On-Device Training with ONNX Runtime

Kshama Pawar and Baiju Meswani

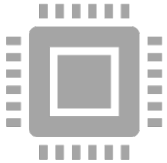
Microsoft Corporation

Outline

- Motivation
- Scenarios
- Tech Deep Dive
- Questions

Motivation

Data created through application interaction



Resource constrained devices

Privacy for personal devices is important

Portable solutions across frameworks and devices needed

ONNX Runtime provides an efficient, framework agnostic, local trainer, that trains with device data on the edge

Key Benefits

Extends the ORT Inference solution

Memory and performance efficiency for lower resource consumption on device

Simple APIs and multiple language bindings make it easy to scale across multiple platform targets

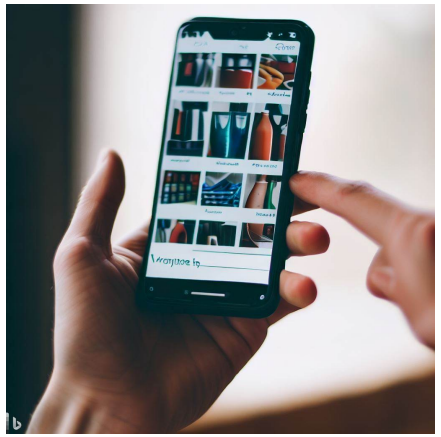
Support privacy while leveraging customer data

Same solution runs on cloud, desktop, edge, mobile

Scenarios

Personalization – Fine-tune on the device

- Image/Audio classification, Text Prediction



Federated Learning – Train a global model

- Medical Research, Autonomous vehicles, Robotics

Tech Deep Dive

Now and Upcoming

Now

- ORT 1.15 launches On-Device Training!
- Platform Support - Windows, Linux, Android
- Language Bindings – C, C++, C#, Python, Java
- Integrates with Microsoft’s FL SDK (federated learning SDK)
- Minimum build for smaller binaries – base build size 1.5 MB

Upcoming

- Support for iOS and ORT-Web (extending existing mobile and web offerings)
- Language bindings – JS, Objective-C, Swift
- Quantized model support
- More optimizations

Get Started

- Blogs: On-Device Training: Efficient training on the edge with ONNX Runtime - Microsoft Open Source Blog
- Tutorial: On-Device Training | onnxruntime
- Docs: On-Device Training | onnxruntime
- C++ example: https://github.com/microsoft/onnxruntime/tree/main/orttraining/orttraining/test/training_api/trainer
- Python and Android Demo: https://github.com/microsoft/onnxruntime-training-examples/tree/master/on_device_training

Resources

- Offline Prep Tools: https://github.com/microsoft/onnxruntime/tree/main/orttraining/orttraining/python/training/onnxblock
- Python Bindings: https://github.com/microsoft/onnxruntime/tree/main/orttraining/orttraining/python/training/api
- C++ Bindings: onnxruntime/onnxruntime_training_cxx_api.h at main · microsoft/onnxruntime (github.com)
- C# Bindings: https://github.com/microsoft/onnxruntime/tree/main/csharp/src/Microsoft.ML.OnnxRuntime/Training
- C API: onnxruntime/onnxruntime_training_c_api.h at main · microsoft/onnxruntime (github.com)

Questions?

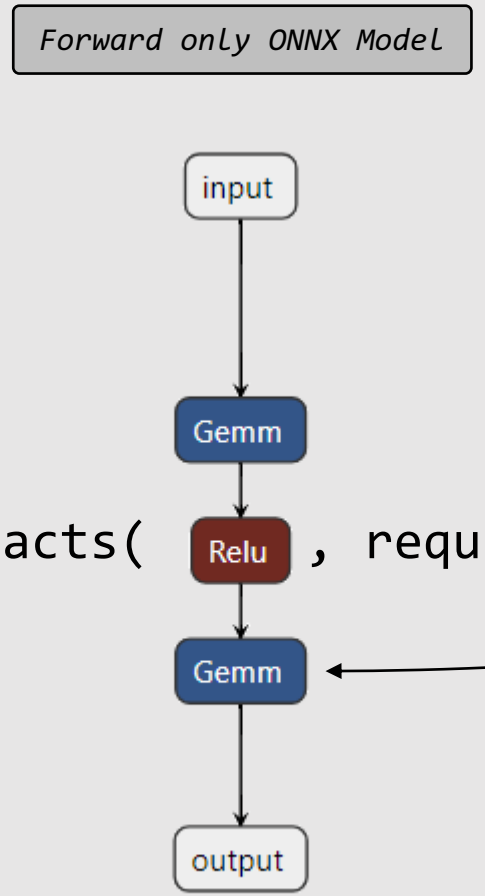
Offline Phase – Prepare the training artifacts in an offline step:

