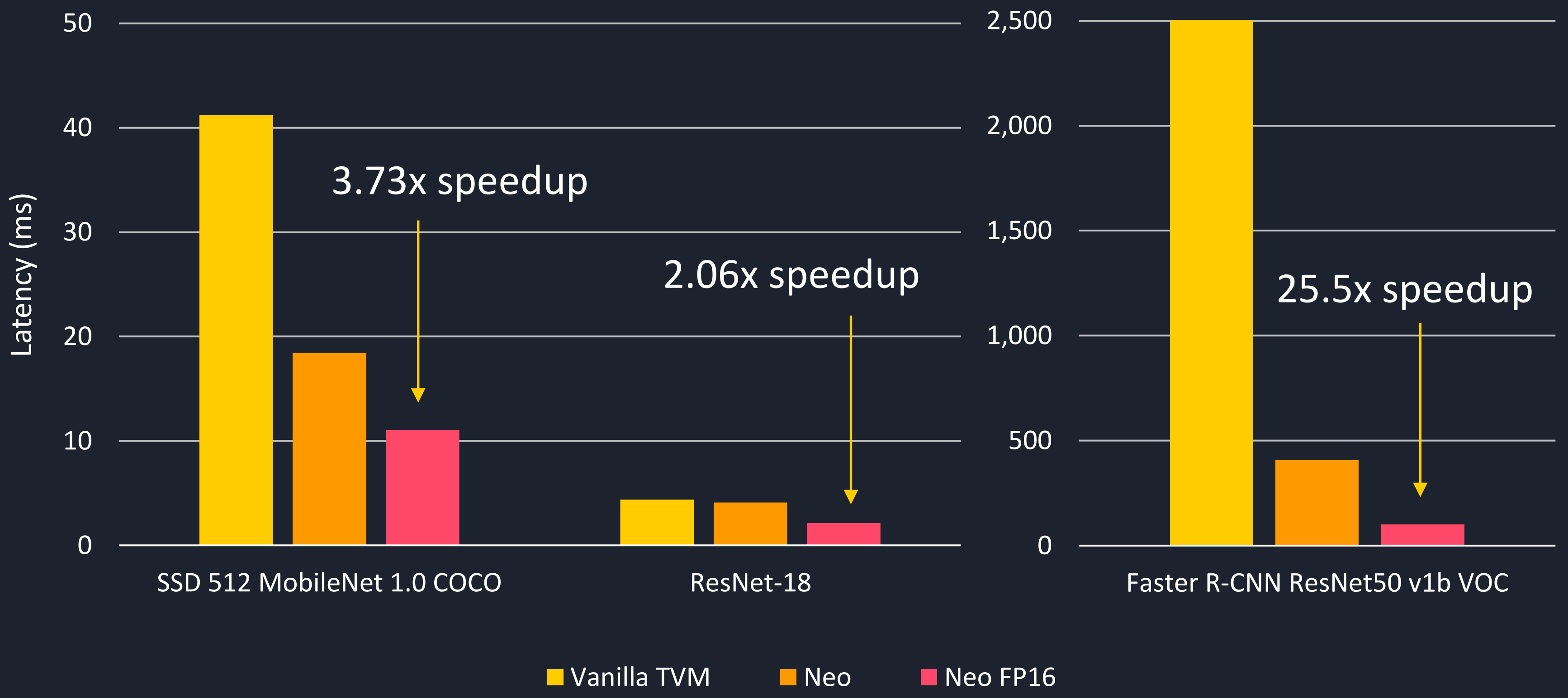
```
pip install dlr
import dlr

model = dlr.DLRModel('path/to/model/', 'gpu')
y = model.run(x)
```

