Progress Report

Kuei-Yueh Ko

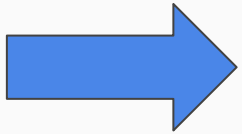
Progress Report

- Convolution function (2D)
  - Stride
  - padding
- **CellCnn**
  - Categorical cross entropy
  - Batch, mini-batch, & stochastic
  - conv1D

Batch, Mini-batch, and stochastic

x(1), y(1) x(2), y(2) ● ● ● ● ● ● ● x(n), y(n)

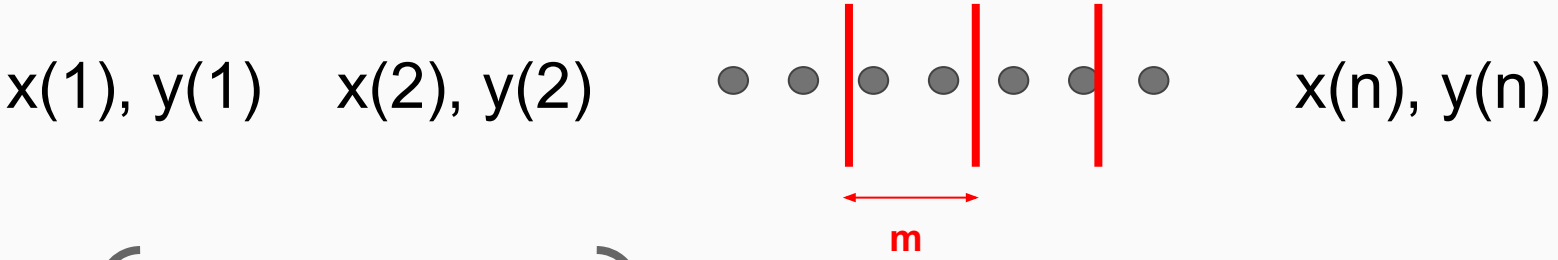
$$X = \begin{pmatrix} x(1) & x(2) & \bullet & \bullet & \bullet & x(n) \end{pmatrix}$$



