Machine Learning Department
School of Computer Science
Carnegie Mellon University

Quiz 1 Review
+
Convolutional Neural Networks

Matt Gormley
Lecture 11
Mar. 8, 2021

Reminders

- **Homework 2: Exact inference and supervised learning (CRF+RNN)**
  - **Out: Wed, Feb. 24**
  - **Due: Wed, Mar. 10 at 11:59pm**
- **Quiz 1: Mon, Mar. 15**
- **Homework 3: Structured SVM**
  - **Out: Wed, Mar. 10**
  - **Due: Wed, Mar. 4 at 11:59pm**
- **Shortened (10 min) after-class OHs today**

QUIZ 1 LOGISTICS

Quiz 1

- **Time / Location**
  - **Time:** In-Class Quiz
    Mon, Oct. 17 at 6:30pm – 8:00pm
  - **Location:** The same Zoom meeting as lecture/recitation. Please arrive online early.
  - Please watch Piazza carefully for announcements.
- **Logistics**
  - Covered material: Lecture 1 – Lecture 8
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
    - Drawing
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back)

Quiz 1

- **Advice (for before the exam)**
  - Try out the Gradescope quiz-style interface in the "Fake Quiz" now available
- **Advice (for during the exam)**
  - Solve the easy problems first (e.g. multiple choice before derivations)
    - if a problem seems extremely complicated you're likely missing something
  - Don't leave any answer blank!
  - If you make an assumption, write it down
  - If you look at a question and don't know the answer:
    - we probably haven't told you the answer
    - but we've told you enough to work it out
    - imagine arguing for some answer and see if you like it

Topics for Quiz 1

- Graphical Model Representation
  - Directed GMs vs. Undirected GMs vs. Factor Graphs
  - Bayesian Networks vs. Markov Random Fields vs. Conditional Random Fields
- Graphical Model Learning
  - Fully observed Bayesian Network learning
  - Fully observed MRF learning
  - Fully observed CRF learning
  - Parameterization of a GM
  - Neural potential functions
- Exact Inference
  - Three inference problems:
    (1) marginals
    (2) partition function
    (3) most probably assignment
  - Variable Elimination
  - Belief Propagation (sum-product and max-product)

SAMPLE QUESTIONS

Sample Questions

6 Factor Graphs

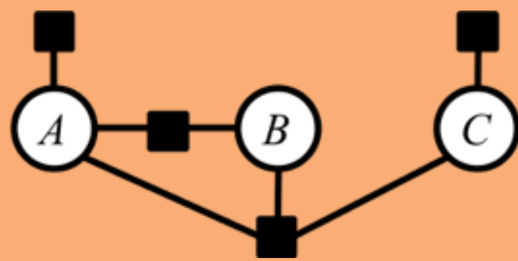


Figure 4: A factor graph over three binary random variables A, B, C, i.e. sampled values a, b, c from the random variables are in $\{0, 1\}$. Assume the factors are named $\psi_A(a)$, $\psi_{A,B}(a, b)$, $\psi_{A,B,C}(a, b, c)$, and $\psi_C(c)$.

1. (2 points) **Short answer:** Consider the factor graph in Figure 4. Using the given factor names, write the partition function Z that ensures the joint probability distribution $p(a, b, c)$ sums-to-one.

Sample Questions

6 Factor Graphs

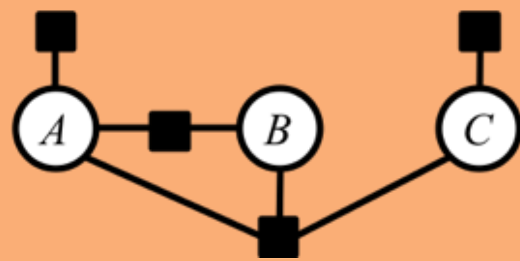


Figure 4: A factor graph over three binary random variables A, B, C, i.e. sampled values a, b, c from the random variables are in $\{0, 1\}$. Assume the factors are named $\psi_A(a)$, $\psi_{A,B}(a, b)$, $\psi_{A,B,C}(a, b, c)$, and $\psi_C(c)$.

2. (2 points) **Short answer:** Using the given factor names, write the joint probability mass function $p(a, b, c)$ defined by the factor graph shown in Figure 4. *You may include the term Z directly in your answer—no need to copy it from above.*

Sample Questions

6 Factor Graphs

3. (2 points) **Drawing:** Suppose we have a joint probability distribution that factorizes as below:

$$p(w, x, y, z) \propto \psi_X(x)\psi_{X,Y}(x, y)\psi_{X,Y,Z}(x, y, z)\psi_{W,Z}(w, z)\psi_{Y,Z}(y, z)$$

where $\propto$ denotes *proportional to*. Draw the factor graph corresponding to this factorization of the joint distribution.

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

1. (2 points) **Short answer:** Draw a table containing all values of the function $s(q,r) = \psi_Q(q)\psi_{Q,R}(q,r)$. *You may use the integer abbreviations: red=1, green=2, blue=3, pencil=1, crayon=2.*

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

2. (2 points) **Numerical answer:** What is the value of the partition function Z for the joint distribution $p(q, r)$?

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

3. (2 points) **Numerical answer:** What is the value of the joint probability $P(Q = green, R = crayon)$? *You may leave your answer in the form of an unsimplified fraction—no calculator necessary.*

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

4. (2 points) **Numerical answer:** What is the value of the marginal probability $P(Q = green)$? *You may leave your answer in the form of an unsimplified fraction—no calculator necessary.*

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

5. (2 points) **Short answer:** Suppose you run the Variable Elimination algorithm to eliminate the variable Q, resulting in a new factor graph with just one factor $m(r)$. Draw a table containing the values of this new factor.

Sample Questions

7 Inference in Graphical Models

Consider yet another factor graph consisting of two random variables $Q \in \{\text{red,green,blue}\}$, $R \in \{\text{pencil, crayon}\}$. Suppose we have the following factors:

Q	$\psi_Q(q)$
red	3
green	1
blue	2

Q	R	$\psi_{Q,R}(q,r)$
red	pencil	2
red	crayon	2
green	pencil	1
green	crayon	3
blue	pencil	4
blue	crayon	1

6. (2 points) **Numerical answer:** What is the value of the marginal probability $P(R = crayon)$? *You may leave your answer in the form of an unsimplified fraction—no calculator necessary.*

Sample Questions

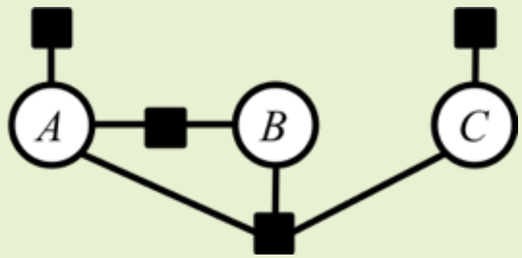


Figure 4: A factor graph over three binary random variables A, B, C, i.e. sampled values a, b, c from the random variables are in $\{0, 1\}$. Assume the factors are named $\psi_A(a)$, $\psi_{A,B}(a, b)$, $\psi_{A,B,C}(a, b, c)$, and $\psi_C(c)$.

1. (1 point) **Drawing**: Suppose you are running the Variable Elimination algorithm. The first variable you eliminate is B. Draw the factor graph that results after you have eliminated variable B.

Sample Questions

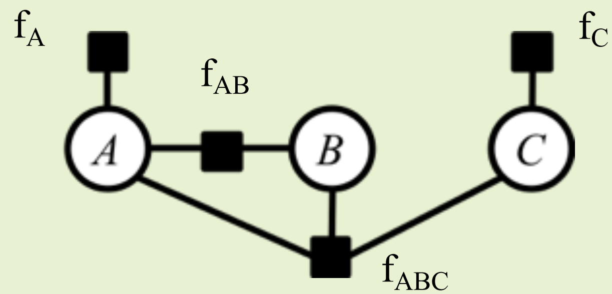
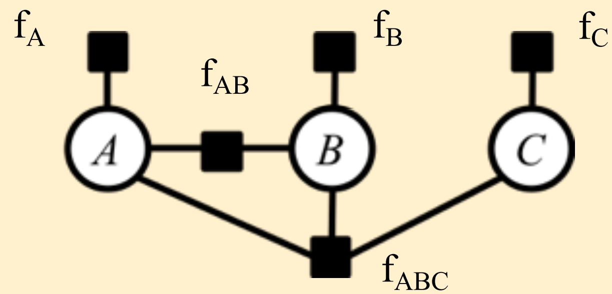


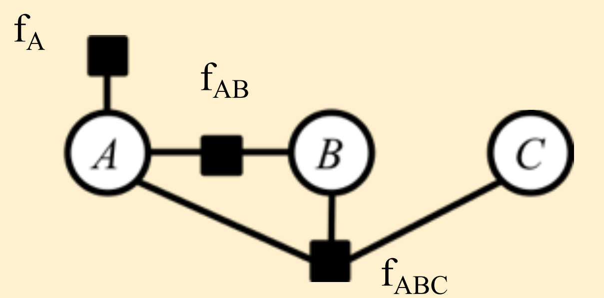
Figure 4: A factor graph over three binary random variables A, B, C, i.e. sampled values a, b, c from the random variables are in $\{0, 1\}$. Assume the factors are named $\psi_A(a)$, $\psi_{A,B}(a, b)$, $\psi_{A,B,C}(a, b, c)$, and $\psi_C(c)$.

2. (1 point) **Numerical Answer**: Suppose you are running the Belief Propagation algorithm? How many messages are required to send a message from f_{ABC} to C?

Sample Questions



1. (1 point) Is there a Bayesian Network that would convert to the factor graph shown above? Is yes, draw an example of such a Bayesian Network. If not, explain why not.



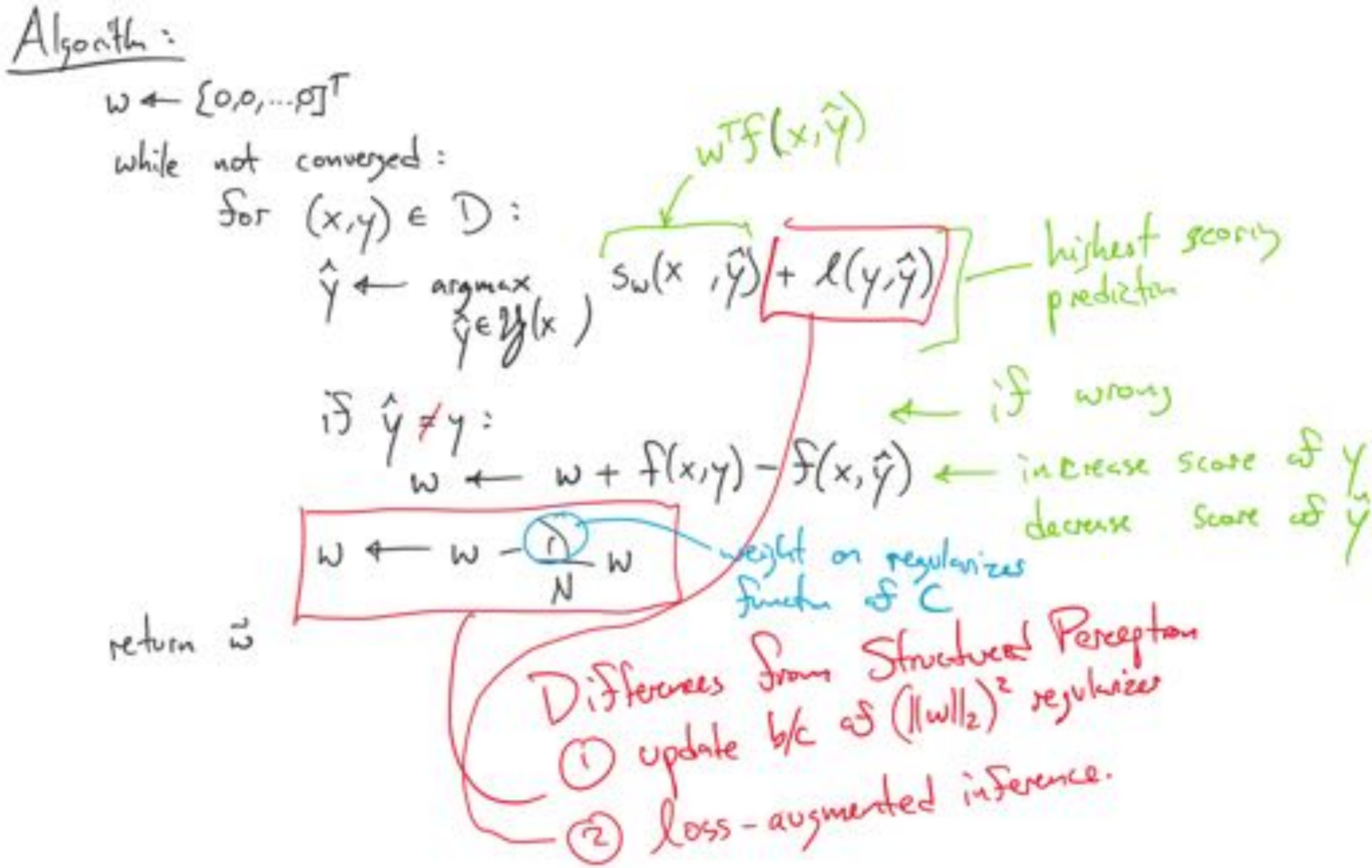
2. (1 point) Is there a Bayesian Network that would convert to the factor graph shown above? Is yes, draw an example of such a Bayesian Network. If not, explain why not.

Q&A

aka. Max-Margin Markov Networks (M^3Ns)

STRUCTURED SVM

SGD for Structured SVM



Structured SVM

Whiteboard

- Structured Large Margin
- Structured Hinge Loss
- Gradient of Structured Hinge Loss
- SGD for Structured SVM
- Loss Augmented MAP Inference

Max vs "Soft-Max" Margin

- SVMs:

$$\min_{\mathbf{w}} k||\mathbf{w}||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y) \right)}_{\text{Hard (Penalized) Margin}} \right)$$

- Maxent:

$$\min_{\mathbf{w}} \; k||w||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)}_{\text{Soft Margin}} \right)$$

- Very similar! Both try to make the true score better than a function of the other scores.
  - The SVM tries to beat the augmented runner-up
  - The maxent classifier tries to beat the "soft-max"

Structured SVM

The original name for **Structured SVM:**

- **Max-Margin Markov Networks**
- **abbreviated as** M3Ns

Results: Handwriting Recognition

Length: ~8 chars

Letter: 16x8 pixels

10-fold Train/Test

5000/50000 letters

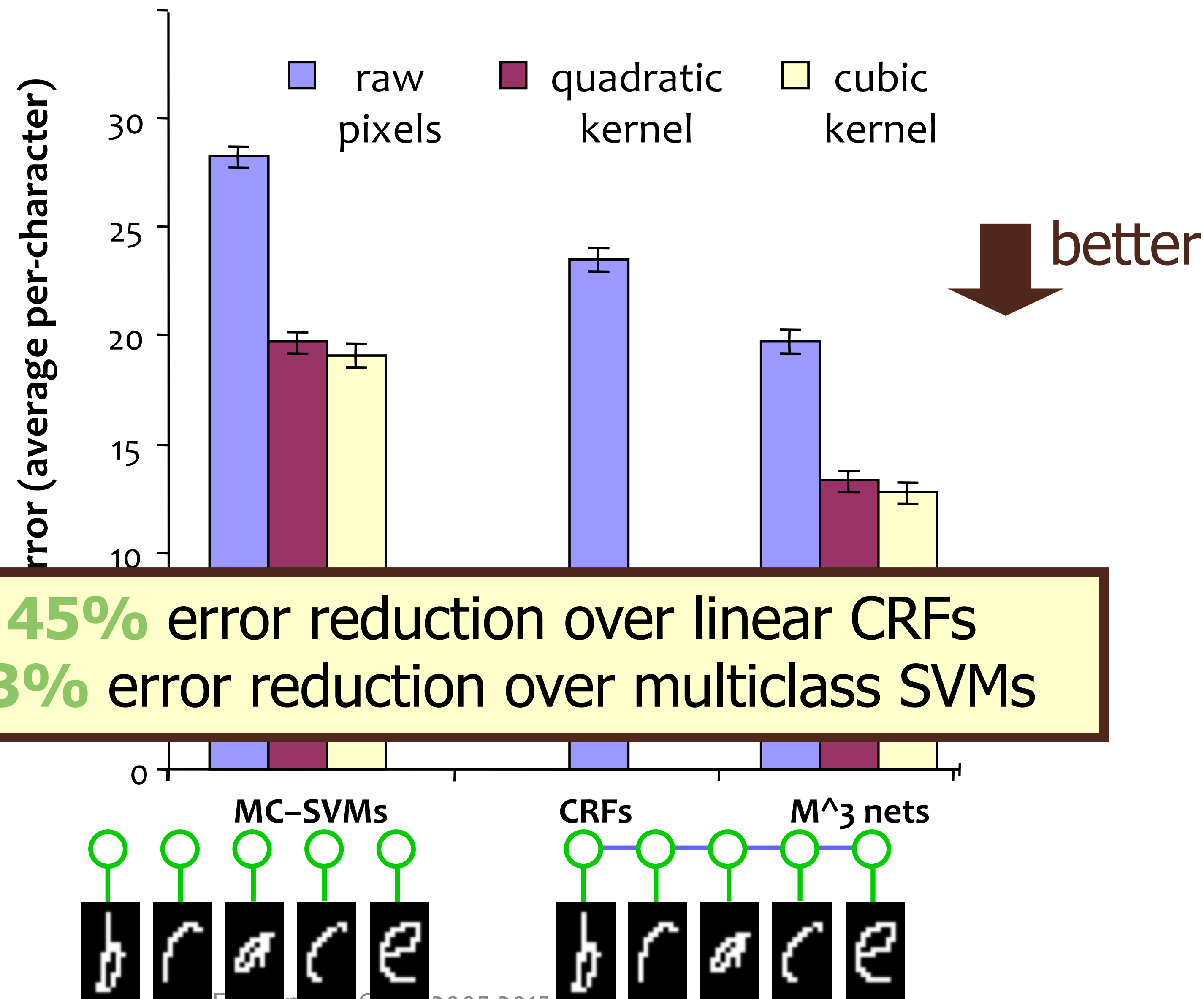
600/6000 words

Models:

Multiclass-SVMs*

CRFs

M^3 nets



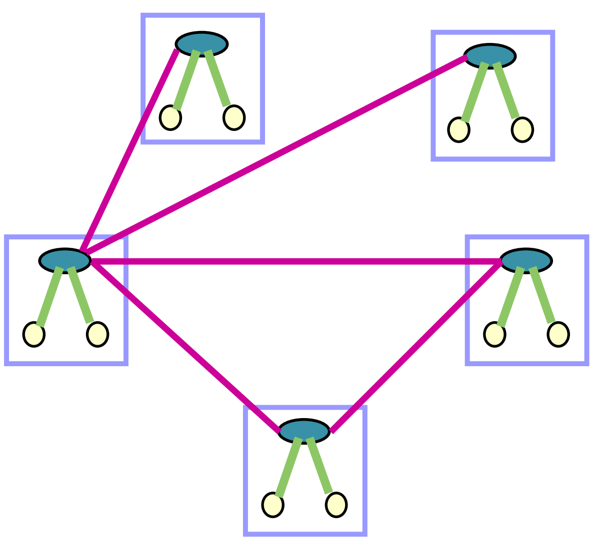
45% error reduction over linear CRFs

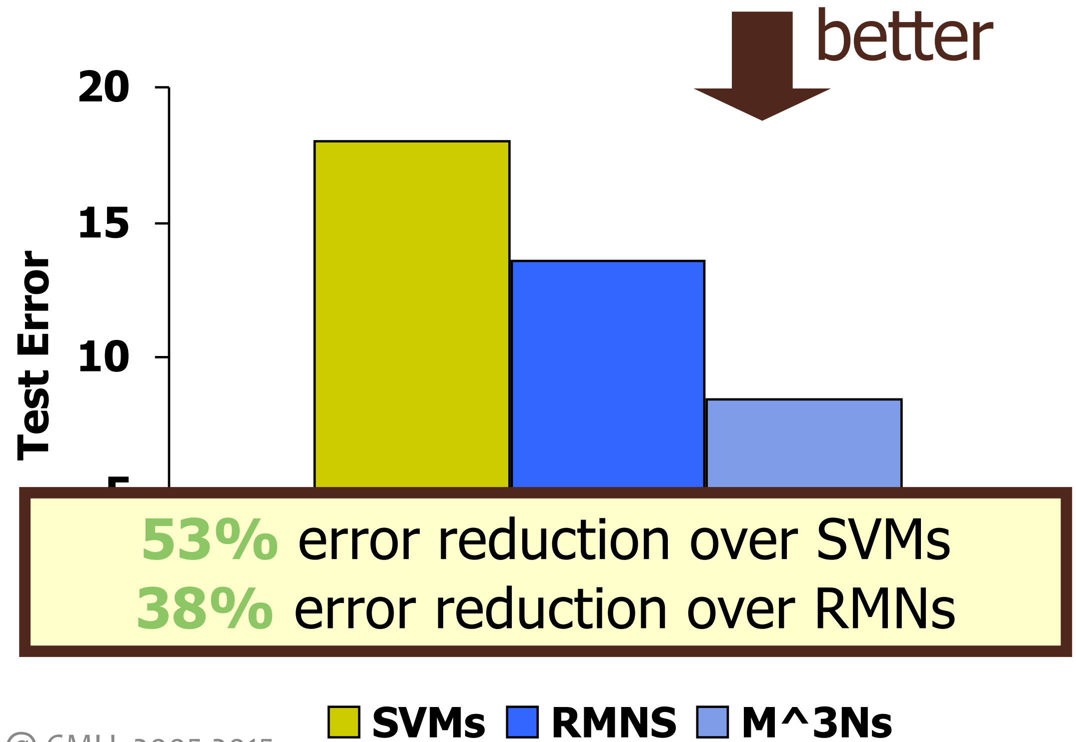
33% error reduction over multiclass SVMs

*Crammer & Singer 01

Results: Hypertext Classification

- WebKB dataset
  - Four CS department websites: 1300 pages/3500 links
  - Classify each page: faculty, course, student, project, other
  - Train on three universities/test on fourth
- Inference: loopy belief propagation
- Learning: relaxed dual

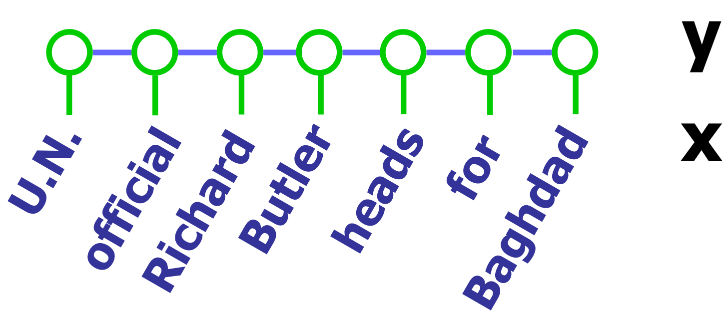




*Taskar et al 02

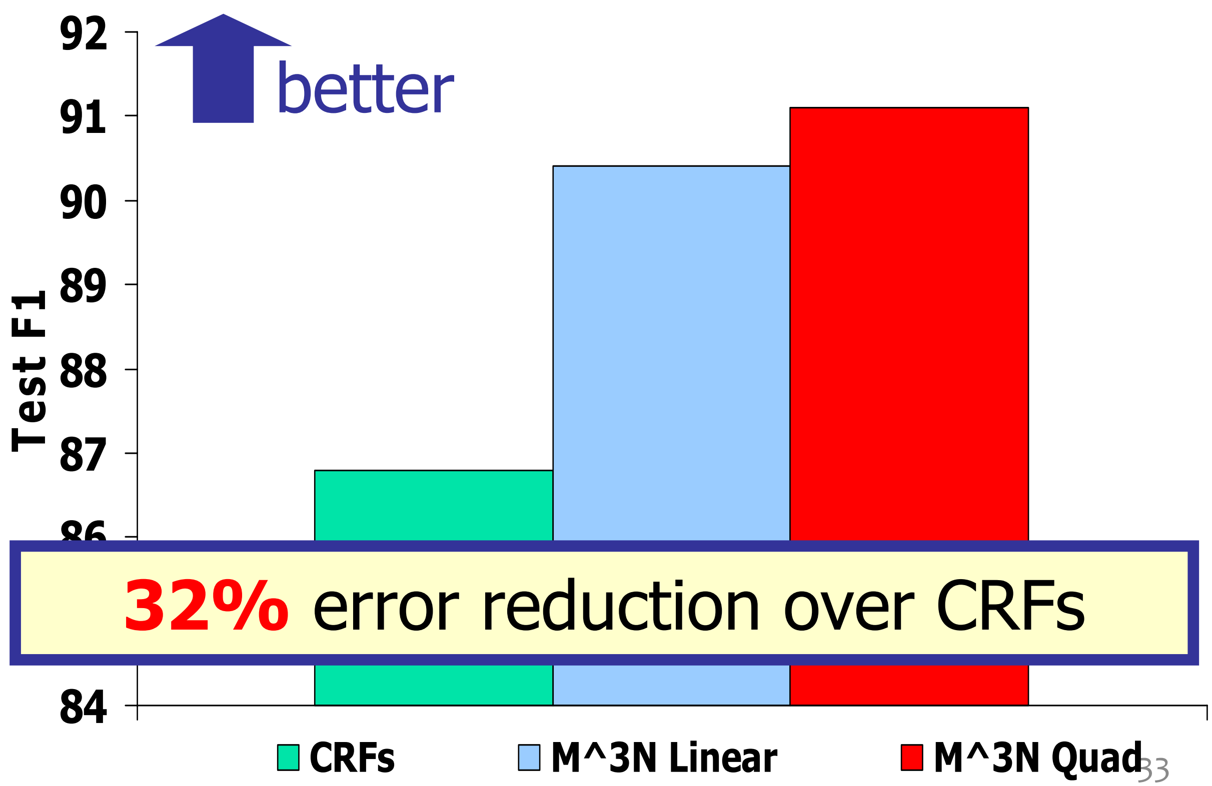
Named Entity Recognition

- Locate and classify named entities in sentences:
  - 4 categories: organization, person, location, misc.
  - e.g. "U.N. official Richard Butler heads for Baghdad".
- CoNLL 03 data set (200K words train, 50K words test)



y_i = org/per/loc/misc/none

$\mathbf{f}(y_i, x) = [\ldots,$
$I(y_i=\text{org}, x_i=\text{"U.N."}),$
$I(y_i=\text{per}, x_i=\text{capitalized}),$
$I(y_i=\text{loc}, x_i=\text{known city}),$
$\ldots,]$



32% error reduction over CRFs

Associative Markov networks

$$P(\mathbf{y} \mid \mathbf{x}) \propto \prod_i \phi_i(y_i, \mathbf{x}_i) \prod_{ij} \phi_{ij}(y_i, y_j, \mathbf{x}_{ij}) = \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$$

Point features
spin-images, point height

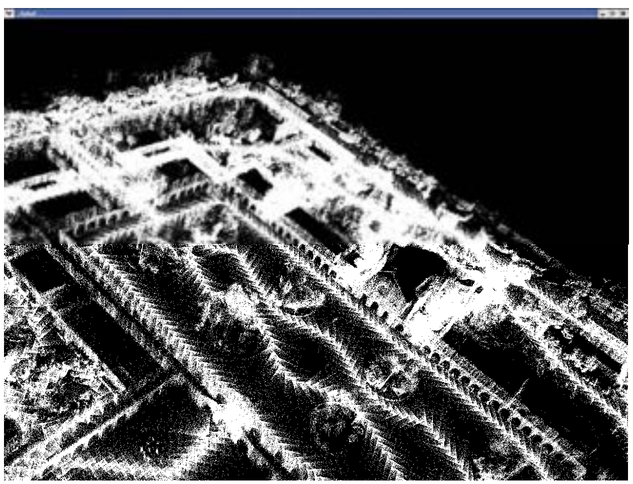
Edge features
length of edge, edge orientation

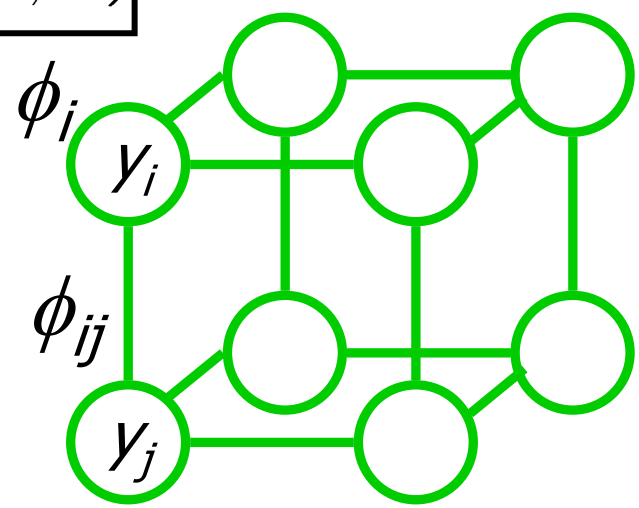
"associative" restriction

$$\phi_{ij}(y_i, y_j) = \begin{bmatrix} \phi_{ij}(1,1) & & 1 \\ & \ddots & \\ 1 & & \phi_{ij}(K,K) \end{bmatrix}$$