aws machine learning

# MXNet GluonCV

## Jetson AGX Xavier (JetPack 4.4)

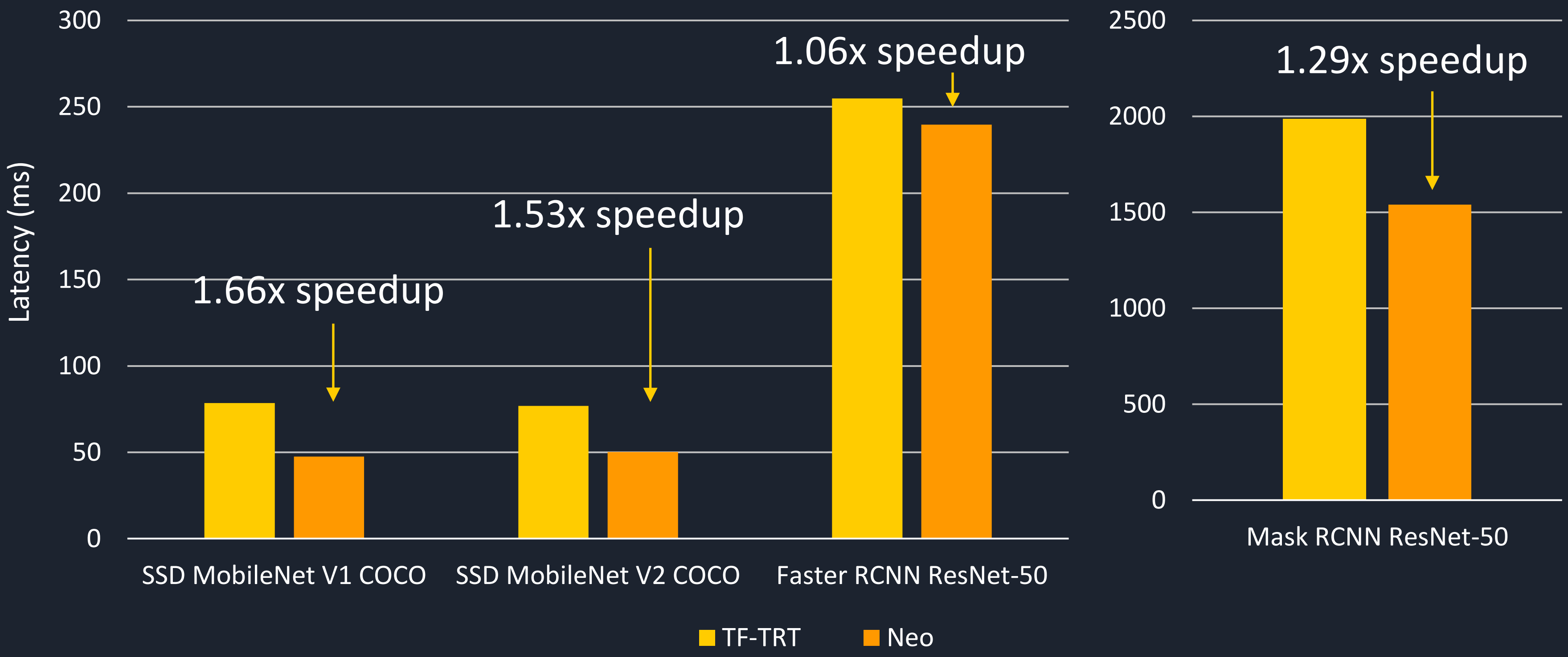



aws machine learning

 | 

# TensorFlow Object Detection

## Jetson AGX Xavier (JetPack 4.4)



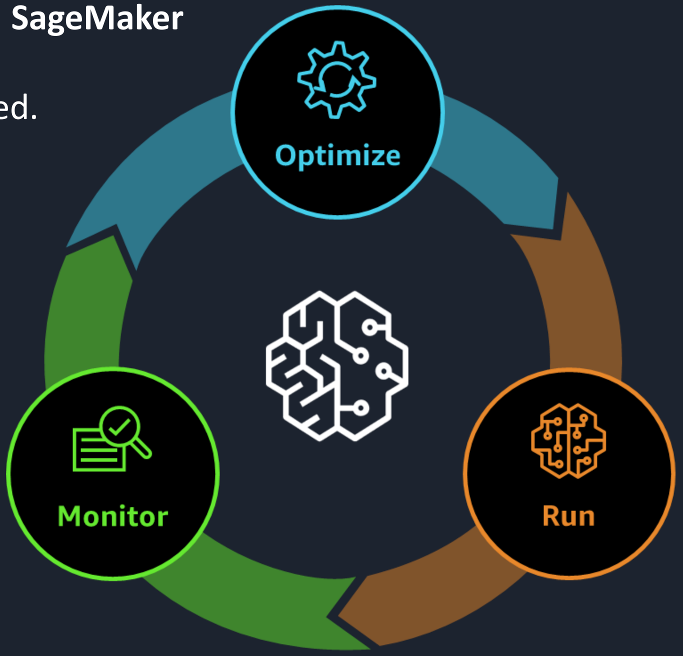

aws machine learning

 | 

# SageMaker Edge Manager

## Optimize, run, monitor and maintain ML models on fleets of devices

Optimize your models with **SageMaker Neo** with TRT built-in.