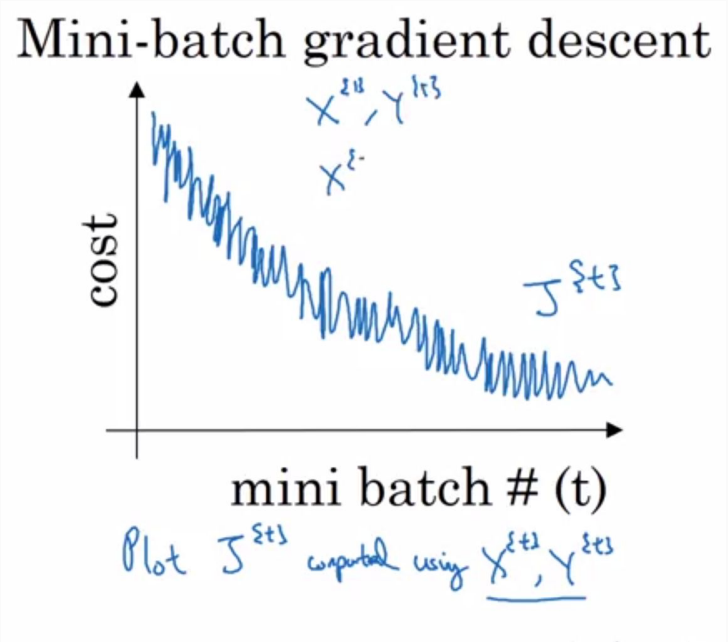
Forward and backward

Stochastic gradient descent

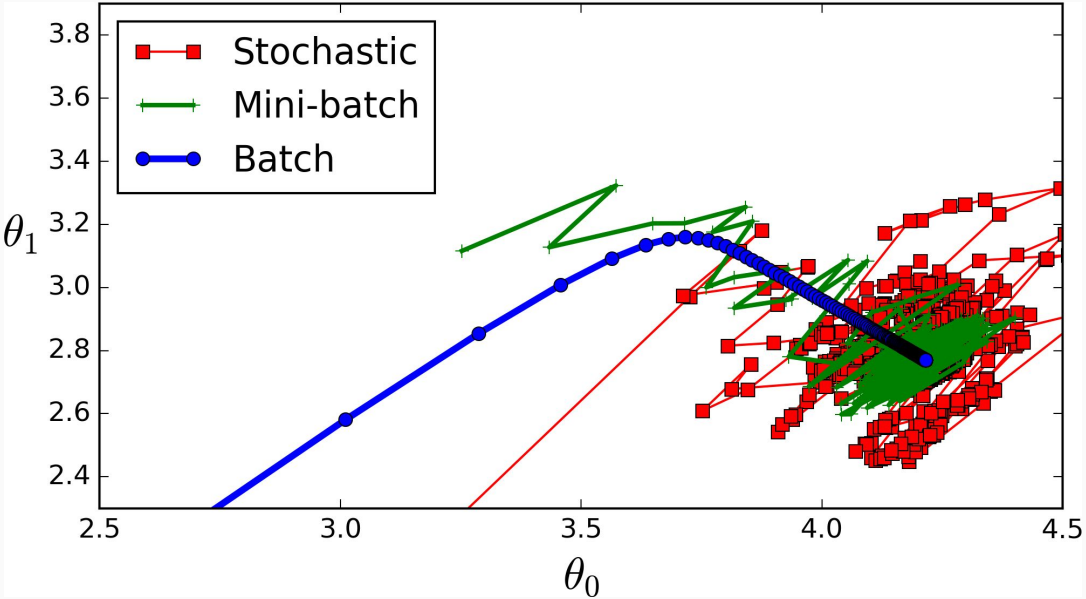
Mini-batch gradient descent



Batch --- Mini-Batch --- Stochastic

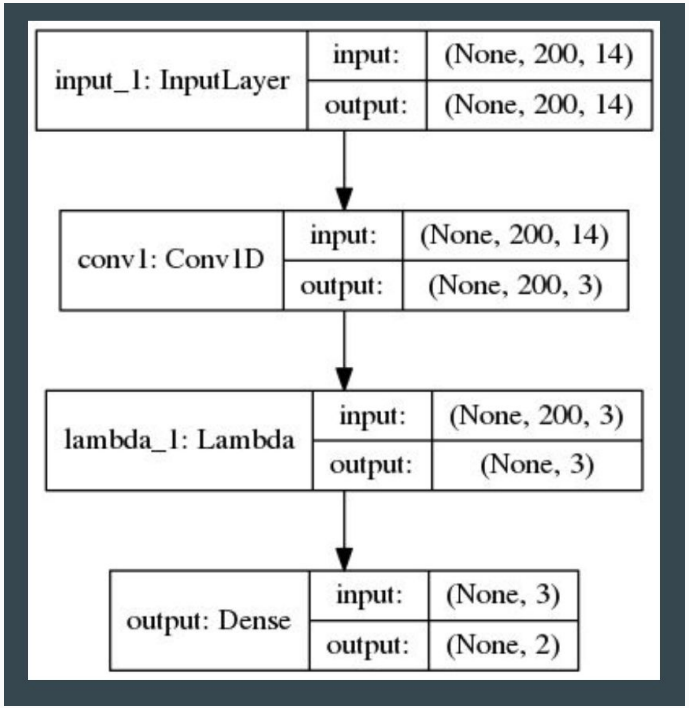


Coursera --- Deep Learning Course
(by Andrew Ng)



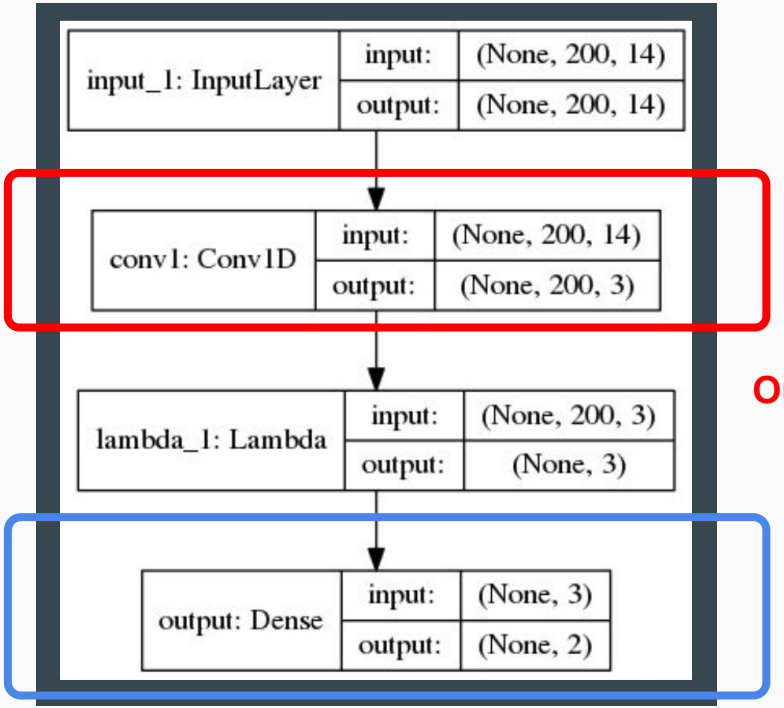
https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

1D Convolution


input_1: InputLayer
input:
(None, 200, 14)
output:
(None, 200, 14)
conv1: Conv1D
input:
(None, 200, 14)
output:
(None, 200, 3)
lambda_1: Lambda
input:
(None, 200, 3)
output:
(None, 3)
output: Dense
input:
(None, 3)
output:
(None, 2)

