```
#layer utils

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_params):
    """
    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)
    # print("a shape", a.shape)
    # print("gamma shape", gamma.shape)
    batchnorm, batch_cache = batchnorm_forward(a, gamma, beta,
bn_params)
```

```
    out, relu_cache = relu_forward(batchnorm)
    cache = (fc_cache, batch_cache, relu_cache)
    return out, cache



def affine_batchnorm_relu_backward(dout, cache):
    """
    Backward pass for the affine-relu convenience layer
    """
    fc_cache, batch_cache, relu_cache = cache
    da = relu_backward(dout, relu_cache)
    # print("relu bwd done")
    dx, dgamma, dbeta = batchnorm_backward(da, batch_cache)

    dx, dw, db = affine_backward(dx, fc_cache)
    # print("aff bwd done")
    return dx, dw, db, dgamma, dbeta
```