Midterm Review

CS230 Fall 2018

Broadcasting

Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

$$M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix}$$

Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

$$M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix}$$

Rows: `row_mu = np.sum(M, axis=1) / M.shape[1]`

Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

$$M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix}$$

Rows: `row_mu = np.sum(M, axis=1) / M.shape[1]`

Cols: `col_mu = np.sum(M, axis=0) / M.shape[0]`

Computing Softmax

How would you compute the softmax across the columns of the following matrix?

$$M = \begin{bmatrix} 20 & 1 \\ 3 & 24 \\ 31 & 13 \end{bmatrix}$$

Computing Softmax

How would you compute the softmax across the columns of the following matrix?

$$M = \begin{bmatrix} 20 & 1 \\ 3 & 24 \\ 31 & 13 \end{bmatrix}$$

```
exp = np.exp(M)
```

Computing Softmax

How would you compute the softmax across the columns of the following matrix?

$$M = \begin{bmatrix} 20 & 1 \\ 3 & 24 \\ 31 & 13 \end{bmatrix}$$

```
exp = np.exp(M)

smx = exp / np.sum(exp, axis=0)
```

Computing Distances

How would you compute the closest column in the matrix X to the vector V (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

Computing Distances

How would you compute the closest column in the matrix X to the vector V (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

Computing Distances

How would you compute the closest column in the matrix X to the vector V (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

```
sq_diff = np.square(X-V)
```

Computing Distances

How would you compute the closest column in the matrix X to the vector V (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

```
sq_diff = np.square(X-V)
dists = np.sqrt(np.sum(sq_diff, axis=0))
```

Computing Distances

How would you compute the closest column in the matrix X to the vector V (in terms of Euclidean distance)?

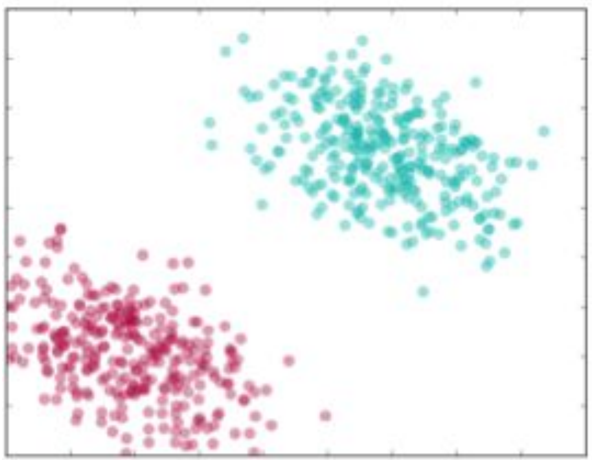
$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

```
sq_diff = np.square(X-V)
dists = np.sqrt(np.sum(sq_diff, axis=0))
nearest = np.argmin(dists)
```

L1/L2 Regularization

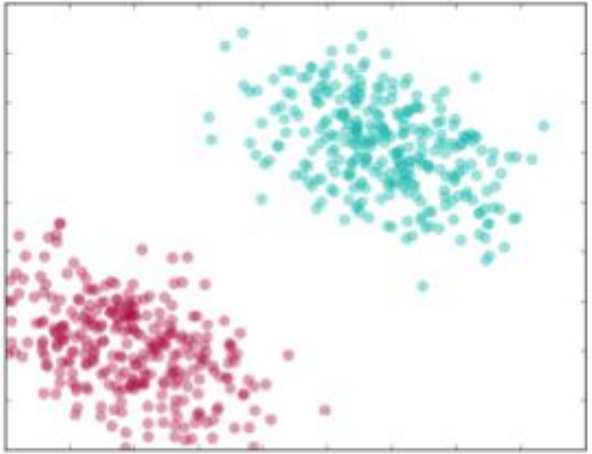
Logistic Regression and Separable Data

What's the issue with training a logistic regression model on the following data?



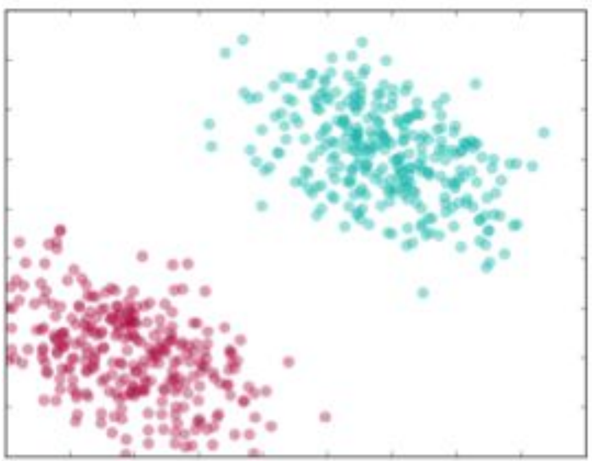
Logistic Regression and Separable Data

What’s the issue with training a logistic regression model on the following data?



Logistic Regression and Separable Data

What's the issue with training a logistic regression model on the following data?



The parameters will tend to plus/minus infinity! So, it will never converge.

Solving the Exploding Weights Issue

What modification of the loss function can you implement so solve this issue? Write out the new loss function.

Solving the Exploding Weights Issue

What modification of the loss function can you implement so solve this issue? Write out the new loss function.

