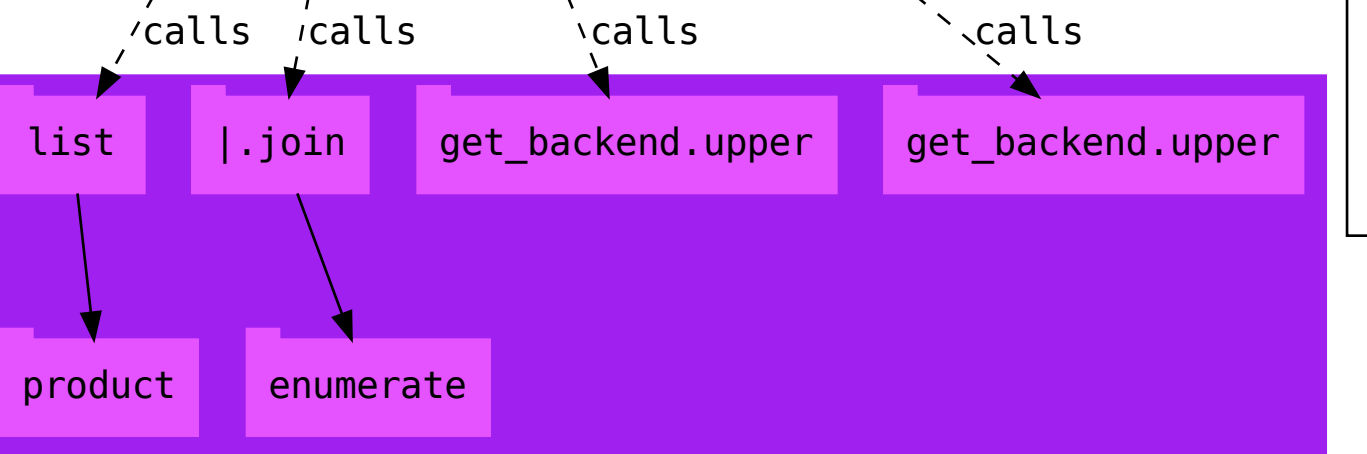
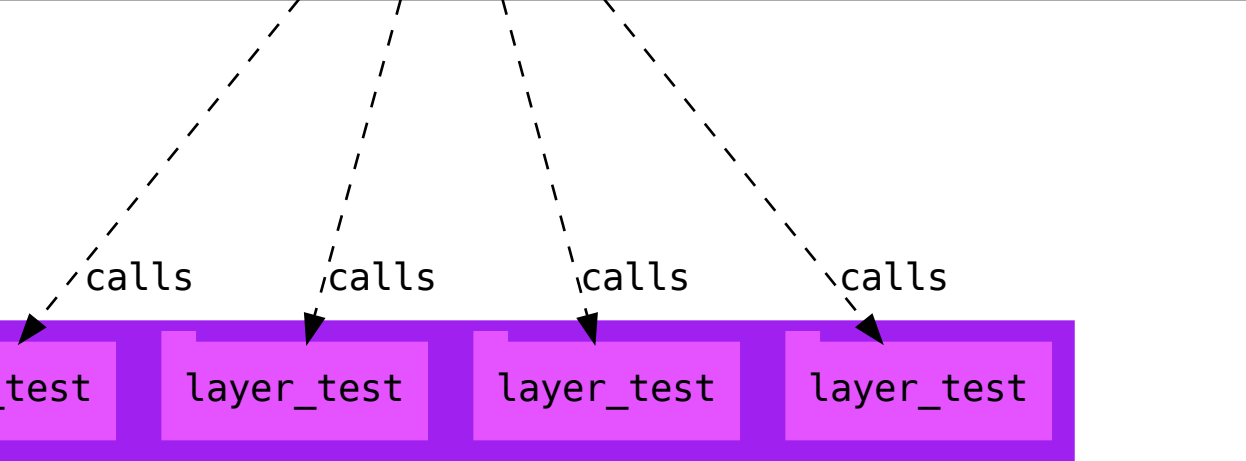


```
from itertools import product
import numpy as np
import pytest
from tensorflow.keras import regularizers, models, layers
from lib.model import normalization
from lib.utils import get_backend
from tests.lib.model.layers_test import layer_test
@pytest.mark.parametrize('dummy', [None], ids=[get_backend().upper()])...
