layer_utils

February 10, 2023

[ ]:
```
# %load layer_utils.py
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):
    #forward of Affine - BN - Relu
    aff_out, aff_cache = affine_forward(x, w, b)
    bn_out, bn_cache = batchnorm_forward(aff_out, gamma, beta, bn_param)
    out, relu_cache = relu_forward(bn_out)
    cache = (aff_cache, bn_cache, relu_cache)
    return out, cache
```

```python
def affine_batchnorm_relu_backward(dout, cache):
    aff_cache, bn_cache, relu_cache = cache
    d_relu = relu_backward(dout, relu_cache)
    db, dgamma, dbeta = batchnorm_backward(d_relu, bn_cache)
    dx, dw, db = affine_backward(db, aff_cache)
    return dx, dw, db, dgamma, dbeta

def affine_batchnorm_relu_dropout_forward(x, w, b, gamma, beta, bn_param,␣
 ↪dropout_param):
    #forward of Affine - BN - Relu
    aff_out, aff_cache = affine_forward(x, w, b)
    bn_out, bn_cache = batchnorm_forward(aff_out, gamma, beta, bn_param)
    relu_out, relu_cache = relu_forward(bn_out)
    out, dropout_cache = dropout_forward(relu_out, dropout_param)
    cache = (aff_cache, bn_cache, relu_cache, dropout_cache)
    return out, cache

def affine_batchnorm_relu_dropout_backward(dout, cache):
    aff_cache, bn_cache, relu_cache, dropout_cache = cache
    d_dropout = dropout_backward(dout, dropout_cache)
    d_relu = relu_backward(d_dropout, relu_cache)
    db, dgamma, dbeta = batchnorm_backward(d_relu, bn_cache)
    dx, dw, db = affine_backward(db, aff_cache)
    return dx, dw, db, dgamma, dbeta
```