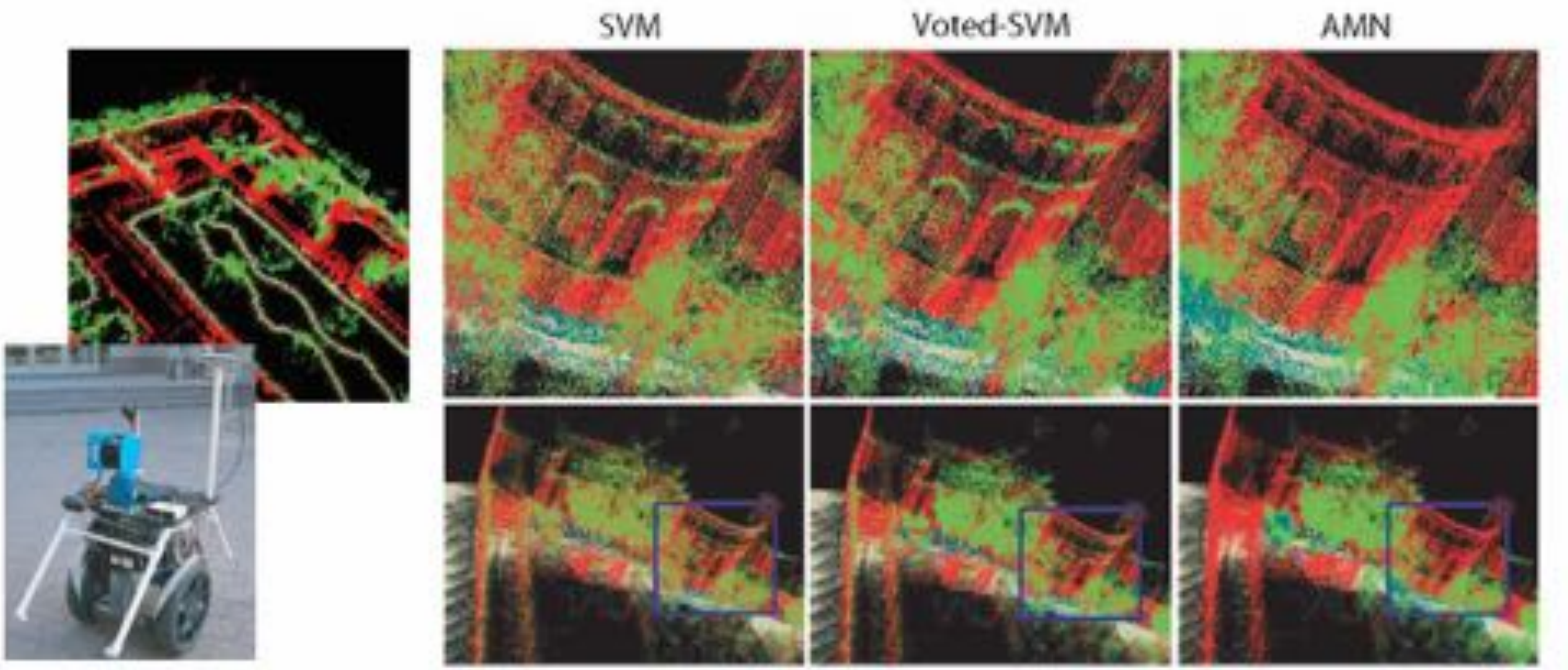
bonus

$$\phi_{ij}(k,k) \geq 1$$





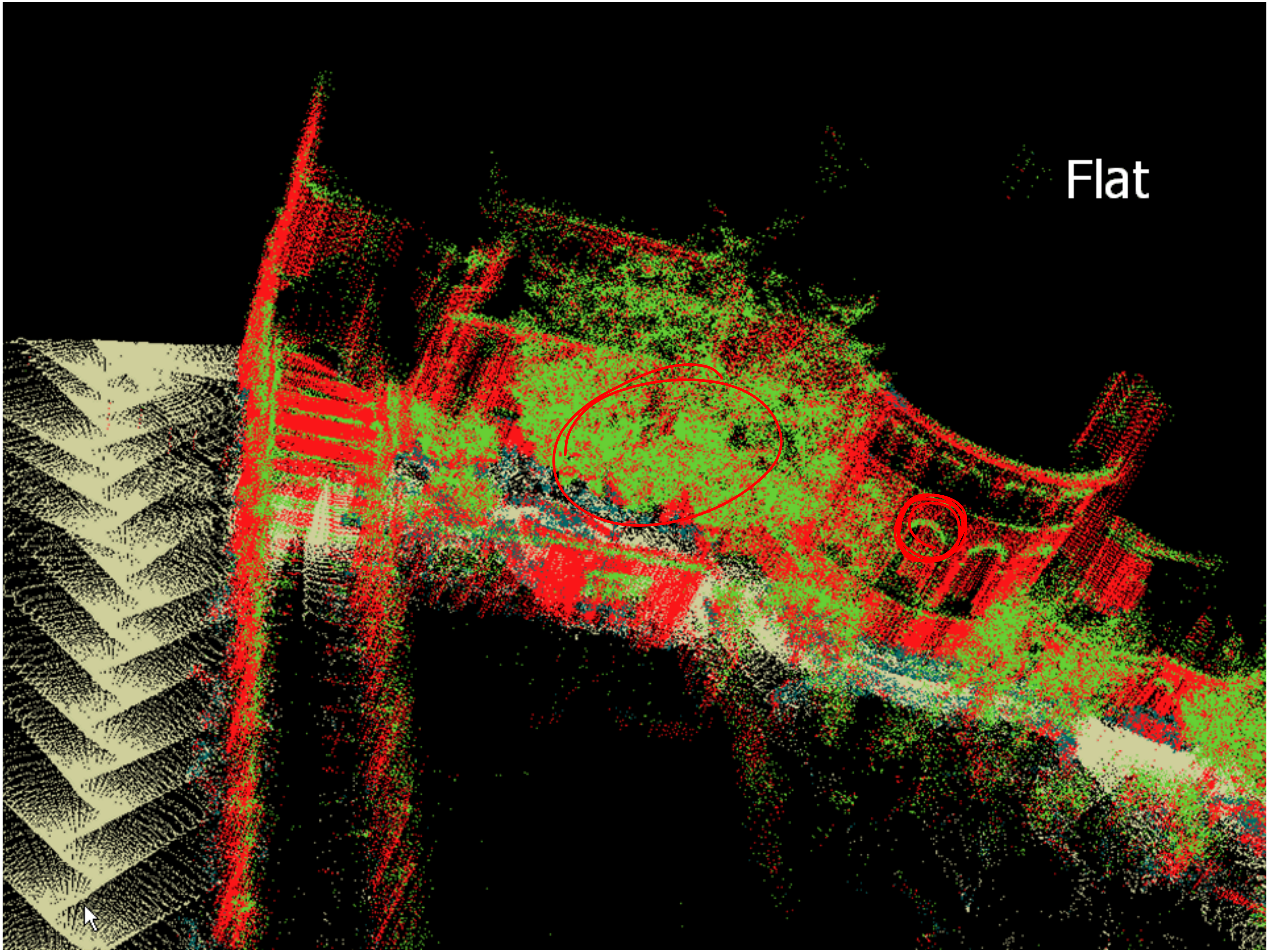
Max-margin AMNs results

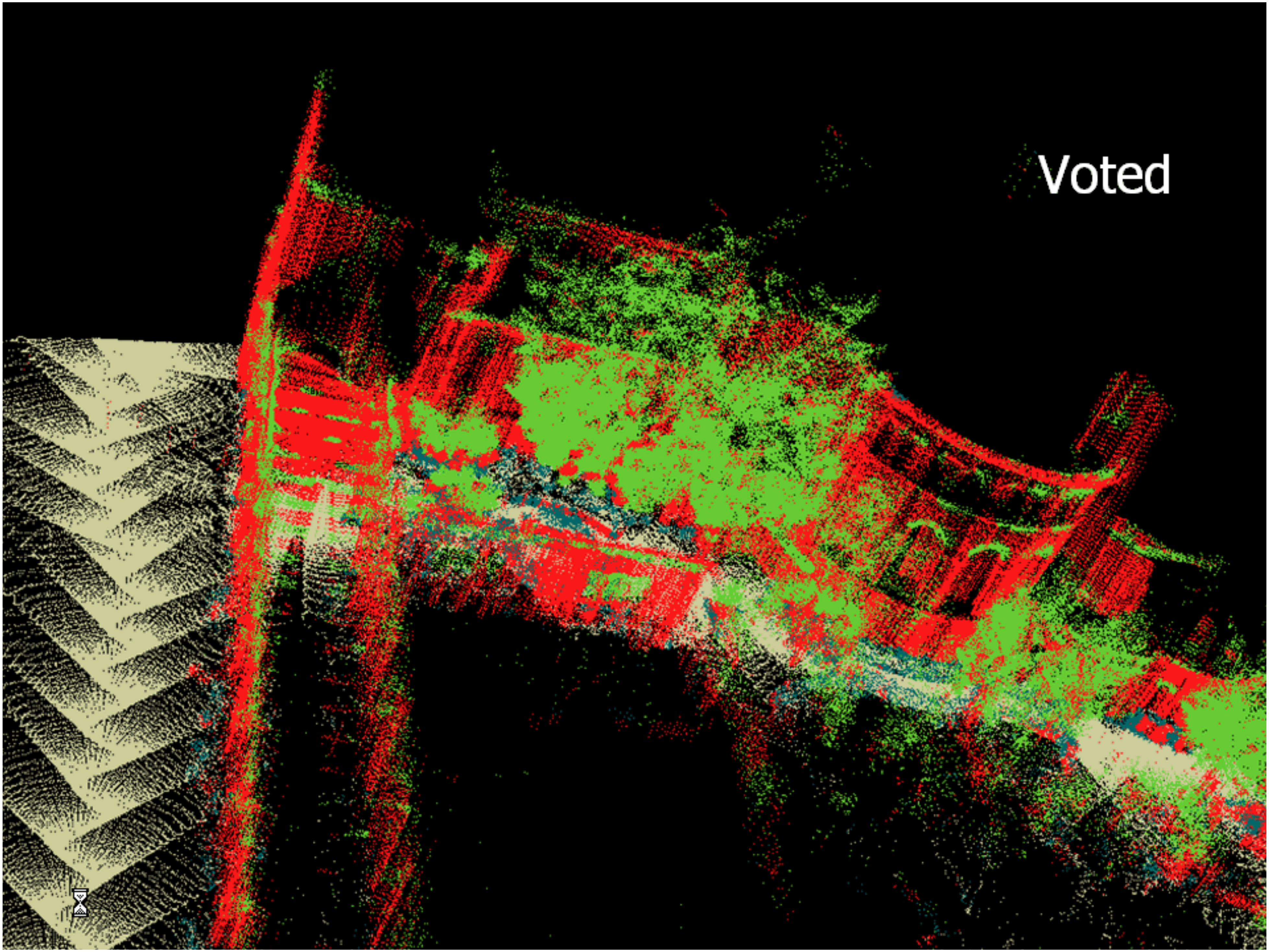


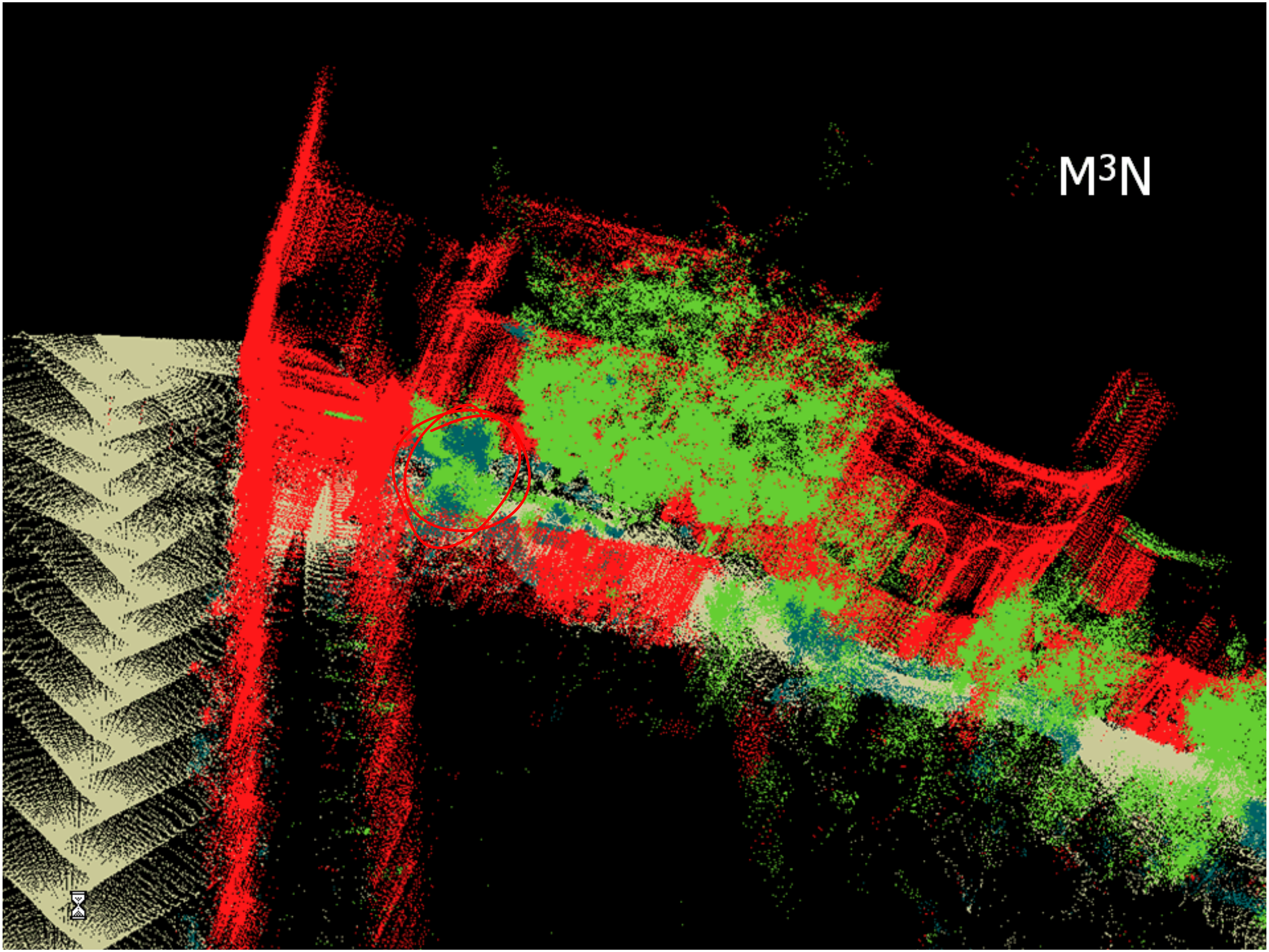
Label: ground, building, tree, shrub

Training: 30 thousand points   Testing: 3 million points

Slide from Guestrin, 10-701, 2005



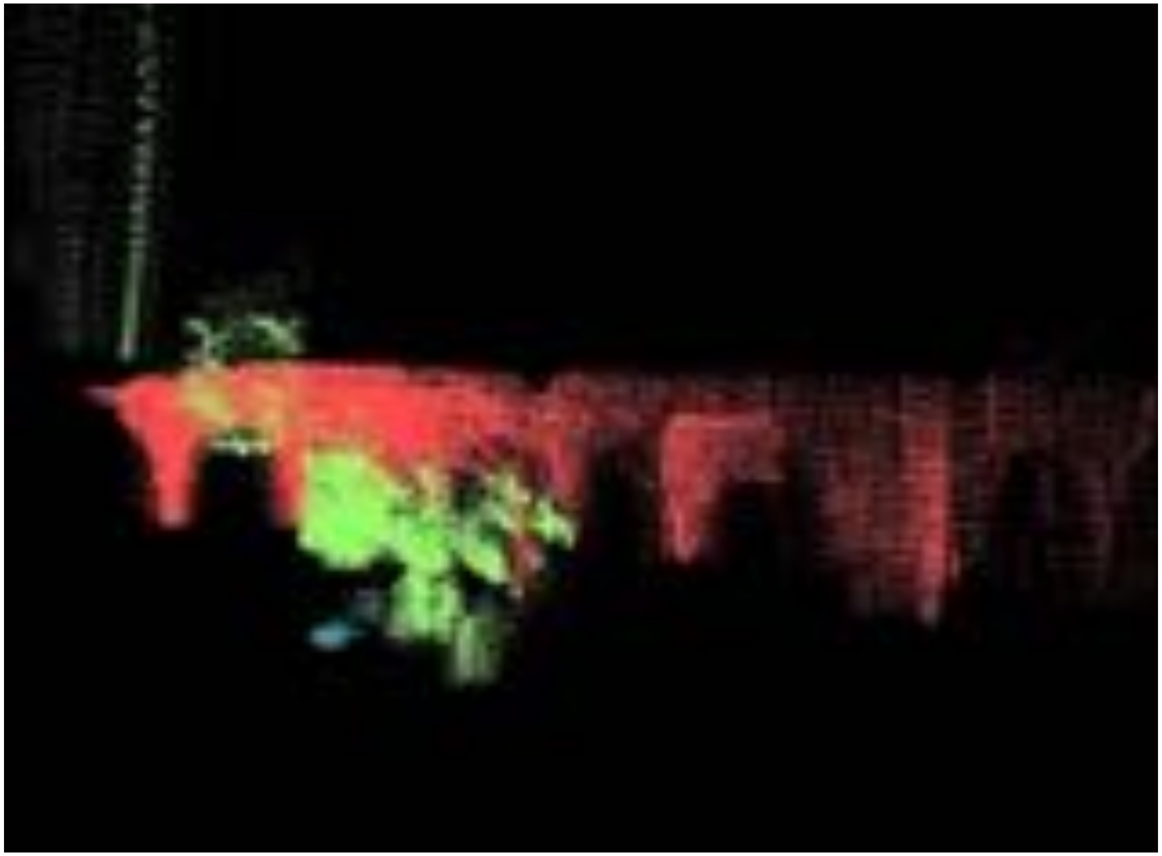

Voted


M³N

Segmentation results

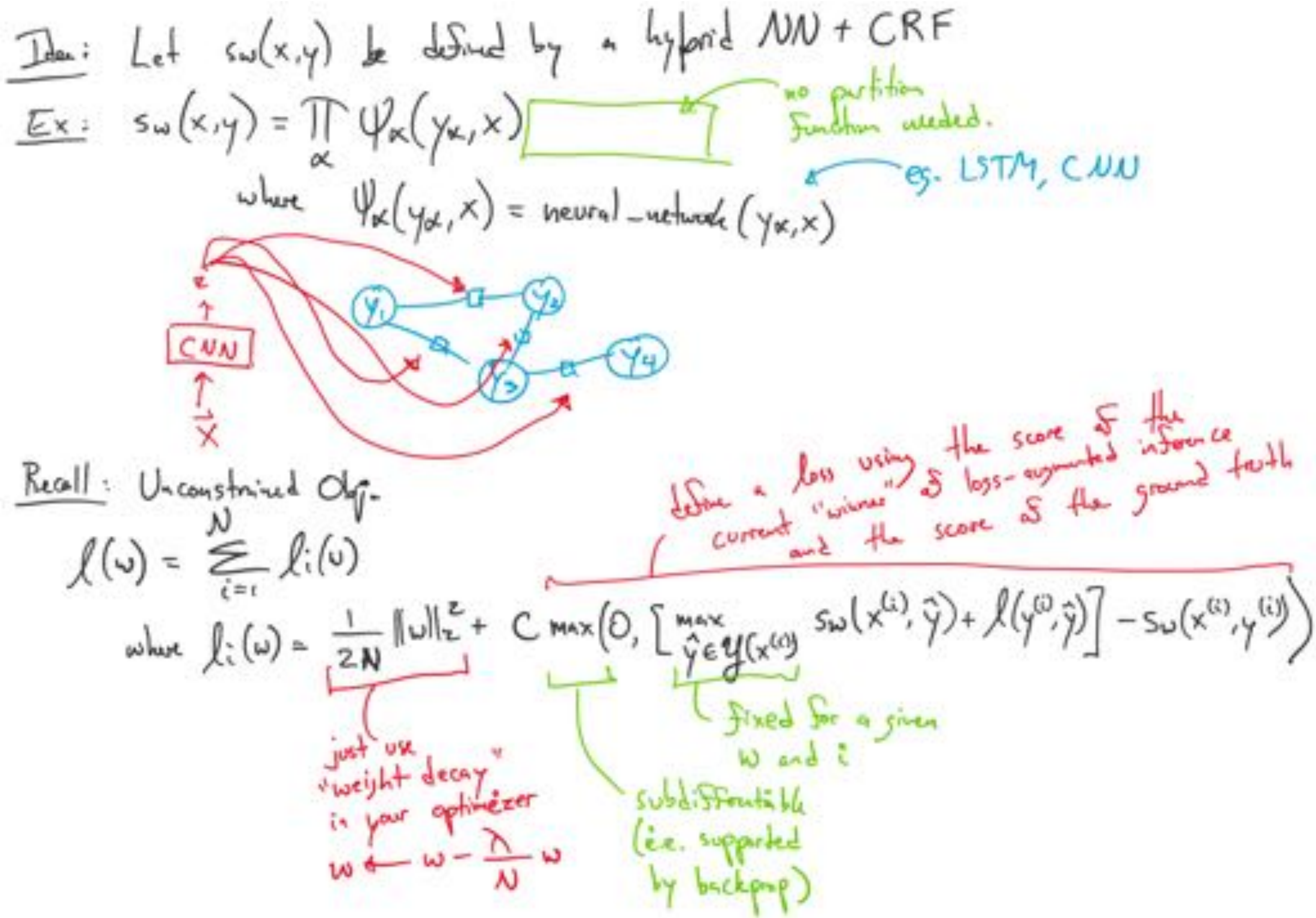
Hand labeled 180K test points

Model	Accuracy
SVM	68%
V-SVM	73%
M^3N	93%



STRUCTURED SVM WITH NEURAL POTENTIALS

Structured SVM with Neural Potentials



Hinge Losses in Deep Learning

Application of structured support vector machine backpropagation to a convolutional neural network for human pose estimation

Peerajak Witoonchart*, Prabhas Chongstitvatana*

Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 17th floor, Engineering 4 Building (Charoenvidsavakham), Phayathai Road, Wang Mai, Pathumwan, Bangkok 10330, Thailand

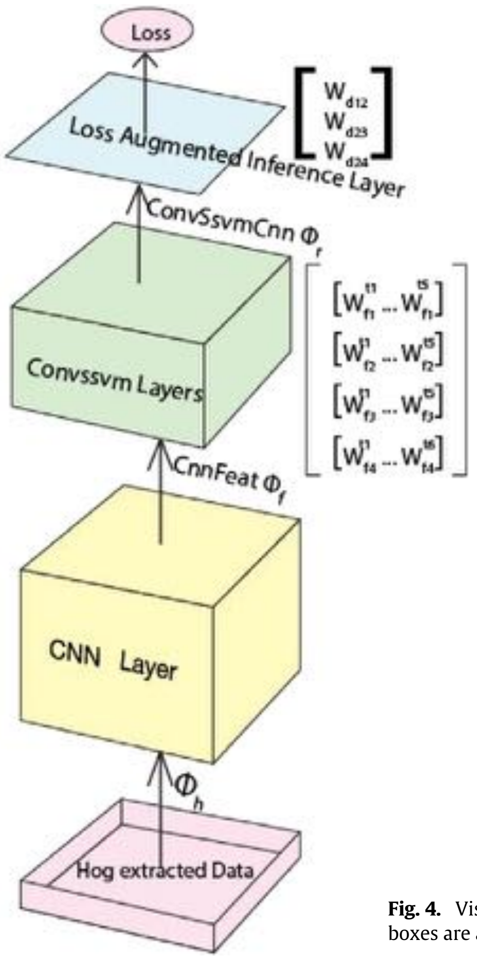


Fig. 4. Visualization of our HPE results based on the PARSE test dataset. The green bounding boxes are a head. The yellow bounding boxes are a torso. The cyan bounding boxes are a left arm. The blue bounding boxes are a right arm. The red bounding boxes are a left limb. The deep blue bounding boxes are a right limb.

Hinge Losses in Deep Learning

Sequence-to-Sequence Learning as Beam-Search Optimization

Sam Wiseman and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,srush}@seas.harvard.edu

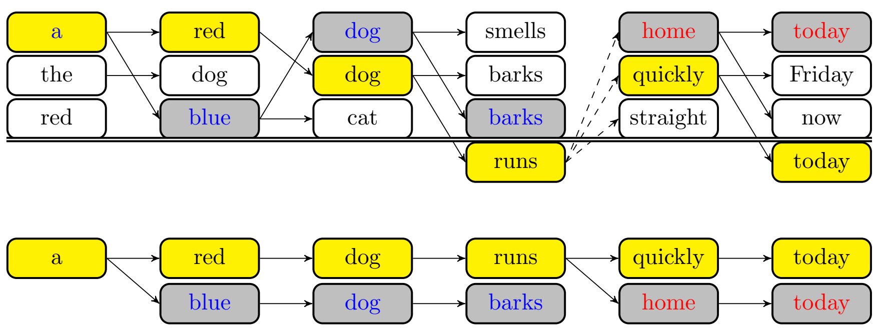


Figure 1: Top: possible $\hat{y}^{(k)}_{1:t}$ formed in training with a beam of size $K = 3$ and with gold sequence $y_{1:6}$ = "a red dog runs quickly today". The gold sequence is highlighted in yellow, and the predicted prefixes involved in margin violations (at $t = 4$ and $t = 6$) are in gray. Note that time-step $T = 6$ uses a different loss criterion. Bottom: prefixes that actually participate in the loss, arranged to illustrate the back-propagation process.

We now define a loss function that gives loss each time the score of the gold prefix $y_{1:t}$ does not exceed that of $\hat{y}^{(K)}_{1:t}$ by a margin:

$$\mathcal{L}(f) = \sum_{t=1}^{T} \Delta(\hat{y}^{(K)}_{1:t}) \left[1 - f(y_t, \boldsymbol{h}_{t-1}) + f(\hat{y}^{(K)}_t, \hat{\boldsymbol{h}}^{(K)}_{t-1})\right].$$

Above, the $\Delta(\hat{y}^{(K)}_{1:t})$ term denotes a mistake-specific cost-function, which allows us to scale the loss depending on the severity of erroneously predicting $\hat{y}^{(K)}_{1:t}$; it is assumed to return 0 when the margin requirement is satisfied, and a positive number otherwise. It is this term that allows us to use sequence- rather than word-level costs in training (addressing the 2nd issue in the introduction). For instance, when training a seq2seq model for machine translation, it may be desirable to have $\Delta(\hat{y}^{(K)}_{1:t})$ be inversely related to the partial sentence-level BLEU score of $\hat{y}^{(K)}_{1:t}$ with $y_{1:t}$; we experiment along these lines in Section 5.3.

Finally, because we want the full gold sequence to be at the top of the beam at the end of search, when $t = T$ we modify the loss to require the score of $y_{1:T}$ to exceed the score of the *highest* ranked incorrect prediction by a margin.

Hinge Losses in Deep Learning

Sequence-to-Sequence Learning as Beam-Search Optimization

Sam Wiseman and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,srush}@seas.harvard.edu

	Machine Translation (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	22.53	24.03	23.87
BSO, SB-Δ	**23.83**	**26.36**	**25.48**
XENT	17.74	20.10	20.28
DAD	20.12	22.25	22.40
MIXER	20.73	21.81	21.83

Table 4: Machine translation experiments on test set; results below middle line are from MIXER model of Ranzato et al. (2016). SB-Δ indicates sentence BLEU costs are used in defining Δ. XENT is similar to our seq2seq model but with a convolutional encoder and simpler attention. DAD trains seq2seq with scheduled sampling (Bengio et al., 2015). BSO, SB-Δ experiments above have $K_{tr} = 6$.

	Dependency Parsing (UAS/LAS)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	**87.33/82.26**	88.53/84.16	88.66/84.33
BSO	86.91/82.11	91.00/**87.18**	91.17/**87.41**
ConBSO	85.11/79.32	**91.25**/86.92	**91.57**/87.26
Andor	93.17/91.18	-	-

Table 3: Dependency parsing. UAS/LAS of seq2seq, BSO, ConBSO and baselines on PTB test set. Andor is the current state-of-the-art model for this data set (Andor et al. 2016), and we note that with a beam of size 32 they obtain 94.41/92.55. All experiments above have $K_{tr} = 6$.

	Word Ordering (BLEU)		
	K_{te} = 1	K_{te} = 5	K_{te} = 10
seq2seq	25.2	29.8	31.0
BSO	28.0	33.2	34.3
ConBSO	**28.6**	**34.3**	**34.5**
LSTM-LM	15.4	-	26.8

Table 1: Word ordering. BLEU Scores of seq2seq, BSO, constrained BSO, and a vanilla LSTM language model (from Schmaltz et al, 2016). All experiments above have $K_{tr} = 6$.