Weights[0]

Weights[1]



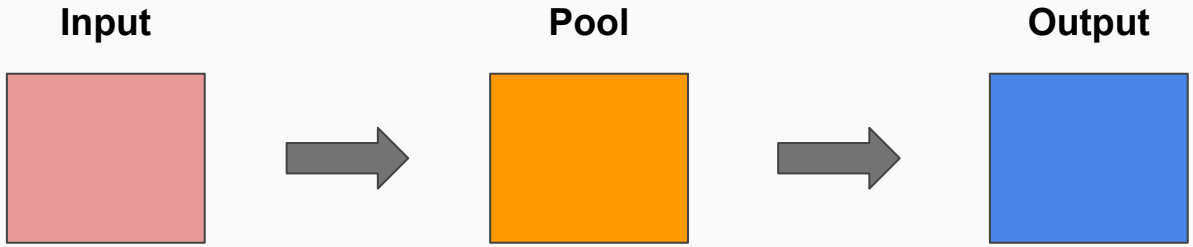
Output: length = # filters

Weights[2]

Weights[3]

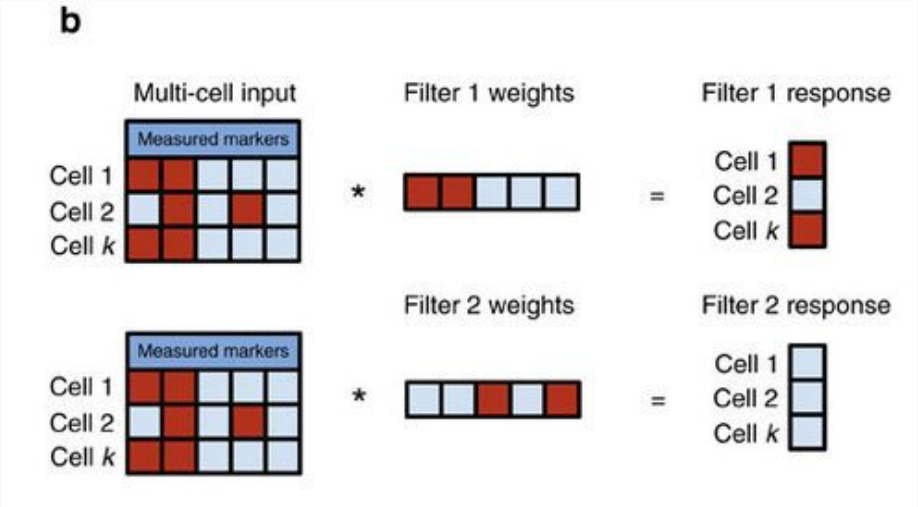
Output: length = # classes

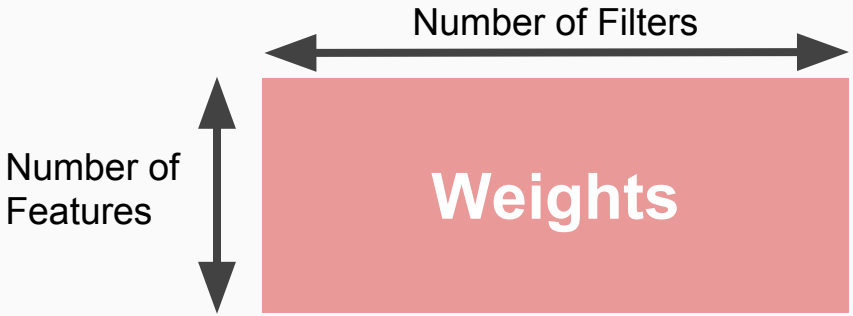
pooled = **MaxPool1D**(pool_size=n_pool)(convolution)




```
convolution = Conv1D(
        n_filters,
        1,
        activation='relu',
        kernel_regularizer=l1_l2(l1=l1, l2=l2),
        strides=1,
        padding='same',
        name='conv1'
    )(data_input)
```

```
    output = Dense(
        n_classes,
        activation='softmax',
        kernel_regularizer=l1_l2(l1=l1, l2=l2),
        name='output'
    )(pooled)
    #)(convolution)
```

