technische universität
dortmund

Word Spotting: From Bag-of-Features to Deep Learning

— ICDAR 2017 Tutorial, Kyoto, Japan —

Gernot A. Fink & Sebastian Sudholt

November 12, 2017

- Introduction
- Bag-of-Features: Fundamentals
- Learning Document Image Representations
- Learning Word Spotting Models
- Deep Learning Fundamentals
- Deep Learning for Word Spotting
- Understanding CNN-Based Word Spotting
- Summary

with contributions by René Grzeszick and Leonard Rothacker

Introduction: Automatic Reading Systems

State of Automatic Reading:

- One of the earliest application fields studied in computer science
- So-called OCR achieves high-quality results for machine-printed text in well-defined settings.
- Online handwriting recognition again gaining popularity
- Offline handwriting recognition: Remarkable results, but still an open research problem

General Methodology:

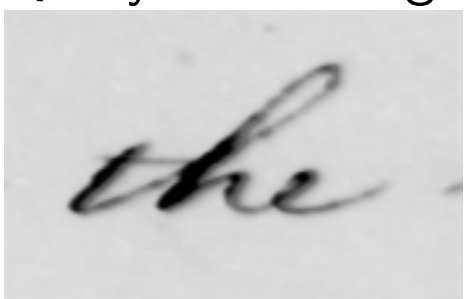
Statistical sequence models (e.g. HMMs, BLSTMs) that are trained from *extensive* amounts of example data

Introduction: Why Word Spotting?

What if automatic transcription of handwriting is no longer feasible?

Alternative: Retrieval of individual words rather than transcription ("query-by-example")

Query word image



Document image

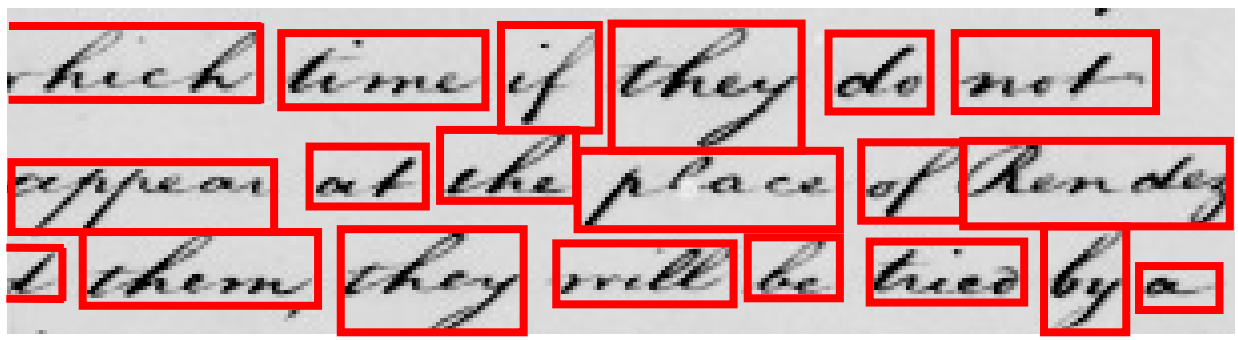
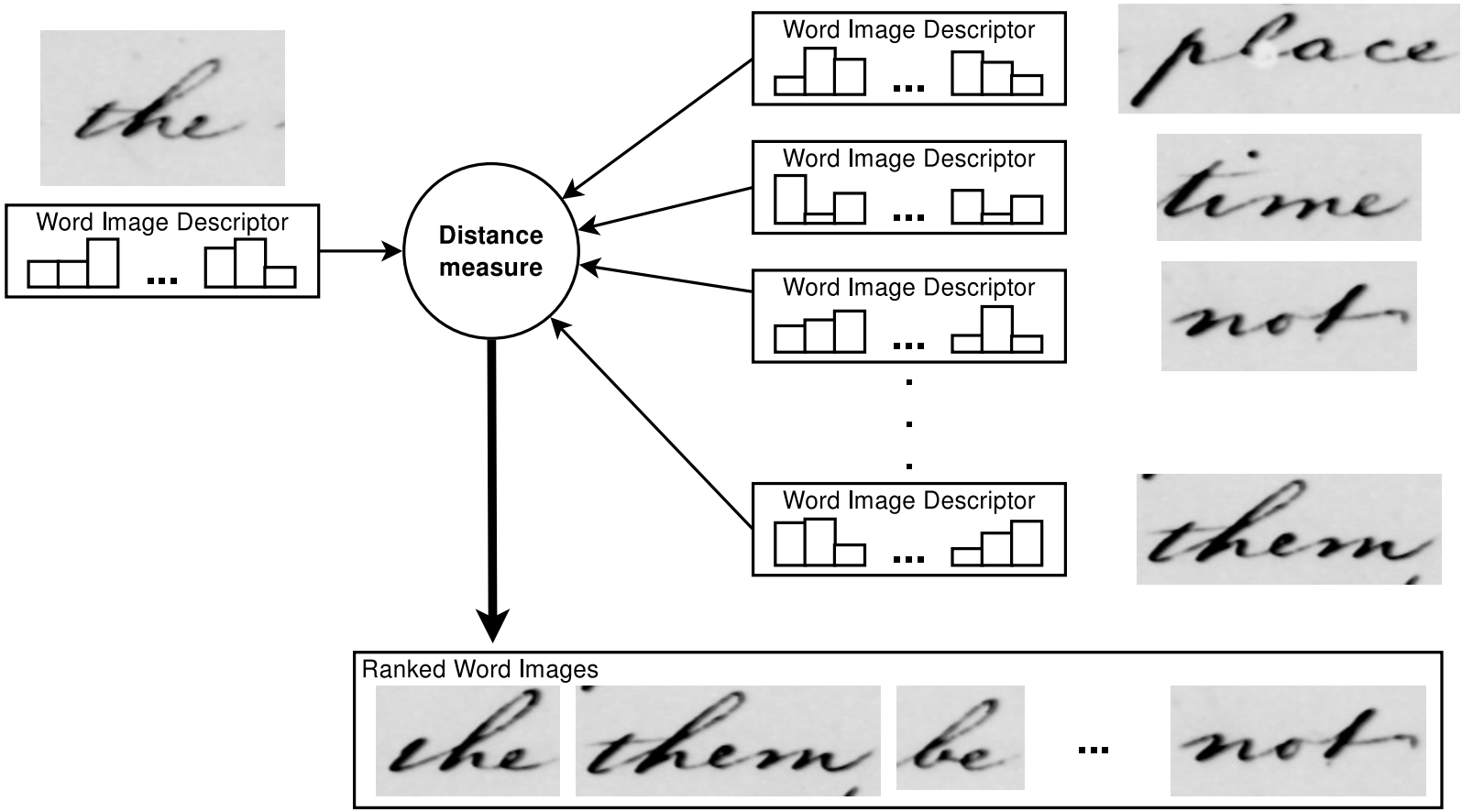


Image of Feldpost postcard from *Private Collection Dr. Britta Bley*, Dortmund, Germany Images from *The George Washington Papers at the Library of Congress, 1741-1799*

Introduction: Basic Methodology

Query-by-example word spotting



Based on [Rath & Manmatha, IJRAR'07]

Word Spotting: Fundamentals

Core Methodology:

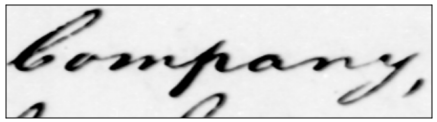
- Specialized image retrieval
- Important ingredient: Image matching procedure
- Frequently required: Pre-segmentation (words / lines)

Taxonomy:

- *Segmentation-based*
- *Segmentation-free*, i.e., segmentation problem covered during retrieval
- *Query-by-Example*, i.e., word image directly used as query
- *Query-by-String*, i.e., query model derived from textual query ("string")

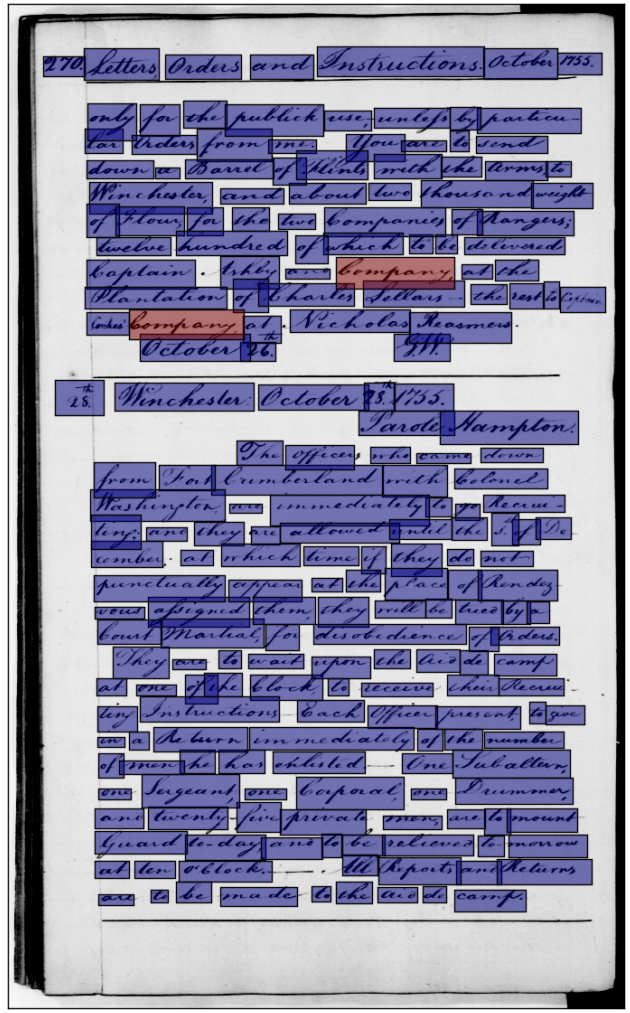
Word Spotting Tasks

Query by Example



Query by String

Company



Images from *The George Washington Papers at the Library of Congress, 1741-1799*

Word-Spotting: Milestones

Manmatha *et al.* 1996: First influential work

(Binarization, Alignment, XOR distance)

Rath & Manmatha 2003: DTW matching

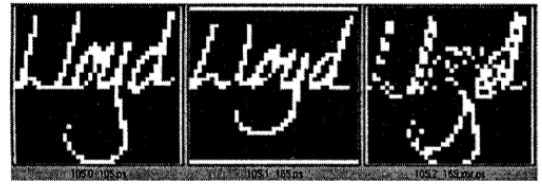
(Normalization, profile features)

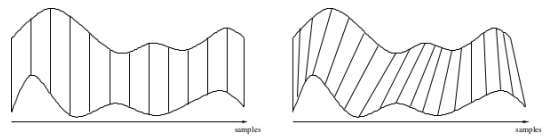
Rusiñol *et al.* 2011: First influential work using BoF, first with Spatial Pyramid

(SIFT, BoF, Spatial Pyramid, LSI, segmentation-free decoding)

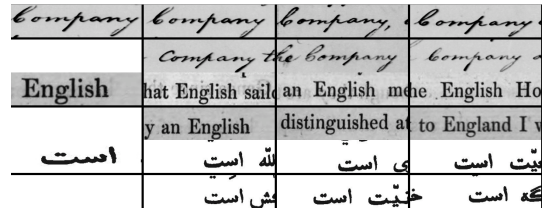
Almazan *et al.* 2014: HOG features

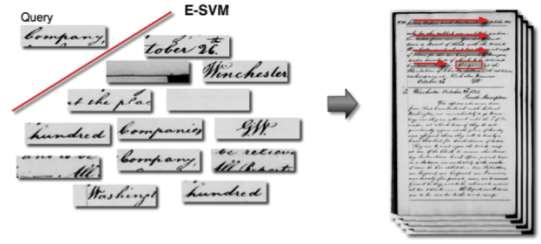
("Exemplar SVM", query expansion)





(a) naive alignment after resampling, (b) alignment with DTW.





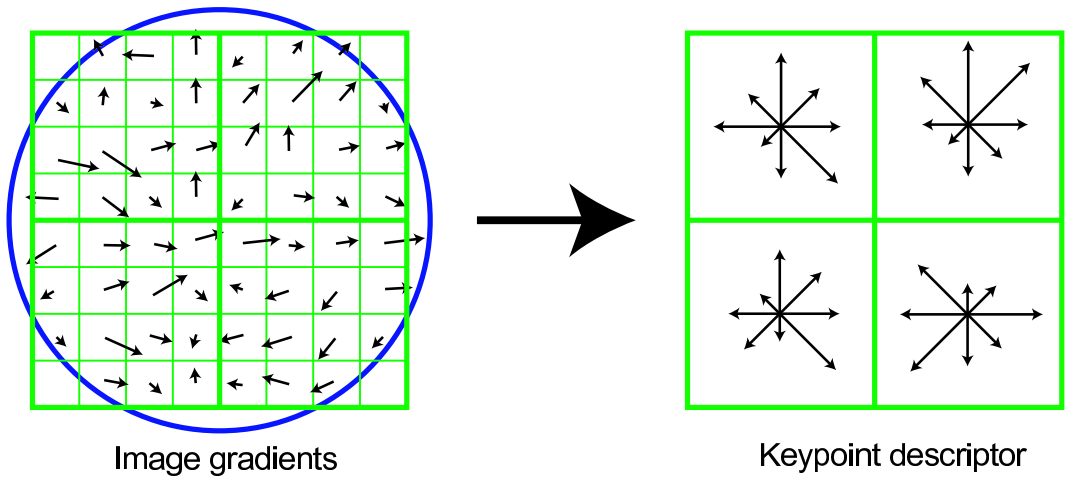
Overview

Local Image Descriptors

Fundamentals:

- Local gradient statistics
  (i.e. implicit description of object contours)
- Grouping of multiple such local statistics in a certain neighborhood and *normalization*
  (coarse description of structural properties)

Example: SIFT descriptor [Lowe, 2004]



(Source: [Lowe, IJCV 2004])

Note: Ex. several similar descriptors, e.g., HOG, SURF

Lowe, D.: *Distinctive Image Features from Scale-Invariant Keypoints*. Int. J. of Computer Vision, 60(2), pp. 91–110, 2004.

Statistical Image Modeling

Representation: Bag-of-Features Models (BoF)

- Originally proposed as Bag-of-Words models for representing texts

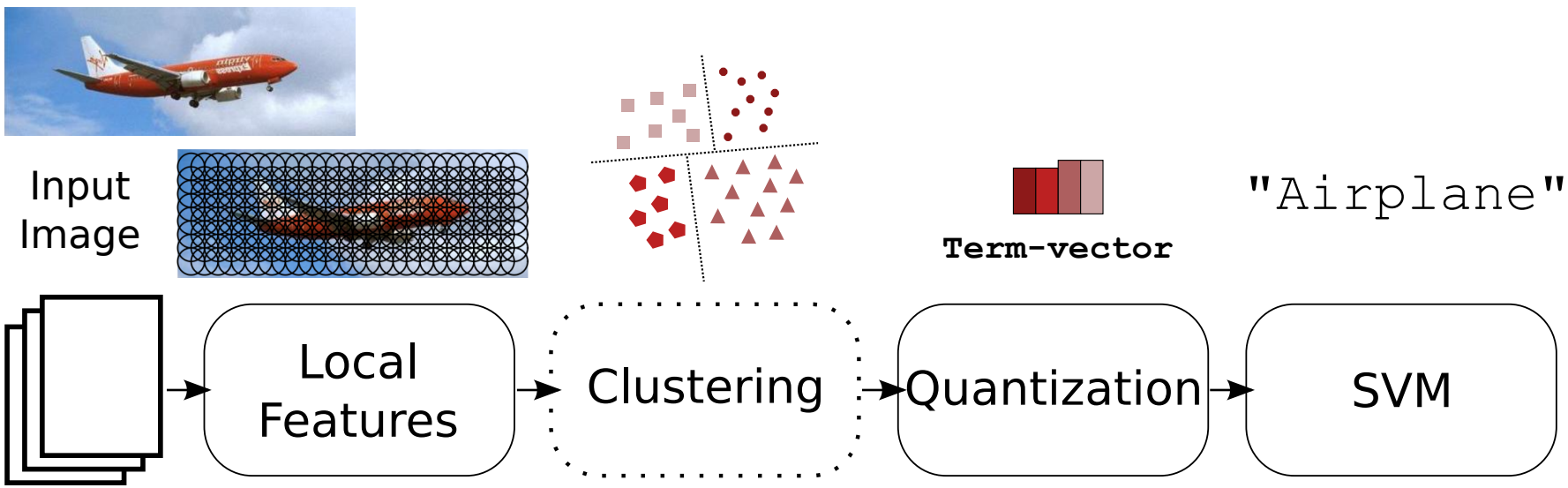
BoF-Approach: (cf. e.g. [O'Hara & Draper 2011])

0. Extract local image features (usually in a grid)
1. Compute *visual vocabulary* by quantizing local image features
2. For given image, compute histogram of quantized features (i.e. orderless "bag" of features)
3. Any classification / matching technique can be applied to BoF representations

S. O'Hara & B. A. Draper: Introduction to the Bag of Features Paradigm for Image Classification and Retrieval, Computing Research Repository, arXiv:1101.3354v1, 2011.

Statistical Image Modeling II

BoF-Approach: Processing Pipeline

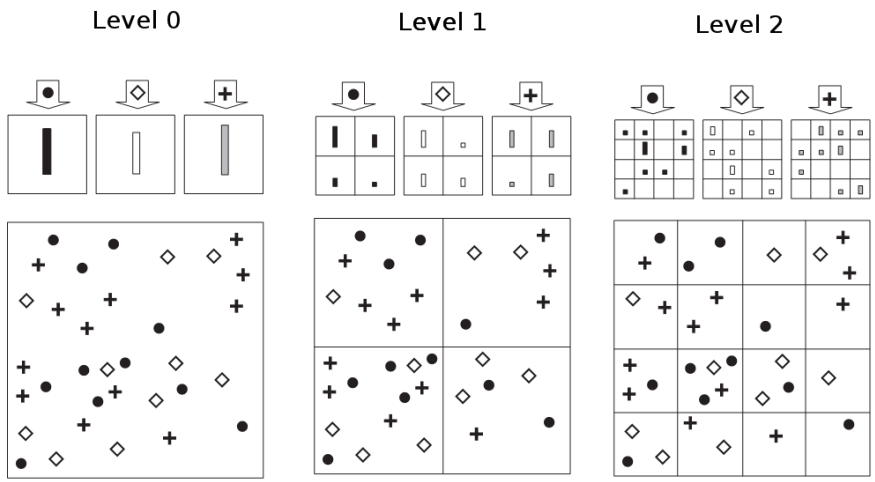


Main drawback: Spatial information is lost!

