GAN

Name	Dharini Baskaran
Identity Key	dhba5060

	Level	Completed
	Beginner	7
	Intermediate	5
	Advanced	2
	Expert	0

Goal	
4722	15
5722	17
Total Completed	
0	

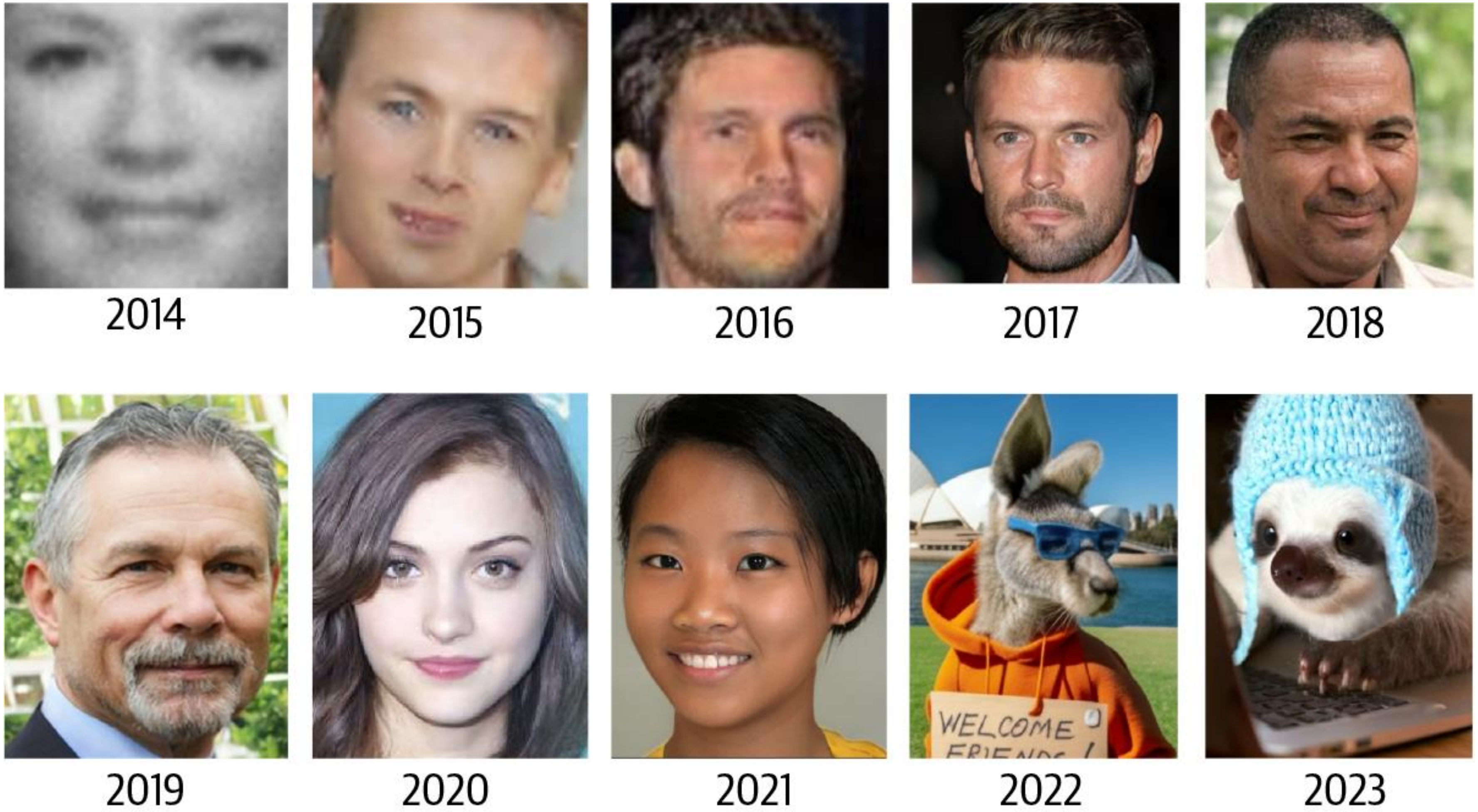
Generative Adversarial Network (GAN)

CSCI 5722/4722: Computer Vision

Spring 2024

Dr. Tom Yeh

Dr. Mehdi Moghari



https://www.oreilly.com/library/view/generative-deep-learning/9781098134174/ch01.html

Generative Adversarial Nets

Ian J. Goodfellow,[*] Jean Pouget-Abadie,[†] Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,[‡] Aaron Courville, Yoshua Bengio[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

1 Introduction

The promise of deep learning is to discover rich, hierarchical models [2] that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional, rich sensory input to a class label [14, 20]. These striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units [17, 8, 9] which have a particularly well-behaved gradient . Deep *generative* models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context. We propose a new generative model estimation procedure that sidesteps these difficulties. [1]

In the proposed *adversarial nets* framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

[*] Ian Goodfellow is now a research scientist at Google, but did this work earlier as a UdeM student
[†] Jean Pouget-Abadie did this work while visiting Université de Montréal from Ecole Polytechnique.
[‡] Sherjil Ozair is visiting Université de Montréal from Indian Institute of Technology Delhi
[§] Yoshua Bengio is a CIFAR Senior Fellow.
[1] All code and hyperparameters available at `http://www.github.com/goodfeli/adversarial`

Share your AI-generated image of an ANN

https://www.linkedin.com/feed/update/urn:li:activity:7160617831787548672

Computer Vision - S24
Tom Yeh • You
2m • Edited •

Activity: Share your AI-generated image of an ANN

Post a comment with your image, the prompt you provided, and the tool you

1 comment

Like Comment

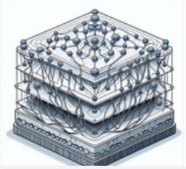
Add a comment...

Most recent

Tom Yeh (He/Him) Author 2m (edited)
Associate Professor of Computer Science at University of Colorado B...