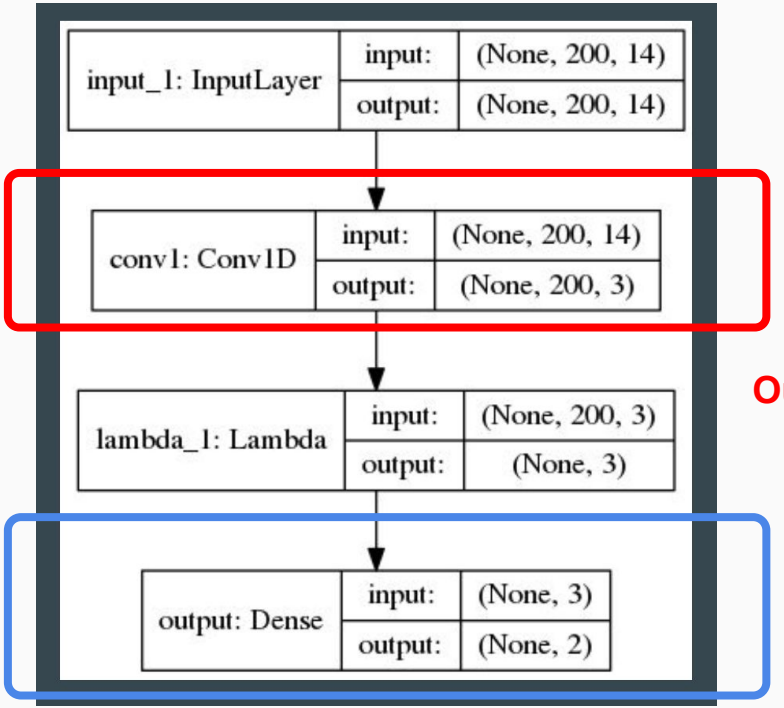




(b) Illustration of cell-filter response computations for individual cells. For instance, marker profiles of cell 1 and 3 exhibit perfect/no match with weights of filter 1/2 and therefore result in a high/low (red/blue) cell-filter response.

Weights[0]

Weights[1]



Output: length = # filters

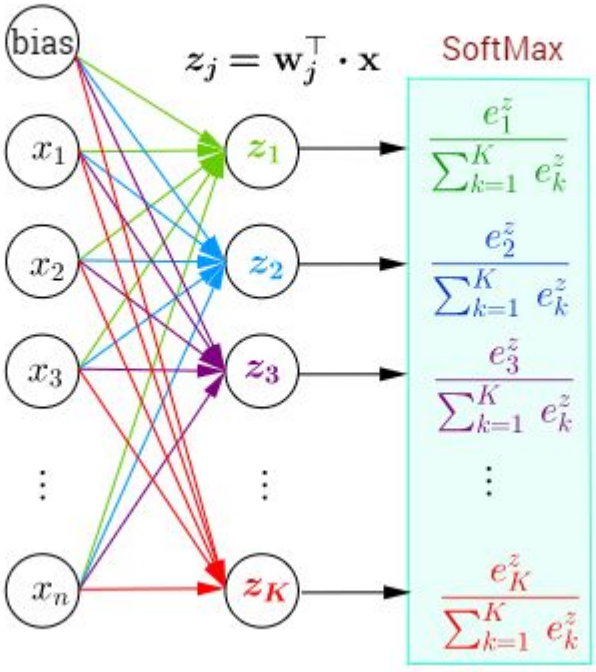
Weights[2]

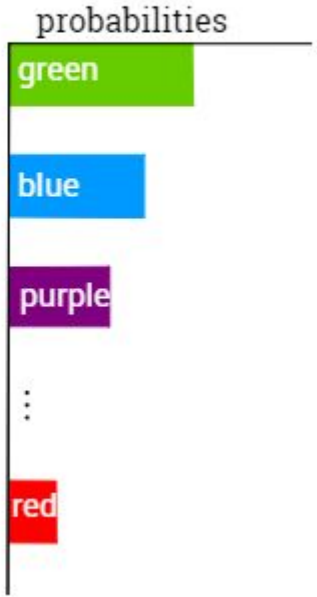
Weights[3]