Statistical Image Modeling III

Goal: Compensate loss of spatial information

Method: Spatial-pyramid models



Lazebnik, S., Schmid, C., Ponce, J.: *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories*, IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Disadvantage: Considerably increased model complexity

Overview

Bag-of-Features Models for Word Spotting

Basic Methodology:

Image Features:

Gradient-based descriptors, e.g., SIFT, HOG

Feature Extraction:

Dense grid, i.e., no keypoint detection involved

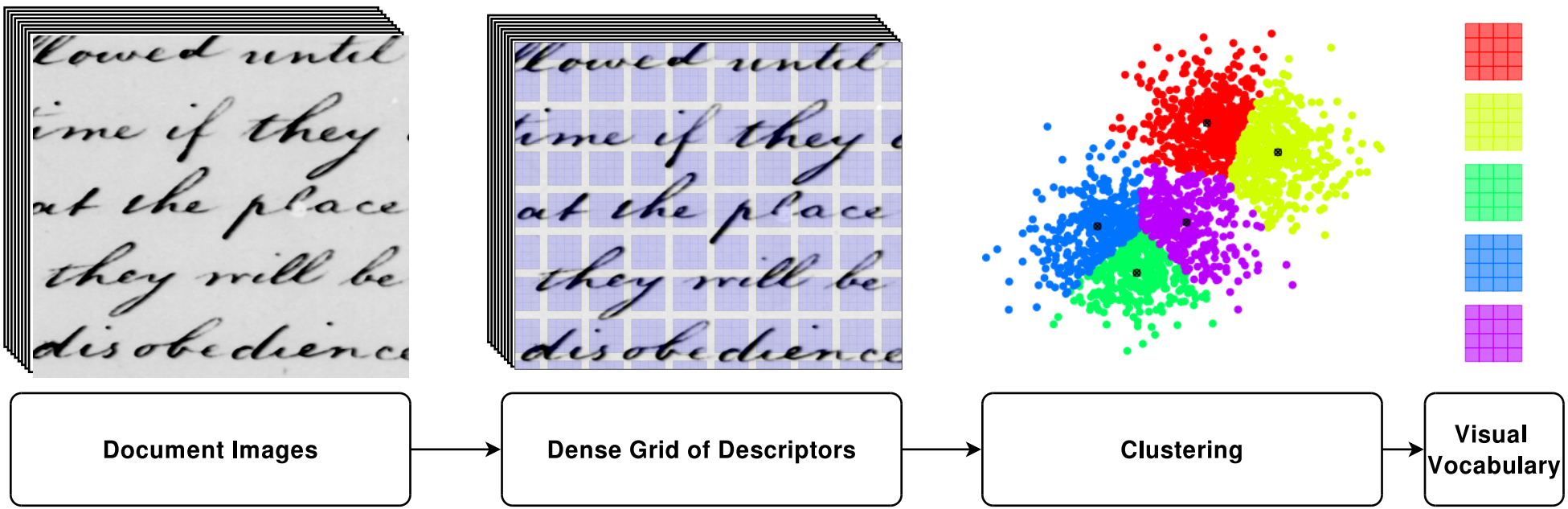
Visual Vocabulary:

- Quantization of descriptors (as "usual")
- Special: Large vocabularies (i.e. 2K to 4K) in order to capture appearance of "individuals" precicely

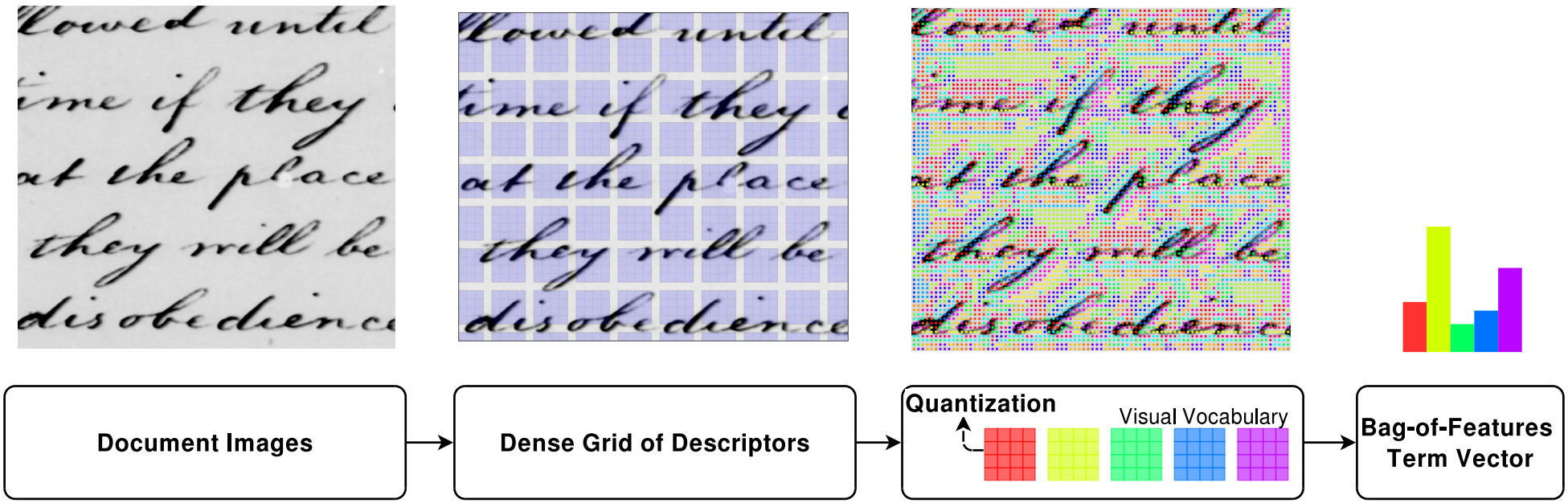
Query Model:

- Segmentation- or patch-based processing
- 1D spatial pyramid for improved spatial modeling

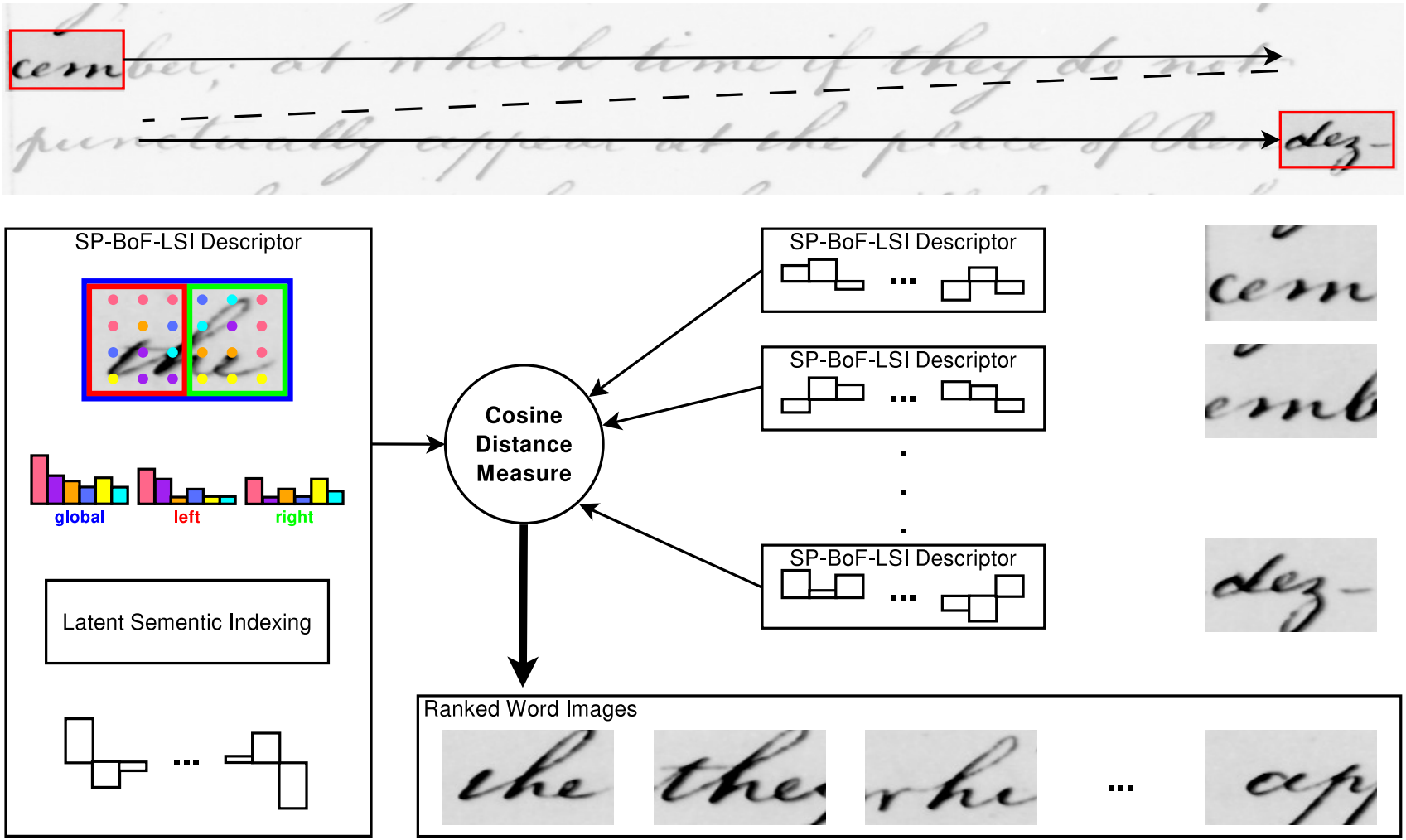
Bag-of-Features Models: Visual Vocabulary



Bag-of-Features Models: Term Vector

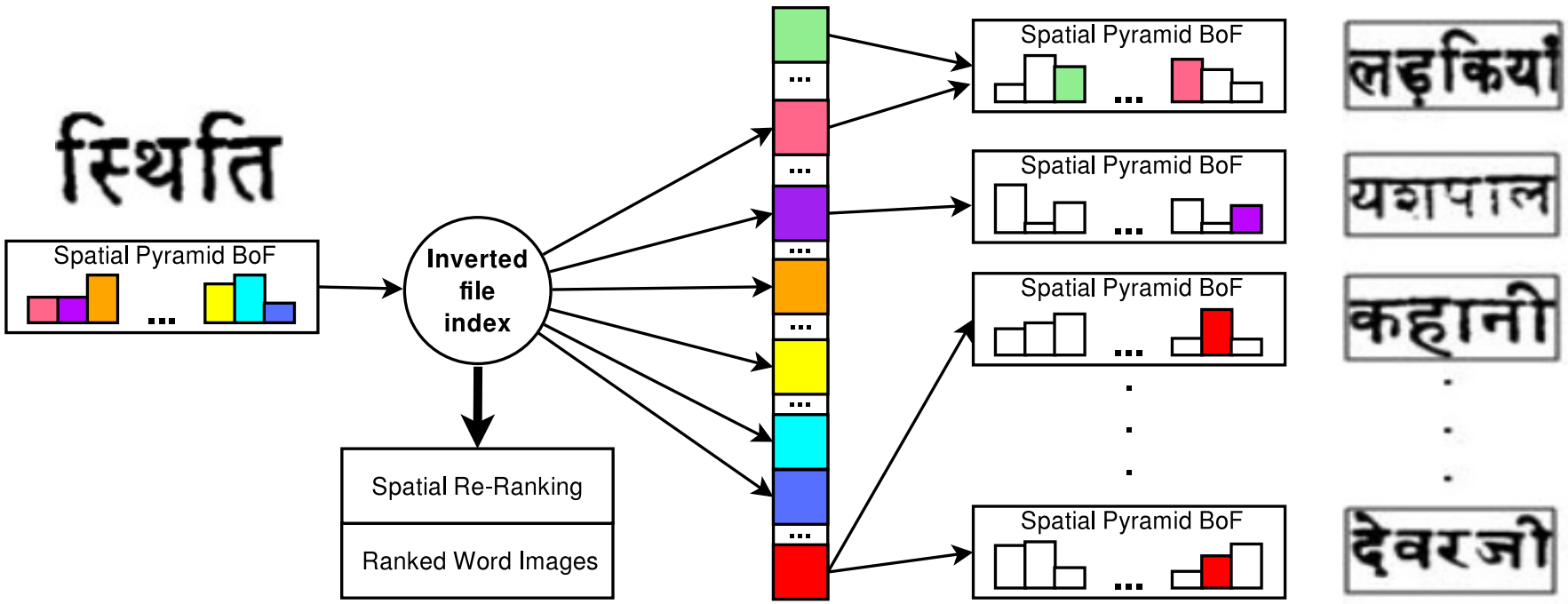


Segmentation-free Word Spotting with Bag-of-Features



Rusiñol, M., and Aldavert, D., and Toledo, R., and Lladós, J.: Browsing heterogeneous document collections by a segmentation-free word spotting method, Int. Conf. on Document Analysis and Recognition, Bejing, pp. 63-67, 2011.

Segmentation-based Word Spotting with Bag-of-Features



Shekhar, R., and Jawahar, C.: Word image retrieval using bag of visual words, Int. Workshop on Document Analysis Systems, pp. 297-301, 2012.

Overview

Learning BoF-Based Word Spotting Models

What we have so far:

- Expert-designed image descriptors
- Automatically learned document image features
  (visual vocabulary $\rightarrow$ histograms of quantized descriptors)
- Matching: Nearest neighbor

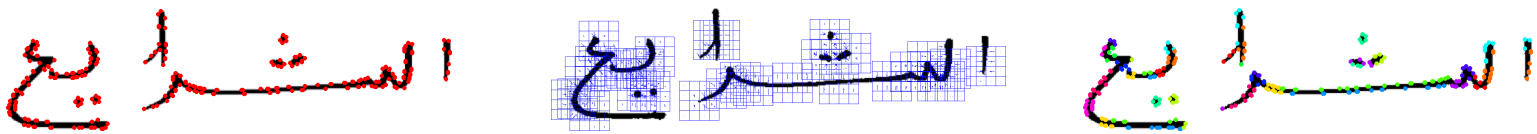
What we want in addition:

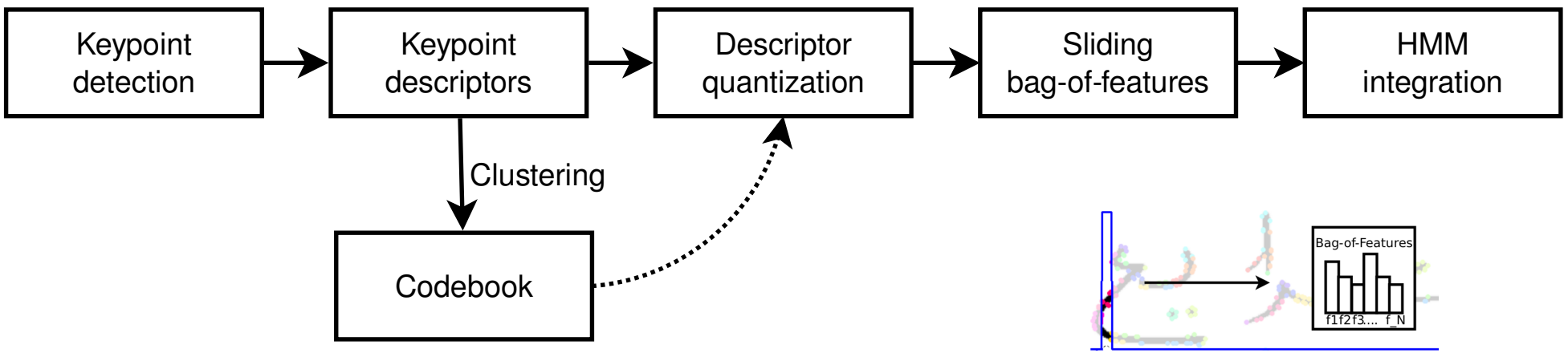
- More powerful image matching
- Categorization capabilities
  (i.e. links between image appearance and textual representation)

What we need: Learned classification models

Bag-of-Features HMMs

- *Extension* of HMMs towards learned feature representation
- *Extension* of BoF models towards fine-grained script representation

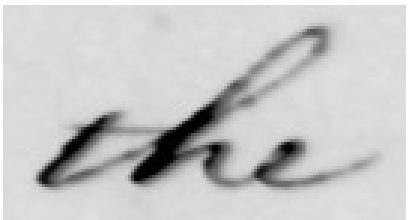




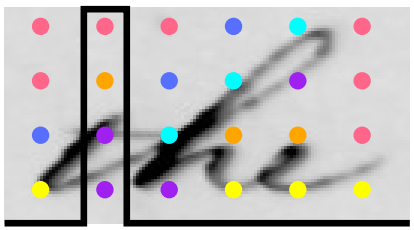
$$b_j(\boldsymbol{f}) = \sum_{k=1}^{|V|} c_{jk} f_k \quad \text{with} \quad \begin{array}{l} \boldsymbol{f} : \text{term vector} \\ V : \text{vis. voc.} \end{array}$$

Rothacker, L., Vajda, S., Fink, G. A.: *Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script*, In Proc. ICFHR, Bari, 2012.