Prompt:
Generate a graphic representation of an artificial neural network

Tool:
ChatGPT4



Like | Reply

Visit the link above
Add a comment
Attach your image

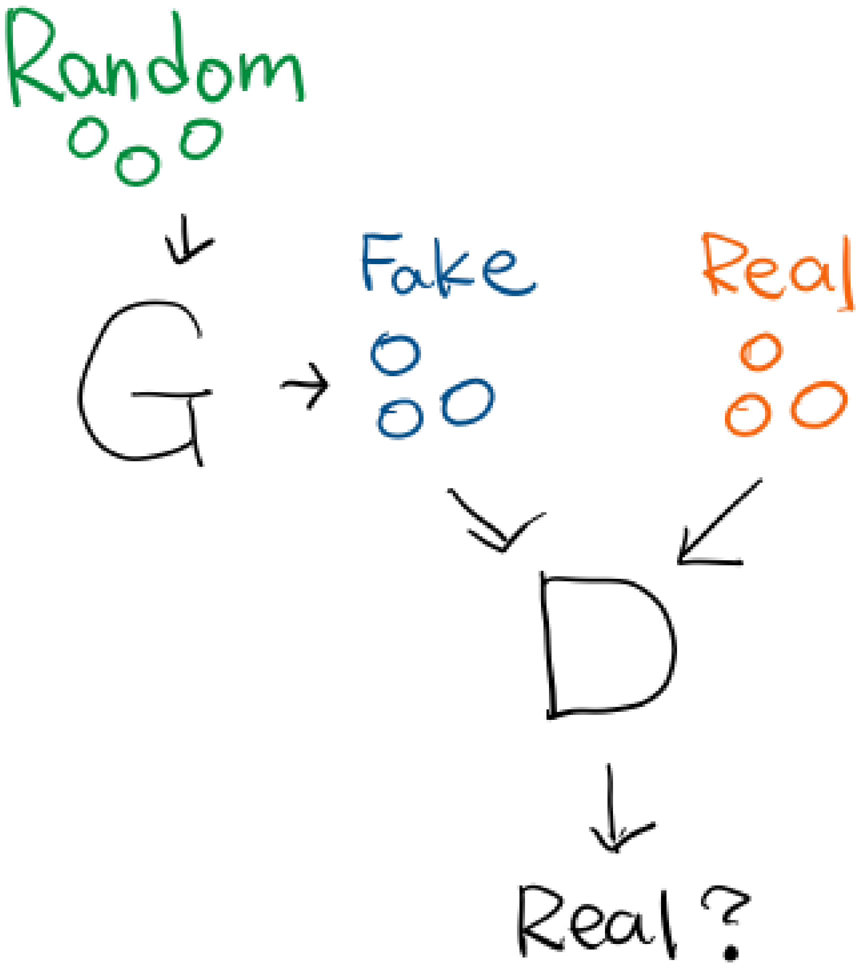
GAN Architecture

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Architecture Diagram

Diagram → ANN



Random

1	1	-1
1	-1	1
1	1	1

1	0	-2
0	2	1

Fake

-1	-1	-3
3	-1	3
1	1	1

Real

0	0	-1
3	-1	1
1	1	1

1	0	1
0	1	1

1	1	1

1	1	1

1	1	-3

Pred			
Fake	.3	.2	.4
Real	.5	.3	.7

Generator

CSCI 5722/4722 Computer Vision

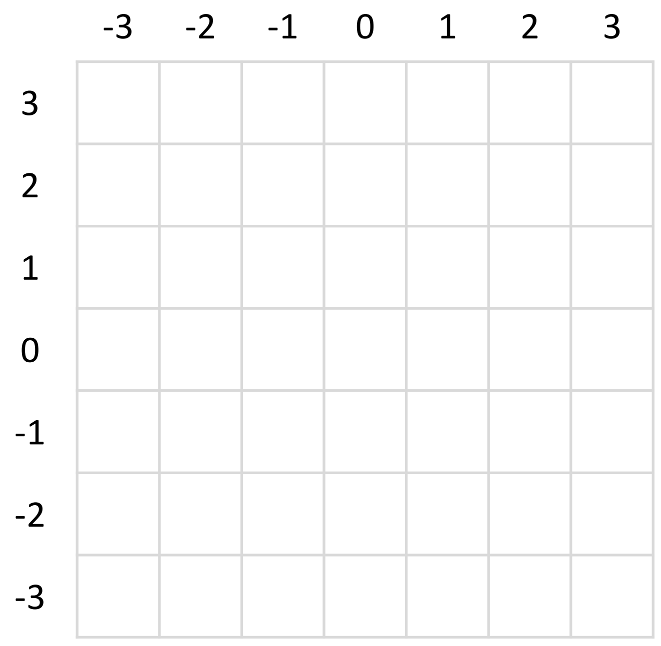
University of Colorado Boulder

Generator

1	1	-1
1	-1	1

1 1 1

1	0	-2
0	2	1



Discriminator

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Discriminator

A	B	C	D	E	F
-1	-1	-3	0	0	-1
3	-1	3	3	-1	1
1	1	1	1	1	1

1	0	1
0	1	1

1	1	1	1	1	1

1	1	-3

	-3	-2	-1	0	1	2	3
3							
2							
1							
0							
-1							
-2							
-3							

x >= 1

x >= 1

	-3	-2	-1	0	1	2	3
3	C		A	D			
2							
1			F				
0							
-1			B	E			
-2							
-3							

	-3	-2	-1	0	1	2	3
3	C		A	D			
2							
1			F				
0							
-1			B	E			
-2							
-3							

				A	B	C	D	E	F
				-1	-1	-3	0	0	-1
				3	-1	3	3	-1	1
				1	1	1	1	1	1
V_1									
V_2									

y >= 1

	-3	-2	-1	0	1	2	3
3	C		A	D			
2							
1			F				
0							
-1			B	E			
-2							
-3							

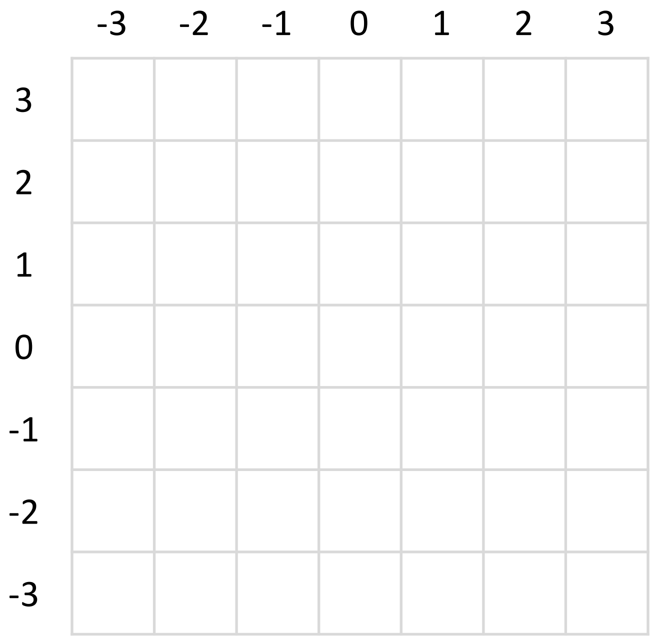
y >= 1

	-3	-2	-1	0	1	2	3
3	C		A	D			
2							
1			F				
0							
-1			B	E			
-2							
-3							

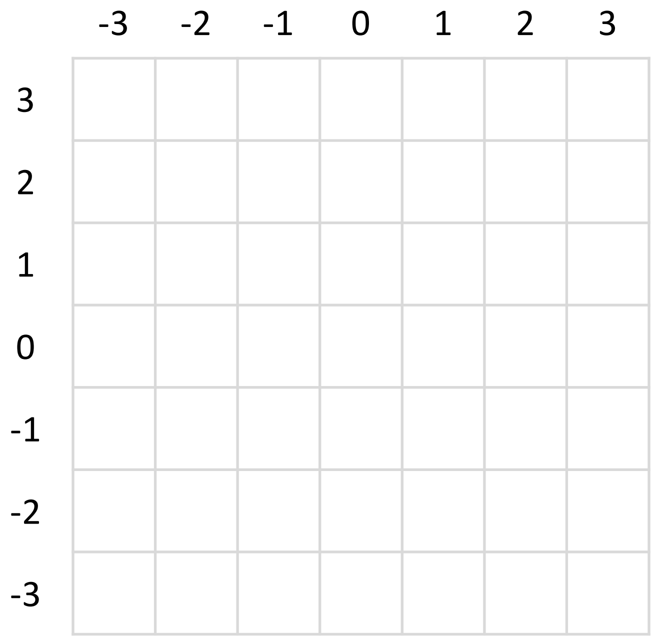
				A	B	C	D	E	F
				-1	-1	-3	0	0	-1
				3	-1	3	3	-1	1
				1	1	1	1	1	1
H_1									
H_2									

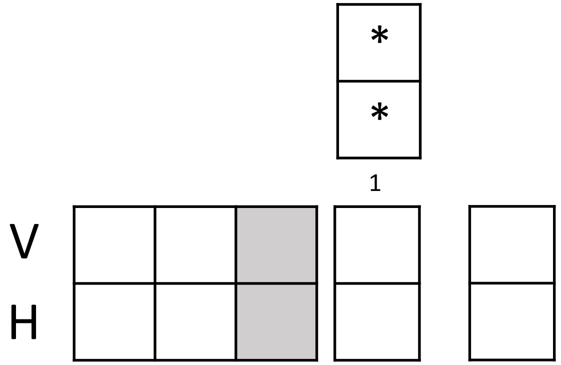
Activation Field (ReLU)

V: $x = -1$



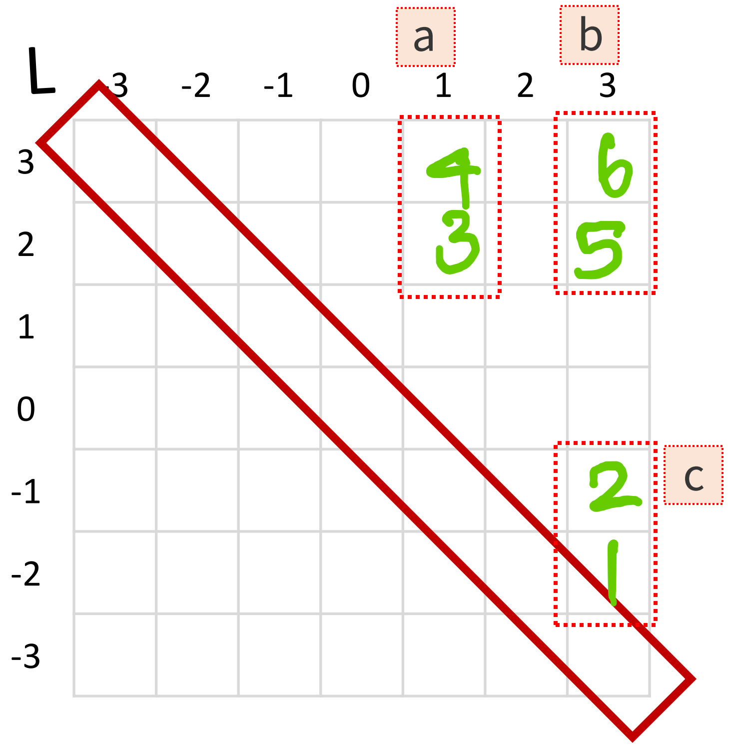
H: $y = -1$

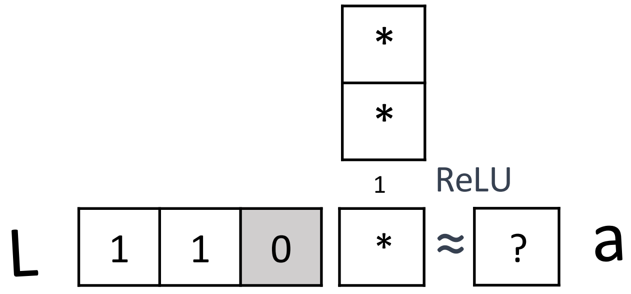




Activation Field (b = 0)

L: x+y = 0   a: ReLU(x+y)



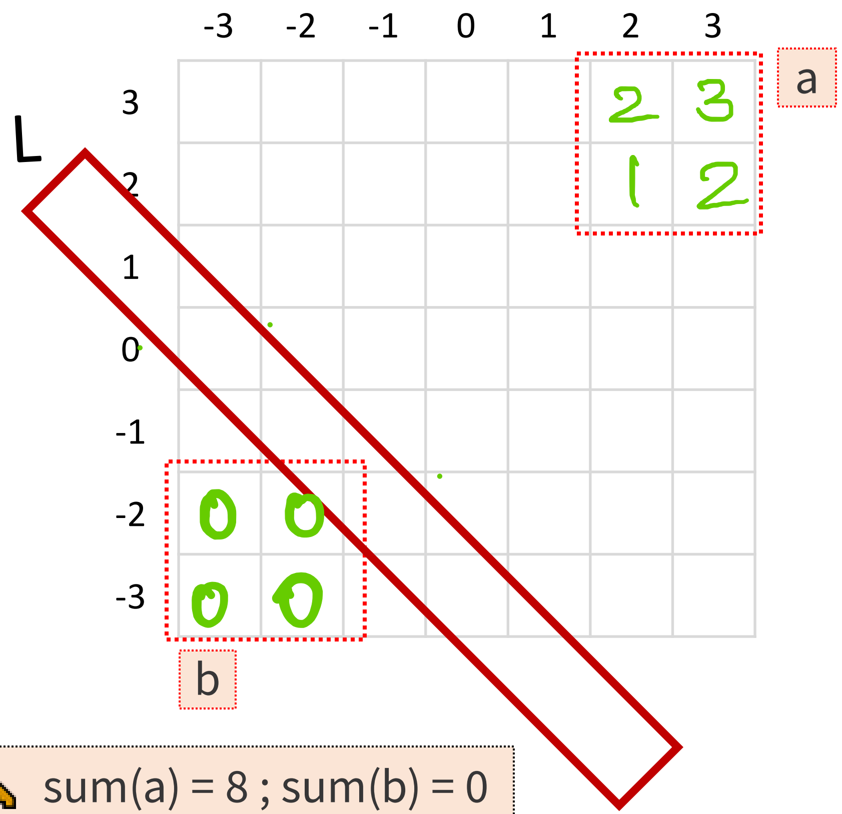


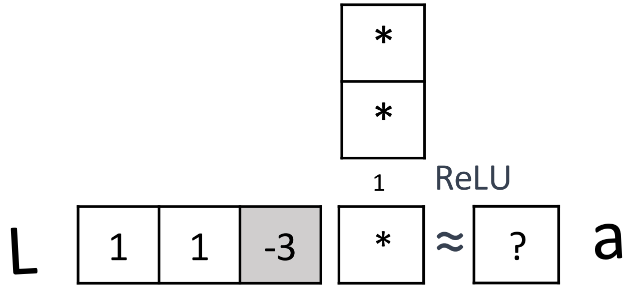
🔑 sum(a) = 7; sum(b) = 11; sum(c) = 3

(Drag L to the right)

Activation Field (b = -3)

L: x+y = 3 a: ReLU(x+y-3)





sum(a) = 8 ; sum(b) = 0

Weighted Sum Field

L: 2*x+y = 0

a	-3	-2	-1	0	1	2	3
3	-3	-1	1	3	5	7	9
2	-4	-2	0	2	4	6	8
1	-5	-3	-1	1	3	5	7
0	-6	-4	-2	0	2	4	6
-1	-7	-5				3	5
-2			-4	-2	0		
-3			-5	-3	-1		

b

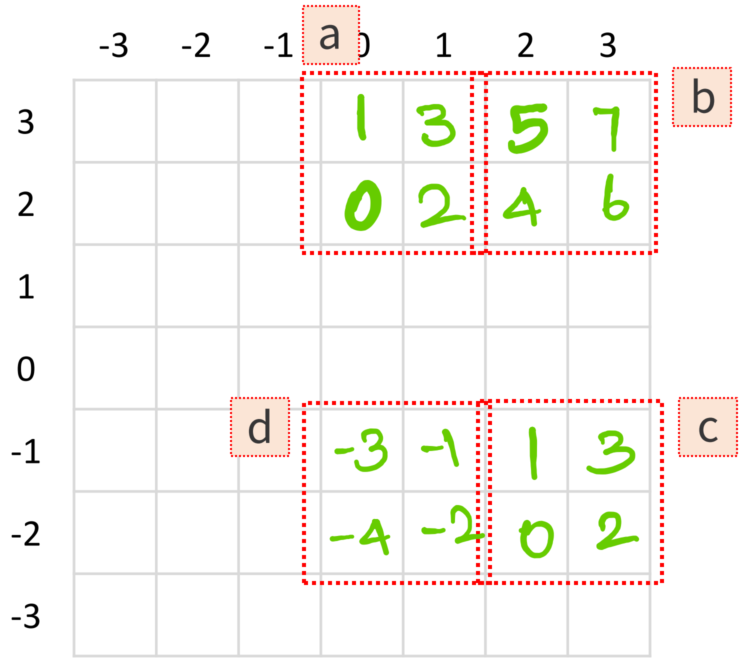
c

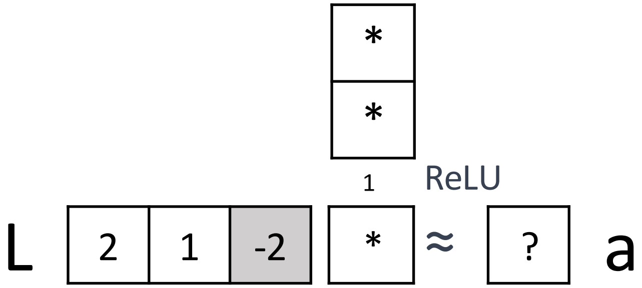
				*
				*
				1
L	2	1	0	?