Add L2 Regularization

$$L(y, \hat{y}, w) = -ylog(\hat{y}) - (1 - y)\log(1 - \hat{y}) + \frac{\lambda}{2}\sum_{i=1}^{n} w_i^2$$

This new loss function will keep the magnitude of the weights from exploding!

Gradient of the New Loss

Compute the gradient of the weight vector with respect to this new loss function.

Gradient of the New Loss

Compute the gradient of the new loss function with respect to the weight vector.

$$\frac{dL}{dw} = (\hat{y} - y)x + \lambda w$$

Another Solution…

What is another, similar modification to the loss function that could help with this issue? Compute its gradient?

Another Solution...

What is another, similar modification to the loss function that could help with this issue? Compute its gradient?

Add L1 Regularization:

$$L(y, \hat{y}, w) = -y log(\hat{y}) - (1 - y)\log(1 - \hat{y}) + \sum_{i=1}^{n} |w_i|$$

$$\frac{dL}{dw} = (\hat{y} - y)x + sign(w)$$

Backprop

$L = (y_{pred} - y)^{3/2}$ where $y_{pred} = tanh(x) \cdot W$ Assume that $y_{pred} \in R^{1xn}$ and $x \in R^{1xm}$

Using backpropagation, obtain $\frac{\partial L}{\partial x}$

$$L = (y_{pred} - y)^{3/2}$$

$$y_{pred} \in R^{1 \times n}$$

$$y_{pred} = \tanh(x) . W$$

$$x \in R^{1 \times m}$$

$$\frac{\partial L}{\partial x} = ?$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial \tanh(x)} \circ \frac{\partial \tanh(x)}{\partial x}$$

$$= \underbrace{\frac{3}{2}(y_{pred} - y)^{1/2}}_{1 \times n} \circ \underbrace{W}_{m \times n} \cdot \underbrace{(1 - \tanh^2(x))}_{1 \times m}$$

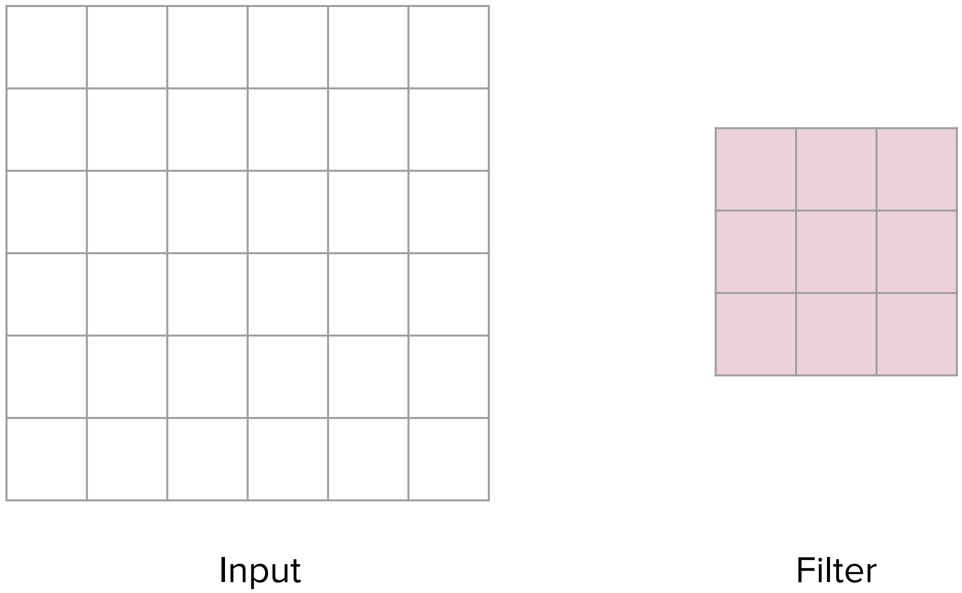
$$\underbrace{y_{pred}}_{1 \times n} = \underbrace{\tanh(x)}_{1 \times m} . \underbrace{W}_{m \times n}$$

$$= \frac{3}{2}(y_{pred} - y)^{1/2} \cdot W^T \odot (1 - \tanh^2(x))$$

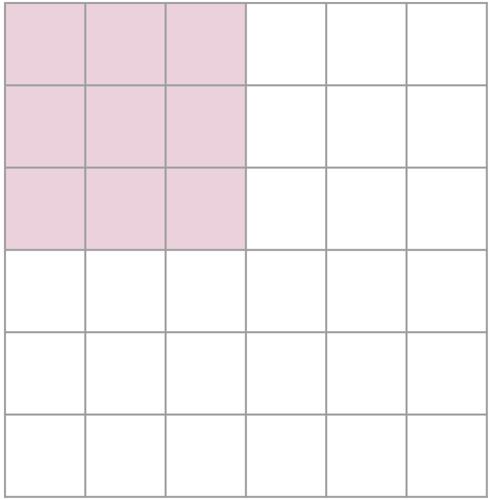
element-wise

CNN Input/Output Sizes

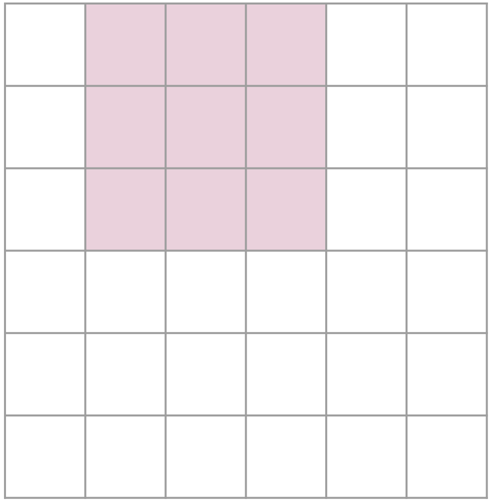
Basic (no padding, stride 1)