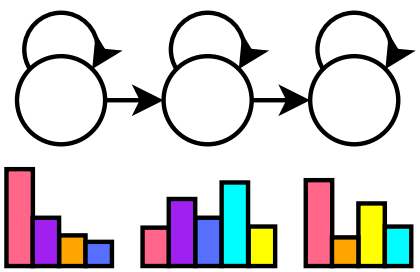
Bag-of-Features HMMs for QbE Word Spotting



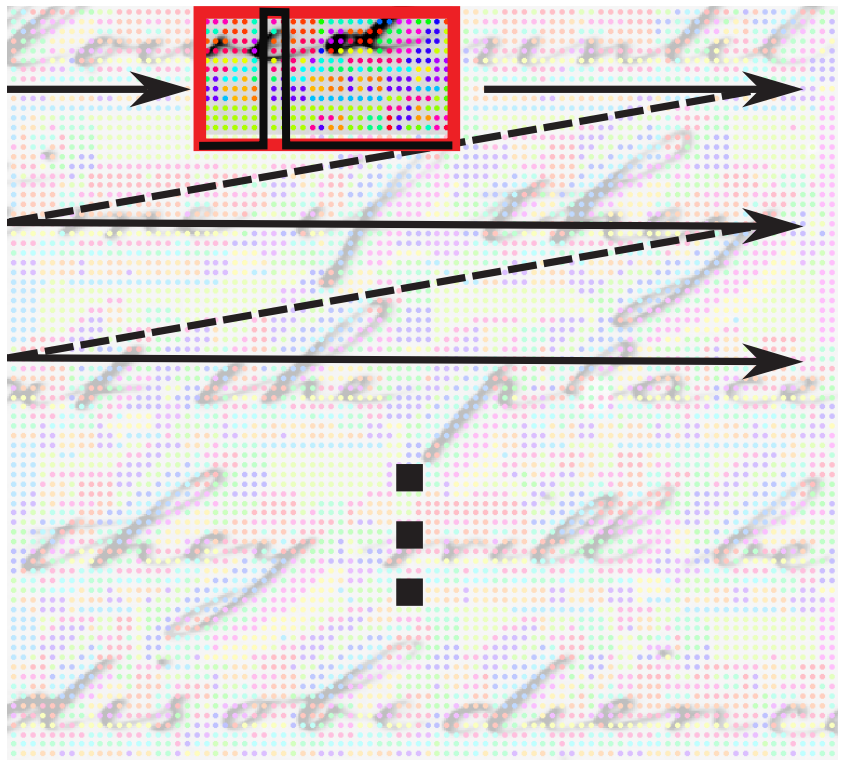
Query Word



Bag-of-Features Sequence
(sliding window)



Bag-of-Features
Hidden Markov Model

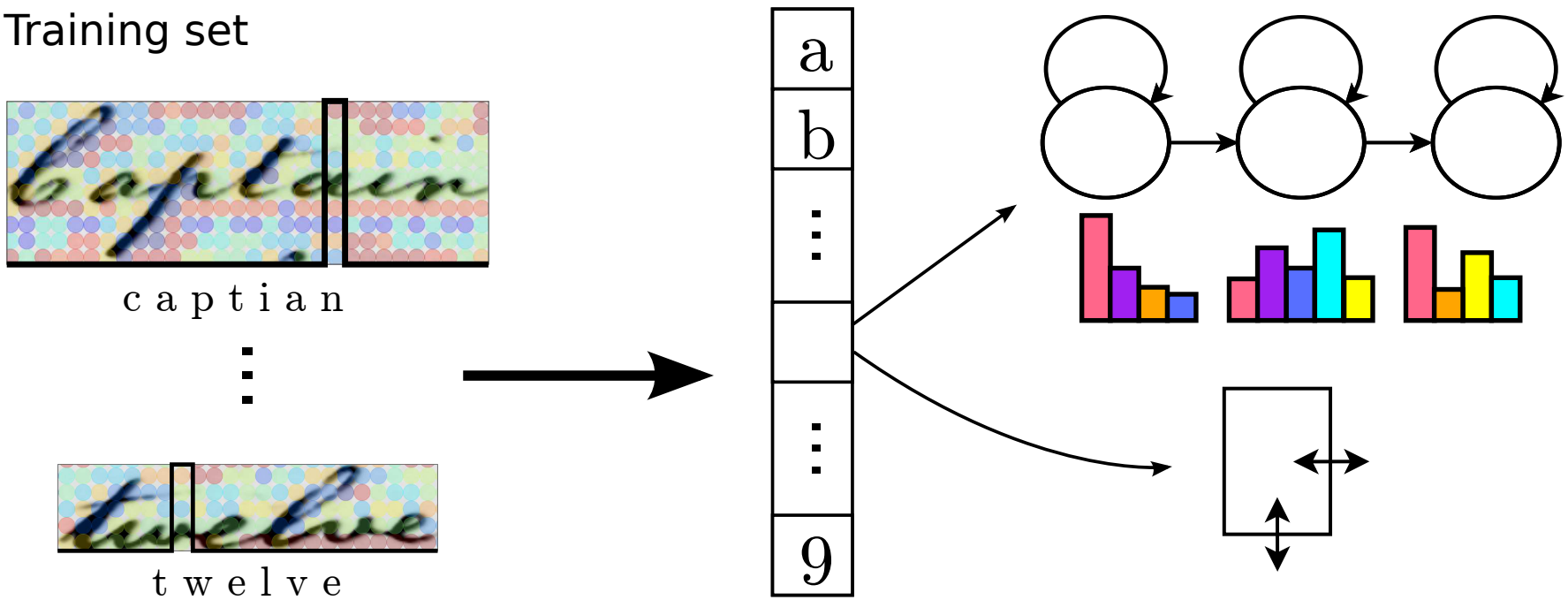


- Approach can be sped-up by intelligent patch pre-filtering!

Rothacker, L., Rusinol, M., Fink, G. A.: *Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents*, In Proc. Int. Conf. on Document Analysis and Recognition, Washington DC, USA, 2013.

Rothacker, L., Rusinol, M., Llados, J., Fink, G. A.: *A Two-Stage Approach to Segmentation-Free Query-by-Example Word Spotting*, manuscript cultures, 1(7), pages 47-57, 2014.

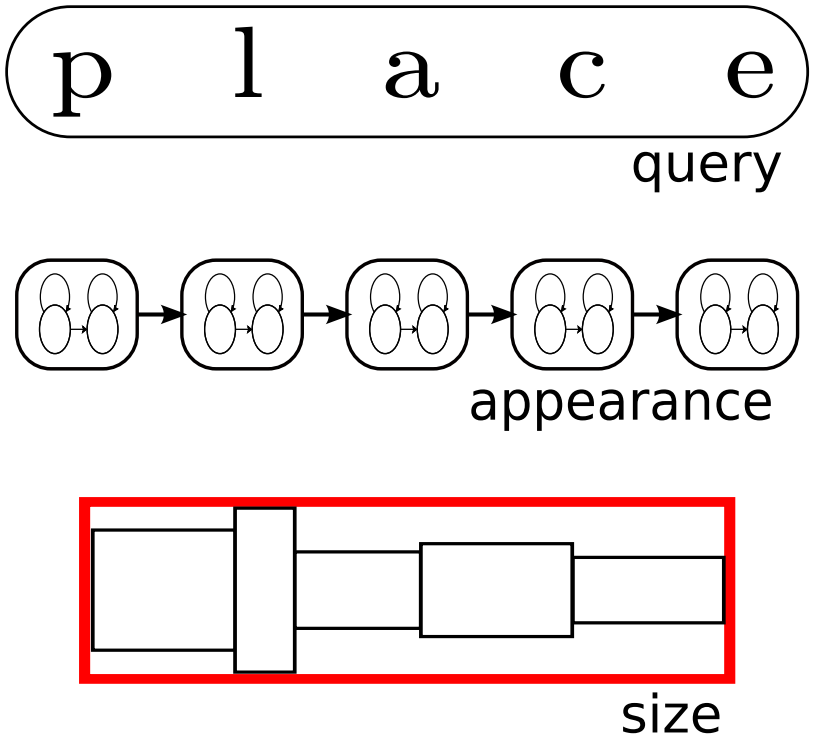
Bag-of-Features HMMs for QbS Word Spotting

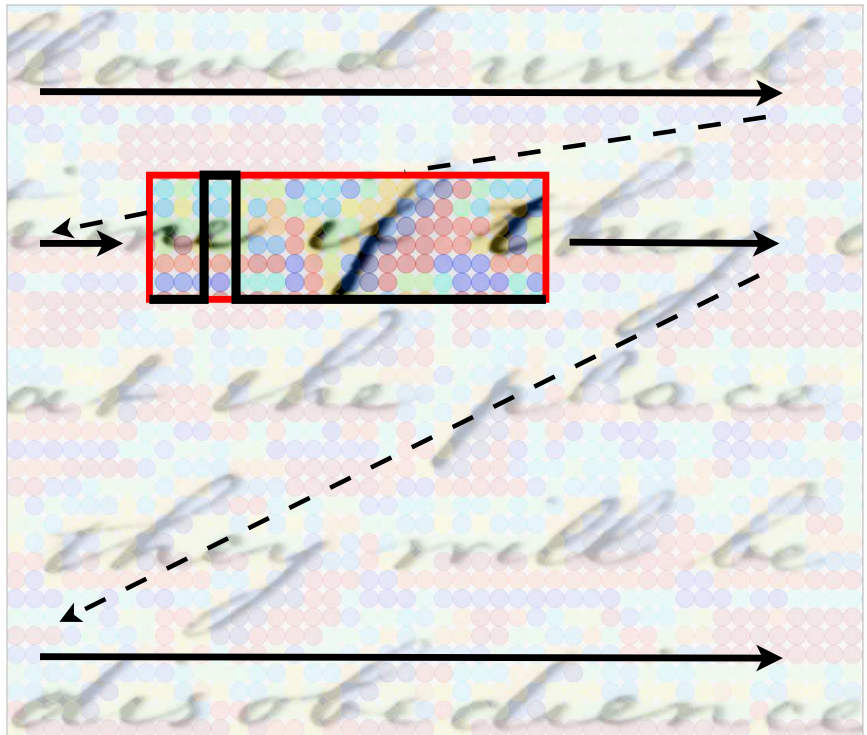


- Appearance model: Bag-of-Features HMMs (per character)
- Spatial size model: Width & height estimates

Rothacker, L., Fink, G. A.: *Segmentation-Free Query-by-String Word Spotting with Bag-of-Features HMMs*, Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015.

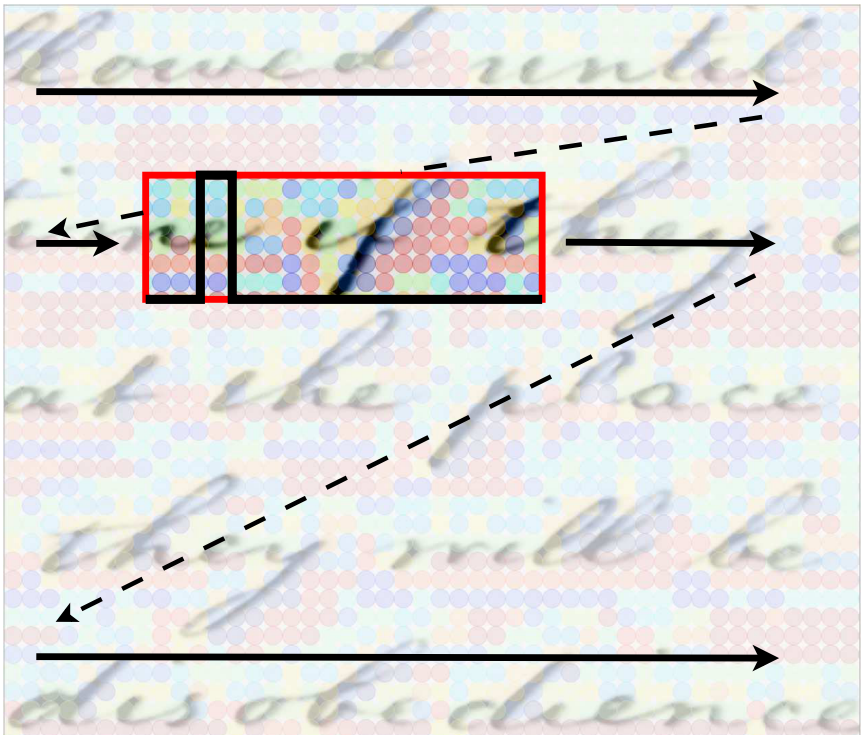
Bag-of-Features HMMs for QbS Word Spotting II

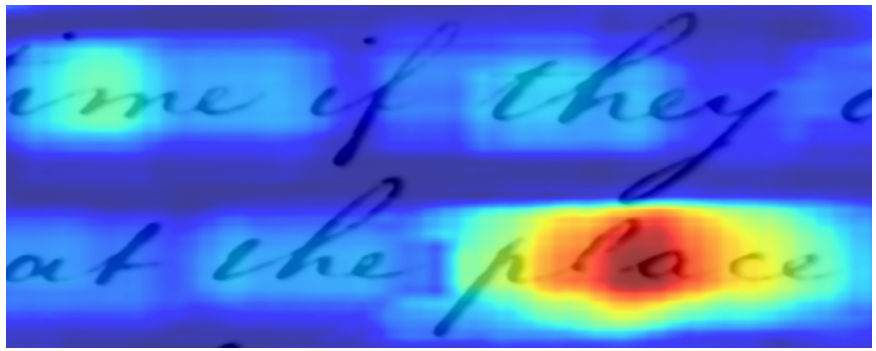


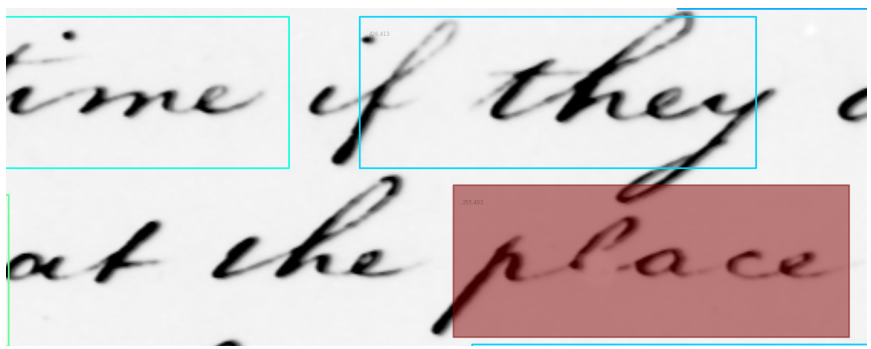


Rothacker, L., Fink, G. A.: *Segmentation-Free Query-by-String Word Spotting with Bag-of-Features HMMs*, Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015.

Bag-of-Features HMMs for QbS Word Spotting II





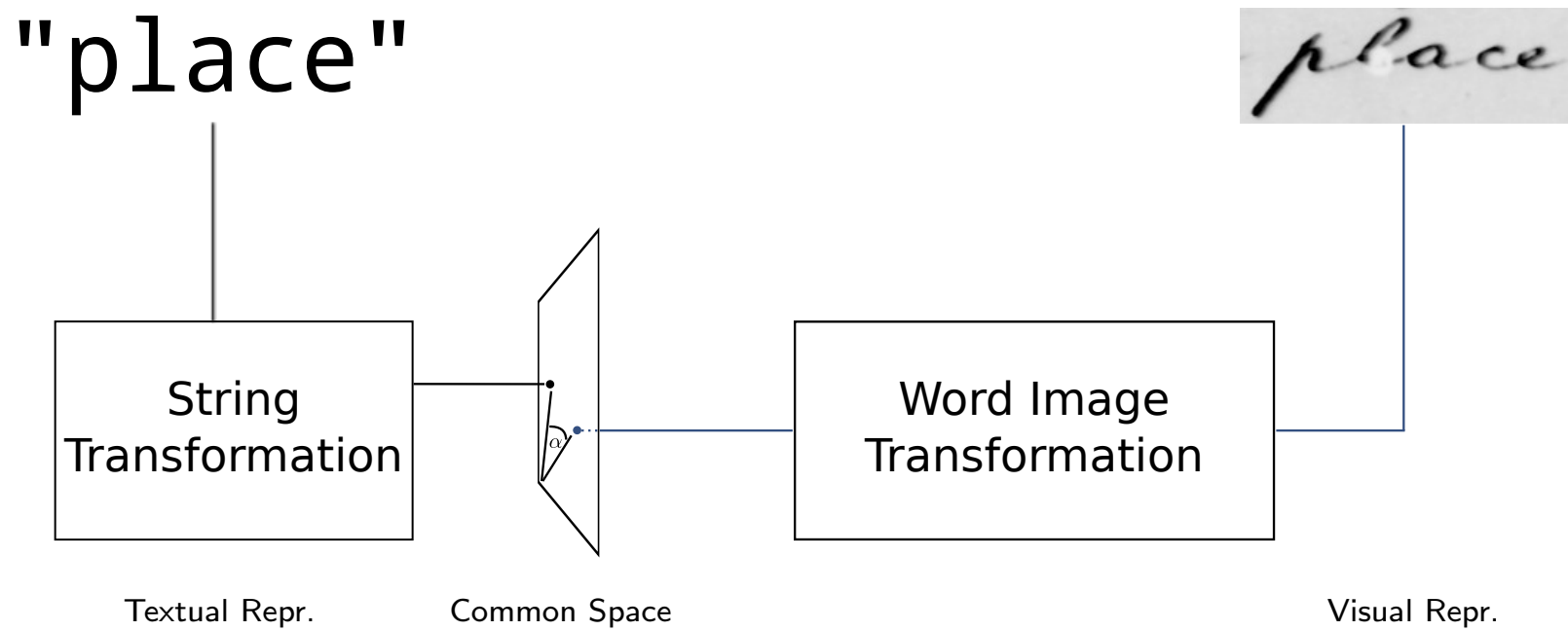


Rothacker, L., Fink, G. A.: *Segmentation-Free Query-by-String Word Spotting with Bag-of-Features HMMs*, Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015.

