Guide to CNN’s

A basic guide to convolutional neural networks, stepping through some basic examples to help you understand them.

Inputs and Outputs

- Inputs:
  - Data which has patterns, in which a certain point can be determined by the points around it.
  - Images, videos, language processing, even playing GO.
- Outputs:
  - Some sort of prediction.
  - Faces, search query processing, sentence modelling.

Today I will be going through a toy image example.

Image Example: Setting up

- Lets assume we have a 5x5 greyscale image. It can be represented as follows:

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

Where

- 0 is White,
- 1 is Grey,
- 2 is Black.

Classified as $y^* = 1$

Image Example: Hyperparameters

Now to decide on a couple of things:

- Weights:
  - How many of them? K (Number of Filters)
  - What size? F (Size of the filters F x F)
  - How much do we move them by? S (Stride)
- Processing the image:
  - Do we to want to preprocess our image? P (Padding)

Image Example: Hyperparameters, K

Think about a neural network:

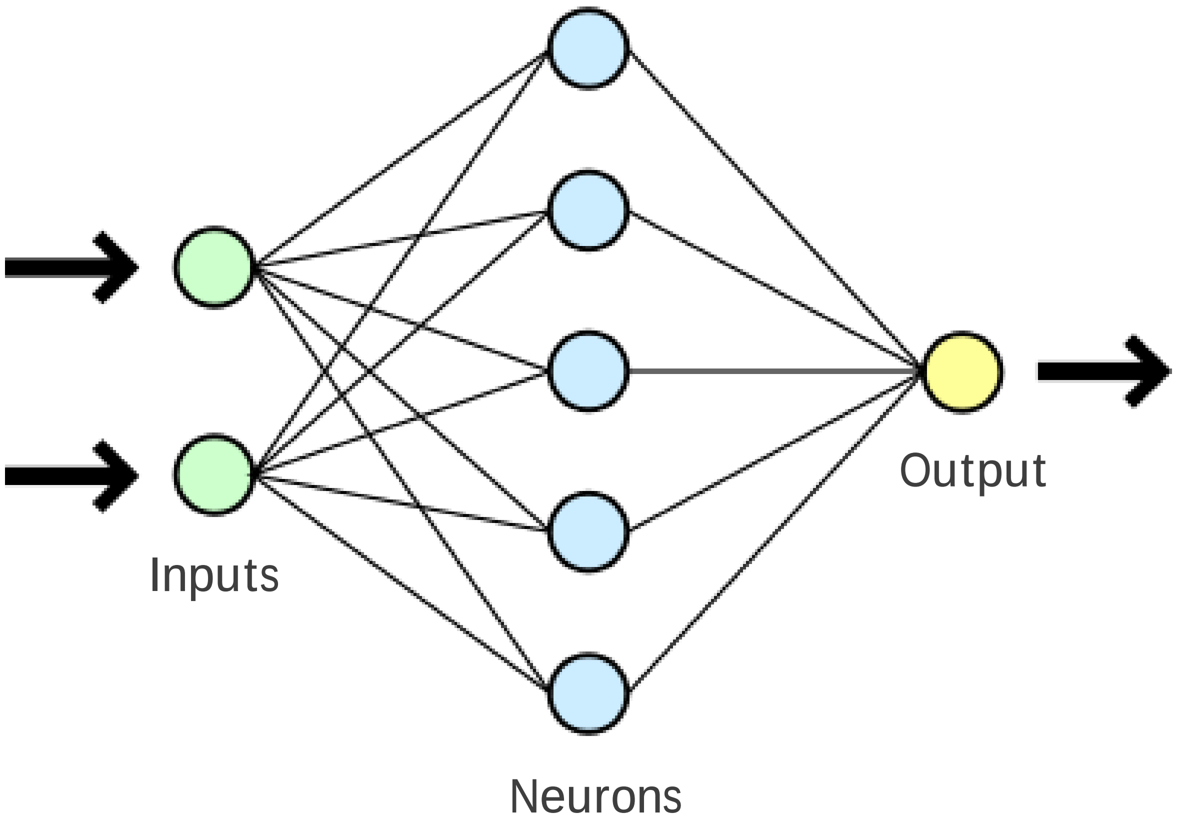
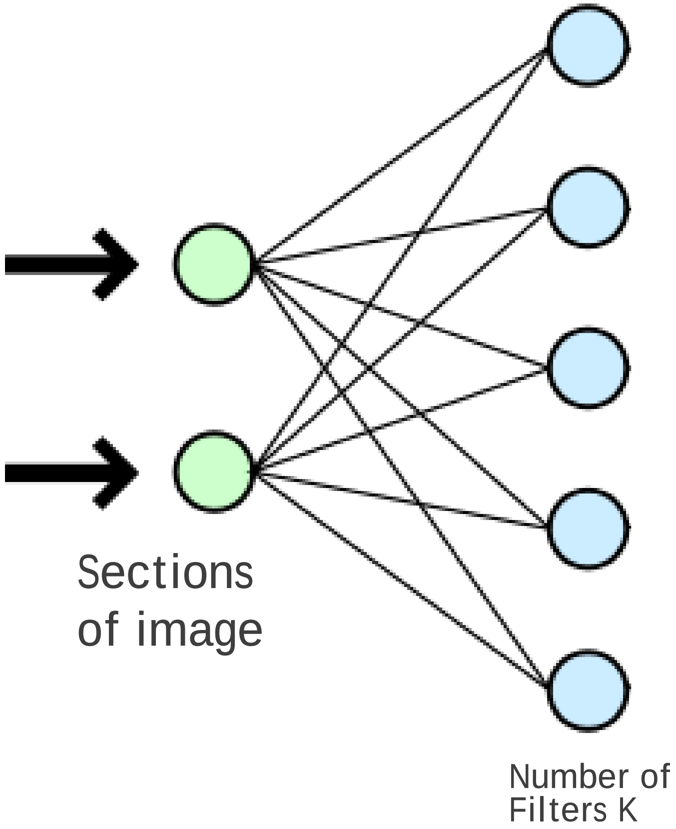