- The training ONNX model
- The evaluation ONNX model
- The optimizer ONNX model
- The checkpoint file

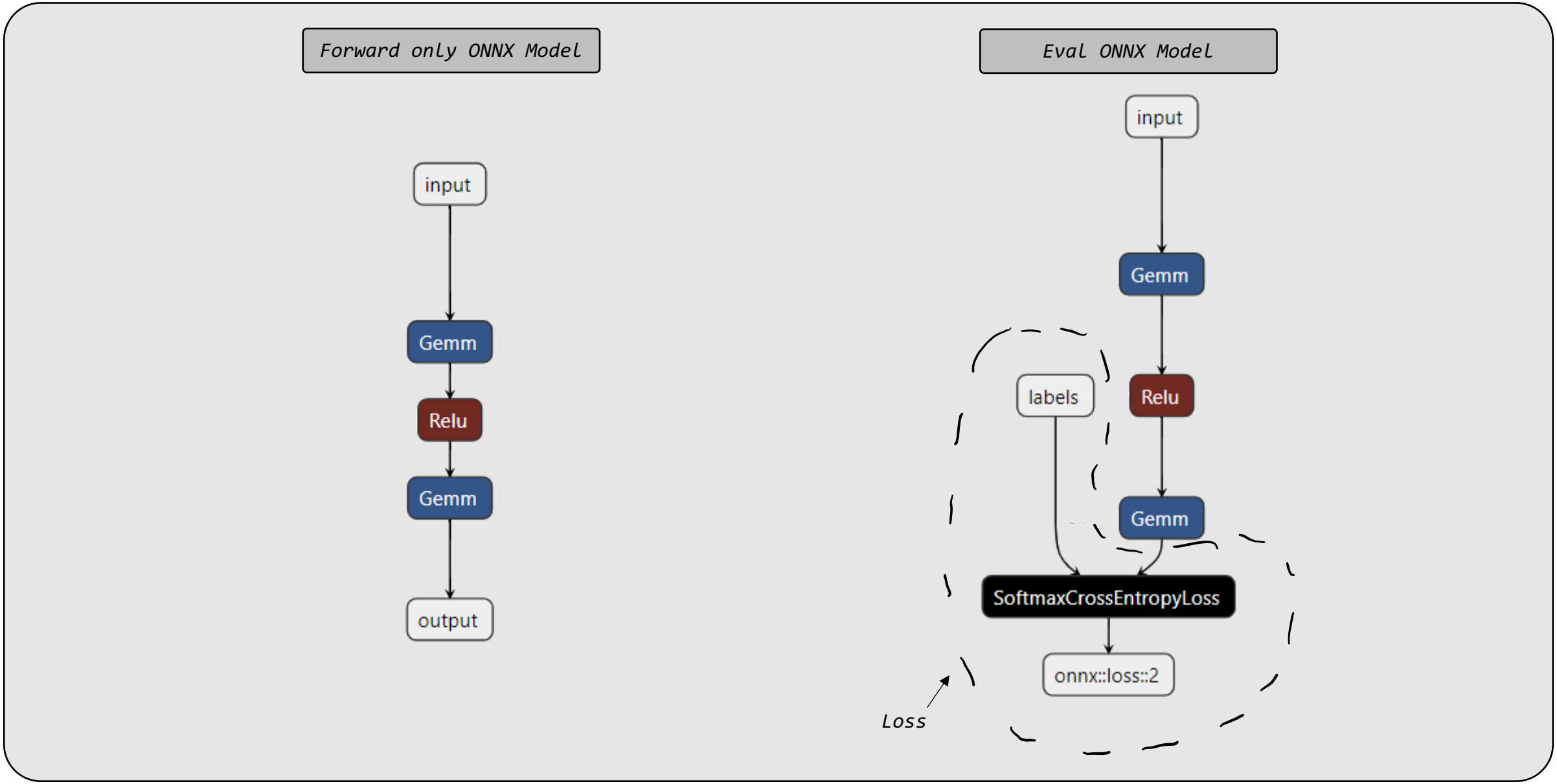
Training Phase – Interface with the ONNX Runtime training API on the device.