Subspace Representations for Word Spotting

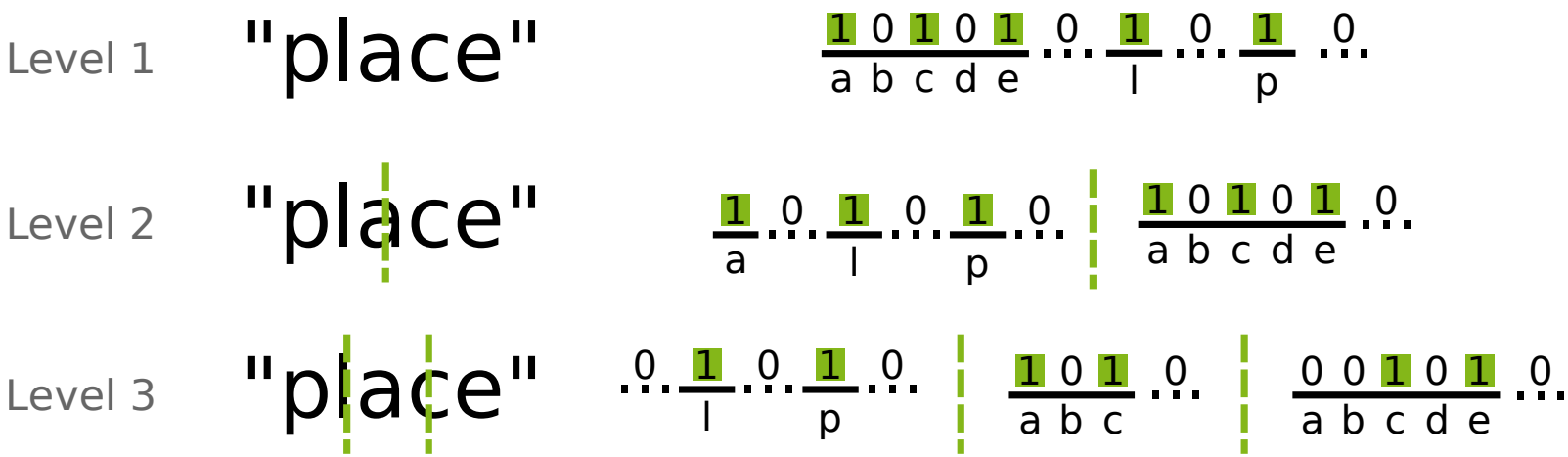
Idea: Project both textual and visual representation into a *common* space

Benefits: QbE and QbS are now a simple nearest neighbor search



J. Almazán, A. Gordo, A. Fornés and E. Valveny: Word Spotting and Recognition with Embedded Attributes, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, no. 12, pp. 2552-2566, 2014.

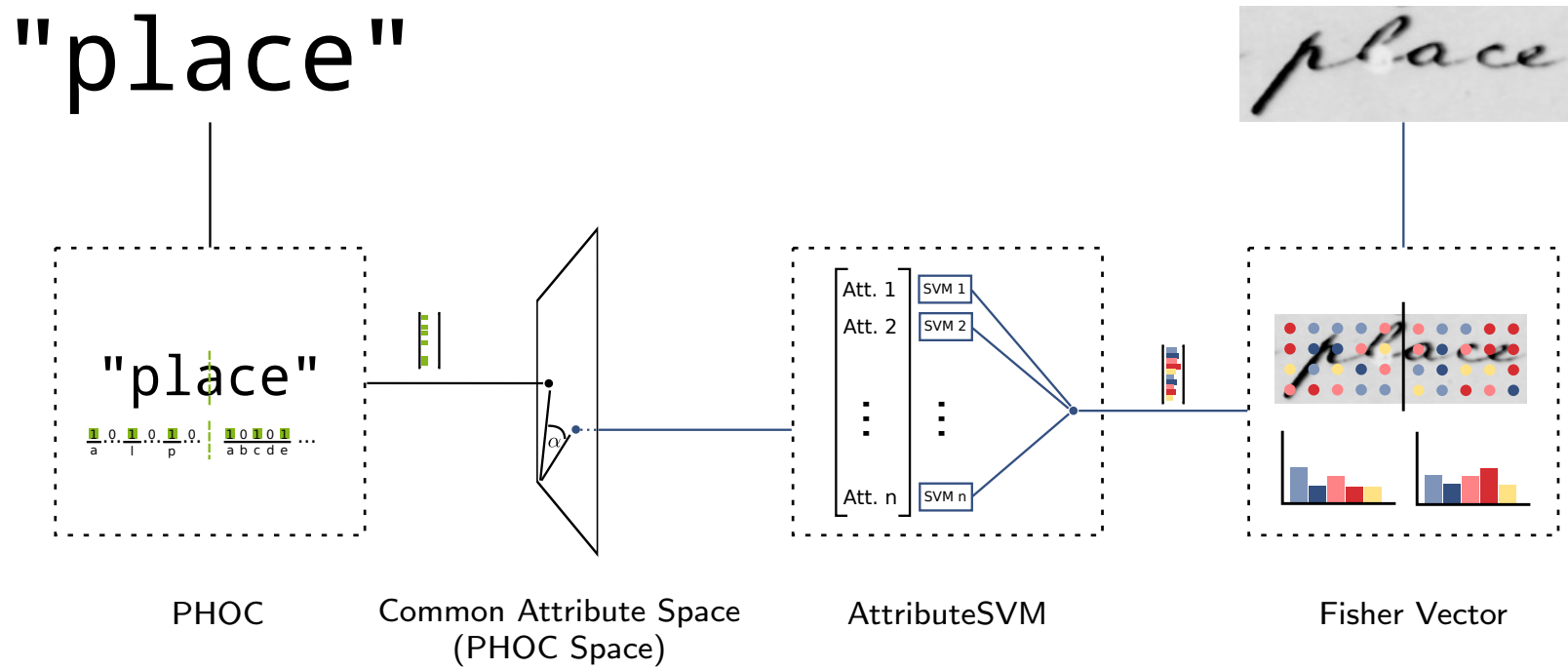
Pyramidal Histogram of Characters (PHOC)



- Concatenate histograms for all levels to form PHOC
- Levels used by Almazán *et al.*: 2,3,4 and 5
- 26 Characters + 10 Digits
- PHOC $\in \{0,1\}^{604}$

J. Almazán, A. Gordo, A. Fornés and E. Valveny: Word Spotting and Recognition with Embedded Attributes, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, no. 12, pp. 2552-2566, 2014.

Learning the PHOC representation



- AttributeSVM: ensemble of SVMs
- each SVM predicts the presence or absence of one element of the PHOC

J. Almazán, A. Gordo, A. Fornés and E. Valveny: Word Spotting and Recognition with Embedded Attributes, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, no. 12, pp. 2552-2566, 2014.

Pros and Cons of Methods so Far

Basic BoF-Based Models:

- ✓ (partly) Learned image features / representations
- ✓ No annotations required (unsupervised)
- ↯ Purely image-based (no categorial information)

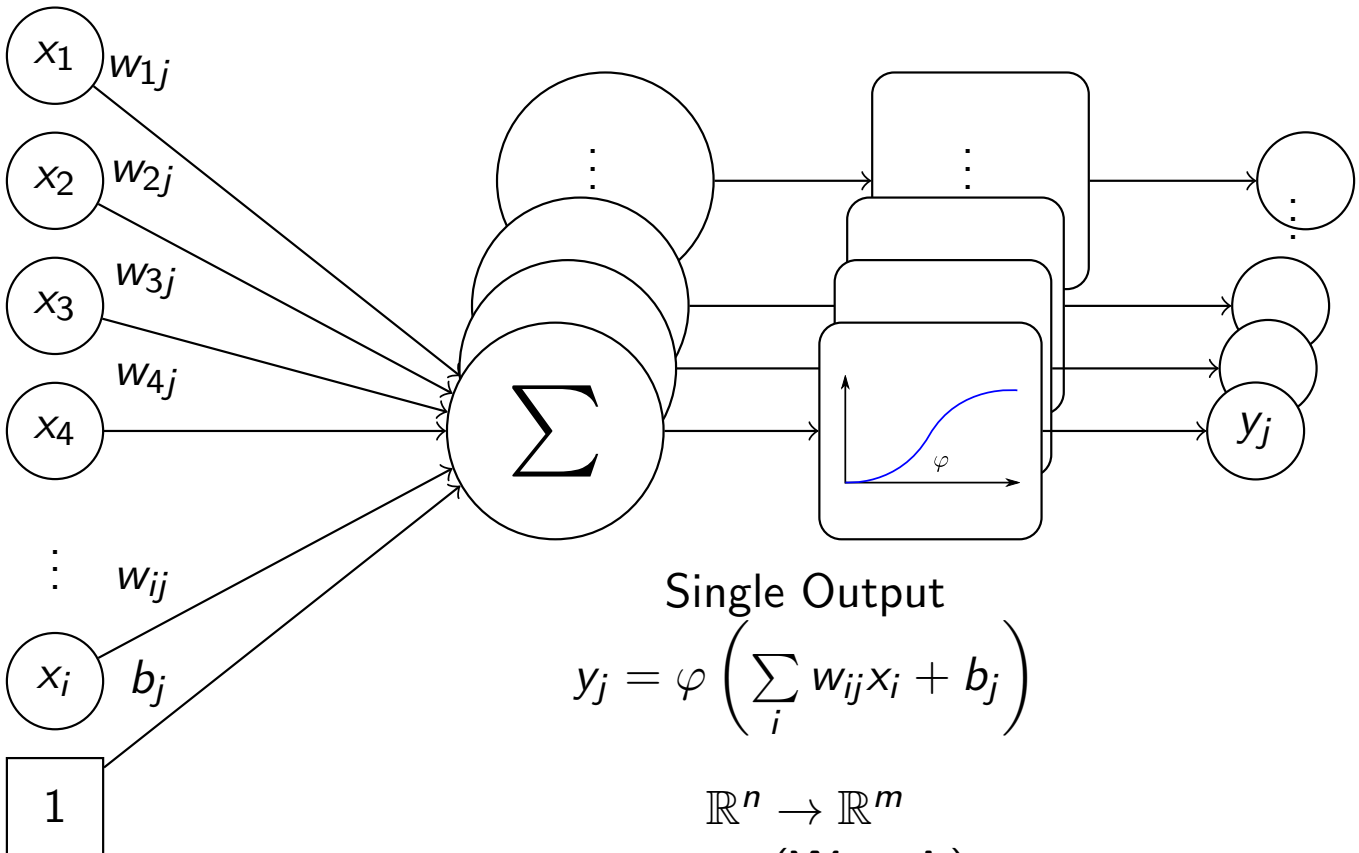
Advanced BoF-Based Models (BoF-HMMs, AttributeSVMs):

- ✓ Learned (categorial) models (supervised)
- ✓ Common representation of image appearance and categorial information (embedded attributes)
- ↯ Features and model learned separately

Desired: Integrated framework with overall / end-to-end optimization

Overview

The Perceptron



Single Output

$$y_j = \varphi\left(\sum_i w_{ij} x_i + b_j\right)$$

$$\mathbb{R}^n \rightarrow \mathbb{R}^m$$

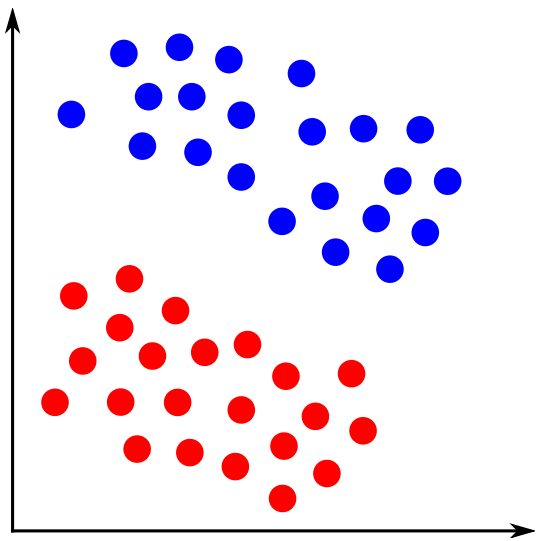
$$\mathbf{y} = \varphi\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

F. Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review, 65(6), 1958.

Capabilities of the Perceptron

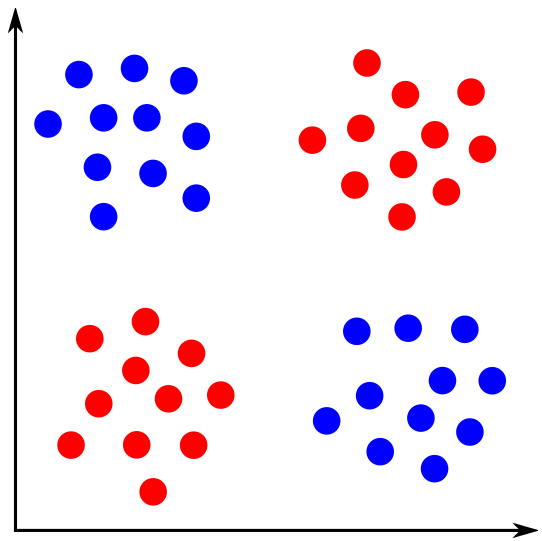
What a Perceptron can do:

Calssify two linearly separable classes



What a Perceptron can't do:

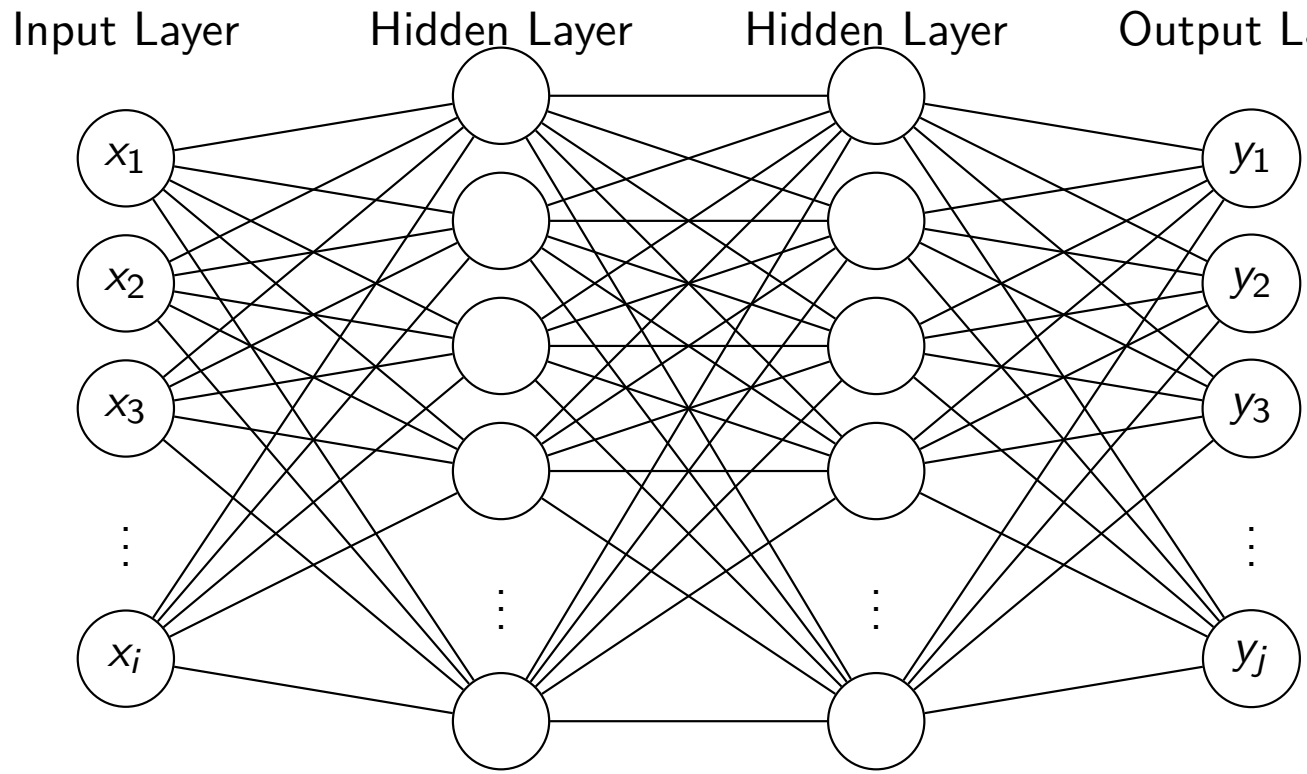
Classify two non-linearly separable classes (XOR-Problem)



Solution: Stack layers of Perceptrons

$\Rightarrow$ Multi Layer Perceptron

Multi Layer Perceptron (MLP)



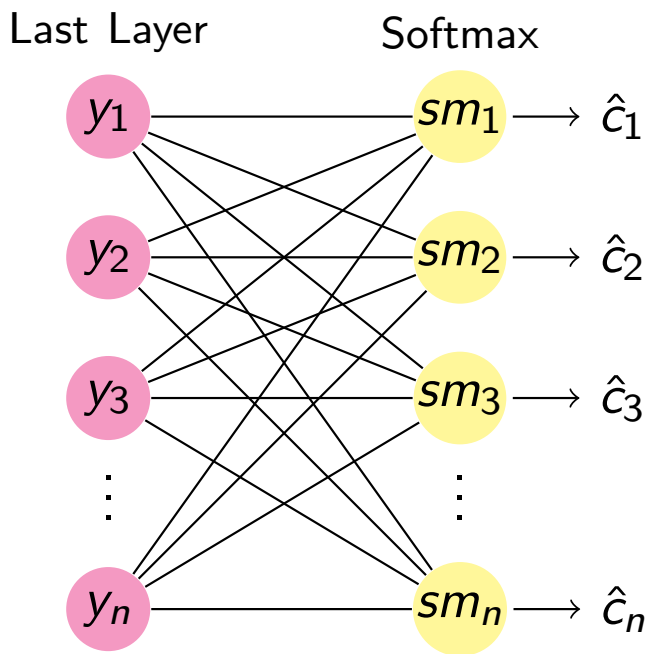
$$\mathbf{y} = \boldsymbol{f}^L\left(\boldsymbol{f}^{L-1}(\ldots \boldsymbol{f}^2\left(\boldsymbol{f}^1(\mathbf{x})\right)\right)$$

here: $\mathbf{y} = \mathbf{W}_{\text{out}} \cdot \varphi\left(\mathbf{W}_{h2} \cdot \varphi\left(\mathbf{W}_{h1}\mathbf{x} + b_{h1}\right) + b_{h2}\right) + b_{\text{out}}$