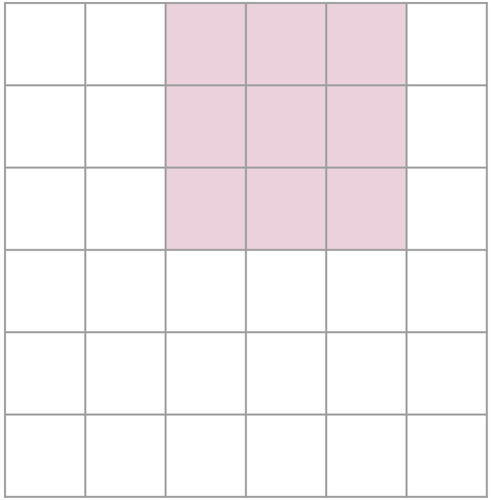
Basic



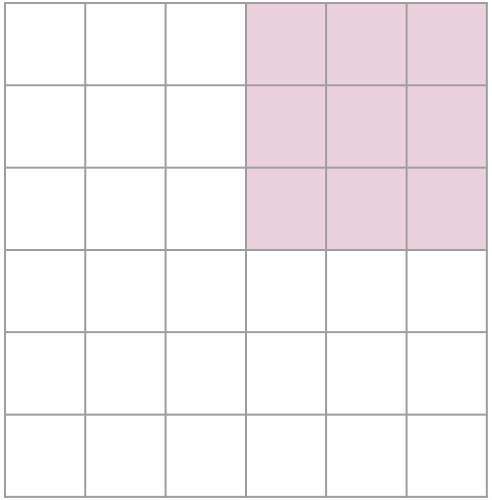
Basic



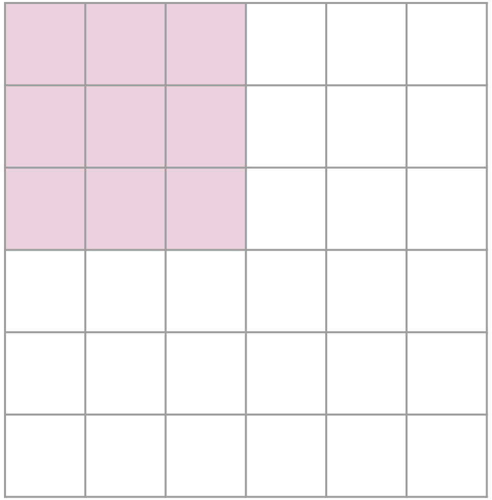
Basic



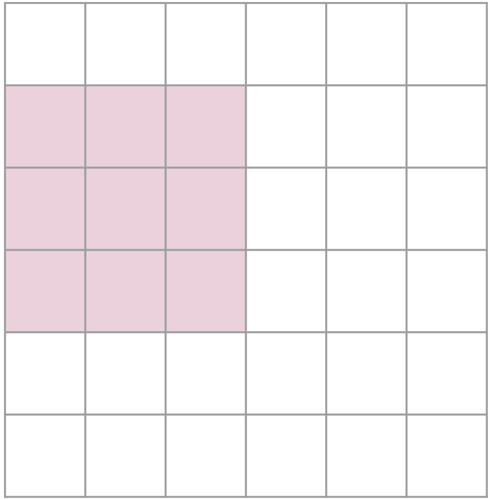
Basic



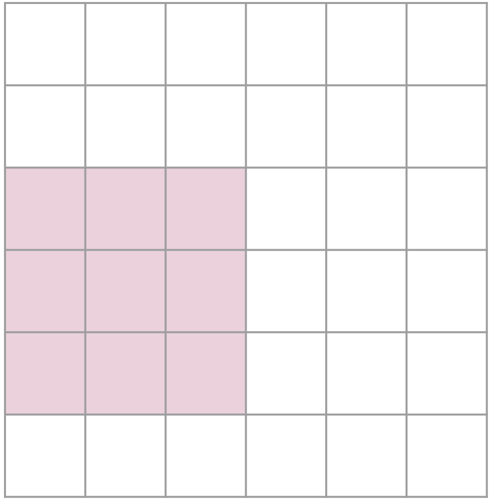
Basic



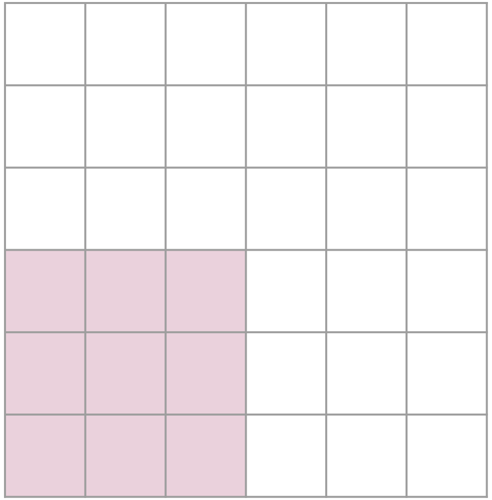
Basic



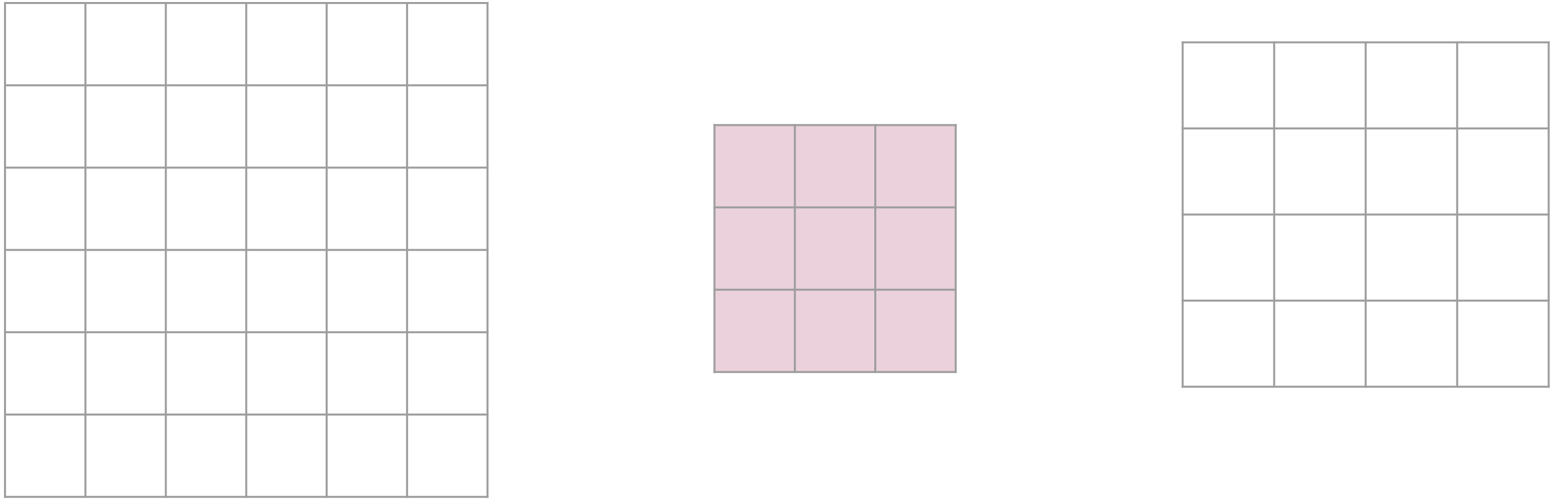
Basic



Basic