On-Device Training | Offline Phase – Preparing for Training

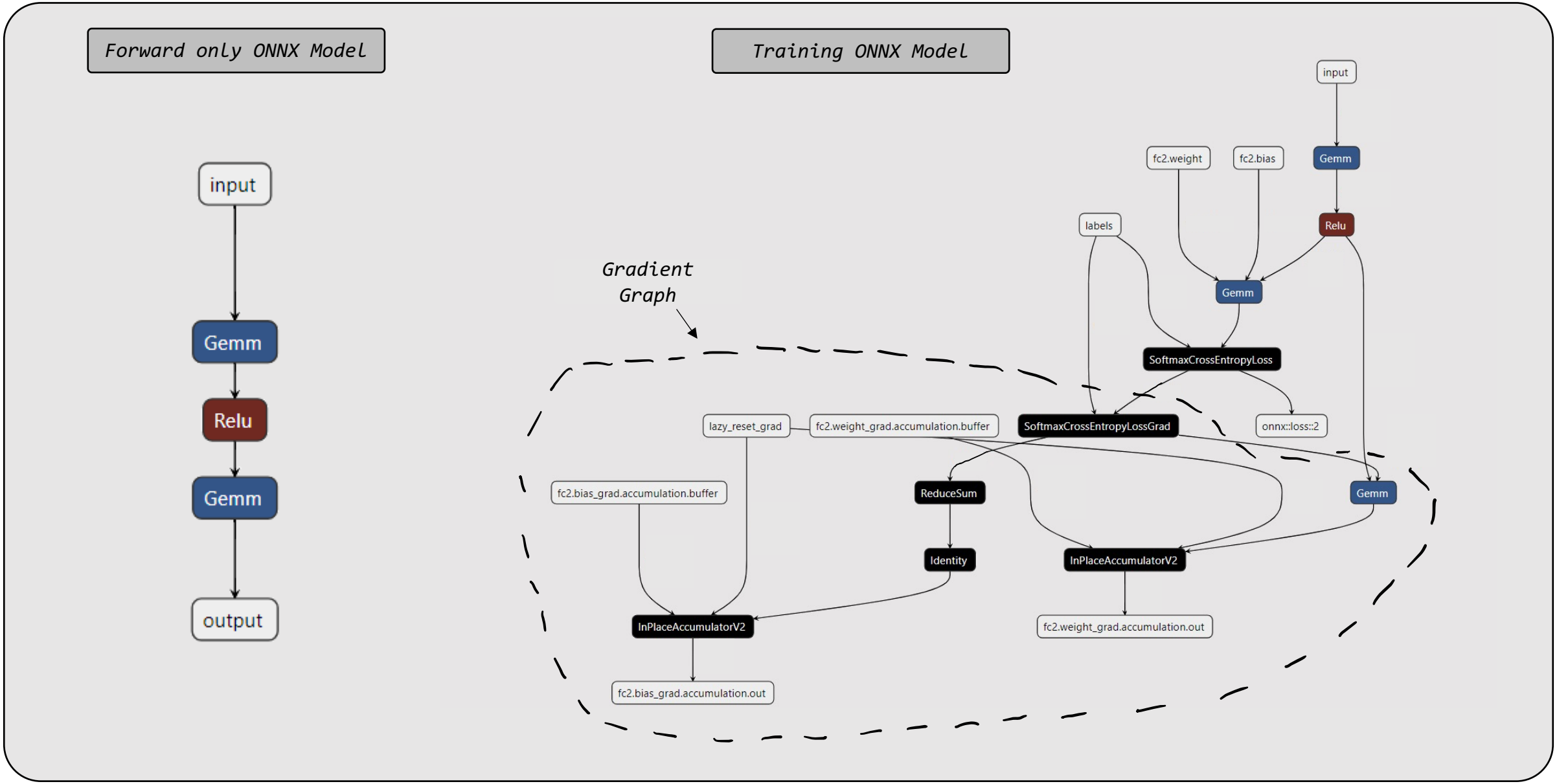
```
onnxruntime.training.artifacts.generate_artifacts(                , requires_grad = )
```