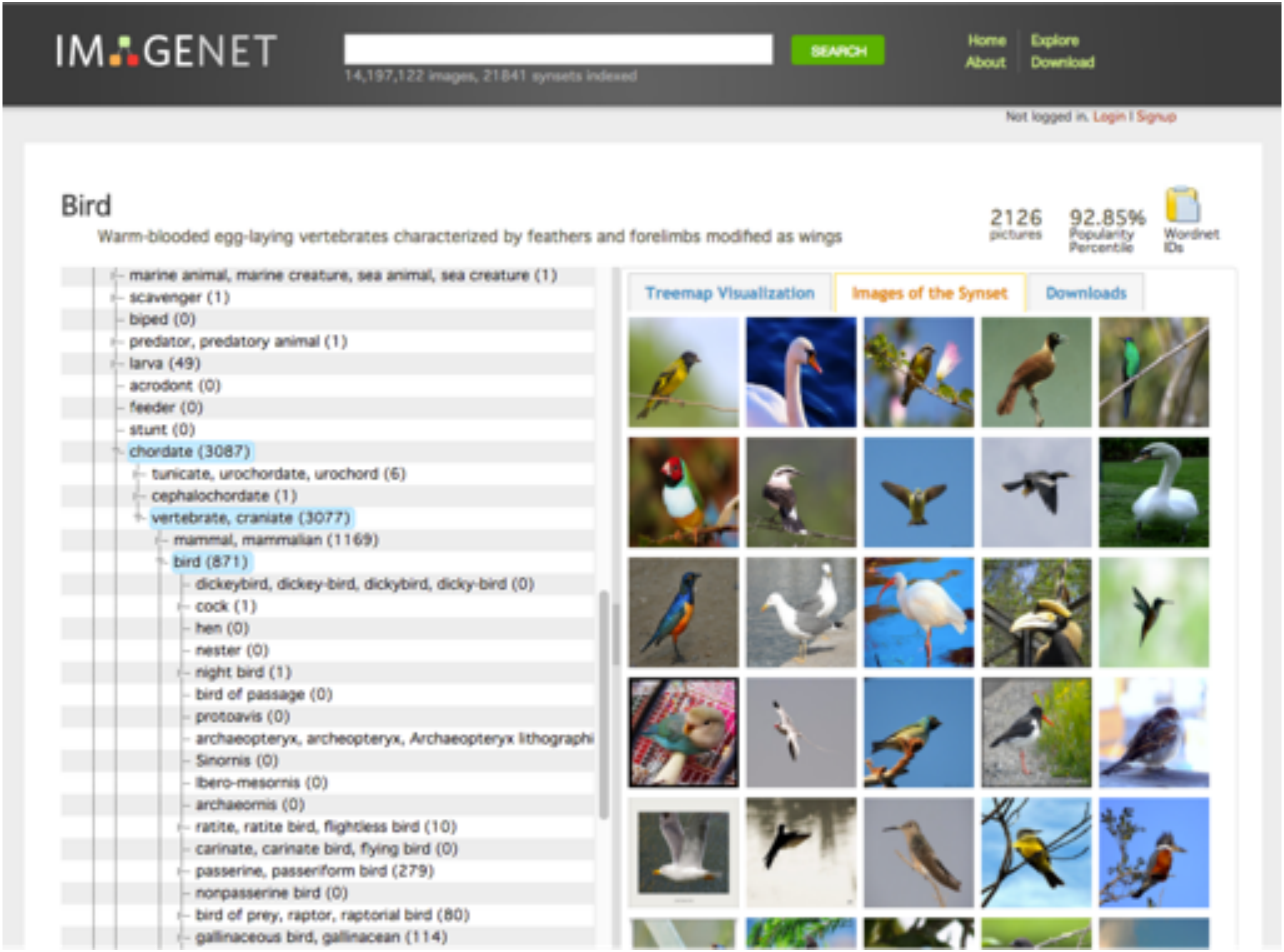
CNNs Outline

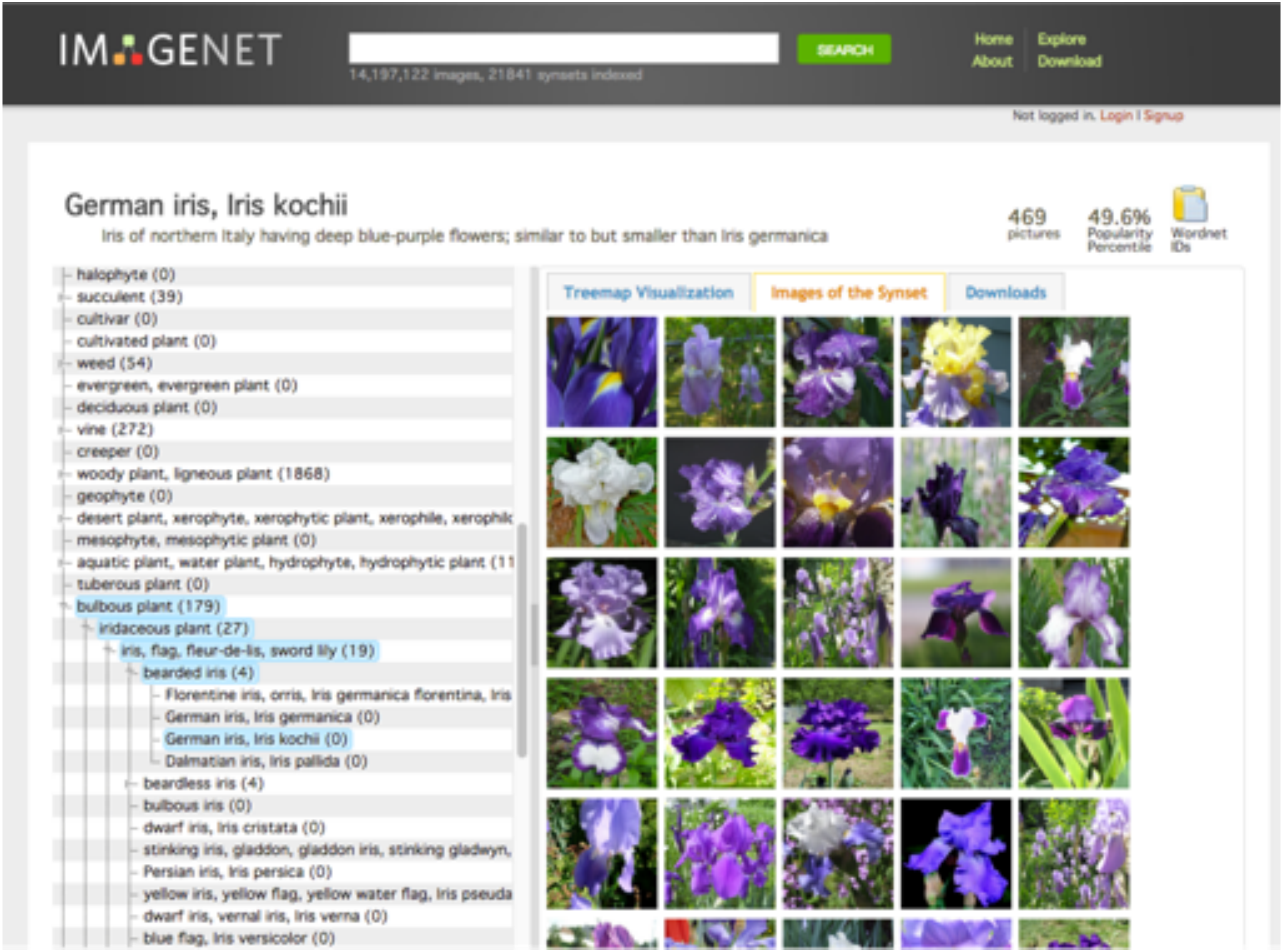
- **Background: Computer Vision**
  - Image Classification
  - ILSVRC 2010 - 2016
  - Traditional Feature Extraction Methods
  - Convolution as Feature Extraction
- **Convolutional Neural Networks (CNNs)**
  - Learning Feature Abstractions
  - Common CNN Layers:
    - Convolutional Layer
    - Max-Pooling Layer
    - Fully-connected Layer (w/tensor input)
    - Softmax Layer
    - ReLU Layer
  - Background: Subgradient
  - Architecture: LeNet
  - Architecture: AlexNet
  - Architecture: ResNet
- **Training a CNN**
  - SGD for CNNs
  - Backpropagation for CNNs

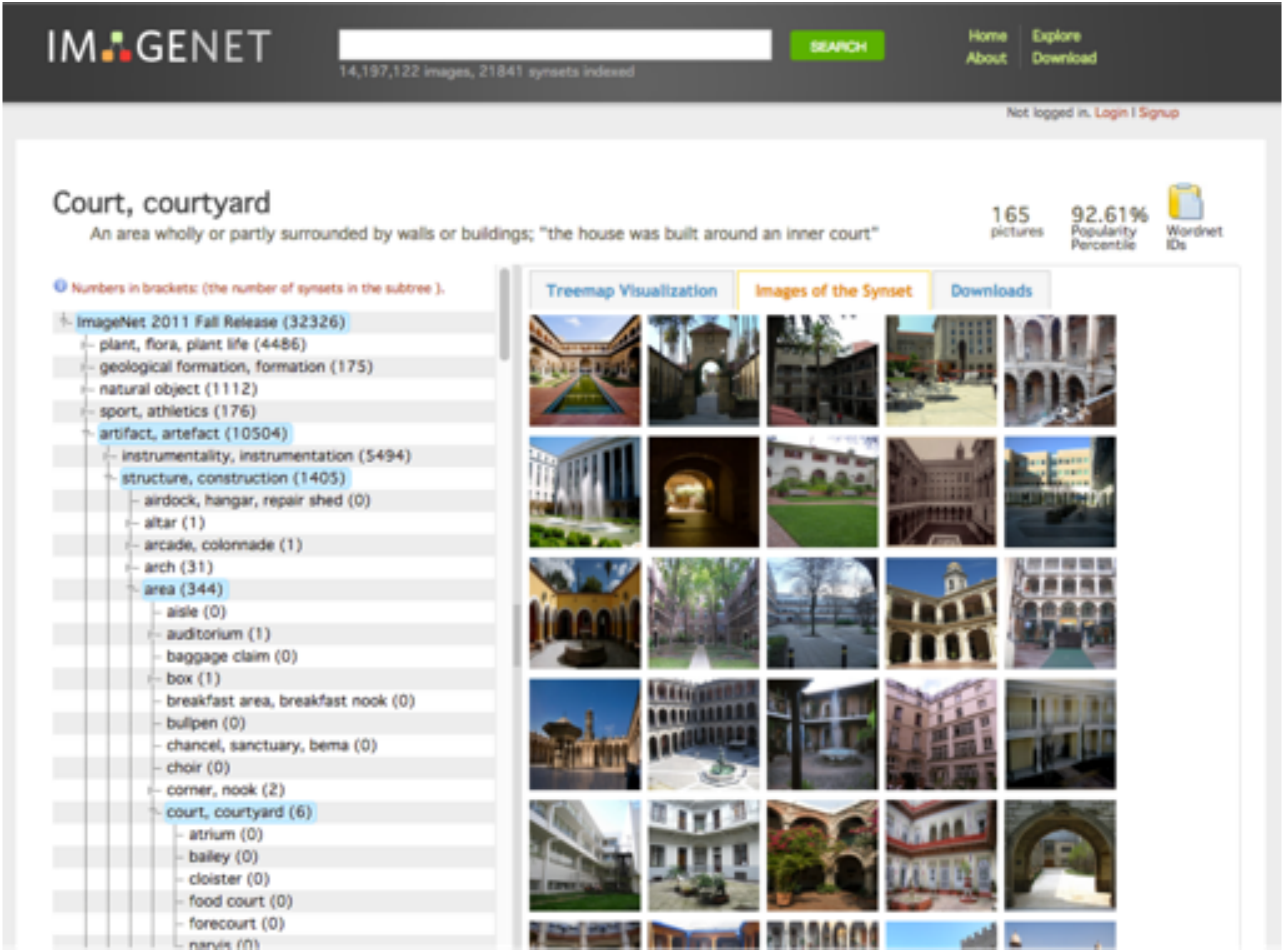
BACKGROUND: COMPUTER VISION

Example: Image Classification

- ImageNet LSVRC-2011 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task**: Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126 pictures
92.85% Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads

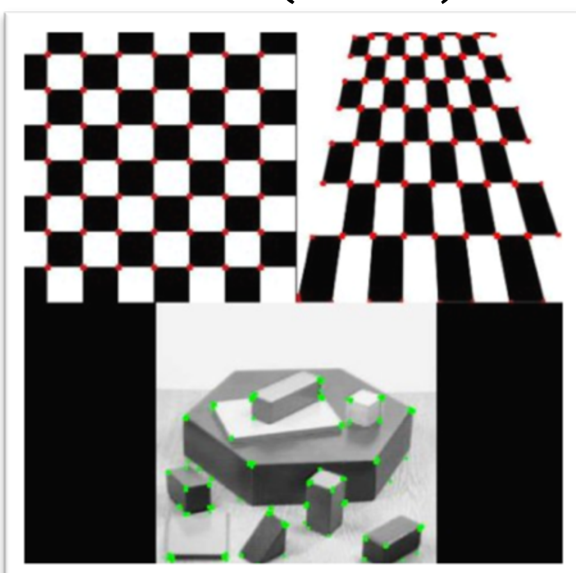

IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup
German iris, Iris kochii
Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica
469 pictures
49.6% Popularity Percentile
Wordnet IDs
halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11
tuberous plant (0)
bulbous plant (179)
iridaceous plant (27)
iris, flag, fleur-de-lis, sword lily (19)
bearded iris (4)
Florentine iris, orris, Iris germanica florentina, Iris
German iris, Iris germanica (0)
German iris, Iris kochii (0)
Dalmatian iris, Iris pallida (0)
beardless iris (4)
bulbous iris (0)
dwarf iris, Iris cristata (0)
stinking iris, gladdon, gladdon iris, stinking gladwyn,
Persian iris, Iris persica (0)
yellow iris, yellow flag, yellow water flag, Iris pseuda
dwarf iris, vernal iris, Iris verna (0)
blue flag, Iris versicolor (0)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Court, courtyard
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"
165 pictures
92.61% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
instrumentality, instrumentation (5494)
structure, construction (1405)
airdock, hangar, repair shed (0)
altar (1)
arcade, colonnade (1)
arch (31)
area (344)
aisle (0)
auditorium (1)
baggage claim (0)
box (1)
breakfast area, breakfast nook (0)
bullpen (0)
chancel, sanctuary, bema (0)
choir (0)
corner, nook (2)
court, courtyard (6)
atrium (0)
bailey (0)
cloister (0)
food court (0)
forecourt (0)
Treemap Visualization
Images of the Synset
Downloads

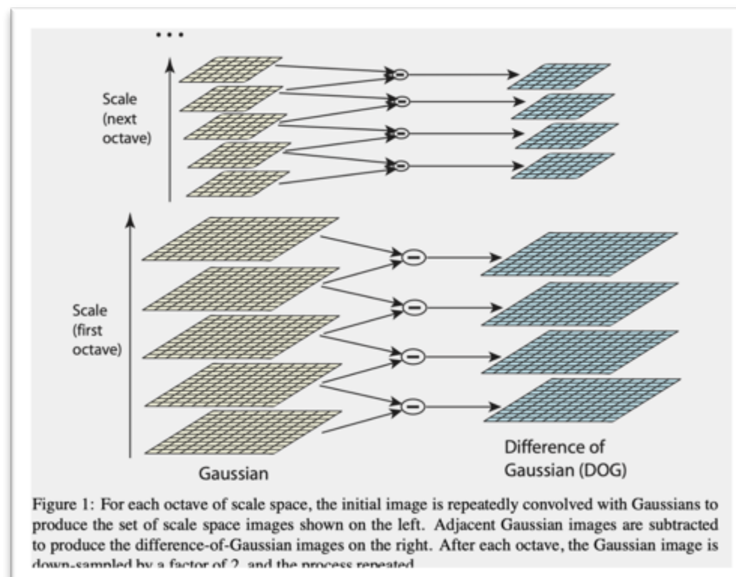
Feature Engineering for CV

Edge detection (Canny)

Corner Detection (Harris)



Scale Invariant Feature Transform (SIFT)



Figures from http://opencv.org

Figure from Lowe (1999) and Lowe (2004)

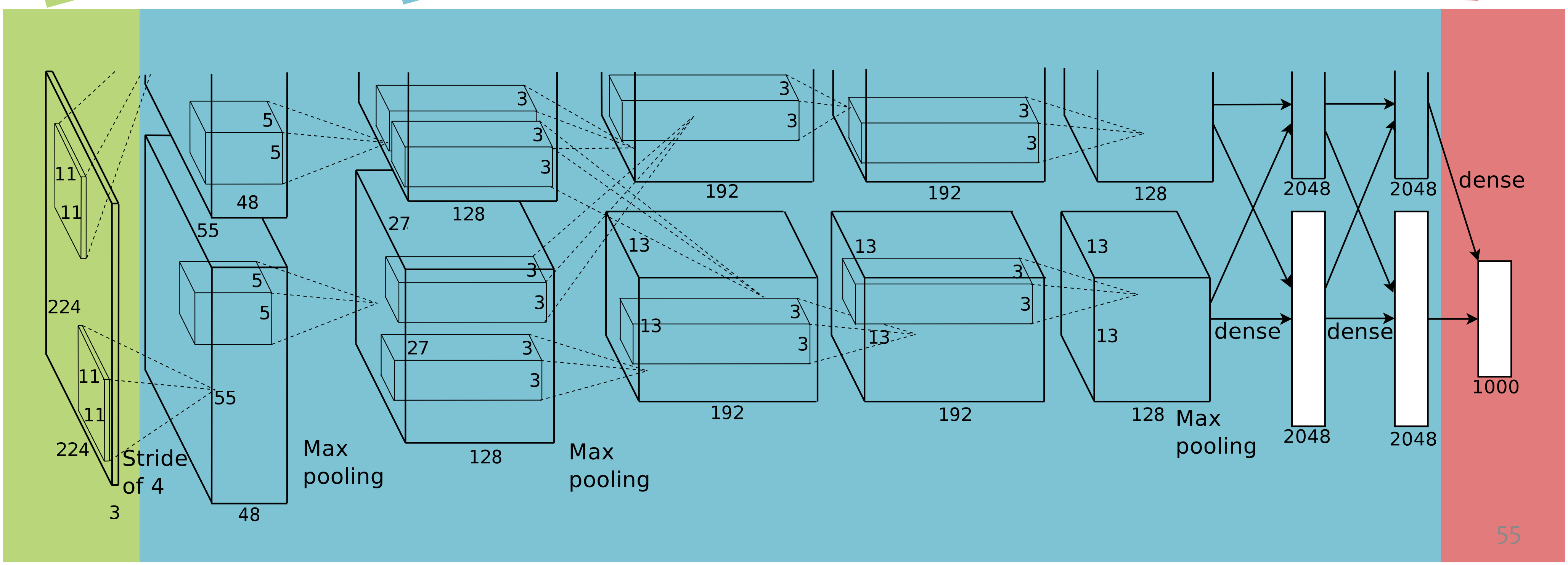
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2012)
15.3% error on ImageNet LSVRC-2012 contest

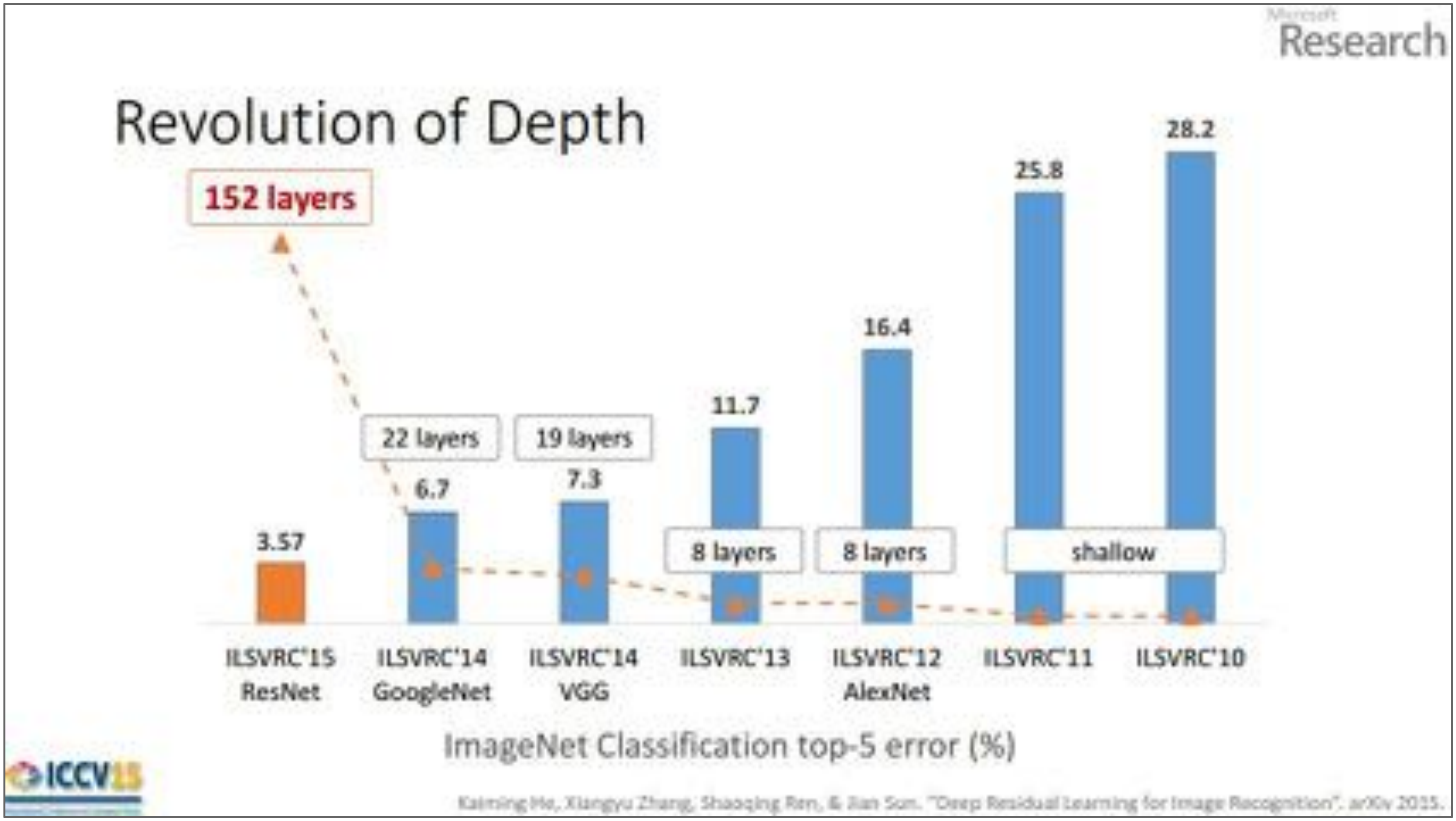
Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



CNNs for Image Recognition



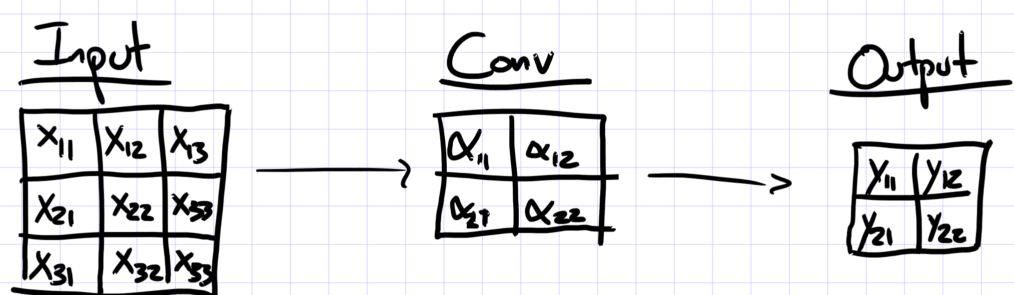
Slide from Kaiming He

CONVOLUTION

What's a convolution?

- Basic idea:
  - Pick a 3x3 matrix F of weights
  - Slide this over an image and compute the "inner product" (similarity) of F and the corresponding field of the image, and replace the pixel in the center of the field with the output of the inner product operation
- Key point:
  - Different convolutions extract different types of low-level "features" from an image
  - All that we need to vary to generate these different features is the weights of F

Ex: 1 input channel, 1 output channel



$$y_{11} = \alpha_{11}x_{11} + \alpha_{12}x_{12} + \alpha_{21}x_{21} + \alpha_{22}x_{22} + \alpha_0$$
$$y_{12} = \alpha_{11}x_{12} + \alpha_{12}x_{13} + \alpha_{21}x_{22} + \alpha_{22}x_{23} + \alpha_0$$
$$y_{21} = \alpha_{11}x_{21} + \alpha_{12}x_{22} + \alpha_{21}x_{31} + \alpha_{22}x_{32} + \alpha_0$$
$$y_{22} = \alpha_{11}x_{22} + \alpha_{12}x_{23} + \alpha_{21}x_{32} + \alpha_{22}x_{33} + \alpha_0$$

Slide adapted from William Cohen

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

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

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

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

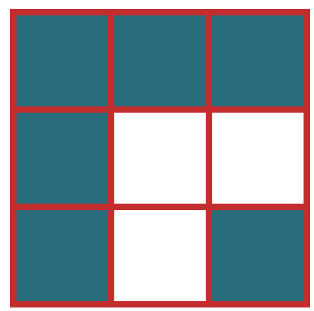
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution



Convolved Image

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

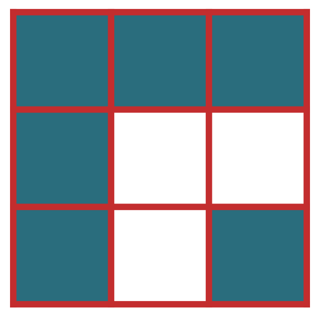
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution



Convolved Image

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

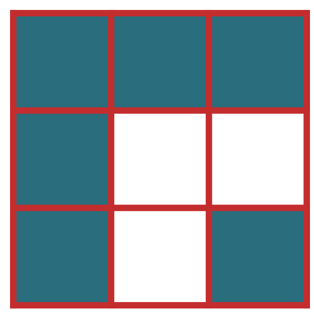
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution

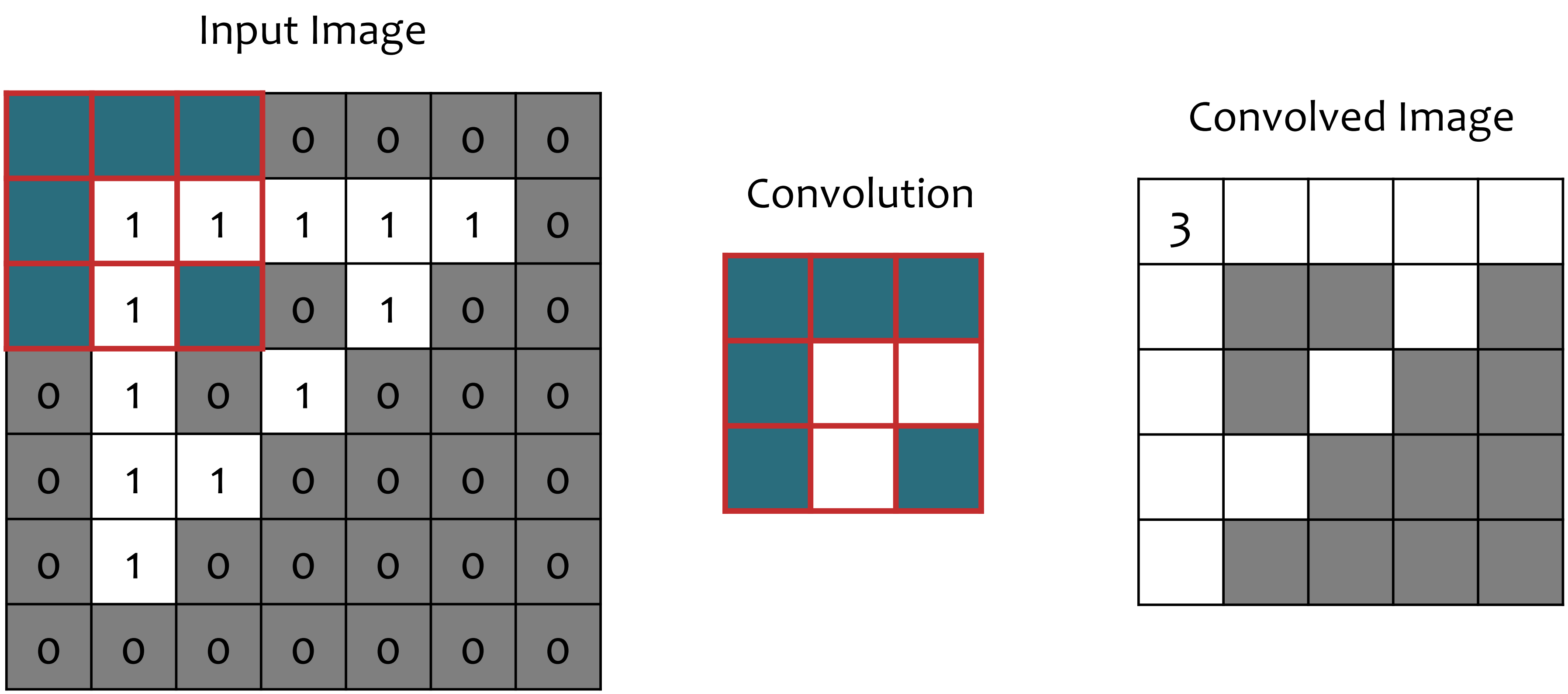


Convolved Image

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

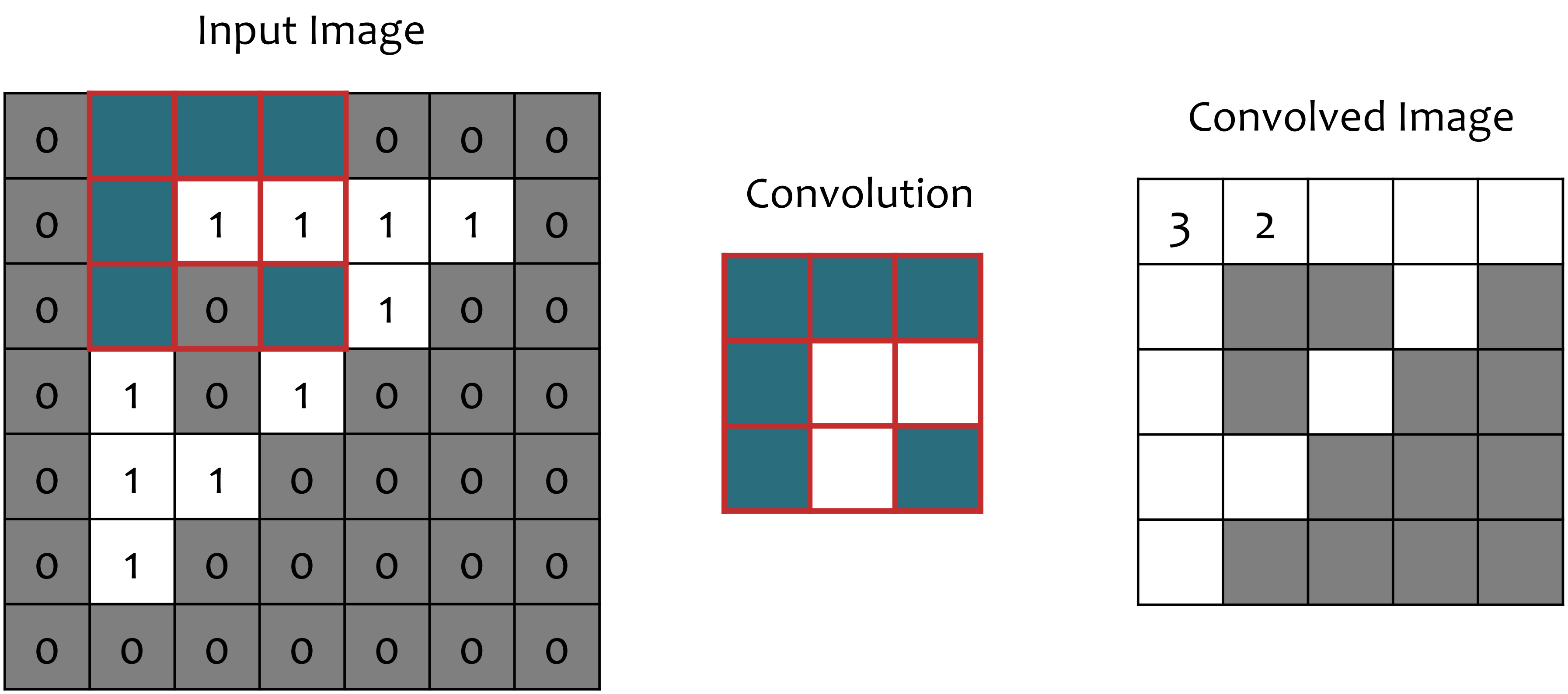
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



