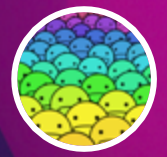
AI编译器-系列之PyTorch

AOT AutoGrad



ZOMI

Talk Overview

1. **PyTroch 2.0 新特性**
   - 2.0 新特性回顾
   - PyTorch 2.0 安装与新特性使用
   - PyTorch 2.0 对厂商的启发和思考
2. **TorchDynamo 解读**
   - TorchDynamo 特性
   - TorchDynamo 实现方案
3. **AOTAutograd 解读**
   - AOTAutograd 效果
   - AOTAutograd 实现方案
4. **TorchInductor 新特性**
   - Triton 使用解读
   - Triton 深度剖析

Talk Overview

1. AOTAutograd 解读

- AOTAutograd 实现方案
- AOTAutograd 效果
- Torch dispatch 机制和原理

TorchDynamo

- TorchDynamo hooks into the frame evaluation API in CPython to dynamically modify Python bytecode right before it is executed.
- It rewrites Python bytecode in order to extract sequences of PyTorch operations into an FX Graph which is then just-in-time compiled with an ensemble of different backends and autotuning.

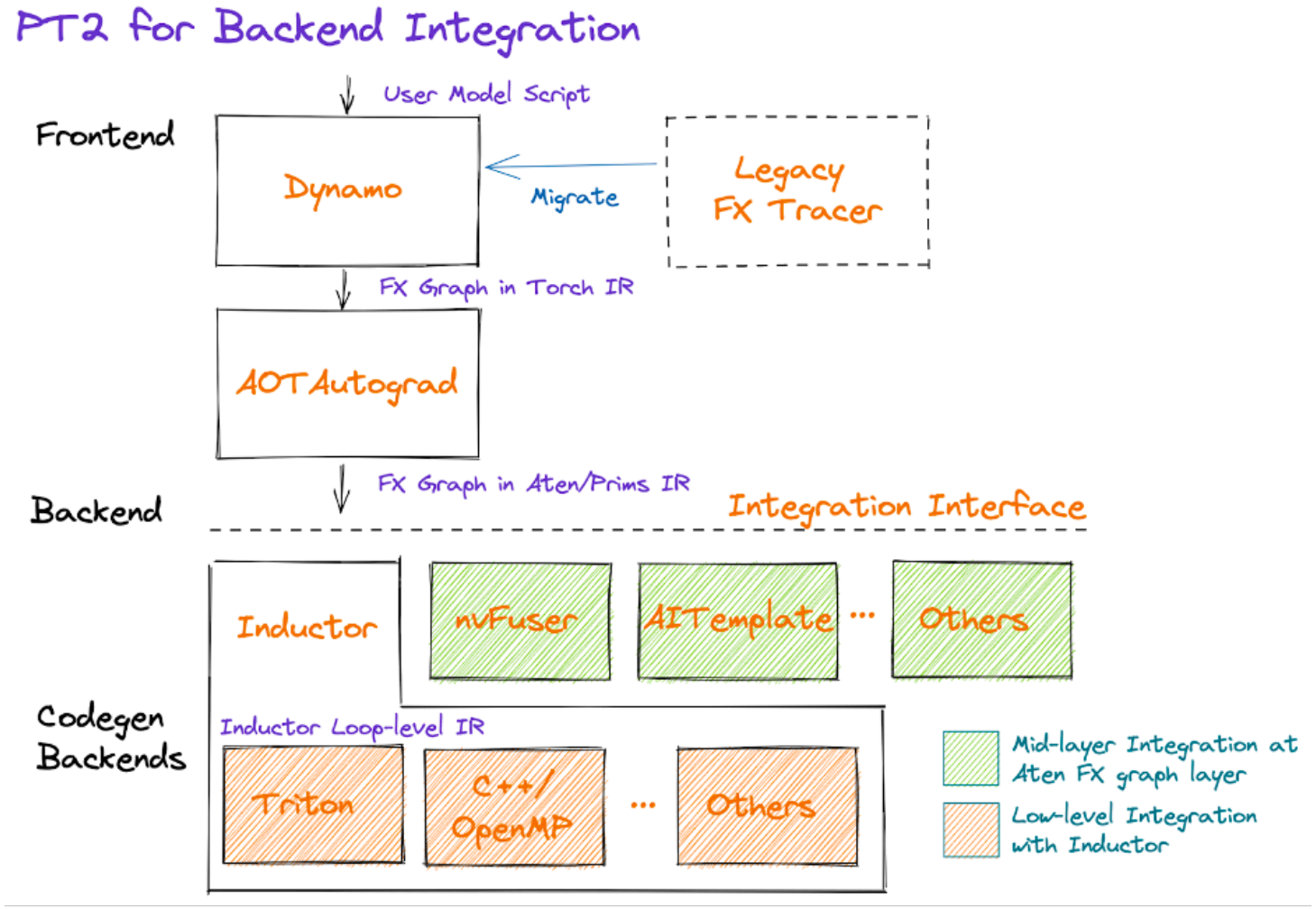
Adding Training in TorchDynamo

- Training adds challenges because the PyTorch Automatic Differentiation engine sits below the PyTorch dispatcher in C++. Therefore, the operators running in the backward pass are not directly visible to TorchDynamo at the Python level.

Adding Training in TorchDynamo

- To support training with TorchDynamo, we need to run/optimize operations that happen in the .backward() pass. Supporting backwards can be done in a few different ways:
- **Eagerly**: backends could use the dynamic autograd tape on every call, the same as eager mode.
- **TorchScript**: Difficult to maintain and does not support all operations.
- **AOTAutograd**: Records the behavior of the eager dispatcher-based autograd once at compile time. This allows it to capture everything with a simpler and more robust implementation that reuses much more of eager mode autograd. In addition, it allows us to easily make optimization decisions with visibility of both the forwards and backwards graphs.

PyTorch Compiler Mode



Speed Up During Training

- Integrate again TorchScript.
  - a. How do you access the backwards pass?
- Using Symbolic Script
  - a. That Does not have full coverage. cannot capture other transforms like vmap.
- Using LazyTensor
  - a. Not a particularly hackable API.
- Write your own auto diff.
  - a. A lot of work…

AOTAutograd

1. Use whatever compiler you want for training.
2. Use the resulting function however you want in you PyTorch training code.
3. All in Python.

