```python
from .layers import *

def affine_relu_forward(x, w, b):
    """
    Convenience layer that performs an affine transform followed by a ReLU

    Inputs:
    - x: Input to the affine layer
    - w, b: Weights for the affine layer

    Returns a tuple of:
    - out: Output from the ReLU
    - cache: Object to give to the backward pass
    """
    a, fc_cache = affine_forward(x, w, b)
    out, relu_cache = relu_forward(a)
    cache = (fc_cache, relu_cache)
    return out, cache


def affine_relu_backward(dout, cache):
    """
    Backward pass for the affine-relu convenience layer
    """
    fc_cache, relu_cache = cache
    da = relu_backward(dout, relu_cache)
    dx, dw, db = affine_backward(da, fc_cache)
    return dx, dw, db

def affine_batchnorm_relu_forward(x,w,b,gamma,beta,bn_param):
    a_out, a_cache = affine_forward(x,w,b)
    batch_out, batch_cache = batchnorm_forward(a_out, gamma,beta,bn_param)
    out, relu_cache = relu_forward(batch_out)
    cache = (a_cache, relu_cache, batch_cache)
    return out, cache

def affine_batchnorm_relu_backward(dout, cache):
    a_cache, relu_cache, batch_cache = cache
    dbatch = relu_backward(dout, relu_cache)
    da, dgamma, dbeta = batchnorm_backward(dbatch, batch_cache)
    dx, dw, db = affine_backward(da, a_cache)
    return dx,dw, db, dgamma, dbeta
```