Note: Activation function is typically left out for last layer

Classifying with MLPs

- For classification, the output of the MLP is *usually* forwarded through a Softmax Function: $sm_i(\mathbf{y}) = \frac{e^{y_i}}{\sum_j e^{y_j}}$
- Softmax can be seen as an additional layer in the MLP
- sm_i is pseudo-probability for class c_i
- Predicted class: $\hat{c} = \max_i sm_i$

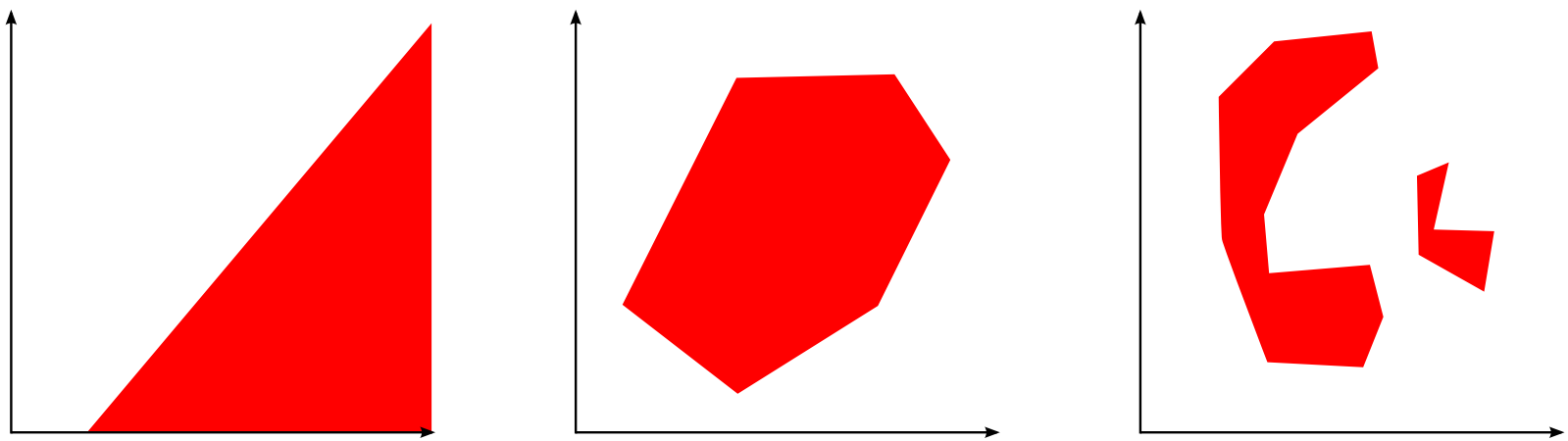


What an MLP Can Do!

... approximate any function (even with only 2 layers!)

[Hornik *et al.* 1989]

Interpretation with 3 layers (2 hidden, 1 output):



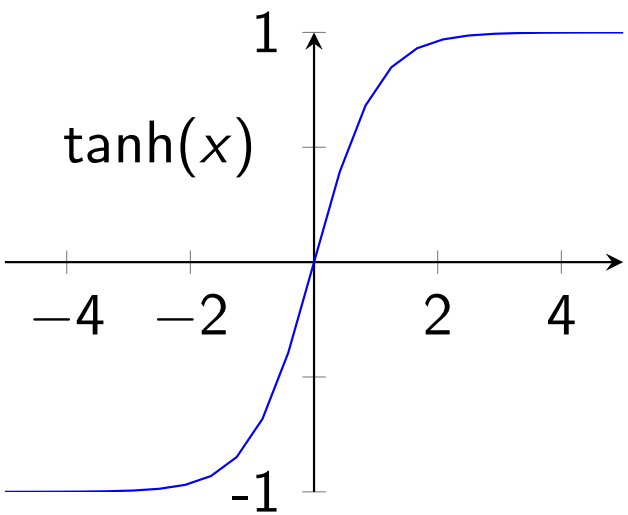
1. Layer: Halfspaces
2. Layer: Convex polyhedron
3. Layer: Multiple non-convex, non-connected polyhedra

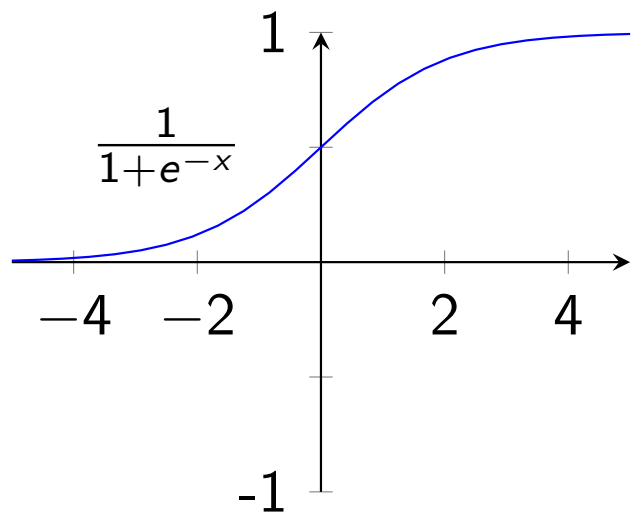
A Word on Activation Functions

- Activation functions are crucial for MLP
- Without non-linearities, an MLP implements a linear transform:

$$\mathbf{y} = \mathbf{W}_L \mathbf{W}_{L-1} \dots \mathbf{W}_2 \mathbf{W}_1 \mathbf{x} = \mathbf{W}' \mathbf{x}$$

Classic Activation Functions: sigmoidal shape ("threshold-like")





Note: With $\phi_s(x) = \frac{1}{1+e^{(-x)}}$ we have $\frac{\partial \phi_s(x)}{\partial x} = \phi_s(x)\,(1 - \phi_s(x))$.

Training an MLP

How to determine weights such that desired functition is performed?

Basic Idea: Compare (computed) output of MLP

$$\hat{\boldsymbol{y}} = \boldsymbol{f}^L\left(\boldsymbol{f}^{L-1}(\ldots \boldsymbol{f}^2\left(\boldsymbol{f}^1\left(\boldsymbol{x}\right)\right)\right)$$

to *desired* (ideal) output $\boldsymbol{y}$ and update weights such that $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ become more similar.

Comparison requires *loss function* that evaluates similarity of $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$:

- *Mean Square Error* (MSE):

$$\epsilon_{\text{MSE}} = \frac{1}{2} \cdot \sum_i (y_i - \hat{y}_i)^2$$

- *Cross-Entropy* (in comb. w. Softmax):

$$\epsilon_{\text{CE}} = -\sum_i y_i \log \hat{y}_i$$

Training an MLP II

How to update weights in order to reduce loss?

Compute gradient of loss wrt. the weights:

$$\frac{\partial \epsilon}{\partial w_{ij}^{l}}$$

Update weights in the negative direction of the gradient:

$$w_{ij}^{l} \leftarrow w_{ij}^{l} - \beta \cdot \frac{\partial \epsilon}{\partial w_{ij}^{l}}$$

$\Longrightarrow$ Training the network $\hat{=}$ Gradient Descent

Note: *Learning rate* β controls step size!

Training an MLP: Error Backpropagation

How to compute gradient of loss wrt. network weights?

Define computation of outputs of neurons per layer[1]:

$$f_j^l = \phi(g_j^l) = \phi(\sum_i w_{ij}^l f_i^{l-1})$$

Rewrite gradient applying chain rule:

$$\frac{\partial \epsilon}{\partial w_{ij}^l} = \frac{\partial \epsilon}{\partial f_j^l} \cdot \frac{\partial f_j^l}{\partial g_j^l} \cdot \frac{\partial g_j^l}{\partial w_{ij}^l}$$

$\Rightarrow$ Evaluation *straight forward* for final layer (using ϵ_{MSE} and ϕ_s):

$$\frac{\partial \epsilon}{\partial w_{ij}^L} = -(y_j - f_j^L) \cdot f_j^L(1 - f_j^L) \cdot f_i^{L-1}$$

[1] ignoring bias for siplicity

Training an MLP: Error Backpropagation II

How to evaluate gradient for inner weights?

Hidden layers: define gradient based on "local error" $\delta_j^k = \frac{\partial \epsilon}{\partial g_j^k}$:

$$\frac{\partial \epsilon}{\partial w_{ij}^k} = \frac{\partial \epsilon}{\partial \boldsymbol{f}^L} \cdot \frac{\partial \boldsymbol{f}^L}{\partial \boldsymbol{f}^{L-1}} \cdot \ldots \cdot \frac{\partial \boldsymbol{f}^{k+1}}{\partial \boldsymbol{f}^k} \cdot \frac{\partial \boldsymbol{f}^k}{\partial g_j^k} \cdot \frac{\partial g_j^k}{\partial w_{ij}^k} = \delta_j^k \cdot \frac{\partial g_j^k}{\partial w_{ij}^k}$$

Exploit *generalized chain rule*: For the gradient we obtain:

$$\frac{\partial \epsilon}{\partial f_j^l} = \sum_{k=1} \underbrace{\frac{\partial \epsilon}{\partial f_k^{l+1}} \cdot \frac{\partial f_k^{l+1}}{\partial g_k^{l+1}}}_{\delta_k^{l+1}} \cdot \underbrace{\frac{\partial g_k^{l+1}}{\partial f_j^l}}_{w_{jk}^{l+1}} = \sum_{k=1} w_{jk}^{l+1}\, \delta_k^{l+1}$$

$$\Rightarrow \qquad \delta_j^l = \frac{\partial \epsilon}{\partial f_j^l} \frac{\partial f_j^l}{\partial g_j^l} = \left\{ \sum_{k=1} w_{jk}^{l+1}\, \delta_k^{l+1} \right\} \cdot \frac{\partial f_j^l}{\partial g_j^l}$$

Local error can be computed *backward* through the network

$\Rightarrow$ Error Backpropagation

Stochastic Gradient Descent (SGD)

Gradient can be computed ...

(1) for complete training data (avg.)

(2) "online" for every new sample.

But *neither* works well!

⇒ Training takes forever!

⇒ Gradient is extremely noisy!

Solution:

Compute estimate of gradient for a small sub-set of the training data (so-called "mini batch")

Note: Samples are drawn *randomly* from the training set!

⇒ Gradient estimate is *stochastic*!

Stochastic Gradient Descent (SGD) II

Disadvantage of mini-batch processing:
Gradient is still somewhat noisy.

Improved Solution:
Introduce so-called *momentum* η ($0 \ll \eta < 1$):

$$w_{ij}^{(t+1)} \leftarrow w_{ij}^{(t)} + \beta \left[(1-\eta) \Delta w_{ij}^{(t+1)} + \eta \Delta w_{ij}^{(t)} \right]$$

Note: Computes smoothed sequence of gradient estimates
(i.e. sliding average with exponential decay).

Stochastic Gradient Descent (SGD) III

Warning:

In the widely used Caffe toolbox, the gradient is scaled with the actual loss ϵ:

$$w_{ij}^{l} \leftarrow w_{ij}^{l} - \beta \frac{\partial \epsilon}{\partial w_{ij}^{l}} \cdot \epsilon$$

Result: Learning rate is always *dynamic*.

Rationale:

Loss is big/small $\hat{=}$ Optimization is far from/near to solution

$\Rightarrow$ perform big/small steps in gradient direction.

Classifying Images with Neural Networks

Problem:

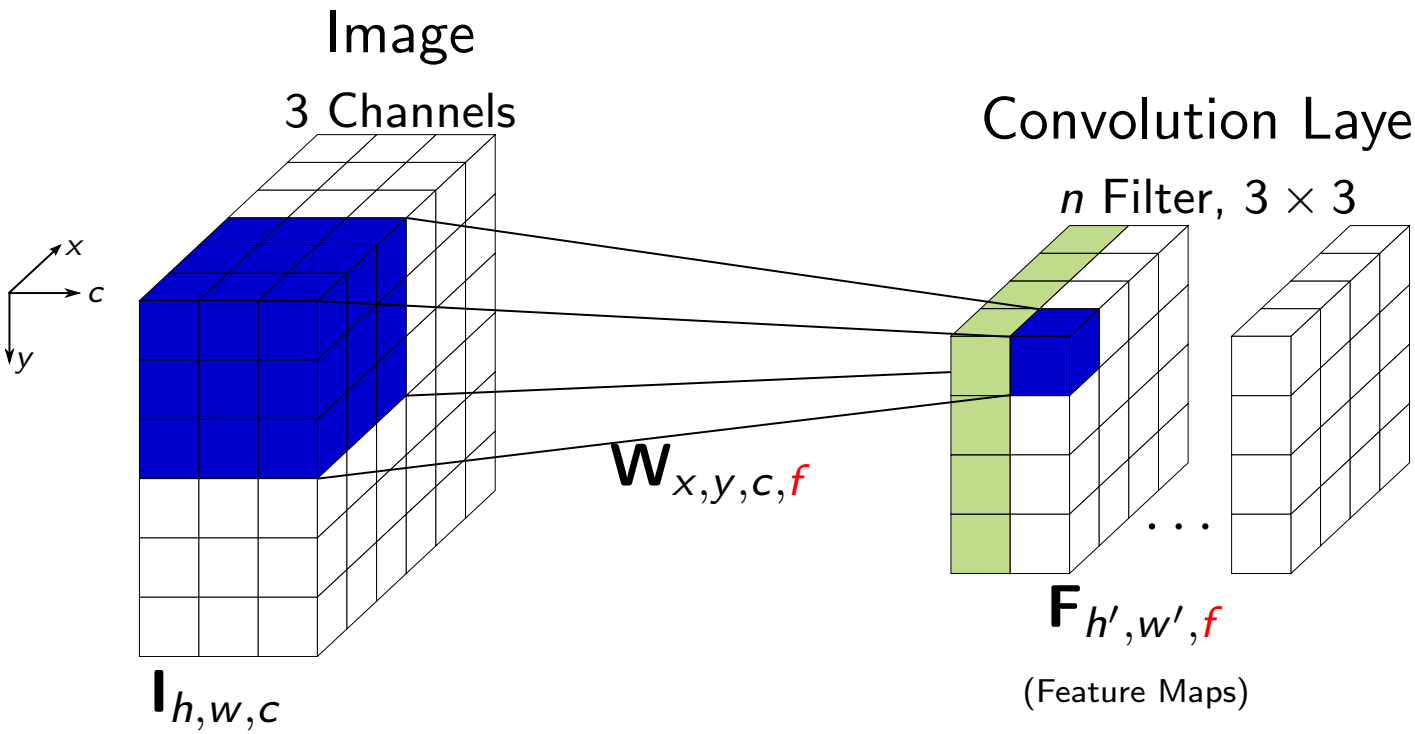
- Using MLPs for image classification is only possible for very small images (e.g. 28×28 pixels)
- Number of weights would explode for bigger images

Example: RGB Image of 224×224 pixels,
first hidden MLP layer has 768 neurons (small layer):
$224 \cdot 224 \cdot 3 \cdot 768 \approx 10^8$ weights in the first layer (441 MB)

Solution: Don't use fully connected layer but rather apply a small number of weights at all possible locations in the image