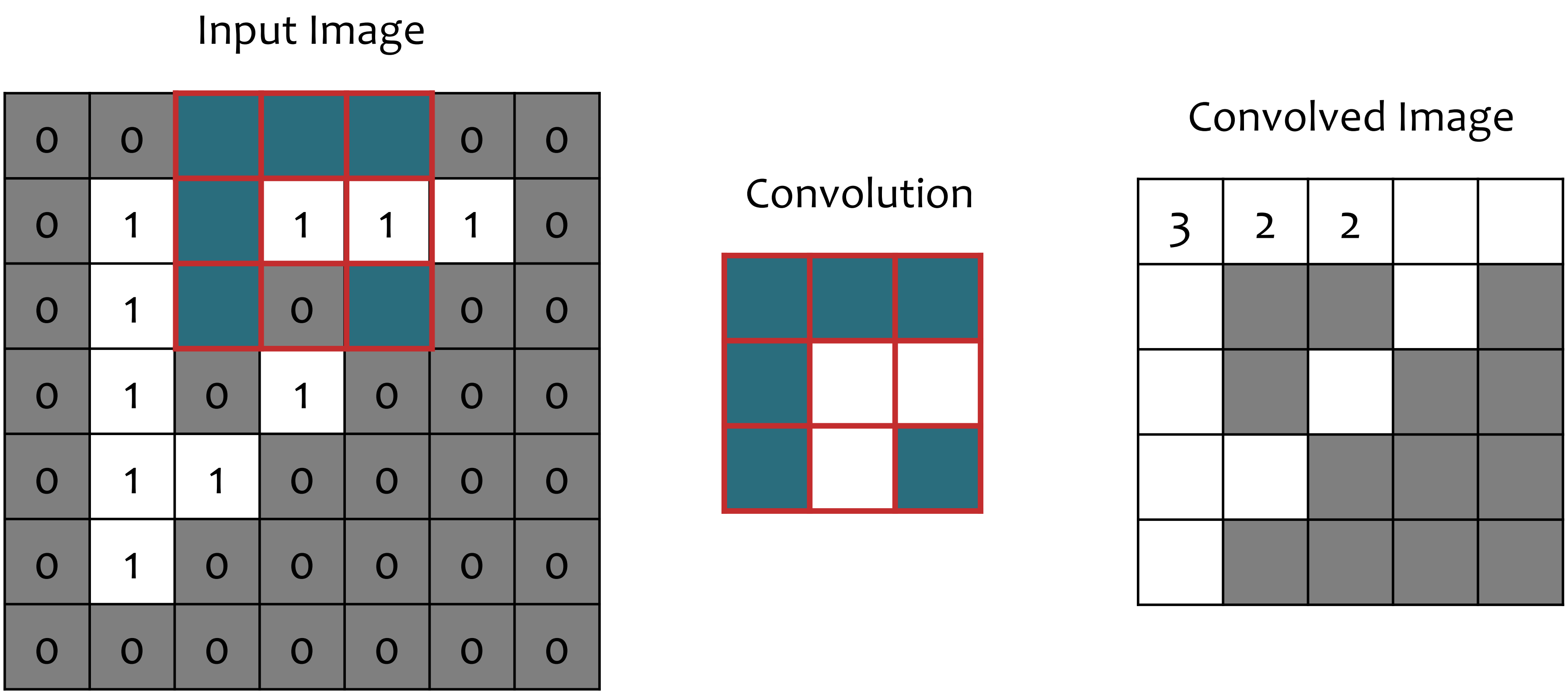
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



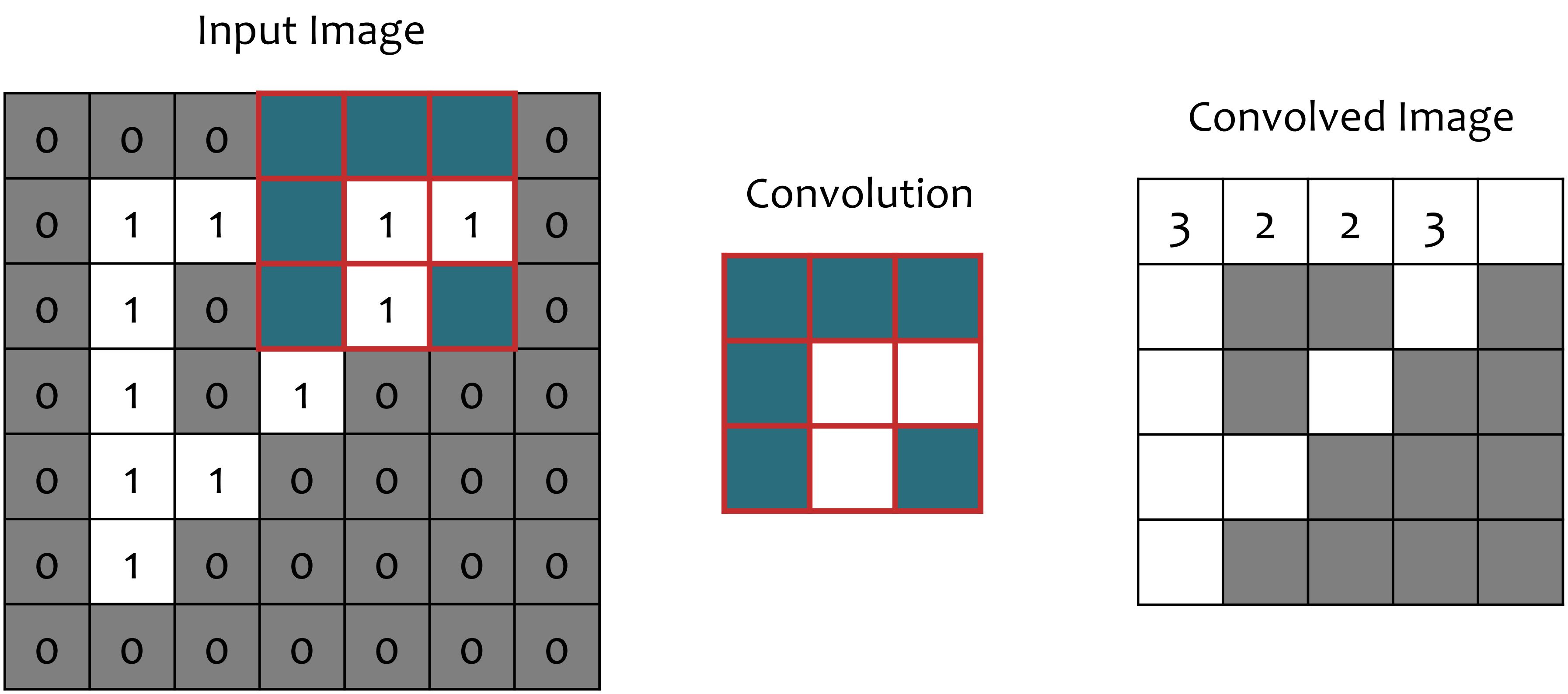
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



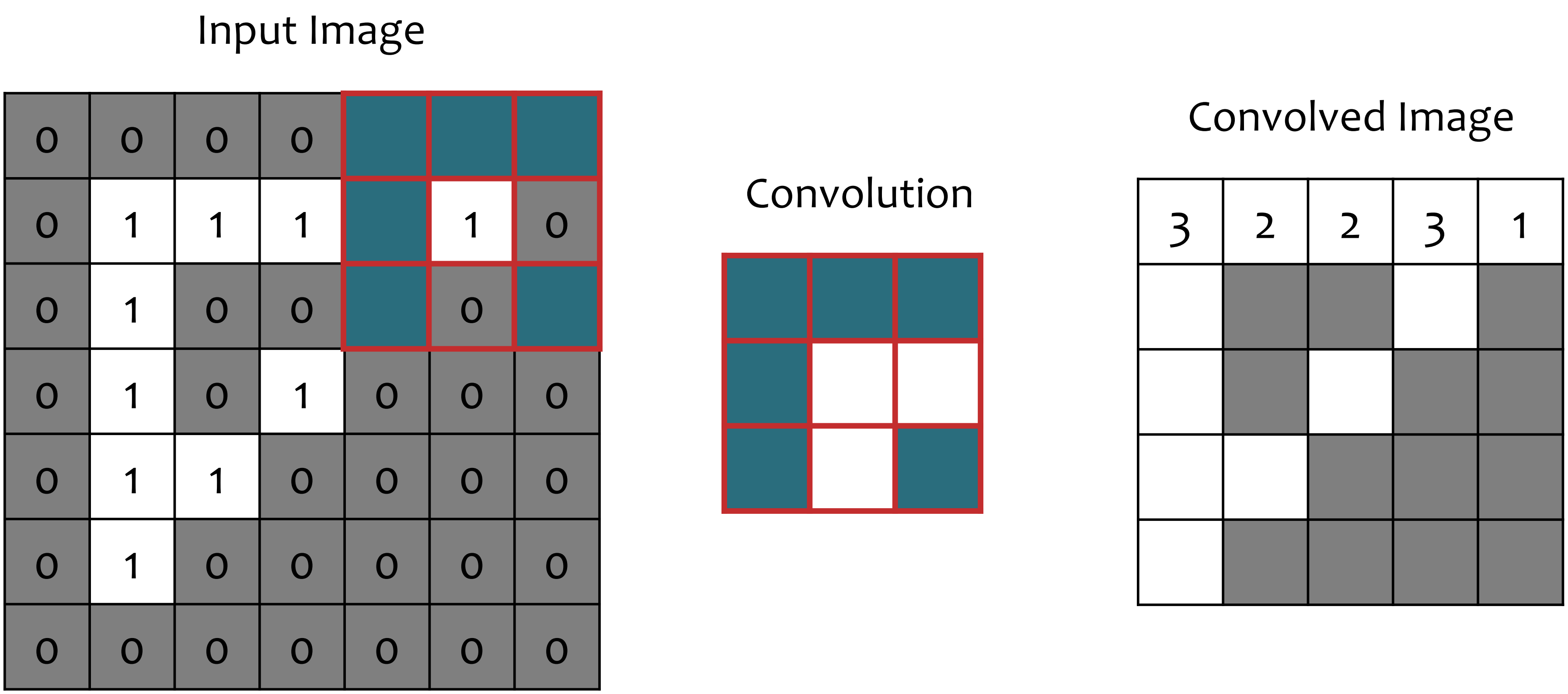
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



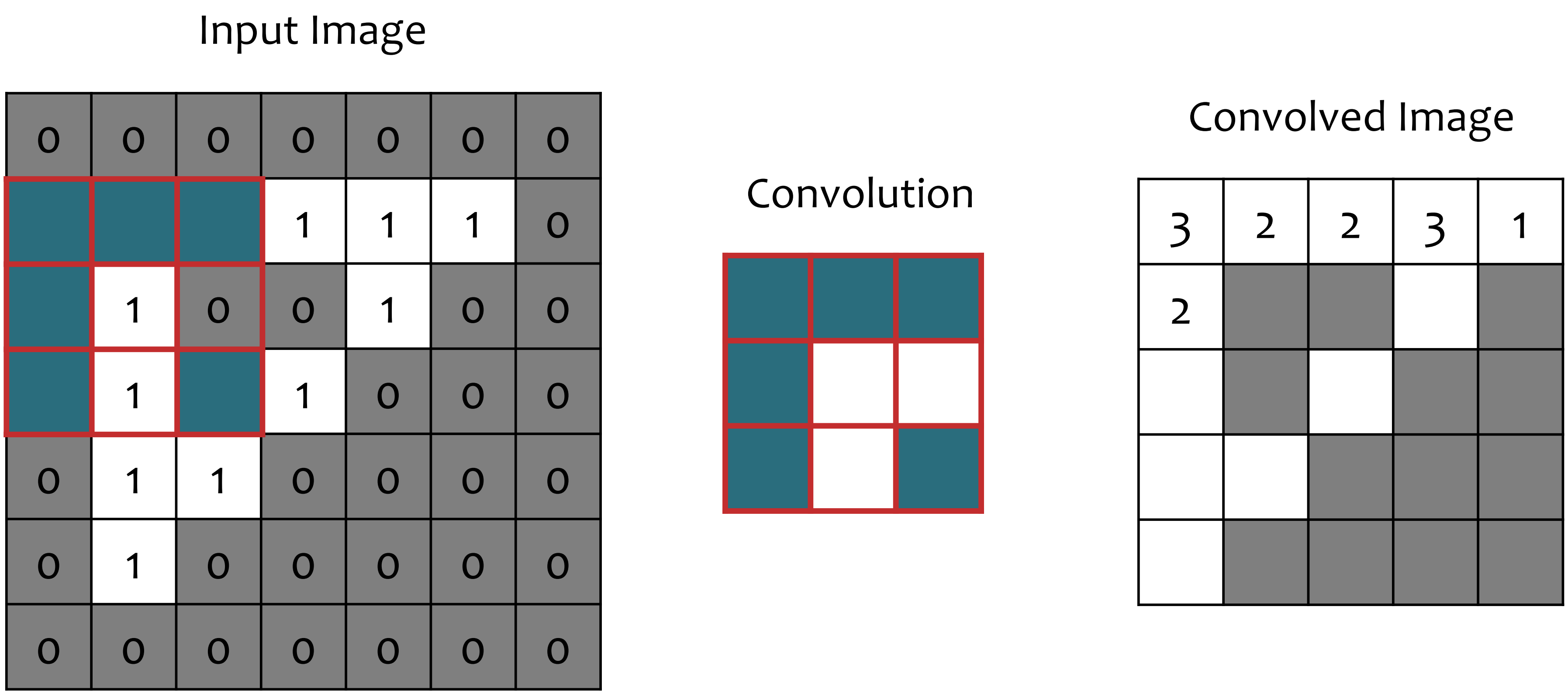
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



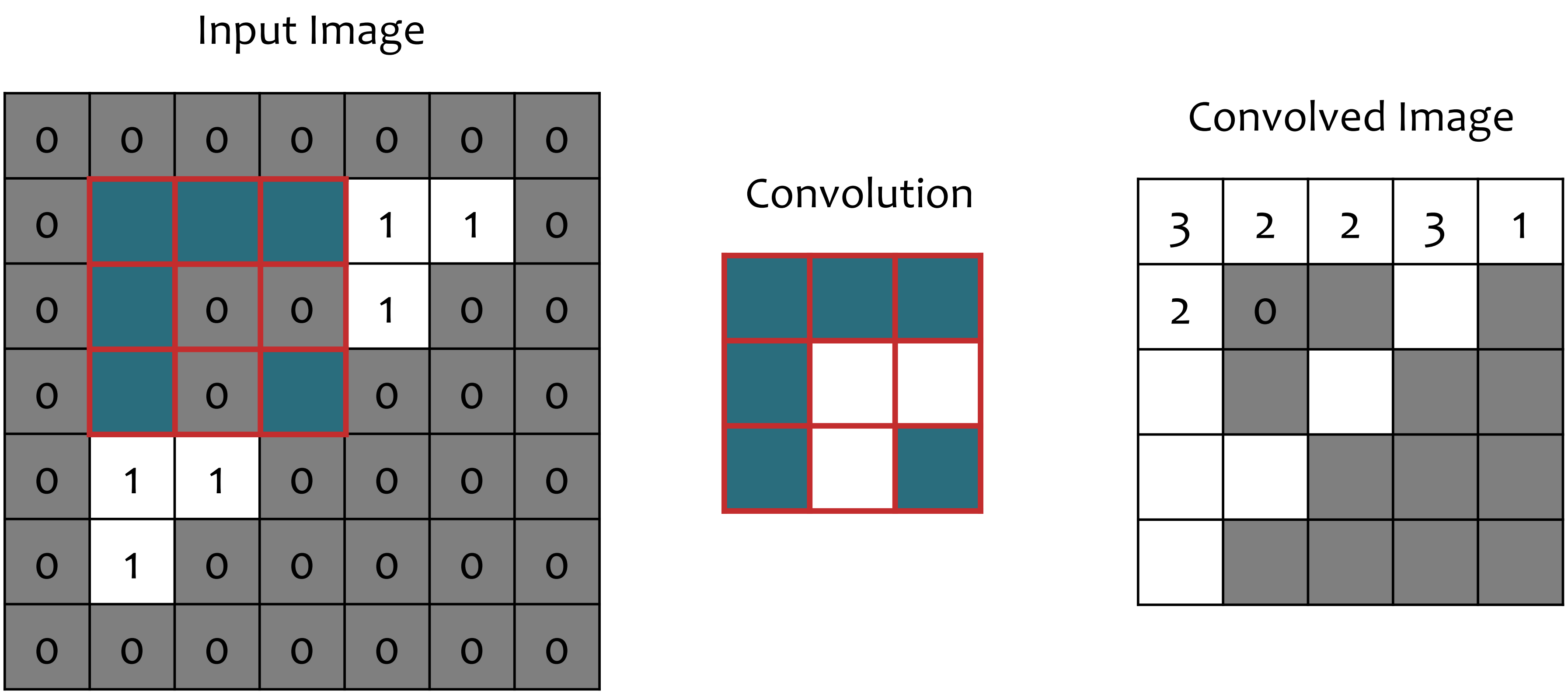
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



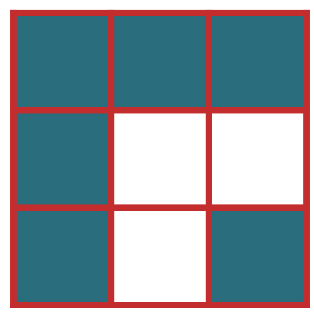
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Convolution



Convolved Image

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

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Identity Convolution

0	0	0
0	1	0
0	0	0

Convolved Image

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

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

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

Blurring Convolution

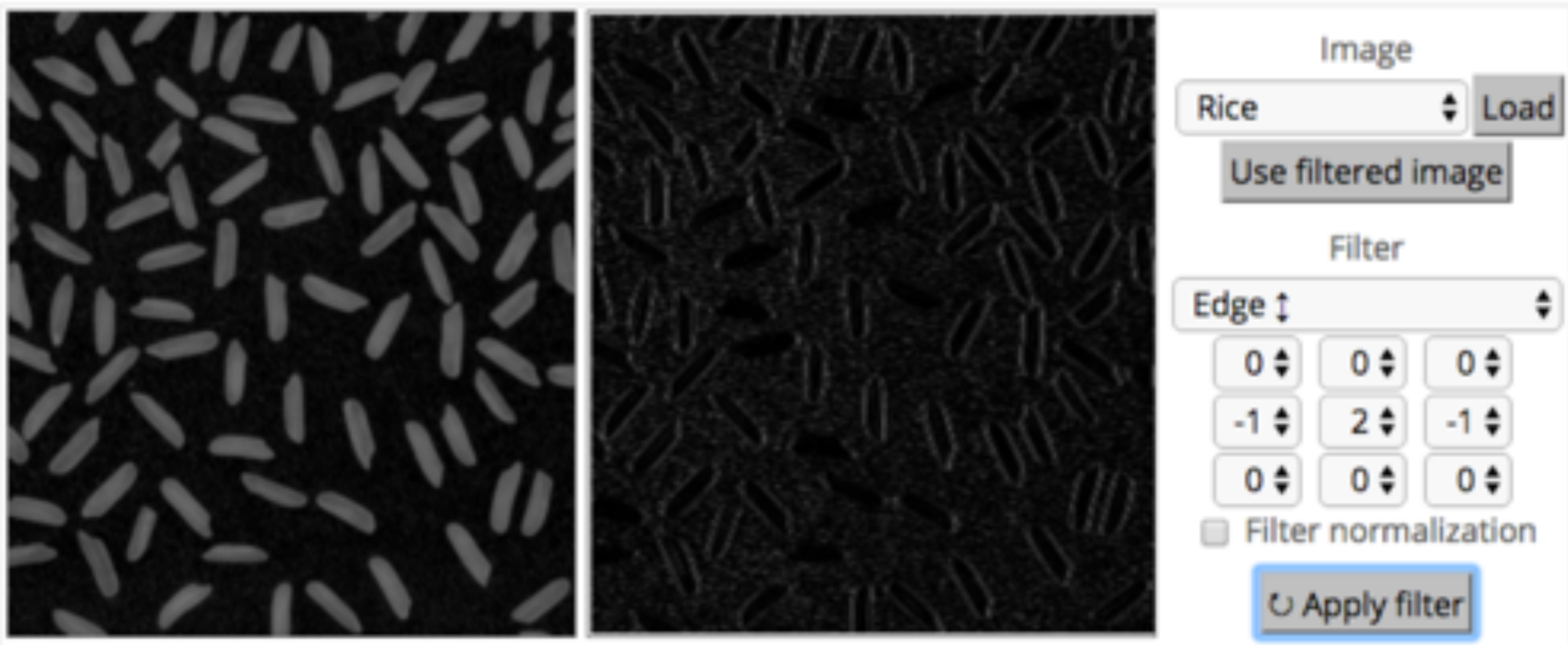
.1	.1	.1
.1	.2	.1
.1	.1	.1

Convolved Image

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

What's a convolution?

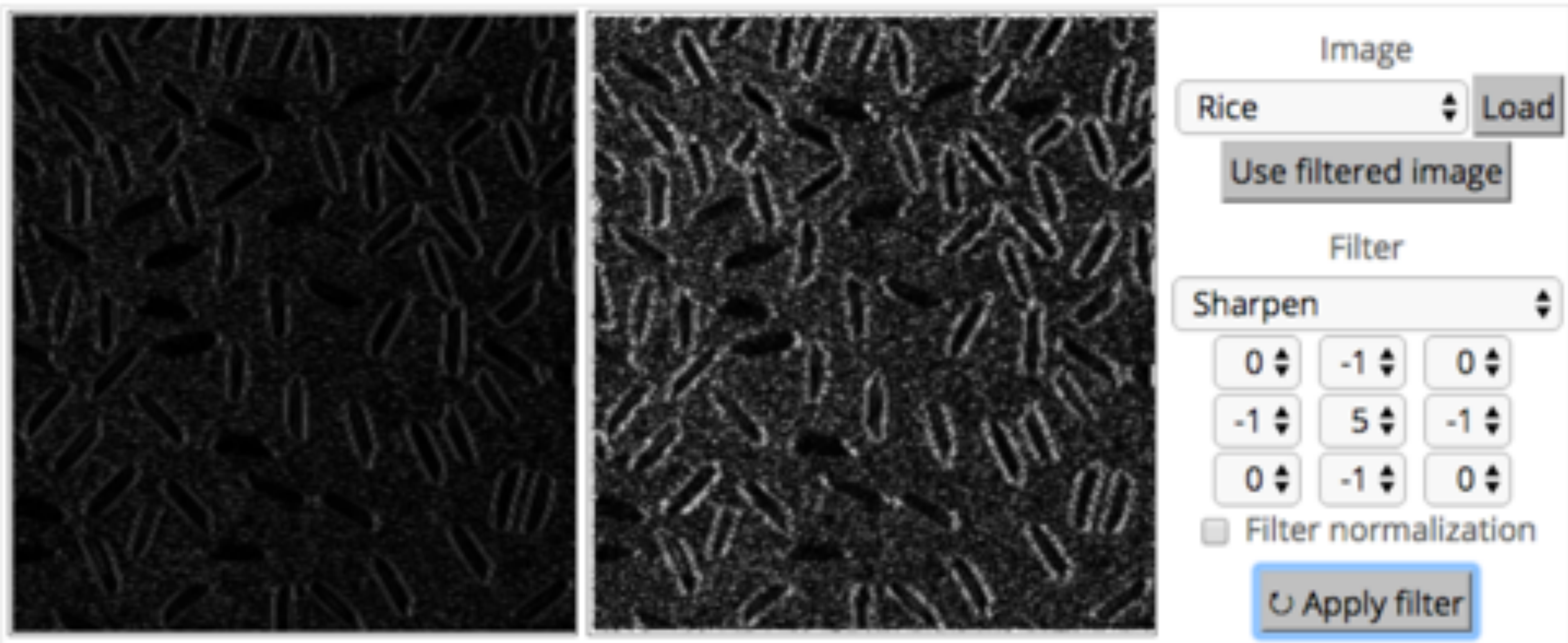
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What's a convolution?

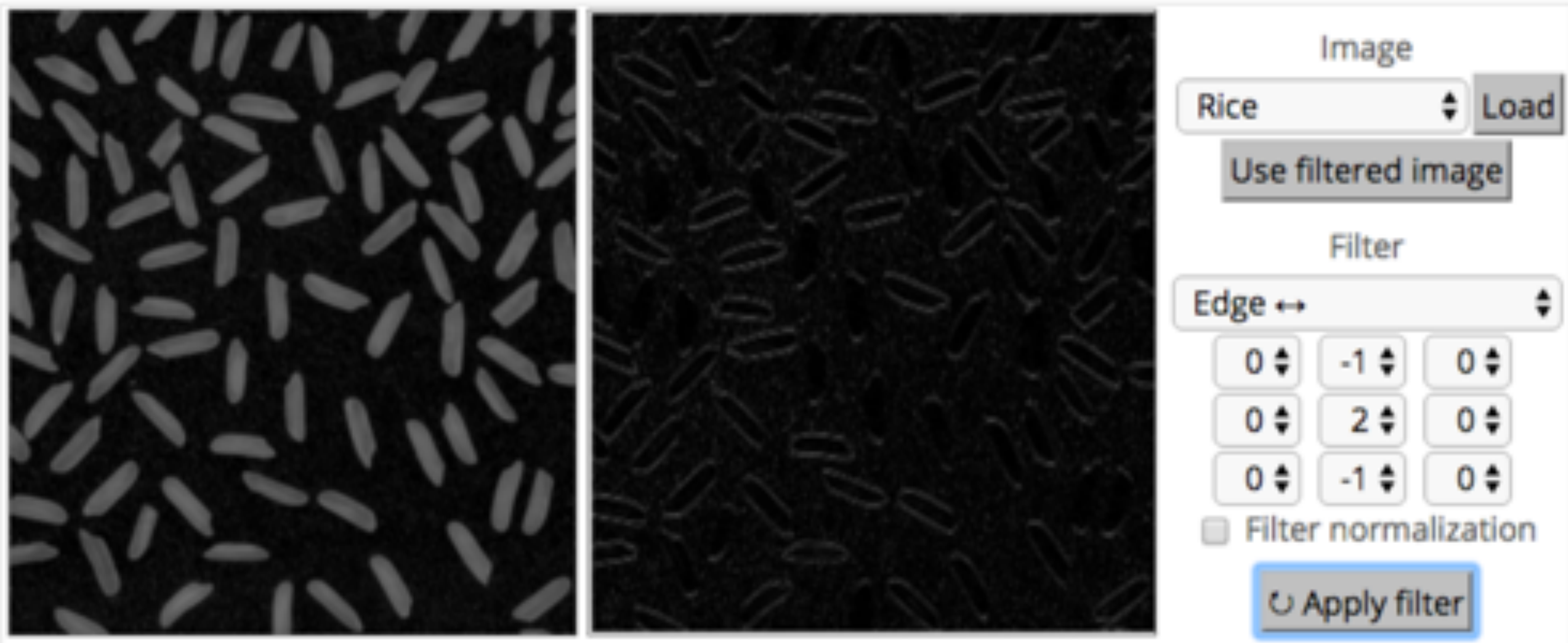
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What's a convolution?

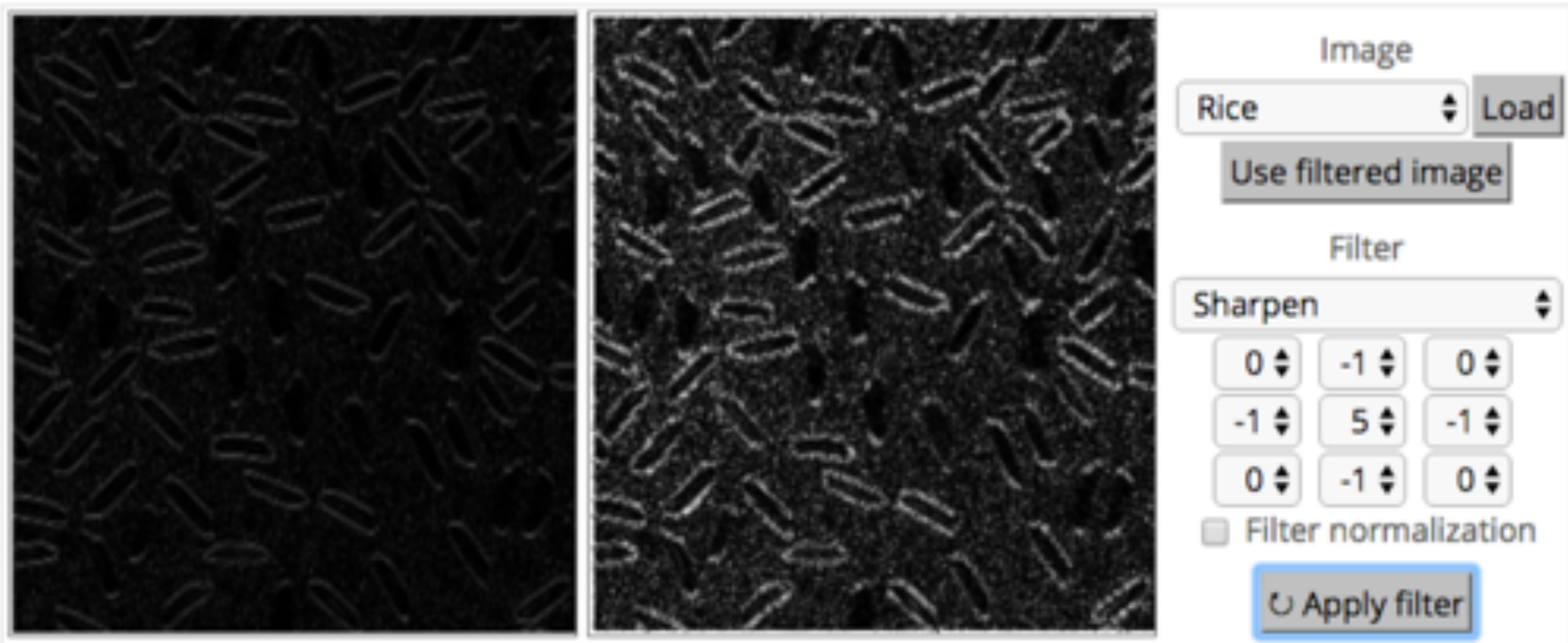
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What's a convolution?

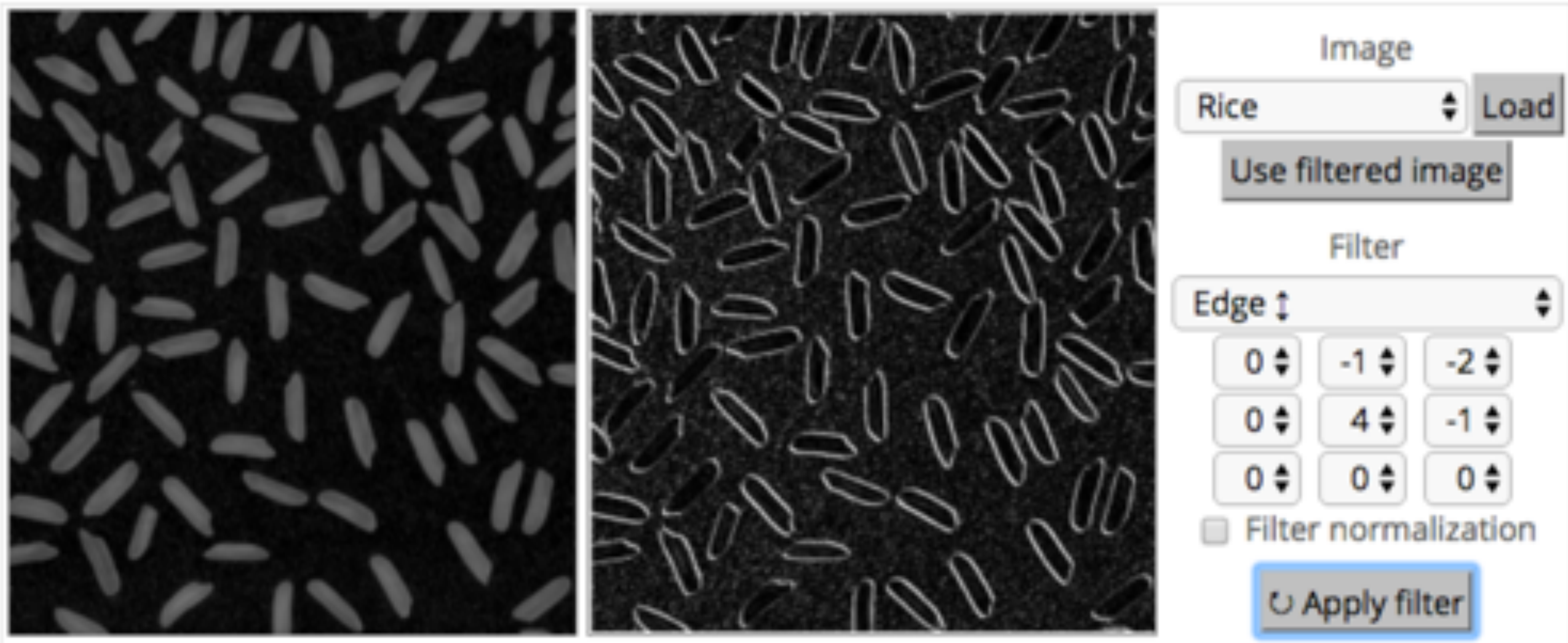
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What's a convolution?