Basic



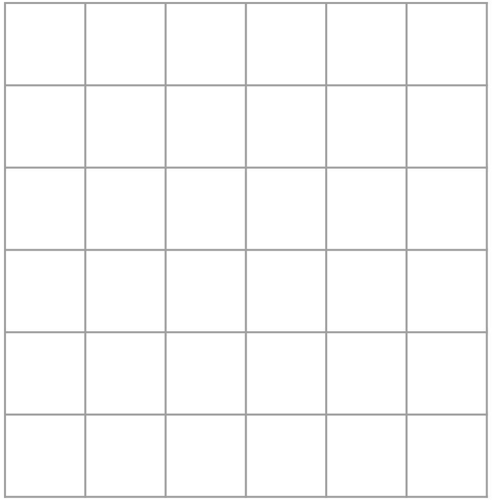
Input

Filter

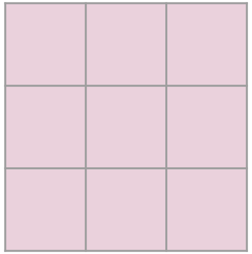
Conv Output

Basic

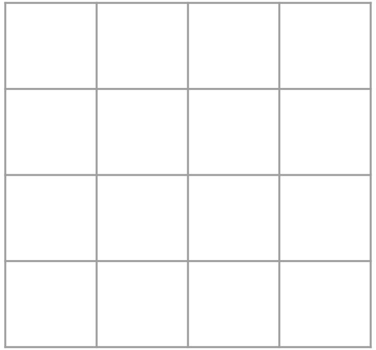
Shape = n - f + 1



Input



Filter



Conv Output

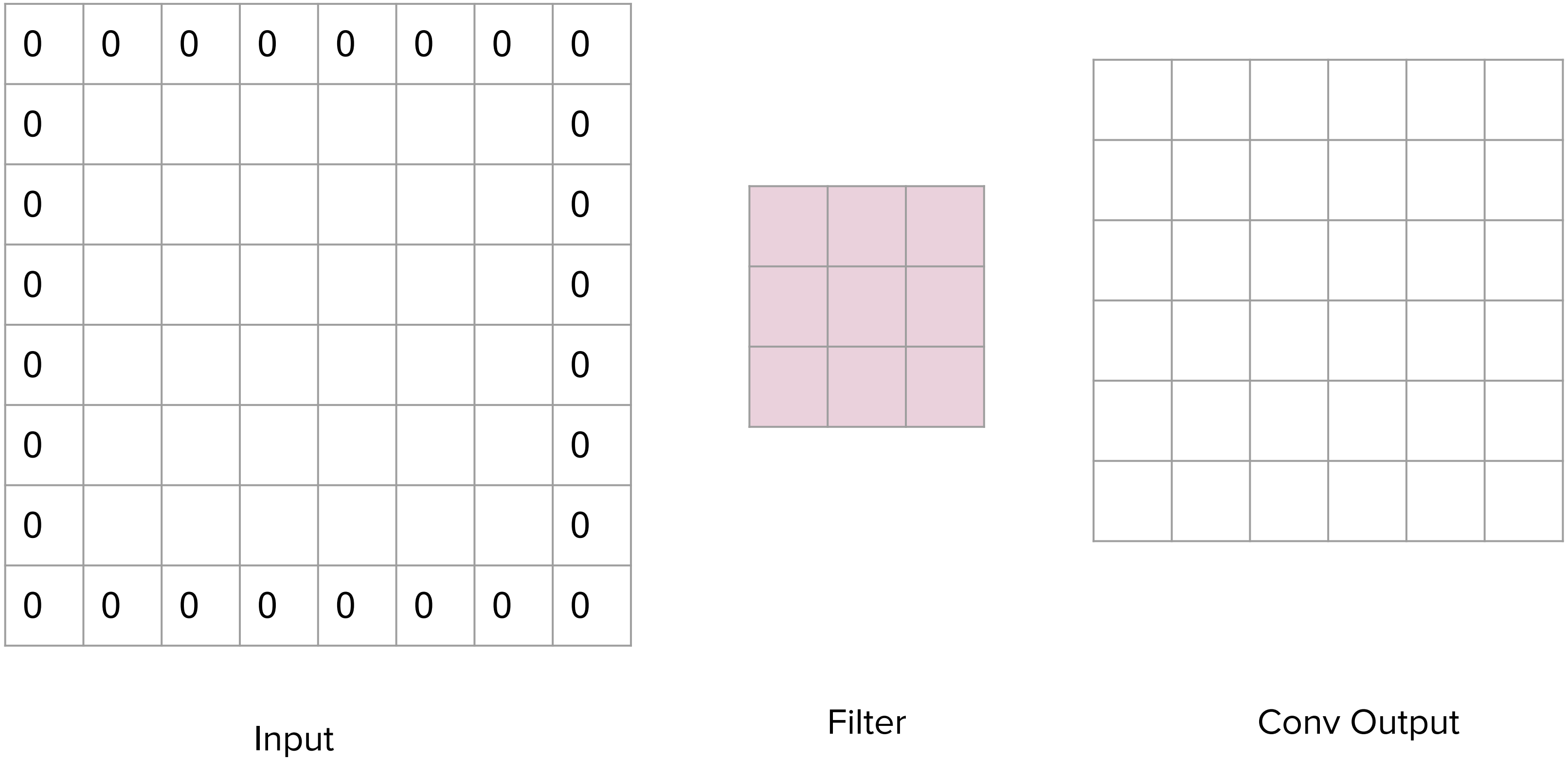
Padding

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

Input

Filter

Padding



Padding

