```
───────────────────── Traceback (most recent call last) ─────────────────────
in <module>:5

   2 model = torch.compile(model, backend="hpu_backend")
   3
   4 for epoch in trange(1, 171):
❱  5 │   train()
   6 │   train_acc, train_loss, _ = test(train_loader)
   7 │   test_acc, test_loss, test_table = test(test_loader, create_table=True)
   8

in train:15

   12 │
   13 │   for data in train_loader:  # Iterate in batches over the training dataset.
   14 │   │   data = data.to(device)
❱  15 │   │   out = model(data.x, data.edge_index, data.batch)  # Perform a single forward pas
   16 │   │   loss = criterion(out, data.y)  # Compute the loss.
   17 │   │   loss.backward()  # Derive gradients.
   18 │   │   optimizer.step()  # Update parameters based on gradients.

/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

/usr/local/lib/python3.10/dist-packages/torch/_dynamo/eval_frame.py:433 in _fn

   430 │   │   │   │   )
   431 │
   432 │   │   │   │   try:
❱  433 │   │   │   │   │   return fn(*args, **kwargs)
   434 │   │   │   │   finally:
   435 │   │   │   │   │   # Restore the dynamic layer stack depth if necessary.
   436 │   │   │   │   │   torch._C._functorch.pop_dynamic_layer_stack_and_undo_to_depth(

/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:18

   15 │
   16 │   def forward(self, x, edge_index, batch):
   17 │   │   # 1. Obtain node embeddings
❱  18 │   │   x = self.conv1(x, edge_index)
   19 │   │   x = x.relu()
   20 │   │   x = self.conv2(x, edge_index)
   21 │   │   x = x.relu()

/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

/usr/local/lib/python3.10/dist-packages/torch_geometric/nn/conv/gcn_conv.py:241 in forward

  238 │   │   │       if isinstance(edge_index, Tensor):
  239 │   │   │   │       cache = self._cached_edge_index
  240 │   │   │   │       if cache is None:
❱ 241 │   │   │   │   │       edge_index, edge_weight = gcn_norm(  # yapf: disable
  242 │   │   │   │   │           edge_index, edge_weight, x.size(self.node_dim),
  243 │   │   │   │   │           self.improved, self.add_self_loops, self.flow, x.dtype)
  244 │   │   │   │   │       if self.cached:

/usr/local/lib/python3.10/dist-packages/torch_geometric/nn/conv/gcn_conv.py:241 in
torch_dynamo_resume_in_forward_at_241

  238 │   │   │       if isinstance(edge_index, Tensor):
  239 │   │   │   │       cache = self._cached_edge_index
  240 │   │   │   │       if cache is None:
❱ 241 │   │   │   │   │       edge_index, edge_weight = gcn_norm(  # yapf: disable
  242 │   │   │   │   │           edge_index, edge_weight, x.size(self.node_dim),
  243 │   │   │   │   │           self.improved, self.add_self_loops, self.flow, x.dtype)
  244 │   │   │   │   │       if self.cached:

/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.py:987 in forward

  984 │   │       full_args = []
  985 │   │       full_args.extend(params_flat)
  986 │   │       full_args.extend(runtime_args)
❱ 987 │   │       return compiled_fn(full_args)
  988 │
  989 │       # Just for convenience
  990 │       forward.zero_grad = mod.zero_grad

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/runtime_wrappers.py:204
in runtime_wrapper

  201 │   │   │   │       if isinstance(args_[idx], torch.Tensor):
  202 │   │   │   │   │       args_[idx] = args_[idx].detach()
  203 │   │   │       with torch.autograd._force_original_view_tracking(True):
❱ 204 │   │   │   │       all_outs = call_func_at_runtime_with_args(
  205 │   │   │   │   │           compiled_fn, args_, disable_amp=disable_amp, steal_args=True
  206 │   │   │   │       )
  207 │   │       else:

/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/_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/autograd/function.py:574 in apply

  571 │   │       if not torch._C._are_functorch_transforms_active():
  572 │   │           # See NOTE: [functorch vjp and autograd interaction]
  573 │   │           args = _functorch.utils.unwrap_dead_wrappers(args)
❱ 574 │   │           return super().apply(*args, **kwargs)  # type: ignore[misc]
  575
  576 │   │       if not is_setup_ctx_defined:
```

```
 577 │    │    │       raise RuntimeError(
```

/usr/local/lib/python3.10/dist-packages/torch/_functorch/_aot_autograd/**runtime_wrappers.py**:1451
in forward

```
1448 │    │    │    │       # (*tokens, *mutated_inputs, *fw_outs, *fw_intermediate_bases, *saved_te
1449 │    │    │    │       # - Note that in the synthetic bases case, mutated_inputs will correspon
1450 │    │    │    │       #   of the original view, and not the synthetic base
❱ 1451 │    │    │    │       fw_outs = call_func_at_runtime_with_args(
1452 │    │    │    │    │       CompiledFunction.compiled_fw,
1453 │    │    │    │    │       args,
1454 │    │    │    │    │       disable_amp=disable_amp,
```

/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/_functorch/_aot_autograd/**runtime_wrappers.py**:451
in wrapper

```
 448 │    │    │    │    │       runtime_metadata.num_forward_returns,
 449 │    │    │    │       )
 450 │    │    │    │       return out
❱ 451 │    │    │       return compiled_fn(runtime_args)
 452 │    │
 453 │    │       return wrapper
 454
```

/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:6

/usr/local/lib/python3.10/dist-packages/torch/_ops.py:667 in __call__

   664 │     def __call__(self_, *args, **kwargs):  # noqa: B902
   665 │     │     # use `self_` to avoid naming collide with aten ops arguments that
   666 │     │     # are named "self". This way, all the aten ops can be called by kwargs.
❱  667 │     │     return self_._op(*args, **kwargs)
   668 │
   669 │     def redispatch(self_, keyset, *args, **kwargs):  # noqa: B902
   670 │     │     # use `self_` to avoid naming collide with aten ops arguments that
```

RuntimeError: [Rank:0] FATAL ERROR :: MODULE:PT_EAGER HabanaLaunchOpPT Run returned exception....
synNodeCreateWithId failed for node: concat with synStatus 1 [Invalid argument]. .
[Rank:0] Habana exception raised from add_node at graph.cpp:507