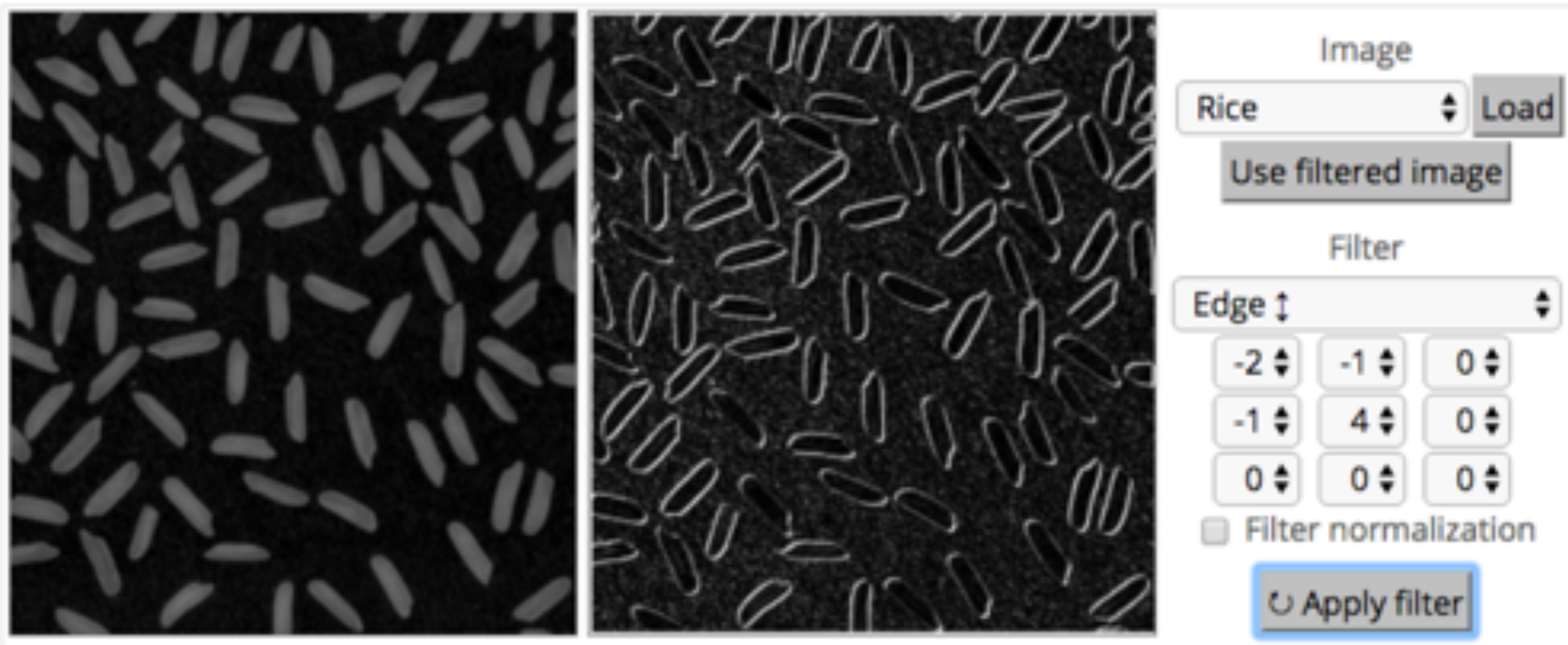
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo

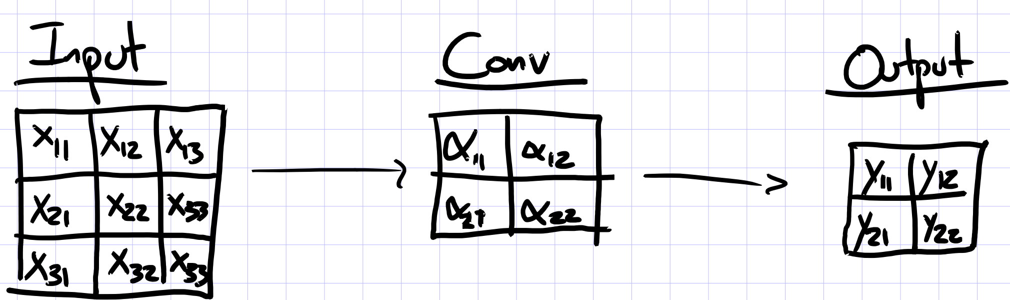


Slide from William Cohen

What's a convolution?

- Basic idea:
  - Pick a 3x3 matrix F of weights
  - Slide this over an image and compute the "inner product" (similarity) of F and the corresponding field of the image, and replace the pixel in the center of the field with the output of the inner product operation
- Key point:
  - Different convolutions extract different types of low-level "features" from an image
  - All that we need to vary to generate these different features is the weights of F

Ex: 1 input channel, 1 output channel

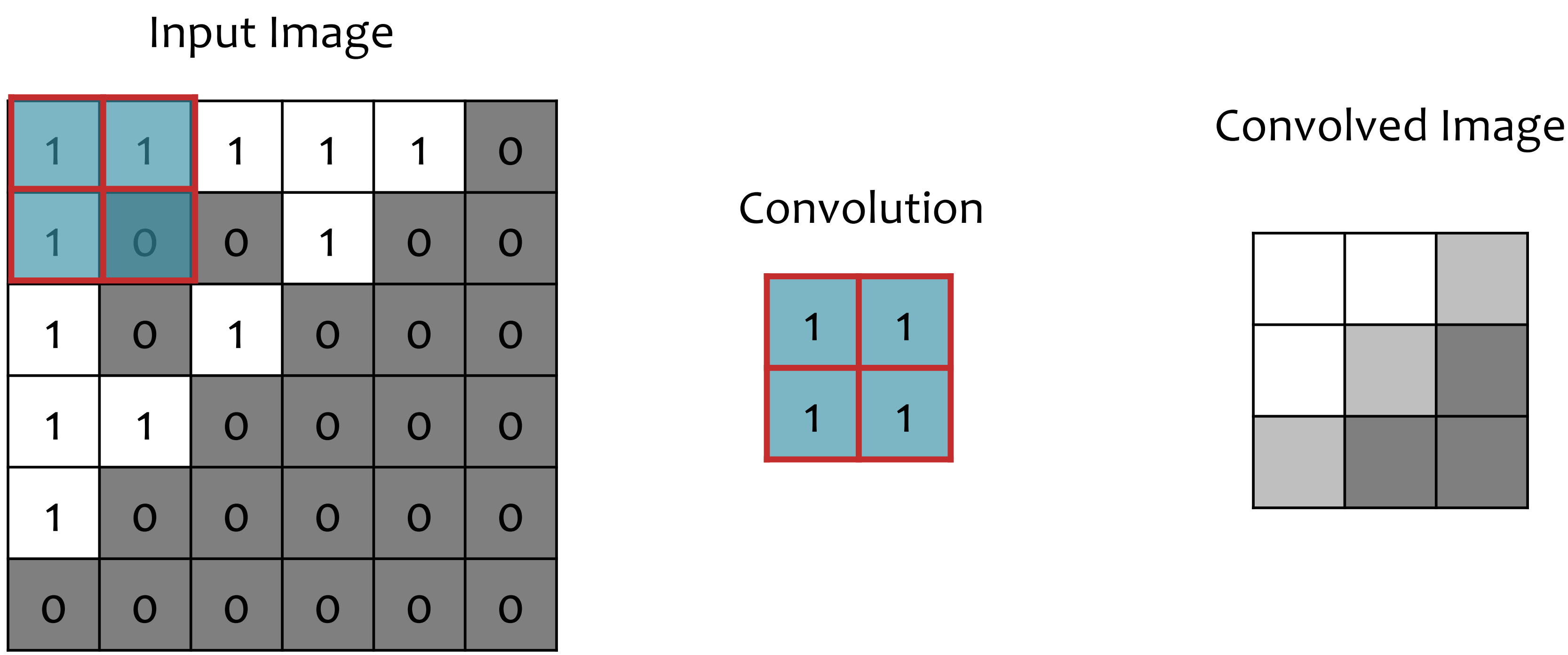


$$y_{11} = \alpha_{11} x_{11} + \alpha_{12} x_{12} + \alpha_{21} x_{21} + \alpha_{22} x_{22} + \alpha_0$$
$$y_{12} = \alpha_{11} x_{12} + \alpha_{12} x_{13} + \alpha_{21} x_{22} + \alpha_{22} x_{23} + \alpha_0$$
$$y_{21} = \alpha_{11} x_{21} + \alpha_{12} x_{22} + \alpha_{21} x_{31} + \alpha_{22} x_{32} + \alpha_0$$
$$y_{22} = \alpha_{11} x_{22} + \alpha_{12} x_{23} + \alpha_{21} x_{32} + \alpha_{22} x_{33} + \alpha_0$$

Slide adapted from William Cohen

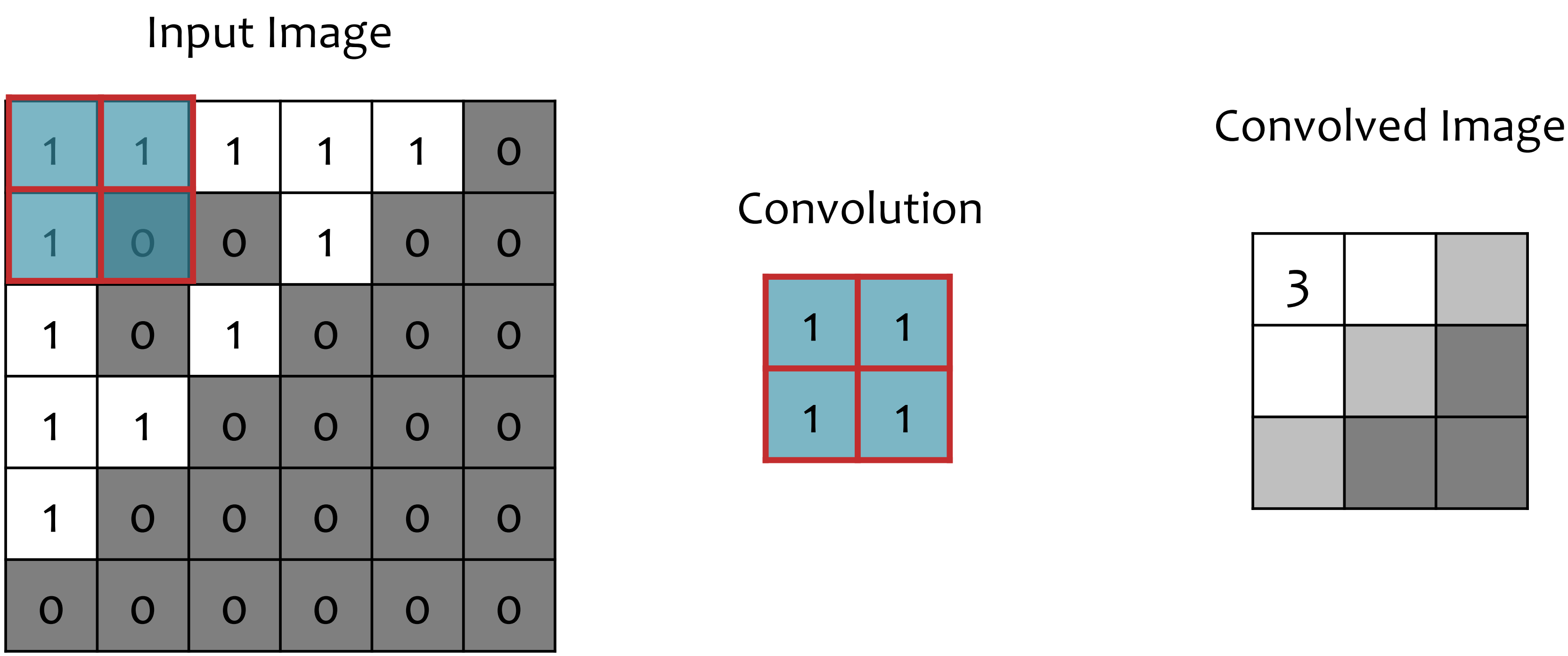
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



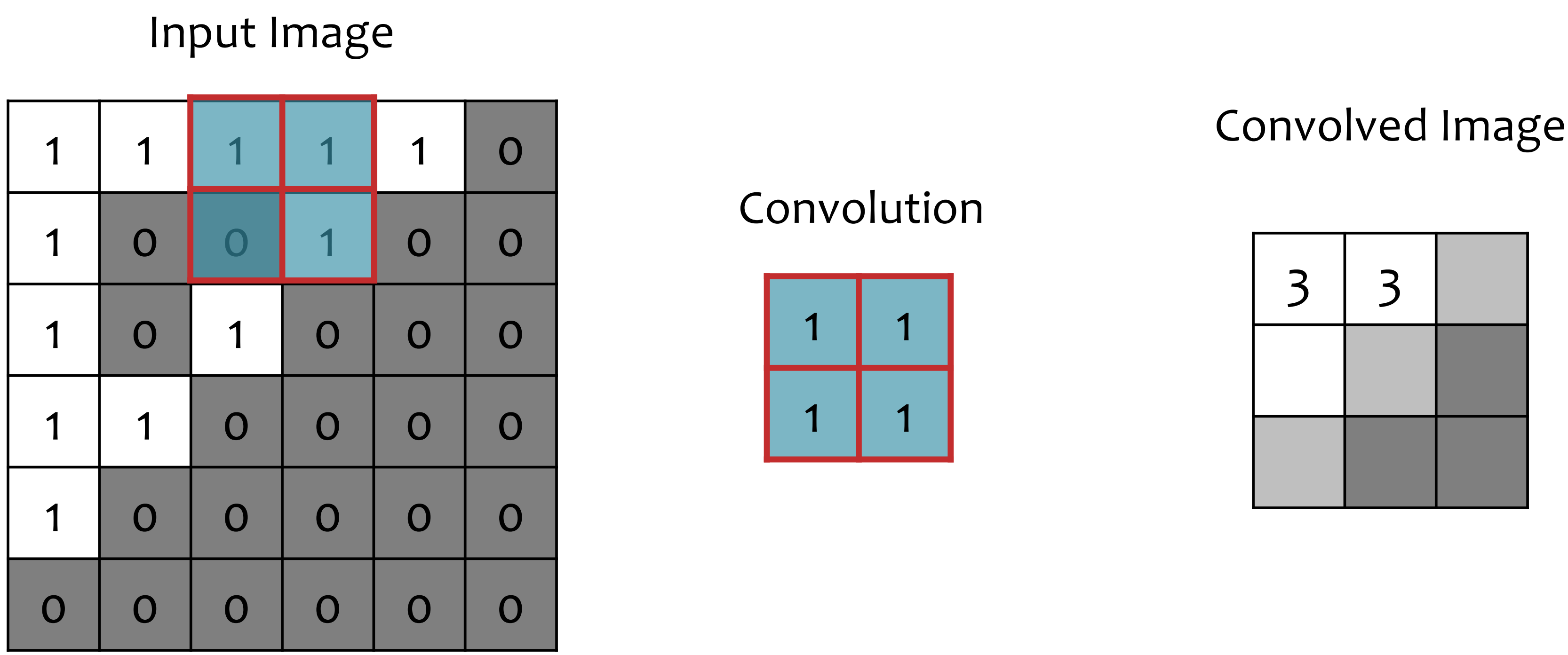
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



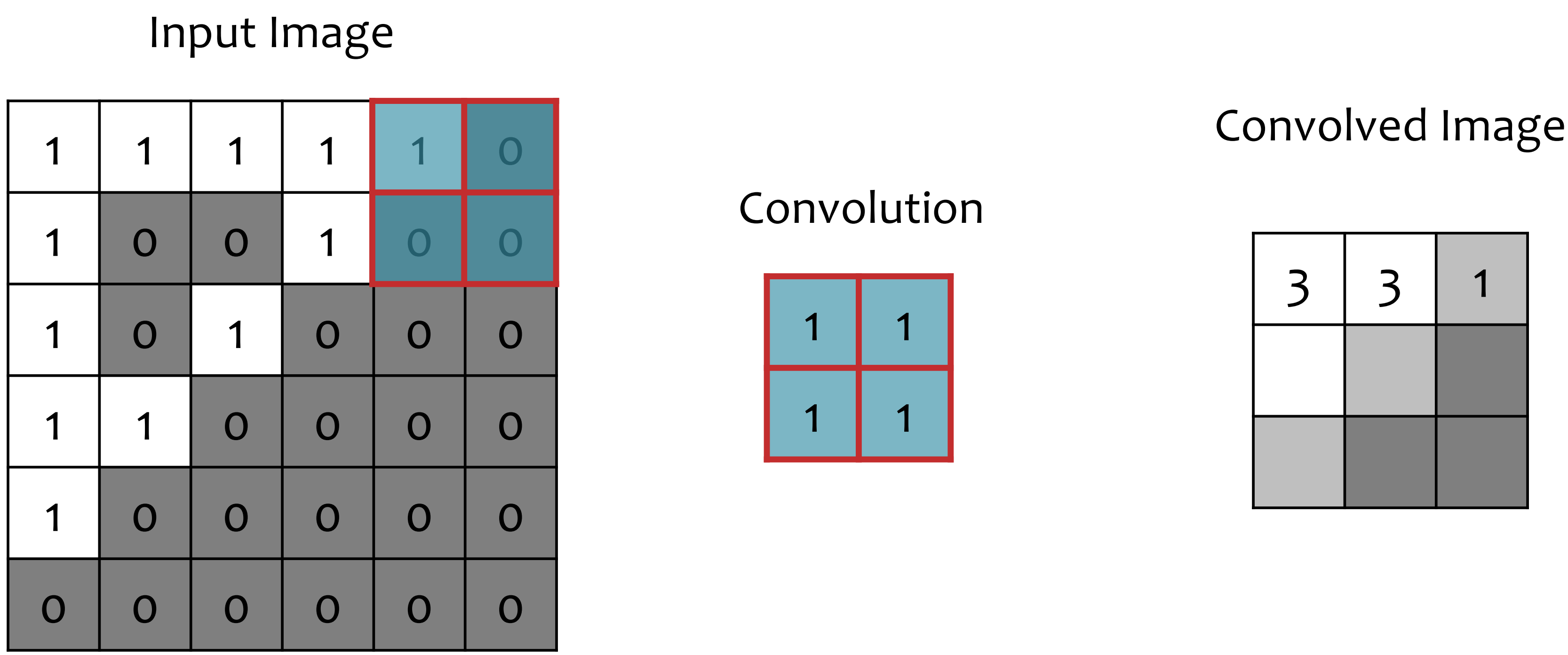
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



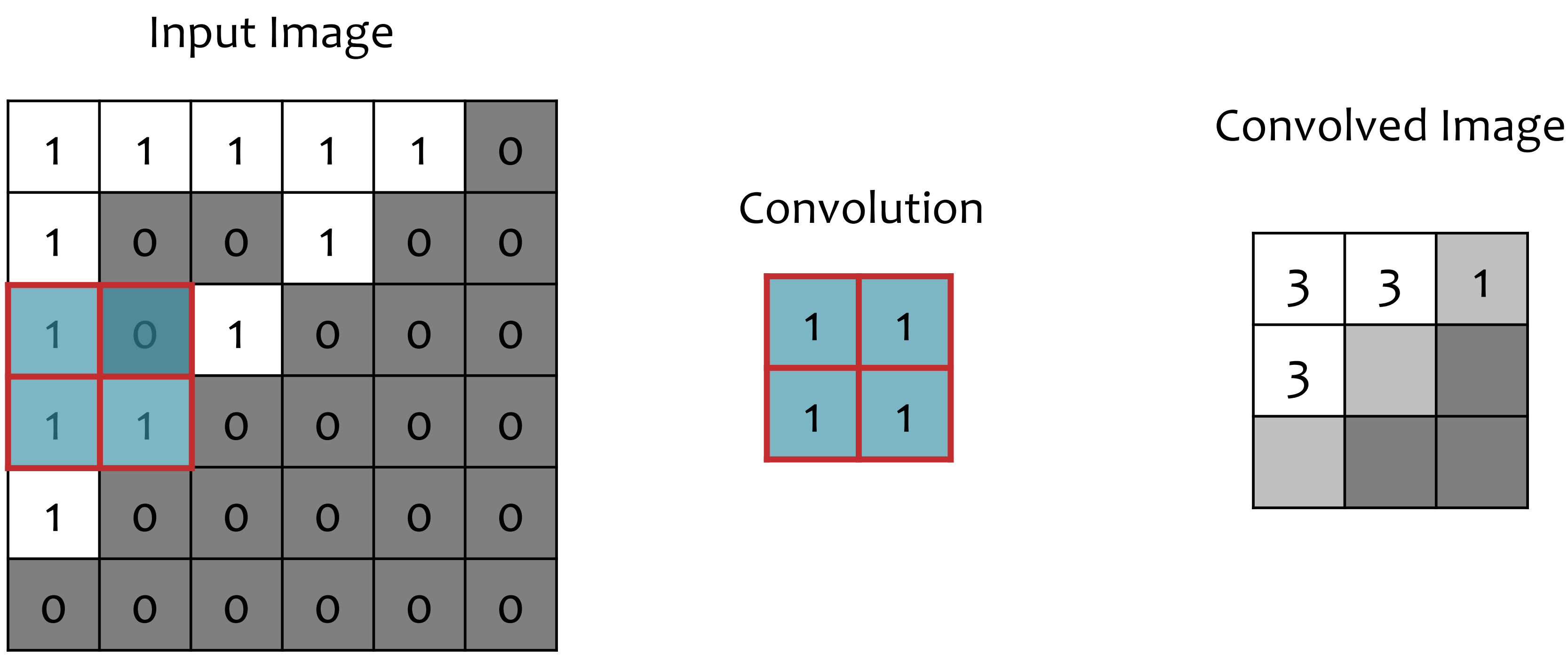
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



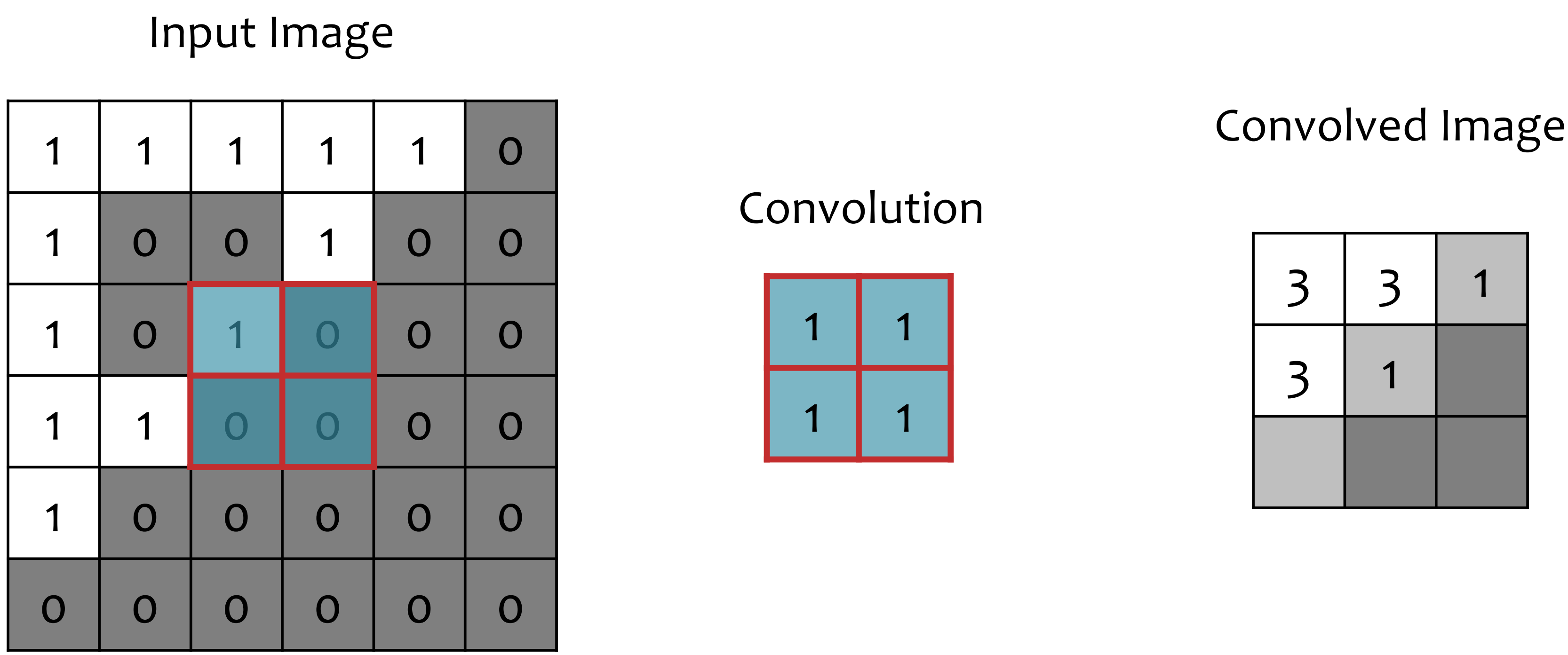
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



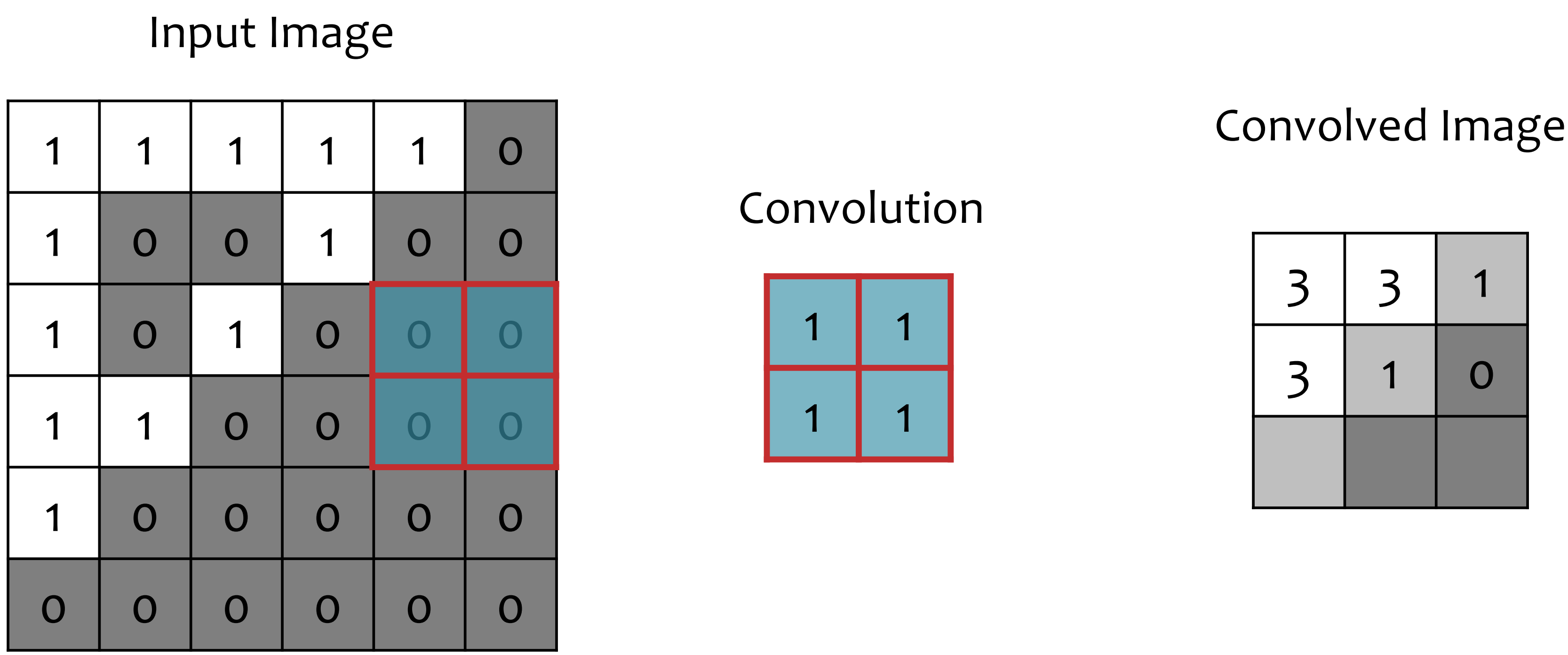
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