Shape = $n + 2p - f + 1$

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

Input

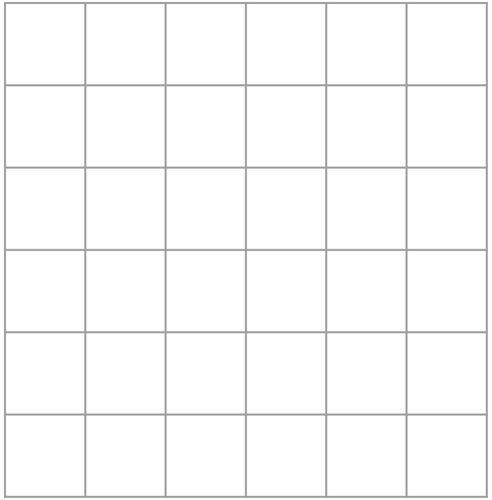
Filter

Conv Output

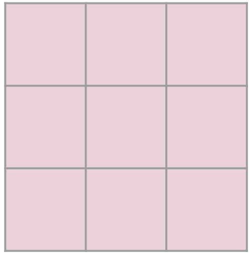
Valid and Same Convolutions

- **Valid**
  - No padding
  - Output shape -> n - f + 1
- **Same**
  - Pad so that input is same as output size
  - Output shape -> n + 2p - f + 1

