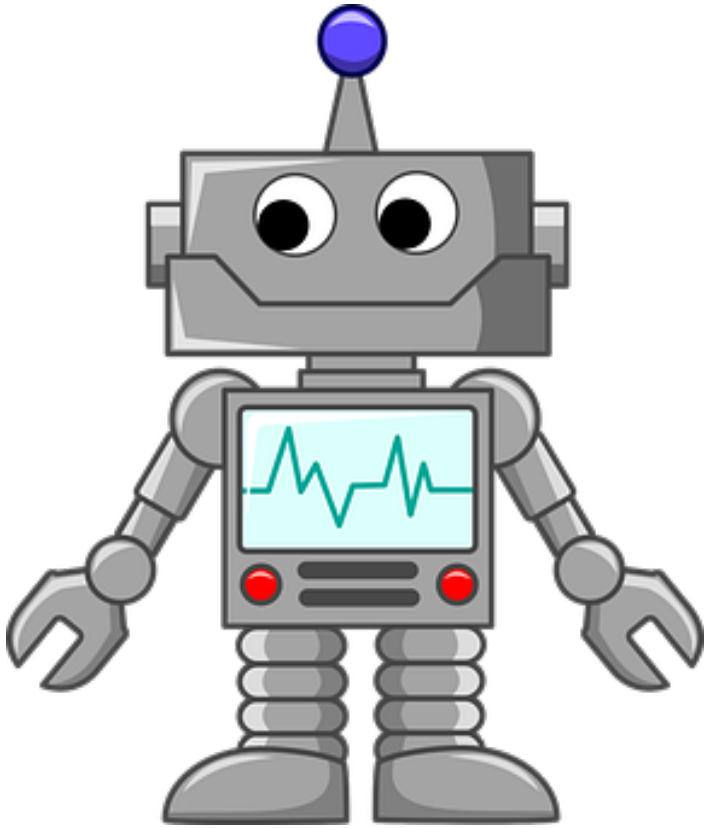
Artificial Intelligence, Image Recognition, and The Birth of the Robots!

Chris Deotte, UCSD
October 30, 2018



Remember the year


Robots Beat Humans
2015
Classifying
Images
ImageNet Comp

2016

World's best Go player flummoxed by Google's 'godlike' AlphaGo AI

Ke Jie, who once boasted he would never be beaten by a computer at the ancient Chinese game, said he had 'horrible experience'

2017

Superhuman AI for heads-up no-limit poker: Libratus beats top professionals

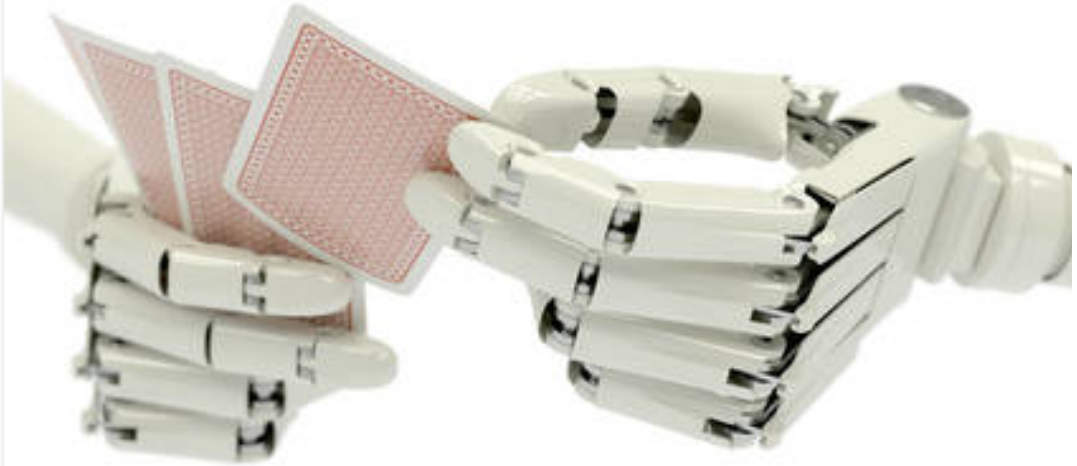
Science

Andrew Ng ✓
@AndrewYNg

CMU just made history: AI beats top humans at Texas Hold'em poker. A stunning accomplishment, comparable to Deep Blue & AlphaGo!

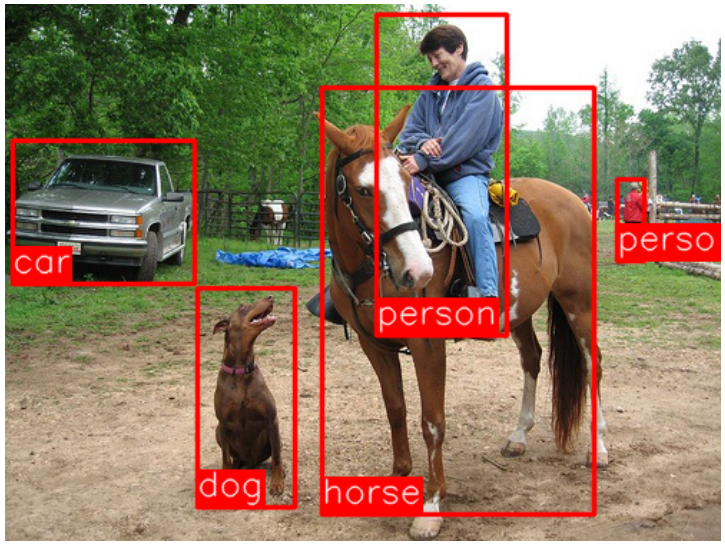
10:42 PM - Jan 30, 2017 · Mountain View, CA

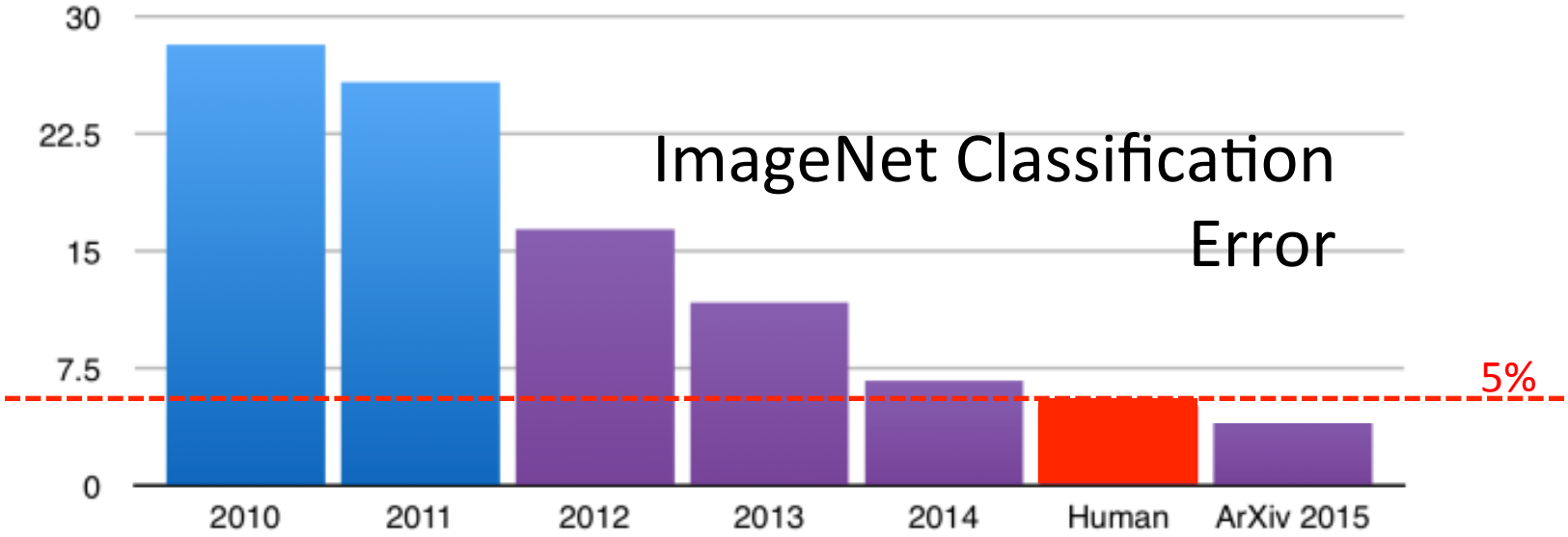
♡ 1,857 💬 1,507 people are talking about this



History of Image Recognition

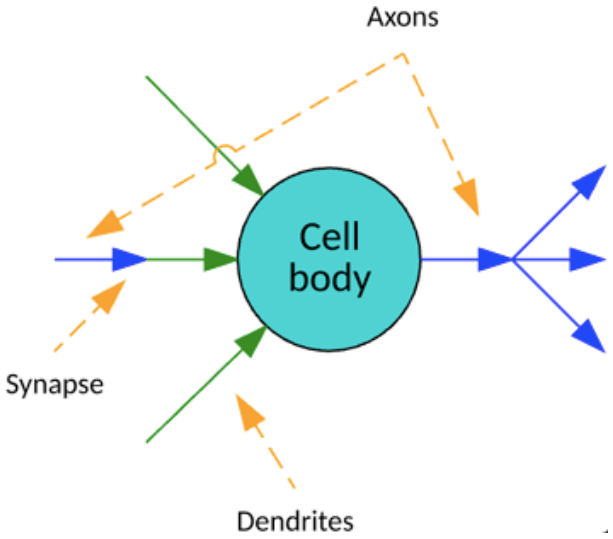
1995: MNIST released (dataset)
1998: LeNet5 classifies MNIST (CNN technique)
2007: Nvidia releases CUDA (GPU computing)
2009: GPUs recognized (GPU computing)
2010: ImageNet released (big dataset)
2012: Deep Learning Revolution! (DNN technique)
2014: Nvidia releases cuDNN (GPU computing)
2015: Machine learning beats humans
(DNN technique, data, GPU computing)





Q: How did robots beat us in 2015?

A: Deep Learning, Big Data, GPU Computing



Talk Outline:

2000-2012: Birth of Convolutional Neural Networks

- MNIST dataset (handwritten digits)
- Pattern Recognition
- Linear classifiers
- Neural Networks (Non-linear classifiers)
- Convolutional Neural Networks

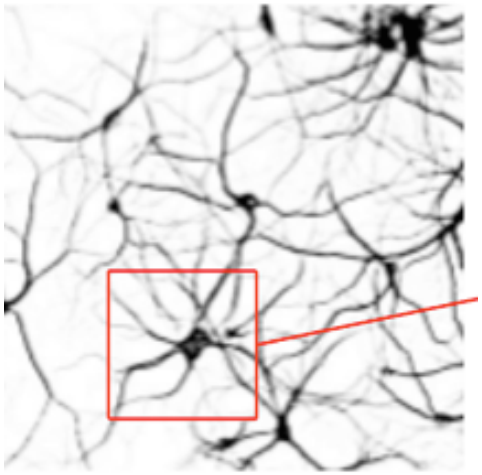
2012: AlexNet (Deep Neural Networks + GPU)

2013: ZF Net

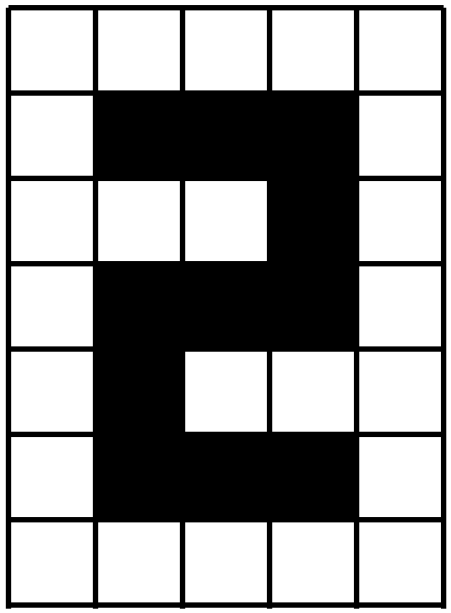
2014: VGG Net

2014: GoogLeNet

2015: Microsoft ResNet (Super Deep Neural Networks!)



2000: MNIST dataset is 70,000 handwritten digits 28x28 pixels



$$\bar{x}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{784}$$

(for 28 x 28)

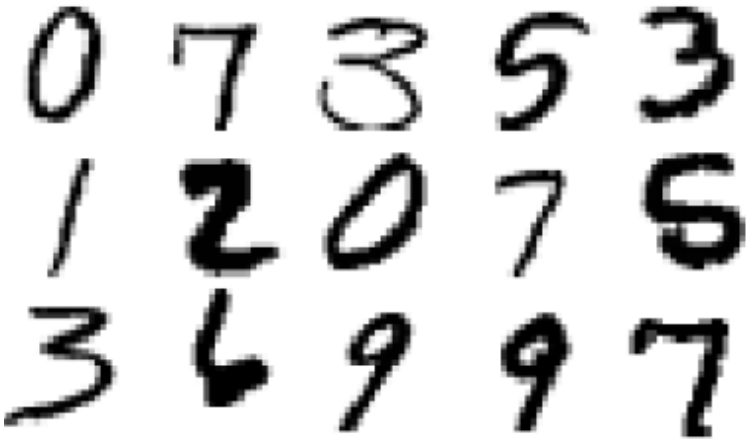
$$y_i = 2 \in \mathbb{R}$$

To classify, find function *f*

$$f : \mathbb{R}^{784} \rightarrow \mathbb{R}$$

specifically

$$f : [0,1]^{784} \rightarrow \{0, 1, 2, ..., 9\}$$

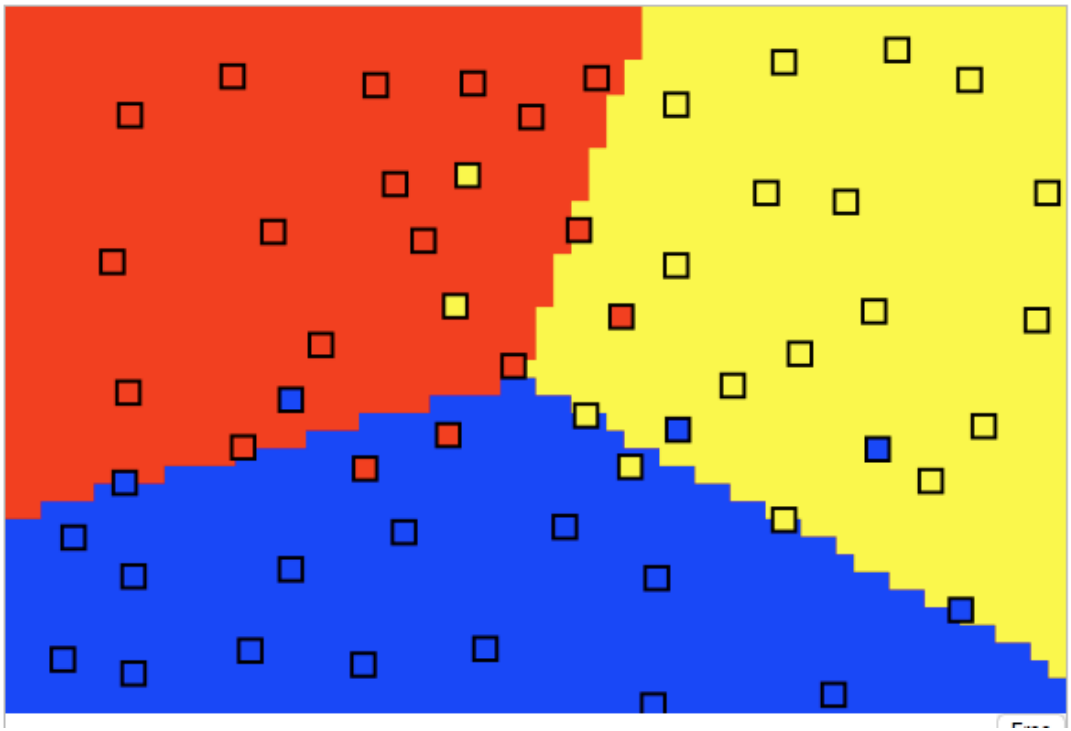


First Attempt: **Linear Classifier**

(logistic regression, linear SVM, naive Bayes)

Given $\bar{w}_0, \bar{w}_1, \bar{w}_2 ..., \bar{w}_9 \in \mathbb{R}^{784}$ and $b_0, b_1, b_2, ..., b_9 \in \mathbb{R}$

Classify $\bar{x}_i \in \mathbb{R}^{784}$ as $\hat{y}_i = \underset{0 \leq k \leq 9}{\operatorname{argmax}}(b_k + \bar{w}_k \cdot \bar{x}_i)$



2D Example

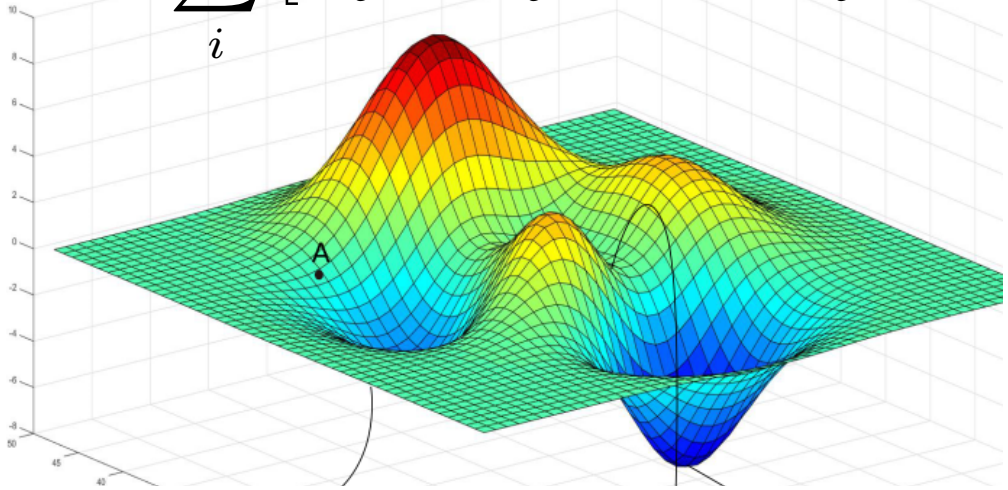
$\bar{x}_i \in \mathbb{R}^2$

$y_i \in \{0, 1, 2\}$

First Attempt: **How do we find *w*'s and *b*'s?**

Find $\bar{w}_0, \bar{w}_1, \bar{w}_2 ..., \bar{w}_9 \in \mathbb{R}^{784}$ and $b_0, b_1, b_2, ..., b_9 \in \mathbb{R}$ using stochastic **gradient descent** to minimize a cost (error) function. For example, a simple cost function is

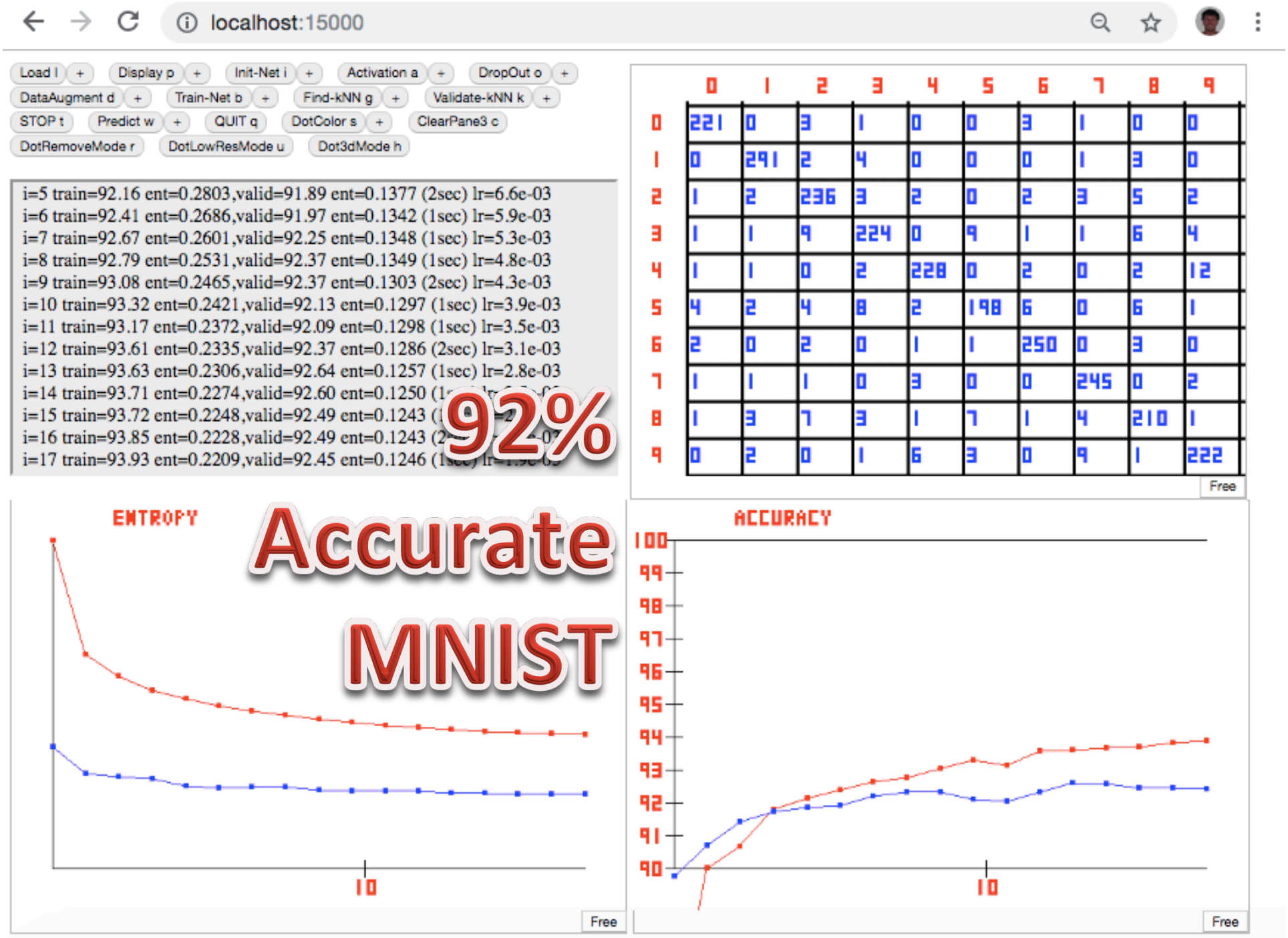
$$C(x, y, w, b) = \sum_i \left[(b_{\hat{y}_i} + \bar{w}_{\hat{y}_i} \cdot \bar{x}_i) - (b_{y_i} + \bar{w}_{y_i} \cdot \bar{x}_i) \right]$$



Each epoch, iterate through all $\bar{x}_i$ and update all $\bar{w}_k$ and b_k

$\bar{w}_k = \bar{w}_k - \eta \nabla_{\bar{w}_k} C(\bar{x}_i, y_i, w, b)$ for all $k = 0, 1, 2, ..., 9$

where η is the learning rate.