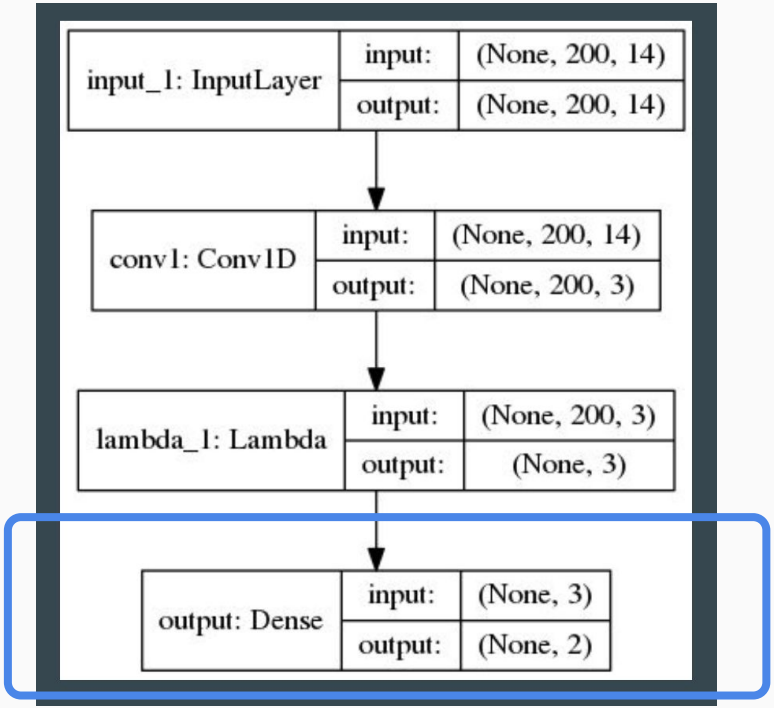
Output: length = # classes

Multi-Class Classification with NN and SoftMax Function

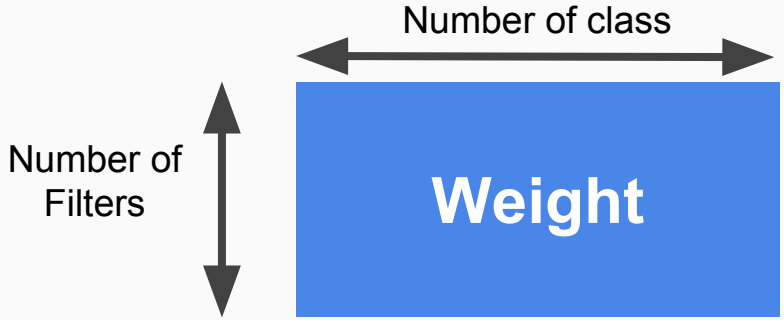
$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$





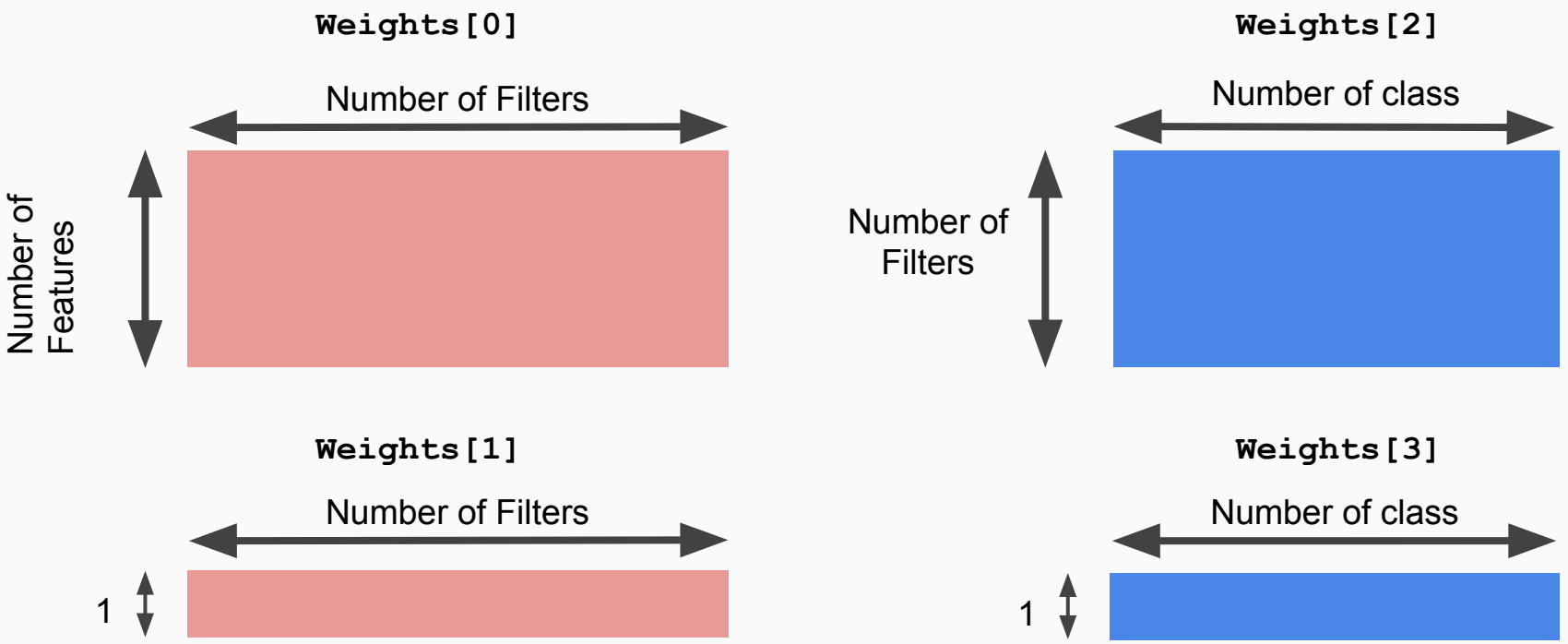


Output: length = # classes



Demonstration

Parameters


Weights[0]
Number of Filters
Number of Features
Weights[2]
Number of class
Number of Filters
Weights[1]
Number of Filters
1
Weights[3]
Number of class
1