Stride

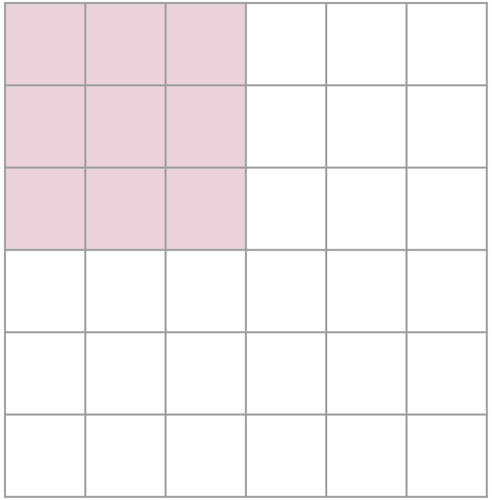


Input

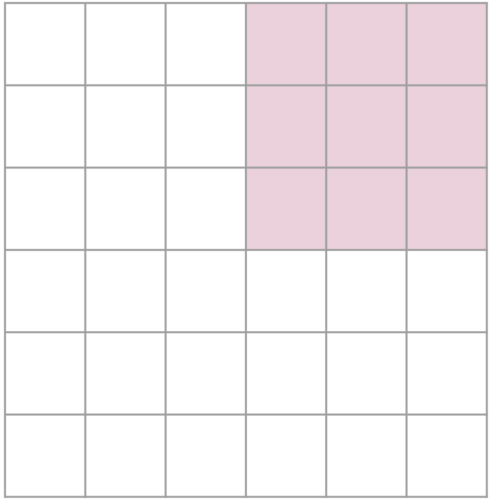


Filter

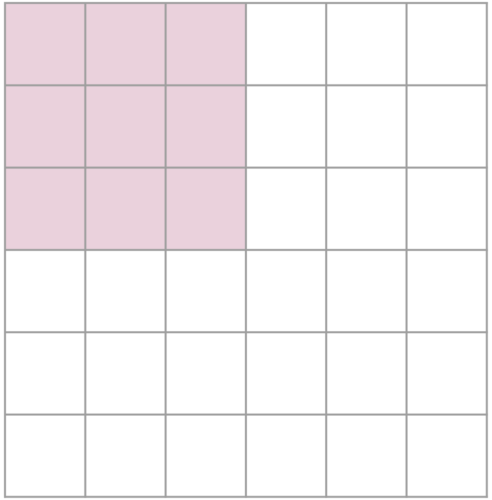
Stride



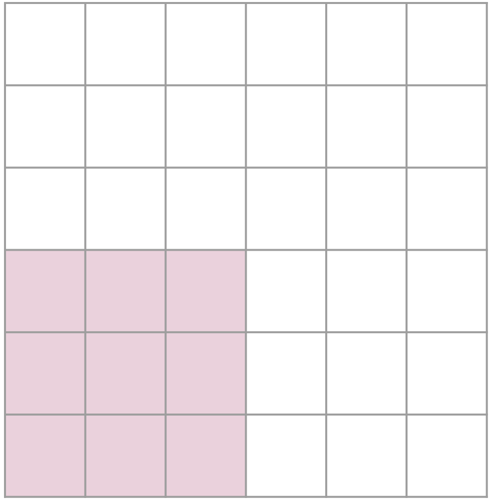
Stride



Stride

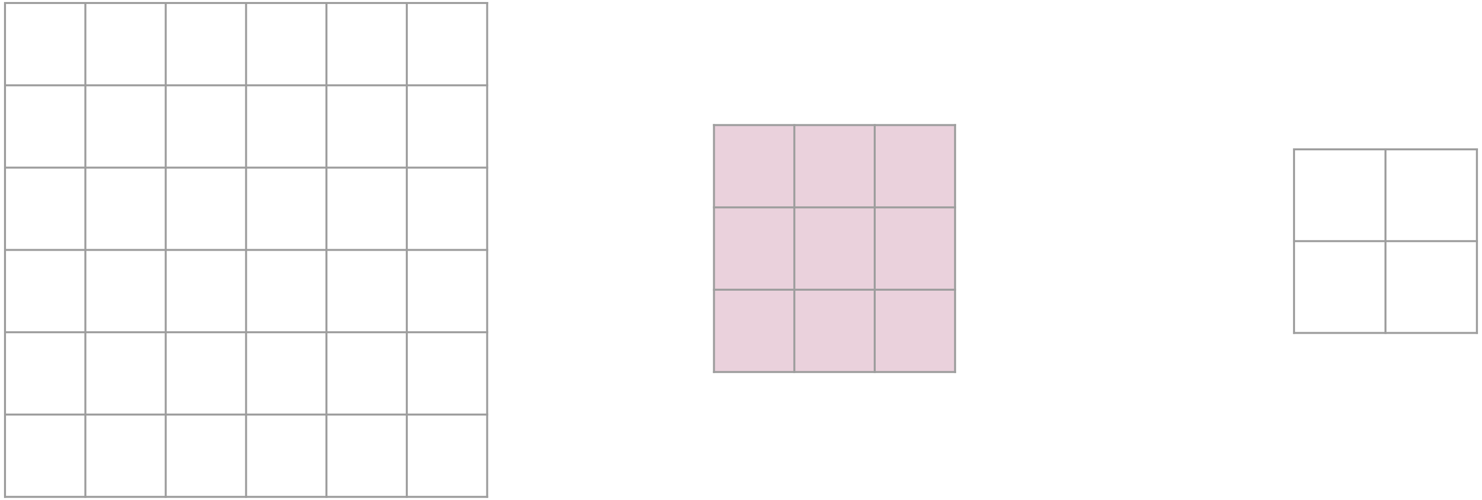


Stride



Basic

Shape = $(n - f)/s + 1$



Input

Filter

Conv Output

With Stride

n x n image

f x f filter

p padding

s stride

Output size -> (**n + 2p - f)/s + 1**

Maxpool

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

Output

Size of output
(n-f)/s + 1

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	

Output

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	5

Output

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	5
6	

Output

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	5
6	9

Output

Forward prop

2 x 2 Pooling layer with stride 2

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	5
6	9

Output

Backprop

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input to maxpool layer

10	5
6	9

Output of maxpool layer

-4	7
5	-6

Gradient w.r.t output

Backprop

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Gradient w.r.t input

-4	7
5	-6

Gradient w.r.t output

ReLU

$$ReLU(x) = \max(0, x)$$