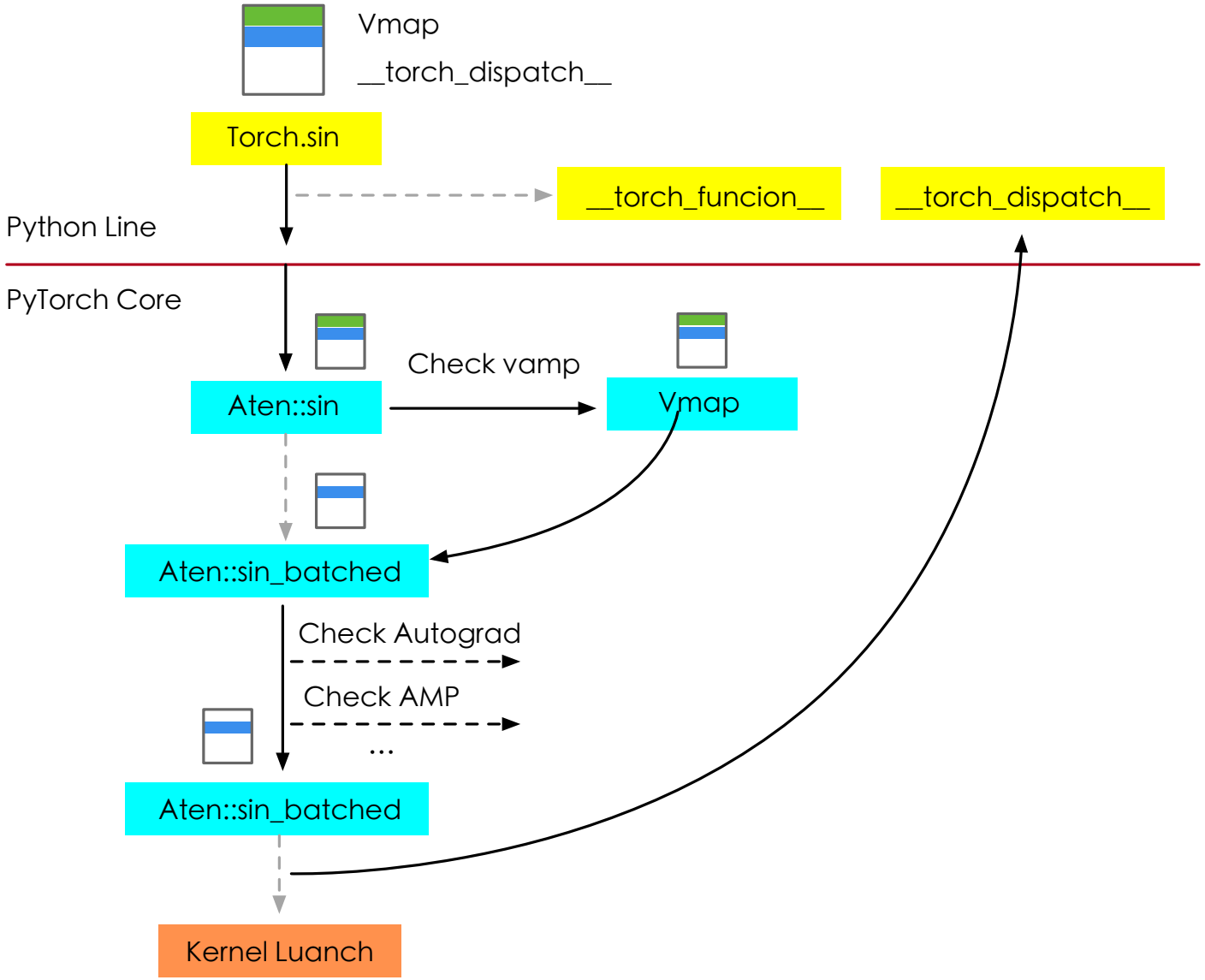
AOTAutograd

- AOTAutograd relies on the recently introduced **torch_dispatch** based tracing mechanism to capture the backward graph ahead of time. Therefore, it reuses the PyTorch core Autograd engine to generate the backward graph.