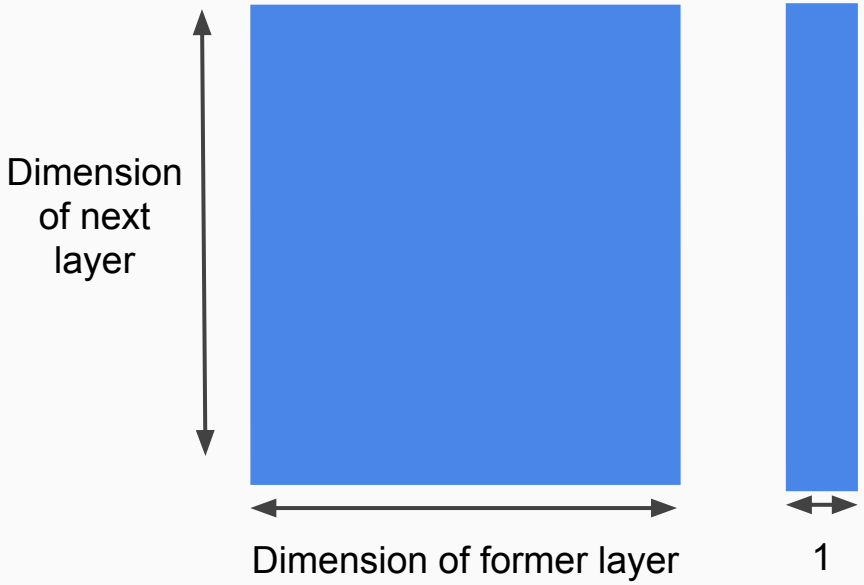
Guessing……

wx + b

Weight matrix

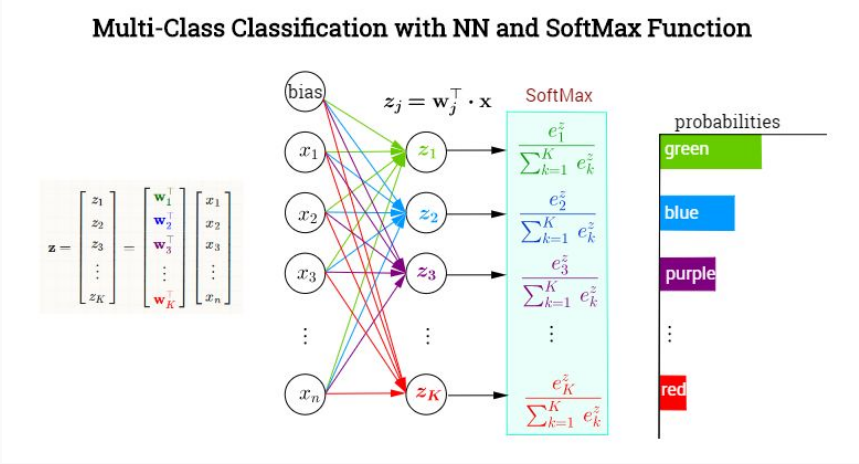
WX + b

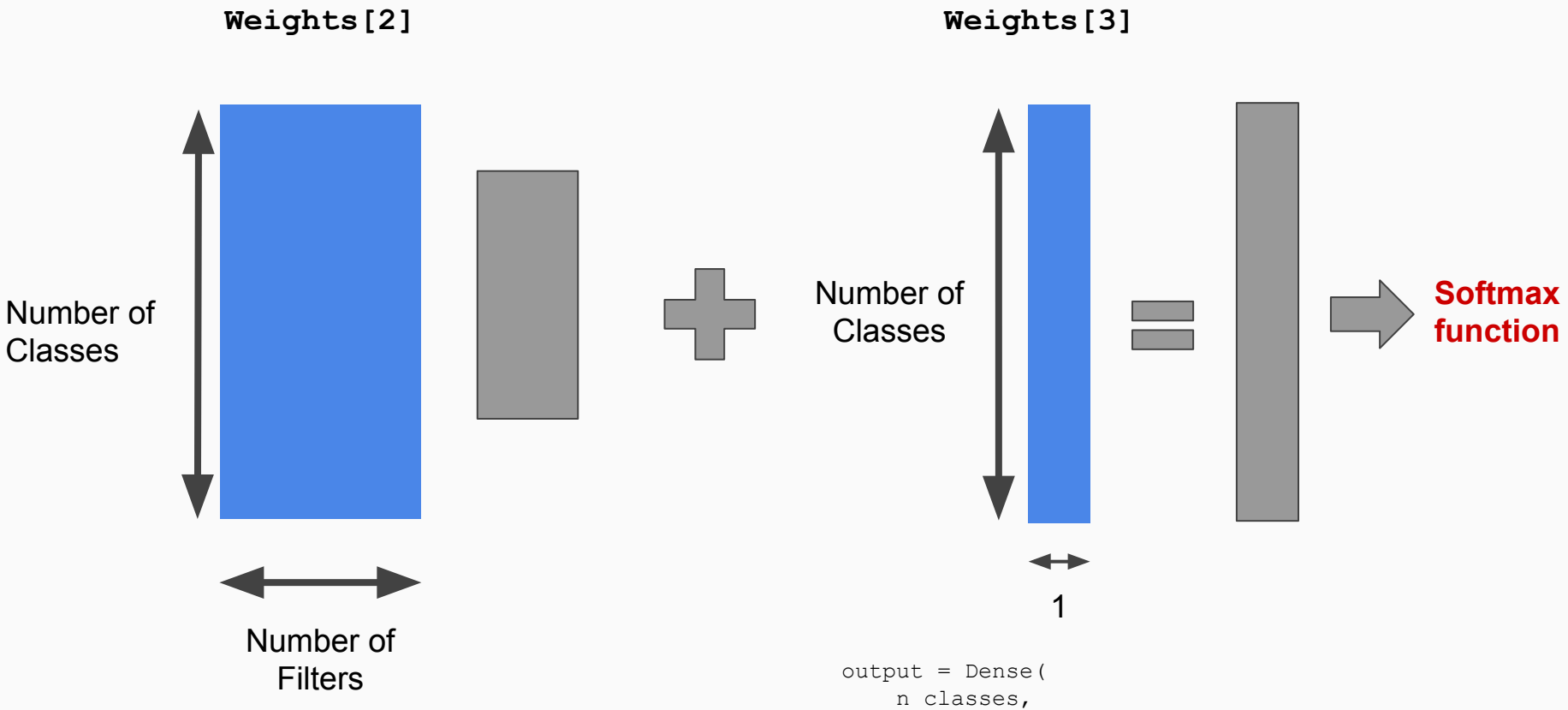
b = bias




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1, 2)              0
_________________________________________________________________
conv1 (Conv1D)               (None, 1, 5)              15
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 1, 5)              0