Image Example: Hyperparameters, K

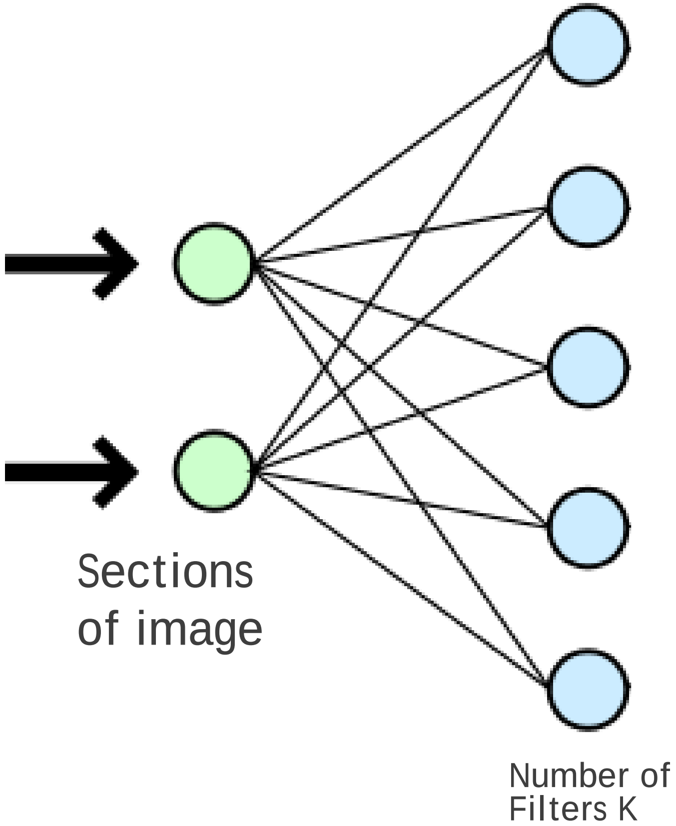
With a convolutional neural network:



We want each of these filters to learn a different aspect of the same section of the image.

Image Example: Hyperparameters, K

With a convolutional neural network:



In this example lets assume we have 2 filters.

F^1 and F^2

K = 2

Image Example: Hyperparameters, F

We must also decide on the size of these filters F x F. Usually 3 x 3 or 5 x 5.

$F^1 =$

θ^1_{00}	θ^1_{01}	θ^1_{02}
θ^1_{10}	θ^1_{11}	θ^1_{12}
θ^1_{20}	θ^1_{21}	θ^1_{22}

$F^2 =$

θ^2_{00}	θ^2_{01}	θ^2_{02}
θ^2_{10}	θ^2_{11}	θ^2_{12}
θ^2_{20}	θ^2_{21}	θ^2_{22}

$$\theta^K_{Row\ Column}$$

Image Example: Hyperparameters, F

Let's initialize as follows:

$F^1 =$

-1	0	-1
1	1	-1
-1	1	-1

$F^2 =$

-1	-1	0
0	1	1
-1	0	1

$$\theta^{K}_{Row\ Column}$$

Image Example: Hyperparameters, S

We must decide how we process the filters over our input, we do this by determining our step size or stride S.

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

S = 1

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

Image Example: Hyperparameters, S

If our S = 1

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

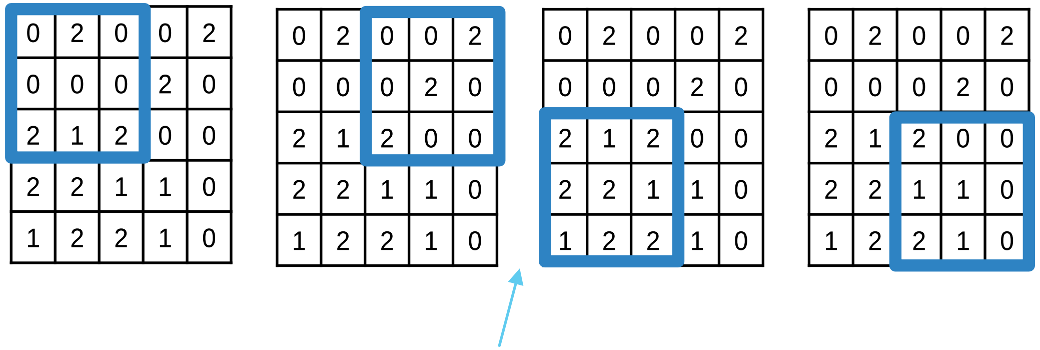
0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

Our output will be: 3 x 3

Image Example: Hyperparameters, S

If our S = 2



Notice here we jump down 2 rows as well!

Our output will be: 2 x 2

How do we calculate this ahead of time?

Image Example: Hyperparameters, K, F, S

The shape of our output is something we should know right?

$$\frac{(N-F)}{S}+1 \times \frac{(N-F)}{S}+1,$$

where N x N is the size of the input image.

N – Size of input
F – Size of filter
S - Stride

But what if our S was 3?

$$\frac{(5-3)}{3}+1 \times \frac{(5-3)}{3}+1 = \frac{2}{3}+1 \times \frac{2}{3}+1$$

Recall
N – 5
F – 3

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

0	2	0	0	2
0	0	0	2	0
2	1	2	0	0
2	2	1	1	0
1	2	2	1	0

Oops?

We don’t want to run off the image like this.