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{otherwise.} \end{cases}$$

$$\frac{d}{dx} ReLU(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases}$$

Maxpool

$$\text{Maxpool}(m_{ij}) = \begin{cases} 0, & \text{if } m_{ij} \neq \max(m) \\ x, & \text{if } m_{ij} = \max(m) \end{cases}$$

$$\frac{d}{dx}\text{Maxpool}(m_{ij}) = \begin{cases} 0, & \text{if } m_{ij} \neq \max(m) \\ 1, & \text{if } m_{ij} = \max(m) \end{cases}$$

Backprop

Keep track of where the maximum value is

1	3	2	1
4	10	5	1
1	6	6	5
2	4	2	9

Input

10	5
6	9

Output

0	0	0	0
0	1	1	0
0	1	0	0
0	0	0	1

Mask

Backprop

0	0	0	0
0	1	1	0
0	1	0	0
0	0	0	1

Mask

-4	7
5	-6

Gradient w.r.t output

0	0	0	0
0	-4	5	0
0	5	0	0
0	0	0	-6

Gradient w.r.t input

Error Analysis

Dog Classifier

Trying to predict **dog** vs **not dog.**

Improving performance

Two kinds of errors - misclassification on **muffins** and **fried chicken**

https://medium.freecodecamp.org/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d

https://barkpost.com/doodle-or-fried-chicken-twitter/

Error analysis

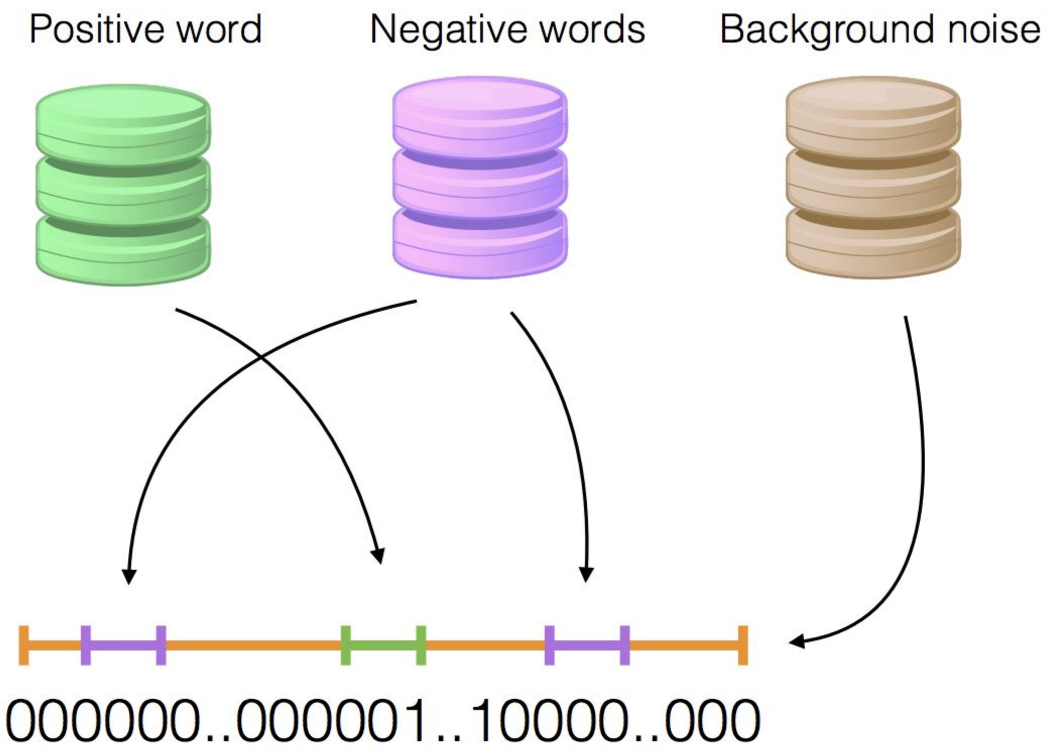
- Get 100 examples on dev set

Image number	Classified as muffin	Classified as chicken	...	Comments
1	Y	-		
2	-	Y		
...	-	-		
100	-	Y		
	5%	**50%**		