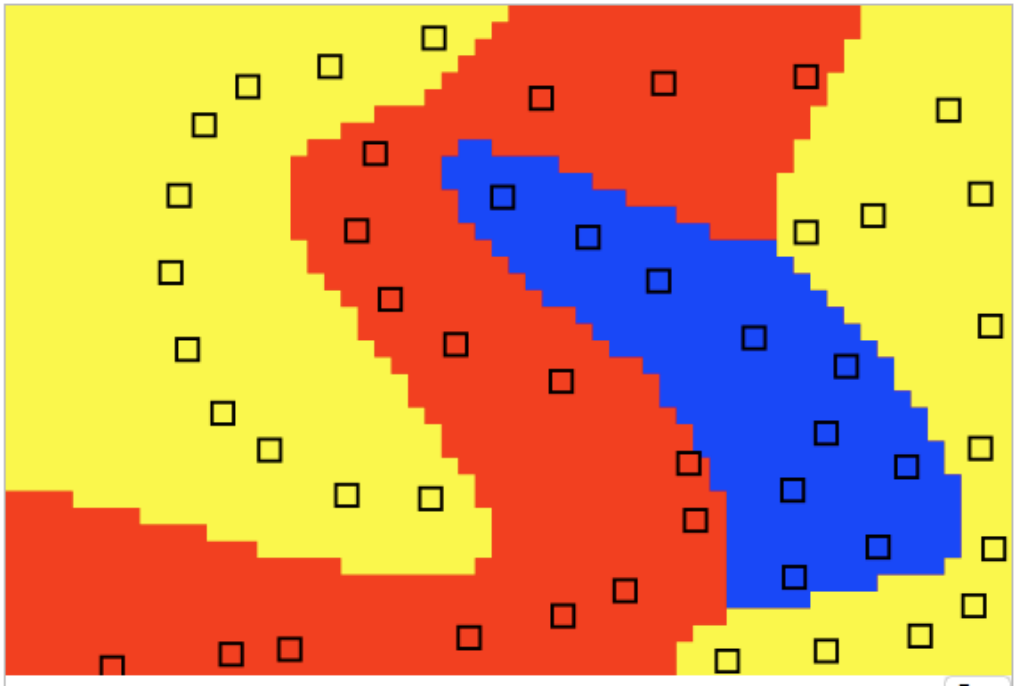

localhost:15000
Load l + Display p + Init-Net i + Activation a + DropOut o +
DataAugment d + Train-Net b + Find-kNN g + Validate-kNN k +
STOP t Predict w + QUIT q DotColor s + ClearPane3 c
DotRemoveMode r DotLowResMode u Dot3dMode h
i=5 train=92.16 ent=0.2803,valid=91.89 ent=0.1377 (2sec) lr=6.6e-03
i=6 train=92.41 ent=0.2686,valid=91.97 ent=0.1342 (1sec) lr=5.9e-03
i=7 train=92.67 ent=0.2601,valid=92.25 ent=0.1348 (1sec) lr=5.3e-03
i=8 train=92.79 ent=0.2531,valid=92.37 ent=0.1349 (1sec) lr=4.8e-03
i=9 train=93.08 ent=0.2465,valid=92.37 ent=0.1303 (2sec) lr=4.3e-03
i=10 train=93.32 ent=0.2421,valid=92.13 ent=0.1297 (1sec) lr=3.9e-03
i=11 train=93.17 ent=0.2372,valid=92.09 ent=0.1298 (1sec) lr=3.5e-03
i=12 train=93.61 ent=0.2335,valid=92.37 ent=0.1286 (2sec) lr=3.1e-03
i=13 train=93.63 ent=0.2306,valid=92.64 ent=0.1257 (1sec) lr=2.8e-03
i=14 train=93.71 ent=0.2274,valid=92.60 ent=0.1250
i=15 train=93.72 ent=0.2248,valid=92.49 ent=0.1243
i=16 train=93.85 ent=0.2228,valid=92.49 ent=0.1243
i=17 train=93.93 ent=0.2209,valid=92.45 ent=0.1246
92%
Accurate
MNIST
0 1 2 3 4 5 6 7 8 9
0 221 0 3 1 0 0 3 1 0 0
1 0 291 2 4 0 0 0 1 3 0
2 1 2 236 3 2 0 2 3 5 2
3 1 1 9 224 0 9 1 1 6 4
4 1 1 0 2 228 0 2 0 2 12
5 4 2 4 8 2 198 6 0 6 1
6 2 0 2 0 1 1 250 0 3 0
7 1 1 1 0 3 0 0 245 0 2
8 1 3 7 3 1 7 1 4 210 1
9 0 2 0 1 6 3 0 9 1 222
Free
ENTROPY
10
Free
ACCURACY
100
99
98
97
96
95
94
93
92
91
90
10
Free

Second Attempt: **Nonlinear Classifier**

(neural network, nonlinear SVM, kNN, CART, random forest, boosted trees)

Given $\bar{w}_0, \bar{w}_1, \bar{w}_2 ..., \bar{w}_9 \in \mathbb{R}^n$ and $b_0, b_1, b_2, ..., b_9 \in \mathbb{R}$
and nonlinear function $f : \mathbb{R}^{784} \to \mathbb{R}^n$

Classify $\bar{x}_i \in \mathbb{R}^{784}$ as $\hat{y}_i = \underset{0 \leq k \leq 9}{\operatorname{argmax}} \left(b_k + \bar{w}_k \cdot f(\bar{x}_i) \right)$



2D Example

$\bar{x}_i \in \mathbb{R}^2$

$y_i \in \{0, 1, 2\}$

Second Attempt: **How do we find *f*, *w*'s, and *b*'s?**

First, define *f* using *m* parameters $p_0, p_1, p_2, \ldots, p_m \in \mathbb{R}$

Then, find $\bar{w}_0, \bar{w}_1, \bar{w}_2 ..., \bar{w}_9 \in \mathbb{R}^n$ and $b_0, b_1, b_2, ..., b_9 \in \mathbb{R}$ and $p_0, p_1, p_2, \ldots, p_m \in \mathbb{R}$ using stochastic **gradient descent** to minimize a cost function. For example, a simple cost function is

$$C(x, y, w, b, p) = \sum_i \left[(b_{\hat{y}_i} + \bar{w}_{\hat{y}_i} \cdot f_p(\bar{x}_i)) - (b_{y_i} + \bar{w}_{y_i} \cdot f_p(\bar{x}_i)) \right]$$

Each epoch, iterate through all $\bar{x}_i$ and update all $\bar{w}_k$, b_k , and p_k

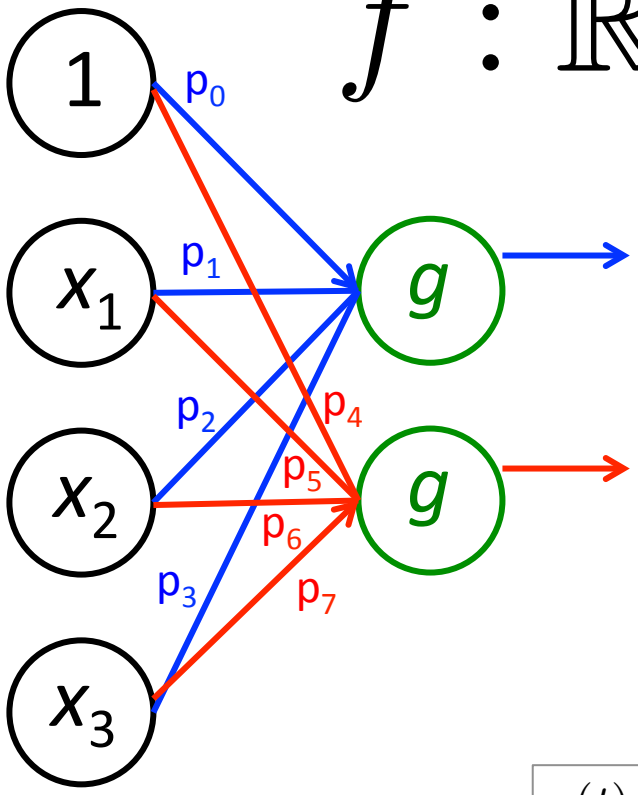
$$\bar{w}_k = \bar{w}_k - \eta \nabla_{\bar{w}_k} C(\bar{x}_i, y_i, w, b, p) \text{ for all } k = 0, 1, 2, ..., 9$$

$$p_k = p_k - \eta \frac{\partial}{\partial p_k} C(\bar{x}_i, y_i, w, b, p)$$

where η is the learning rate.

Second Attempt: **Use a neural network for function *f***

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

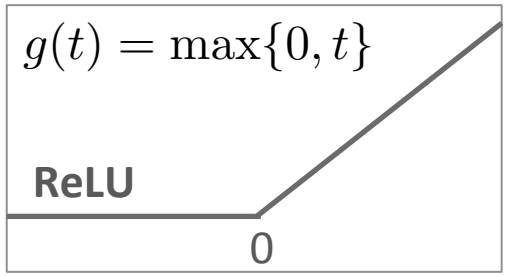


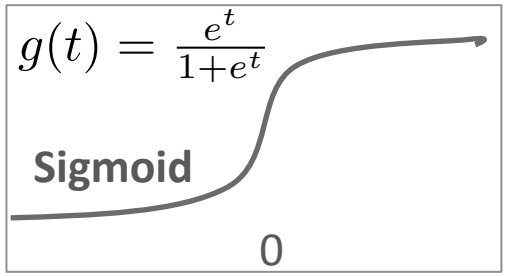
$$y_1 = g(p_0 + x_1 p_1 + x_2 p_2 + x_3 p_3)$$

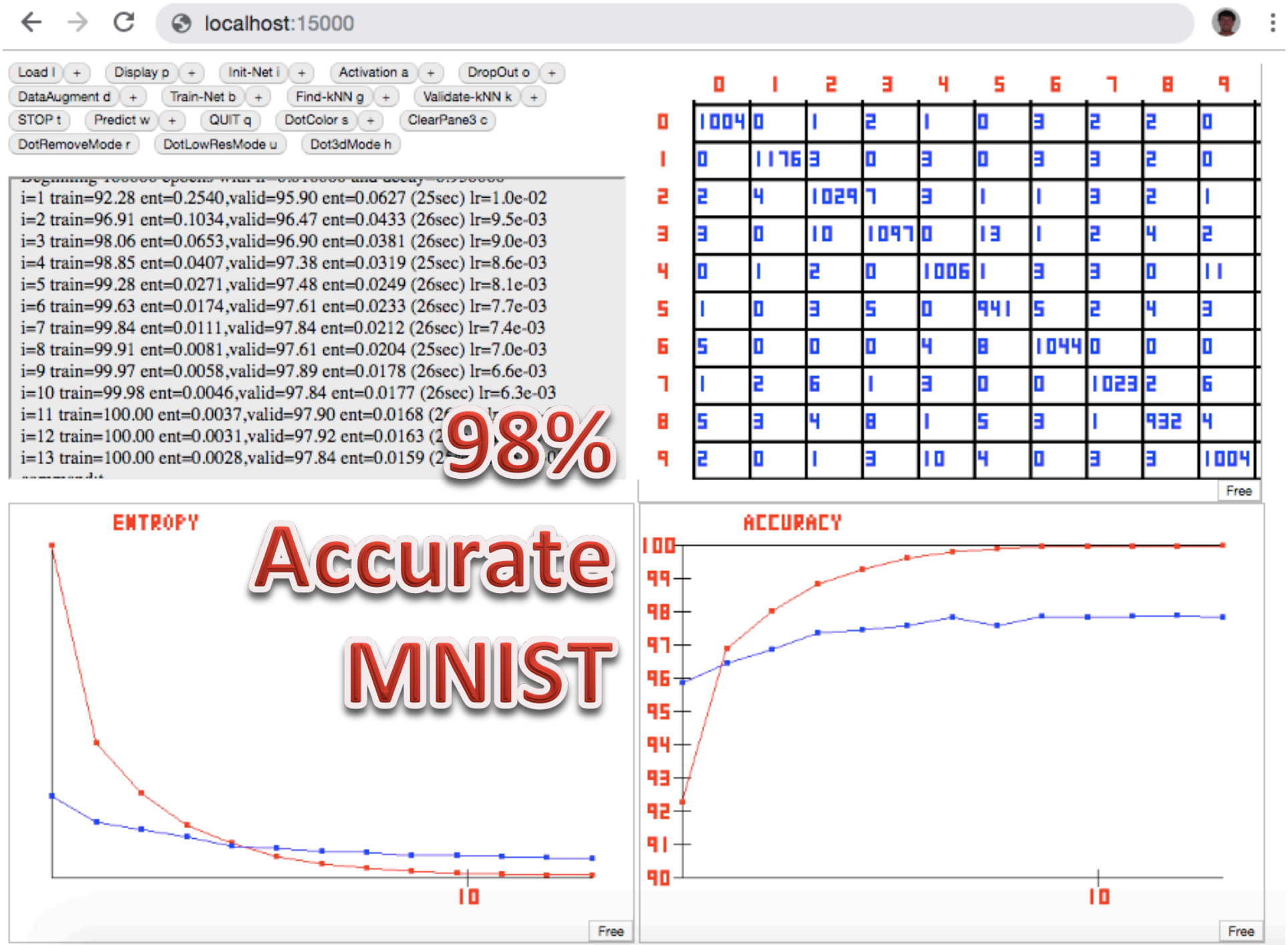
$$y_2 = g(p_4 + x_1 p_5 + x_2 p_6 + x_3 p_7)$$

Let g be a nonlinear step like function.

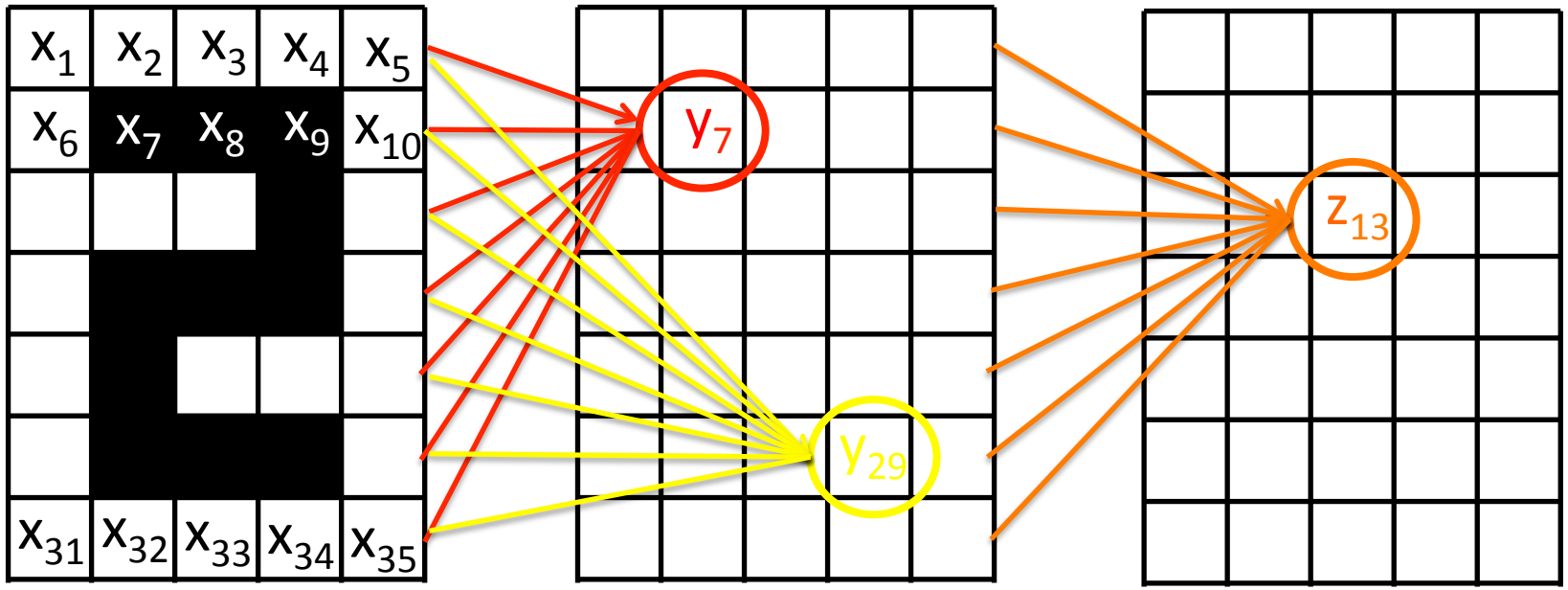
$$[x_1, x_2, x_3]^T \rightarrow [y_1, y_2]^T$$

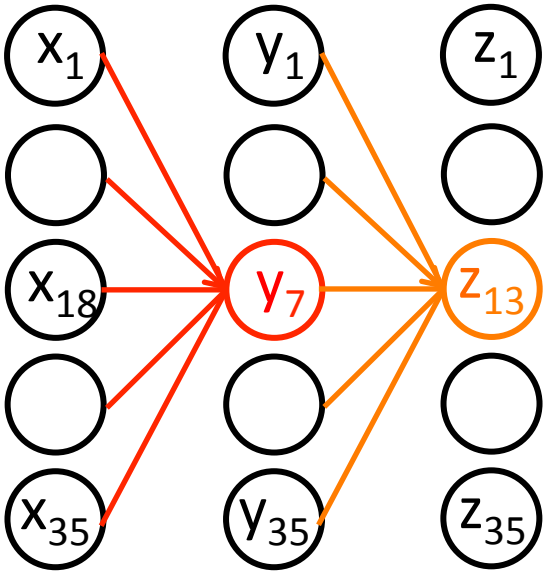





localhost:15000
Load l + Display p + Init-Net i + Activation a + DropOut o +
DataAugment d + Train-Net b + Find-kNN g + Validate-kNN k +
STOP t Predict w + QUIT q DotColor s + ClearPane3 c
DotRemoveMode r DotLowResMode u Dot3dMode h
i=1 train=92.28 ent=0.2540,valid=95.90 ent=0.0627 (25sec) lr=1.0e-02
i=2 train=96.91 ent=0.1034,valid=96.47 ent=0.0433 (26sec) lr=9.5e-03
i=3 train=98.06 ent=0.0653,valid=96.90 ent=0.0381 (26sec) lr=9.0e-03
i=4 train=98.85 ent=0.0407,valid=97.38 ent=0.0319 (25sec) lr=8.6e-03
i=5 train=99.28 ent=0.0271,valid=97.48 ent=0.0249 (26sec) lr=8.1e-03
i=6 train=99.63 ent=0.0174,valid=97.61 ent=0.0233 (26sec) lr=7.7e-03
i=7 train=99.84 ent=0.0111,valid=97.84 ent=0.0212 (26sec) lr=7.4e-03
i=8 train=99.91 ent=0.0081,valid=97.61 ent=0.0204 (25sec) lr=7.0e-03
i=9 train=99.97 ent=0.0058,valid=97.89 ent=0.0178 (26sec) lr=6.6e-03
i=10 train=99.98 ent=0.0046,valid=97.84 ent=0.0177 (26sec) lr=6.3e-03
i=11 train=100.00 ent=0.0037,valid=97.90 ent=0.0168
i=12 train=100.00 ent=0.0031,valid=97.92 ent=0.0163
i=13 train=100.00 ent=0.0028,valid=97.84 ent=0.0159
98%
0 1 2 3 4 5 6 7 8 9
0 1004 0 1 2 1 0 3 2 2 0
1 0 1176 3 0 3 0 3 3 2 0
2 2 4 1029 7 3 1 1 3 2 1
3 3 0 10 1097 0 13 1 2 4 2
4 0 1 2 0 1006 1 3 3 0 11
5 1 0 3 5 0 941 5 2 4 3
6 5 0 0 0 4 8 1044 0 0 0
7 1 2 6 1 3 0 0 1023 2 6
8 5 3 4 8 1 5 3 1 932 4
9 2 0 1 3 10 4 0 3 3 1004
Free
ENTROPY
10
Free
Accurate
MNIST
ACCURACY
100
99
98
97
96
95
94
93
92
91
90
10
Free