sum(a) = -10; sum(b) = 3; sum(c) = 30;

Activation Field (b = -2)

L: 2*x+y = 2   a: ReLU(2*x+y – 2)





2(0)-1-2

2(0)-2-2

2(1)-1-2

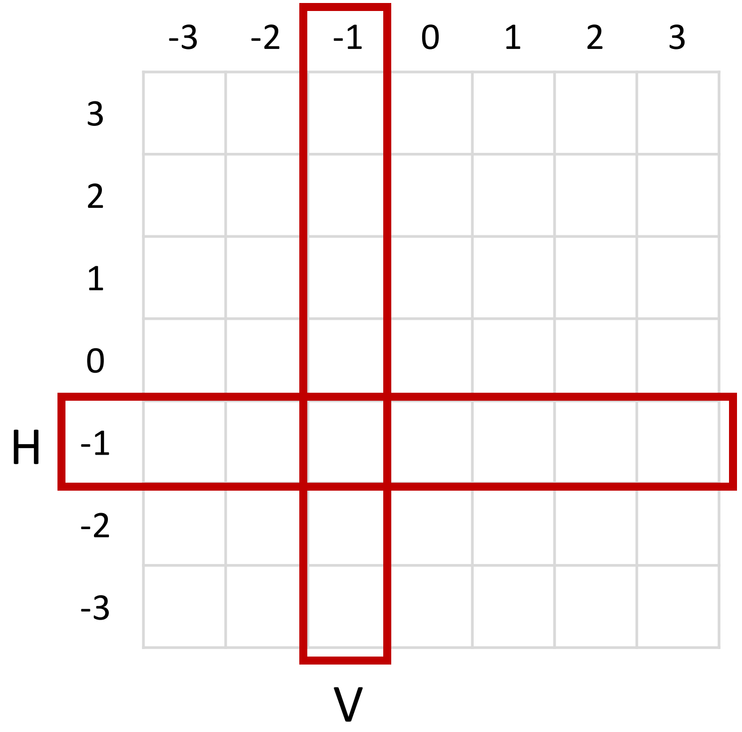
2(1)-2-2

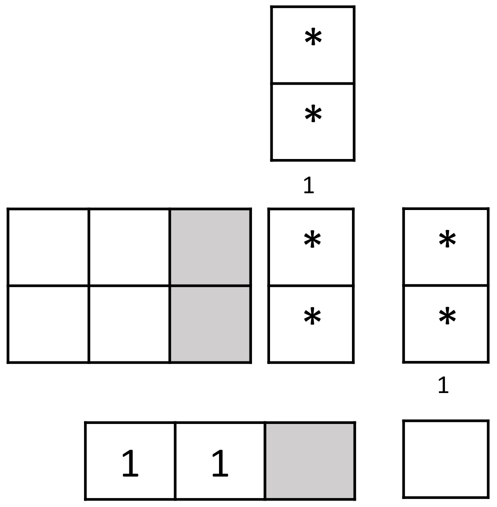
sum(a) = ~~7~~ 6; sum(b) = 22; sum(c) = 6;

Logit Field (b = 0)

V:

H:





Logit Field (b = -3)

V: $x = -1$

H: $y = -1$

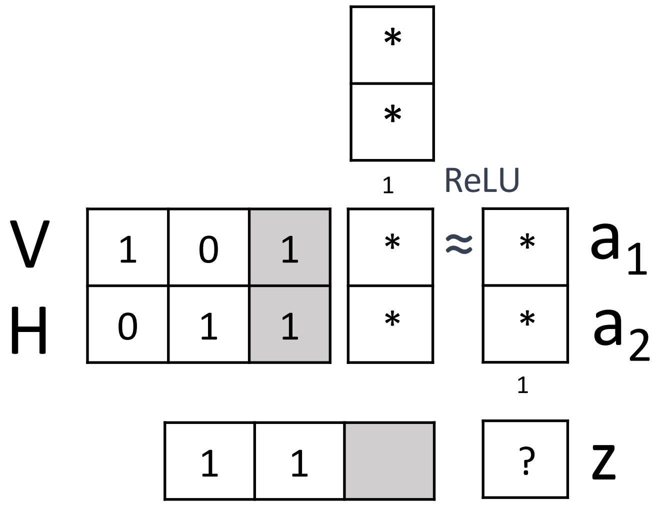
a_1: $\mathrm{ReLU}(x+1)$

a_2: $\mathrm{ReLU}(y+1)$

$z = a_1 + a_2$

	-3	-2	-1	0	1	2	3
3	4	4	4	5	6	7	8
2	3	3	3	4	5	6	7
1	2	2	2	3	4	5	6
0	1	1	1	2	3	4	5
H -1				1	2	3	4
-2				1	2	3	4
-3				1	2	3	4

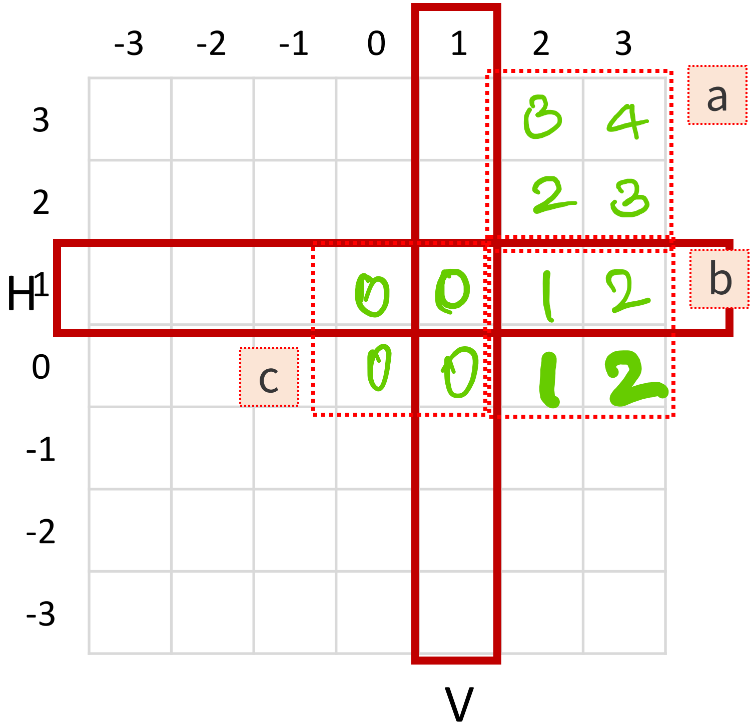
V



Logit Field (b = 0)

V: x = 1

H: y = 1



(Drag H, V to the right places)

2 0

$$\begin{array}{c} \\ V \\ H \end{array} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} * \\ * \\ 1 \end{bmatrix} \begin{bmatrix} * \\ * \end{bmatrix} \overset{\text{ReLU}}{\approx} \begin{bmatrix} * \\ * \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} ? \end{bmatrix}$$

3-1 = 2
0-1 = -1
3

1+0
1+1

2+0
2+0

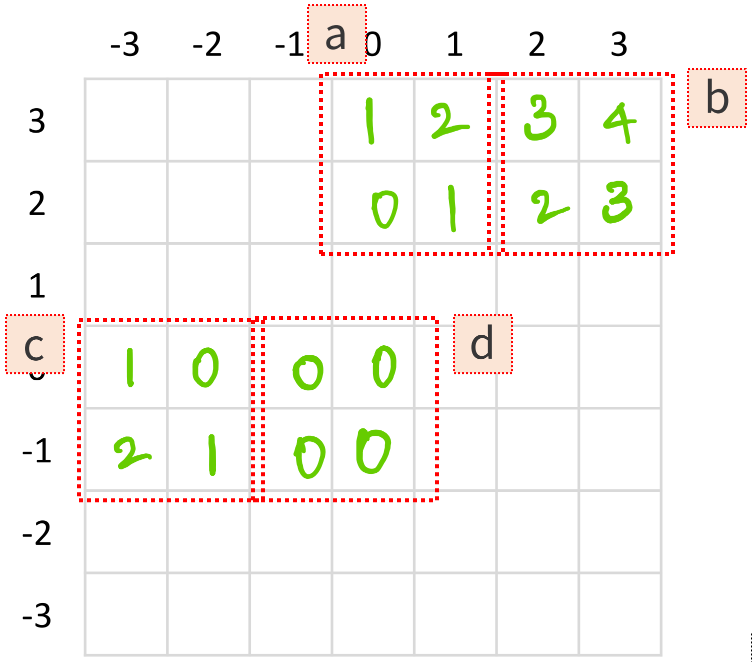
12 6

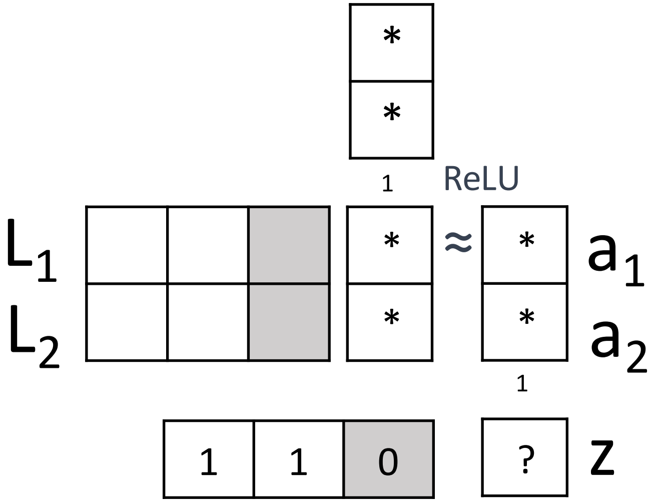
🔑 sum(a) = ~~7~~; sum(b) = ~~3~~;