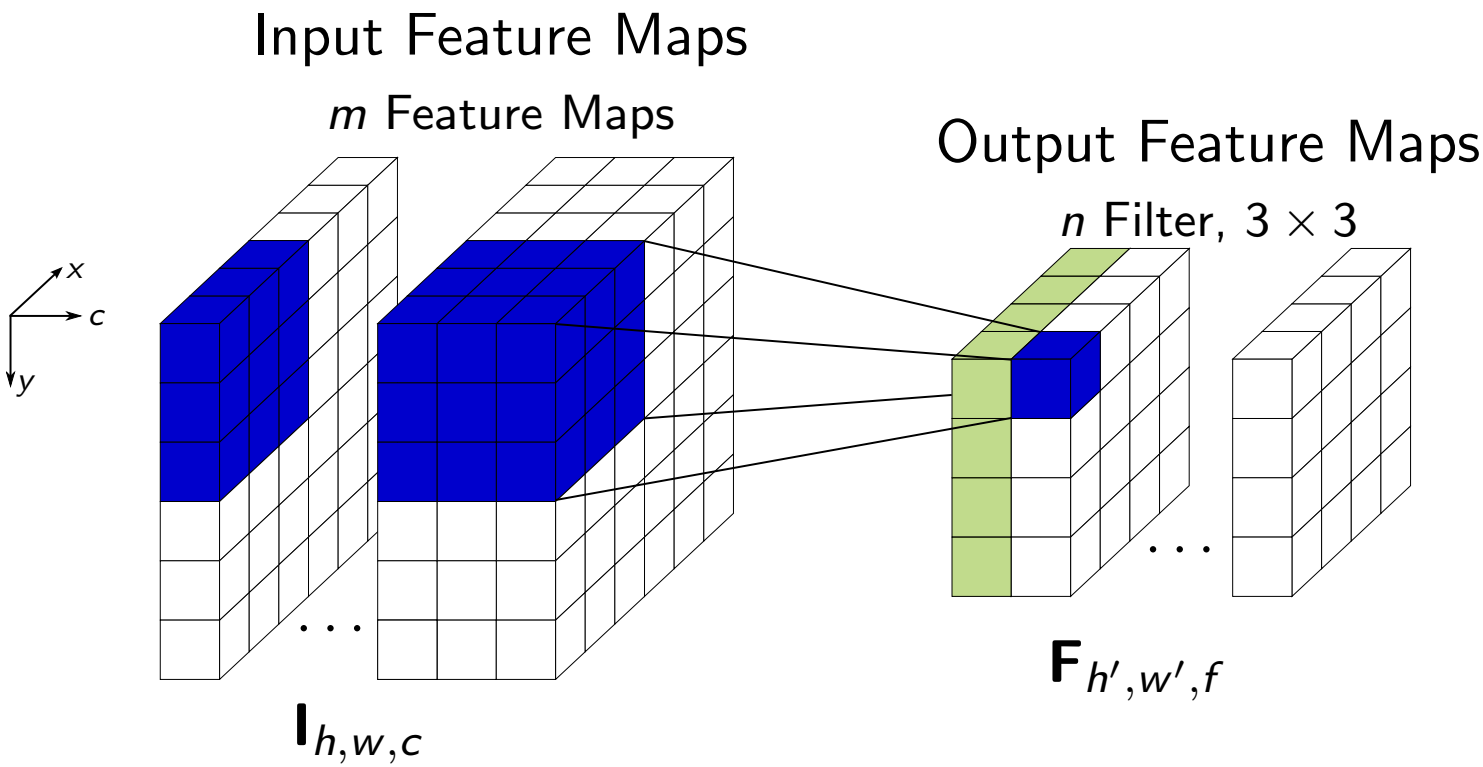
$\Rightarrow$ Convolutional Layer

Convolutional Layer

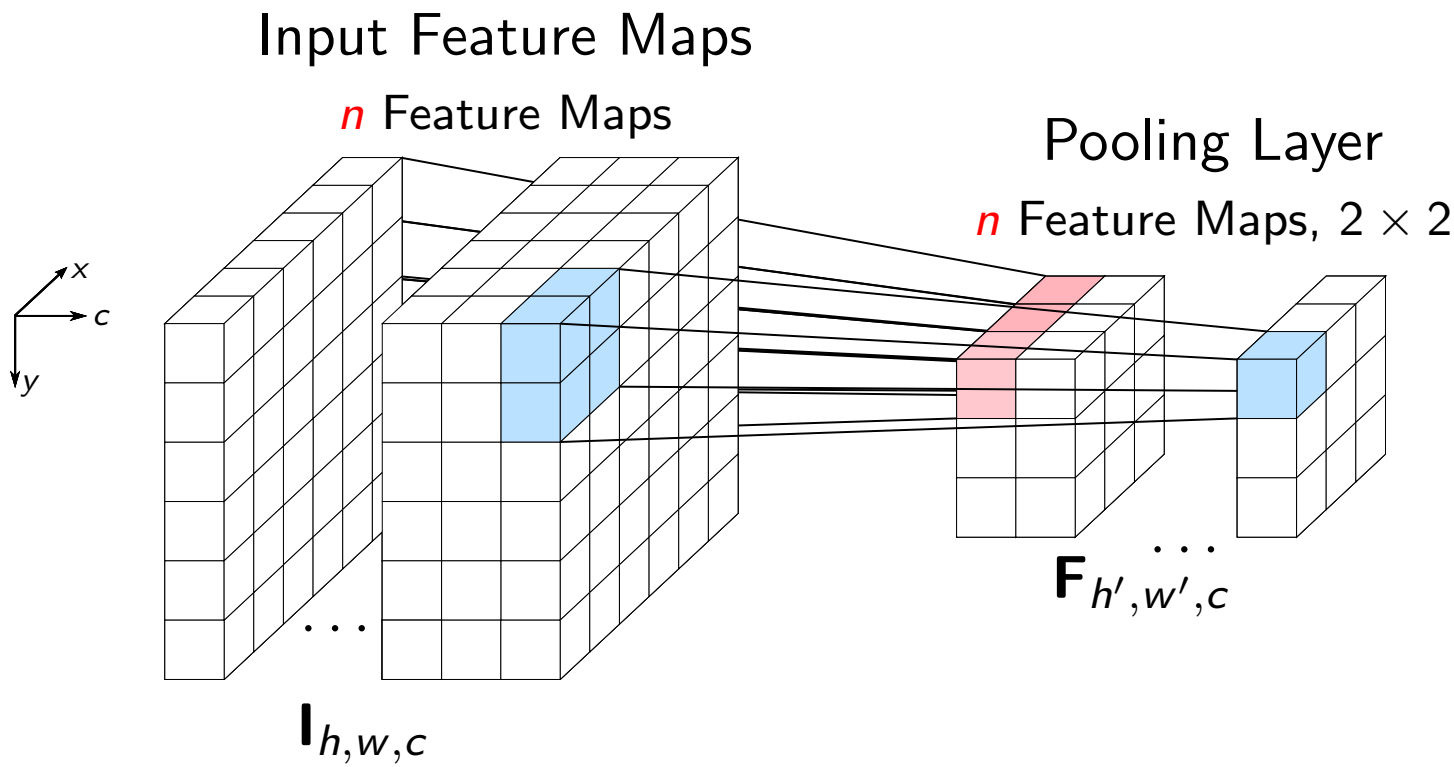


$$\mathbf{F}_{x,y,f} = \varphi\left(\sum_{c=1}^{K}\sum_{i=1}^{3}\sum_{j=1}^{3}\mathbf{W}_{i,j,c,f}\cdot\mathbf{I}_{x+i,y+j,c} + b_f\right)$$

Cascade of Convolutional Layers

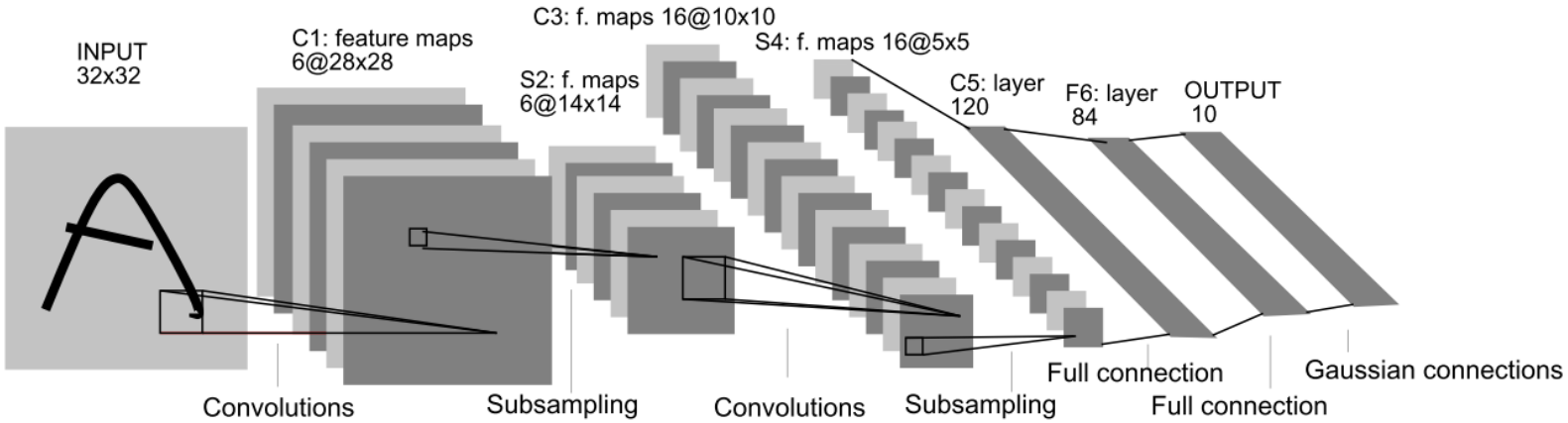


Pooling Layer



$$\mathbf{F}_{x,y,f} = \max_{i,j} \mathbf{I}_{x+i,y+j,f}$$

LeNet



(Source: [LeCun *et al.*, 1990])

- LeNet predicts one of 10 character classes for a given input image
- Subsampling = Pooling Layer
- Gaussian Connections = FC Layer + Euclidean Loss

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L.D. Jackel: Handwritten Digit Recognition with a Back-Propagation Network, Neural Information Processing Systems, pp. 396–404, 1990.

Deep Learning

In general: Deeper network architectures perform better than shallower ones for vision tasks

Important: Only empirical evidence (no theoretical proofs)

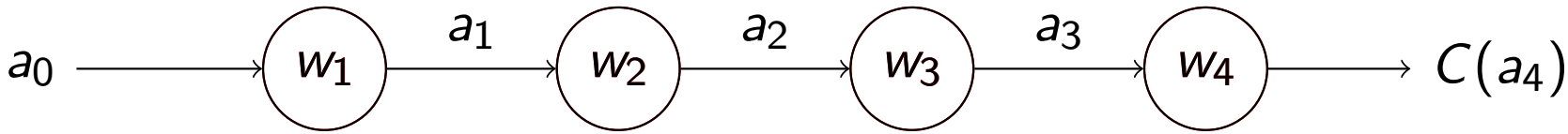
Technically: Deeper means more layers, not a deeper understanding

Even with high computation power and large datasets, Deep Learning did not really pick up until 2012!

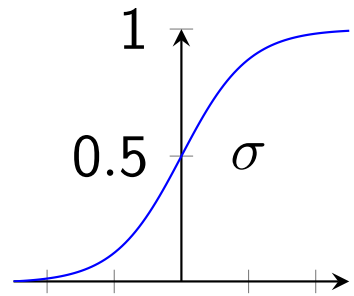
Why? Vanishing Gradient Problem

Vanishing Gradient Problem

Four neuron network, 1D input, 1D output



$$z_i = w_i a_{i-1} + b_i \qquad a_i = \sigma(z_i)$$

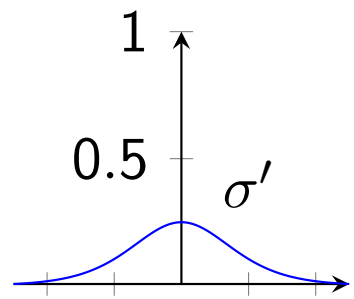


$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial z_4}\frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

$$= \sigma'(z_4) w_4 \cdot \sigma'(z_3) w_3 \cdot \sigma'(z_2) w_2 \cdot \sigma'(z_1) a_0$$

$$= \sigma'(z_4)\sigma'(z_3)\sigma'(z_2)\sigma'(z_1) \cdot w_4 w_3 w_2 a_0$$

$$= \underbrace{\sigma'(z_4)\sigma'(z_3)\sigma'(z_2)\sigma'(z_1)}_{\leq \frac{1}{4}\cdot\frac{1}{4}\cdot\frac{1}{4}\cdot\frac{1}{4}} \cdot w_4 w_3 w_2 a_0$$



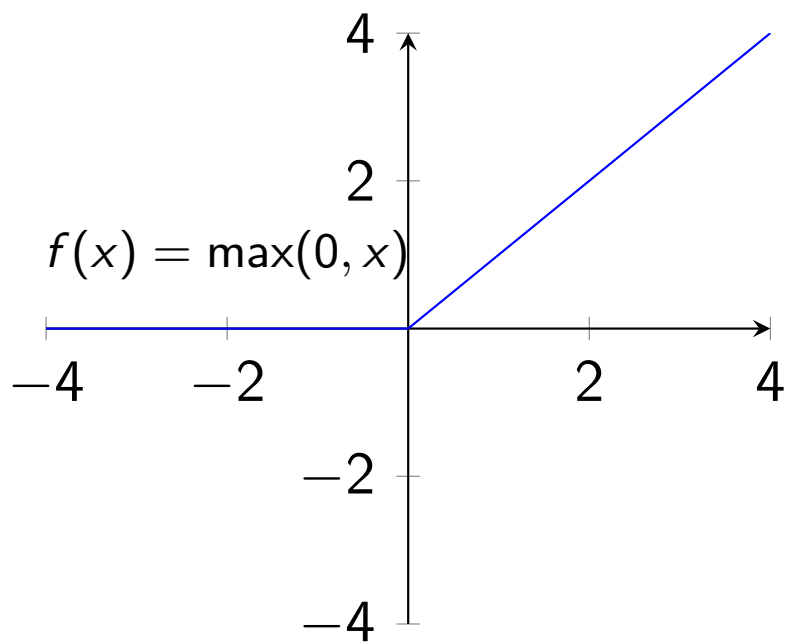
$$\leq \frac{1}{256} \cdot w_4 w_3 w_2 a_0$$

Vanishing Gradient Problem

- Derivative of sigmoidal activation functions < 1
- Exponential decay of gradient magnitude

Desirable: Activation function with derivative $= 1$ but non-linear ($> 1 =$ exploding gradient)

Solution: Rectified Linear Unit (ReLU) [Glorot & Bengio 2010]



How to Get Along With Limited Training Data?

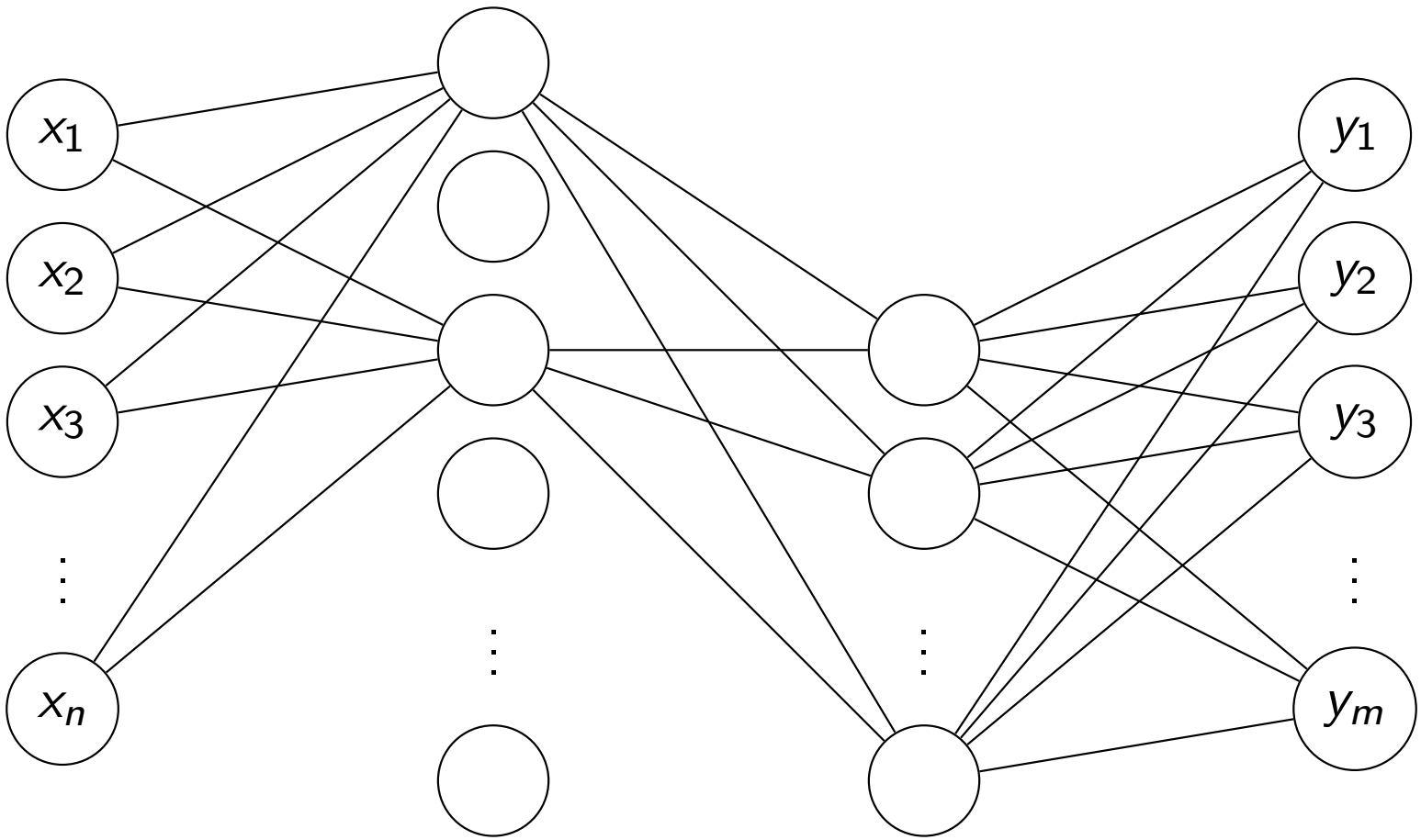
Problem: CNNs easily contain billions of parameters (weights)!
$\Rightarrow$ Could easily learn training samples "by heart".