Third Attempt: **Fully Connected Neural Network**



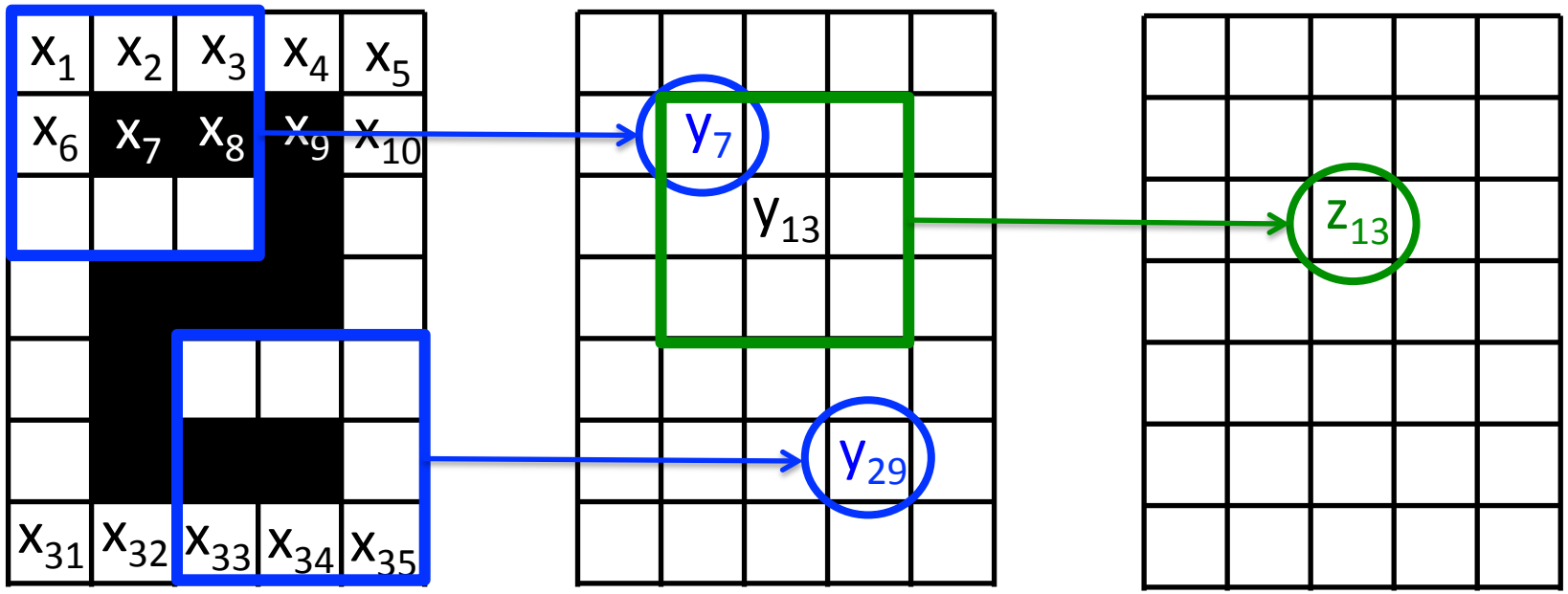


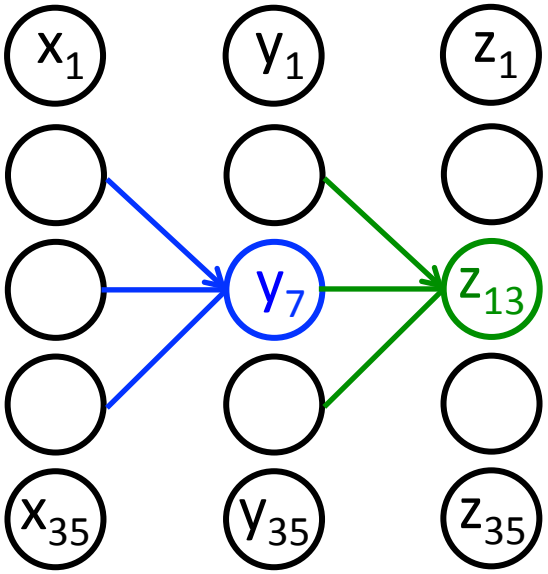
$$y_7 = g\left(p_{j_0} + \sum_{i=1}^{35} p_{j_i} x_i\right)$$

$$y_{29} = g\left(p_{k_0} + \sum_{i=1}^{35} p_{k_i} x_i\right)$$

$$z_{13} = g\left(p_{l_0} + \sum_{i=1}^{35} p_{l_i} y_i\right)$$

Third Attempt: **Convolutional Neural Network**





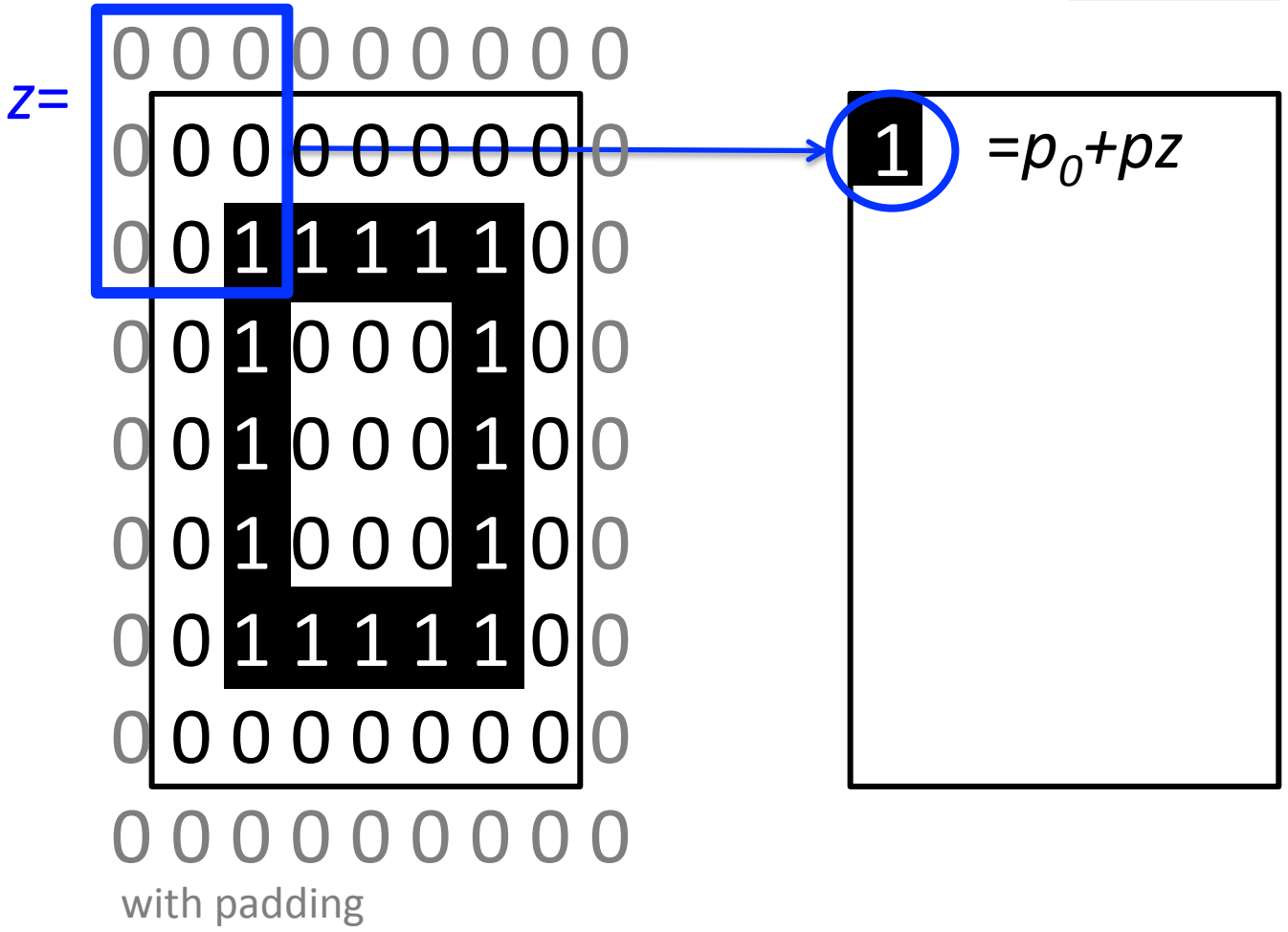
$$y_7 = g\left(p_{r_0} + \sum_{i=1}^{9} p_{r_i} x_{s_i}\right)$$

$$y_{29} = g\left(p_{r_0} + \sum_{i=1}^{9} p_{r_i} x_{t_i}\right)$$

$$z_{13} = g\left(p_{q_0} + \sum_{i=1}^{9} p_{q_i} y_{u_i}\right)$$

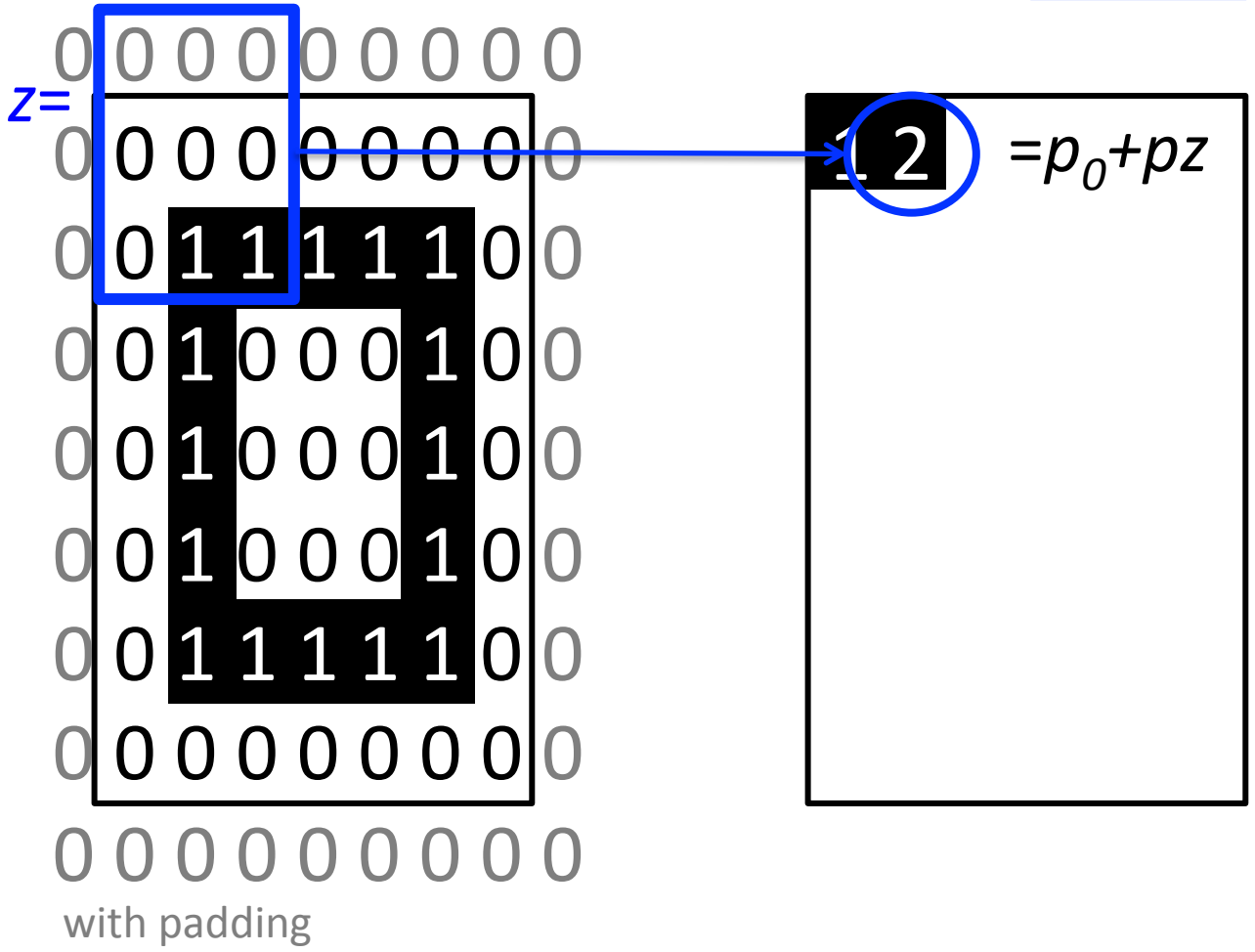
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



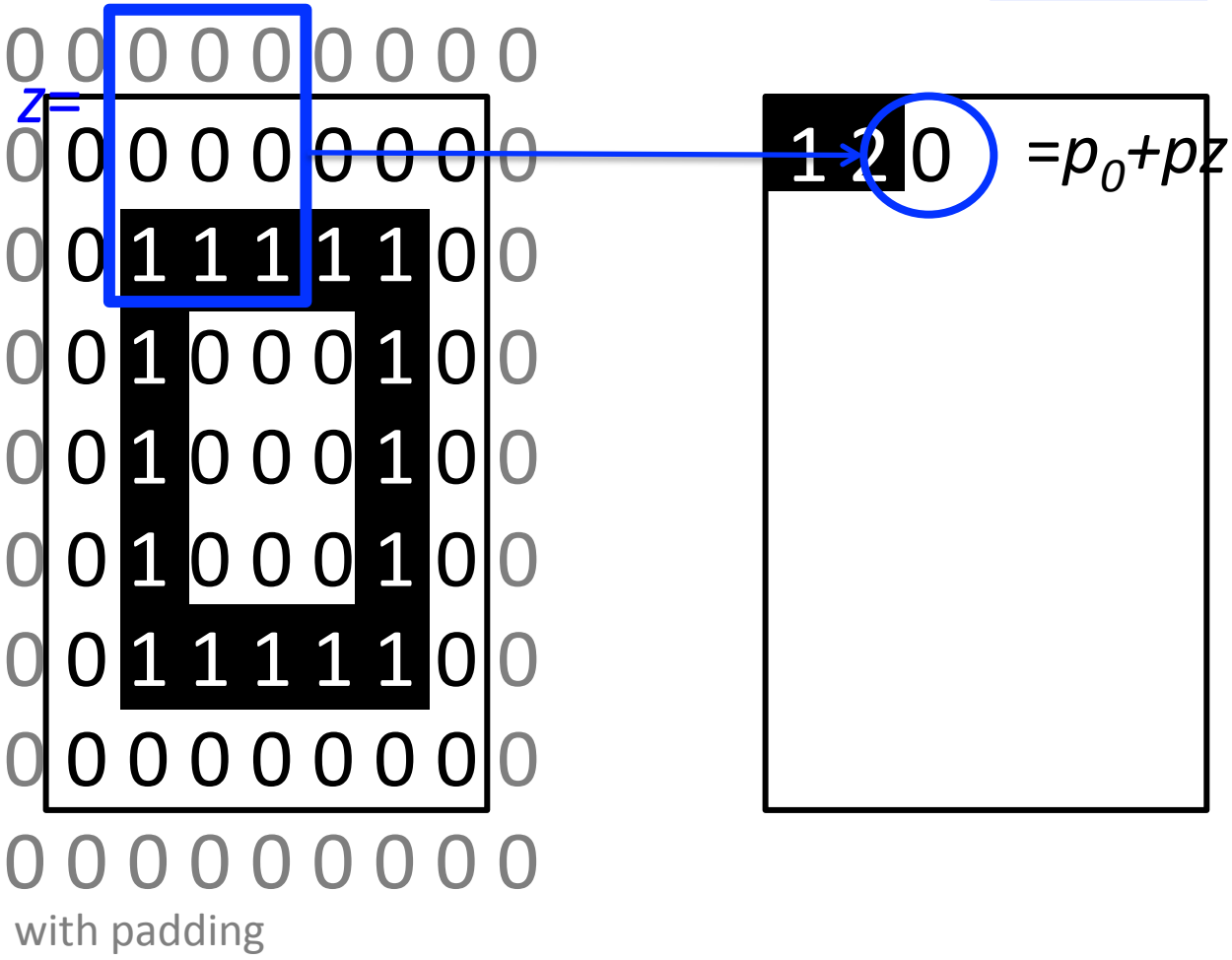
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



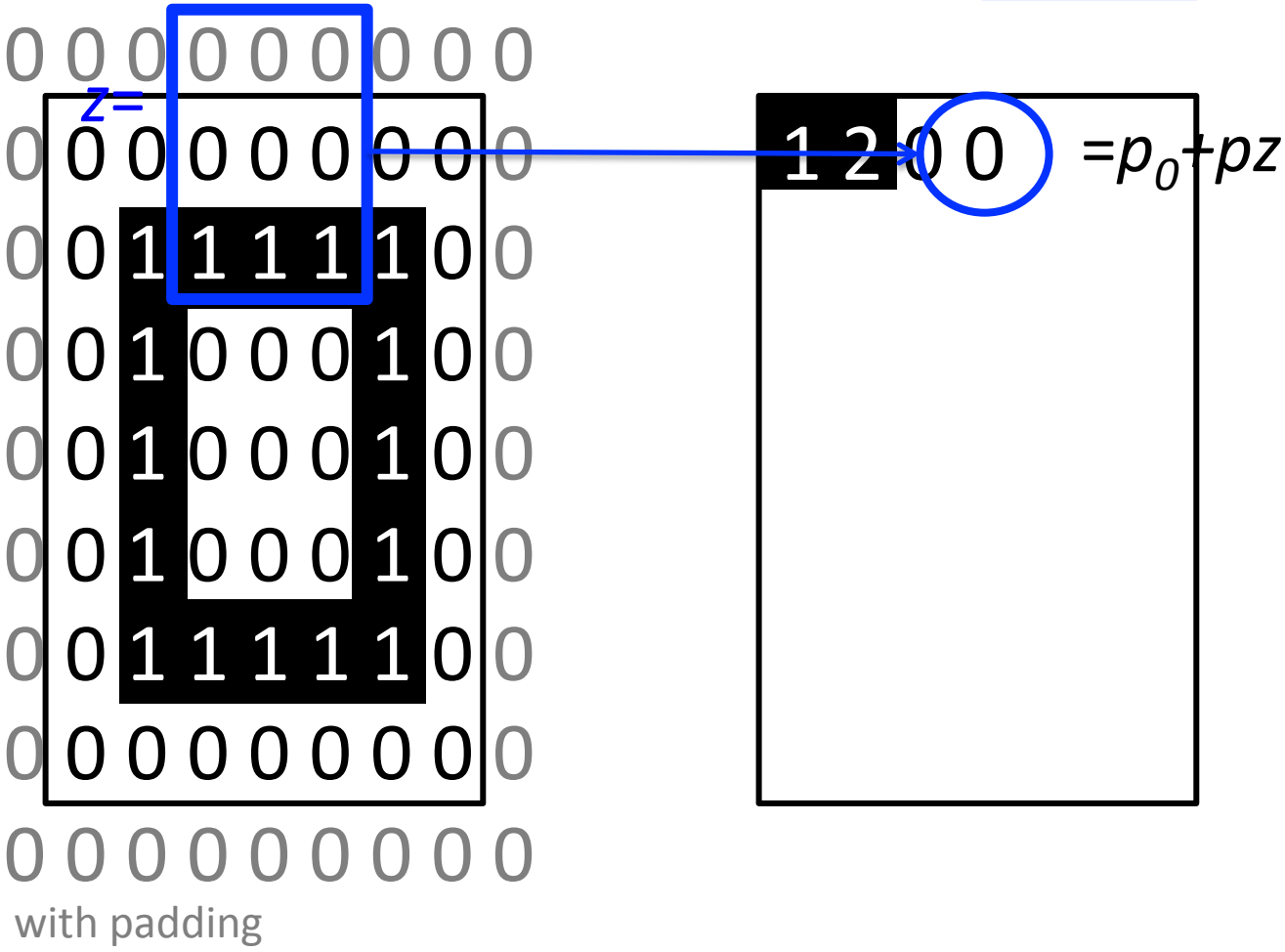
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



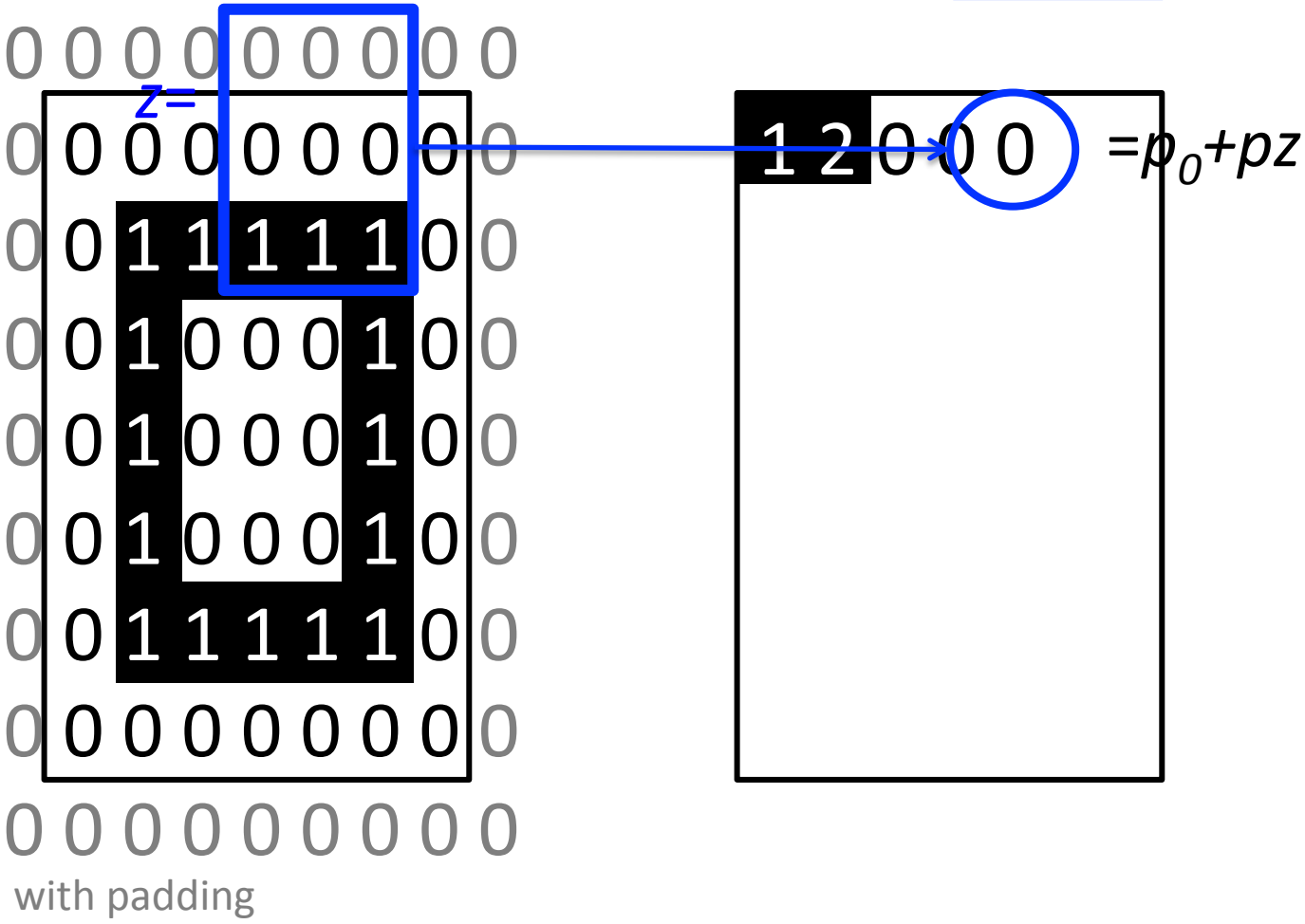
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