Image Example: Hyperparameters, P

To fix this we can use a preprocessing method called Padding, P. P = 1

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Our Original Image

Notice we just put an outside layer of 0's around our original image.

Image Example: Hyperparameters, P

P = 2

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	2	0	0
0	0	0	0	0	2	0	0	0
0	0	2	1	2	0	0	0	0
0	0	2	2	1	1	0	0	0
0	0	1	2	2	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Our Original Image

Notice we just put an additional outside layer of 0's around our original image.

Image Example: Hyperparameters, K,F,S,P

So the padding change our shape equation:

$$\frac{(N+2P-F)}{S}+1 \times \frac{(N+2P-F)}{S}+1,$$

where N x N is the size of the input image.

N – Size of input
F – Size of filter
P - Padding
S - Stride

Back to our example, what if our S was 3? Well if we chose P = 2

$$\frac{(5+4-3)}{3}+1 \times \frac{(5+4-3)}{3}+1 = \frac{6}{3}+1 \times \frac{6}{3}+1$$

Recall
N – 5
F – 3

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	2	0	0
0	0	0	0	0	2	0	0	0
0	0	2	1	2	0	0	0	0
0	0	2	2	1	1	0	0	0
0	0	1	2	2	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	2	0	0
0	0	0	0	0	2	0	0	0
0	0	2	1	2	0	0	0	0
0	0	2	2	1	1	0	0	0
0	0	1	2	2	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	2	0	0
0	0	0	0	0	2	0	0	0
0	0	2	1	2	0	0	0	0
0	0	2	2	1	1	0	0	0
0	0	1	2	2	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Image Example: Choosing our Hyperparameters

Lets use:

K = 2

F = 3

S = 2 = ε^1

P = 1

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

F^2 =

-1	-1	0
0	1	1
-1	0	1

So what is our output dimension?

Image Example: Choosing our Hyperparameters

Lets use:

K = 2

F = 3

S = 2 = ε^1

P = 1

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

F^2 =

-1	-1	0
0	1	1
-1	0	1

So what is our output dimension?

3 x 3

Image Example

Similarly to regular neural networks we can include a bias term.

However we need a bias term for each of our weights. b^1 and b^2 , which are just scalars.

Lets actually do some math. Lets start with b^1 = 1 and F_1 as follows

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1 = 1

$$z_{st}^k = b^k + \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \theta_{uv}^k \, x_{(s \cdot \varepsilon^1 + u)(t \cdot \varepsilon^1 + v)}$$

Image Example Annotated Convolution Equation

$$z_{st}^{k} = b^{k} + \sum_{u=0}^{m-1}\sum_{v=0}^{m-1} \theta_{uv}^{k}\, x_{(s\cdot\varepsilon^1+u)(t\cdot\varepsilon^1+v)}$$

The output is a matrix with element

$z_{00}^{k} = b^{k} + \sum_{u=0}^{3-1}\sum_{v=0}^{3-1} \theta_{uv}^{k}\, x_{(0\cdot2+u)(0\cdot2+v)}$

Notice that v is the current column and u is the current row of the filter which we are applying to that specific section of x

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1 = 1

z^1=

-1		

Image Example Annotated Convolution Equation

$$z_{st}^{k} = b^{k} + \sum_{u=0}^{m-1}\sum_{v=0}^{m-1} \theta_{uv}^{k}\, x_{(s\cdot\varepsilon^{1}+u)(t\cdot\varepsilon^{1}+v)}$$

The output is a matrix with element say we are looking at

$z_{11}^{k} = b^{k} + \sum_{u=0}^{3-1}\sum_{v=0}^{3-1} \theta_{uv}^{k}\, x_{(1\cdot 2+u)(1\cdot 2+v)}$

Recall our stride ε^{1}=2 so we are applying the filter over

THIS part of the input

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^{1} =

-1	0	-1
1	1	-1
-1	1	-1

b^{1} = 1

z^{1}=

-1		
	0	

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1 = 1

0x-1	0x0	0x-1
0x1	0x1	2x-1
0x-1	0x1	0x-1

= -2 + b^1 = -1

z^1=

-1		

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

$b^1 = 1$

0x-1	0x0	0x-1
2x1	0x1	0x-1
0x-1	0x1	2x-1

$= 0 + b^1 = 1$

z^1=

-1	1	

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1= 1

0x-1	0x0	0x-1
0x1	2x1	0x-1
2x-1	0x1	0x-1

= 0 + b^1 = 1

z^1=

-1	1	1

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

$b^1 = 1$

0x-1	0x0	0x-1
0x1	2x1	1x-1
0x-1	2x1	2x-1

$= 1 + b^1 = 2$

z^1=

-1	1	1
2		

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

$F^1 =$

-1	0	-1
1	1	-1
-1	1	-1

$b^1 = 1$

0x-1	0x0	2x-1
1x1	2x1	0x-1
2x-1	1x1	1x-1

$= -1 + b^1 = 0$

$z^1 =$

-1	1	1
2	0	

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1 = 1

2x-1	0x0	0x-1
0x1	0x1	0x-1
1x-1	0x1	0x-1

= -3 + b^1 = -2

z^1=

-1	1	1
2	0	-2

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

b^1 = 1

0x-1	2x0	2x-1
0x1	1x1	2x-1
0x-1	0x1	0x-1

= -3 + b^1 = -2

z^1=

-1	1	1
2	0	-2
-2		

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

$b^1 = 1$

2x-1	1x0	1x-1
2x1	2x1	1x-1
0x-1	0x1	0x-1

$= 0 + b^1 = 1$

z^1=

-1	1	1
2	0	-2
-2	1	

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^1 =

-1	0	-1
1	1	-1
-1	1	-1

$b^1 = 1$

1x-1	0x0	0x-1
1x1	0x1	0x-1
0x-1	0x1	0x-1

$= 0 + b^1 = 1$

z^1=

-1	1	1
2	0	-2
-2	1	1

So this is our output for F^1

Image Example

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

F^2=

-1	-1	0
0	1	1
-1	0	1

$b^2 = 0$

z^2=

2	2	0
5	1	-3
1	0	-1

So do this yourself to get our output for F^2

Image Example: ReLU

z^1=

-1	1	1
2	0	-2
-2	1	1

z^2=

2	2	0
5	1	-3
1	0	-1

Similarly to NN we can apply some sort of transformation function to this output.