Solution: Apply *Regularization* during training

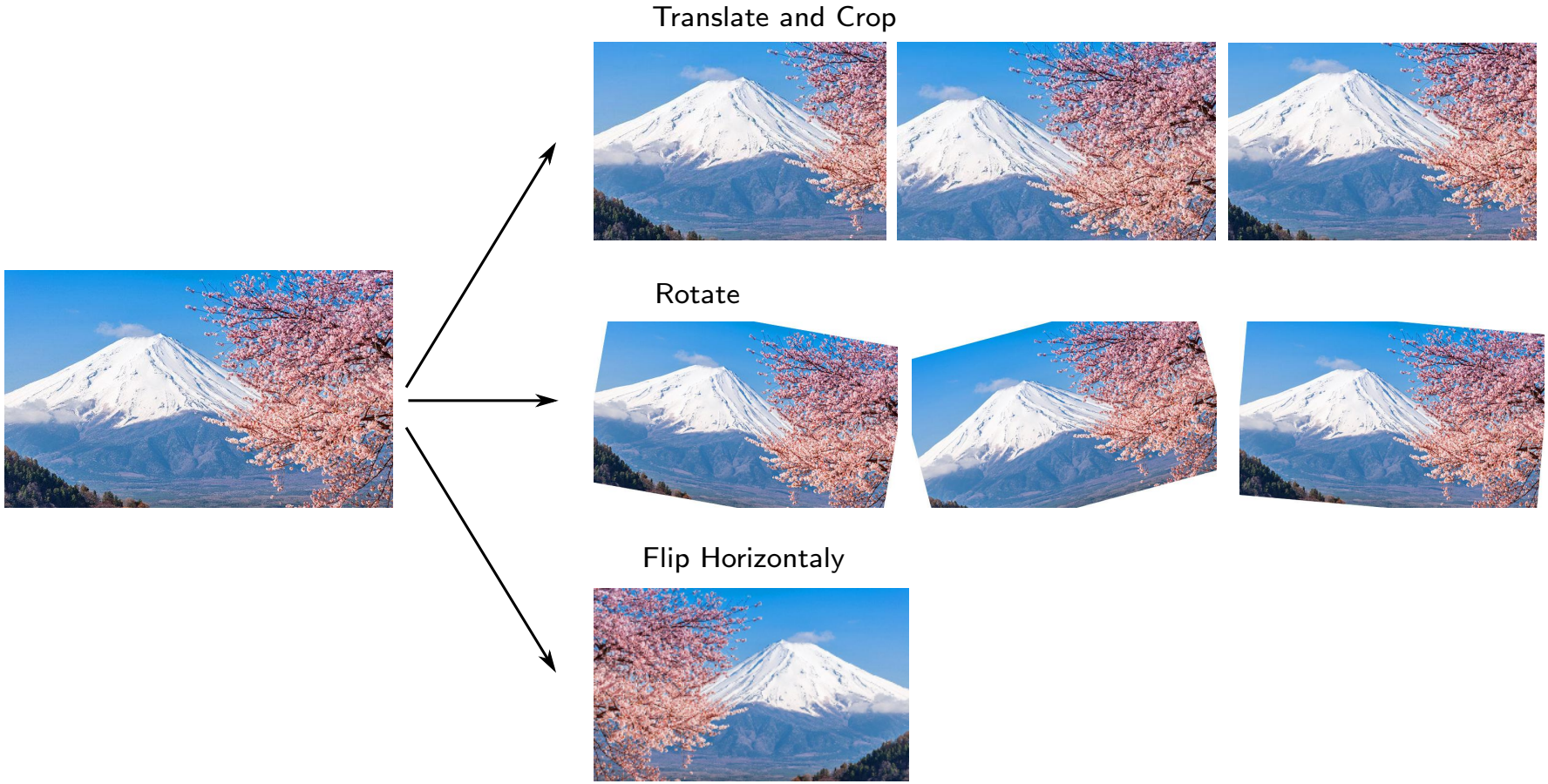
Fundamental Techniques:

- *Convolutional layers*
- *Dropout*
  Randomly set outputs of neurons to zero
  (usually 50% of fully connected layers)
- *Data Agumentation:*
  Generate new, slightly different training samples from existing ones by certain transforms
  (e.g. slight translations, rotations, ...)

Dropout

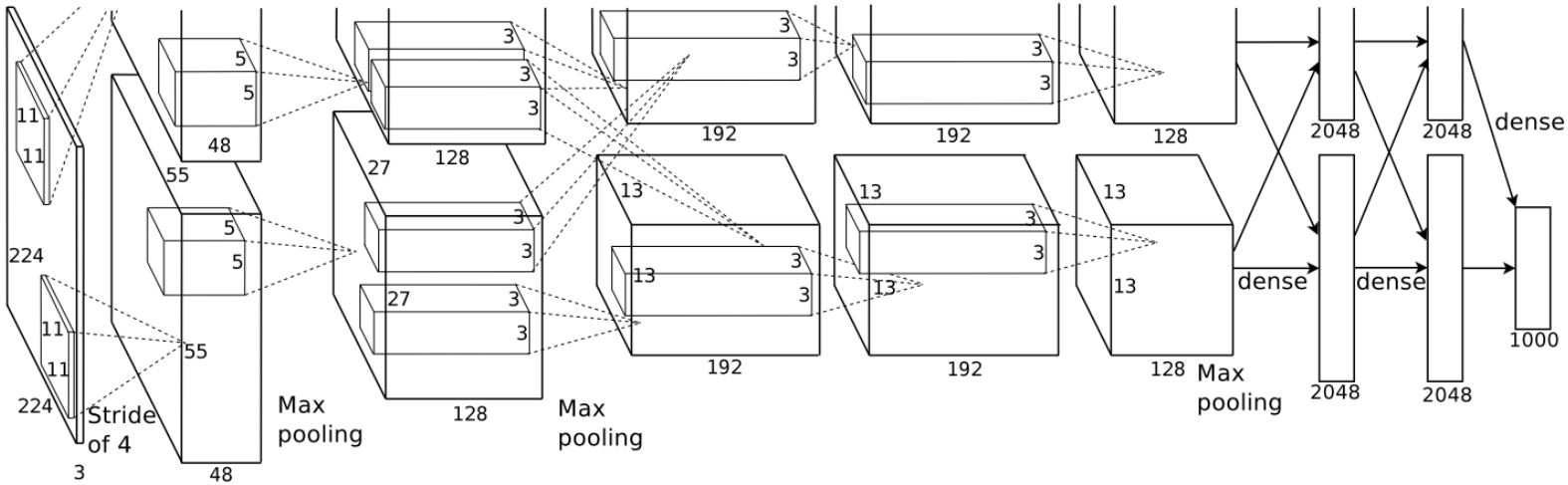


Data Augmentation



Note: Usually different augmentation techniques are mixed to create a single augmented image

Well-known Deep Learning Architectures: AlexNet

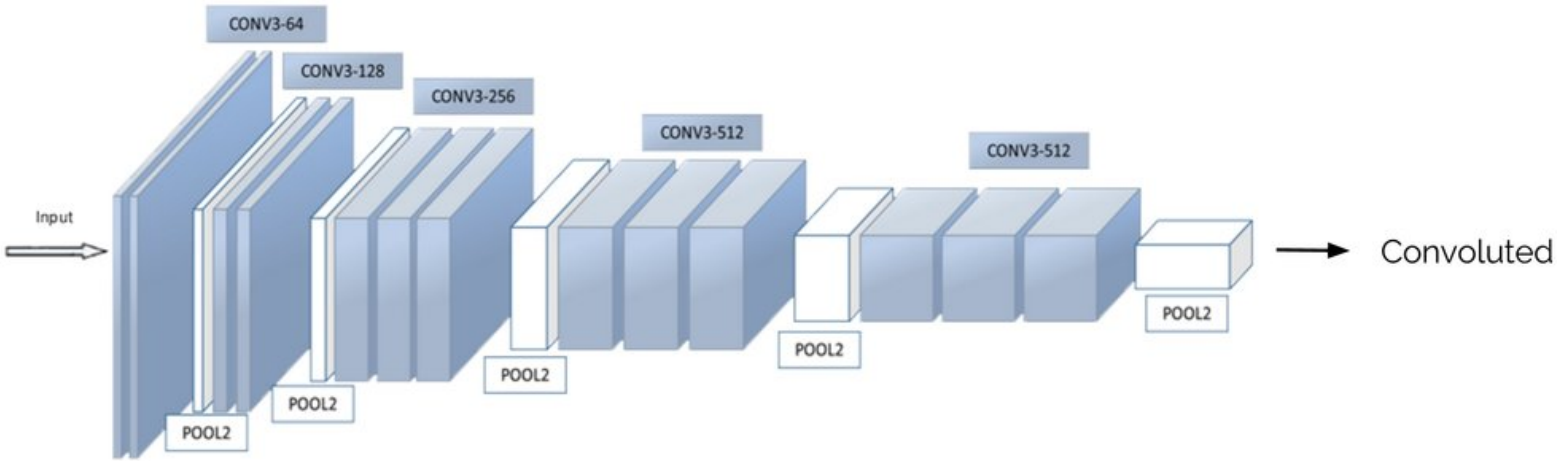


(Source: [Krizhevsky *et al.*, 2012])

- CNN which kicked off the current Deep Learning hype
- Architecture similar to LeNet but more layers/parameters
- Trained on two graphic cards for over a week on ImageNet

A. Krizhevsky, I. Sutskever, G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks, Neural Information Processing Systems, pp. 1097–1105, 2012.

Well-known Deep Learning Architectures: VGGNet



(Source: http://html.scirp.org/)

- First CNN to use only 3×3 convolutions (standard for current CNNs)
- Low number of filters in the early layers, high number of filters in the later layers
- Anytime pooling is applied, the number of filters is doubled

K. Simonyan, A. Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv, 2014.

Summary Deep Learning

- Deep Learning is a hot topic in Computer Vision *and* Document Image Analsis

  ⇒ *You should know about it!*

- Deep Neural Networks give state of the art results on many benchmarks across tasks and domains

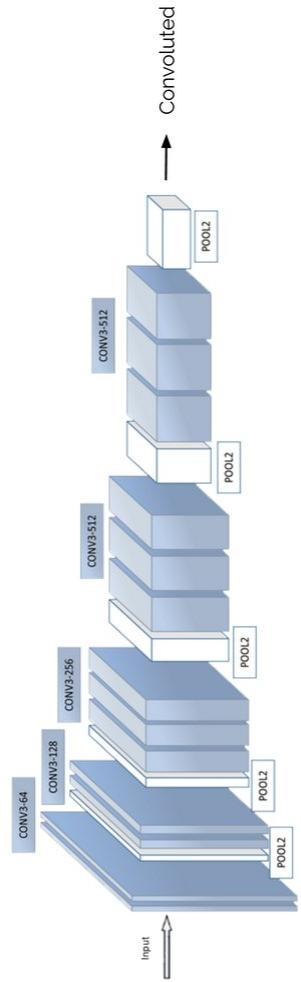
  ⇒ *Only empirical evicende for superiority!*

- CNNs are the most popular deep networks

  ⇒ *Especially suitable for (document) images!*

- Toolboxes (e.g. Caffe) make it easy to apply Deep Learning to ones own problems

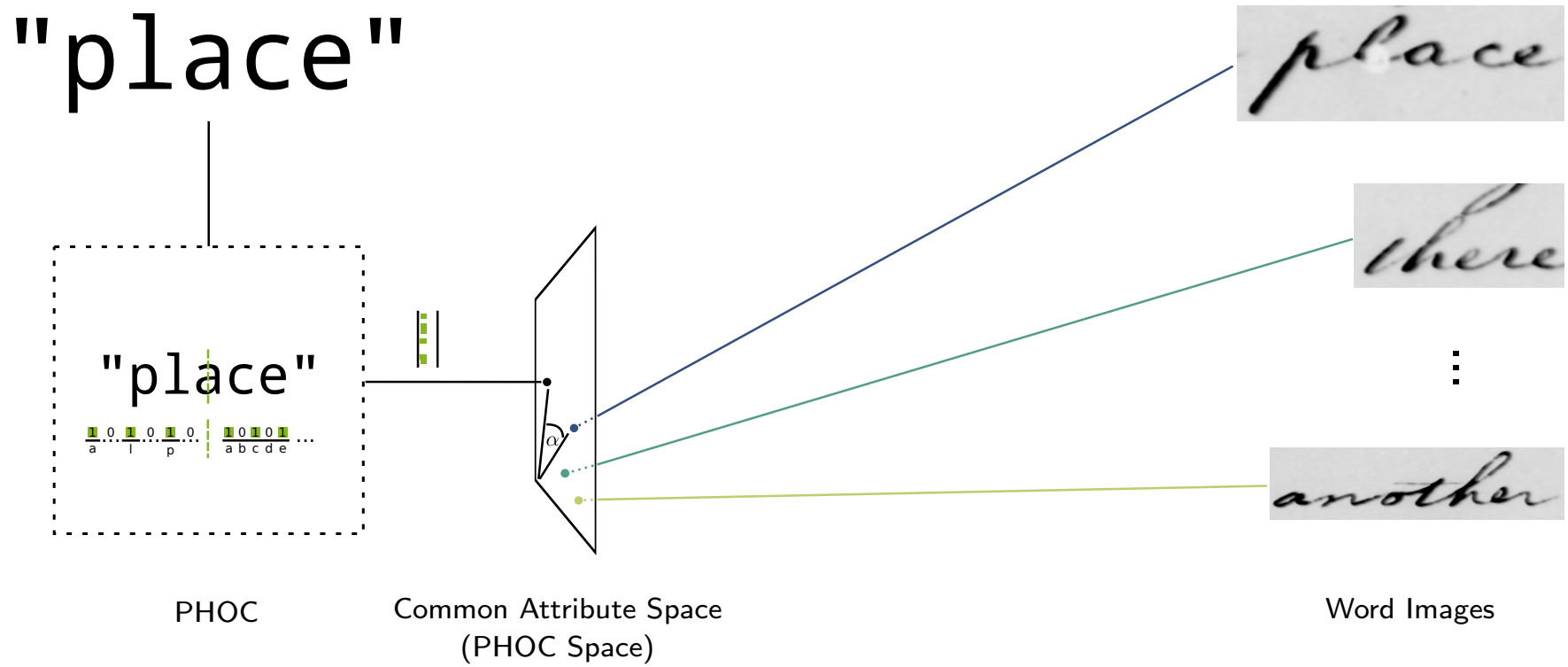
  ⇒ *Beware of blindly using them!*



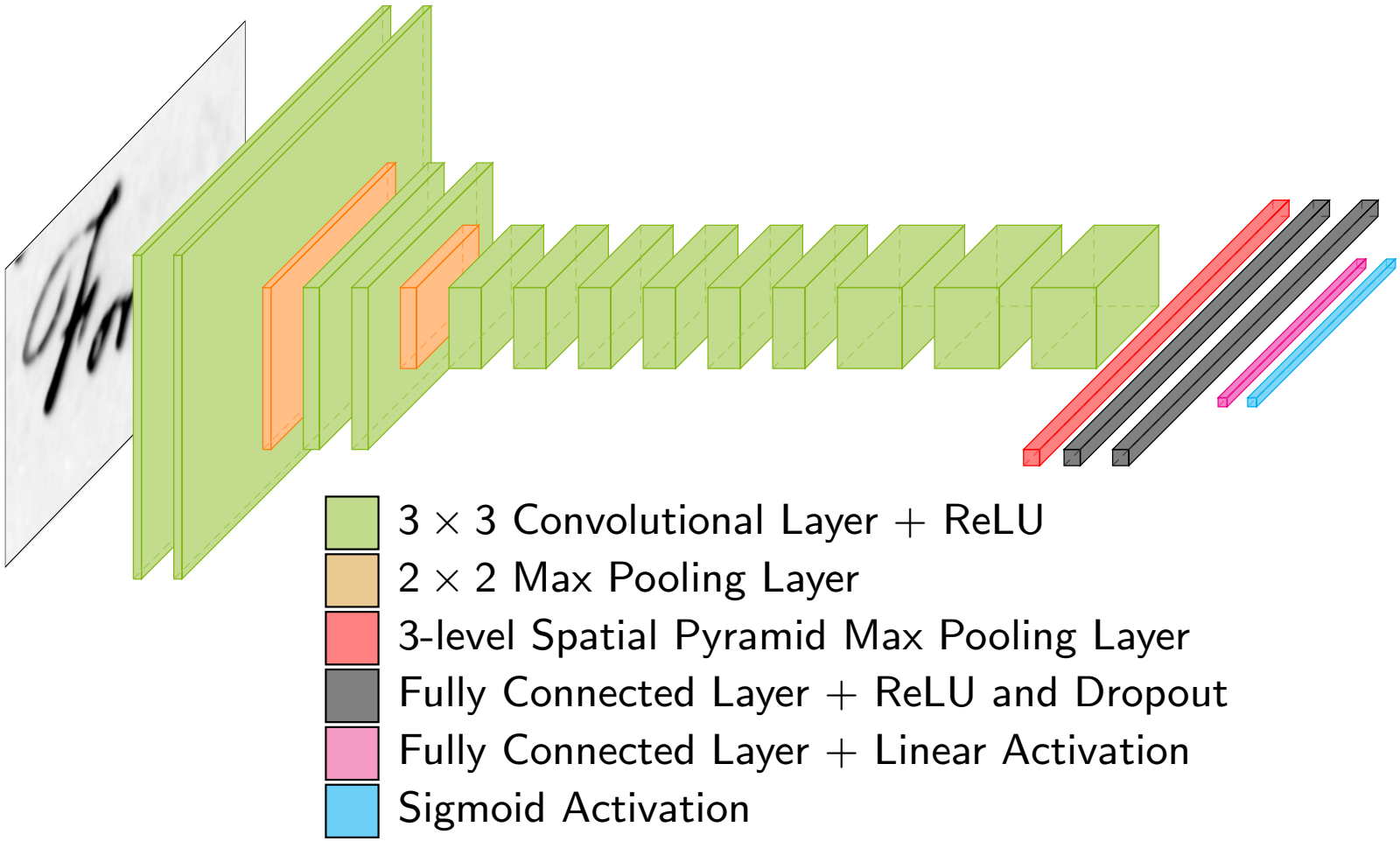
(For a recent introduction to Deep Learning see [LeCun *et. al*, 2015])

Overview

Reminder: General Framework



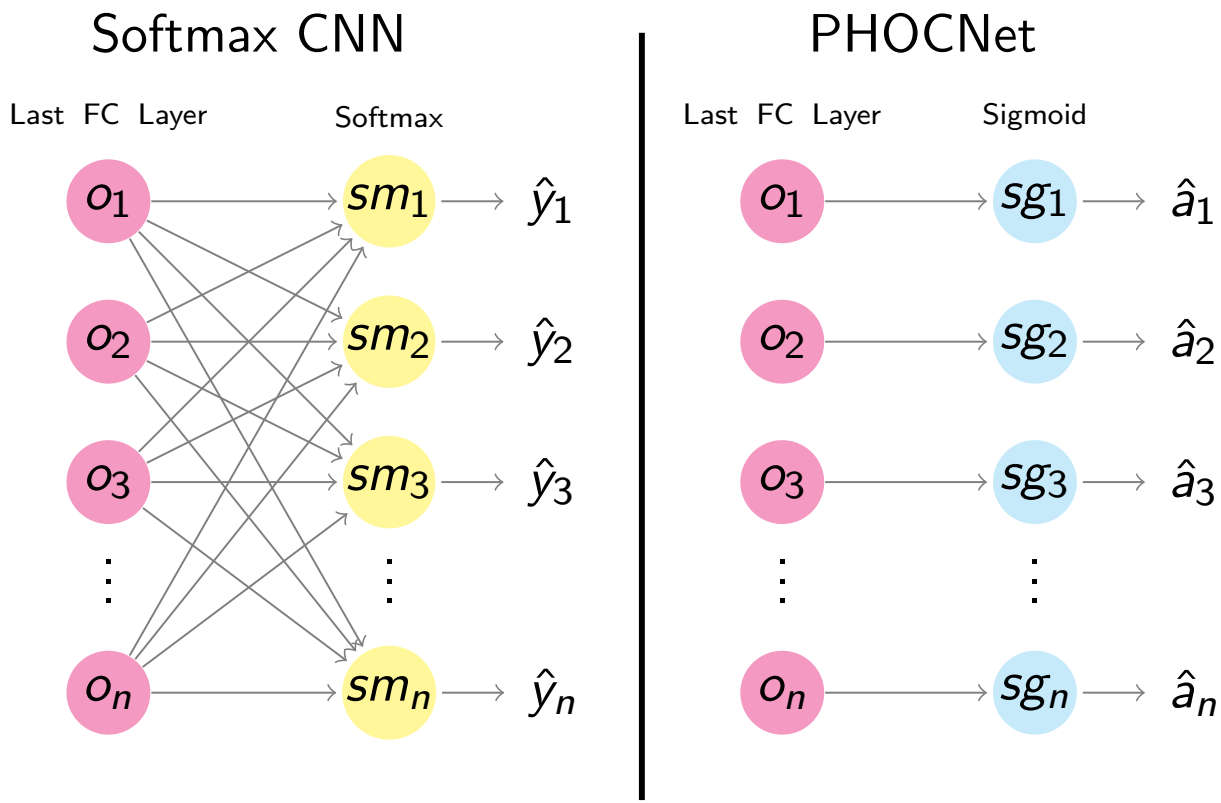
PHOCNet



S. Sudholt, G. A. Fink: PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents, Proc. Int. Conf. on Frontiers in Handwriting Recognition, Shenzhen, China, 2016.