Logit Field

L_1: $x+y = 2$ $\quad$ a_1: $\mathrm{ReLU}(x+y-2)$ $\quad$ z: a_1+a_2

L_2: $x+y = -2$ $\quad$ a_2: $\mathrm{ReLU}(-x-y-2)$





sum(a) = ~~3~~ 4; sum(b) = ~~16~~ 12; sum(c) = 4

Logit Field

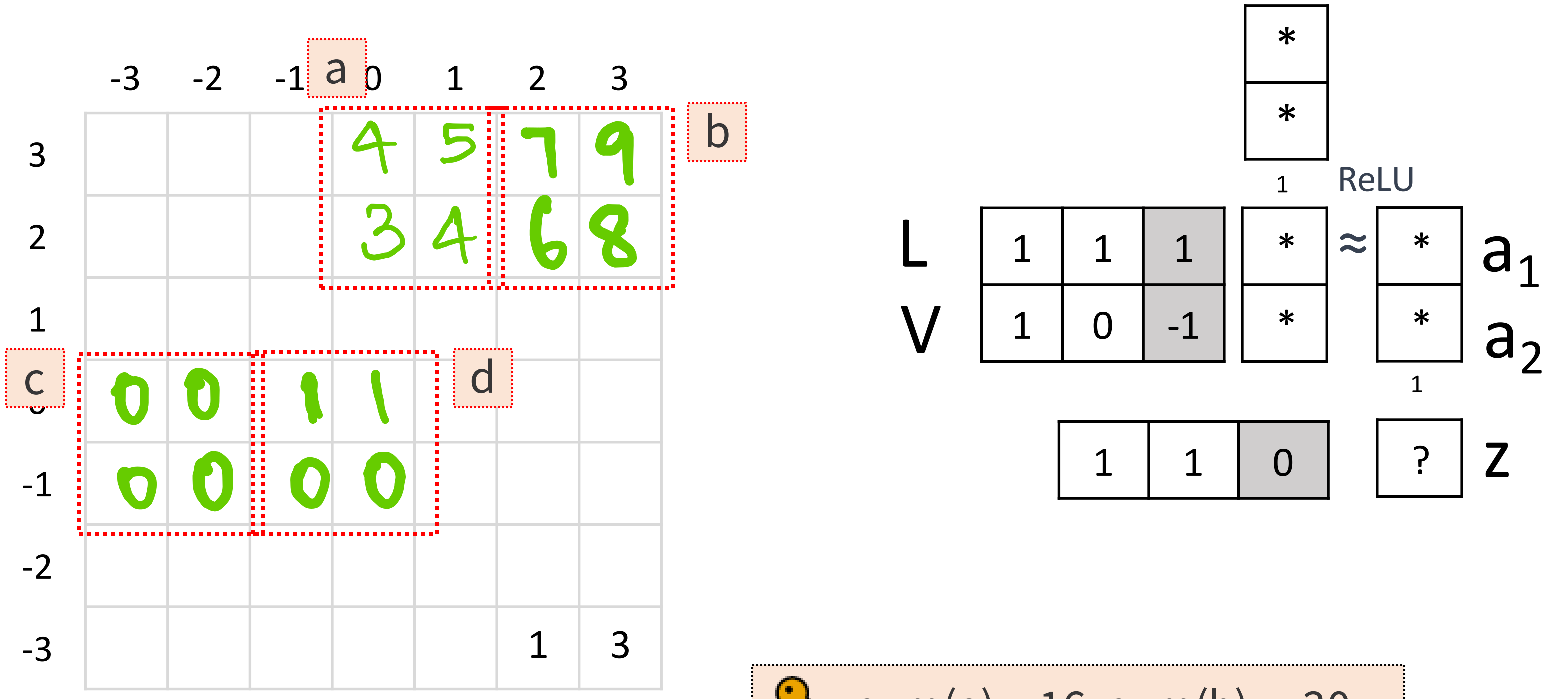
L: $x+y = -1$

V: $x = 1$

a_1: ReLU$(x+y+1)$

a_2: ReLU$(x-1)$

z: a_1+a_2



🔑 sum(a) = 16; sum(b) = 30;

Logit Field → Output Probability Field

V: $x = -1$

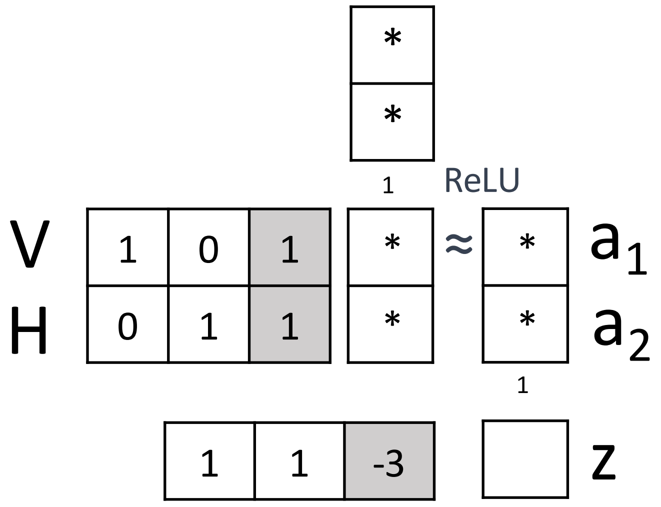
H: $y = -1$

a_1: ReLU(x+1)

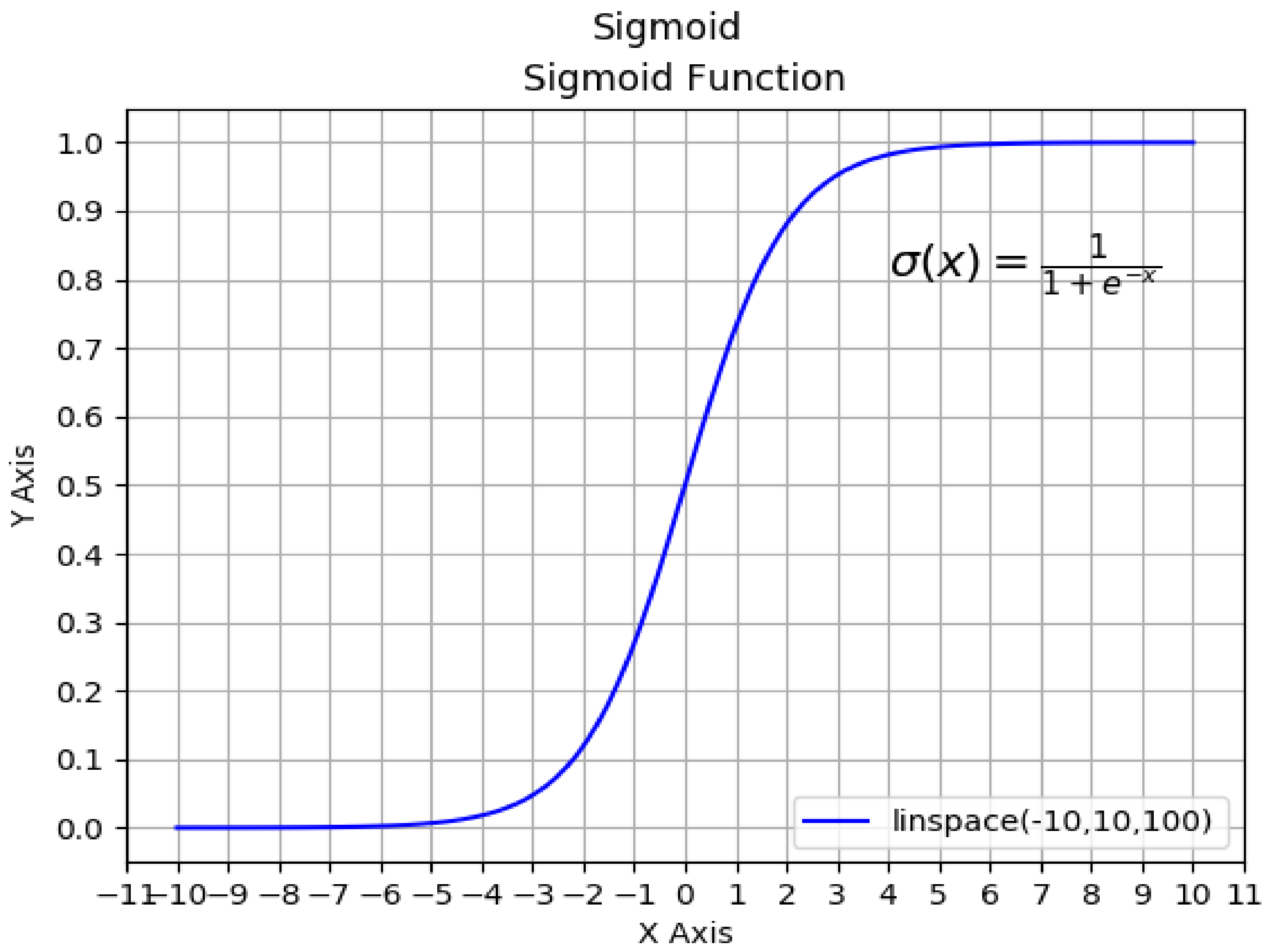
a_2: ReLU(y+1)

$z = a_1 + a_2 - 3$

		-3	-2	-1	0	1	2	3
	3	1	1	1	2	3	4	5
	2	0	0	0	1	2	3	4
	1	-1	-1	-1	0	1	2	3
	0	-2	-2	-2	-1	0	1	2
H	-1	-3	-3	-3	-2	-1	0	1
	-2	-3	-3	-3	-2	-1	0	1
	-3	-3	-3	-3	-2	-1	0	1
				V				



Approximate Sigmoid Function by Hand ✍



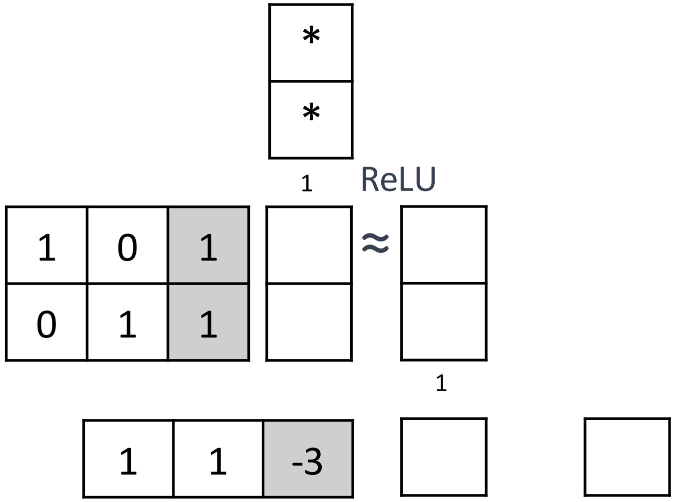
σ(<=-3) ≈
σ(-2) ≈
σ(-1) ≈
σ(0) ≈
σ(-1) ≈
σ(-2) ≈
σ(>=3) ≈

Image Credit: Alexander | Hackaday.io

Output Probability Field

	-3	-2	-1	0	1	2	3
3	1	1	1	2	3	4	5
2	0	0	0	1	2	3	4
1	-1	-1	-1	0	1	2	3
0	-2	-2	-2	-1	0	1	2
-1	-3	-3	-3	-2	-1	0	1
-2	-3	-3	-3	-2	-1	0	1
-3	-3	-3	-3	-2	-1	0	1

$\sigma(<=-3) \approx 0$
$\sigma(-2) \approx 0.1$
$\sigma(-1) \approx 0.3$
$\sigma(0) \approx 0.5$
$\sigma(-1) \approx 0.7$
$\sigma(-2) \approx 0.9$
$\sigma(>=3) \approx 1$



GAN

Noise

1	1	-1
1	-1	1
1	1	1

1	0	-2
0	2	1

Fake A	Fake B	Fake C	Real D	Real E	Real F
-1	-1	-3	0	0	-1
3	-1	3	3	-1	1
1	1	1	1	1	1

1	0	1
0	1	1

[ReLU]

1	1	1	1	1	1

1	1	-3

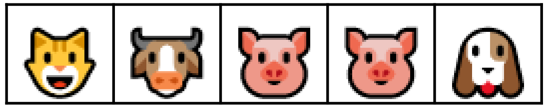
C		A	D			
		F				
		B	E			

.7	.7	.7	.9	1	1	1
.5	.5	.5	.7	.9	1	1
.3	.3	.3	.5	.7	.9	1
.1	.1	.1	.3	.5	.7	.9
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

Binary Cross Entropy (BCE) Loss

Categorical → Binary CE

	P_1	P_2	P_3	P_4	P_5
dog					
cat					
cow					
pig					

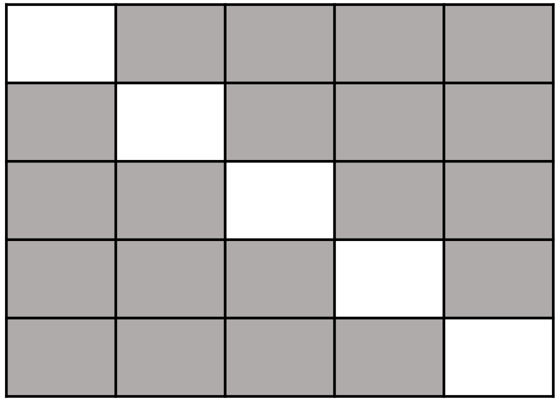


	Q_1	Q_2	Q_3	Q_4	Q_5
dog	0	0	.9	.1	1
cat	0	.9	.1	.9	0
cow	.1	.1	0	0	0
pig	.9	0	0	0	0