@pytest.mark.parametrize('dummy', [None], ids=[get_backend().upper()])...
_PARAMS = ['center', 'scale']
_VALUES = list(product([True, False], repeat=len(_PARAMS)))
_IDS = [
    f"'{'|'}.join(_PARAMS[idx] ..."
    for v in _VALUES]
@pytest.mark.parametrize(_PARAMS, _VALUES, ids=_IDS)...
_PARAMS = ['partial', 'bias']
_VALUES = [(0.0, False), (0.25, False), (0.5, True), (0.75, False), (1.0, True)]
_IDS = [f'partial={v[0]}|bias=...' for v in _VALUES]
@pytest.mark.parametrize(_PARAMS, _VALUES, ids=_IDS)...
```



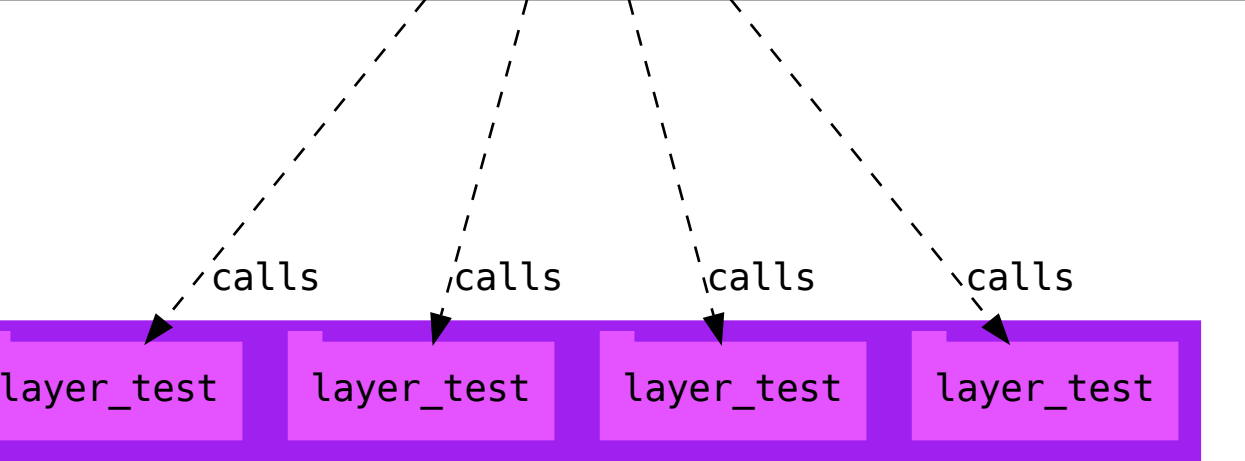
```
test_instance_normalization

layer_test(normalization.InstanceNormalization, kwargs={'epsilon': 0.1,
'gamma_regularizer': regularizers.l2(0... 'beta_regularizer':
regularizers.l2(0.01)}, input_shape=(3, 4, 2))
layer_test(normalization.InstanceNormalization, kwargs={'epsilon': 0.1,
'axis': 1}, input_shape=(1, 4, 1))
layer_test(normalization.InstanceNormalization, kwargs={'gamma_initializer':
'ones', 'beta_initializer': 'ones'}, input_shape=(3, 4, 2, 4))
layer_test(normalization.InstanceNormalization, kwargs={'epsilon': 0.1,
'axis': 1, 'scale': False, 'center': False}, input_shape=(3, 4, 2, 4))
```



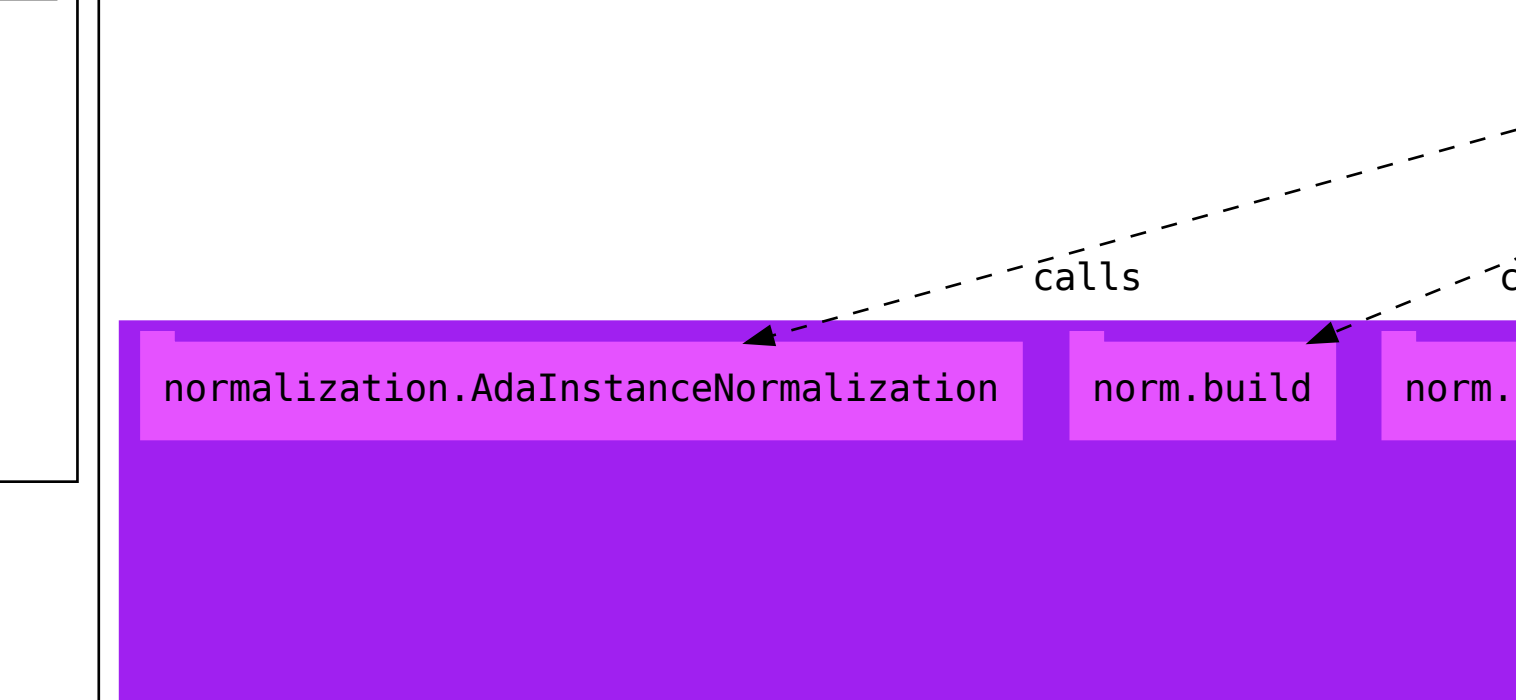
```
test_group_normalization

layer_test(normalization.GroupNormalization, kwargs={'epsilon': 0.1,