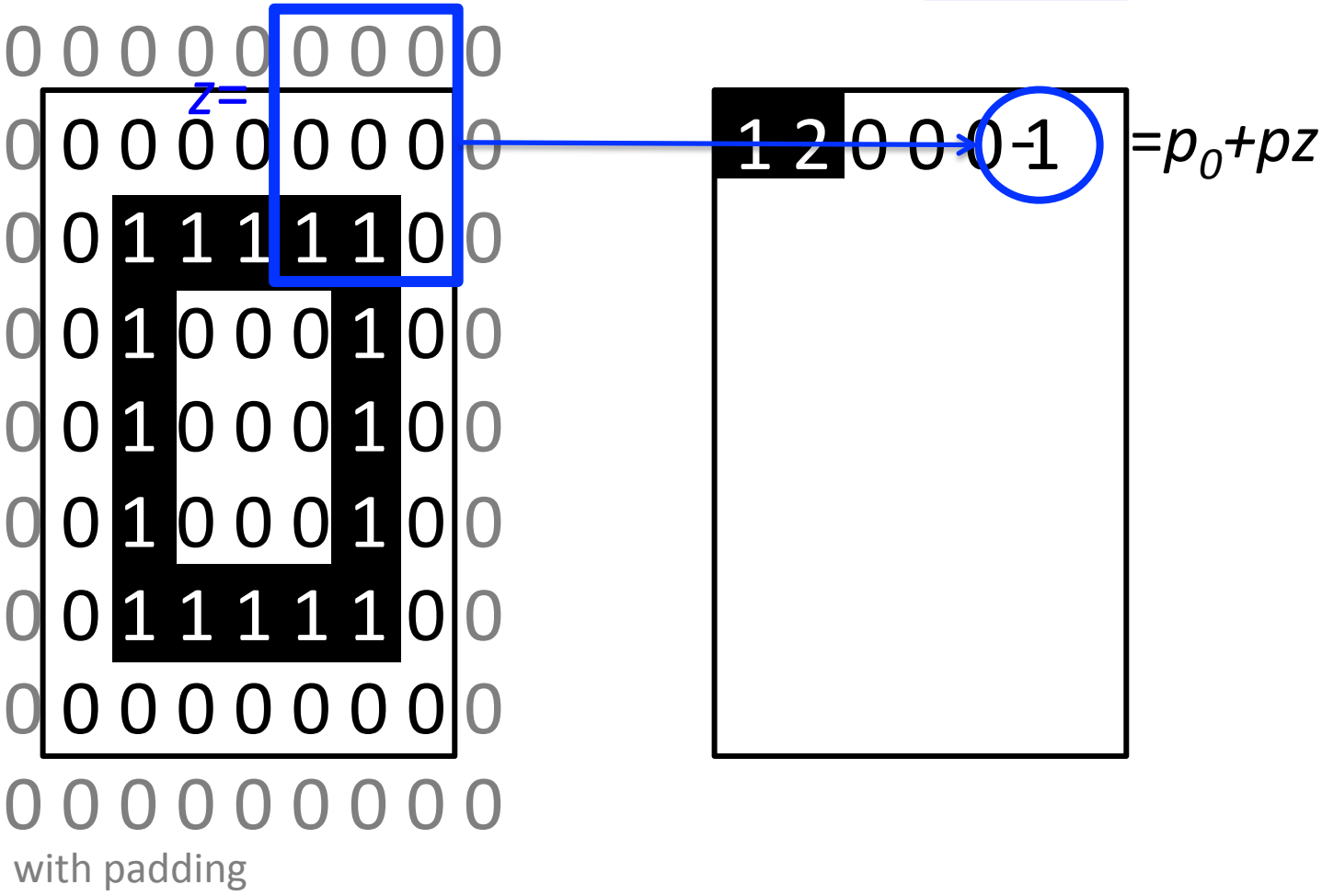
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



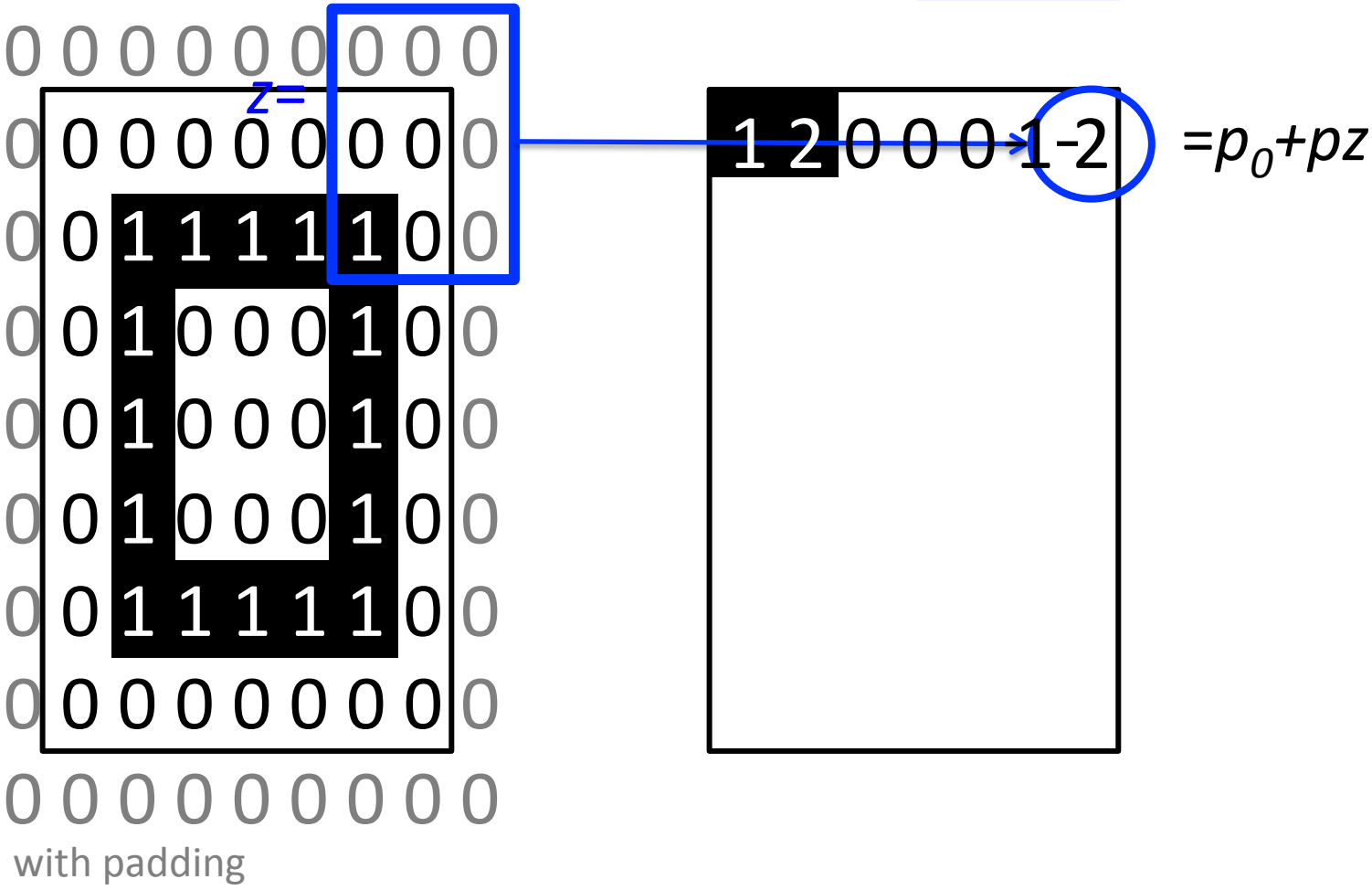
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



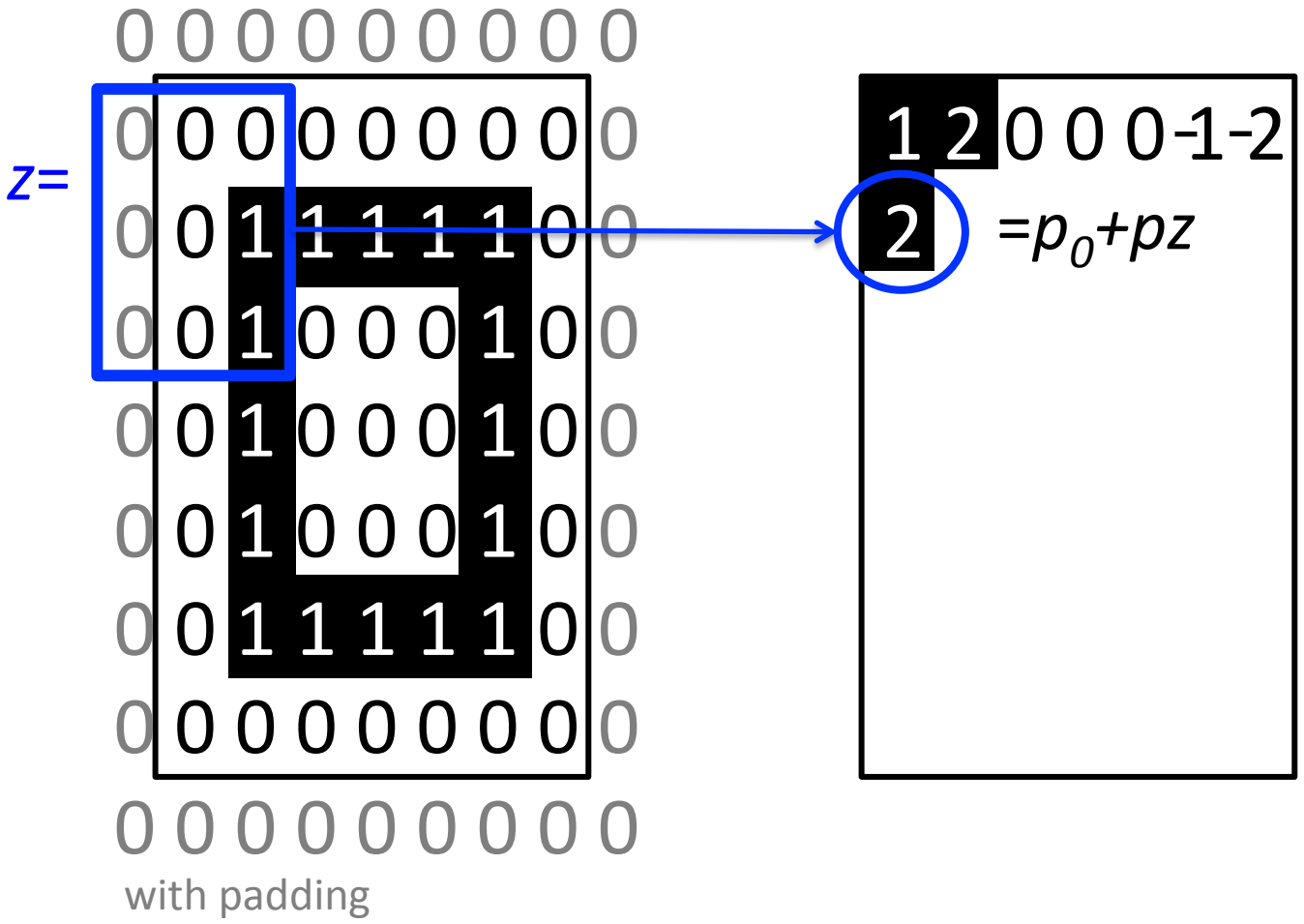
Example: A 3x3 convolution filter (left edge detector).

$p=$ -2 1 1 / -2 1 1 / -2 1 1 & $p_0 = 0$



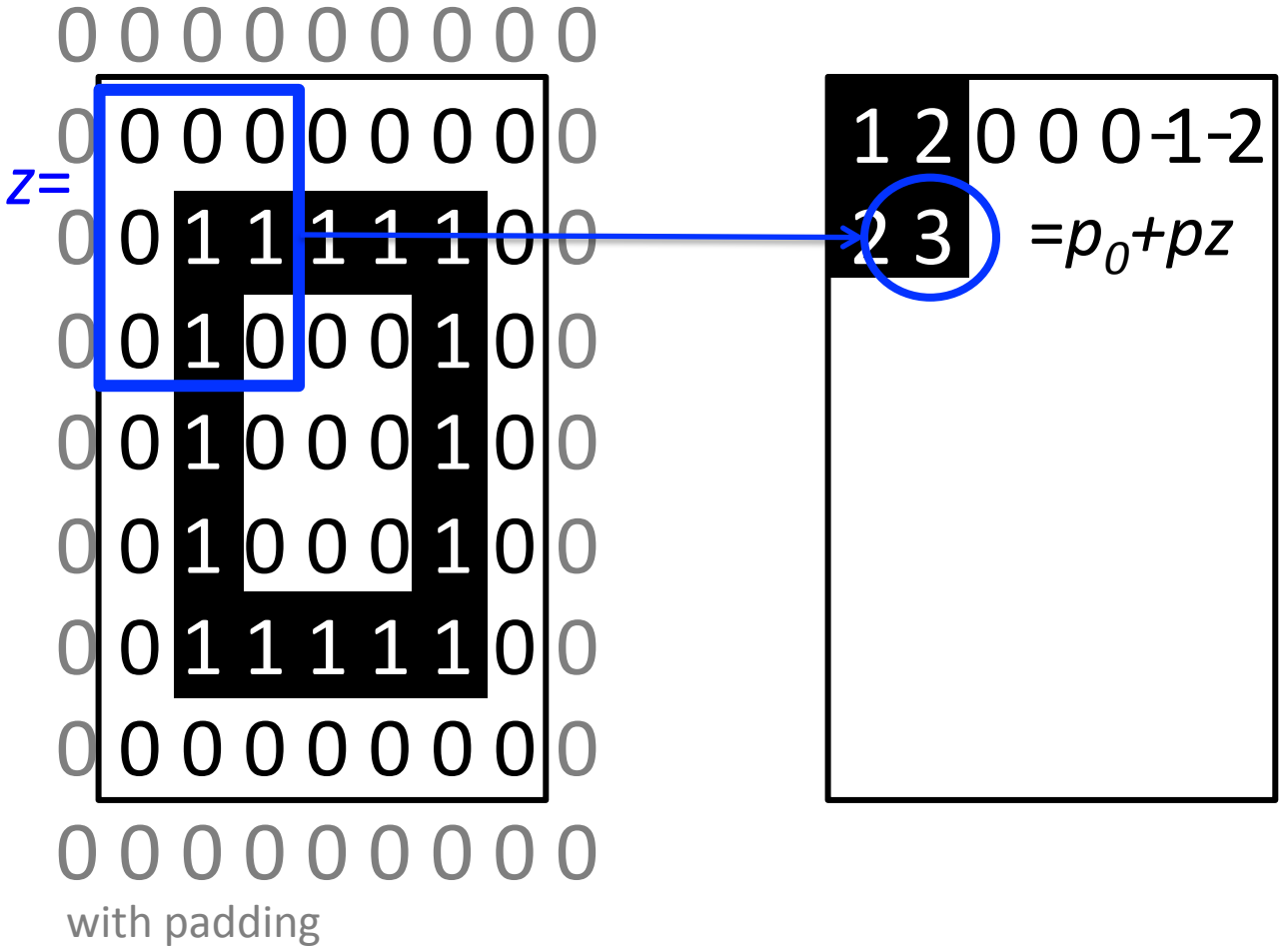
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{matrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{matrix}$ & $p_0 = 0$



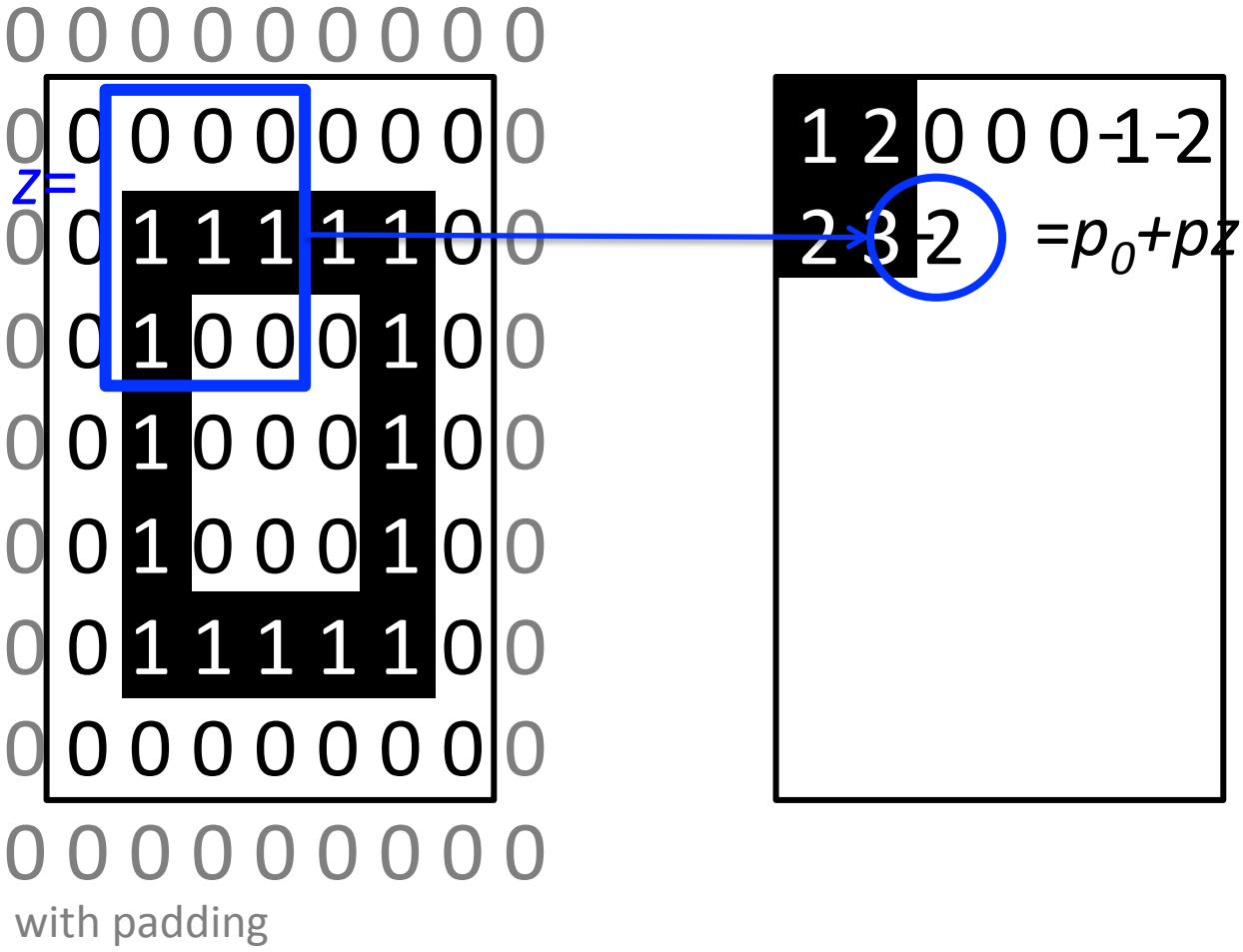
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