-log(0) ≈32
-log(-1) ≈ 3
-log(0.9) ≈ 0.2
-log(1) = 0

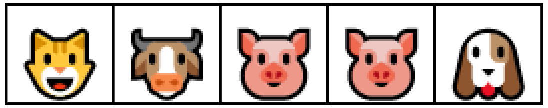
T

	dog	cat	cow	pig
P_1				
P_2				
P_3				
P_4				
P_5				



Binary CE Loss

	P_1	P_2	P_3	P_4	P_5
dog					
!dog					

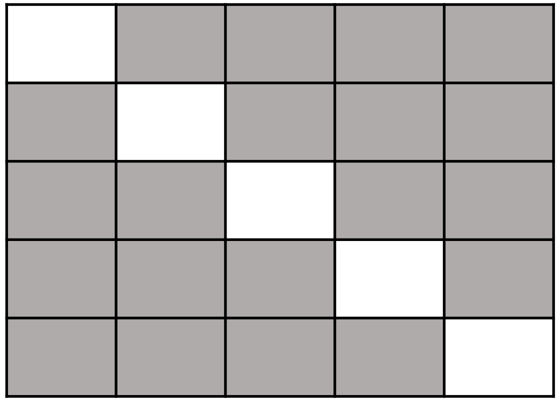


	Q_1	Q_2	Q_3	Q_4	Q_5
dog	0	0	.9	.1	1
!dog					

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.9) ≈ 0.2
-log(1) = 0

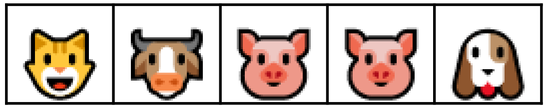
T

	dog	!dog
P_1		
P_2		
P_3		
P_4		
P_5		



Math: BCE Loss

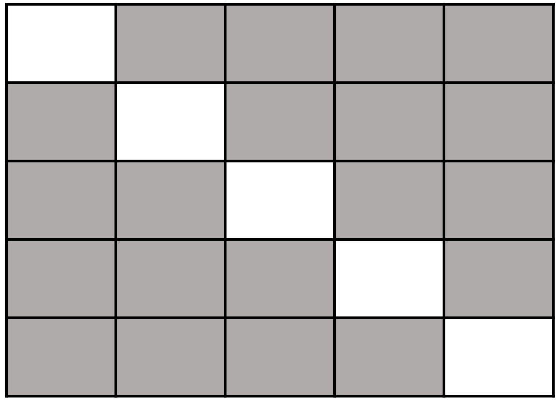
	P_1	P_2	P_3	P_4	P_5
dog	0	0	0	0	1
!dog					



	Q_1	Q_2	Q_3	Q_4	Q_5
dog	0	0	.9	.1	1
!dog					

T

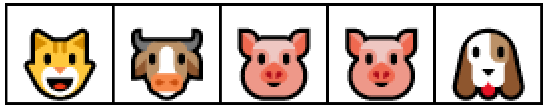
	dog	!dog
P_1	0	
P_2	0	
P_3	0	
P_4	0	
P_5	1	



BCE Loss

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.9) ≈ 0.2
-log(1) = 0

	P_1	P_2	P_3	P_4	P_5
pig	0	0	1	1	0
!pig	1	1	0	0	1



	Q_1	Q_2	Q_3	Q_4	Q_5
pig	0	1	.9	.1	0
!pig	1	0	0.1	0.9	1

x	x	0.2	3	x
0	32	x	x	0

T

	pig	!pig
P_1	0	1
P_2	0	1
P_3	1	0
P_4	1	0
P_5	0	1

a: 0				
	b: 32			
		c: 0.2		
			d: 3	
				e: 0

a+b+c+d+e =
35.2;

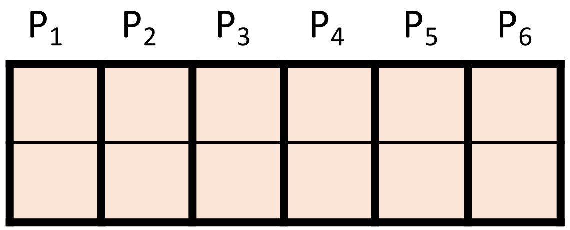
GAN's Loss

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

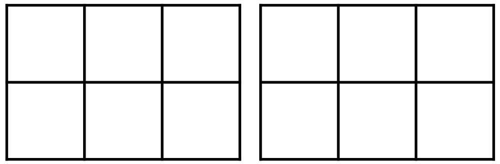
Discriminator Loss

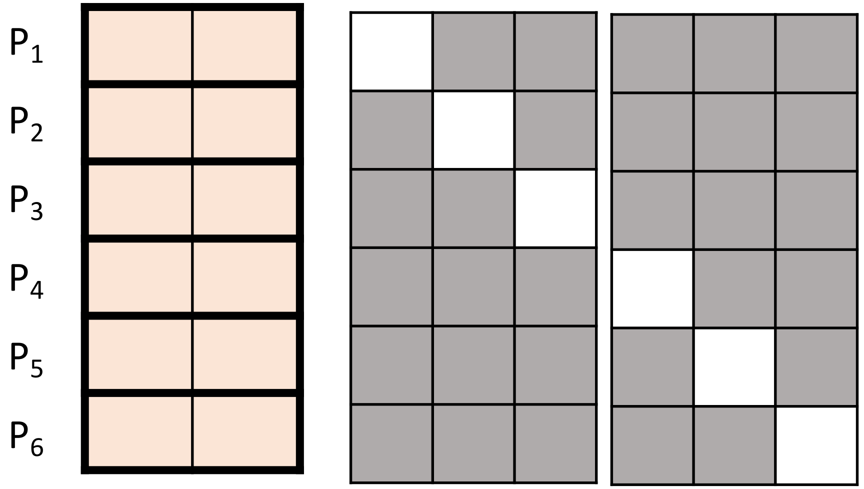
Fake			Real		
-1	-1	-3	0	0	-1
3	-1	3	3	-1	1



.7	0	.7	.9	.1	.3

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

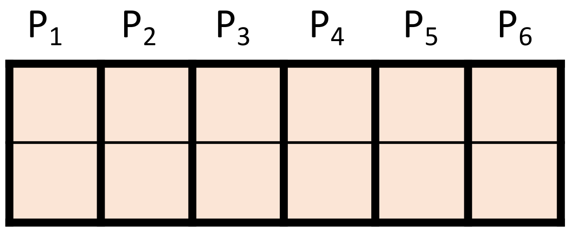




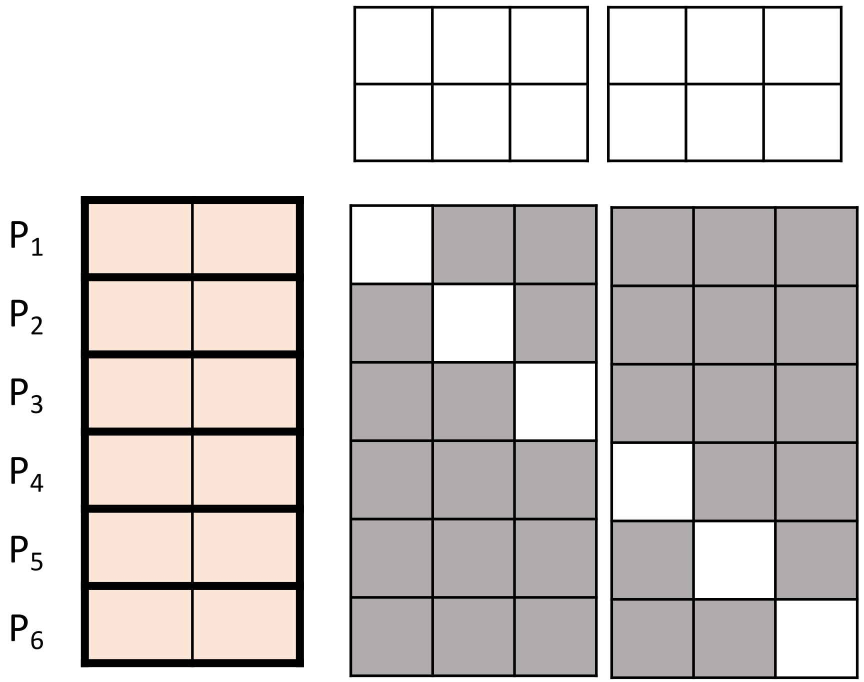
Generator Loss

Fake			Real		
-1	-1	-3	0	0	-1
3	-1	3	3	-1	1

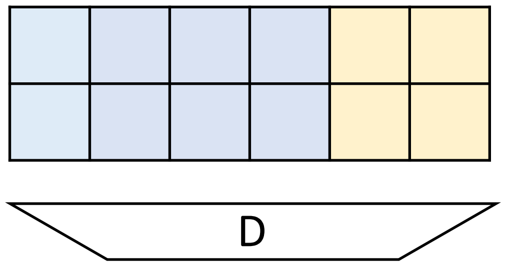
-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0



.7	0	.7	.9	.1	.3



Discriminator Loss



Fake Real

	P_1	P_2	P_3	P_4	P_5	P_6
real	0	0	0	0	1	1
!real	1	1	1	1	0	0

.3	.7	.5	.3	.1	1	Y
0.7	0.3	0.5	0.7	0.9	0	1-Y

x	x	x	x	3	0	-log
0.5	2	1	0.5	x	x	

	real	!real
P_1	0	1
P_2	0	1
P_3	0	1
P_4	0	1
P_5	1	0
P_6	1	0

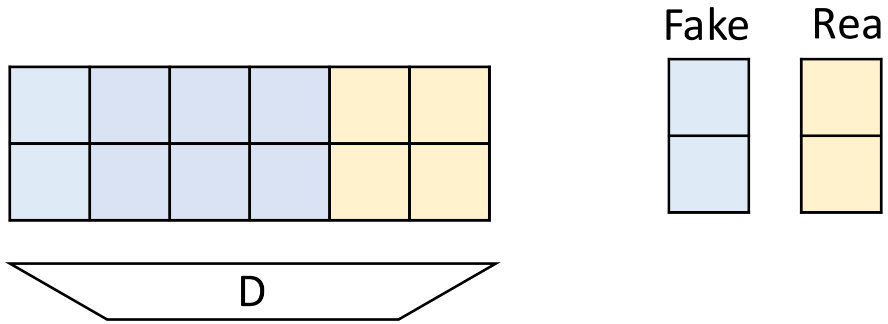
0.5	a					
	2	b				
		1	c			
			0.5	d		
				3	e	
					0	f

- $-\log(0) \approx 32$
- $-\log(0.1) \approx 3$
- $-\log(0.3) \approx 2$
- $-\log(0.5) \approx 1$
- $-\log(0.7) \approx 0.5$
- $-\log(0.9) \approx 0.2$
- $-\log(1) = 0$

a+b+c+d+e+f = 7;

Generator Loss

	P_1	P_2	P_3	P_4	P_5	P_6
real	1	1	1	1	0	0
!real	0	0	0	0	0	0



Y	.3	.7	.5	.3	.1	1
1-Y	0.7	0.3	0.5	0.7	0.9	0

-log	2	0.5	1	2	x	x
	x	x	x	x	x	x

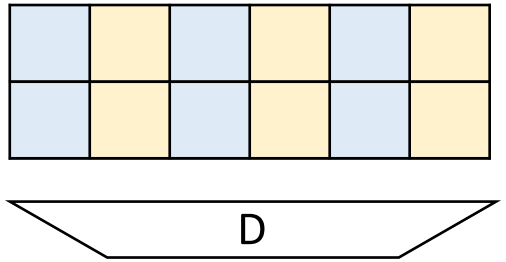
	real	!real
P_1	1	0
P_2	1	0
P_3	1	0
P_4	1	0
P_5	0	0
P_6	0	0

2	a					
	0.5	b				
		1	c			
			2	d		
				0	e	
					0	f

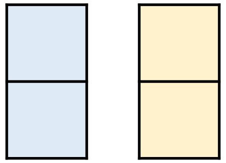
-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

a+b+c+d+e+f = 5.5;