$ReLU_{ab}(Output) = \max\{0, Output_{ab}\}$ for a and b indices of the matrix

It doesn't seem intuitive but works better than sigmoid function in the case of deep learning.

So our outputs are:

Rz^1=

0	1	1
2	0	0
0	1	1

Rz^2=

2	2	0
5	1	0
1	0	0

ReLU – If it's negative set the value to 0, if its positive leave it alone. (Has no parameters)

Image Example: Max Pooling

Rz^1=

0	1	1
2	0	0
0	1	1

Rz^2=

2	2	0
5	1	0
1	0	0

Think of this as another hidden layer in the neural network, which has no weights and is exclusively there to simplify what the features are telling us.

$$mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{n-1}\}_{c=0}^{n-1}$$

$n \times n$ is our filter size

Image Example: Max Pooling Annotation

$$mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{n-1}\}_{c=0}^{n-1}$$

$n \times n$ is our pooling size

u and v are the positions in out max pooling output

c and d keep incrementing through out our pooling sections

$$\max\{\{Rz_{(0\cdot 2+c)(0\cdot 2+d)}\}_{d=0}^{2-1}\}_{c=0}^{2-1} = \max\{\mathrm{Rz}_{00}, \mathrm{Rz}_{01}, \mathrm{Rz}_{10}, \mathrm{Rz}_{11}\} = \max\{1, 2, \text{-}4, 5\} = mz_{00}^{k}$$

1	2	6	7
-4	5	0	1
1	3	1	1
3	3	2	1

5	

Image Example: Max Pooling Annotation

$$mz_{uv}^{k} = \max\{\{Rz^{k}_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}\}_{d=0}^{n-1}\}_{c=0}^{n-1}$$

$n \times n$ is our pooling size

u and v are the positions in out max pooling output

c and d keep incrementing through out our pooling sections

$$\max\{\{Rz_{(0\cdot 2+c)(1\cdot 2+d)}\}_{d=0}^{2-1}\}_{c=0}^{2-1} = \max\{\mathrm{Rz}_{02}, \mathrm{Rz}_{03}, \mathrm{Rz}_{12}, \mathrm{Rz}_{13}\} = \max\{6, 7, 0, 1\} = mz_{01}^{k}$$

1	2	6	7
-4	5	0	1
1	3	1	1
3	3	2	1

5	7

Image Example: Max Pooling

1	2	6	7
-4	5	0	1
1	3	1	1
3	3	2	1

5	

Assume we have the 4x4 image matrix O above.

We take the max of a certain 'pool' of our data, so if our pool is 2 x 2 and we have pool stride of $\varepsilon^2 = 2$.

$$
\begin{aligned}
mz_{00}(O) &= \max\{O_{00}, O_{01}, O_{10}, O_{11}\} \\
mz_{01}(O) &= \max\{O_{02}, O_{03}, O_{12}, O_{13}\} \\
mz_{10}(O) &= \max\{O_{20}, O_{21}, O_{30}, O_{31}\} \\
mz_{11}(O) &= \max\{O_{22}, O_{23}, O_{32}, O_{33}\}
\end{aligned}
$$

$$
mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{2-1}\}_{c=0}^{2-1}
$$

Image Example: Max Pooling

1	2	6	7
-4	5	0	1
1	3	1	1
3	3	2	1

5	7

Assume we have the 4x4 image matrix O above.

We take the max of a certain ‘pool’ of our data, so if our pool is 2 x 2 and we have pool stride of $\varepsilon^2 = 2$.

$$\begin{aligned}
mz_{00}(\text{O}) &= \max\{O_{00}, O_{01}, O_{10}, O_{11}\} \\
mz_{01}(\text{O}) &= \max\{O_{02}, O_{03}, O_{12}, O_{13}\} \\
mz_{10}(\text{O}) &= \max\{O_{20}, O_{21}, O_{30}, O_{31}\} \\
mz_{11}(\text{O}) &= \max\{O_{22}, O_{23}, O_{32}, O_{33}\}
\end{aligned}$$

$$mz_{uv}^{k} = \max\{\{Rz^{k}_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}\}_{d=0}^{2-1}\}_{c=0}^{2-1}$$

Image Example: Max Pooling

1	2	6	7
-4	5	0	1
$\boxed{1}$	$\boxed{3}$	1	1
$\boxed{3}$	$\boxed{3}$	2	1

$\Rightarrow$

5	7
$\boxed{3}$	

Assume we have the 4x4 image matrix O above.

We take the max of a certain 'pool' of our data, so if our pool is 2 x 2 and we have pool stride of $\varepsilon^2 = 2$.

$$
\begin{aligned}
mz_{00}(O) &= \max\{O_{00}, O_{01}, O_{10}, O_{11}\} \\
mz_{01}(O) &= \max\{O_{02}, O_{03}, O_{12}, O_{13}\} \\
mz_{10}(O) &= \max\{O_{20}, O_{21}, O_{30}, O_{31}\} \\
mz_{11}(O) &= \max\{O_{22}, O_{23}, O_{32}, O_{33}\}
\end{aligned}
$$

$$mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{2-1}\}_{c=0}^{2-1}$$

Image Example: Max Pooling

1	2	6	7
-4	5	0	1
1	3	1	1
3	3	2	1

5	7
3	2

Assume we have the 4x4 image matrix O above.