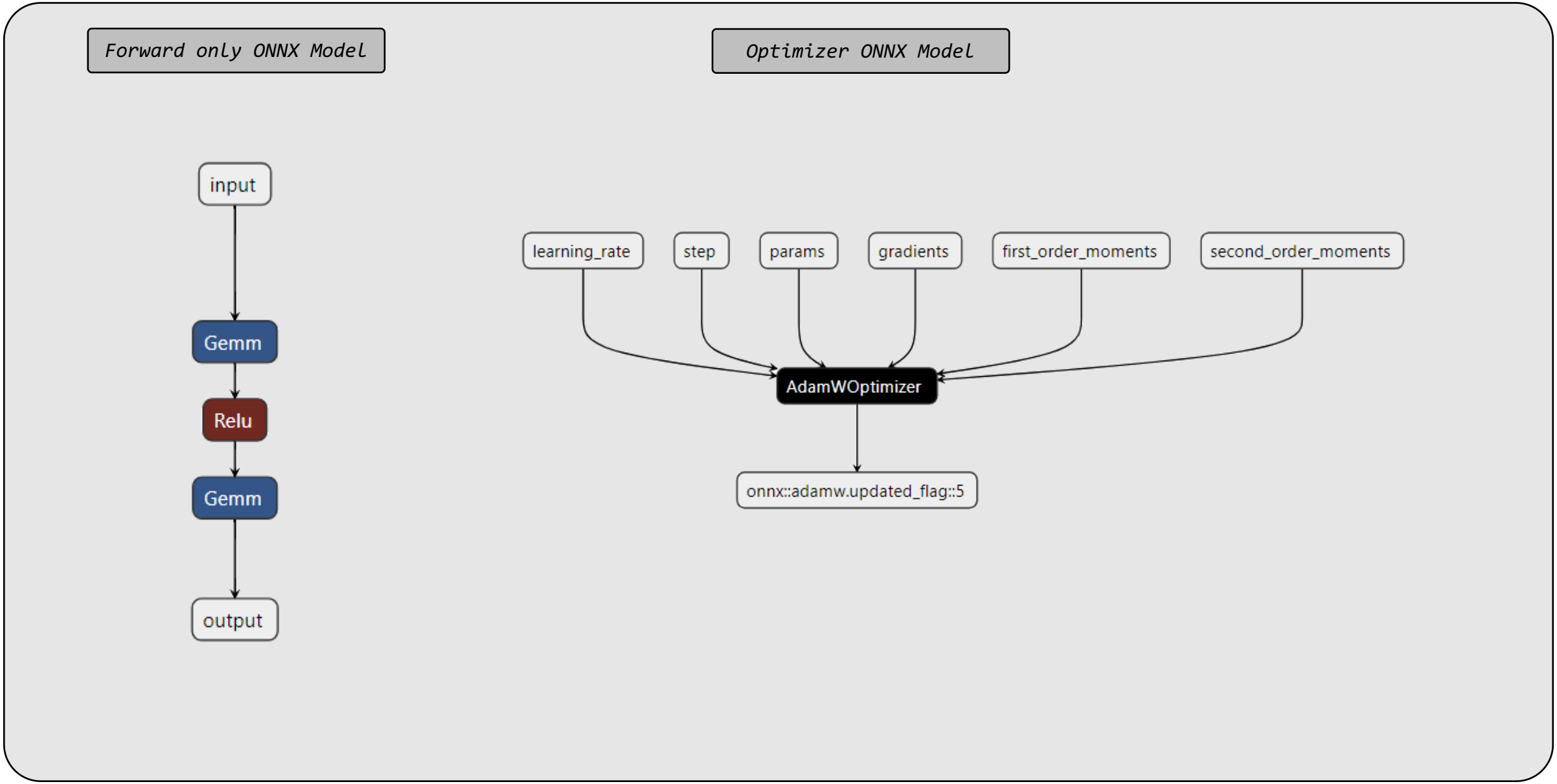
On-Device Training | Offline Phase – Preparing the Eval ONNX Model