Discriminator Loss



Fake Real



-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

	P_1	P_2	P_3	P_4	P_5	P_6
real	0	1	0	1	0	1
!real	1	0	1	0	1	0

.3	.1	.5	.9	.1	0	Y
0.7	0.9	0.5	0.1	0.9	1	1-Y

x	3	x	0.2	x	32	-log
0.5	x	1	x	0.2	x	

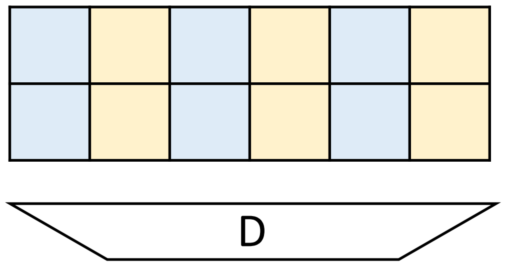
	real	!real
P_1	0	1
P_2	1	0
P_3	0	1
P_4	1	0
P_5	0	1
P_6	1	0

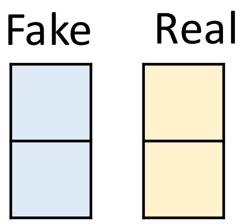
0.5 a					
	3 b				
		1 c			
			0.2 d		
				0.2 e	
					32 f

🔑 a+b+c+d+e+f =

36.9;

Generator Loss





-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

	P_1	~~P_2~~	P_3	~~P_4~~	P_5	~~P_6~~
real	1		1		1	
!real	0		0		0	

Y	.3	~~.1~~	.5	~~.9~~	.1	~~0~~
1-Y	0.7		0.5		0.9	

-log	2		1		3	
	x		x		x	

	real	!real
P_1	1	0
~~P_2~~		
P_3	1	0
~~P_4~~		
P_5	1	0
~~P_6~~		

2	a				
		b			
		1	c		
				d	
				3	e
					f

a+b+c+d+e+f = 6;

BCE Loss Gradients

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Update a parameter

*
*

1

1	0	1		
0	1	1		

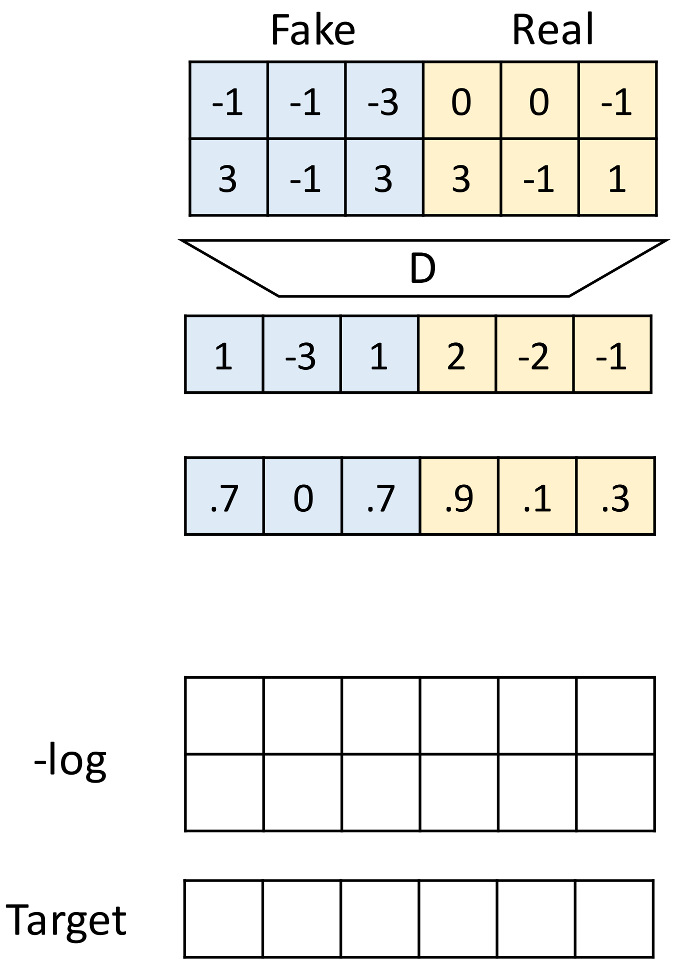
1	1	-3		

1	1	1	2	3	4	5
0	0	0	1	2	3	4
-1	-1	-1	0	1	2	3
-2	-2	-2	-1	0	1	2
-3	-3	-3	-2	-1	0	1
-3	-3	-3	-2	-1	0	1
-3	-3	-3	-2	-1	0	1

.7	.7	.7	.9	1	1	1
.5	.5	.5	.7	.9	1	1
.3	.3	.3	.5	.7	.9	1
.1	.1	.1	.3	.5	.7	.9
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

Fake			Real		
.7	0	.7	.9	.1	.3

Discriminator Loss Gradients



σ(<=-3) ≈ 0
σ(-2) ≈ 0.1
σ(-1) ≈ 0.3
σ(0) ≈ 0.5
σ(1) ≈ 0.7
σ(2) ≈ 0.9
σ(>=3) ≈ 1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Generator Loss Gradients

Fake			Real		
-1	-1	-3	0	0	-1
3	-1	3	3	-1	1

D

1	-3	1	2	-2	-1

.7	0	.7	.9	.1	.3

-log

Target

$\sigma(<=-3) \approx 0$
$\sigma(-2) \approx 0.1$
$\sigma(-1) \approx 0.3$
$\sigma(0) \approx 0.5$
$\sigma(1) \approx 0.7$
$\sigma(2) \approx 0.9$
$\sigma(>=3) \approx 1$

$-\log(0) \approx 32$
$-\log(0.1) \approx 3$
$-\log(0.3) \approx 2$
$-\log(0.5) \approx 1$
$-\log(0.7) \approx 0.5$
$-\log(0.9) \approx 0.2$
$-\log(1) = 0$

Discriminator Loss Gradients

D

	Fake			Real			
Z	-2	1	2	0	-1	3	
Pred	0.1	0.7	0.9	0.5	0.3	1	σ
-log							
Target	0	0	0	1	1	1	
$\frac{\partial L}{\partial z}$	0.1	0.7	0.9	-0.5	-0.7	0	a

- σ(<=-3) ≈ 0
- σ(-2) ≈ 0.1
- σ(-1) ≈ 0.3
- σ(0) ≈ 0.5
- σ(1) ≈ 0.7
- σ(2) ≈ 0.9
- σ(>=3) ≈ 1

- -log(0) ≈32
- -log(0.1) ≈ 3
- -log(0.3) ≈ 2
- -log(0.5) ≈ 1
- -log(0.7) ≈ 0.5
- -log(0.9) ≈ 0.2
- -log(1) = 0

|sum(a)| = .5

Generator Loss Gradients

	Fake			Real		

D

Z	-2	1	2	0	-1	3	
Pred	0.1	0.7	0.9				σ
-log							
Target	1	1	1				
$\frac{\partial L}{\partial z}$	-0.9	-0.3	-0.1				a

σ(<=-3) ≈ 0
σ(-2) ≈ 0.1
σ(-1) ≈ 0.3
σ(0) ≈ 0.5
σ(1) ≈ 0.7
σ(2) ≈ 0.9
σ(>=3) ≈ 1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