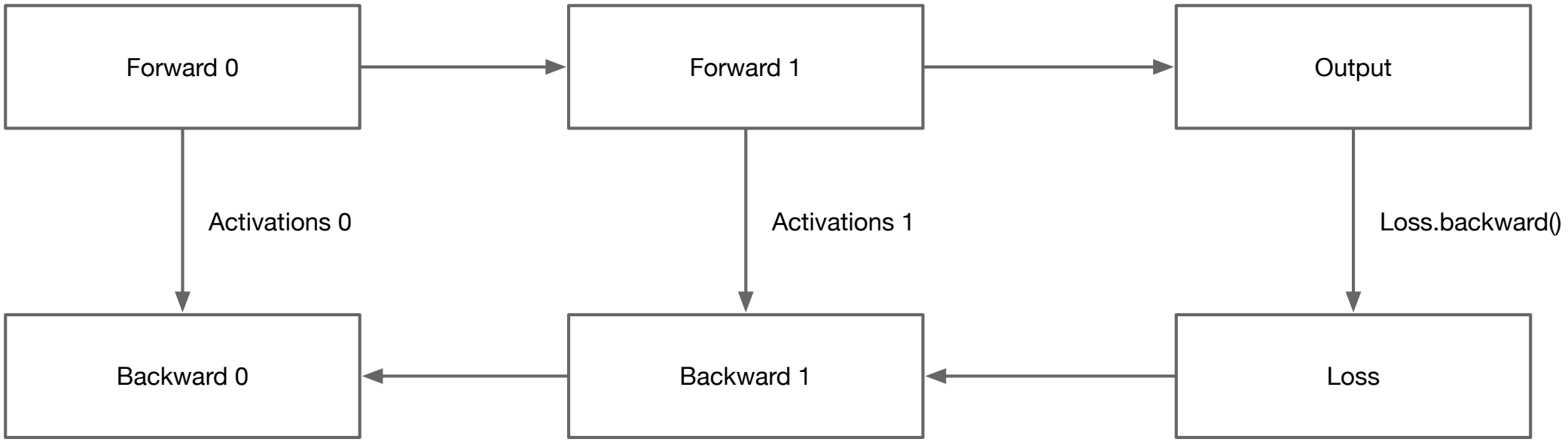
AOTAutograd

1. Jointly trace the forwards and backwards using __torch_dispatch__
2. Partition this joint fw + bw graph into 2 graphs: the forward graph and the backwards graph.
3. Wrap up the new fw + bw graphs into an autograd. Function that can be used in eager mode.

