Traceback (most recent call last)

```
in <module>:45

  42 optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
  43
  44 for epoch in range(1, 101):
❱ 45 |   train_loss = train(epoch)
  46 |   with torch.inference_mode():
  47 |   |   model.eval()
  48 |   |   train_acc = test(train_loader)

in train:25

  22 |   |   optimizer.zero_grad()
  23 |   |   output = model(data.x, data.edge_index, data.batch)
  24 |   |   loss = F.nll_loss(output, data.y)
❱ 25 |   |   loss.backward()
  26 |   |   loss_all += loss.item() * data.num_graphs
  27 |   |   optimizer.step()
  28 |   return loss_all / len(train_loader.dataset)

/usr/local/lib/python3.10/dist-packages/torch/_tensor.py:531 in backward

   528 |   |   |   |   create_graph=create_graph,
   529 |   |   |   |   inputs=inputs,
   530 |   |   |   )
❱  531 |   |   torch.autograd.backward(
   532 |   |   |   self, gradient, retain_graph, create_graph, inputs=inputs
   533 |   |   )
   534

/usr/local/lib/python3.10/dist-packages/torch/autograd/__init__.py:289 in backward

  286 |   # The reason we repeat the same comment below is that
  287 |   # some Python versions print out the first line of a multi-line function
  288 |   # calls in the traceback and some print out the last line
❱ 289 |   _engine_run_backward(
  290 |   |   tensors,
  291 |   |   grad_tensors_,
  292 |   |   retain_graph,

/usr/local/lib/python3.10/dist-packages/torch/autograd/graph.py:768 in _engine_run_backward

  765 |   if attach_logging_hooks:
  766 |   |   unregister_hooks = _register_logging_hooks_on_whole_graph(t_outputs)
  767 |   try:
❱ 768 |   |   return Variable._execution_engine.run_backward(  # Calls into the C++ engine to
  769 |   |   |   t_outputs, *args, **kwargs
  770 |   |   )  # Calls into the C++ engine to run the backward pass
  771 |   finally:

/usr/local/lib/python3.10/dist-packages/torch/autograd/function.py:306 in apply

  303 |   |   |   |   "of them."
  304 |   |   |   )
  305 |   |   user_fn = vjp_fn if vjp_fn is not Function.vjp else backward_fn
❱ 306 |   |   return user_fn(self, *args)
  307 |
  308 |   def apply_jvp(self, *args):
  309 |   |   r"""

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/runtime_wrappers.py:1861
in backward

  1858 |   |   |   |   |   # Pass args even though they're unused, so that the graph is built
  1859 |   |   |   |   |   out = CompiledFunctionBackward.apply(*all_args)
  1860 |   |   |   |   else:
❱ 1861 |   |   |   |   |   out = call_compiled_backward()
  1862 |   |   |   |
  1863 |   |   |   |   # TODO: figure out how to refactor the backward properly so I can use ao
  1864 |   |   |   |   if CompiledFunction.maybe_subclass_metadata is not None:

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/runtime_wrappers.py:1809
in call_compiled_backward

  1806 |   |   |   |   |   |   |   |   bw_module, placeholder_list
  1807 |   |   |   |   |   |   |   )
  1808 |   |   |   |   |
```

```
❱ 1809 │   │   │   │   │   out = call_func_at_runtime_with_args(
  1810 │   │   │   │   │   │   CompiledFunction.compiled_bw,
  1811 │   │   │   │   │   │   all_args,
  1812 │   │   │   │   │   │   steal_args=True,
```

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/**utils.py**:120 in call_func_at_runtime_with_args

```
  117 │   context = torch._C._DisableAutocast if disable_amp else nullcontext
  118 │   with context():
  119 │   │   if hasattr(f, "_boxed_call"):
❱ 120 │   │   │   out = normalize_as_list(f(args))
  121 │   │   else:
  122 │   │   │   # TODO: Please remove soon
  123 │   │   │   # https://github.com/pytorch/pytorch/pull/83137#issuecomment-1211320670
```