🔑 |sum(a)| = 1.3

Adversarial Training

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Train Discriminator

1	1	-1
1	-1	1

1 1 1

1	0	-2
0	2	1

Fake

-1	-1	-3
3	-1	3

1 1 1

Real

0	0	-1
3	-1	1

1 1 1

1	0	1
0	1	1

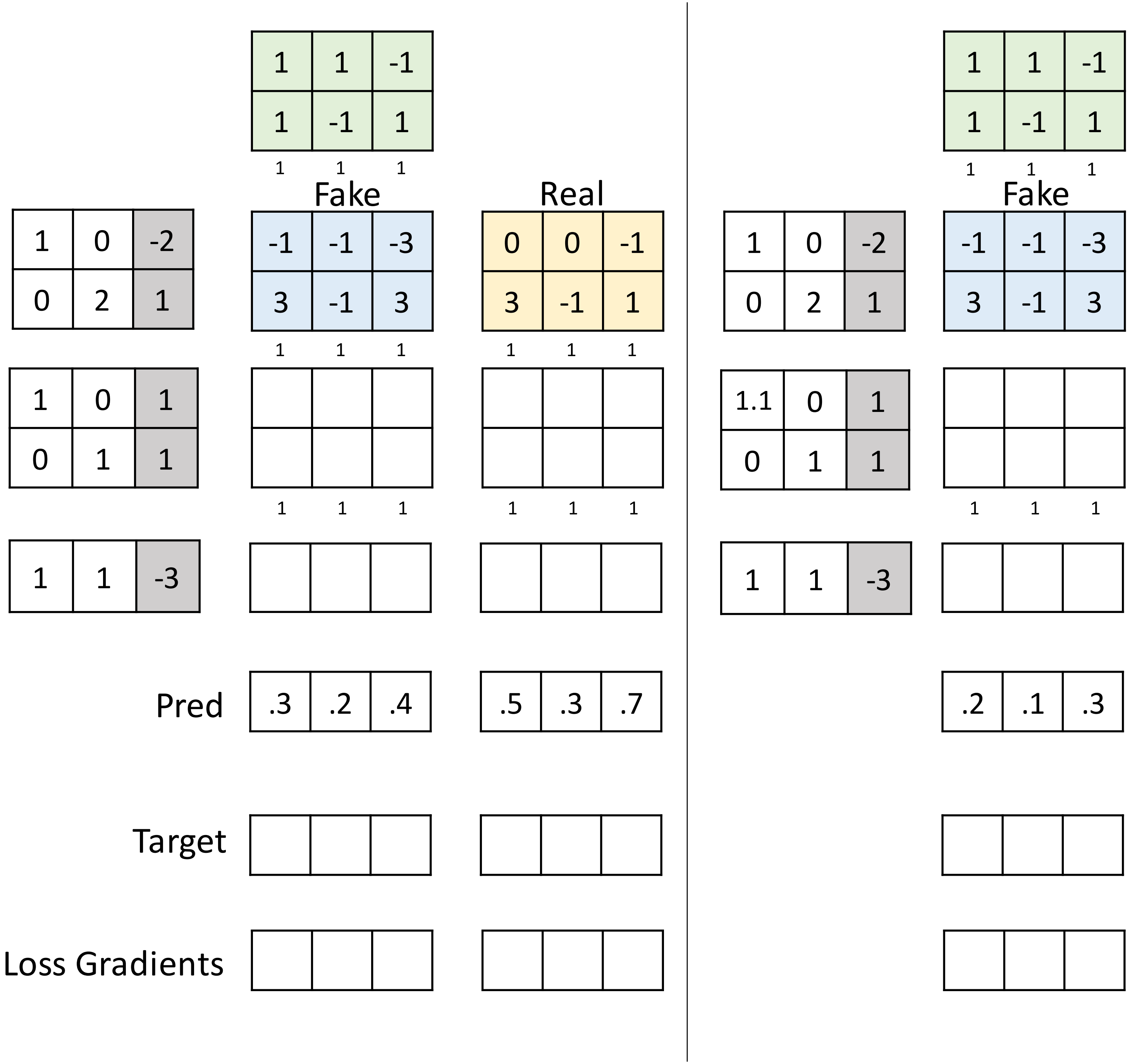
1 1 1

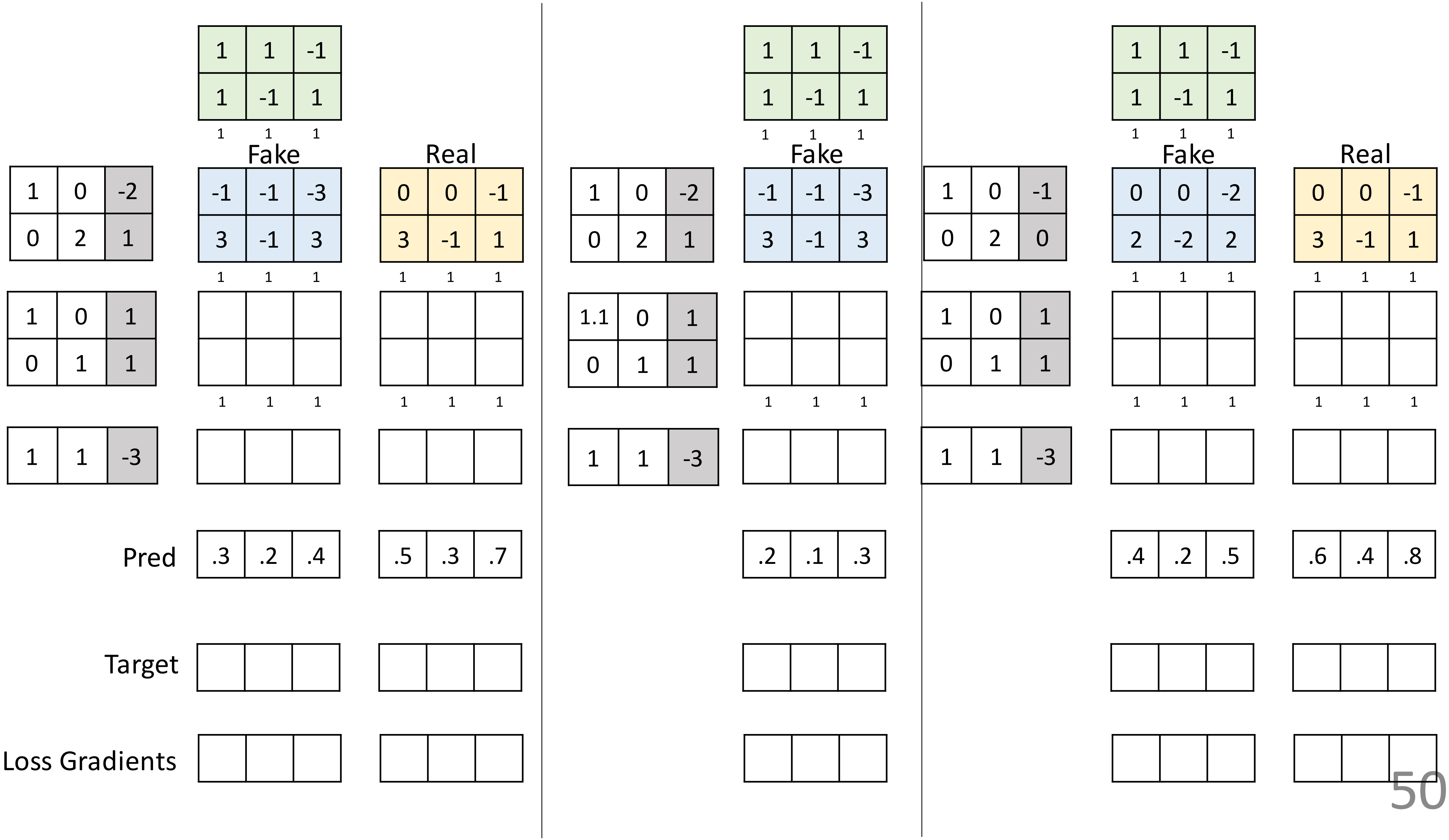
1 1 1

1	1	-3

Pred	.3	.2	.4	.5	.3	.7
Target						
Loss Gradients						

Train Generator





1 1 -1
1 -1 1
1 1 1
Fake
Real
1 0 -2
0 2 1
-1 -1 -3
3 -1 3
0 0 -1
3 -1 1
1 1 1
1 1 1
1 0 1
0 1 1
1 1 1
1 1 1
1 1 -3
Pred
.3 .2 .4
.5 .3 .7
Target
Loss Gradients
1 1 -1
1 -1 1
1 1 1
Fake
1 0 -2
0 2 1
-1 -1 -3
3 -1 3
1.1 0 1
0 1 1
1 1 1
1 1 -3
.2 .1 .3
1 1 -1
1 -1 1
1 1 1
Fake
Real
1 0 -1
0 2 0
0 0 -2
2 -2 2
0 0 -1
3 -1 1
1 1 1
1 1 1
1 0 1
0 1 1
1 1 1
1 1 1
1 1 -3
.4 .2 .5
.6 .4 .8

ROC Curve

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Discriminator = Binary Classifier

1*

1	0	1
0	1	1

1*

1	1	-3

Pred	.7	.1	.5	.7	.1	.3	.7	0	0	.1
Target	1	1	0	1	0	1	0	1	0	0

Threshold vs. TPR / FPR

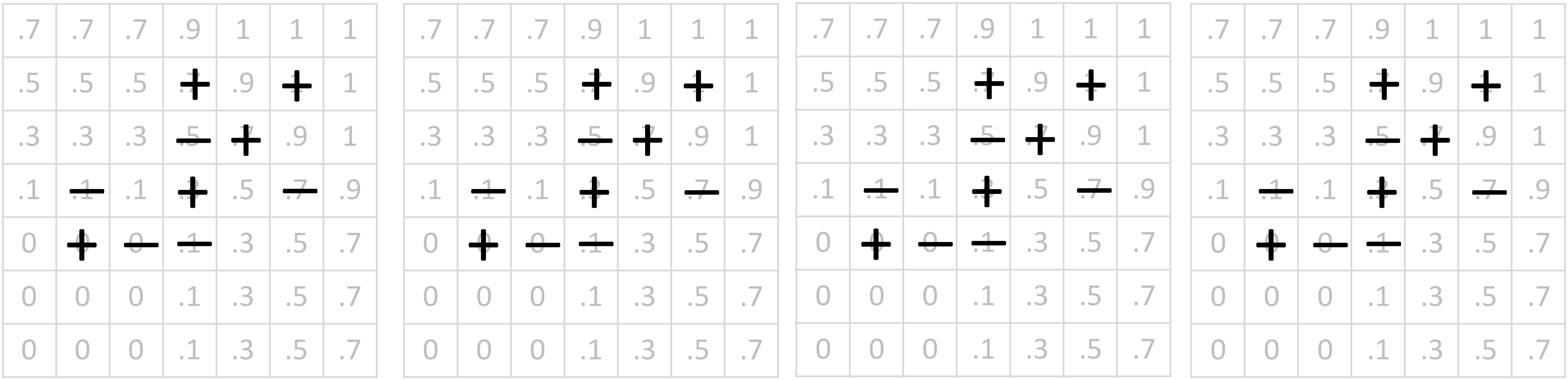
.7	.7	.7	.9	1	1	1
.5	.5	.5	+	.9	+	1
.3	.3	.3	−	+	.9	1
.1	−	.1	+	.5	−	.9
0	+	−	−	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

.7	.7	.7	.9	1	1	1
.5	.5	.5	+	.9	+	1
.3	.3	.3	−	+	.9	1
.1	−	.1	+	.5	−	.9
0	+	−	−	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

.7	.7	.7	.9	1	1	1
.5	.5	.5	+	.9	+	1
.3	.3	.3	−	+	.9	1
.1	−	.1	+	.5	−	.9
0	+	−	−	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

.7	.7	.7	.9	1	1	1
.5	.5	.5	+	.9	+	1
.3	.3	.3	−	+	.9	1
.1	−	.1	+	.5	−	.9
0	+	−	−	.3	.5	.7
0	0	0	.1	.3	.5	.7
0	0	0	.1	.3	.5	.7

Threshold vs. TPR / FPR



>=.7

.5	.5	.7 +	.9	.7
.3	.3 −	.5 −	.7 +	.5
.1	.1	.3 +	.5 +	.3 −
0	0 −	.1 −	.3	.1
0	0	0	.1	0

a

TPR = __________

d

FPR = __________

>=.5

.5	.5	.7 +	.9	.7
.3	.3 −	.5 −	.7 +	.5
.1	.1	.3 +	.5 +	.3 −
0	0 −	.1 −	.3	.1
0	0	0	.1	0

b

TPR = __________

d

FPR = __________

>=.3

.5	.5	.7 +	.9	.7
.3	.3 −	.5 −	.7 +	.5
.1	.1	.3 +	.5 +	.3 −
0	0 −	.1 −	.3	.1
0	0	0	.1	0

c

TPR = __________

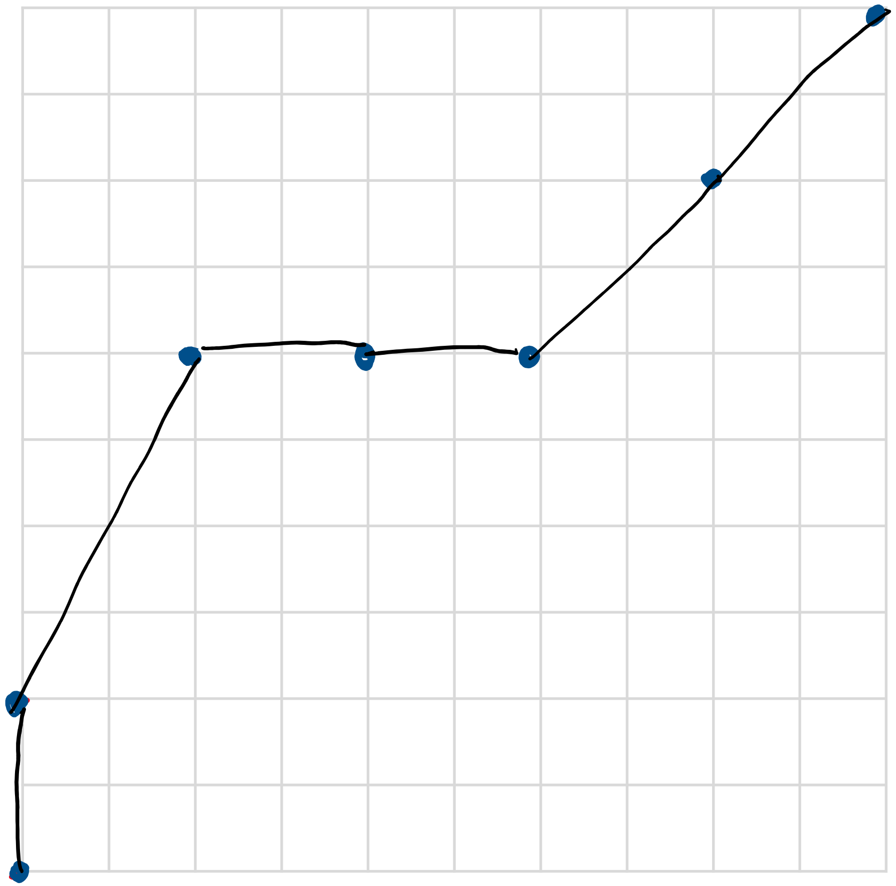
f

FPR = __________

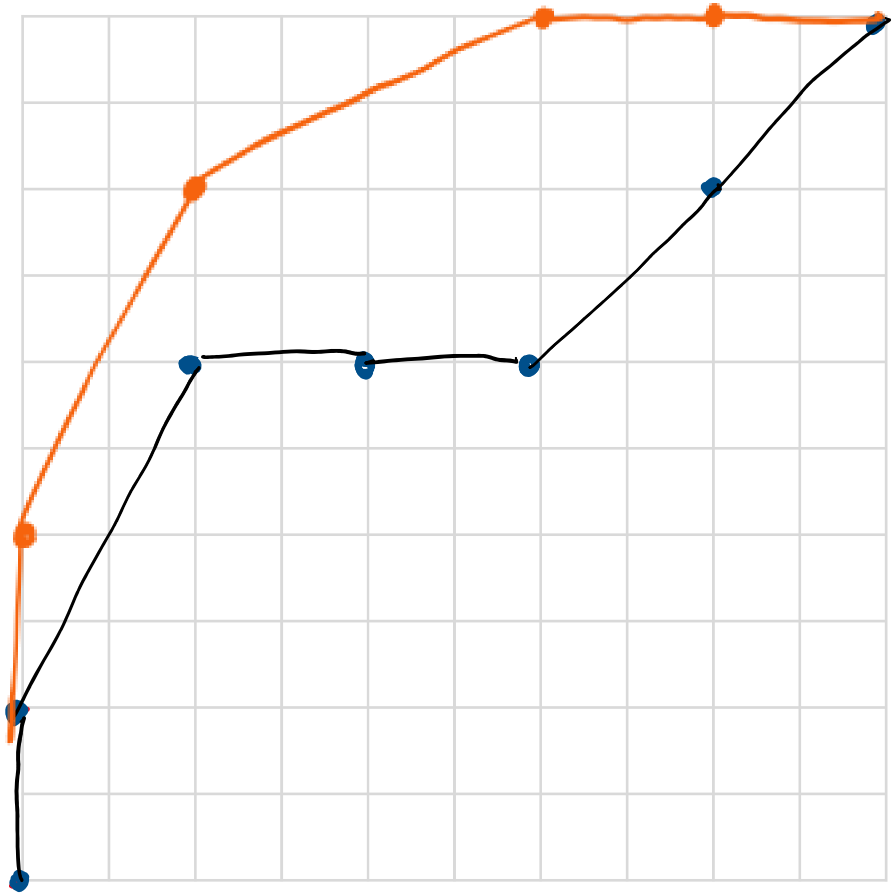
🔑 a+b+c = 2.25; d+e+f = 1

Receiver Operating Characteristic (ROC) Curve

Area Under ROC Curve (AUC)