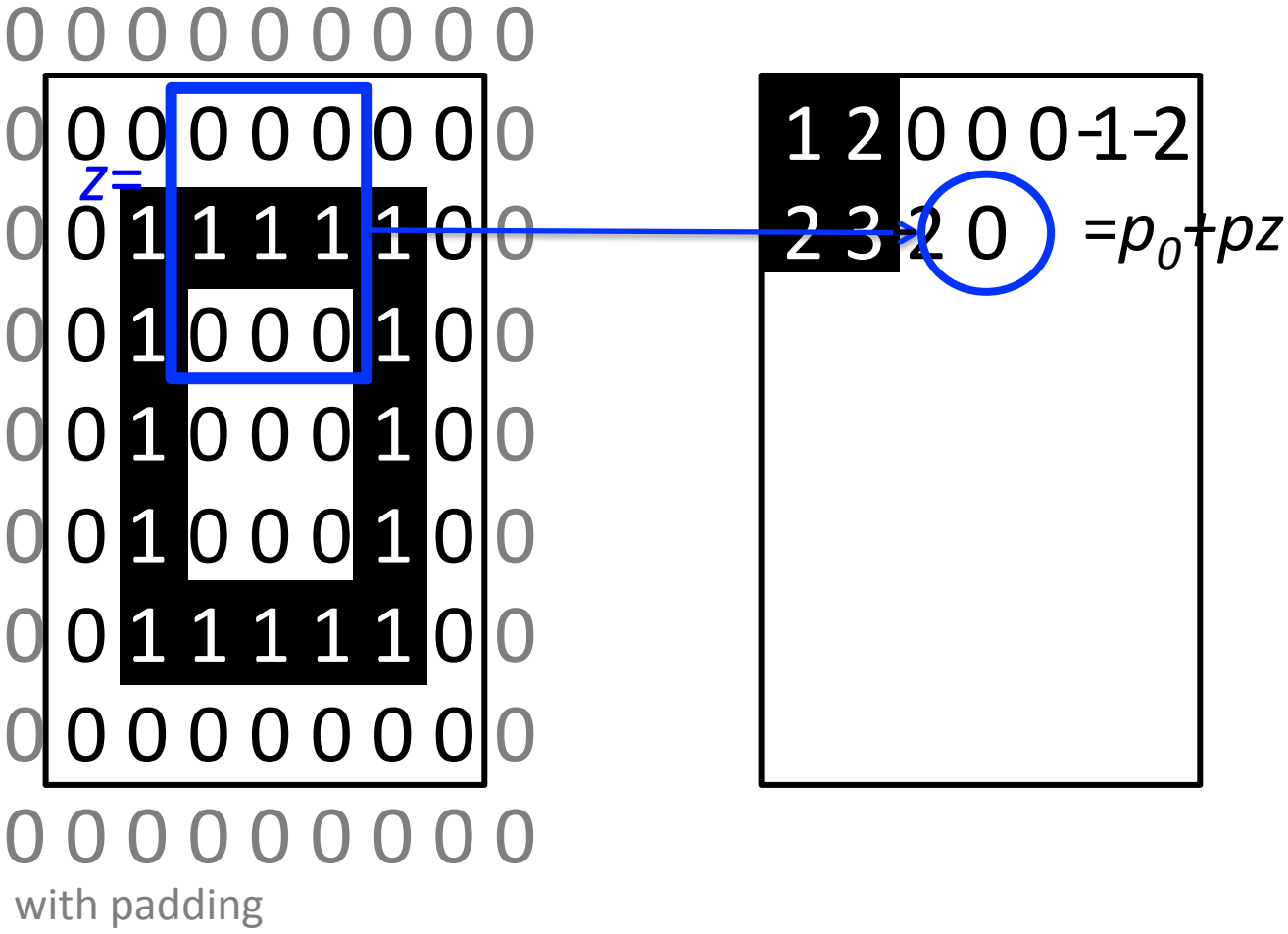
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



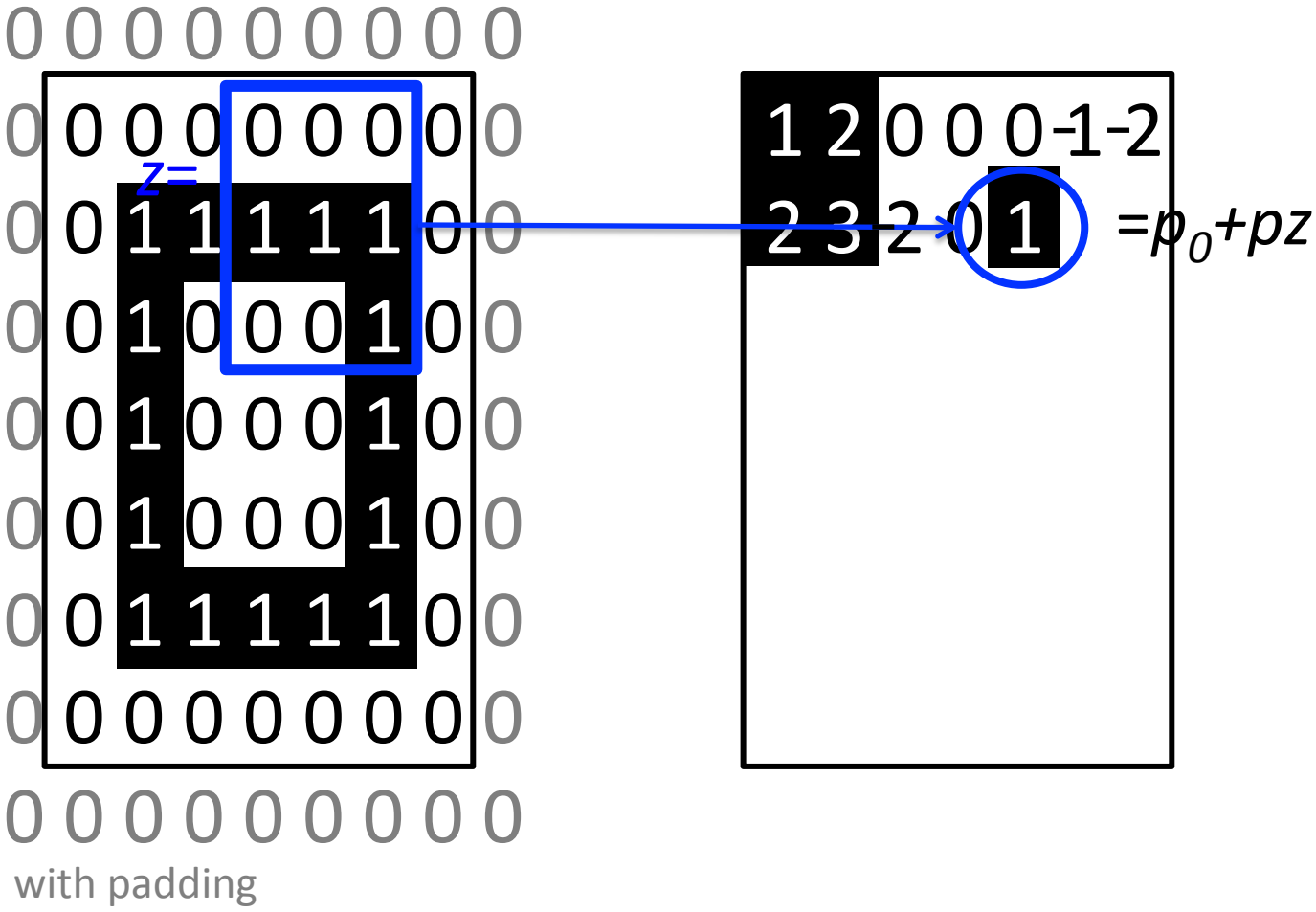
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



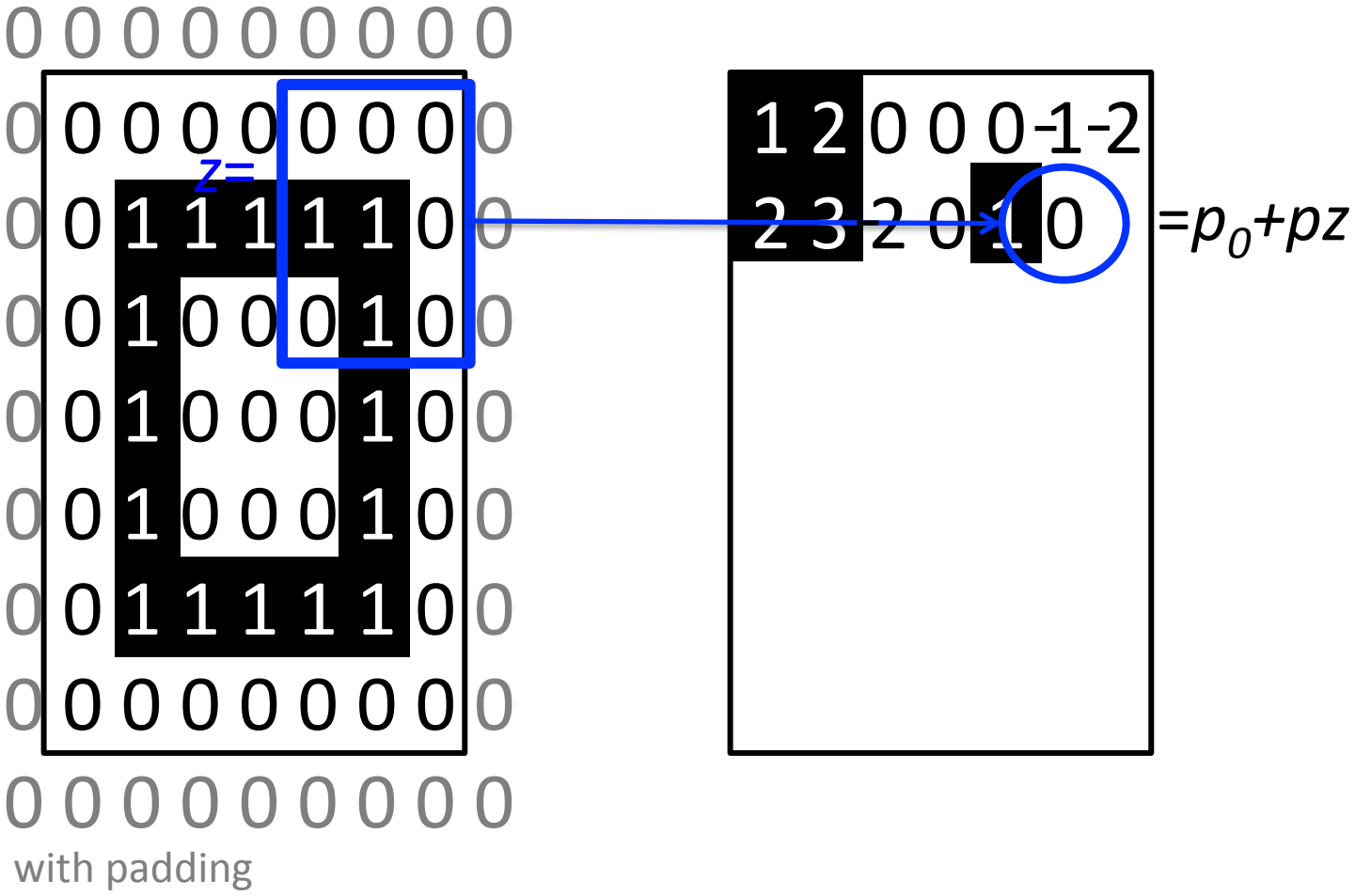
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



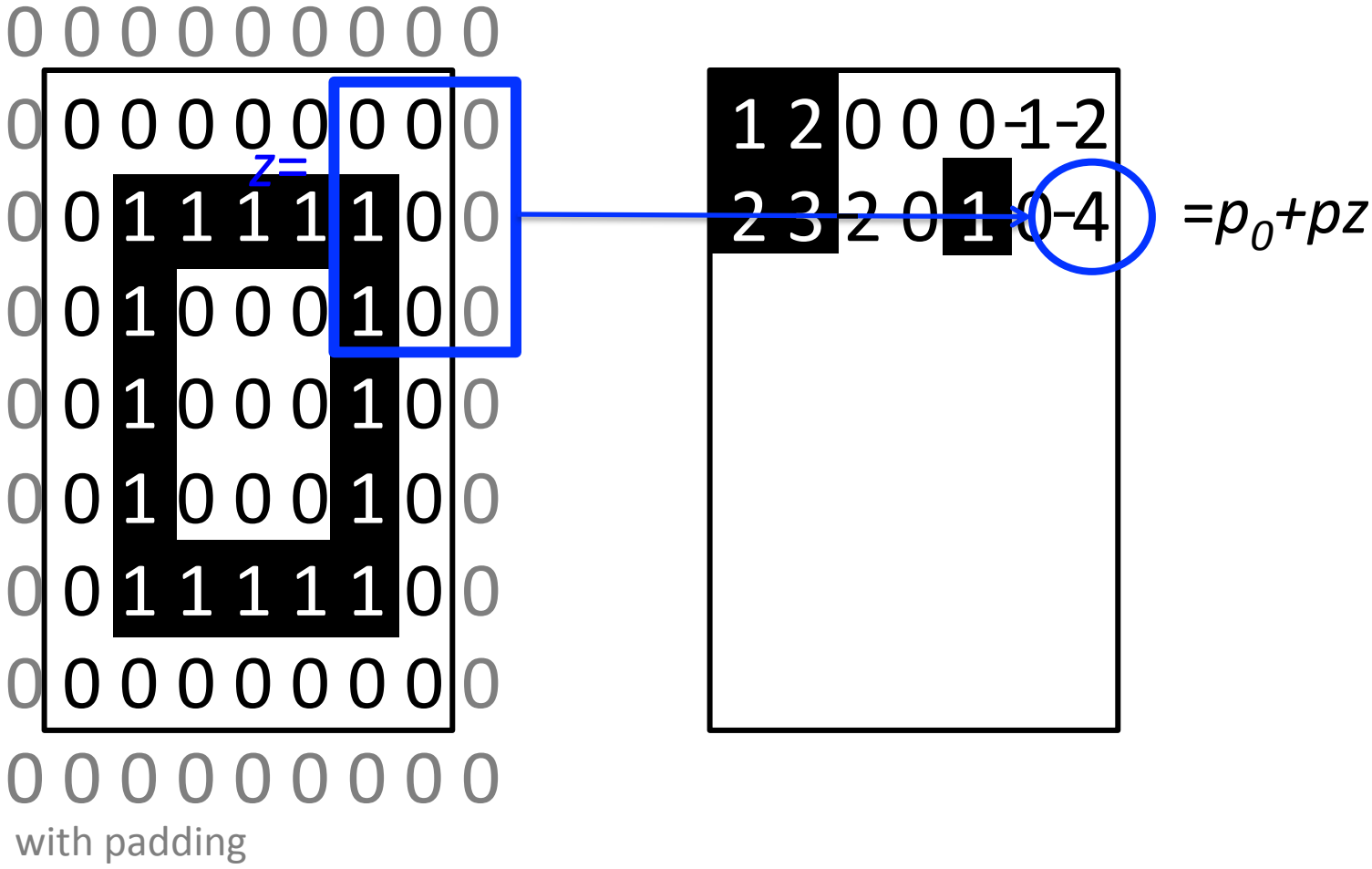
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$



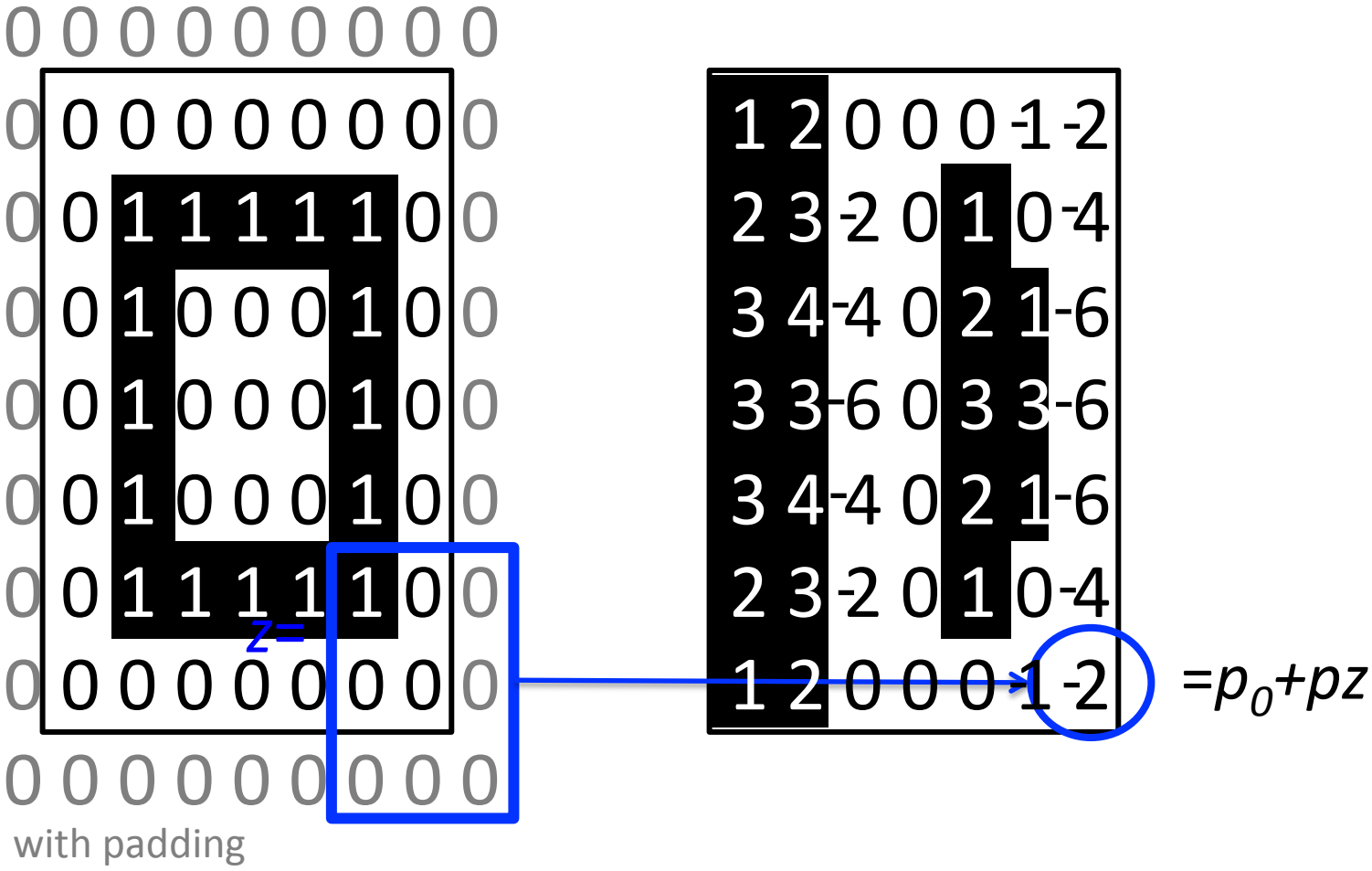
Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$

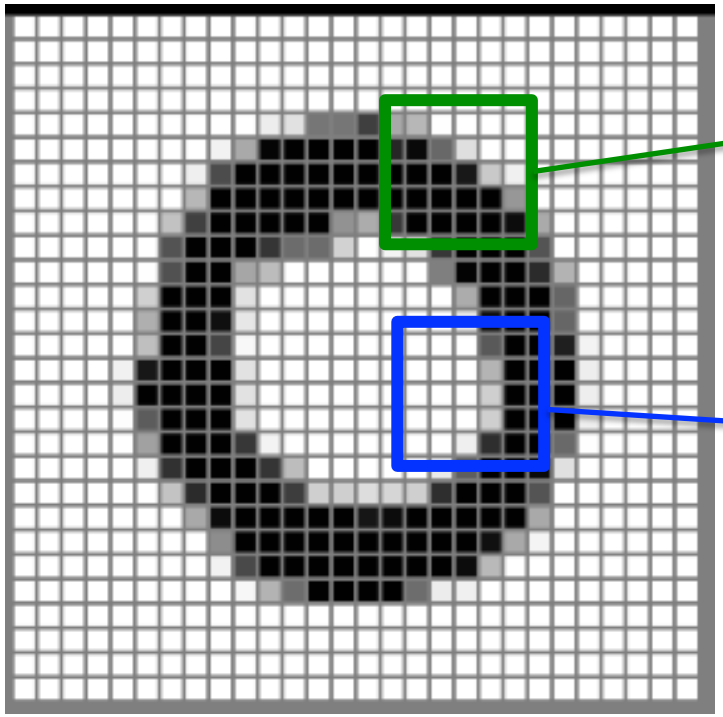


Example: A 3x3 convolution filter (left edge detector).