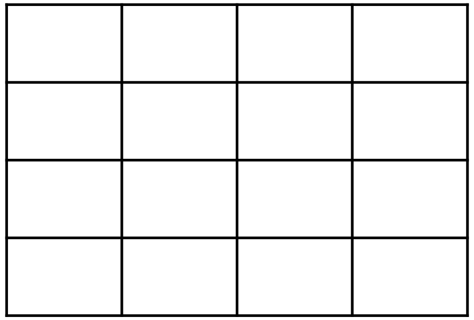
We take the max of a certain 'pool' of our data, so if our pool is 2 x 2 and we have pool stride of $\varepsilon^2 = 2$.

$$
\begin{aligned}
mz_{00}(O) &= \max\{O_{00}, O_{01}, O_{10}, O_{11}\} \\
mz_{01}(O) &= \max\{O_{02}, O_{03}, O_{12}, O_{13}\} \\
mz_{10}(O) &= \max\{O_{20}, O_{21}, O_{30}, O_{31}\} \\
mz_{11}(O) &= \max\{O_{22}, O_{23}, O_{32}, O_{33}\}
\end{aligned}
$$

$$mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{2-1}\}_{c=0}^{2-1}$$

Image Example: Max Pooling

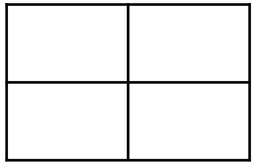
Size of the output of Max Pooling?



$M \ x \ M$

Pool size: $f \ x \ f$

Stride: s



$$\frac{(M - f)}{s} + 1 \times \frac{(M - f)}{s} + 1$$

Usually people choose a standard pool size of n x n and a stride size of n so that there is no overlapping.

Image Example

Rz^1=

0	1	1
2	0	0
0	1	1

Rz^2=

2	2	0
5	1	0
1	0	0

For simplicity in this toy example lets use 3 x 3 max sampling.

$mz^1 = \boxed{2}$

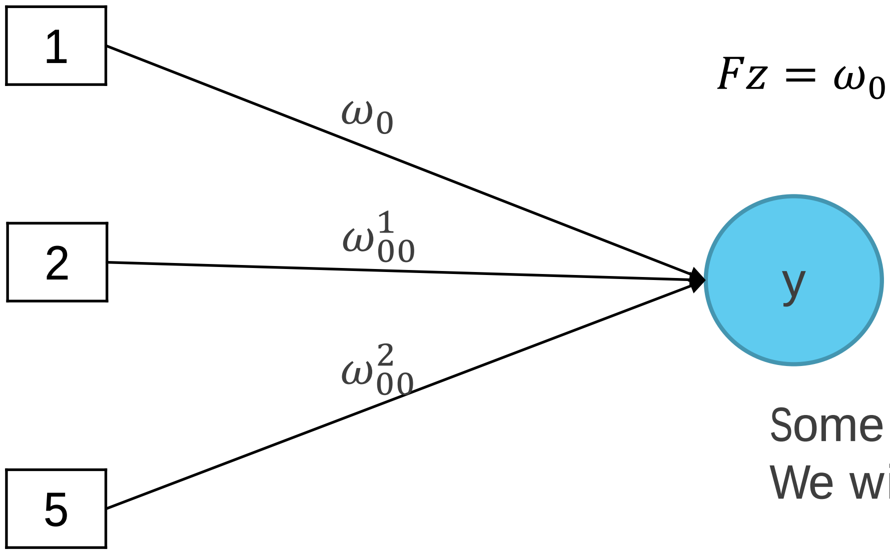
$mz^2 = \boxed{5}$

As I mentioned you wouldn't typically take such a 'big' (Relative to the input) max pooling as you loose out on too much, usually 2x2 with a step size of 2 is a good one, but it depends on how much you are willing to lose of the image compare to training time.

Image Example: Fully Connected

For the final step we <u>must</u> have a fully connected neural network layer.

$mz^1 = \boxed{2}$ $mz^2 = \boxed{5}$



$$Fz = \omega_0 + \sum_{k=1}^{K} \sum_{e=0}^{m-1} \sum_{f=0}^{m-1} \omega^k_{ef} mz^k_{ef}$$

Some activation function,
We will use the sigmoid function

$$y = \sigma(x) = \frac{1}{1 + e^{-x}}$$

HYPATHETICAL SCERNARIO

If our mz^k output was a 2 x 2 matrix

$$mz^k = \begin{bmatrix} mz^1_{00} & mz^1_{01} \\ mz^1_{10} & mz^1_{11} \end{bmatrix}$$

$$\omega^k_{00} = \begin{bmatrix} \omega^1_{00} & \omega^1_{01} \\ \omega^1_{10} & \omega^1_{11} \end{bmatrix}$$

Image Example: Fully Connected Annotated

$$Fz = \omega_0 + \sum_{k=1}^{K} \sum_{e=0}^{m-1} \sum_{f=0}^{m-1} \omega_{ef}^{k} mz_{ef}^{k}$$

Works in a similar way as the filter in the convolutional layer works.

(The homework will not use this method as you convert your inputs to the IP layer into a vector, instead of leaving them in matrix form which is what this equation is doing.)

