torch.utils.mobile_optimizer

Warning

This API is in beta and may change in the near future.

Torch mobile supports `torch.mobile_optimizer.optimize_for_mobile` utility to run a list of optimization pass with modules in eval mode. The method takes the following parameters: a torch.jit.ScriptModule object, a blocklisting optimization set and a preserved method list

By default, if optimization blocklist is None or empty, `optimize_for_mobile` will run the following optimizations:

- **Conv2D + BatchNorm fusion** (blocklisting option *MobileOptimizerType::CONV_BN_FUSION*): This optimization pass folds `Conv2d-BatchNorm2d` into `Conv2d` in `forward` method of this module and all its submodules. The weight and bias of the `Conv2d` are correspondingly updated.
- **Insert and Fold prepacked ops** (blocklisting option *MobileOptimizerType::INSERT_FOLD_PREPACK_OPS*): This optimization pass rewrites the graph to replace 2D convolutions and linear ops with their prepacked counterparts. Prepacked ops are stateful ops in that, they require some state to be created, such as weight prepacking and use this state, i.e. prepacked weights, during op execution. XNNPACK is one such backend that provides prepacked ops, with kernels optimized for mobile platforms (such as ARM CPUs). Prepacking of weight enables efficient memory access and thus faster kernel execution. At the moment `optimize_for_mobile` pass rewrites the graph to replace `Conv2D/Linear` with 1) op that pre-packs weight for XNNPACK conv2d/linear ops and 2) op that takes pre-packed weight and activation as input and generates output activations. Since 1 needs to be done only once, we fold the weight pre-packing such that it is done only once at model load time. This pass of the `optimize_for_mobile` does 1 and 2 and then folds, i.e. removes, weight pre-packing ops.
- **ReLU/Hardtanh fusion**: XNNPACK ops support fusion of clamping. That is clamping of output activation is done as part of the kernel, including for 2D convolution and linear op kernels. Thus clamping effectively comes for free. Thus any op that can be expressed as clamping op, such as `ReLU` or `hardtanh`, can be fused with previous `Conv2D` or `linear` op in XNNPACK. This pass rewrites graph by finding `ReLU/hardtanh` ops that follow XNNPACK `Conv2D/linear` ops, written by the previous pass, and fuses them together.
- **Dropout removal** (blocklisting option *MobileOptimizerType::REMOVE_DROPOUT*): This optimization pass removes `dropout` and `dropout_` nodes from this module when training is false.
- **Conv packed params hoisting** (blocklisting option *MobileOptimizerType::HOIST_CONV_PACKED_PARAMS*): This optimization pass moves convolution packed params to the root module, so that the convolution structs can be deleted. This decreases model size without impacting numerics.

`optimize_for_mobile` will also invoke freeze_module pass which only preserves `forward` method. If you have other method to that needed to be preserved, add them into the preserved method list and pass into the method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\[pytorch-master][docs][source]mobile_optimizer.rst`, line 20)

Unknown directive type "currentmodule".

```
.. currentmodule:: torch.utils.mobile_optimizer
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\[pytorch-master][docs][source]mobile_optimizer.rst`, line 21)

Unknown directive type "autofunction".

```
.. autofunction:: optimize_for_mobile
```