$p=$ $\begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}$ & $p_0 = 0$

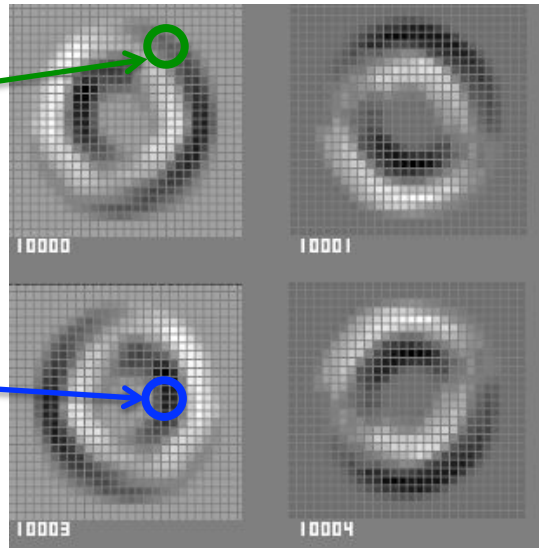


Example: **Four 5x5 convolution filters**

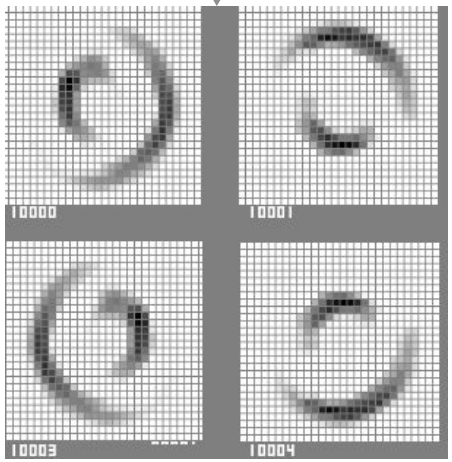


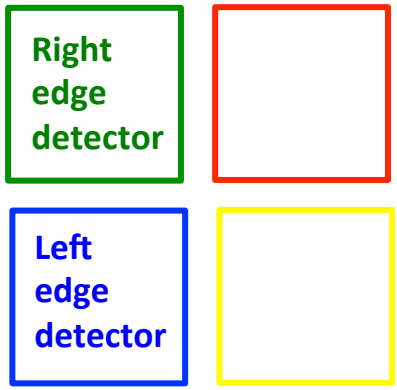
Conv

$p_{k_0} + \bar{p}_k \circledast \bar{x}_i$ for $k = 1, 2, 3, 4$



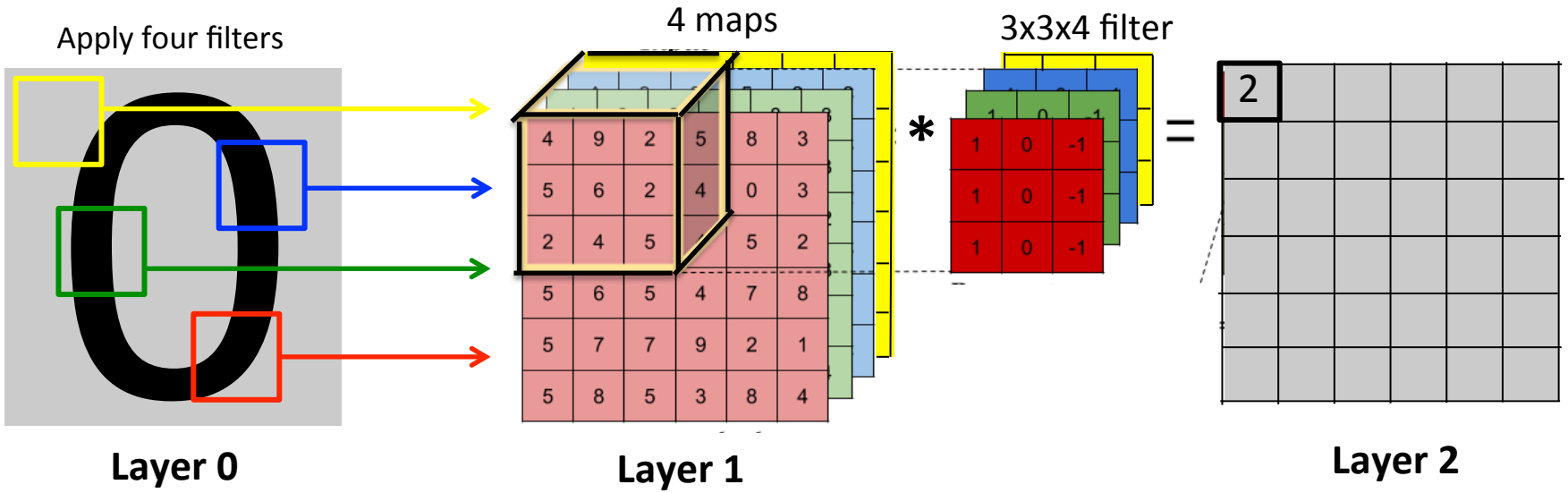
ReLU

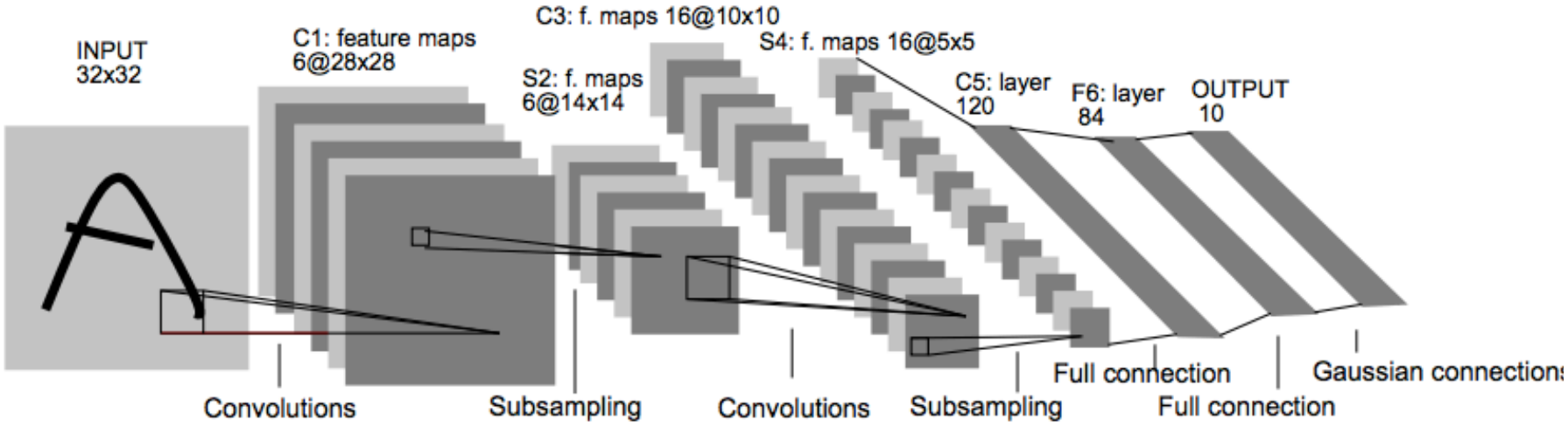




$g(p_{k_0} + \bar{p}_k \circledast \bar{x}_i),\ g(t) = \max\{0, t\}$

Third Attempt: **LeNet5** [LeCun et al., 1998] AT&T Bell Labs



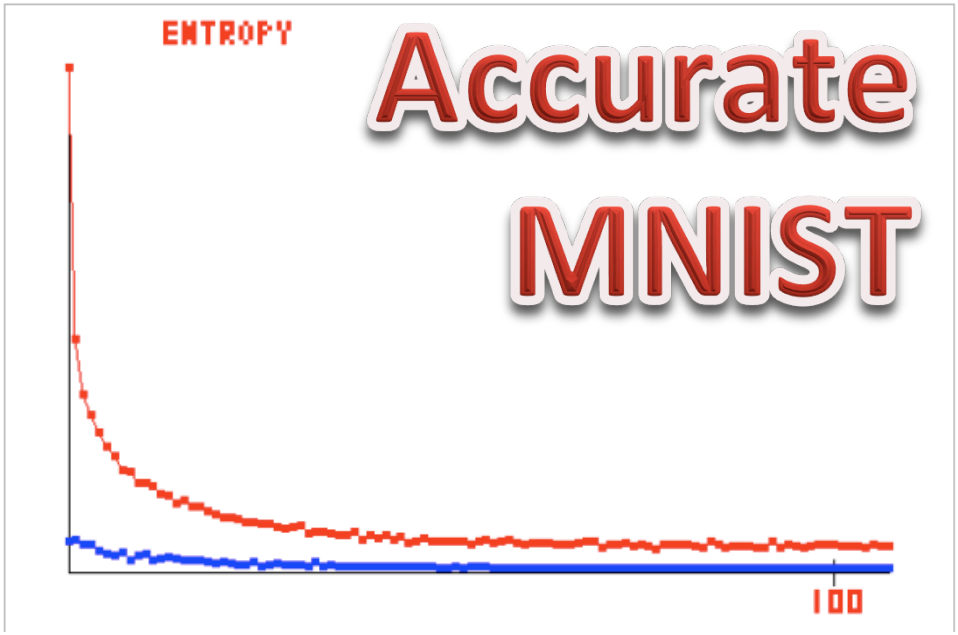


Load l + | Display p + | Init-Net i + | Activation a + | DropOut o +
DataAugment d + | Train-Net b + | Find-kNN g + | Validate-kNN k + | STOP t
Predict w + | QUIT q | DotColor s + | ClearPane3 c | DotRemoveMode r
DotLowResMode u | Dream e + | Heatmap f +

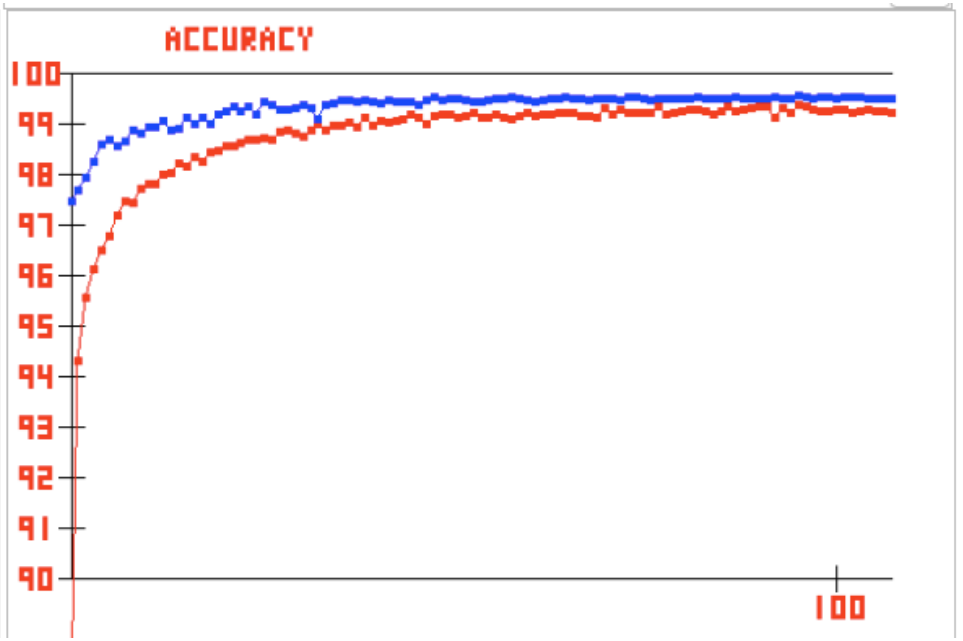
```
i=60 train=99.35 ent=0.0207,valid=99.48 ent=0.0047 (273sec) lr=4.8e-04
i=61 train=99.33 ent=0.0213,valid=99.47 ent=0.0047 (274sec) lr=4.6e-04
i=62 train=99.36 ent=0.0203,valid=99.50 ent=0.0049 (275sec) lr=4.4e-04
i=63 train=99.36 ent=0.0210,valid=99.55 ent=0.0048 (274sec) lr=4.2e-04
i=64 train=99.31 ent=0.0202,valid=99.53 ent=0.0048 (274sec) lr=3.9e-04
i=65 train=99.36 ent=0.0202,valid=99.50 ent=0.0047 (274sec) lr=3.8e-04
i=66 train=99.29 ent=0.0201,valid=99.47 ent=0.0048 (273sec) lr=3.6e-04
i=67 train=99.36 ent=0.0212,valid=99.54 ent=0.0050 (273sec) lr=3.4e-04
i=68 train=99.23 ent=0.0224,valid=99.54 ent=0.0050 (274sec) lr=3.2e-04
i=69 train=99.29 ent=0.0221,valid=99.50 ent=0.0049 (274sec) lr=3.1e-04
i=70 train=99.32 ent=0.0203,valid=99.54 ent=0.0051 (275sec) lr=2.9e-04
i=71 train=99.27 ent=0.0224,valid=99.52 ent=0.0051 (274sec) lr=2.8e-04
i=72 train=99.39 ent=0.0196,valid=99.52 ent=0.00
i=73 train=99.32 ent=0.0205,valid=99.54 ent=0.0
i=74 train=99.33 ent=0.0208,valid=99.54 ent=0.00
i=75 train=99.30 ent=0.0228,valid=99.52 ent=0.0
```

99.5%

	0	1	2	3	4	5	6	7	8	9
0	1015	0	0	0	0	0	2	0	0	0
1	0	1208	0	0	2	0	1	2	1	0
2	0	0	1042	1	0	0	0	1	0	0
3	0	0	0	1074	0	0	0	0	1	1
4	0	0	0	0	982	0	0	0	0	6
5	0	0	0	0	0	988	2	0	1	2
6	0	0	1	0	0	2	1057	0	0	0
7	0	1	2	0	1	0	0	1121	0	1
8	0	0	1	0	1	0	3	1	1000	1
9	0	0	0	0	8	0	0	2	2	1050



Accurate MNIST



2007: Nvidia **GPU** releases **CUDA** [Nvidia, 2007]

2009: Large Scale Deep Unsupervised Learning using **Graphic Processors** [Raina et al., 2009] Stanford University

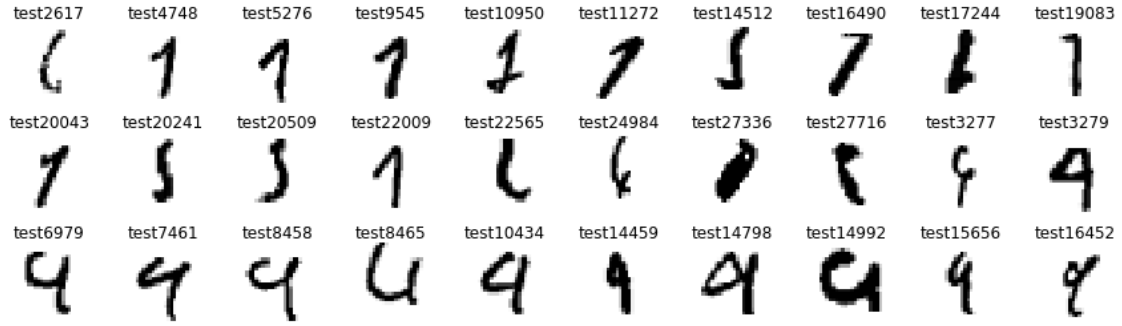
CPU is SISD
(single instruction single data)

GPU is SIMD
(single instruction multiple data)

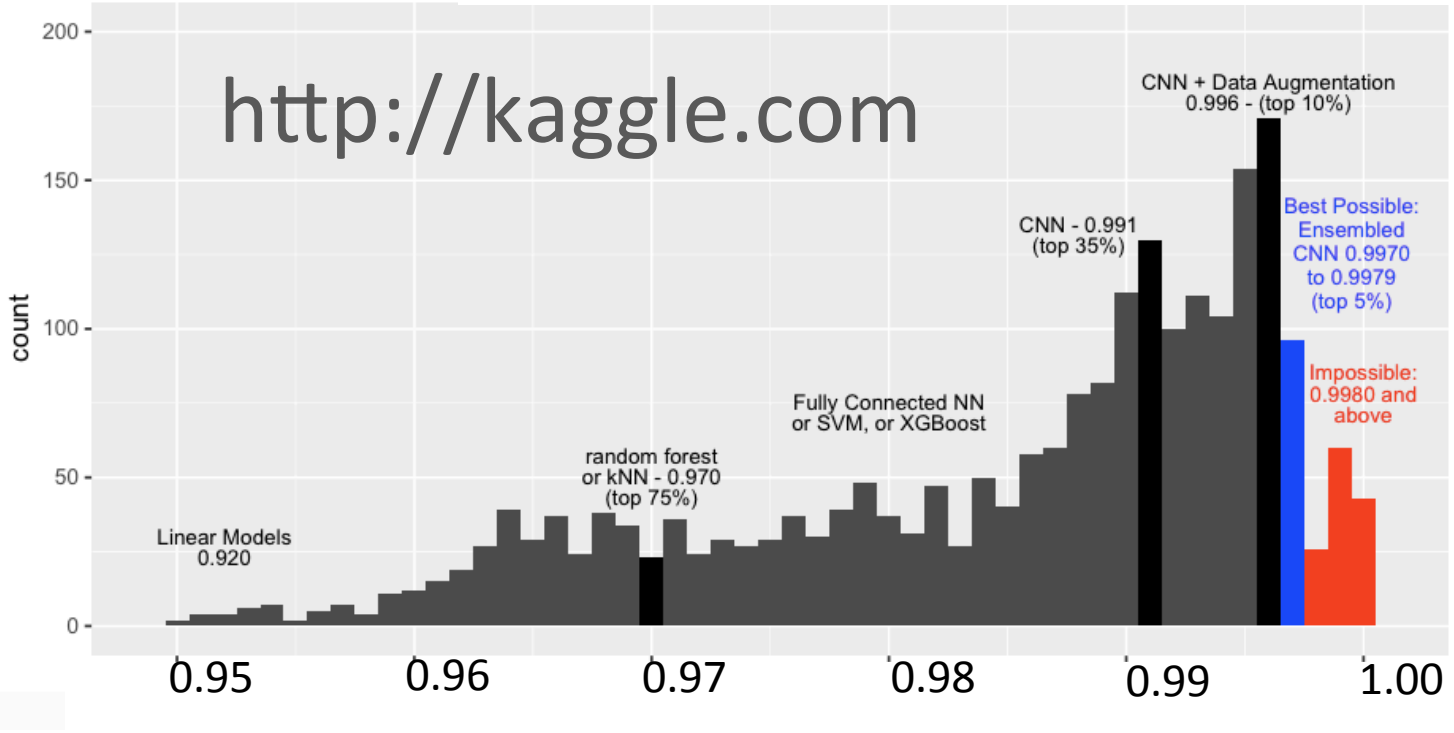
GPU provides 10x to 100x speedup over CPU!

2011: MNIST Solved [Ciresan et al., 2011] Switzerland

99.73% Accuracy Achieved



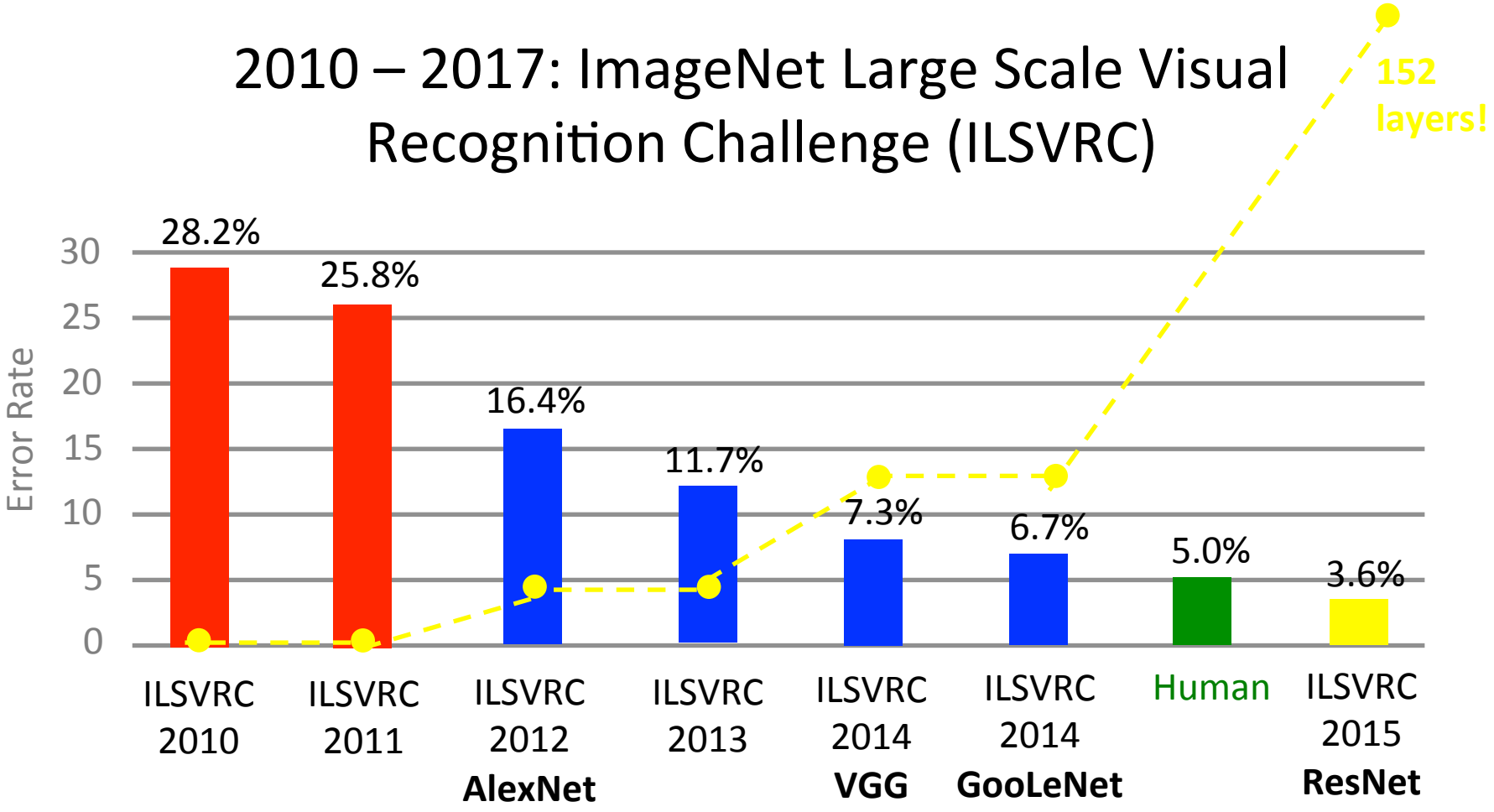
Exceeding 99.8% isn't possible because of mislabeled and illegible digits.



CNN plus GPU plus Big Data!


2010 - 2017: ImageNet Annual Comp
The “Olympics” of Computer Vision
1,000 image categories
1,431,167 images
256x256 pixels

2010 – 2017: ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



CNN plus GPU!