HYPATHETICAL SCERNARIO

If our mz^k output was a 2 x 2 matrix

$$mz^k = \begin{bmatrix} mz_{00}^1 & mz_{01}^1 \\ mz_{10}^1 & mz_{11}^1 \end{bmatrix}$$

$$\omega_{00}^k = \begin{bmatrix} \omega_{00}^1 & \omega_{01}^1 \\ \omega_{10}^1 & \omega_{11}^1 \end{bmatrix}$$

Image Example: Fully Connected Annotated

$$Fz = \omega_0 + \sum_{k=1}^{K} \sum_{e=0}^{m-1} \sum_{f=0}^{m-1} \omega_{ef}^{k} mz_{ef}^{k}$$

The K is the number of filters, essentially we are making a weight matrix for each of these inputs which this multiplies (element wise) with these inputs and sums them over all filters.

(Plus the bias term too)

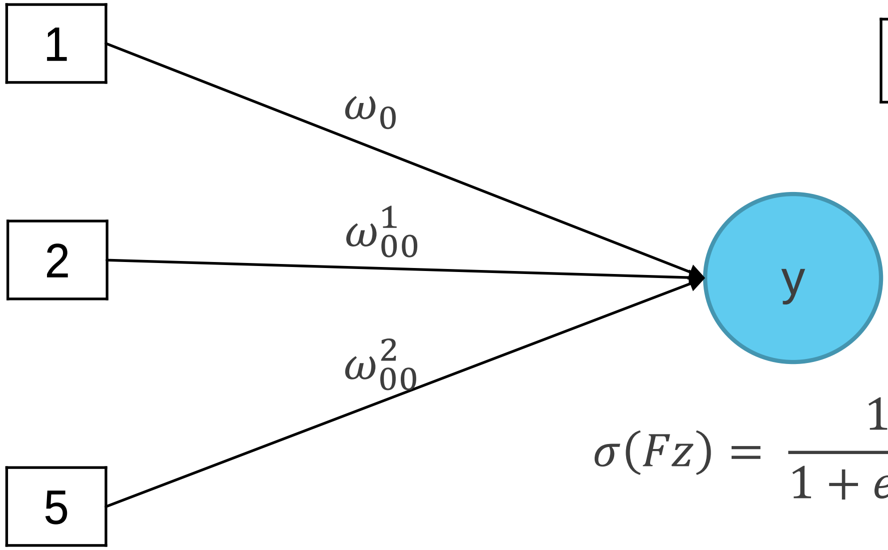
HYPATHETICAL SCERNARIO

If our mz^k output was a
2 x 2 matrix

$$mz^k = \begin{array}{|c|c|} \hline mz_{00}^{1} & mz_{01}^{1} \\ \hline mz_{10}^{1} & mz_{11}^{1} \\ \hline \end{array}$$

$$\omega_{00}^{k} = \begin{array}{|c|c|} \hline \omega_{00}^{1} & \omega_{01}^{1} \\ \hline \omega_{10}^{1} & \omega_{11}^{1} \\ \hline \end{array}$$

Image Example: Fully Connected



ω_0 =0.3	ω_{00}^1=-1	ω_{00}^2=0.2

$$\sigma(Fz) = \frac{1}{1+e^{-Fz}}$$

$\sigma(Fz)$

$= \sigma(\omega_0 + 2\omega_{00}^1 + 5\omega_{00}^2)$

$= \sigma(0.3 - 2 + 1)$

= 0.332

With some cost function

$J(\omega, \theta)$ which compares our predicted output with the actual output

Image Example: Visualize our CNN

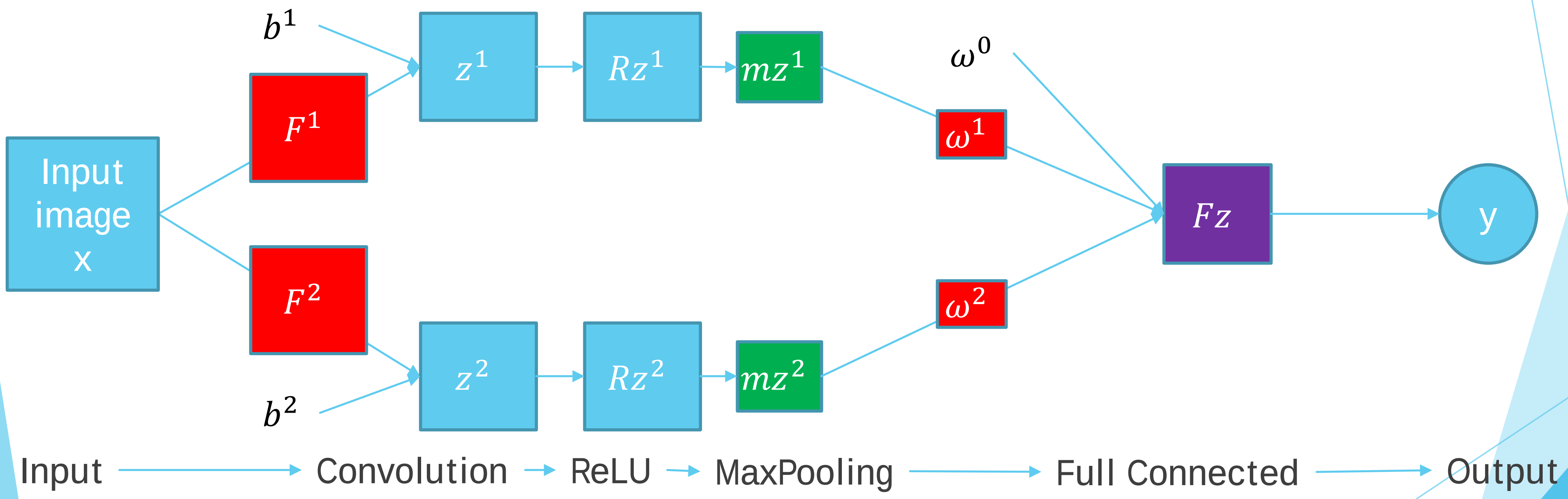


Image Example

Good work!