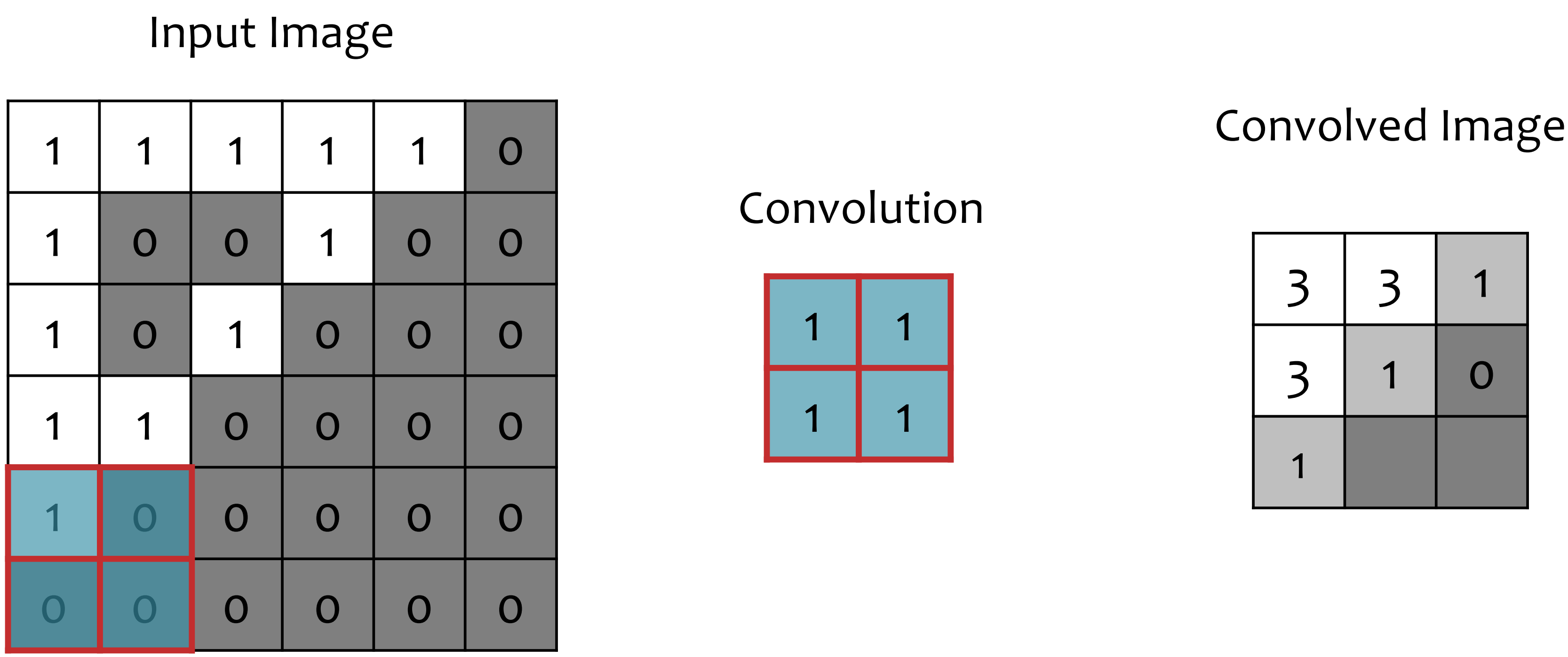
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



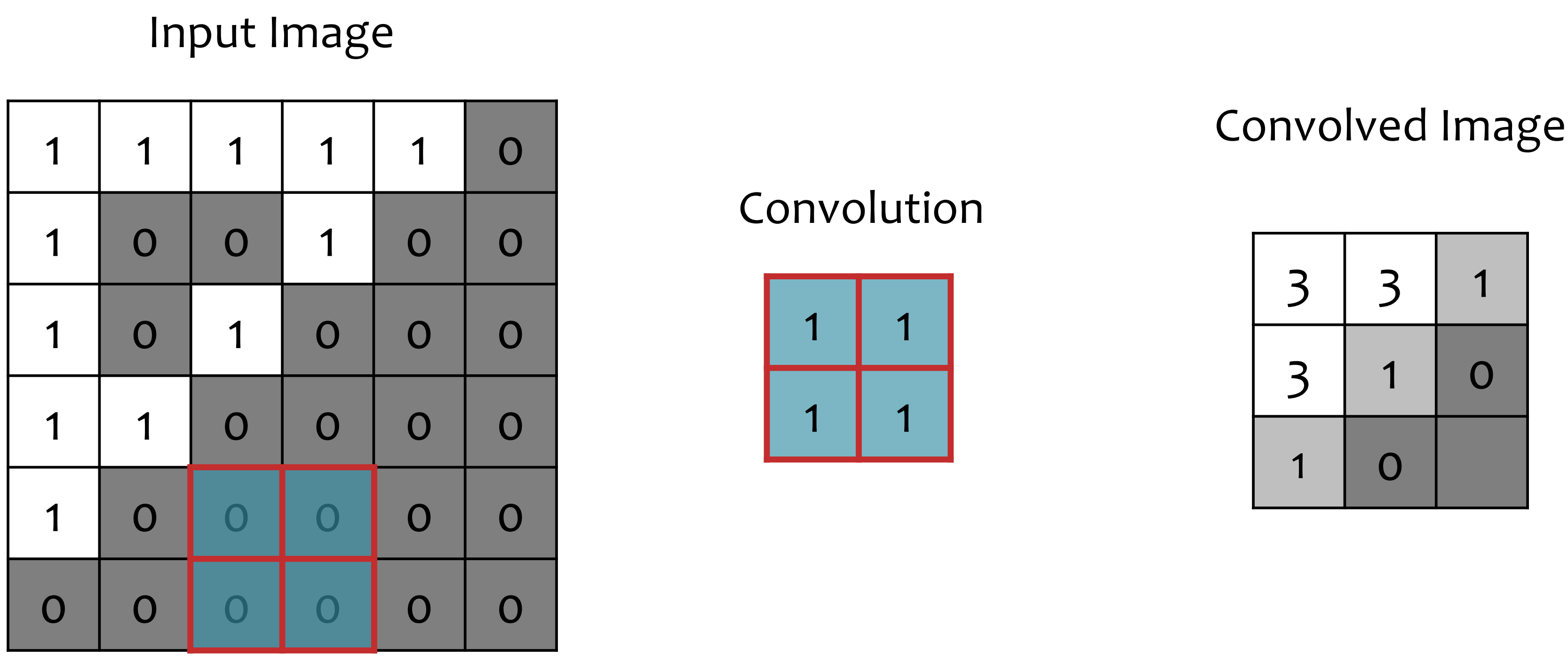
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



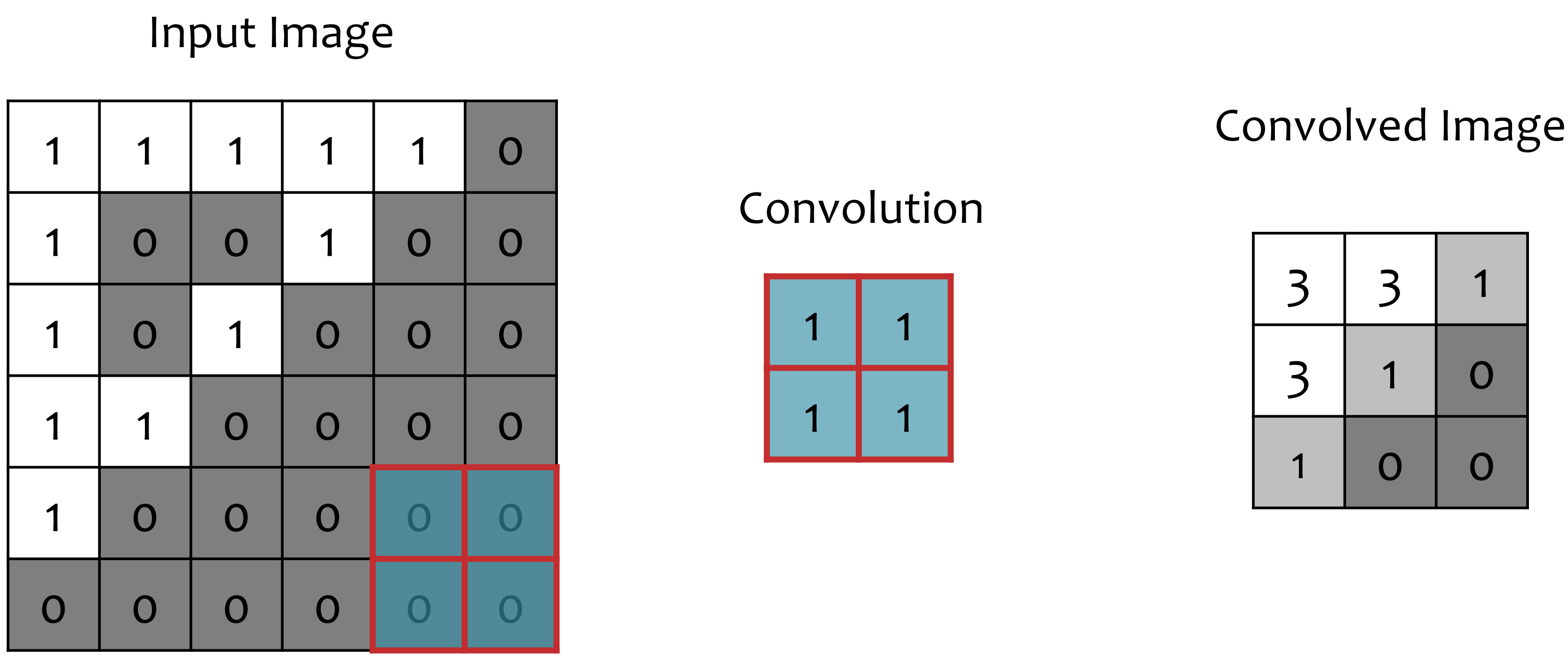
Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



Downsampling

- Suppose we use a convolution with stride 2
- Only 9 patches visited in input, so only 9 pixels in output



CONVOLUTIONAL NEURAL NETS

A Recipe for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of these:

– Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train with SGD:

(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

A Recipe for Machine Learning

- Convolutional Neural Networks (CNNs) provide another form of **decision function**
- Let's see what they look like…

1.

2. Choose each of these:
   - Decision function

     $$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   - Loss function

     $$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

4. Train with SGD:
   (take small steps opposite the gradient)

   $$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Convolutional Neural Network (CNN)

- Typical layers include:
  - Convolutional layer
  - Max-pooling layer
  - Fully-connected (Linear) layer
  - ReLU layer (or some other nonlinear activation function)
  - Softmax
- These can be arranged into arbitrarily deep topologies

Architecture #1: LeNet-5

PROC. OF THE IEEE, NOVEMBER 1998 7

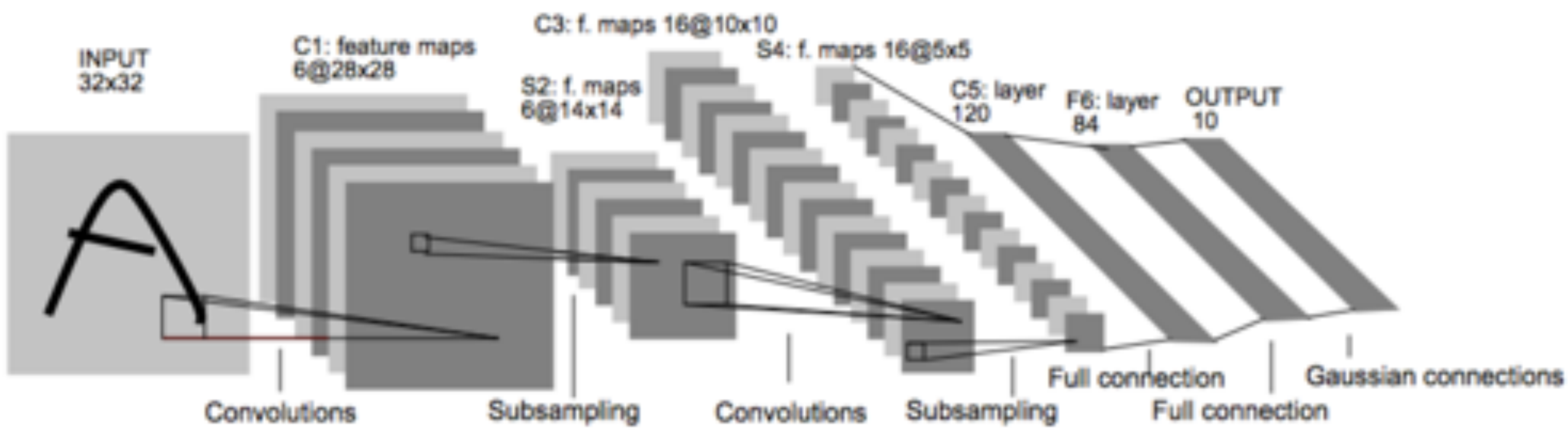


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Convolutional Layer

CNN key idea:
Treat convolution matrix as parameters and learn them!

Input Image

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

Learned Convolution

θ_{11}	θ_{12}	θ_{13}
θ_{21}	θ_{22}	θ_{23}
θ_{31}	θ_{32}	θ_{33}

Convolved Image

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Downsampling by Averaging

- Downsampling by averaging **used to be** a common approach
- This is a special case of convolution where the weights are fixed to a uniform distribution
- The example below uses a stride of 2

Input Image

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

Convolution

1/4	1/4
1/4	1/4

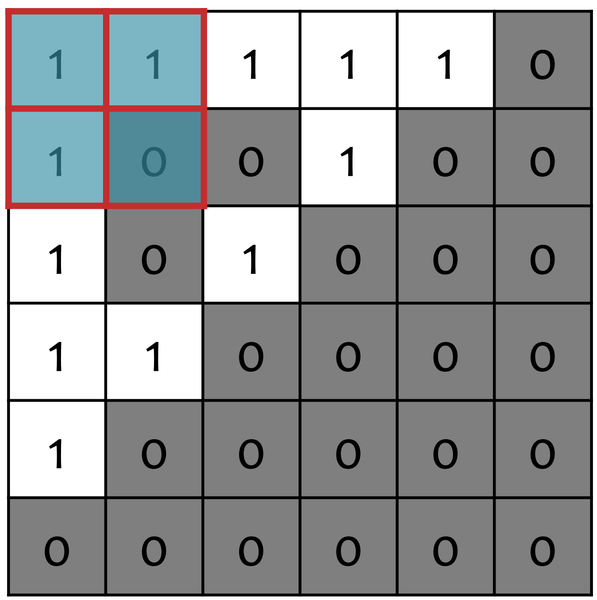
Convolved Image

3/4	3/4	1/4
3/4	1/4	0
1/4	0	0

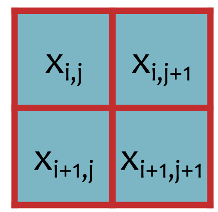
Max-Pooling

- Max-pooling is another (common) form of downsampling
- Instead of averaging, we take the max value within the same range as the equivalently-sized convolution
- The example below uses a stride of 2

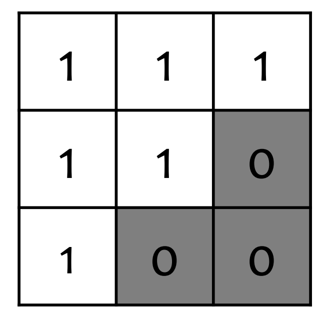
Input Image



Max-pooling



Max-Pooled Image



$$y_{ij} = \max(x_{ij}, x_{i,j+1}, x_{i+1,j}, x_{i+1,j+1})$$

TRAINING CNNS

A Recipe for Machine Learning

Background

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of these:

– Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train with SGD:

(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

A Recipe for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$$

2. Choose each of t[…]

– Decision functio[n]

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$N \quad \boldsymbol{y}_i)$$

opp[o]site the gradient)

$$\boldsymbol{\theta}^{(t} \quad {}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

- Q: Now that we have the CNN as a decision function, how do we compute the gradient?
- A: Backpropagation of course!

SGD for CNNs

SGD for CNNs

Ex: Architecture: Given $\vec{x}, y^*$

$$J = \ell(y, y^*)$$
$$y = \text{softmax}(z^{(5)})$$
$$z^{(5)} = \text{linear}(z^{(4)}, W)$$
$$z^{(4)} = \text{relu}(z^{(3)})$$
$$z^{(3)} = \text{conv}(z^{(2)}, \beta)$$
$$z^{(2)} = \text{max-pool}(z^{(1)})$$
$$z^{(1)} = \text{conv}(\vec{x}, \alpha)$$

Parameters $\vec{\Theta} = [\alpha, \beta, W]$

SGD:

1. Init $\vec{\Theta}$
2. While not converged:

   Sample $i \in \{1, \ldots, N\}$

   Forward: $y = h_\Theta(\vec{x}^{(i)})$, $J_i(\Theta) = \ell(y, y^*)$

   Backward: $\nabla_{\vec{\Theta}} J_i(\Theta) = \ldots$

   Update: $\vec{\Theta} \leftarrow \vec{\Theta} - \lambda \nabla_{\vec{\Theta}} J_i(\Theta)$

LAYERS OF A CNN

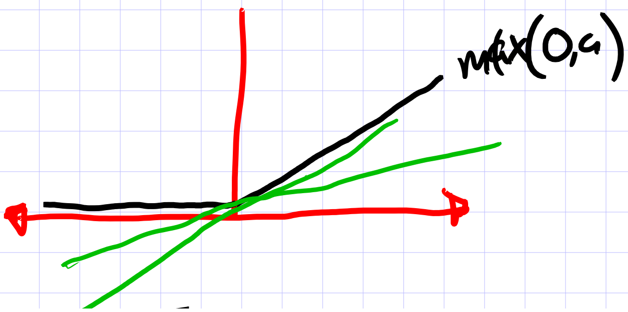
ReLU Layer

ReLU Layer Input: $\vec{x} \in \mathbb{R}^K$ Output: $\vec{y} \in \mathbb{R}^K$

Forward:

$$\vec{y} = \sigma(\vec{x})$$ ← element-wise

$$\sigma(a) = \max(0, a)$$



Backward:

$$\frac{dJ}{dx_i} = \frac{dJ}{dy_i} \frac{dy_i}{dx_i}$$

where $\frac{dy_i}{dx_i} = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$ ← subderivative

Softmax Layer

Softmax Layer

Input: $\vec{x} \in \mathbb{R}^K$ Output: $\vec{y} \in \mathbb{R}^K$

Forward:

$$y_i = \frac{\exp(x_i)}{\sum_{k=1}^{K} \exp(x_k)}$$

Backward:

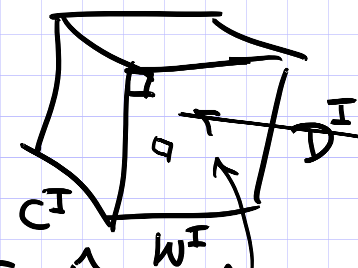
$$\frac{dJ}{dx_j} = \sum_{i=1}^{K} \frac{dJ}{dy_i} \frac{dy_i}{dx_j}$$

where $$\frac{dy_i}{dx_j} = \begin{cases} y_i(1-y_i) & \text{if } i=j \\ -y_i y_j & \text{otherwise} \end{cases}$$

Fully-Connected Layer

Fully Connected Layer (w/ tensor input)

- Suppose input is a 3D Tensor: $X =$



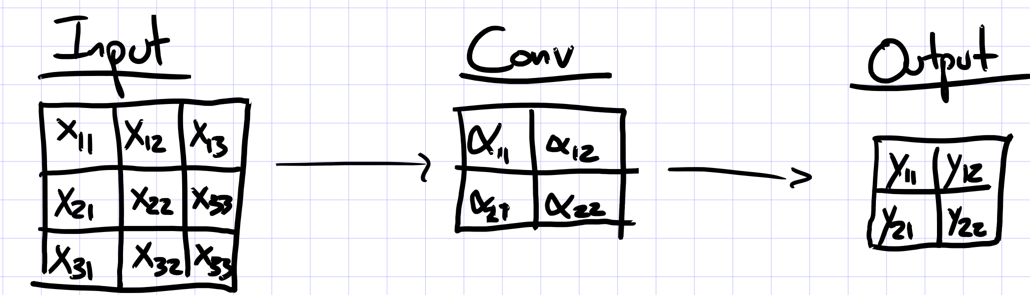
- stretch out into a long vector: $\hat{X} = [\hat{x}_1, \ldots, \hat{x}_{(C\times H\times W)}]$
- then standard linear layer:

$$y = \alpha^T \hat{x} + \alpha_0 \quad \text{where } \alpha \in \mathbb{R}^{A\times B}$$

$$|\hat{x}| = A, \quad |y| = B$$

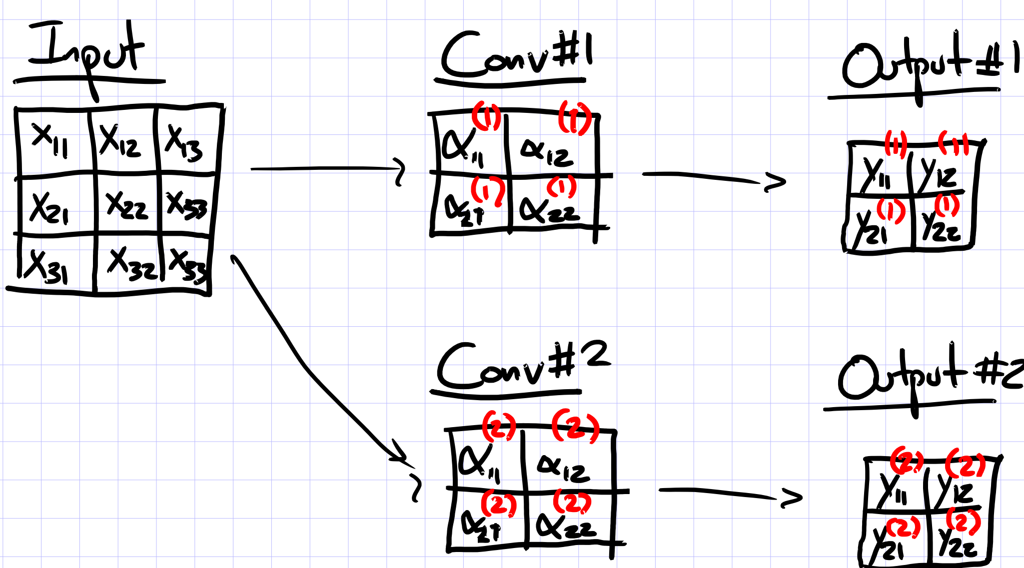
Convolutional Layer

Ex: 1 input channel, 1 output channel



$$y_{11} = \alpha_{11}x_{11} + \alpha_{12}x_{12} + \alpha_{21}x_{21} + \alpha_{22}x_{22} + \alpha_0$$
$$y_{12} = \alpha_{11}x_{12} + \alpha_{12}x_{13} + \alpha_{21}x_{22} + \alpha_{22}x_{23} + \alpha_0$$
$$y_{21} = \alpha_{11}x_{21} + \alpha_{12}x_{22} + \alpha_{21}x_{31} + \alpha_{22}x_{32} + \alpha_0$$
$$y_{22} = \alpha_{11}x_{22} + \alpha_{12}x_{23} + \alpha_{21}x_{32} + \alpha_{22}x_{33} + \alpha_0$$

Ex: 1 input channel, 2 output channels

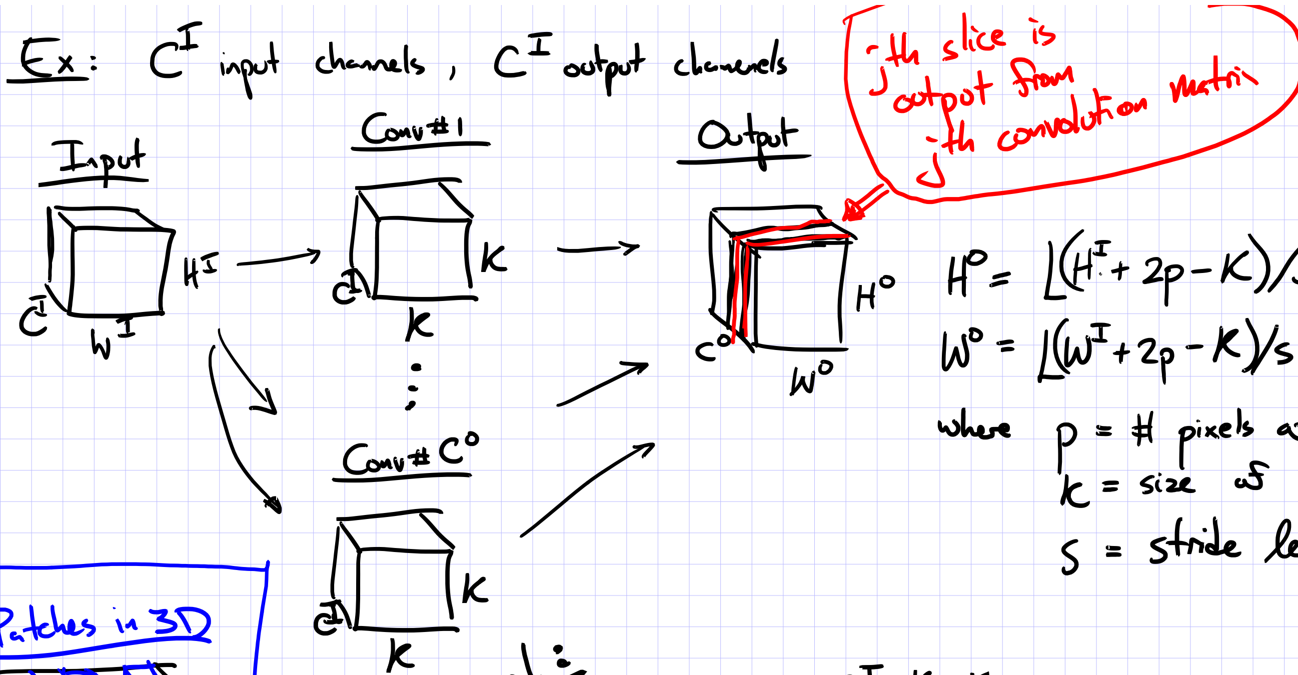


$$y_{11}^{(1)} = \alpha_{11}^{(1)}x_{11} + \alpha_{12}^{(1)}x_{12} + \alpha_{21}^{(1)}x_{21} + \alpha_{22}^{(1)}x_{22} + \alpha_0^{(1)}$$
$$y_{12}^{(1)} = \cdots$$
$$y_{21}^{(1)} = \cdots$$
$$y_{22}^{(1)} = \alpha_{11}^{(1)}x_{22} + \alpha_{12}^{(1)}x_{23} + \alpha_{21}^{(1)}x_{32} + \alpha_{22}^{(1)}x_{33} + \alpha_0^{(1)}$$

$$y_{11}^{(2)} = \alpha_{11}^{(2)}x_{11} + \alpha_{12}^{(2)}x_{12} + \alpha_{21}^{(2)}x_{21} + \alpha_{22}^{(2)}x_{22} + \alpha_0^{(2)}$$
$$y_{12}^{(2)} = \cdots$$
$$y_{21}^{(2)} = \cdots$$
$$y_{22}^{(2)} = \alpha_{11}^{(2)}x_{22} + \alpha_{12}^{(2)}x_{23} + \alpha_{21}^{(2)}x_{32} + \alpha_{22}^{(2)}x_{33} + \alpha_0^{(2)}$$

Convolutional Layer

Ex: C^I input channels, C^I output channels



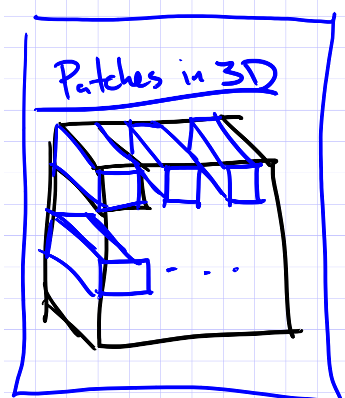
$$H^O = \lfloor (H^I + 2p - K)/s + 1 \rfloor$$

$$W^O = \lfloor (W^I + 2p - K)/s + 1 \rfloor$$

where p = # pixels of padding on input

K = size of conv. matrix

s = stride length



Forward:

$$y_{ij}^{(k)} = \alpha_0^{(k)} + \sum_{c=1}^{C^I} \sum_{q=1}^{K} \sum_{r=1}^{K} \alpha_{qr}^{(k)} x_{mn}^{(c)} \quad \text{where } m = s(i-1)+q,\ n = s(j-1)+r$$

Backward:

$$\frac{dJ}{d\alpha_0^{(k)}} = \sum_i \sum_j \frac{dJ}{dy_{ij}^{(k)}} \frac{dy_{ij}^{(k)}}{d\alpha_0^{(k)}}$$

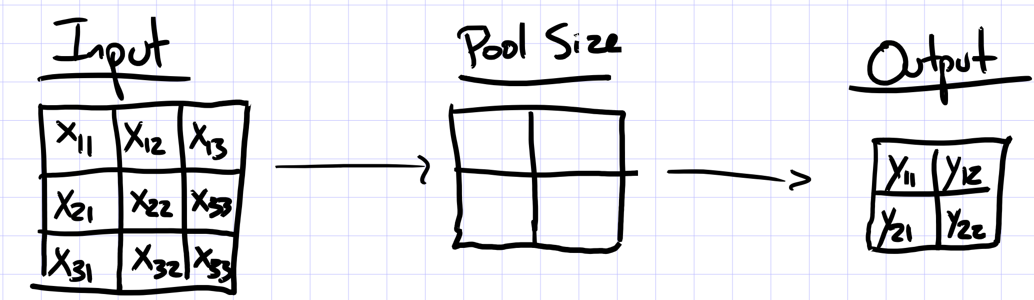
$$\frac{dJ}{d\alpha_{qr}^{(k)}} = \sum_i \sum_j \frac{dJ}{dy_{ij}^{(k)}} \frac{dy_{ij}^{(k)}}{d\alpha_{qr}^{(k)}}$$

$$\frac{dJ}{dx_{mn}^{(c)}} = \sum_i \sum_j \sum_k \frac{dJ}{dy_{ij}^{(k)}} \frac{dy_{ij}^{(k)}}{dx_{mn}^{(c)}}$$

just some calculus

Max-Pooling Layer

Ex: 1 input channel, 1 output channel, stride of 1

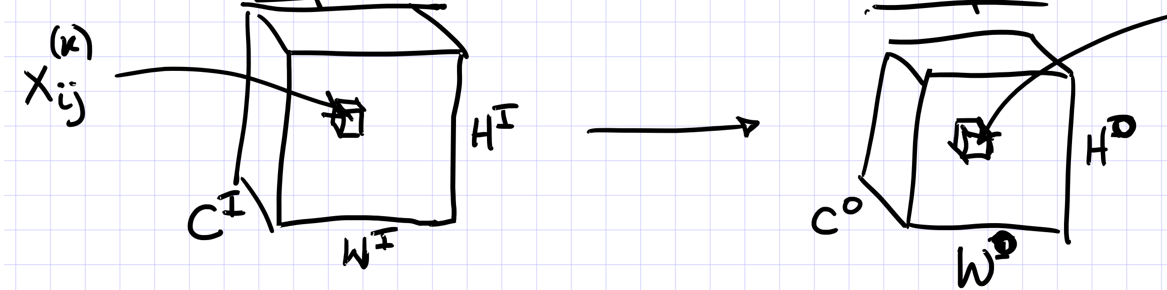


$$y_{11} = \max(x_{11}, x_{12}, x_{21}, x_{22})$$
$$y_{12} = \max(x_{12}, x_{13}, x_{22}, x_{23})$$
$$y_{21} = \max(x_{21}, x_{22}, x_{31}, x_{32})$$
$$y_{22} = \max(x_{22}, x_{23}, x_{32}, x_{33})$$

Max-Pooling Layer

Max-Pooling Layer

Input: $\qquad\qquad\qquad\qquad$ Output:



$y_{ij}^{(k)}$

$C^O = C^I$

$H^O = \ldots$ } same as conv. layer

$W^O = \ldots$ }

Same hyperparams too.

