

```

1 from .layers import *
2
3 """
4 This code was originally written for CS 231n at Stanford University
5 (cs231n.stanford.edu). It has been modified in various areas for use in the
6 ECE 239AS class at UCLA. This includes the descriptions of what code to
7 implement as well as some slight potential changes in variable names to be
8 consistent with class nomenclature. We thank Justin Johnson & Serena Yeung for
9 permission to use this code. To see the original version, please visit
10 cs231n.stanford.edu.
11 """
12
13 def affine_relu_forward(x, w, b):
14     """
15     Convenience layer that performs an affine transform followed by a ReLU
16
17     Inputs:
18     - x: Input to the affine layer
19     - w, b: Weights for the affine layer
20
21     Returns a tuple of:
22     - out: Output from the ReLU
23     - cache: Object to give to the backward pass
24     """
25     a, fc_cache = affine_forward(x, w, b)
26     out, relu_cache = relu_forward(a)
27     cache = (fc_cache, relu_cache)
28     return out, cache
29
30
31 def affine_relu_backward(dout, cache):
32     """
33     Backward pass for the affine-relu convenience layer
34     """
35     fc_cache, relu_cache = cache
36     da = relu_backward(dout, relu_cache)
37     dx, dw, db = affine_backward(da, fc_cache)
38     return dx, dw, db
39
40
41
42 def affine_batchnorm_relu(x, w, b, gamma, beta, bn_param):
43     a, fc_cache = affine_forward(x, w, b)
44     a1, bn_cache = batchnorm_forward(a, gamma, beta, bn_param)
45     out, relu_cache = relu_forward(a1)
46     cache = (fc_cache, bn_cache, relu_cache)
47     return out, cache
48
49
50 def affine_batchnorm_relu_back(dout, cache):
51     fc_cache, bn_cache, relu_cache = cache
52     da = relu_backward(dout, relu_cache)
53     da1, dgamma, dbeta = batchnorm_backward(da, bn_cache)
54     dx, dw, db = affine_backward(da1, fc_cache)
55     return dx, dw, db, dgamma, dbeta

```