You are now done with the ‘Easy’ part!

I wasn’t kidding…

Image Example: Backpropagation

List all of the steps:

- Cost Function, J is $y^{*}\log(y) + (1 - y^{*})\log(1 - y)$
- $y = \frac{1}{1+e^{-Fz}}$
- $Fz = \omega_0 + \sum_{k=1}^{K} \sum_{e=0}^{m-1} \sum_{f=0}^{m-1} \omega_{ef}^{k} mz_{ef}^{k}$
- $mz_{uv}^{k} = \max\{\{Rz_{(u\cdot\varepsilon^2+c)(v\cdot\varepsilon^2+d)}^{k}\}_{d=0}^{n-1}\}_{c=0}^{n-1}$ n x n is the size of your pool and ε^2 is the pool stride size.
- $Rz_{st}^{k} = \max\{0, z_{st}^{k}\}$
- $z_{st}^{k} = \theta_0 + \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \theta_{uv}^{k}\, x_{(s\cdot\varepsilon^1+u)(t\cdot\varepsilon^1+v)}$ m x m is the size of your filter and ε^1 is your stride size.
- x is your M x M input matrix

Image Example: Backpropagation

$$\frac{dJ}{d\theta_{ab}^k} = \frac{dJ}{dy} \times \frac{dy}{d\theta_{ab}^k},$$

$$\frac{dy}{d\theta_{ab}^k} = \frac{dy}{dFz} \times \frac{dFz}{d\theta_{ab}^k},$$

$$\frac{dFz}{d\theta_{ab}^k} = \frac{dFz}{dmz_{ef}^k} \times \frac{dmz_{ef}^k}{d\theta_{ab}^k},$$

$$\frac{dmz_{ef}^k}{d\theta_{ab}^k} = \frac{dmz_{ef}^k}{dRz^k} \times \frac{dRz^k}{d\theta_{ab}^k},$$

$$\frac{dRz^k}{d\theta_{ab}^k} = \frac{dRz^k}{dz^k} \times \frac{dz^k}{d\theta_{ab}^k},$$

$$\frac{dz^k}{d\theta_{ab}^k} = x_{(s\cdot\varepsilon^1+a)(t\cdot\varepsilon^1+b)}$$

$$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*)\frac{1}{y-1}$$

$$\frac{dy}{dFz} = \sigma(Fz)(1 - \sigma(Fz))$$

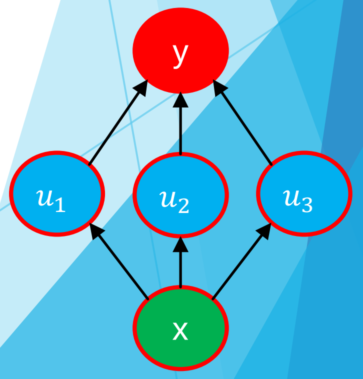
this one is a little awkward $\frac{dFz}{d\theta_{ab}^k} = \sum_{e=0}^{m-1}\sum_{f=0}^{m-1} \omega_{ef}^k \frac{dmz_{ef}^k}{d\theta_{ab}^k}$

$$\frac{dmz^k}{dRz^k} = \begin{cases} 1 \text{ if } Rz^k \text{ was the max} \\ 0 \text{ otherwise} \end{cases}$$

$$\frac{dRz^k}{dz^k} = \begin{cases} 1 \text{ if } z^k > 0 \\ 0 \text{ otherwise} \end{cases}$$

$\Big\{$ Where s and t are brought from the positions of z^k

Recall from lectures that this is what happens when you get a network which looks like this:



$$\frac{dy}{dx} = \sum_{i=1}^{3} \frac{dy}{du_i}\frac{du_i}{dx}$$

Image Example: Backpropagation

Lets calculate: $\frac{dJ}{d\theta_{11}^{1}}$ and update it using SGD.

$$\frac{dJ}{d\theta_{ab}^{k}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)(1-\ \sigma(Fz)) \times \sum_{e=0}^{m-1}\sum_{f=0}^{m-1}\omega_{ef}^{k}\ \frac{dmz_{ef}^{k}}{d\theta_{ab}^{k}}$$

$\frac{dJ}{dy}$ $\frac{dy}{dFz}$ $\frac{dFz}{dmz}$

Becomes:

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)(1-\ \sigma(Fz)) \times \sum_{e=0}^{m-1}\sum_{f=0}^{m-1}\omega_{ef}^{1}\ \frac{dmz_{ef}^{1}}{d\theta_{11}^{1}}$$

Image Example: Backpropagation

$$\frac{dmz^1_{ef}}{d\theta^1_{11}} = \frac{dmz^1_{ef}}{dRz^1} \times \frac{dRz^1}{d\theta^1_{11}}, \quad \frac{dmz^1}{dRz^1} = \begin{cases} 1 \text{ if } Rz^k \text{ was the max} \\ 0 \text{ otherwise} \end{cases}$$

$$\frac{dRz^1}{d\theta^1_{11}} = \frac{dRz^1}{dz^1} \times \frac{dz^1}{d\theta^1_{11}}, \quad \frac{dRz^1}{dz^1} = \begin{cases} 1 \text{ if } z^k > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$\frac{dz^1}{d\theta^1_{11}} = x_{(s \cdot \varepsilon^1 + 1)(t \cdot \varepsilon^1 + 1)}$$

We'll have this chain for:

$$\frac{dmz^1_{ef}}{d\theta^1_{11}} = \frac{dmz^1_{ef}}{dRz^1} \times \frac{dRz^1}{dz^1} \times \frac{dz^1}{d\theta^1_{11}}$$

So if any of these are 0 then we can ignore the rest of them and our $\frac{dmz^1_{ef}}{d\theta^1_{11}} = 0$