Forward:

$$y_{ij}^{(k)} = \max_{\substack{q \in \{1,\ldots,K\} \\ r \in \{1,\ldots,K\}}} x_{mn}^{(k)} \quad \text{where} \quad \begin{aligned} m &= s(i-1)+q \\ n &= s(j-1)+r \end{aligned}$$

Backward:

$$\frac{dJ}{dx_{mn}^{(k)}} = \sum_i \sum_j \frac{dJ}{dy_{ij}^{(k)}} \frac{dy_{ij}^{(k)}}{dx_{mn}^{(k)}}$$

Subderivatives

+ Max() is not differentiable, but subdifferentiable.
+ There are a set of derivatives and we can just choose one for SGD.

$y = \max(a,b)$

$$\Rightarrow \frac{dJ}{da} = \frac{dJ}{dy}\frac{dy}{da} \quad \text{where} \quad \frac{dy}{da} = \begin{cases} 1 & \text{if } a > b \\ 0 & \text{otherwise} \end{cases}$$

Convolutional Neural Network (CNN)

- Typical layers include:
  - Convolutional layer
  - Max-pooling layer
  - Fully-connected (Linear) layer
  - ReLU layer (or some other nonlinear activation function)
  - Softmax
- These can be arranged into arbitrarily deep topologies

Architecture #1: LeNet-5

PROC. OF THE IEEE, NOVEMBER 1998 7

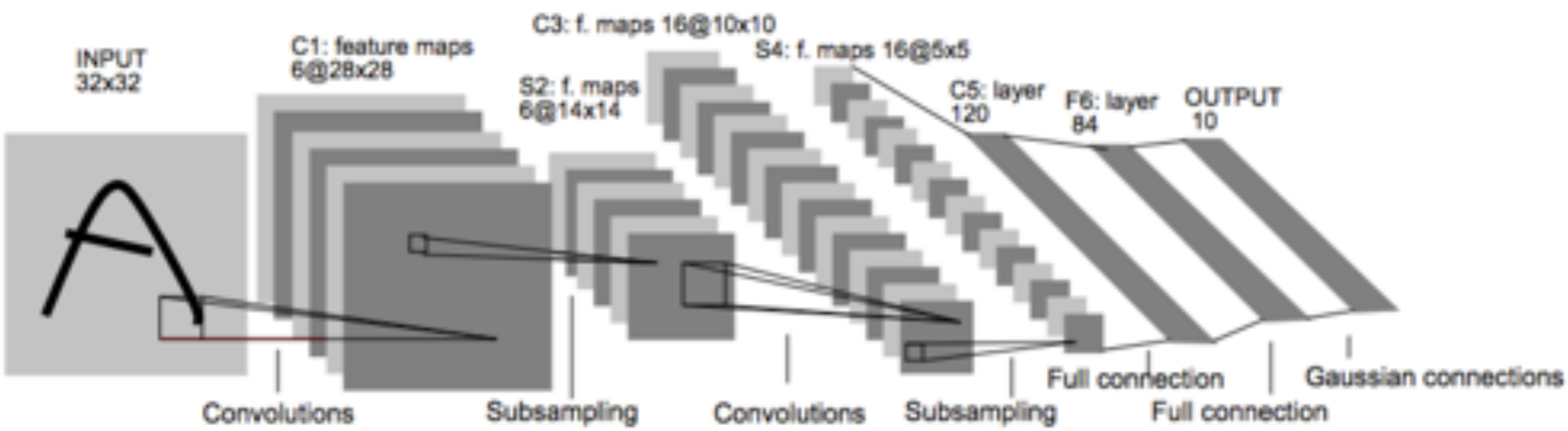


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

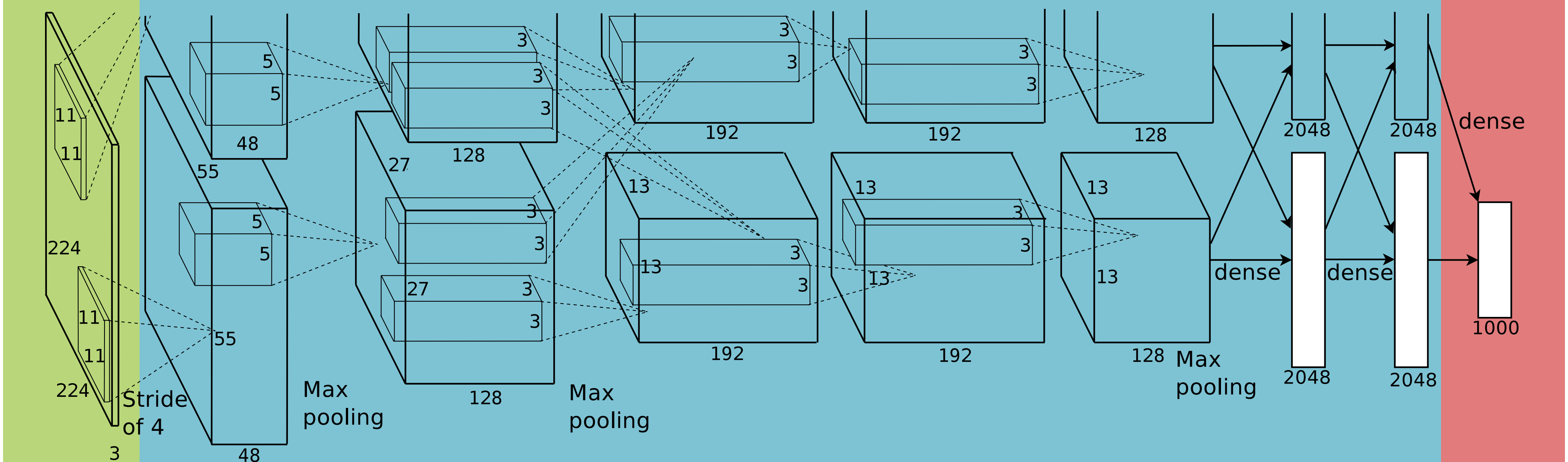
Architecture #2: AlexNet

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2012)
15.3% error on ImageNet LSVRC-2012 contest

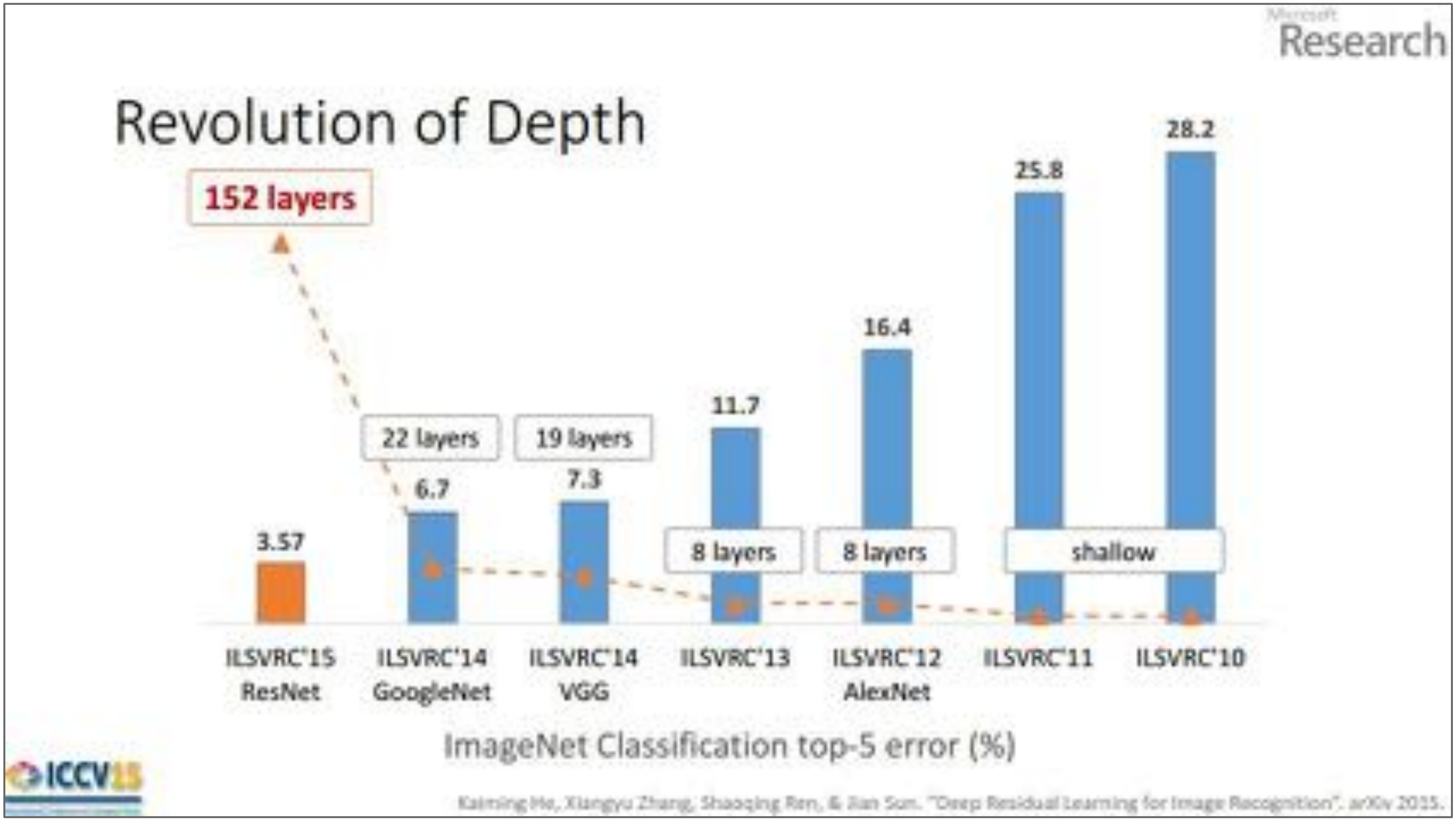
Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



CNNs for Image Recognition



Slide from Kaiming He

The key building block of ResNet

RESIDUAL CONNECTIONS

Slides in this section from…

Deep Residual Learning

MSRA @ ILSVRC & COCO 2015 competitions

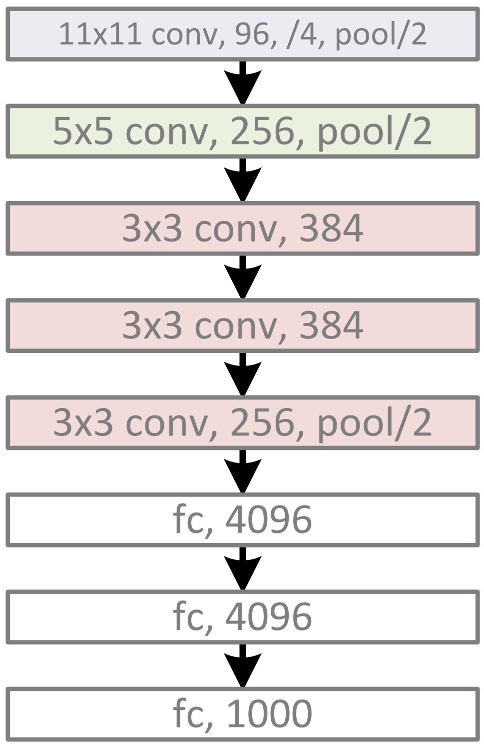
Kaiming He

with Xiangyu Zhang, Shaoqing Ren, Jifeng Dai, & Jian Sun

Microsoft Research Asia (MSRA)

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

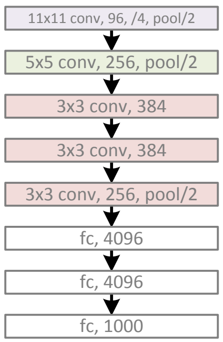


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

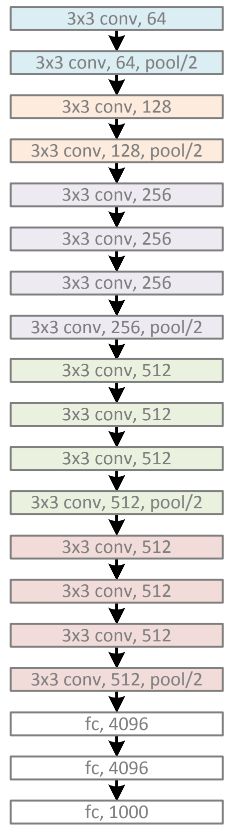
Microsoft Research

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

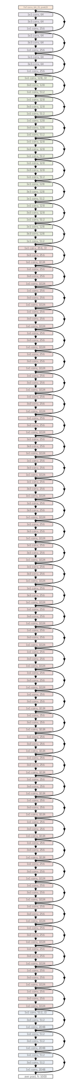
Microsoft Research

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



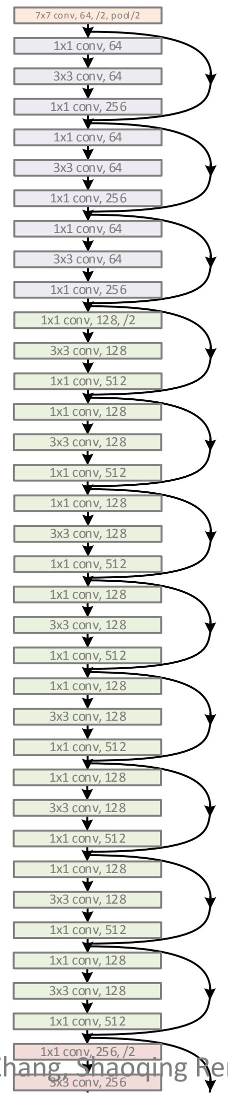
ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Microsoft Research

Revolution of Depth

ResNet, 152 layers



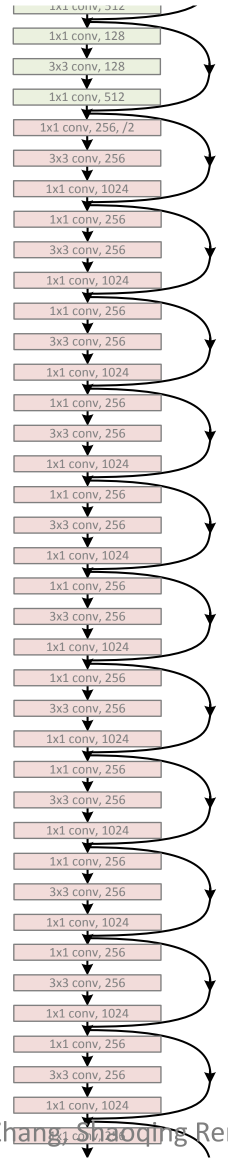
(there was an animation here)

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Revolution of Depth

ResNet, 152 layers



(there was an animation here)

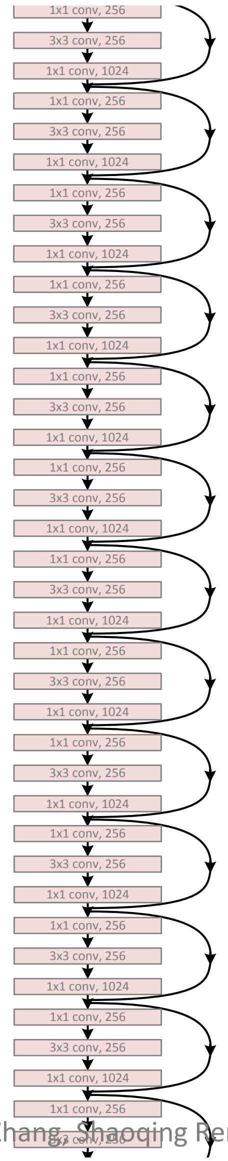
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Microsoft Research

ICCV15 International Conference on Computer Vision

Revolution of Depth

ResNet, 152 layers

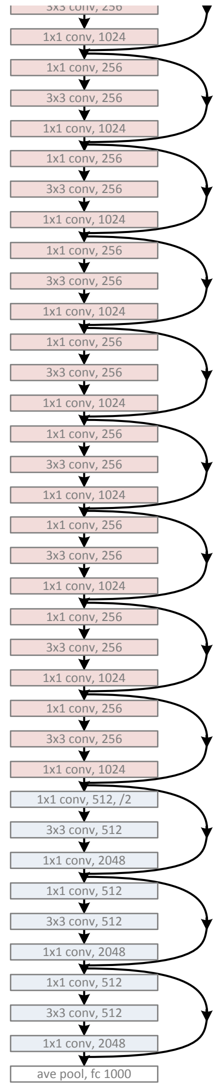


(there was an animation here)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Revolution of Depth

ResNet, 152 layers

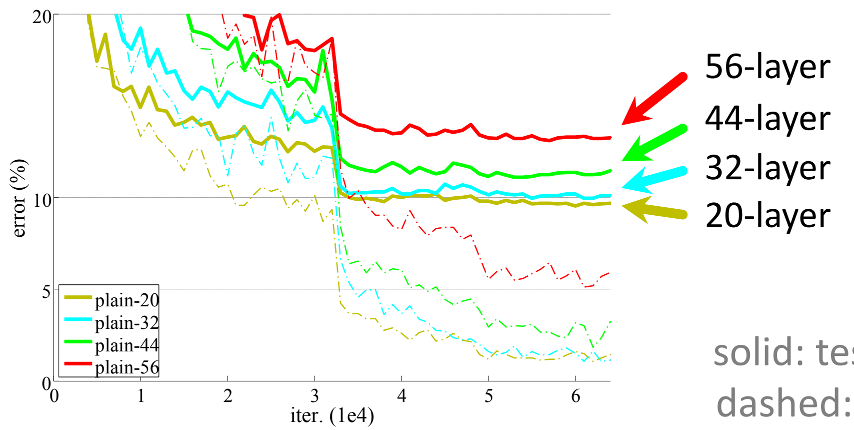


(there was an animation here)

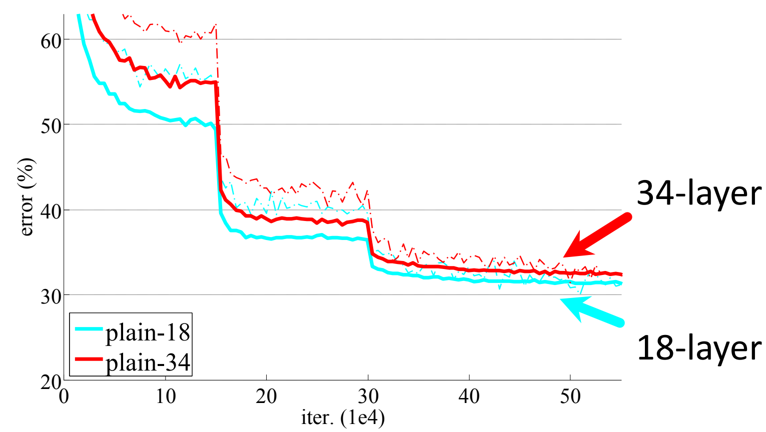
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Simply stacking layers?

CIFAR-10



ImageNet-1000



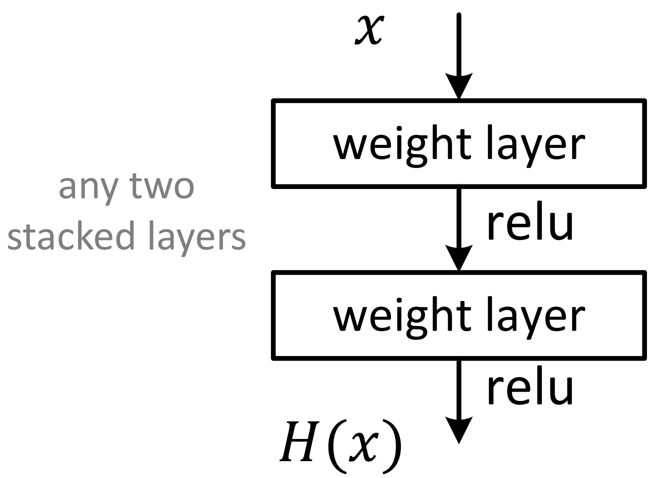
solid: test/val
dashed: train

- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Deep Residual Learning

- Plaint net



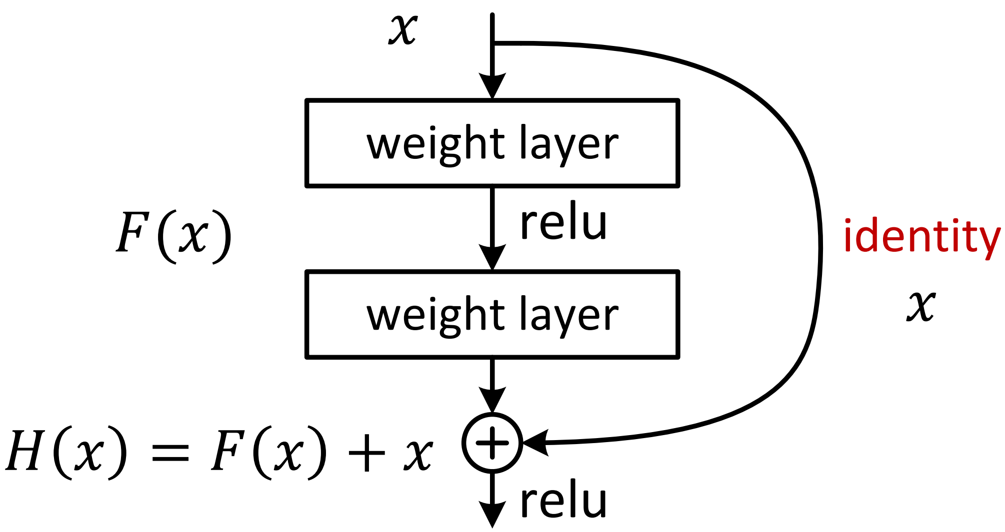
$H(x)$ is any desired mapping,

hope the 2 weight layers fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Deep Residual Learning

- Residual net



$H(x)$ is any desired mapping,

~~hope the 2 weight layers fit $H(x)$~~

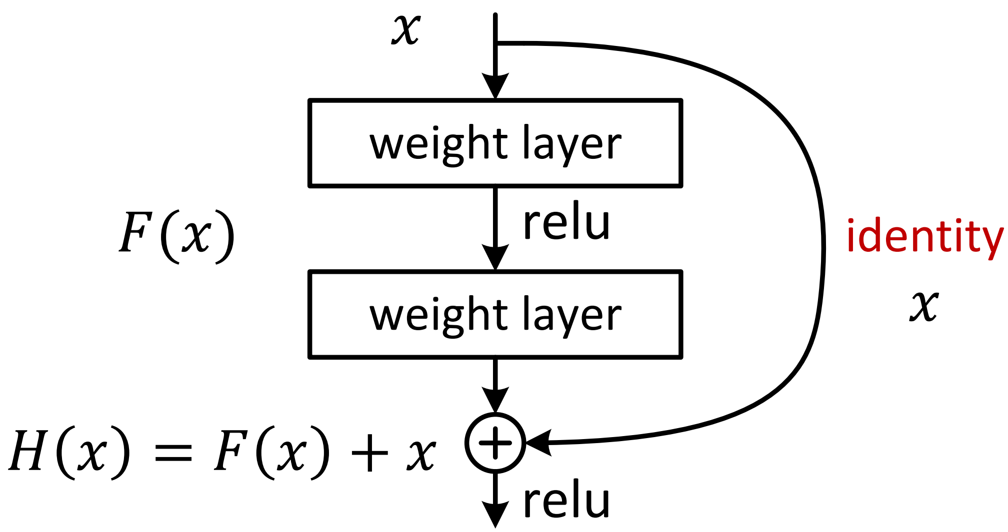
hope the 2 weight layers fit $F(x)$

let $H(x) = F(x) + x$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity

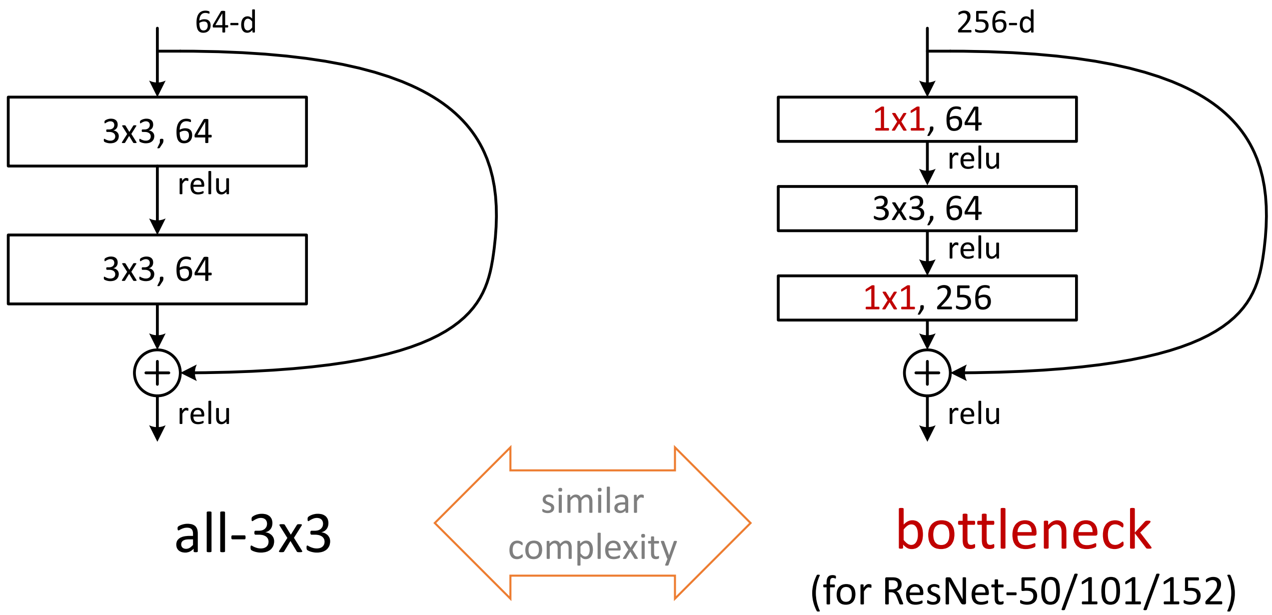


- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

ImageNet experiments

- A practical design of going deeper



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Microsoft Research

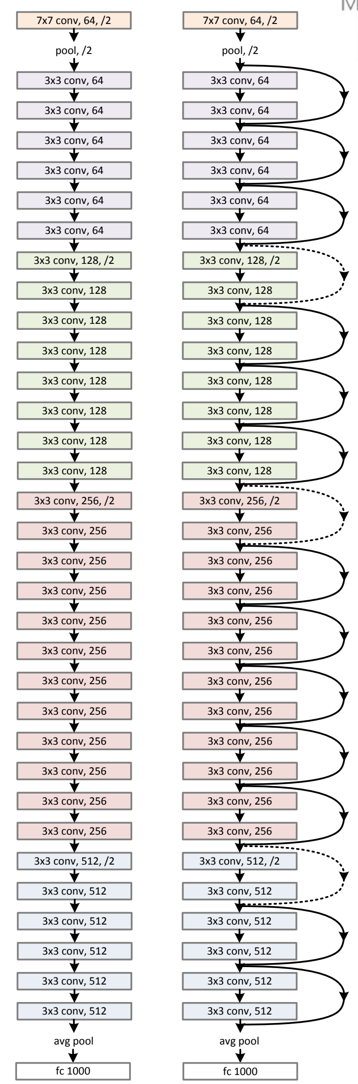
Network "Design"

- Keep it simple

- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2
  - Simple design; just deep!