In 2012 only 1 CNN:

1st U. Toronto – 16%

2nd U. Tokyo – 26%

3rd U. Oxford – 27%

4th INRIA – 27%

5th U. Amsterdam – 29%

In 2013

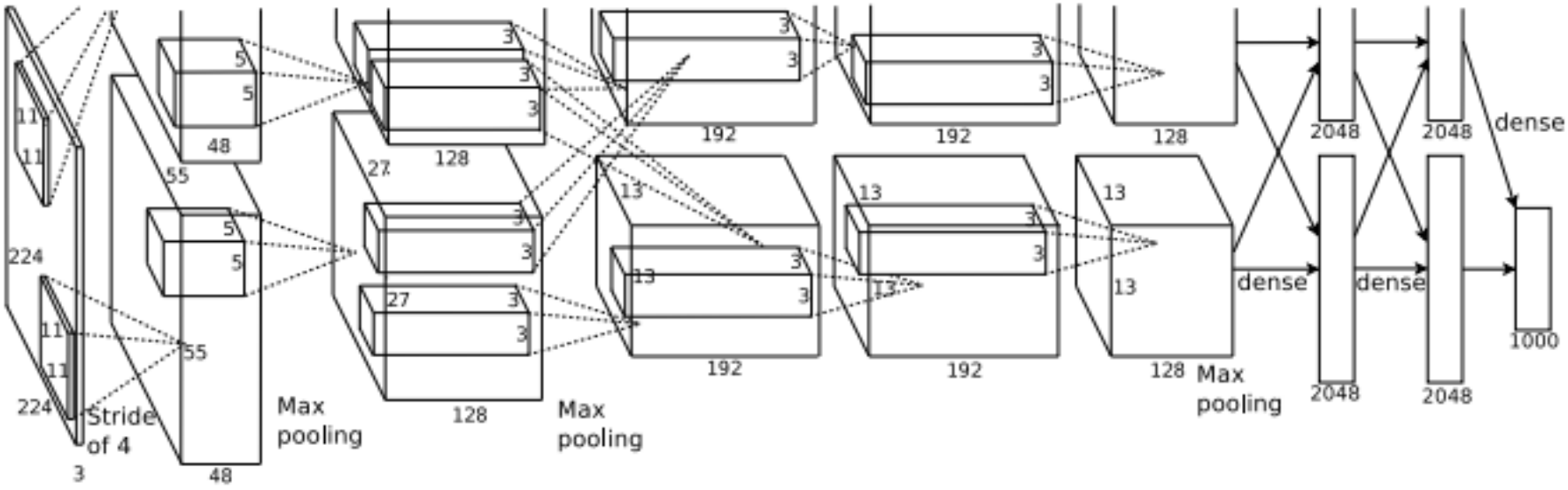
All top 20 used CNN and GPU!

In 2015

Revolution of CNN Depth!!

ResNet won every event!

2012: **AlexNet** [Krizhevsky et al., 2012] Univeristy of Toronto



Winner 2012 ImageNet by a large margin (16% error vs. 26%)

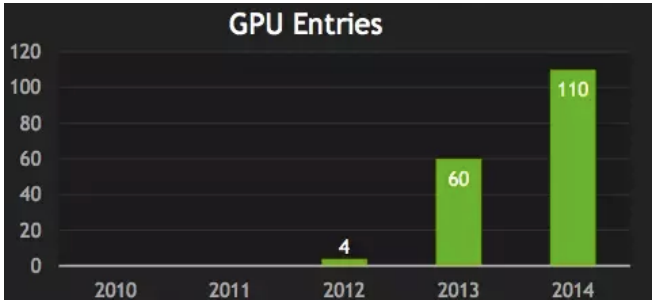
First CNN in competition

First GPU in competition, used CUDA

Trained 6 days on two Nvidia GeForce GTX 580s

First use of **ReLU** activation

Deep 8 layers with 62.3 million parameters!

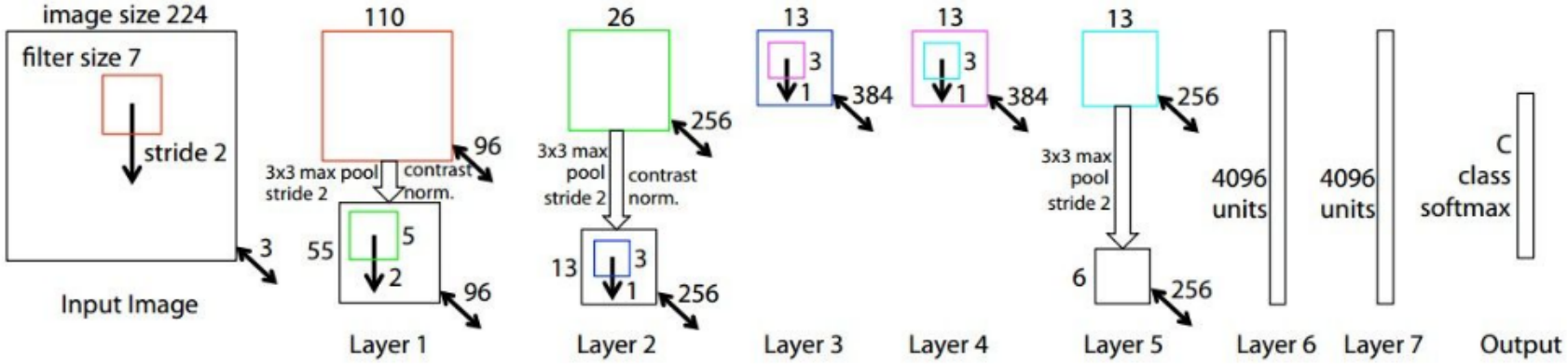


Trained with 15 million images from 22,000+ categories

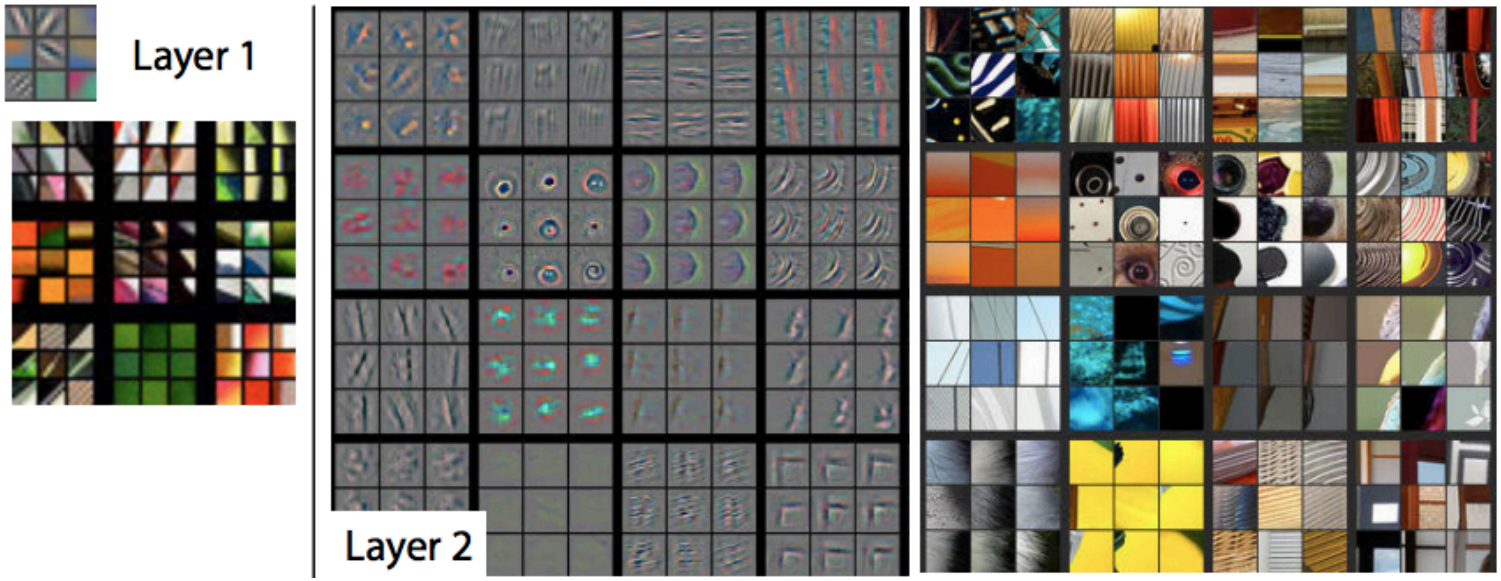
Performs 1.1 billion computations in one forward pass for one image.

Used data augmentation, dropout, SGD with momentum, and ensembling.

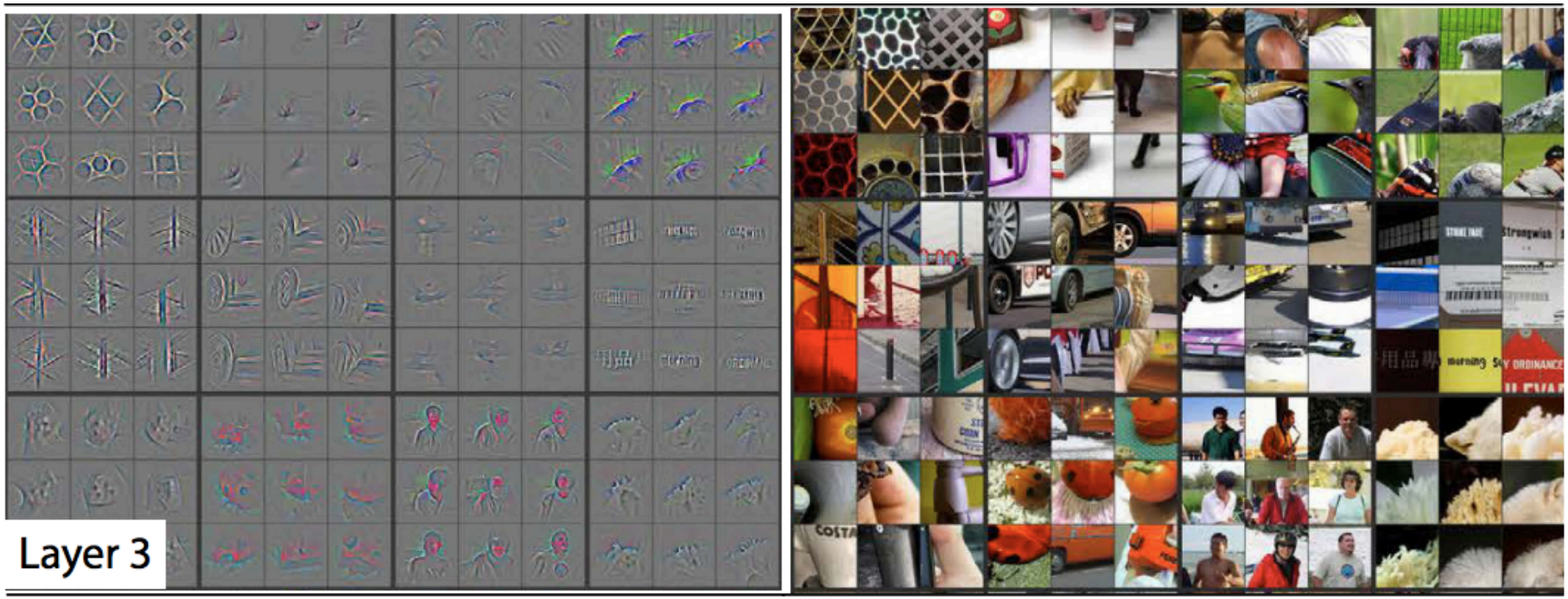
2013: ZFNet [Zeiler & Fergus, 2013] University of Courant, NY



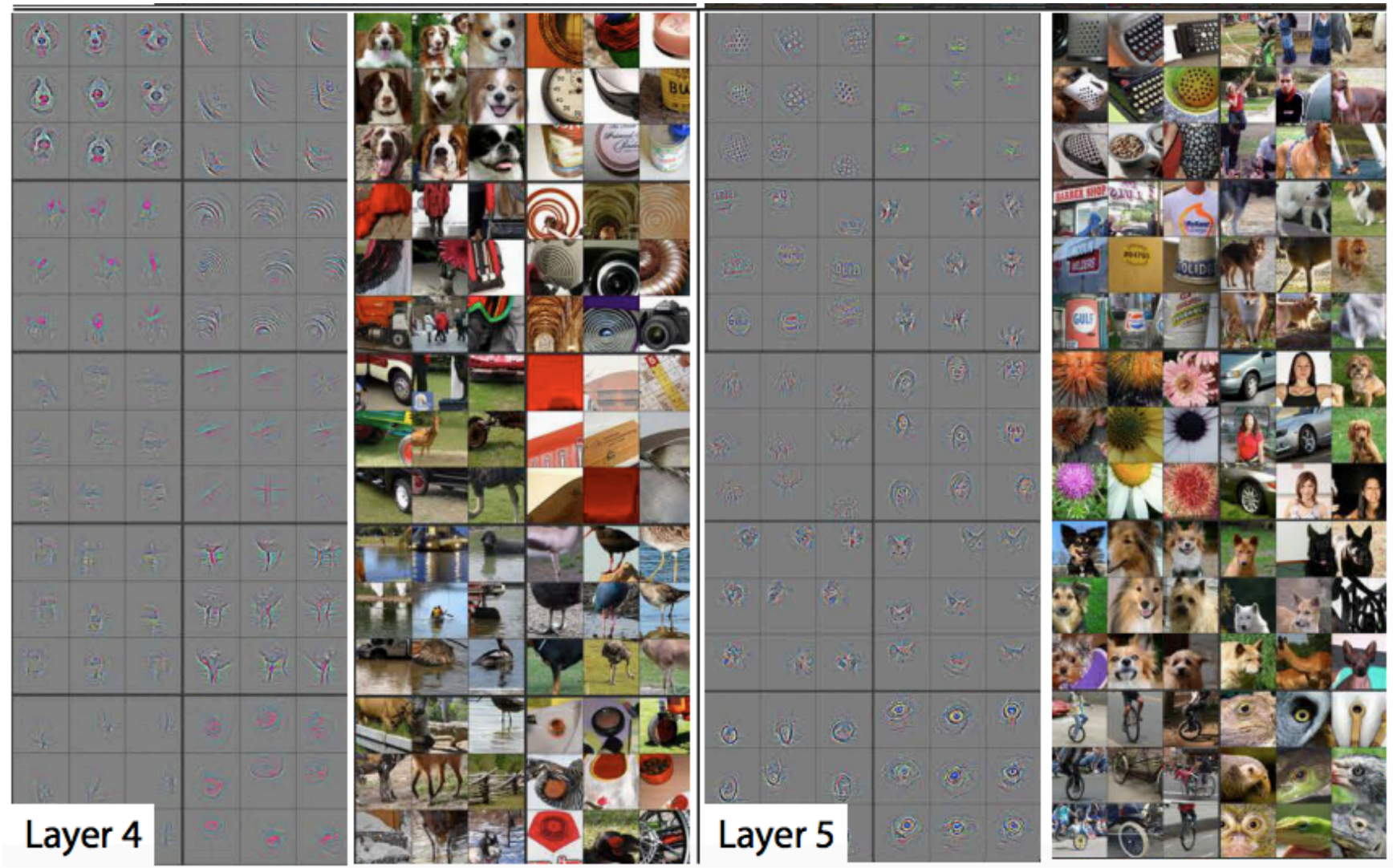
Winner 2013 ImageNet by analyzing **how CNNs work**.



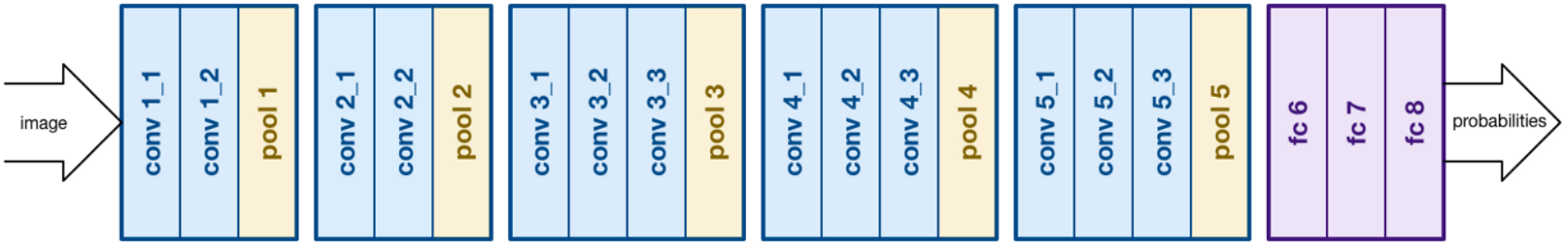
2013: **ZFNet** [Zeiler & Fergus, 2013] University of Courant, NY



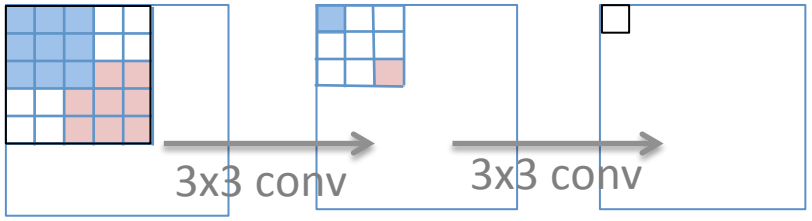
2013: **ZFNet** [Zeiler & Fergus, 2013] University of Courant, NY



2014: VGG Net [Simonyan & Zisserman, 2014] University of Oxford



Two 3x3 convolutions layers



Second place 2014 ImageNet

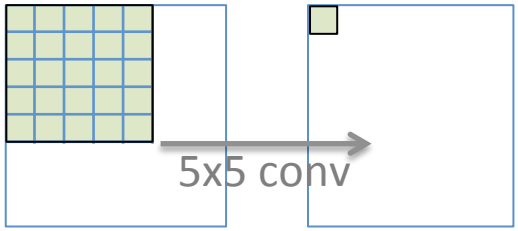
Twice as deep as previous CNNs

Simple architecture of only 3x3 convolutions

Trained on four Nvidia Titan Black GPUs for 2-3 weeks

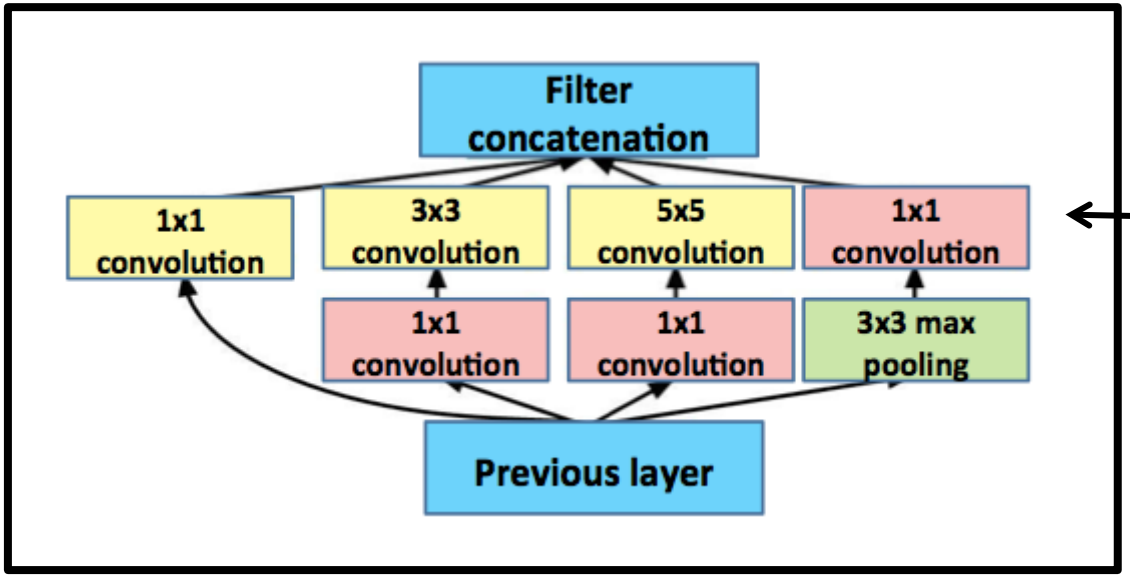
Deeper 16 (or 19) layers with 139 million parameters!

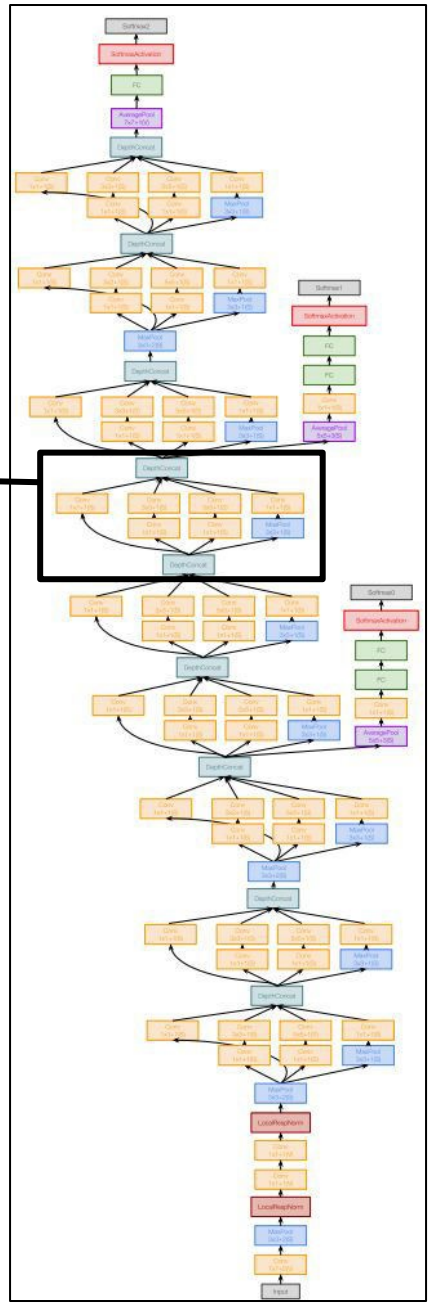
Versus one 5x5 convolution layer



2014: **GooLeNet** [Szegedy et al., 2014]

Inception Module





Winner 2014 ImageNet with 6.7% error

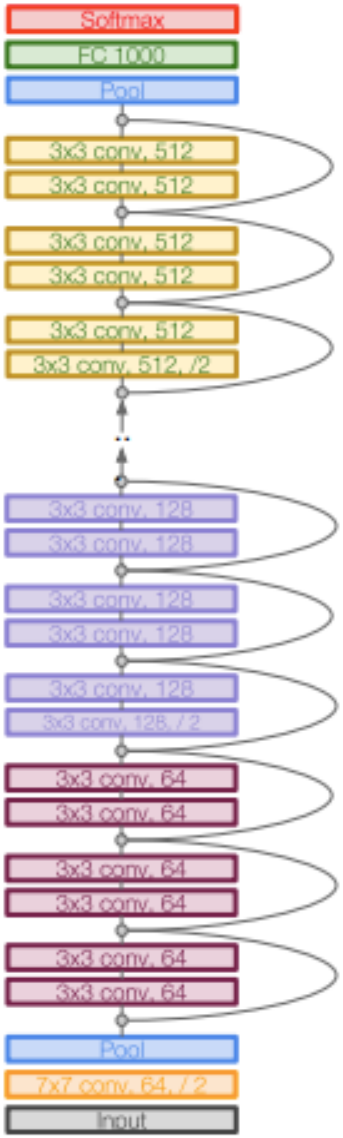
Parallel and computationally efficient architecture

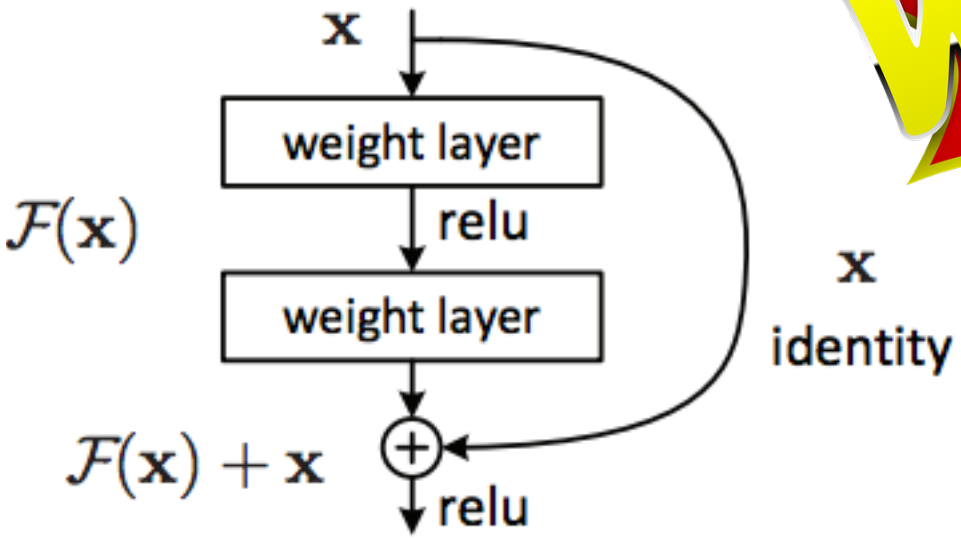
Used batch normalization and RMSprop optimizer

Deeper 21 layers with only 4 million parameters!

