```
1. initialize_parameters_deep (layer dims)
        return parameters { W[1] ..., W[L]
                          | b[1] ... b[L]
2. linear_forward (A, W, b)
        return Z, cache          Z = np.dot (W,A) + b     forward
                                 Cache (A, W, b)          propagation
3. linear_activation_forward (A_prev, w, b, activation)
    if activation = "sigmoid":
        Z, linear_cache = linear_forward (A-prev, W, b)   Way
        A, activation_cache = sigmoid(Z)                   of
                                                           activation
    elif activation = "relu":
              =
              =
        cache (linear-cache, activation_cache)
                                   ↳ Z
        return A, cache (A-prev, w, b, 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")
                  } caches.append (cache)                       parameter
            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
```

# 2. 1.

$dZ^{[l]} = dA^{[l]} * g'(Z^{[l]})$

3. $dW^{[l]} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$

3. $db^{[l]} = \frac{1}{m} \sum dZ^{[l]}$

3. $dA^{[l-1]} = W^{[l]T} dZ^{[l]}$

caches layer l

↘ $dZ^{[l-1]}$

```
7. linear_activation_backward (dA, cache, activation)
    ① linear_cache, activation_cache = cache
    ② if activation = "relu"
                     = "sigmoid"  different way to calcalate dz
                                   ↳ linear_backward (dz, linear_cache)
    ③ return dA-prev, dw, db
8. L-model_backward (AL, Y, caches)
```