On-Device Training | Offline Phase – Preparing the Training ONNX Model



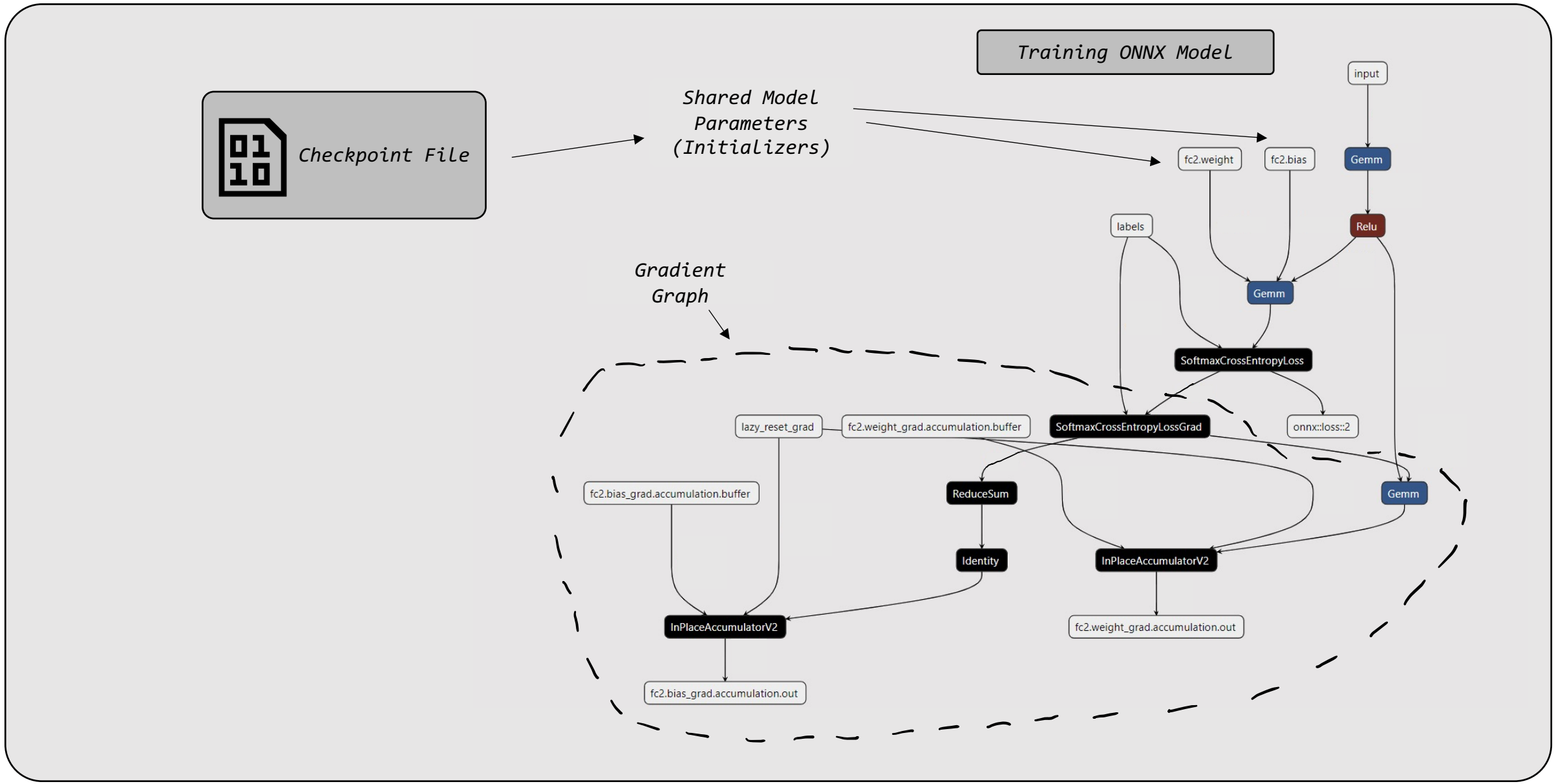
On-Device Training | Offline Phase – Preparing the Optimizer ONNX Model