Trained on a few high-end GPUs under 1 week.

2015: ResNet [He et al., 2015] Microsoft Research





$\mathcal{F}(\mathbf{x})$

$\mathcal{F}(\mathbf{x}) + \mathbf{x}$

Winner 2015 ImageNet in all categories!
Achieved 3.5% error which beat human 5%.

Extremely Deep 152 layers!

Trained on eight GPUs for 2-3 weeks

Used skip connections


2016 – 2018: Explosion of Deep Learning!!

World's best Go player flummoxed by Google's 'godlike' AlphaGo AI

Ke Jie, who once boasted he would never be beaten by a computer at the ancient Chinese game, said he had 'horrible experience'

Superhuman AI for heads-up no-limit poker: Libratus beats top professionals

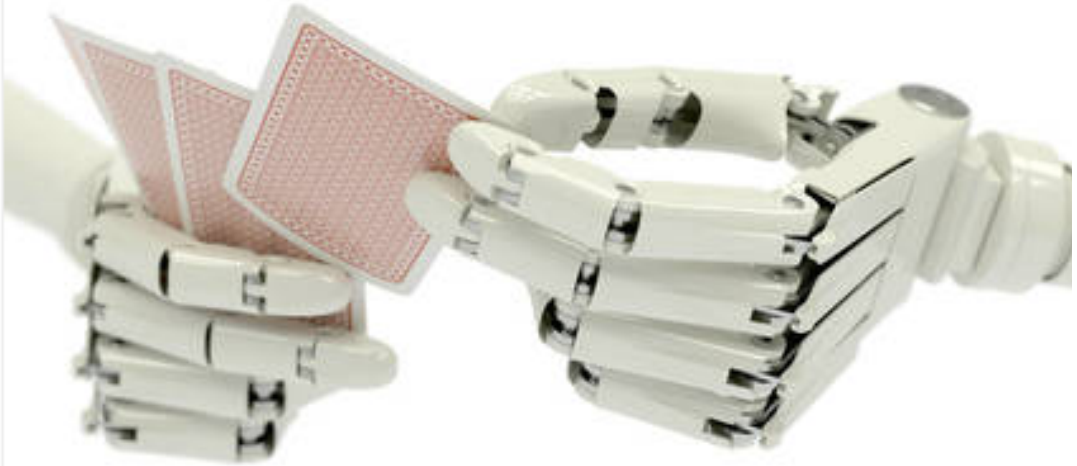
Science

Andrew Ng ✓
@AndrewYNg

CMU just made history: AI beats top humans at Texas Hold'em poker. A stunning accomplishment, comparable to Deep Blue & AlphaGo!

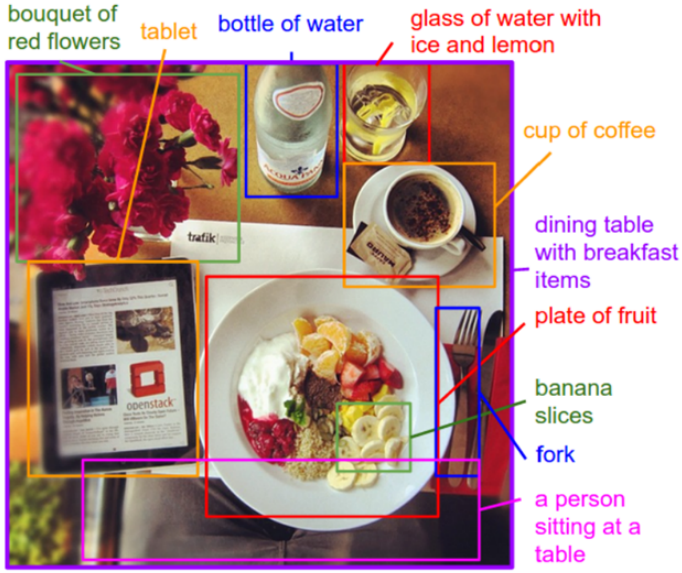
10:42 PM - Jan 30, 2017 · Mountain View, CA

♡ 1,857 💬 1,507 people are talking about this



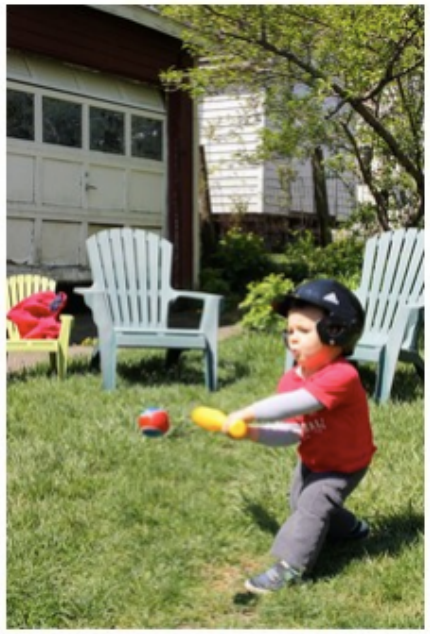
GPUs + Deep Learning + Natural Language Processing

Robots add captions!



Example output of the model

a cow is standing in the middle of a street



a young boy is holding a baseball bat

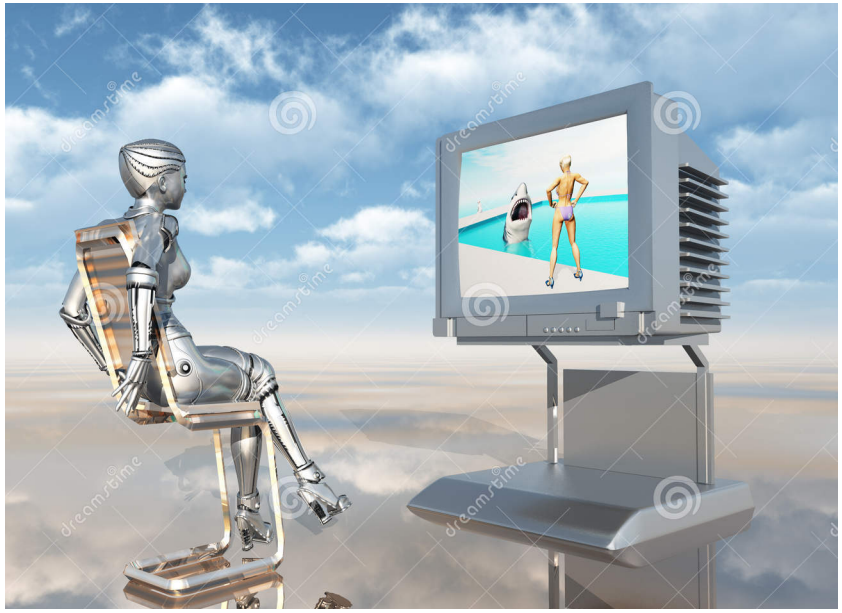
a cat is sitting on a toilet seat

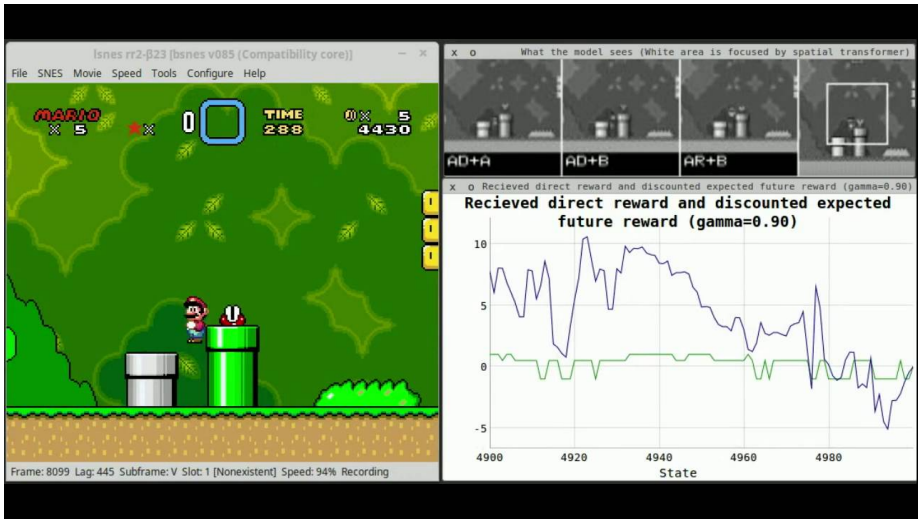
a group of people sitting at a table with wine glasses

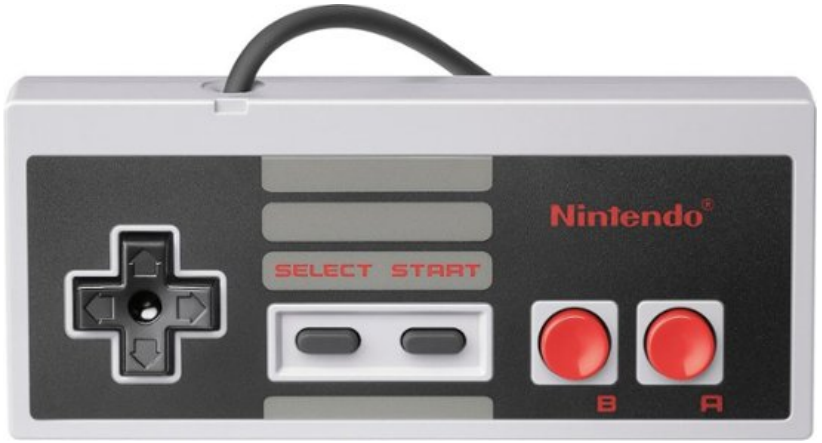
a display case filled with lots of different types of donuts

GPUs + Deep Learning + Reinforcement Learning

Robots win
our video games!

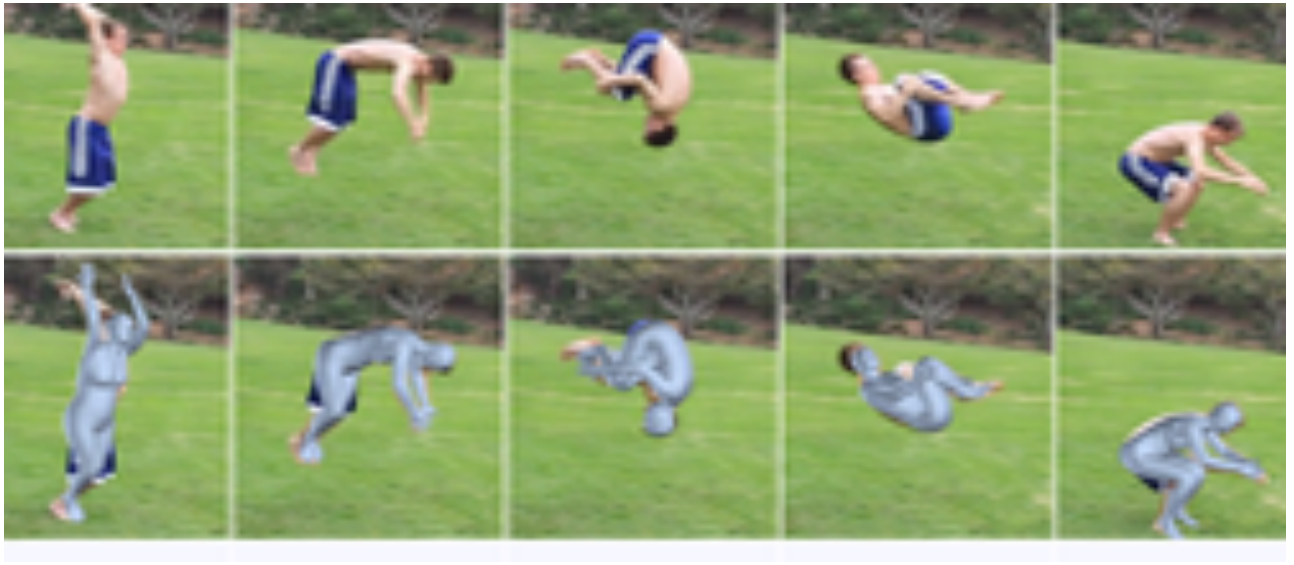






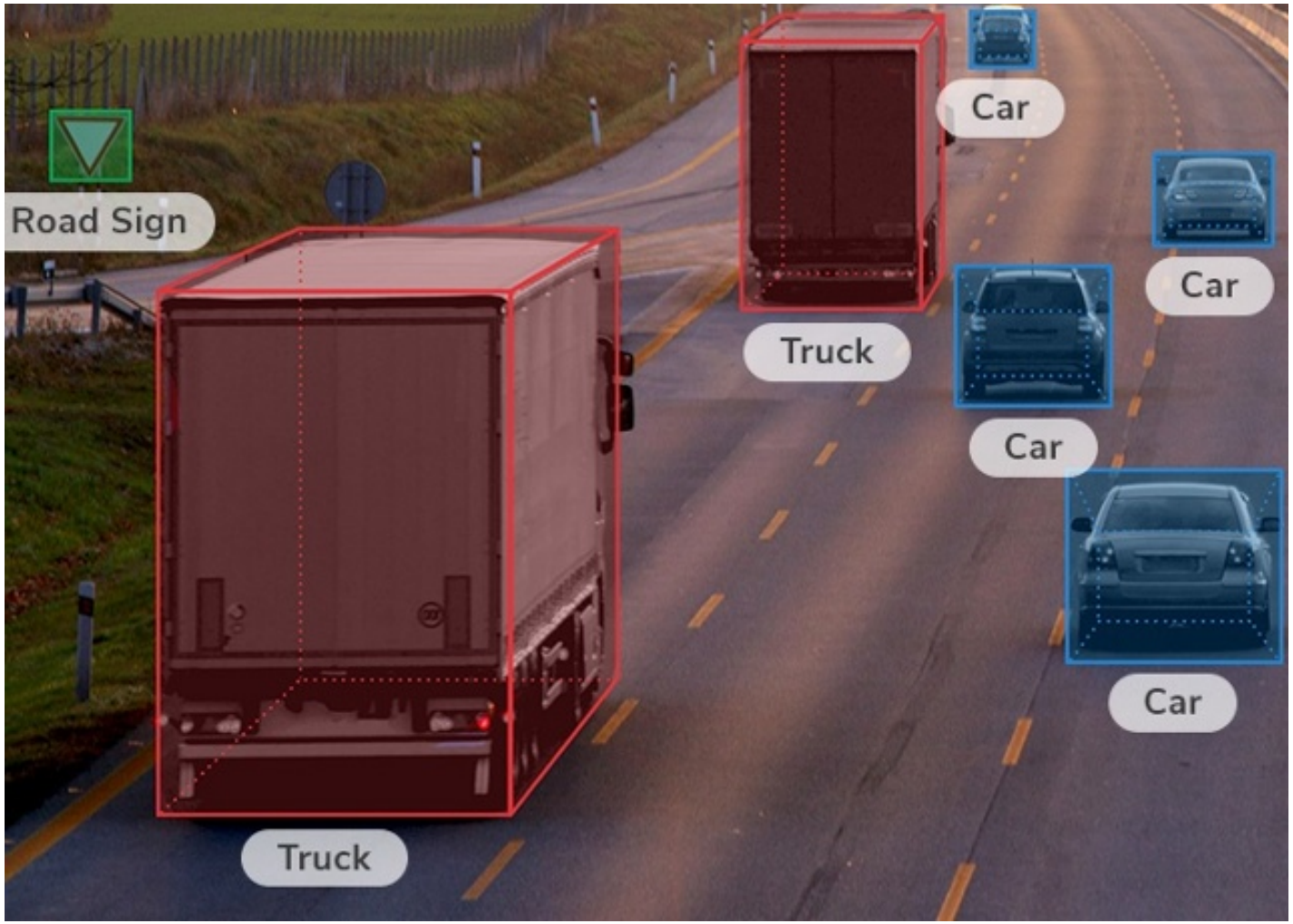
GPUs + Deep Learning + Reinforcement Learning

Robots mimic our behavior



GPUs + Deep Learning + Reinforcement Learning

Robots drive vehicles



By Steve Lohr

Oct 15, 2018

The New York Times

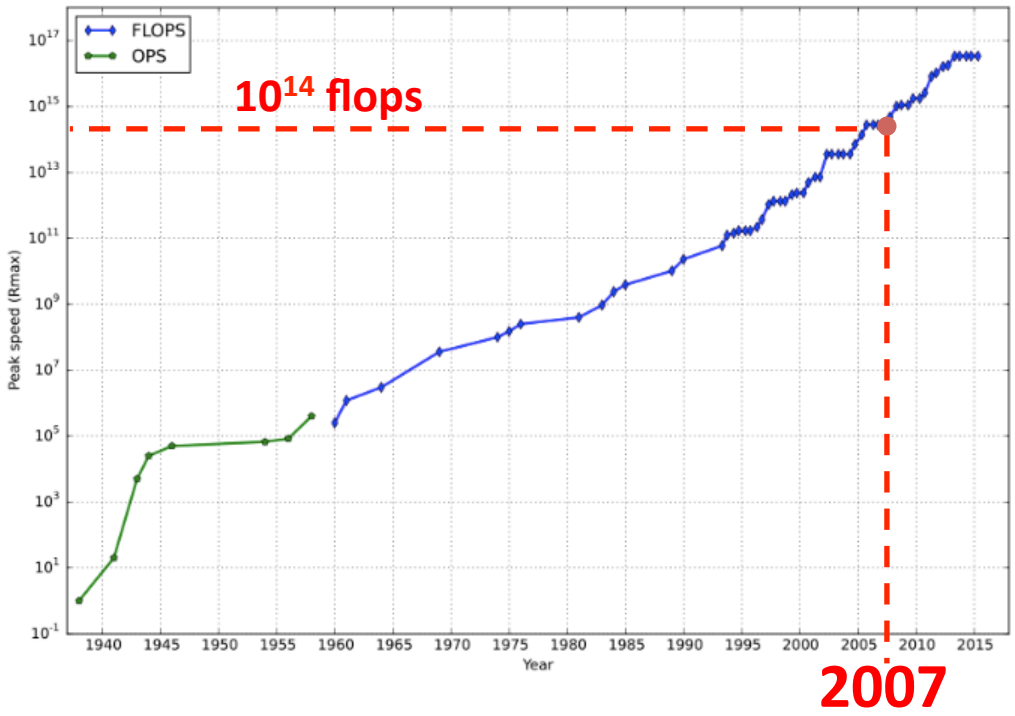
M.I.T. Plans College for Artificial Intelligence, Backed by $1 Billion

Massachusetts Institute of Technology is taking a particularly ambitious step, creating a new college backed by a planned investment of $1 billion. Two-thirds of the funds have already been raised, M.I.T. said, in announcing the initiative on Monday.

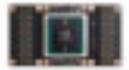
The linchpin gift of $350 million came from Stephen A. Schwarzman, chief executive of the Blackstone Group, the big private equity firm. The college, called the M.I.T. Stephen A. Schwarzman College of Computing, will create 50 new faculty positions and many more fellowships for graduate students.

It is scheduled to begin in the fall semester next year, housed in other buildings before moving into its own new space in 2022.

A 2018 **GPU** is as powerful as the top 2007 supercomputer!



SPECIFICATIONS

	Tesla V100 PCIe	Tesla V100 SXM2
GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.5 TFLOPS
Single-Precision Performance	14 TFLOPS	15 TFLOPS
Tensor Performance	112 TFLOPS	120 TFLOPS
GPU Memory	16 GB HBM2	
Memory Bandwidth	900 GB/sec	

Thank you!