- Other remarks:
  - no max pooling (almost)
  - no hidden fc
  - no dropout

plain net



ResNet

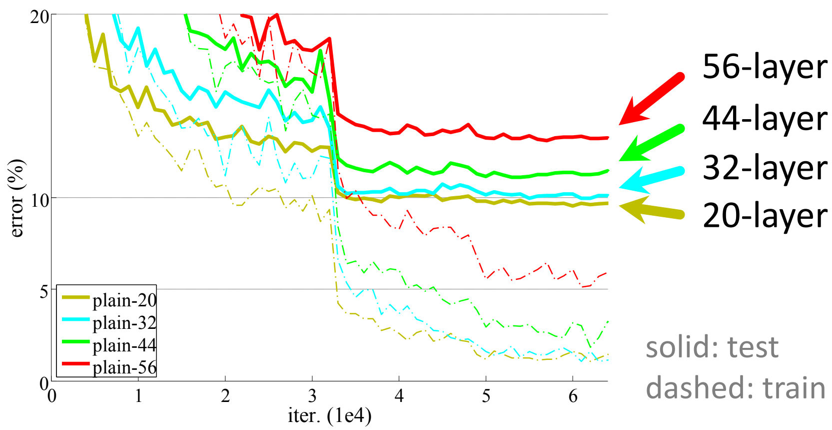
ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

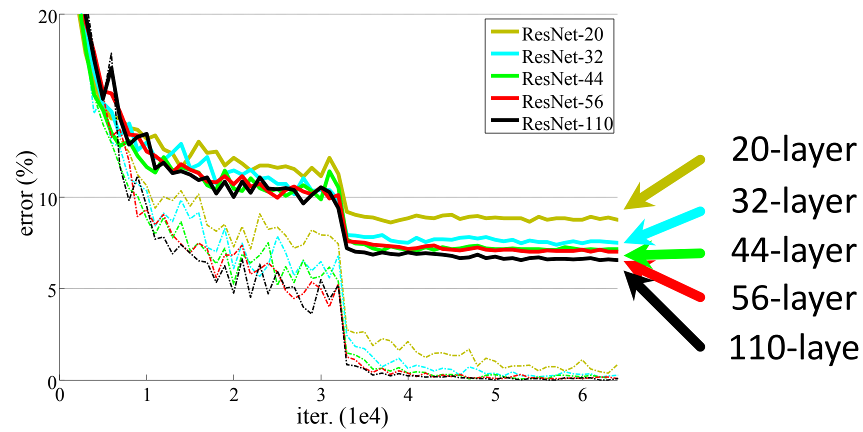
Microsoft Research

CIFAR-10 experiments

CIFAR-10 plain nets



CIFAR-10 ResNets

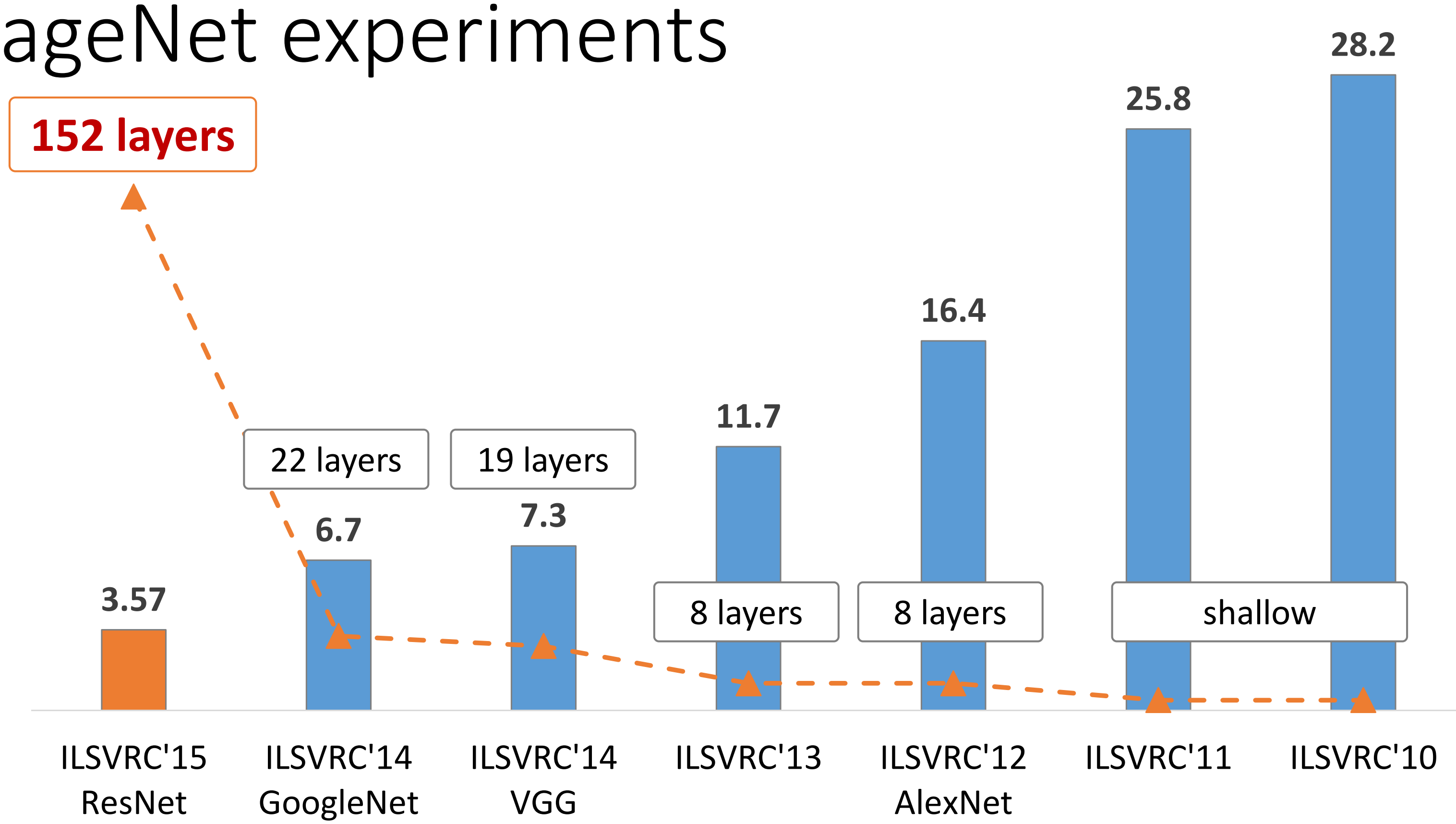


- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

ImageNet experiments



ImageNet Classification top-5 error (%)

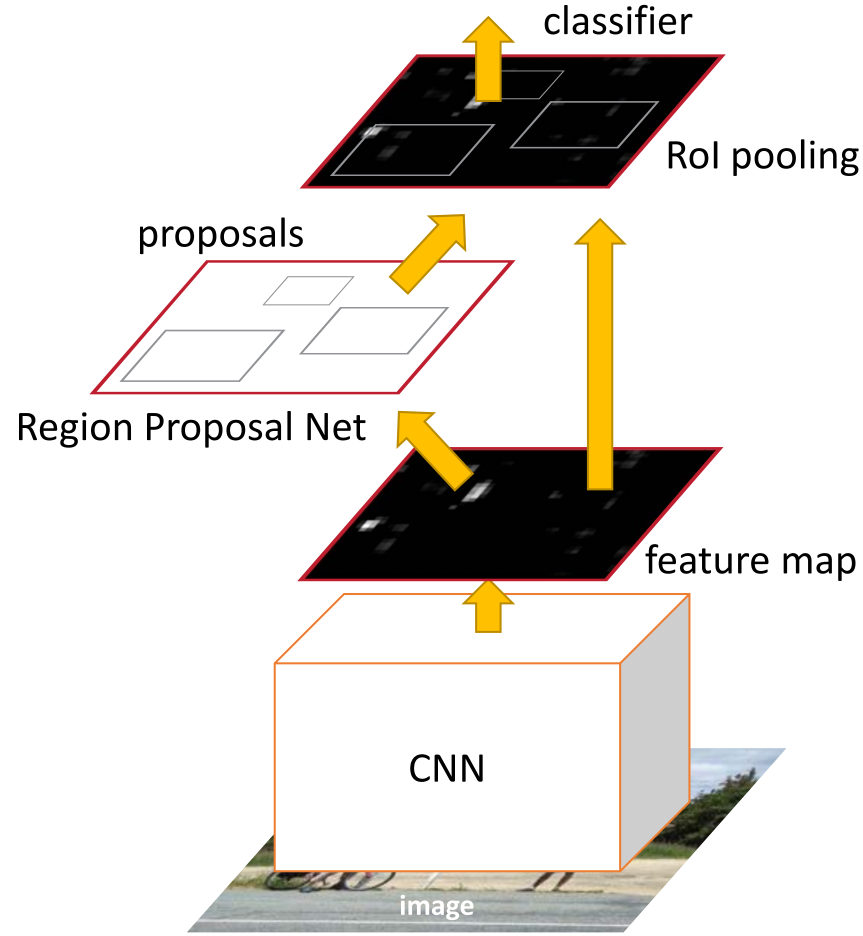
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Object Detection (brief)

- Simply "Faster R-CNN + ResNet"

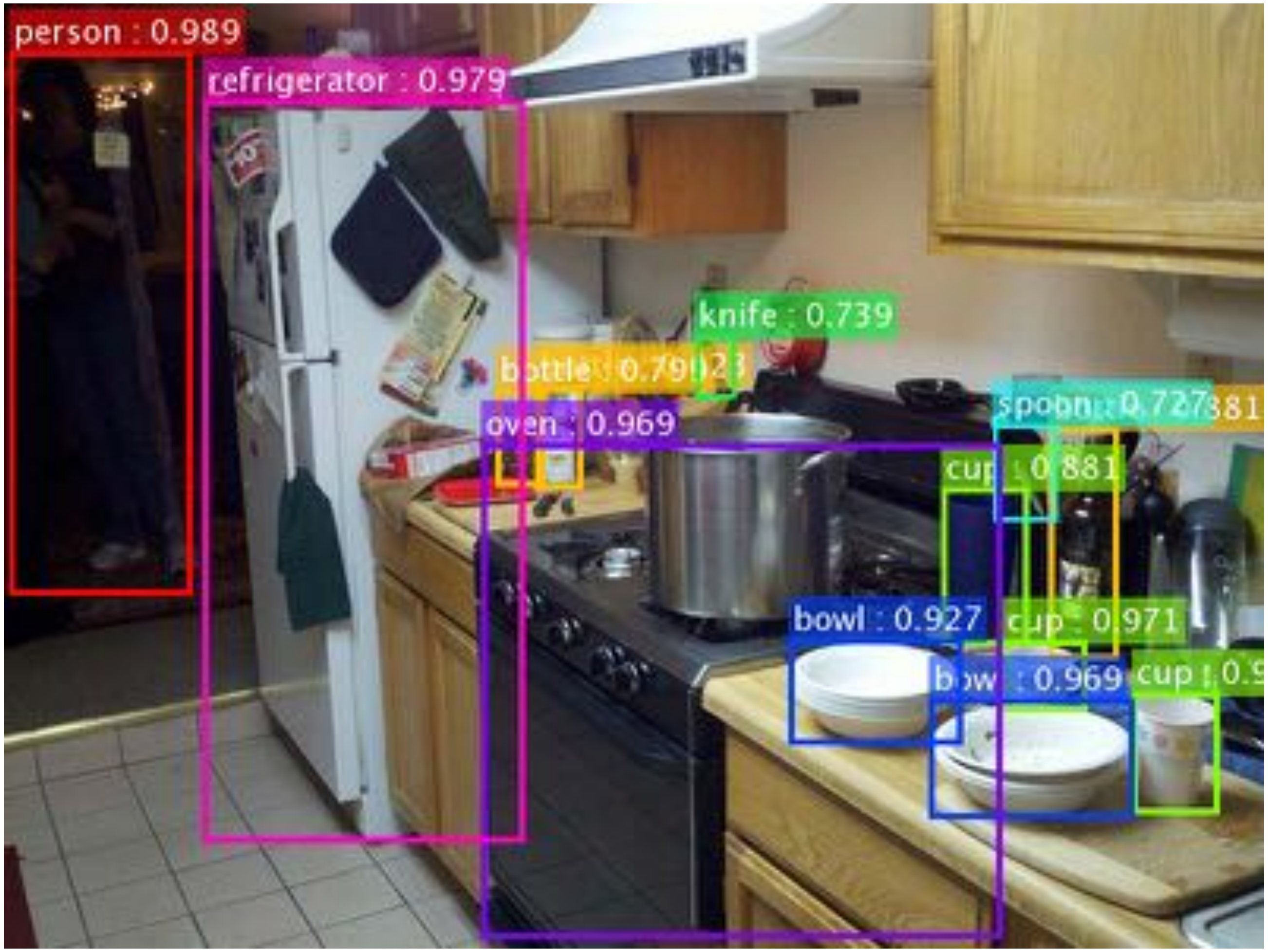
Faster R-CNN baseline	mAP@.5	mAP@.5:.95
VGG-16	41.5	21.5
ResNet-101	**48.4**	**27.2**

COCO detection results
(ResNet has 28% relative gain)



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

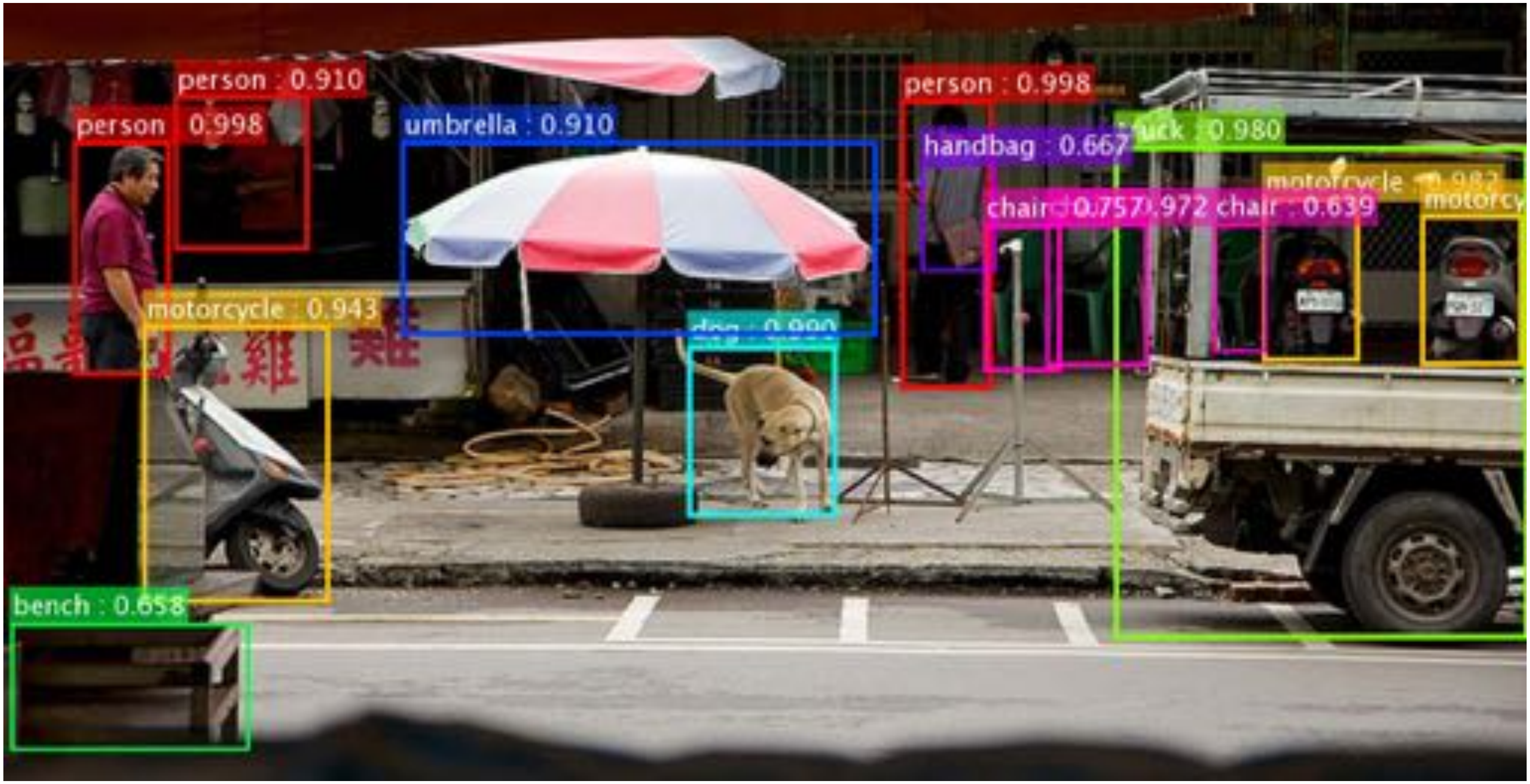
Microsoft Research



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

Microsoft Research



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

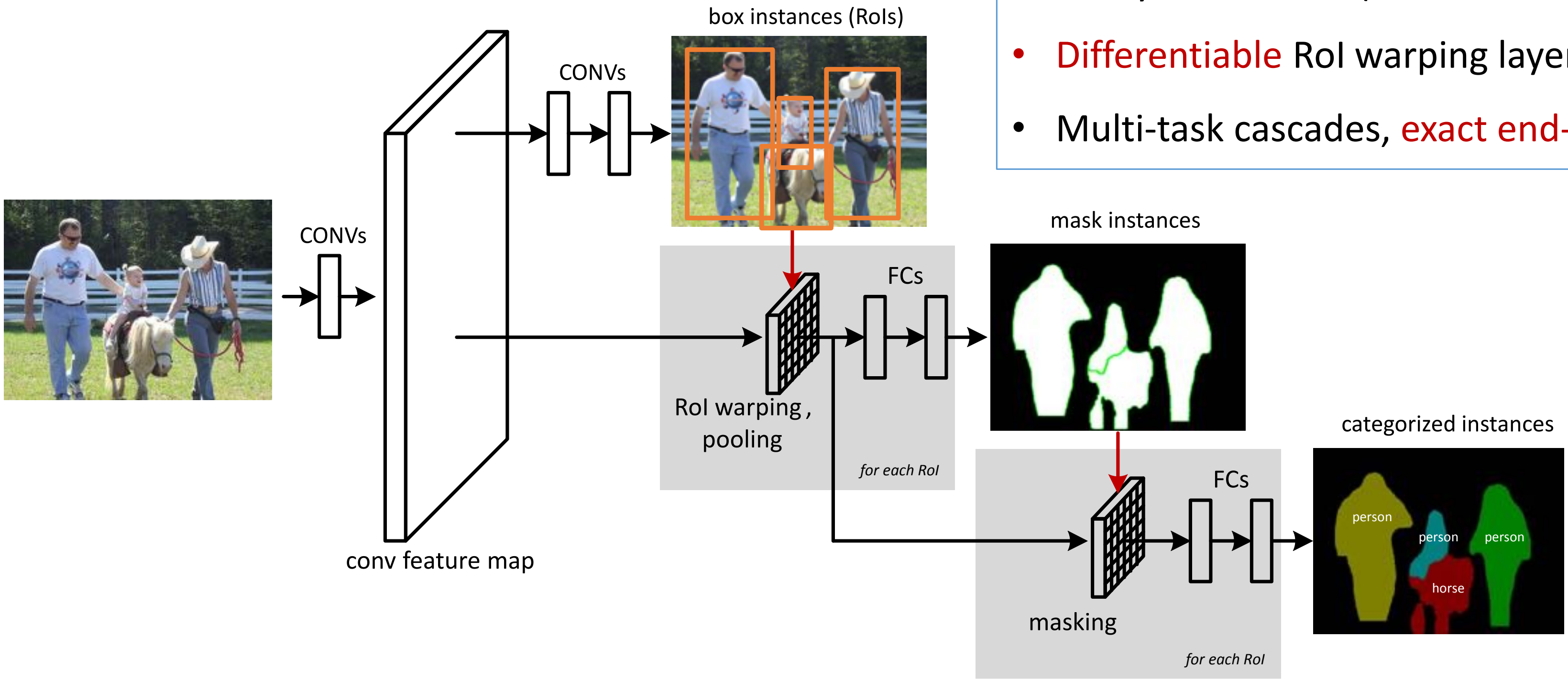
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

ICCV15 International Conference on Computer Vision

Microsoft Research

Instance Segmentation (brief)

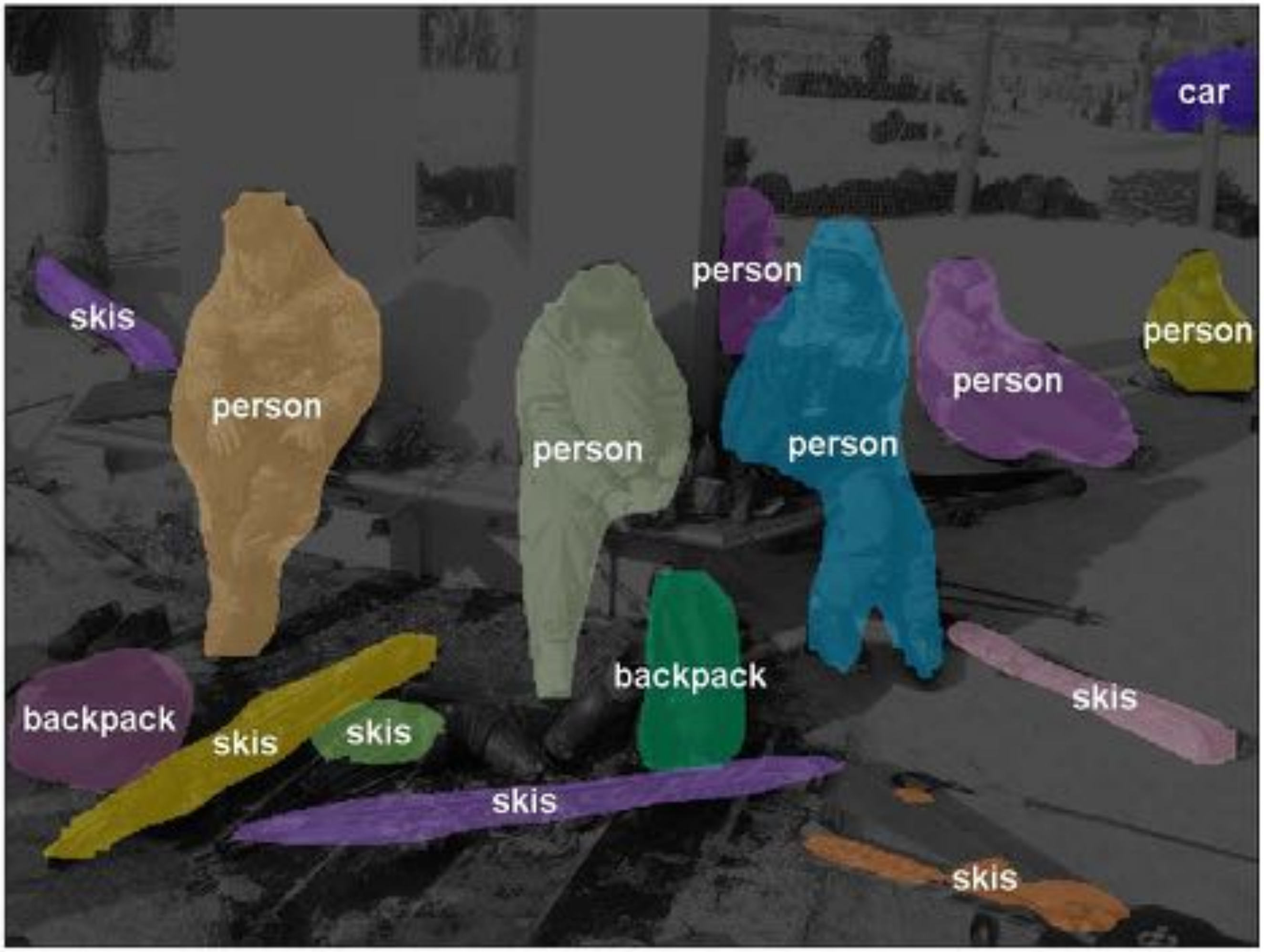
- Solely CNN-based ("features matter")
- Differentiable RoI warping layer (w.r.t box coord.)
- Multi-task cascades, exact end-to-end training



ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.
Jifeng Dai, Kaiming He, & Jian Sun. "Instance-aware Semantic Segmentation via Multi-task Network Cascades". arXiv 2015.

input



Microsoft Research

*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.
Jifeng Dai, Kaiming He, & Jian Sun. "Instance-aware Semantic Segmentation via Multi-task Network Cascades". arXiv 2015.

ICCV15 International Conference on Computer Vision

CNN Summary

CNNs

- Are used for all aspects of **computer vision**, and have won numerous pattern recognition competitions
- Able learn **interpretable features** at different levels of abstraction
- Typically, consist of **convolution** layers, **pooling** layers, **nonlinearities**, and **fully connected** layers

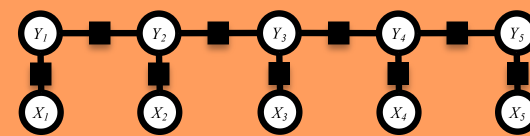
APPROXIMATE MARGINAL INFERENCE

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$



3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

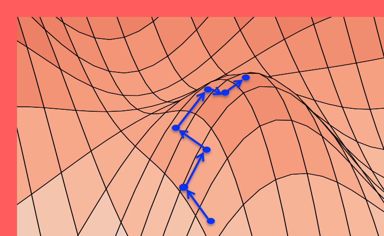
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\text{argmax}}\ p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



A Few Problems for a Factor Graph

Suppose we already have the parameters of a Factor Graph…

1. How do we compute the probability of a specific assignment to the variables?
   P(T=t, H=h, A=a, C=c)

2. How do we draw a sample from the joint distribution?
   t,h,a,c ~ P(T, H, A, C)

3. How do we compute marginal probabilities?
   P(A) = …

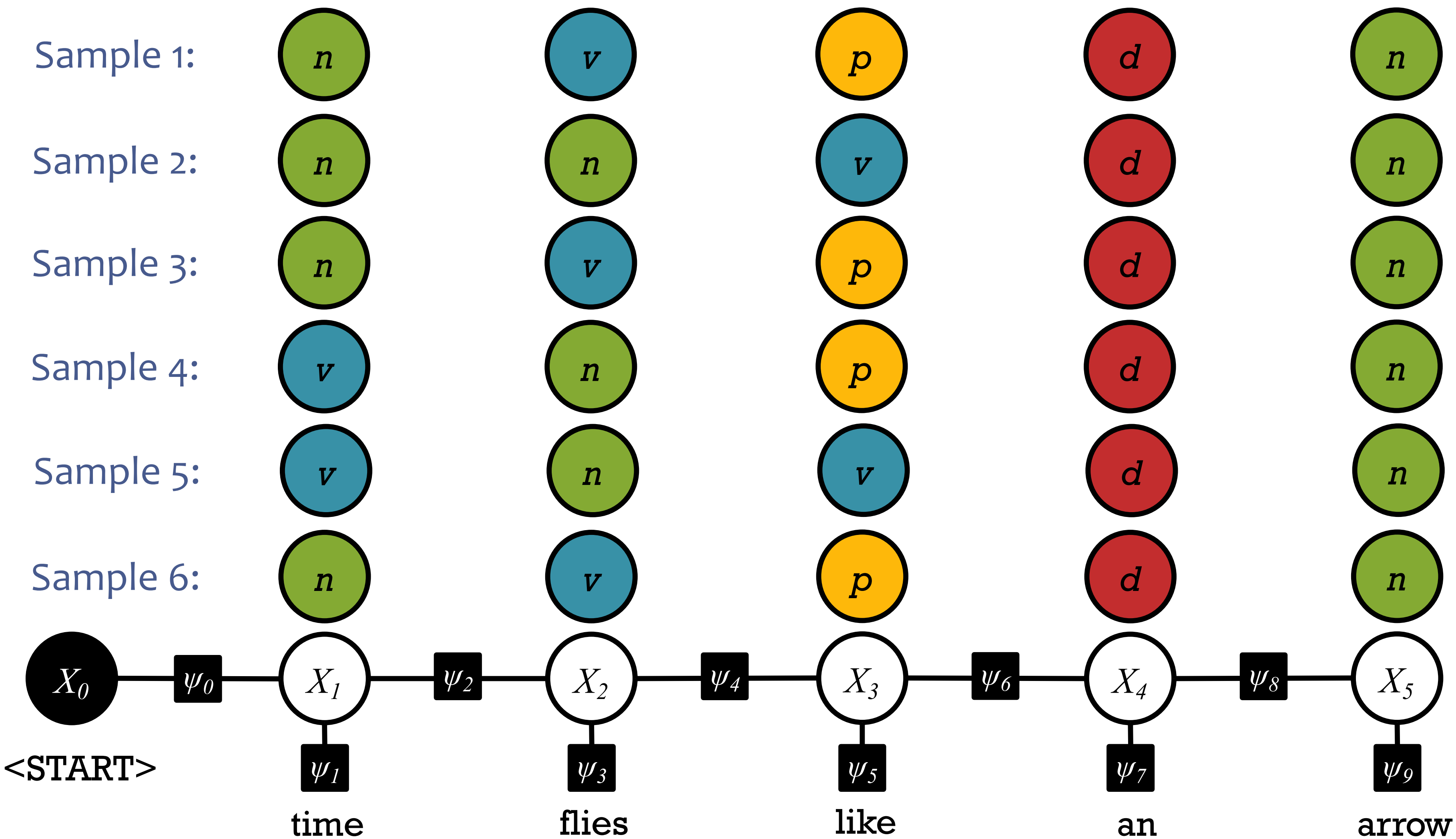
4. How do we draw samples from a conditional distribution?
   t,h,a ~ P(T, H, A | C = c)

5. How do we compute conditional marginal probabilities?
   P(H | C = c) = …

Can we use samples ?

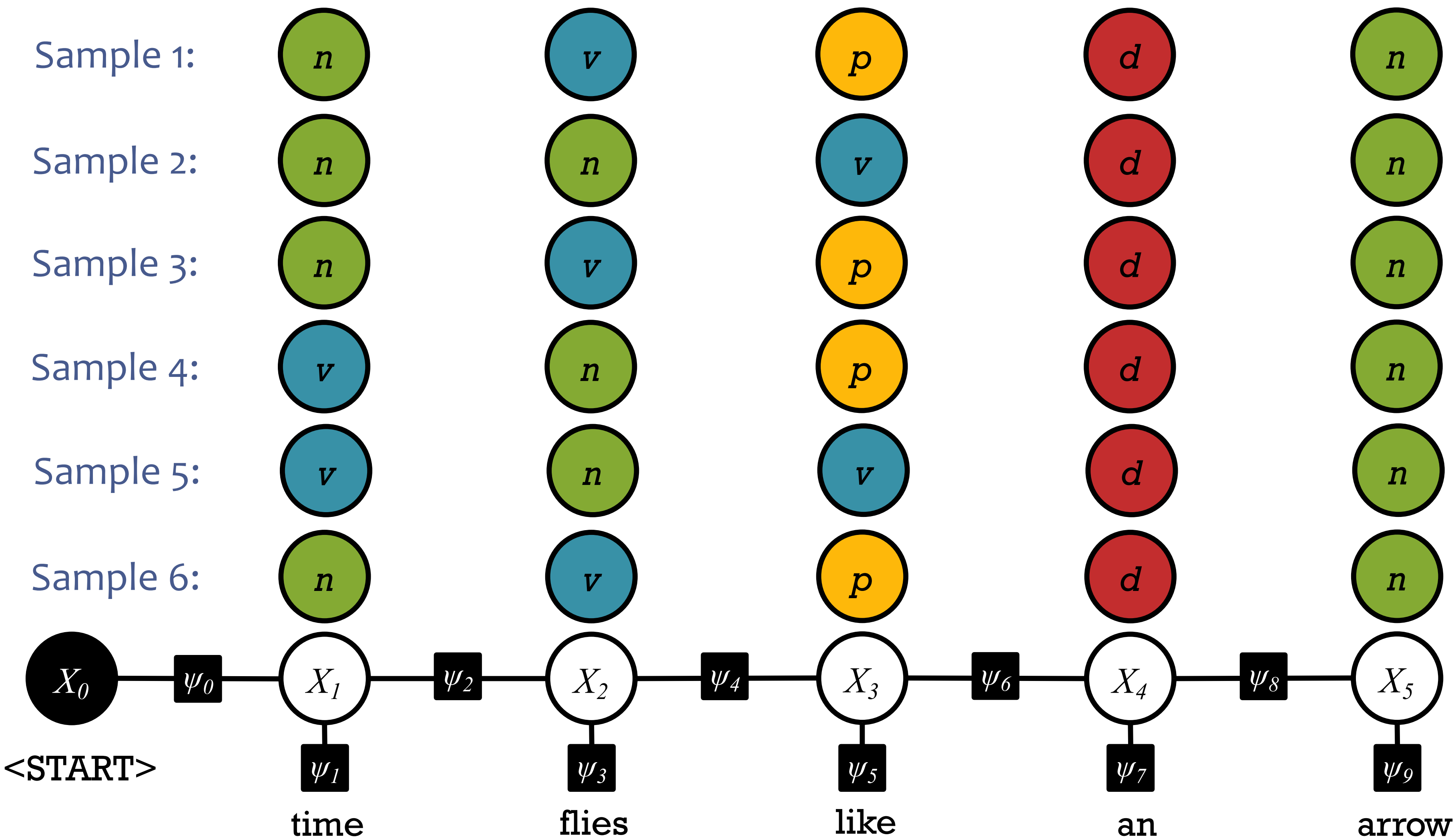
Marginals by Sampling on Factor Graph

Suppose we took many samples from the distribution over taggings: $p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$



Marginals by Sampling on Factor Graph

The marginal $p(X_i = x_i)$ gives the probability that variable X_i takes value x_i in a random sample



Marginals by Sampling on Factor Graph