Calculate the improvement in AUC

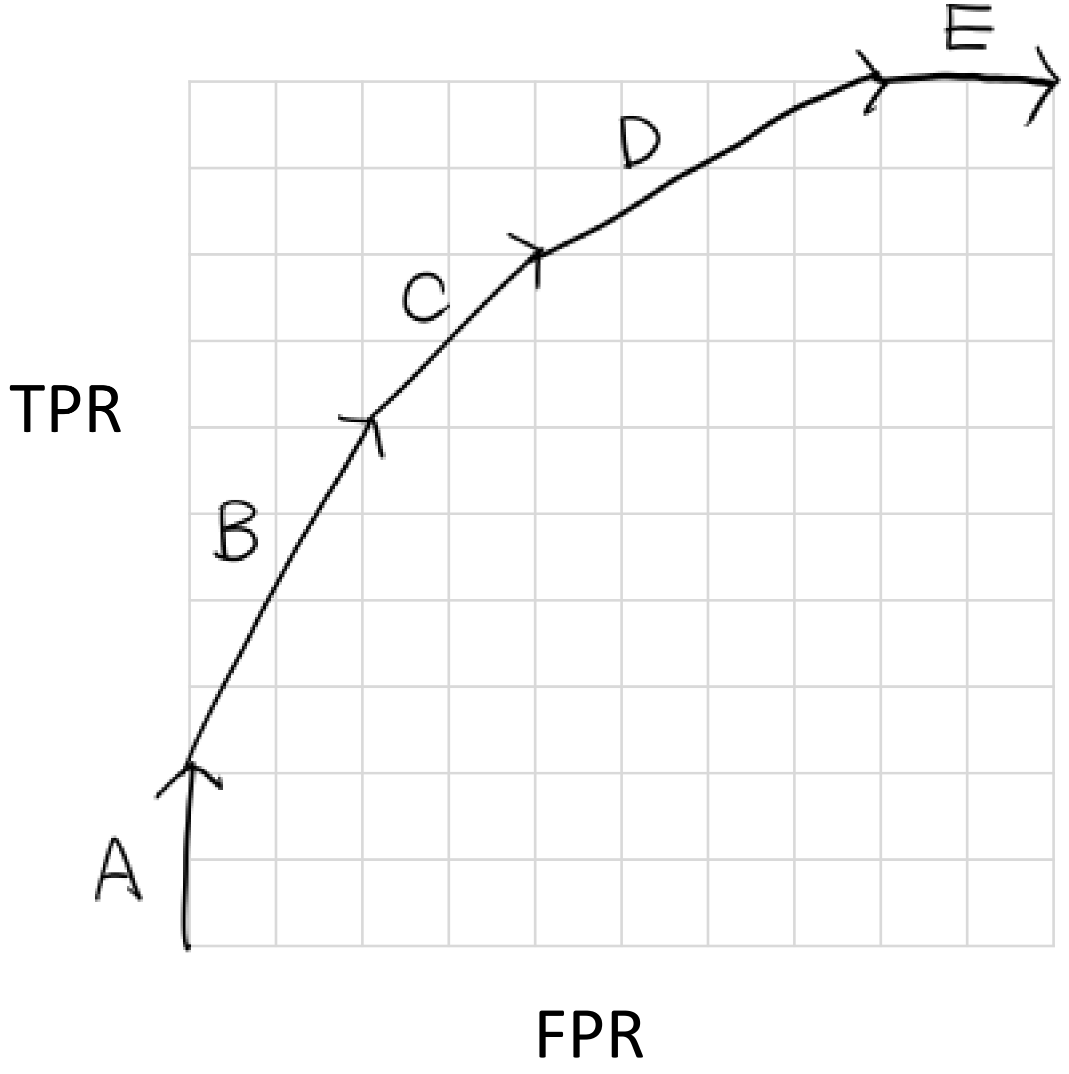


a

Answer = ________________

a mod 7 = 3

Match each movement segment to the correct interpretation of the changes in TP and FP



1. More true positives but at the cost of a lot more false positives.
2. More true positives without any new false positives.
3. No more true positives, just more false positives.
4. More true positives, but at the cost of about the same number of false positives.
5. More true positives at the cost of additional, but relatively fewer false positives.

NumPy by Hand ✍️ [Entropy]

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Cross Entropy by @

1. g = -1 * np.log(q)
2. pt = np.transpose(p)
3. CE = pt @ g

p	q
.7	.3
0	.5
.3	.1
0	.1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Cross Entropy by *

1. g = -1 * np.log(q)
2. pg = p * g
3. CE = np.sum(b)

p	q
.7	.3
0	.5
.3	.1
0	.1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Categorical CE

1. g = -1 * np.log(q)
2. pt = np.transpose(p)
3. CE = pt @ g

p	q
0	.3
1	.5
0	.1
0	.1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Categorical CE

1. g = -1 * np.log(q)
2. pg = p * g
3. CE = np.sum(pg)

p	q
0	.3
1	.5
0	.1
0	.1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Batch Categorical CE

1. g = -1 * np.log(q)
2. pg = p * g
3. CE = np.sum(pq)

q

.3	.1	.1
.5	.5	.1
.1	.3	.5
.1	.1	.3

p

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

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Batch Categorical CE

```
1.  q1 = q[c, [0,1,2]]
2.  g = np.log(q1)
3.  CE = np.sum(g)
```

q

.3	.1	.1
.5	.5	.1
.1	.3	.5
.1	.1	.3

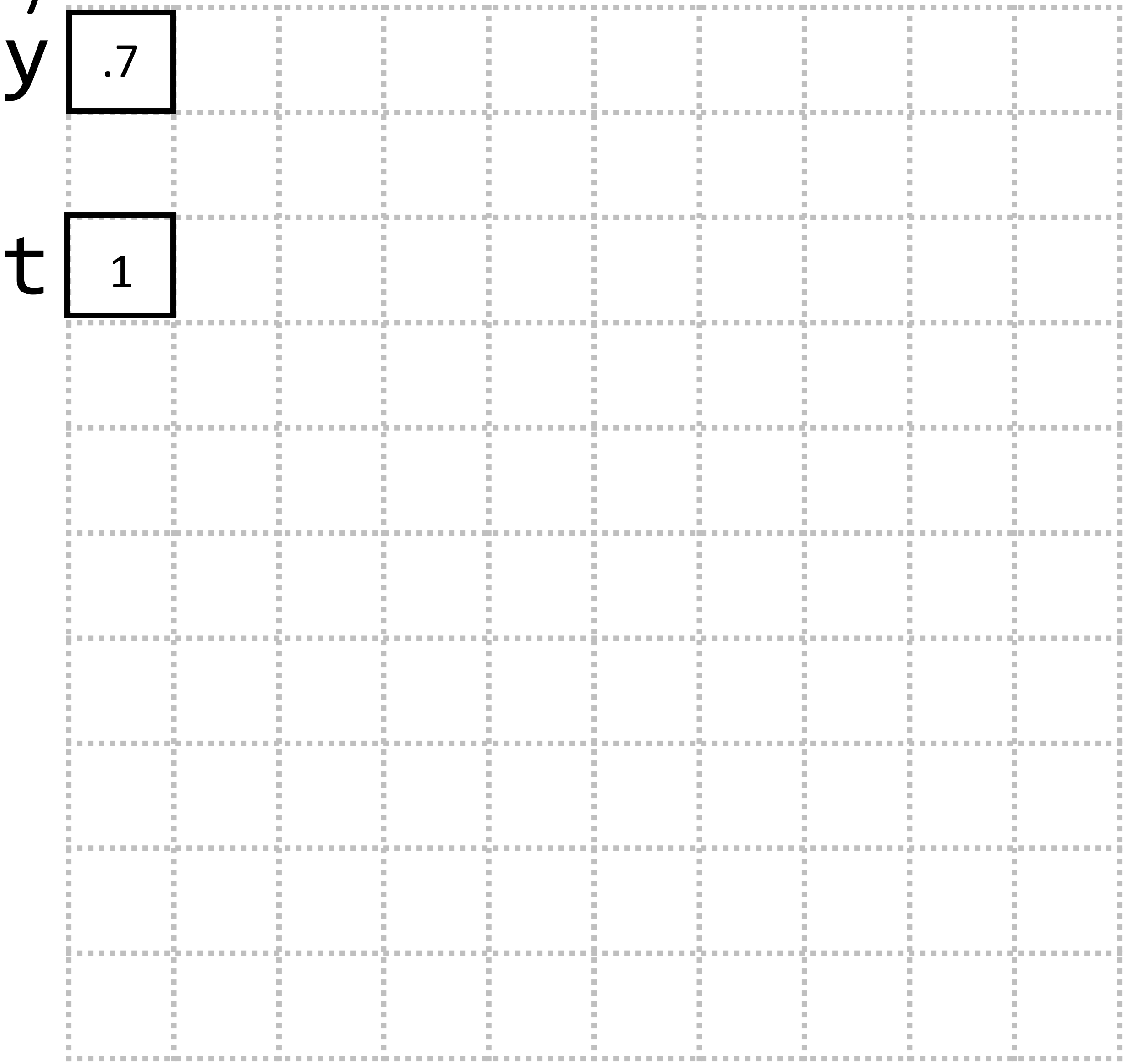
c

0	1	3

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Binary Cross Entropy

1. q = np.vstack((y, 1 – y))
2. p = np.vstack((t, 1-t))
3. g = -np.log(q)
4. pg = p * g
5. CE = np.sum(pg)



-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

Batch BCE

1. q = np.vstack((y, 1 – y))
2. p = np.vstack((t, 1-t))
3. g = -np.log(q)
4. pg = p * g
5. CE = np.sum(pg)

y	.7	.3	.9
t	1	0	1

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

BCE

1. q = np.vstack((y, 1 – y))
2. p = np.vstack((t, 1-t))
3. g = -np.log(q)
4. pg = p * g
5. CE = np.sum(pg)

-log(0) ≈32
-log(0.1) ≈ 3
-log(0.3) ≈ 2
-log(0.5) ≈ 1
-log(0.7) ≈ 0.5
-log(0.9) ≈ 0.2
-log(1) = 0

y	.3	.1	.7	.9	0	1
t	1	0	0	0	1	1
q						
p						
g						
pg						

CE

Show your work

CE * 10 % 7 = 3