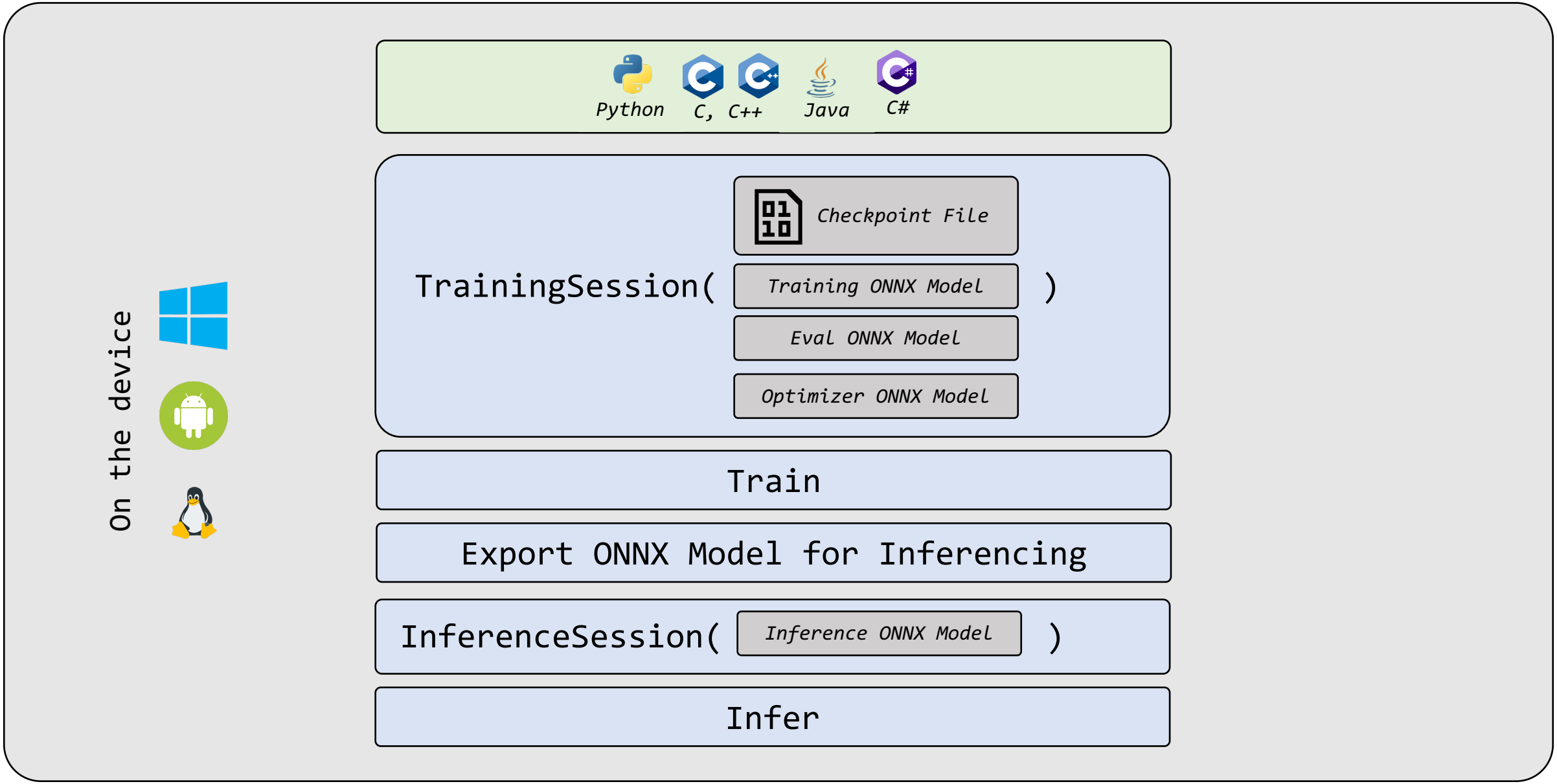
On-Device Training | Resources

- Documentation
- High Level Blog
- Android Demo Tutorial
- C, C++ API
- Python API
- Java API
- C# API
- Source Code GitHub Repository
- Examples GitHub Repository
- Reach Out to the Team

On-Device Training | Offline Phase – Preparing the Training ONNX Model



On-Device Training | Training Phase – Learning on the Device



Thank You!