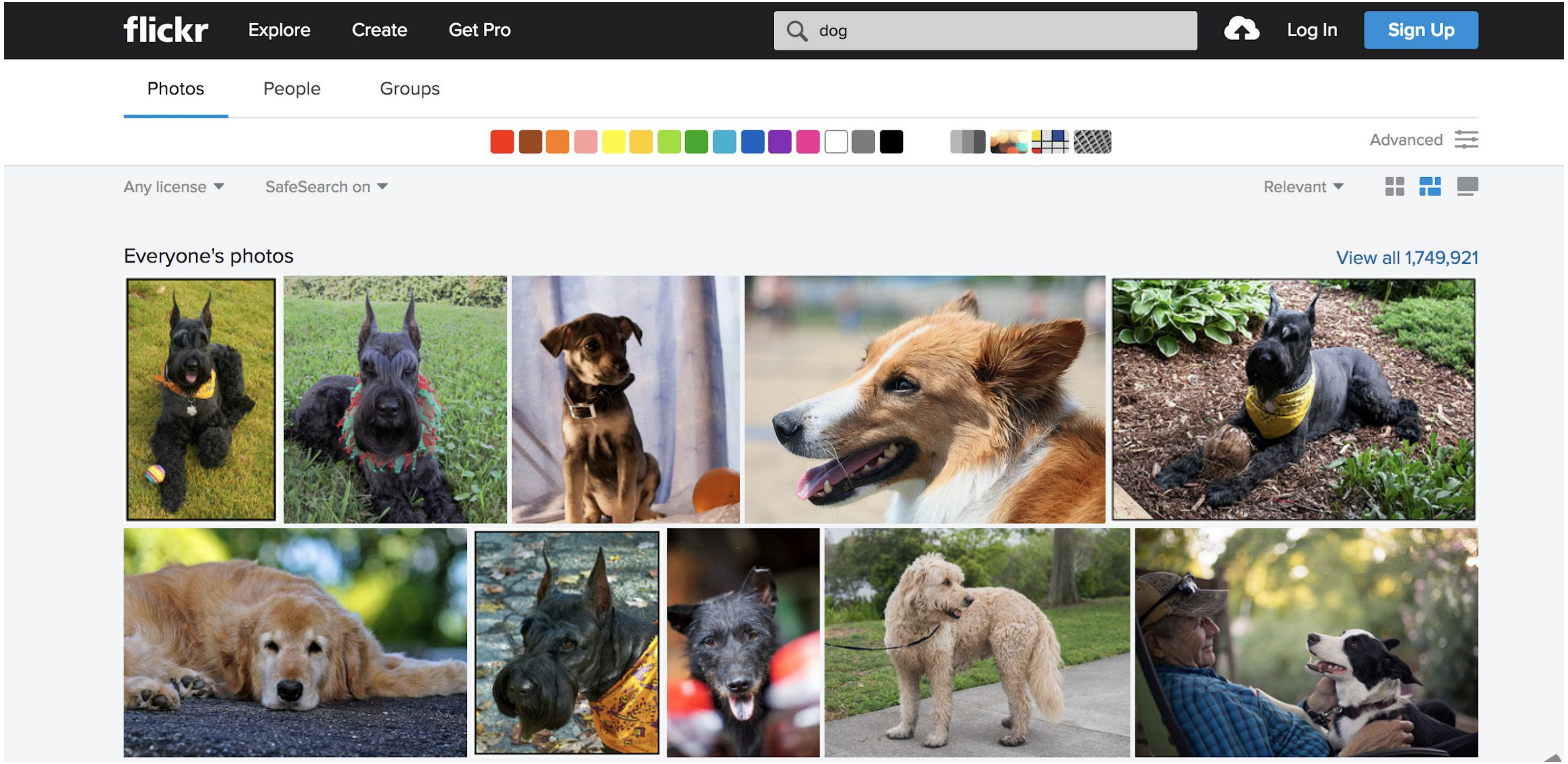
Error Analysis

Strategic Data Acquisition

Trigger Word Detection

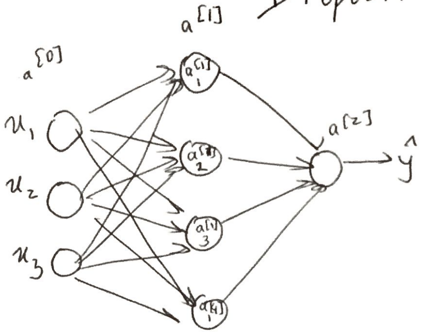


Classification



Dropout

Dropout



Dimensions of $W^{[1]}$?

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

keepprobability (p)

$$a = \begin{bmatrix} 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

$a^{[1]}$ shape = ? $[4,1]$

$$d^{[1]} = np.random.randn(a^{[1]}.shape) < p$$

$$\underbrace{\begin{bmatrix} 0.3 \\ 0.6 \\ 0.1 \\ 0.8 \end{bmatrix} < 0.5}_{\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}$$

$$a^{[1]} *= d^{[1]}$$

new $a^{[1]}$ after dropout = $\begin{bmatrix} 4 \\ 0 \\ 6 \\ 0 \end{bmatrix}$

Done?

Note.

Expected value of $a_i^{[1]}$

$$= p \cdot a_i^{[1]} + (1-p) \cdot 0$$

$$= p \, a_i^{[1]}$$

At test time → keep all on, i.e $p = 1$

$\therefore$ Expected value $= a_i^{[1]}$

$\therefore$ During training to get same expected value

$/= p$

Batchnorm

$$Z = \{z^{(1)}, \ldots., z^{(m)}\}$$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z^{(i)}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (z^{(i)} - \mu_B)^2$$

$$z_{norm}^{(i)} = \frac{z^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z_{norm}^{(i)} + \beta$$