```python
from .layers import *

"""
This code was originally written for CS 231n at Stanford University
(cs231n.stanford.edu).  It has been modified in various areas for use in the
ECE 239AS class at UCLA.  This includes the descriptions of what code to
implement as well as some slight potential changes in variable names to be
consistent with class nomenclature.  We thank Justin Johnson & Serena Yeung for
permission to use this code.  To see the original version, please visit
cs231n.stanford.edu.
"""
#  {affine - [batch norm] - relu - [dropout]} x (L - 1) - affine - softmax

def affine_batch_relu_forward(x, w, b, gamma, beta, bn_param):
    """
    Convenience layer that performs an affine transform followed by batchnorm 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
    """
    x1, fc_cache = affine_forward(x, w, b)
    x2, bn_cache = batchnorm_forward(x1, gamma, beta, bn_param)
    out, relu_cache = relu_forward(x2)
    cache = (fc_cache, bn_cache, relu_cache)
    return out, cache

def affine_batch_relu_backward(dout, cache):
    """
    Backward pass for the affine-batchnorm-relu convenience layer
    """
    fc_cache, bn_cache, relu_cache = cache
    da = relu_backward(dout, relu_cache)
    dx_bn, dgamma, dbeta = batchnorm_backward(da, bn_cache)

    # print(len(db_norm))

    dx, dw, db = affine_backward(dx_bn, fc_cache)
    return dx, dw, db, dgamma, dbeta


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
    """
```

```python
    fc_cache, relu_cache = cache
    da = relu_backward(dout, relu_cache)
    dx, dw, db = affine_backward(da, fc_cache)
    return dx, dw, db
```