Exercise 05 — Solution Reference

I2DL — Introduction to Deep Learning

exercise_code/networks/layer.py

```
###########################################################################
# TODO:                                                                   #
# Implement the forward pass of Sigmoid activation function               #
###########################################################################
out = 1 / (1 + np.exp(-x))
cache = out
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

```
###########################################################################
# TODO:                                                                   #
# Implement the backward pass of Sigmoid activation function              #
###########################################################################
dx = dout * cache * (1 - cache)
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

```
###########################################################################
# TODO:                                                                   #
# Implement the forward pass of Relu activation function                  #
###########################################################################
out = np.maximum(x, 0)
cache = x
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

```
###########################################################################
# TODO:                                                                   #
# Implement the backward pass of Relu activation function                 #
###########################################################################
x = cache
dx = dout
```

```
dx[x < 0] = 0
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

```
###########################################################################
# TODO: Implement the affine forward pass. Store the result in out.       #
# You will need to reshape the input into rows.                           #
###########################################################################
x_reshaped = np.reshape(x, (x.shape[0], -1))
out = x_reshaped.dot(w) + b
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

```
###########################################################################
# TODO: Implement the affine backward pass.                               #
###########################################################################

dw = (np.reshape(x, (x.shape[0], -1)).T).dot(dout)
dw = np.reshape(dw, w.shape)

db = np.sum(dout, axis=0, keepdims=False)

dx = dout.dot(w.T)
dx = np.reshape(dx, x.shape)
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```

exercise_code/networks/optimizer.py

```
###########################################################################
# TODO: Implement the momentum update formula. Store the updated          #
# value in the next_w variable. You should also use and update the        #
# velocity v.                                                             #
###########################################################################
mu = config['momentum']
learning_rate = lr
v = mu * v - learning_rate * dw
next_w = w + v
###########################################################################
#                             END OF YOUR CODE                            #
###########################################################################
```