1. initialize_parameters_deep (layer_dims)
   return parameters { $W^{[1]} \dots W^{[L]}$, $b^{[1]} \dots b^{[L]}$

2. linear_forward (A, W, b)
   return Z, cache $Z = np.dot(W, A) + b$ forward propagation
   cache (A, W, b)

3. linear_activation_forward (A_prev, W, b, activation)
   if activation = "sigmoid":
   Z, linear_cache = linear_forward (A_prev, W, b) Way of activation
   A, activation_cache = sigmoid(Z)

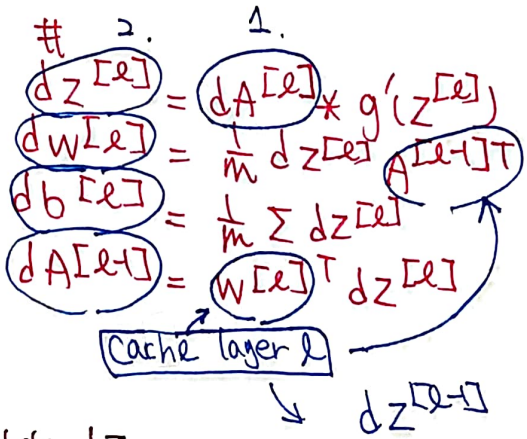
   elif activation = "relu":
   ≈
   ≈

   cache (linear_cache, activation_cache)
   return A, cache (A_prev, W, b, Z) ↳ Z

4. L_model_forward (X, parameter)
   A = X, len(parameters)/2 → how many layers.
   initial
   for 1~L-1 { A, cache = linear_activation_forward (A_prev, W, b, "relu") ↗ parameter
   caches.append (cache)
   L : AL, cache = linear_activation_forward (A, W, b, "sigmoid")
   caches.append (cache)

5. cost function

6. linear_backward (dZ, cache) → {A_prev, W, b, Z}
   return dA_prev, dW, db



7. linear_activation_backward (dA, cache, activation)
   ① linear_cache, activation_cache = cache
   ② if activation = "relu"
   ="sigmoid" different way to calculate dZ
   ↳ linear_backward (dZ, linear_cache)
   ③ return dA_prev, dW, db

8. L_model_backward (AL, Y, caches)