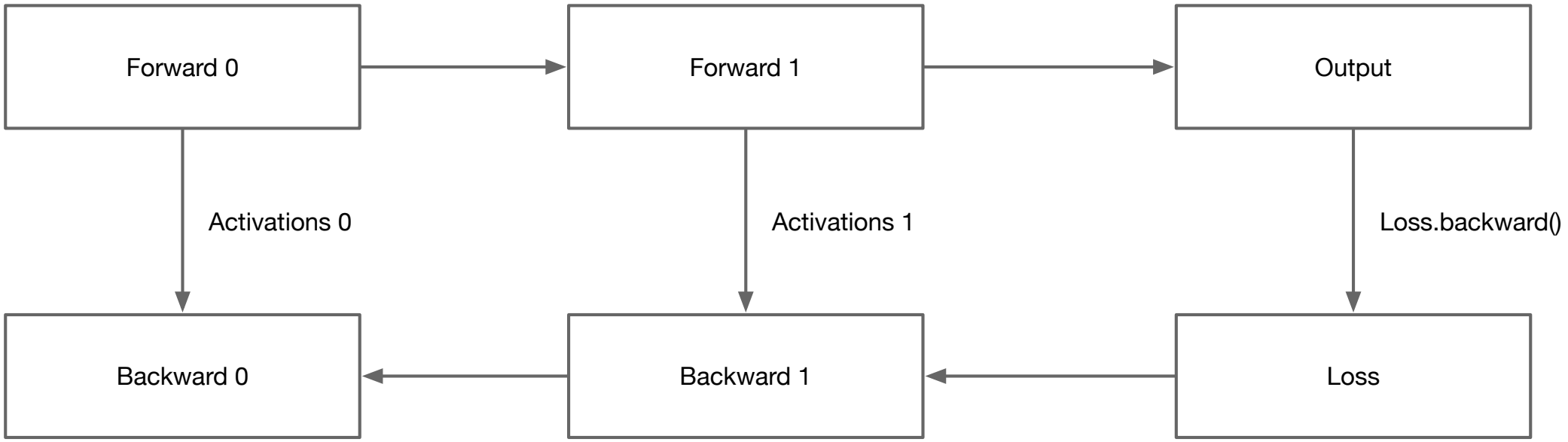
__torch_dispatch__



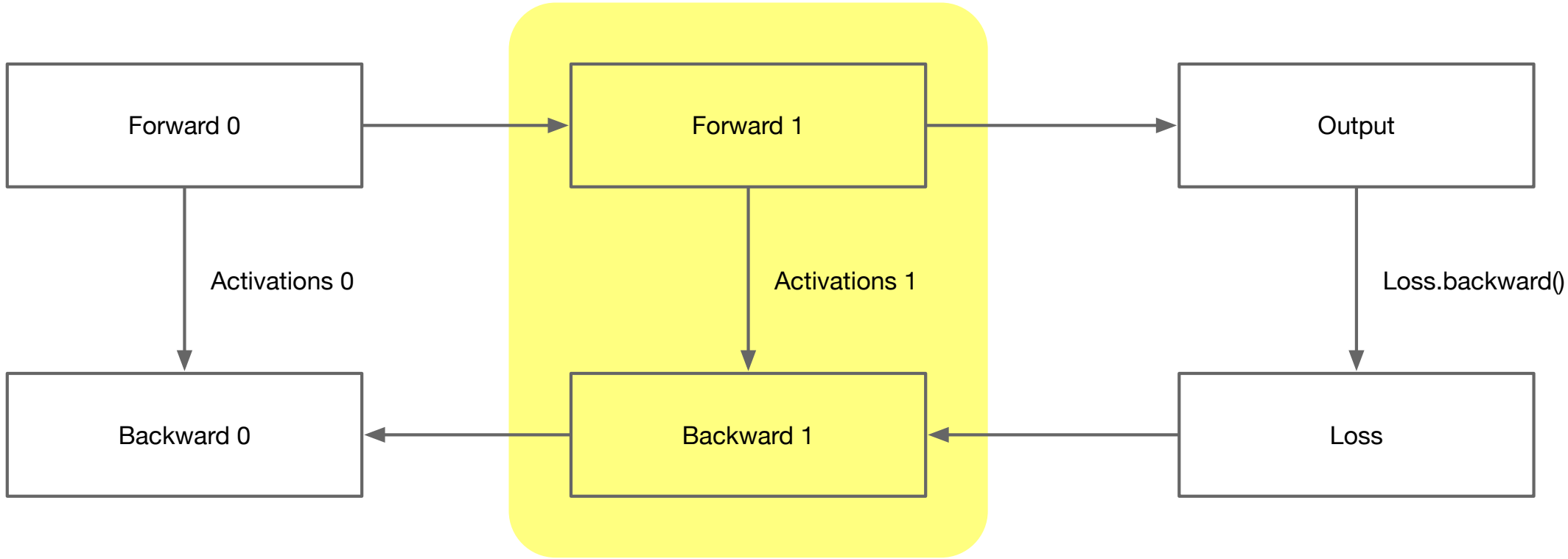
How does atuograd work?



How does atuograd work?



How does atuograd work?



