

```

1  import numpy as np
2
3
4  def random_flips(X):
5      """
6      Take random x-y flips of images.
7
8      Input:
9      - X: (N, C, H, W) array of image data.
10
11      Output:
12      - An array of the same shape as X, containing a copy of the data in X,
13        but with half the examples flipped along the horizontal direction.
14      """
15      out = None
16      #####
17      # TODO: Implement the random_flips function. Store the result in out.      #
18      #####
19      N, C, H, W = X.shape
20
21      flips = np.random.randint(2, size=N)
22      out = np.zeros(X.shape)
23      out[flips == 1] = X[flips == 1, :, :, :-1]
24      out[flips == 0] = X[flips == 0]
25      #####
26      #                                     END OF YOUR CODE                                     #
27      #####
28      return out
29
30
31  def random_crops(X, crop_shape):
32      """
33      Take random crops of images. For each input image we will generate a random
34      crop of that image of the specified size.
35
36      Input:
37      - X: (N, C, H, W) array of image data
38      - crop_shape: Tuple (HH, WW) to which each image will be cropped.
39
40      Output:
41      - Array of shape (N, C, HH, WW)
42      """
43      N, C, H, W = X.shape
44      HH, WW = crop_shape
45      assert HH < H and WW < W
46
47      out = np.zeros((N, C, HH, WW), dtype=X.dtype)
48      #####
49      # TODO: Implement the random_crops function. Store the result in out.      #
50      #####
51      np.random.randint((H-HH), size=N)
52      y_crop = np.random.randint((H-HH), size=N)
53      x_crop = np.random.randint((W-WW), size=N)
54
55      for i in xrange(N):
56          out[i] = X[i, :, y_crop[i]:y_crop[i]+HH, x_crop[i]:x_crop[i]+WW]
57      #####
58      #                                     END OF YOUR CODE                                     #
59      #####
60
61      return out
62
63
64  def random_contrast(X, scale=(0.8, 1.2)):
65      """
66      Randomly adjust the contrast of images. For each input image, choose a
67      number uniformly at random from the range given by the scale parameter,

```

```

68 and multiply each pixel of the image by that number.
69
70 Inputs:
71 - X: (N, C, H, W) array of image data
72 - scale: Tuple (low, high). For each image we sample a scalar in the
73     range (low, high) and multiply the image by that scaler.
74
75 Output:
76 - Rescaled array out of shape (N, C, H, W) where out[i] is a contrast
77   adjusted version of X[i].
78 """
79 low, high = scale
80 N = X.shape[0]
81 out = np.zeros_like(X)
82
83 #####
84 # TODO: Implement the random_contrast function. Store the result in out.      #
85 #####
86 contrast = (scale[1]-scale[0])*np.random.random_sample(N)+scale[0]
87 out = X * contrast[:, None, None, None]
88 #####
89 #                                     END OF YOUR CODE                         #
90 #####
91
92 return out
93
94
95 def random_tint(X, scale=(-10, 10)):
96     """
97     Randomly tint images. For each input image, choose a random color whose
98     red, green, and blue components are each drawn uniformly at random from
99     the range given by scale. Add that color to each pixel of the image.
100
101     Inputs:
102     - X: (N, C, W, H) array of image data
103     - scale: A tuple (low, high) giving the bounds for the random color that
104         will be generated for each image.
105
106     Output:
107     - Tinted array out of shape (N, C, H, W) where out[i] is a tinted version
108       of X[i].
109     """
110     low, high = scale
111     N, C = X.shape[:2]
112     out = np.zeros_like(X)
113
114     #####
115     # TODO: Implement the random_tint function. Store the result in out.      #
116     #####
117     tint = (scale[1]-scale[0])*np.random.random_sample((N, C))+scale[0]
118     out = X+tint[:, :, None, None]
119     #####
120     #                                     END OF YOUR CODE                         #
121     #####
122
123     return out
124
125
126 def fixed_crops(X, crop_shape, crop_type):
127     """
128     Take center or corner crops of images.
129
130     Inputs:
131     - X: Input data, of shape (N, C, H, W)
132     - crop_shape: Tuple of integers (HH, WW) giving the size to which each
133         image will be cropped.
134     - crop_type: One of the following strings, giving the type of crop to

```

```

135         compute:
136         'center': Center crop
137         'ul': Upper left corner
138         'ur': Upper right corner
139         'bl': Bottom left corner
140         'br': Bottom right corner
141
142     Returns:
143     Array of cropped data of shape (N, C, HH, WW)
144     """
145     N, C, H, W = X.shape
146     HH, WW = crop_shape
147
148     x0 = (W - WW) / 2
149     y0 = (H - HH) / 2
150     x1 = x0 + WW
151     y1 = y0 + HH
152
153     if crop_type == 'center':
154         return X[:, :, y0:y1, x0:x1]
155     elif crop_type == 'ul':
156         return X[:, :, :HH, :WW]
157     elif crop_type == 'ur':
158         return X[:, :, :HH, -WW:]
159     elif crop_type == 'bl':
160         return X[:, :, -HH:, :WW]
161     elif crop_type == 'br':
162         return X[:, :, -HH:, -WW:]
163     else:
164         raise ValueError('Unrecognized crop type %s' % crop_type)
165
166

```