/usr/local/lib/python3.10/dist-packages/torch/_dynamo/**eval_frame.py**:600 in _fn

```
  597 │   │   def _fn(*args, **kwargs):
  598 │   │   │   prior = set_eval_frame(callback)
  599 │   │   │   try:
❱ 600 │   │   │   │   return fn(*args, **kwargs)
  601 │   │   │   finally:
  602 │   │   │   │   set_eval_frame(prior)
  603
```

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/**utils.py**:94 in g

```
   91
   92 def make_boxed_func(f):
   93 │   def g(args):
❱  94 │   │   return f(*args)
   95 │
   96 │   g._boxed_call = True  # type: ignore[attr-defined]
   97 │   return g
```

/usr/local/lib/python3.10/dist-packages/torch/fx/**_lazy_graph_module.py**:124 in _lazy_forward

```
  121 │   │   # call `__call__` rather than 'forward' since recompilation may
  122 │   │   # install a wrapper for `__call__` to provide a customized error
  123 │   │   # message.
❱ 124 │   │   return self(*args, **kwargs)
  125 │
  126 │   forward = _lazy_forward
  127
```

/usr/local/lib/python3.10/dist-packages/torch/fx/**graph_module.py**:738 in call_wrapped

```
  735 │   │   │   cls._wrapped_call = _WrappedCall(cls, cls_call)  # type: ignore[attr-defined
  736 │   │
  737 │   │   def call_wrapped(self, *args, **kwargs):
❱ 738 │   │   │   return self._wrapped_call(self, *args, **kwargs)
  739 │   │
  740 │   │   cls.__call__ = call_wrapped  # type: ignore[method-assign]
  741
```

/usr/local/lib/python3.10/dist-packages/torch/fx/**graph_module.py**:316 in __call__

```
  313 │   │   │   │   )
  314 │   │   │   │   raise e.with_traceback(None)  # noqa: B904
  315 │   │   │   else:
❱ 316 │   │   │   │   raise e
  317
  318 @compatibility(is_backward_compatible=True)
  319 class GraphModule(torch.nn.Module):
```

/usr/local/lib/python3.10/dist-packages/torch/fx/**graph_module.py**:303 in __call__

```
  300 │   │   │   if self.cls_call is not None:
  301 │   │   │   │   return self.cls_call(obj, *args, **kwargs)
  302 │   │   │   else:
❱ 303 │   │   │   │   return super(self.cls, obj).__call__(*args, **kwargs)  # type: ignore[mi
  304 │   │   except Exception as e:
  305 │   │   │   assert e.__traceback__
  306 │   │   │   topmost_framesummary: traceback.FrameSummary = (
```

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/**module.py**:1556 in _wrapped_call_impl

```
  1553 |    |    if self._compiled_call_impl is not None:
  1554 |    |    |    return self._compiled_call_impl(*args, **kwargs)  # type: ignore[misc]
  1555 |    |    else:
❱ 1556 |    |    |    return self._call_impl(*args, **kwargs)
  1557 |
  1558 |    def _call_impl(self, *args, **kwargs):
  1559 |    |    forward_call = (self._slow_forward if torch._C._get_tracing_state() else self.fo

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py:1565 in _call_impl

  1562 |    |    if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks
  1563 |    |    |    |    or _global_backward_pre_hooks or _global_backward_hooks
  1564 |    |    |    |    or _global_forward_hooks or _global_forward_pre_hooks):
❱ 1565 |    |    |    return forward_call(*args, **kwargs)
  1566 |    |
  1567 |    |    try:
  1568 |    |    |    result = None
in forward:5

/usr/local/lib/python3.10/dist-packages/habana_frameworks/torch/dynamo/compile_backend/recipe_co
mpiler.py:242 in __call__

  239 |    |    |    |    self._range_list.insert(0, RangeInfo([1], [1], "1", "1", 0))
  240 |    |    |    |    self._range_list.insert(1, RangeInfo([1], [1], "1", "1", 1))
  241 |    |    |
❱ 242 |    |    |    self._recipe_id = graph_compile(
  243 |    |    |    |    graph=self._jit_ir.graph,
  244 |    |    |    |    inputs=inputs,
  245 |    |    |    |    dynamic=self._dynamic,
```

RuntimeError: Input sizes must be equal