# Pytorch 2.0
# Overall introduction

by Minh Huu Nguyen

## PYTORCH 2.X:
## FASTER,
## MORE PYTHONIC
## AND AS DYNAMIC AS EVER

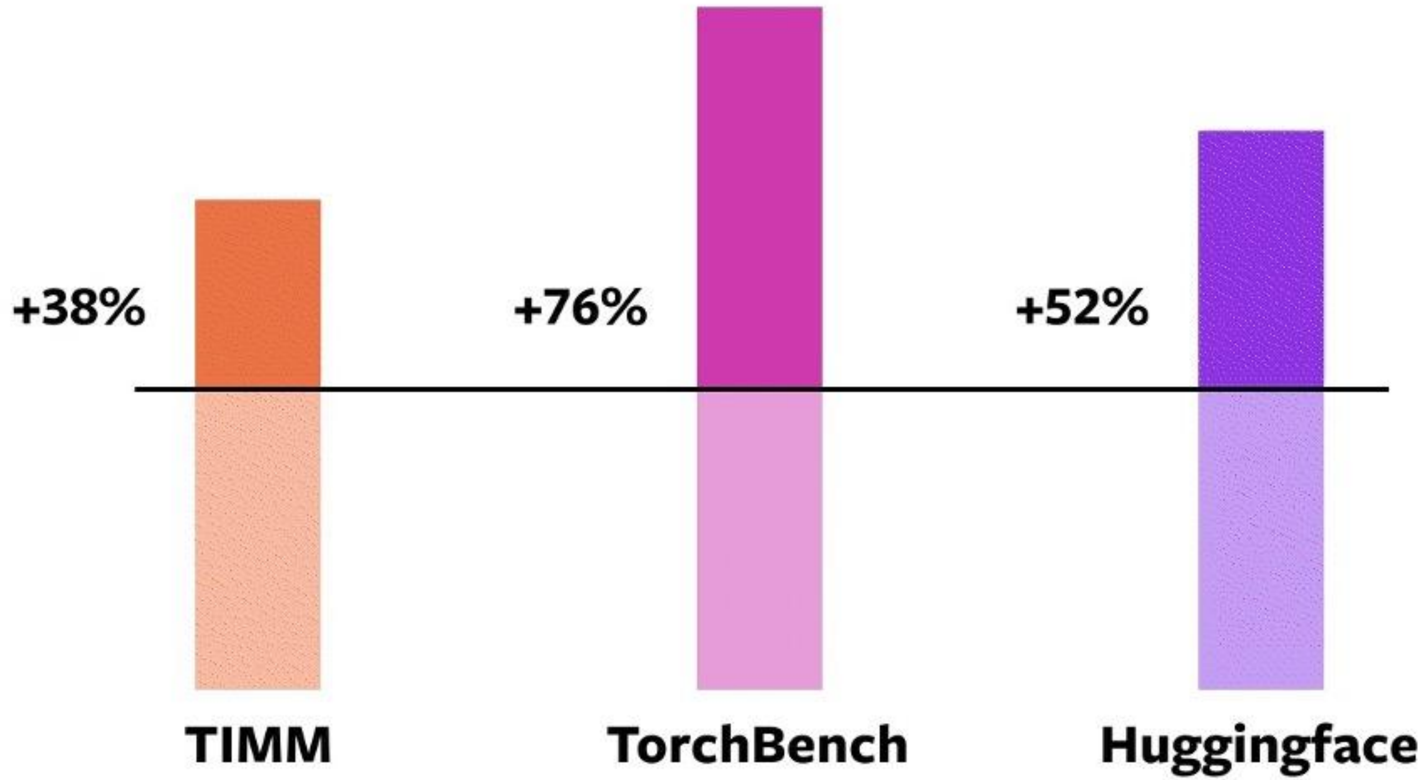
## Overall introduction

Some features of PyTorch 2.0 release are:

- OpenAI Triton deep learning compiler improve performance on Nvidia and AMD GPUs
- Accelerated Transformers with improved scaled dot product attention (SPDA)
- Metal Performance Shaders (MPS) backend accelerate PyTorch training on Mac platforms
- Amazon AWS optimizes the PyTorch CPU inference on AWS Graviton3 based C7g instances
- Some new technologies TensorParallel, DTensor, 2D parallel, TorchDynamo, AOTAutograd, PrimTorch and TorchInductor

## torch.compile()

## Performance



## Easy to use

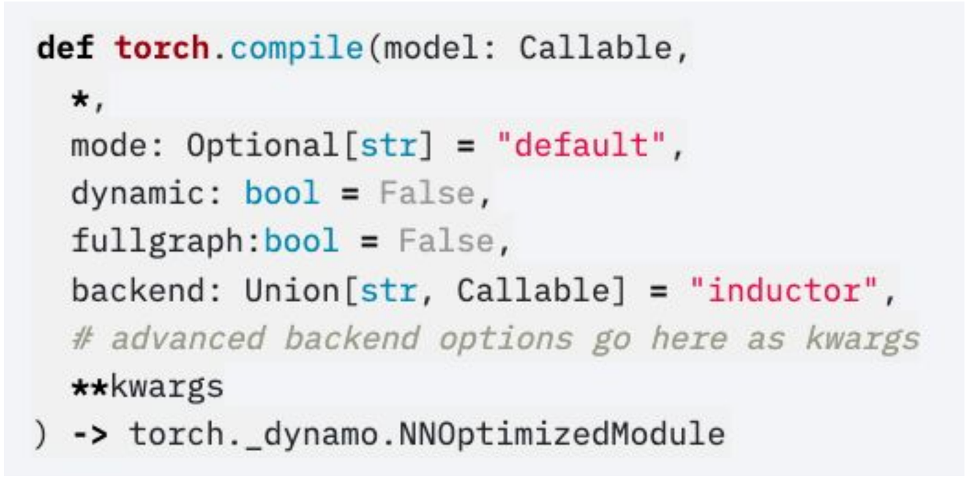
```python
x = torch.randn(32, 3, 64, 64)

optimizer.zero_grad()

out = model(x)

out.sum().backward()

optimizer.step()
```

```python
x = torch.randn(32, 3, 64, 64)

optimizer.zero_grad()

model = torch.compile(model)

out = model(x)

out.sum().backward()

optimizer.step()
```

## Under the hood

| | | | |
|---|---|---|---|
| 1. | TorchDynamo | Graph acquisition | Capture PyTorch program and create a forward graph |
| 2. | AOTAutograd | Backward graph | Create a backward graph |
| 3. | PrimTorch | Lowering | Canonicalize ~2000 PyTorch ops down to ~250 primitive ops |
| 4. | TorchInductor | Backend compiler | Generate fast code for each accelerators |

## Easy to use



```
def torch.compile(model: Callable,
  *,
  mode: Optional[str] = "default",
  dynamic: bool = False,
  fullgraph:bool = False,
  backend: Union[str, Callable] = "inductor",
  # advanced backend options go here as kwargs
  **kwargs
) -> torch._dynamo.NNOptimizedModule
```

mode: specifies what the compiler

- default: compile efficiently
- reduce-overhead: reduce overhead by a lot more, but cost an extra memory
- max-autotune: give the fastest code, but cost a long time

PIXTA
VIETNAM