_________________________________________________________________
output (Dense)               (None, 1, 2)              12
=================================================================
Total params: 27
Trainable params: 27
Non-trainable params: 0
_________________________________________________________________
```

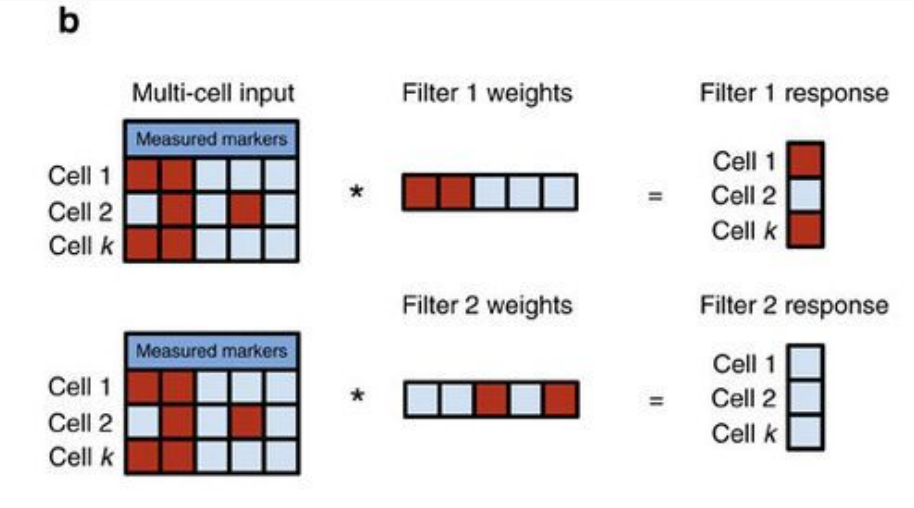
```
output = Dense(
    n_classes,
    activation='softmax',
    kernel_regularizer=l1_l2(l1=l1, l2=l2),
    name='output'
)(pooled)
#)(convolution)
```

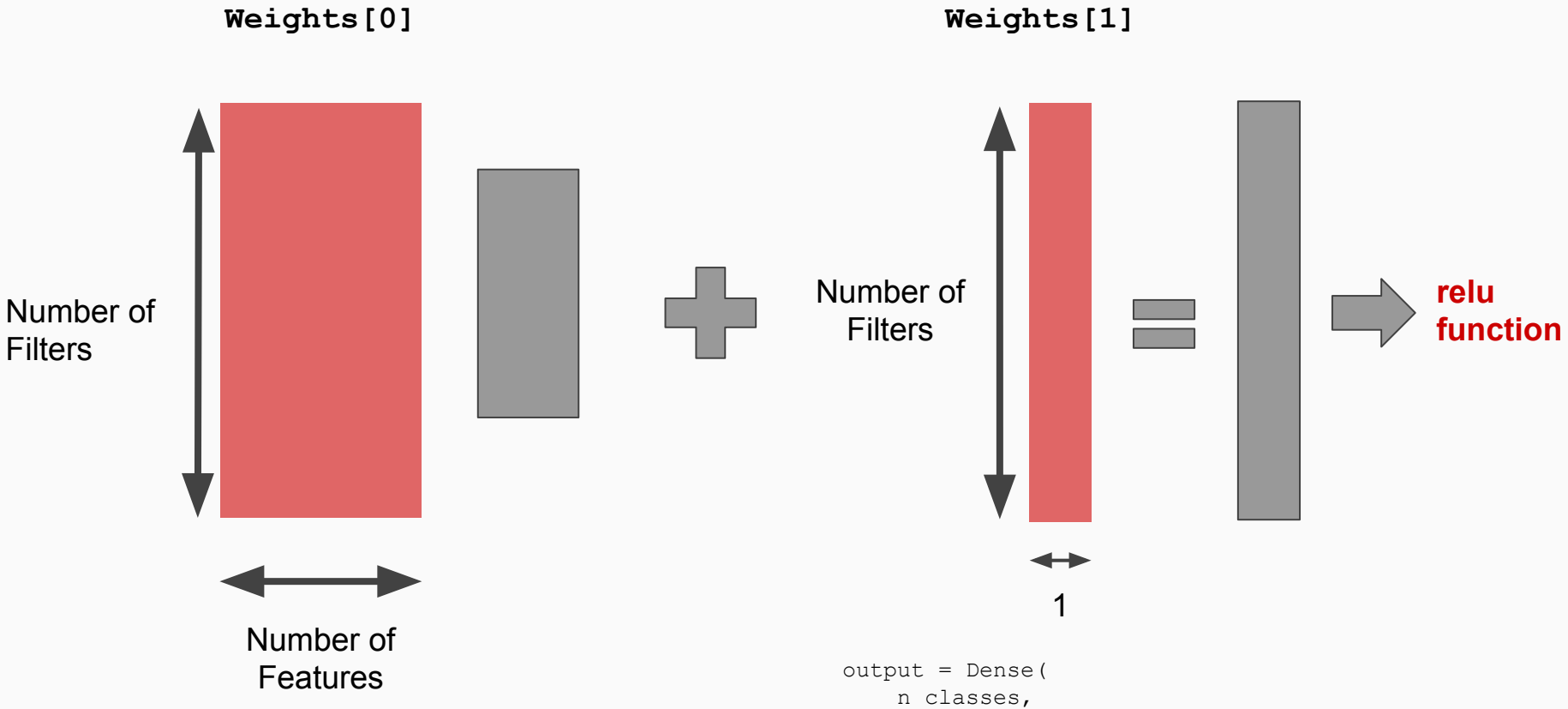