No TVM knowledge required.

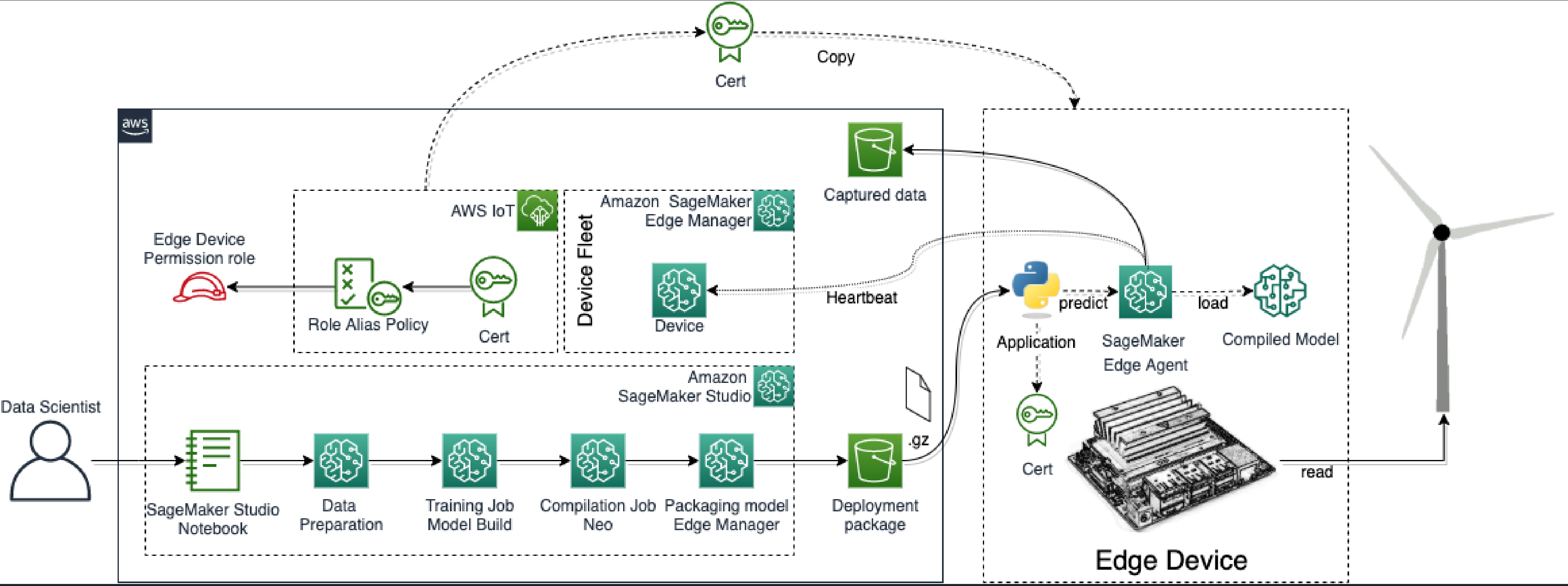



Detect model decay and improve model quality.

Run one or more models on each device in a fleet.

https://aws.amazon.com/sagemaker/edge-manager/

# SageMaker Edge Manager




aws machine learning

# Summary

- Problem
  - Fast Inference
- Solution
  - TVM
  - TensorRT
  - Combining TVM and TensorRT
- Results
- How to use
  - With TVM
  - With SageMaker Neo
  - With SageMaker Edge Manager