fw + bw graph

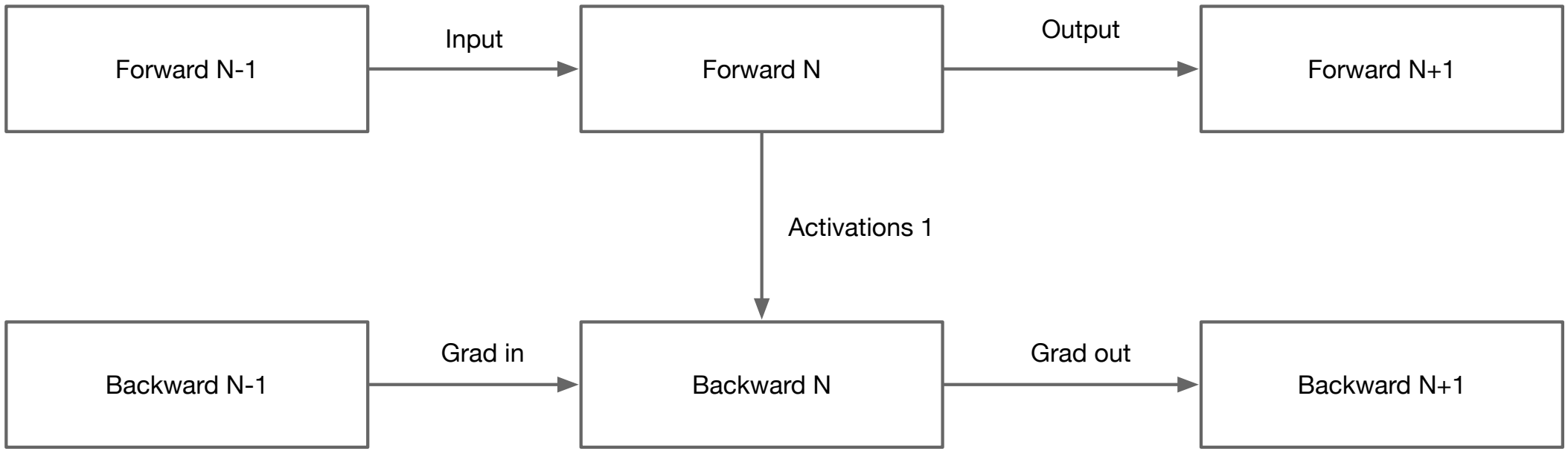
- The tape base object-oriented realization in Automatic differentiation:

```
import torch

def joint_forward_backward(primals, tangents):
    out = fn(*primals)
    backward_out = torch.autograd.grad(out, primals, grad_outputs = tangents)
    return out, backward_out

```

How does atuograd work?



AOTAutograd

	forward()+backward() improvement over eager mode on NVIDIA A100									
Frontend	TorchScript		TorchDynamo							
Backwards Capture			TorchScript				AOTAutograd			
Backend	NNC	nvFuser	NNC		nvFuser		NNC		nvFuser	
Metric	speedup	speedup	speedup	memory savings	speedup	memory savings	speedup	memory savings	speedup	memory savings
timm_vision_transformer	0.98x	-	1.13x	0.92x	0.98x	0.91x	1.33x	1.00x	1.30x	1.00x
mobilenet_v2_quantized_qat	-	-	1.14x	1.00x	-	-	1.09x	1.59x	1.27x	1.59x
BERT_pytorch	1.07x	-	1.14x	1.00x	-	-	1.22x	1.07x	1.27x	1.03x
shufflenet_v2_x1_0	0.99x	0.99x	0.99x	1.00x	1.29x	1.00x	1.00x	1.08x	1.26x	1.08x
timm_efficientnet	1.05x	1.06x	1.08x	1.00x	1.08x	1.00x	1.13x	1.55x	1.26x	1.55x
mobilenet_v2	1.00x	1.00x	1.04x	1.00x	1.28x	1.00x	0.97x	1.30x	1.26x	1.30x
mnasnet1_0	1.04x	1.03x	1.02x	1.00x	1.22x	1.00x	1.03x	1.00x	1.25x	1.00x
mobilenet_v3_large	1.06x	1.06x	1.04x	1.00x	1.13x	1.00x	1.11x	1.05x	1.20x	1.05x
LearningToPaint	1.04x	1.10x	1.02x	1.00x	1.16x	1.00x	1.04x	1.00x	1.19x	1.00x
resnet50	0.99x	0.99x	1.01x	1.00x	1.20x	1.00x	1.01x	1.00x	1.18x	1.00x
timm_nfnet	1.04x	1.06x	1.10x	1.00x	0.80x	1.00x	1.18x	1.64x	1.18x	1.64x
yolov3	-	-	0.99x	0.77x	-	-	0.99x	1.01x	1.18x	1.01x
timm_regnet	1.06x	1.07x	1.05x	1.00x	1.16x	1.00x	1.05x	1.11x	1.16x	1.11x
pytorch_CycleGAN_and_pix2pix	1.08x	1.08x	1.19x	1.00x	1.14x	1.00x	1.14x	1.00x	1.15x	1.00x