```
output = Dense(
    n_classes,
    activation='softmax',
    kernel_regularizer=l1_l2(l1=l1, l2=l2),
    name='output'
)(pooled)
#)(convolution)
```

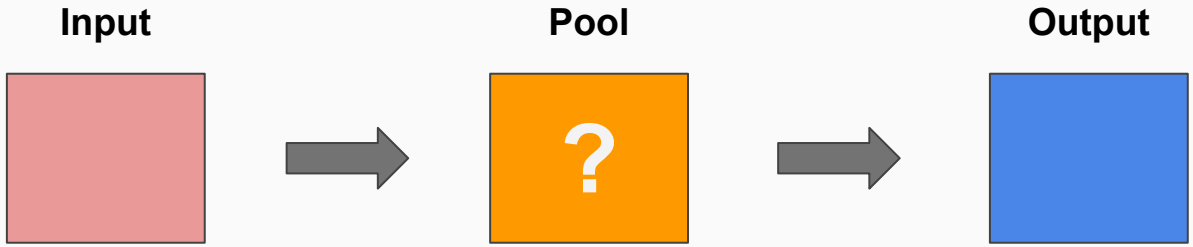
```
convolution = Conv1D(
        n_filters,
        1,
        activation='relu',
        kernel_regularizer=l1_l2(l1=l1, l2=l2),
        strides=1,
        padding='same',
        name='conv1'
    )(data_input)
```






```
output = Dense(
    n_classes,
    activation='softmax',
    kernel_regularizer=l1_l2(l1=l1, l2=l2),
    name='output'
)(pooled)
#)(convolution)
```

```
pooled = MaxPool1D(pool_size=n_pool)(convolution)
```




```
convolution = Conv1D(
        n_filters,
        1,
        activation='relu',
        kernel_regularizer=l1_l2(l1=l1, l2=l2),
        strides=1,
        padding='same',
        name='conv1'
    )(data_input)
```

```
    output = Dense(
        n_classes,
        activation='softmax',
        kernel_regularizer=l1_l2(l1=l1, l2=l2),
        name='output'
    )(pooled)
    #)(convolution)
```

Checking……