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

$Rz^1 =$

0	1	1
2	0	0
0	1	1

$mz^1 = \boxed{2}$

Image Example: Backpropagation

$$\frac{dz^1}{d\theta^1_{11}} = x_{(s\cdot\varepsilon^1+1)(t\cdot\varepsilon^1+1)}$$

Means we would be looking at these segments:

$F^1 =$

θ^1_{00}	θ^1_{01}	θ^1_{02}
θ^1_{10}	θ^1_{11}	θ^1_{12}
θ^1_{20}	θ^1_{21}	θ^1_{22}

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Where as if we were looking at θ^1_{20}

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Image Example: Backpropagation Annotated

$$\frac{dz^1}{d\theta^1_{11}} = x_{(s\cdot\varepsilon^1+1)(t\cdot\varepsilon^1+1)}$$

$$F^1 = \begin{bmatrix} \theta^1_{00} & \theta^1_{01} & \theta^1_{02} \\ \theta^1_{10} & \theta^1_{11} & \theta^1_{12} \\ \theta^1_{20} & \theta^1_{21} & \theta^1_{22} \end{bmatrix}$$

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	$\boxed{0}$	2	$\boxed{0}$	0	$\boxed{2}$	0
0	0	0	0	2	0	0
0	$\boxed{2}$	1	$\boxed{2}$	0	$\boxed{0}$	0
0	2	2	1	1	0	0
0	$\boxed{1}$	2	$\boxed{2}$	1	$\boxed{0}$	0
0	0	0	0	0	0	0

It's hard to see but each of the different coloured squares are where the filter was applied and the black squares on the image below are where the θ^1_{11} are being multiplied in each of these squares

Image Example: Backpropagation

But recall our output was:

$z^1 =$

-1	1	1
2	0	-2
-2	1	1

Which ReLU changed to

$Rz^1 =$

0	1	1
2	0	0
0	1	1

So our only relevant values will be:

0	0	0	0	0	0	0
0	0	2	$\boxed{0}$	0	$\boxed{2}$	0
0	0	0	0	2	0	0
0	$\boxed{2}$	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	$\boxed{2}$	1	$\boxed{0}$	0
0	0	0	0	0	0	0

Notice the center 0 wasn't changed but we still eliminated it as a candidate, this is a consequence of the ReLU derivate being undefined at 0

Since the derivate $\frac{dRz^1}{dz^1}$ will be 0 for the other values

Recall the chain:

$$\frac{dmz_{ef}^1}{d\theta_{11}^1} = \frac{dmz_{ef}^1}{dRz^1} \times \frac{dRz^1}{dz^1} \times \frac{dz^1}{d\theta_{11}^1} = \frac{dmz_{ef}^1}{dRz^1} \times \mathbf{0} \times \frac{dz^1}{d\theta_{11}^1}$$

Image Example: Backpropagation

And max pooling left us with output:

$mz^1 = \boxed{2}$

Which was this entry:

0	1	1
2	0	0
0	1	1

So now our only relevant value will be:

0	0	0	0	0	0	0
0	0	2	0	0	2	0
0	0	0	0	2	0	0
0	2	1	2	0	0	0
0	2	2	1	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Since the derivate $\frac{dmz^1_{ef}}{dRz^1}$ will be 0 for the other values

Recall the chain:

$$\frac{dmz^1_{ef}}{d\theta^1_{11}} = \frac{dmz^1_{ef}}{dRz^1} \times \frac{dRz^1}{dz^1} \times \frac{dz^1}{d\theta^1_{11}} = 0 \times \frac{dRz^1}{dz^1} \times \frac{dz^1}{d\theta^1_{11}}$$

Image Example: Backpropagation

So we get to this:

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)(1-\ \sigma(Fz)) \times \sum_{e=0}^{m-1}\sum_{f=0}^{m-1} \omega_{ef}^{1}\ \frac{dmz_{ef}^{1}}{d\theta_{11}^{1}}\quad \frac{\boldsymbol{dmz_{ef}^{1}}}{\boldsymbol{dRz^{1}}} \times \frac{\boldsymbol{dRz^{1}}}{\boldsymbol{dz^{1}}} \times \frac{\boldsymbol{dz^{1}}}{\boldsymbol{d\theta_{11}^{1}}}$$

$$1 \times 1 \times 2$$

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)(1-\ \sigma(Fz)) \times \sum_{e=0}^{m-1}\sum_{f=0}^{m-1} \omega_{ef}^{1} \times 2$$

Since e = 1, f = 1 in our example.

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)\big(1-\ \sigma(Fz)\big) \times \omega_{00}^{1} \times 2$$

Recall $\omega_{00}^{1} = -1$

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1-y^{*})\frac{1}{y-1} \times \sigma(Fz)\big(1-\ \sigma(Fz)\big) \times -2$$

Image Example: Backpropagation

$$\frac{dJ}{d\theta_{11}^{1}} = y^{*}\frac{1}{y} + (1 - y^{*})\frac{1}{y-1} \times \sigma(Fz)\big(1 - \sigma(Fz)\big) \times -2$$

Recall: $Fz = 0.3 - 2 + 1 = -0.7$ and $y = 0.332$, recall this example is classified as $y^{*} = 1$

These are values we already calculated in the forward propagation, which is why it is useful to store them so we don't have to calculate them again

$$\frac{dJ}{d\theta_{11}^{1}} = \frac{1}{0.332} \times \sigma(-0.7)\big(1 - \sigma(-0.7)\big) \times -2 = -1.336$$

$$\theta_{11}^{1} = \theta_{11}^{1} + 1.336\lambda$$

Note: Make sure you update all of your weights before you change the values of Fz!