'gamma_regularizer': regularizers.l2(0... 'beta_regularizer':
regularizers.l2(0.01)}, input_shape=(4, 3, 4, 128))
layer_test(normalization.GroupNormalization, kwargs={'epsilon': 0.1, 'axis':
1}, input_shape=(4, 1, 4, 256))
layer_test(normalization.GroupNormalization, kwargs={'gamma_init': 'ones',
'beta_init': 'ones'}, input_shape=(4, 64))
layer_test(normalization.GroupNormalization, kwargs={'epsilon': 0.1, 'axis':
1, 'group': 16}, input_shape=(3, 64))
```



```
test_adain_normalization

norm = normalization.AdaInstanceNormalization(center=center, scale=scale)
shapes = [(4, 8, 8, 1280), (4, 1, 1, 1280), (4, 1, 1, 1280)]
norm.build(shapes)
expected_output_shape = norm.compute_output_shape(shapes)
inputs = [layers.Input(shape=shapes[0][1:]), layers.Input(shape=shapes[1][1
:]), layers.Input(shape=shapes[2][1:])]
model = models.Model(inputs, norm(inputs))
data = [(10 * np.random.random(shape)) for shape in shapes]
actual_output = model.predict(data, verbose=0)
actual_output_shape = actual_output.shape
```



```
for expected_dim, actual_dim in zip(expected_output_shape, actual_output_shape)
```



```
test_rms_normalization

layer_test(normalization.RMSNormalization, kwargs={'partial': partial,
'bias': bias}, input_shape=(4, 512))
```