Softmax CNN vs. PHOCNet

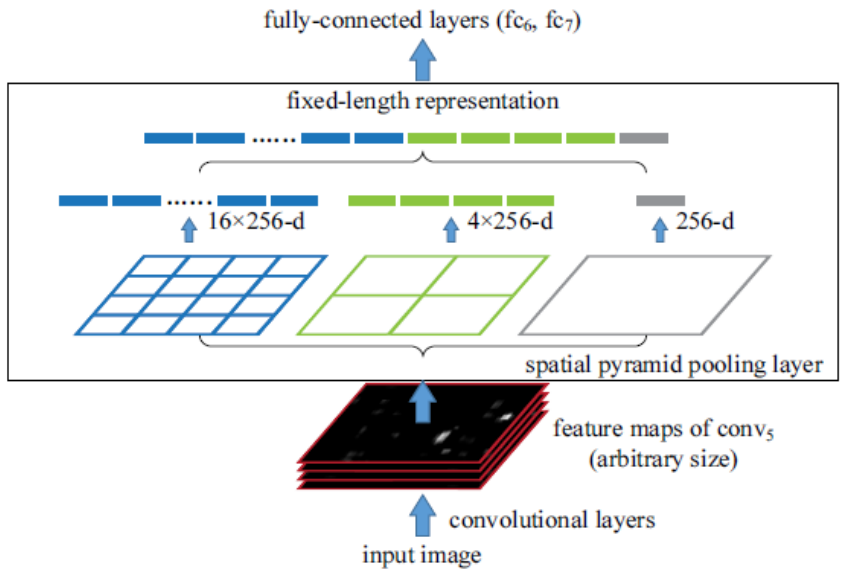
- In order to classify attributes, replace softmax with a sigmoid activation
- Each output neuron predicts one attribute



Spatial Pyramid Pooling Layer

- Convolutional layers can already deal with arbitrary image sizes
- Only MLP part has a problem with changing image sizes

Solution: apply spatial pyramid concept to the last convolutional output to generate fixed-size representation



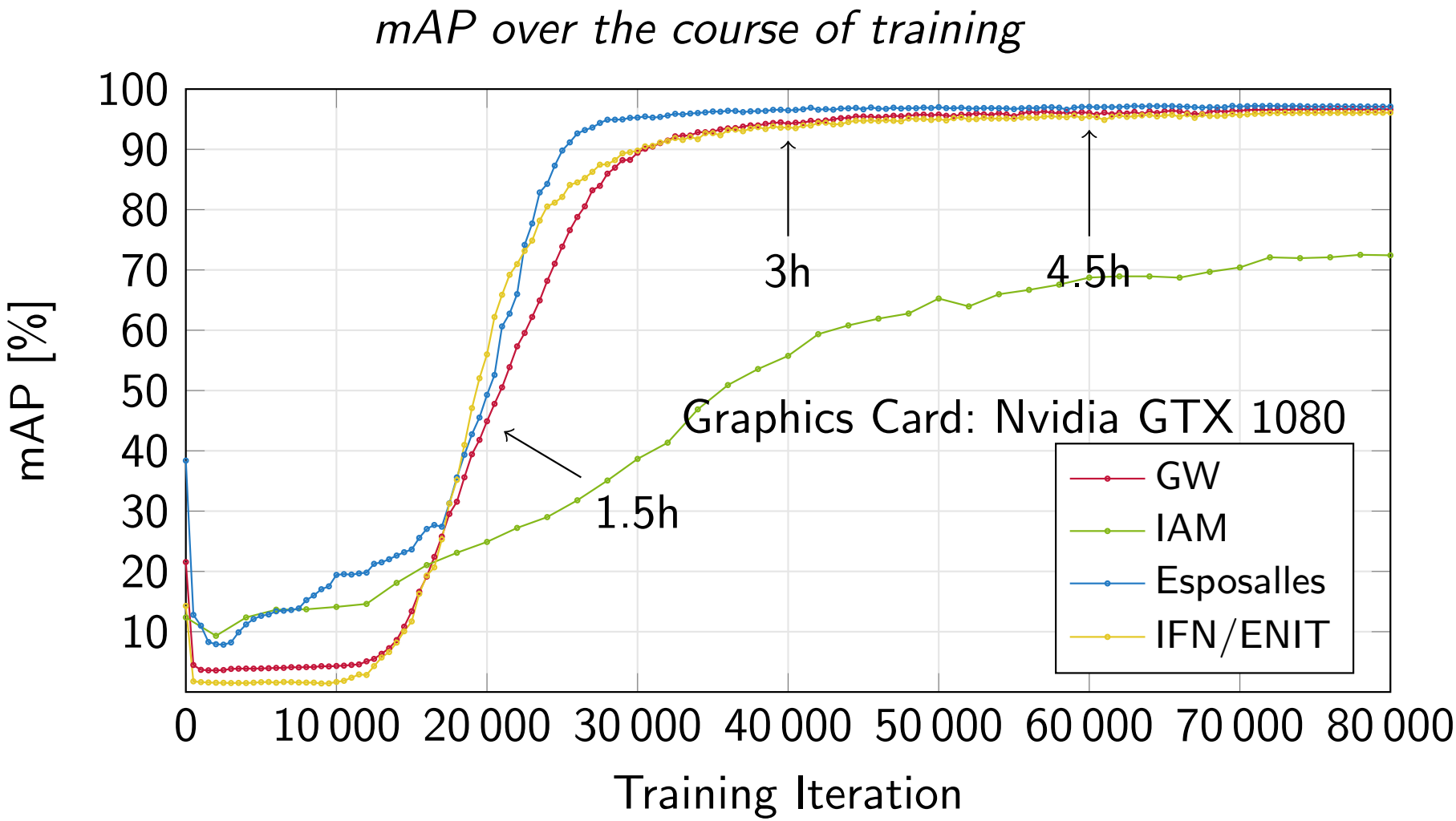
K. He, X. Zhang, S. Ren, J. Sun: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, Proc. European Conference on Computer Vision, pp. 346–361, 2014.

Initial Results

Segmentation-based Word Spotting Performance in mAP [%]

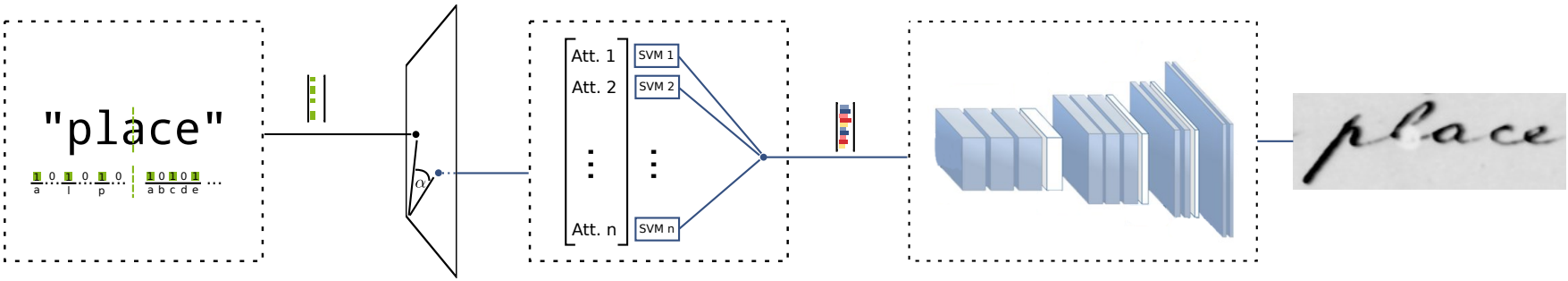
Method	GW		IAM		Esposalles		IFN/ENIT	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
PHOCNet	**97.85**	**97.65**	83.38	**92.59**	**96.93**	**94.33**	**95.63**	**93.51**
Attribute SVM [2]	93.04	91.29	55.73	73.72	-	-	-	-
Finetuned CNN [25]	-	-	46.53	-	-	-	-	-
LSA Embedding [1]	-	56.54	-	-	-	-	-	-
BLSTM [4]	-	84.00	-	78.00	-	-	-	-
SC-HMM [18]	53.10	-	-	-	-	-	41.60	-

Initial Results II

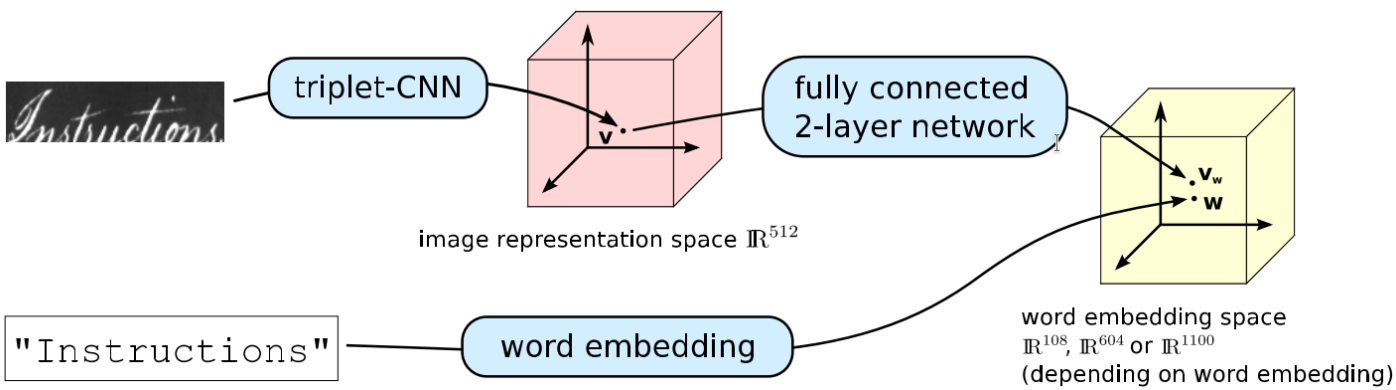


Related Work on Deep Learning for Word Spotting

Deep Feature Embedding [Krishnan *et al.*, 2016]



Triplet-CNN [Wilkinson & Brun, 2016]



Latest Results

Segmentation-based Word Spotting Performance in mAP [%]

Method	GW		IAM		Esposalles		IFN/ENIT	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
PHOCNet	97.96	**97.92**	**85.50**	**93.42**	**97.40**	**94.89**	**96.66**	**94.92**
Deep Feat. Emb. [9]	94.41	92.84	84.24	91.58	-	-	-	-
Triplet-CNN [32]	**98.00**	93.69	81.58	89.49	-	-	-	-

Revisiting the PHOCNet

Advantages PHOCNet:

- no pre-training as is done in the other two methods
- end-to-end

Questions:

- What happens during training?
- What does the PHOCNet learn?

→ focus of the second half of this tutorial

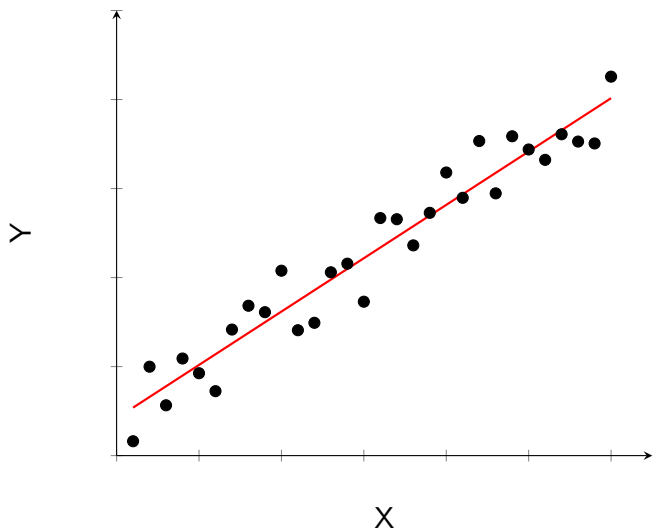
Overview

Predicting Outcomes of Events

Scenario: Outcome of an event y depends on some data x

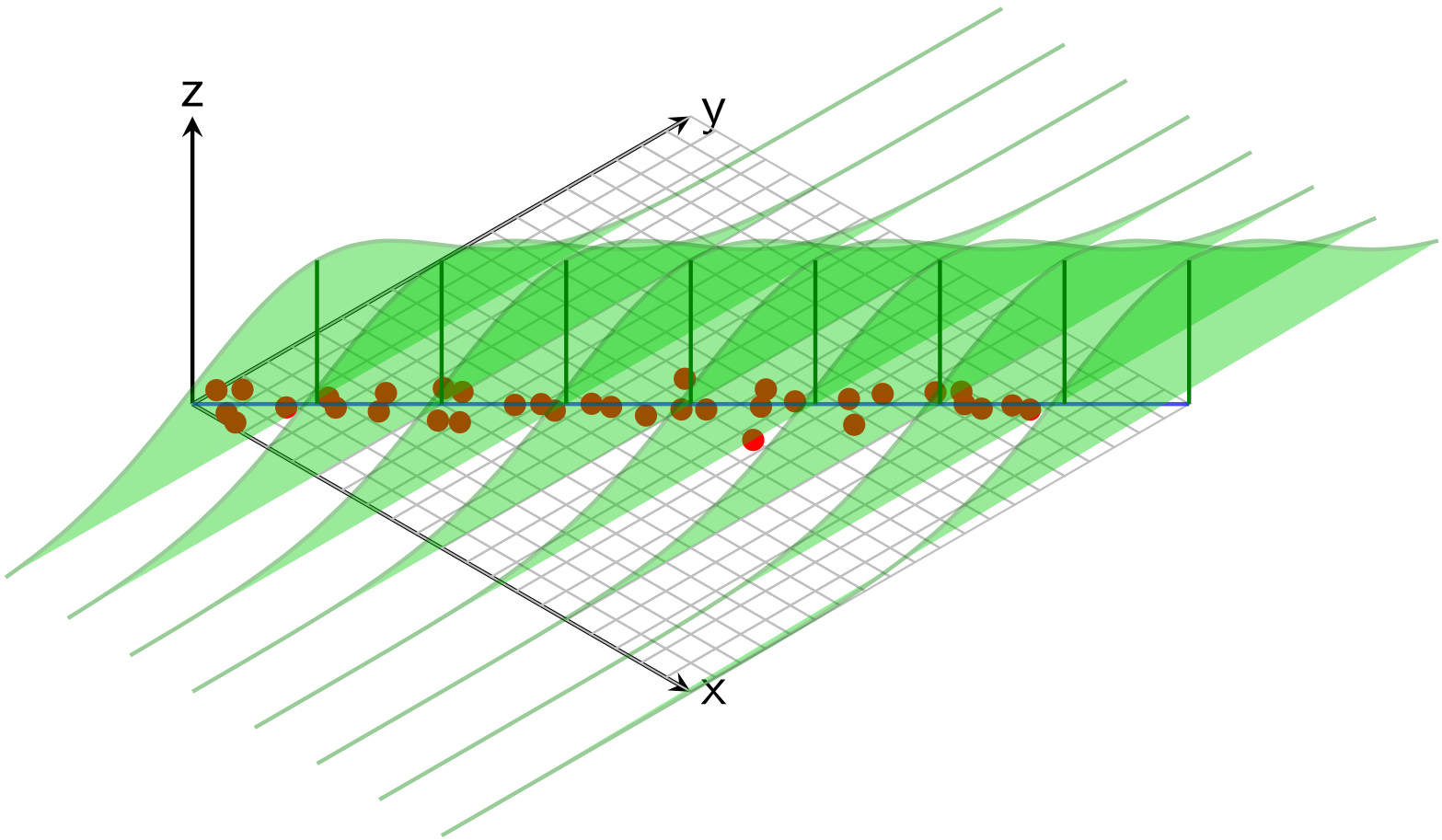
Goal: Predict outcome, i.e. classification or regression

Simplest case: Linear Regression



Assump. LR: Y follows a normal distribution

General Linear Model



General Linear Model

General Linear Model predicts the *conditional expected Value* $\mathbb{E}(Y|X)$

Assumption 1: Y follows a normal distribution

Assumption 2: $E(Y)$ depends on a linear combination of X

$\rightarrow$ $Y \sim N(\mathbf{w}^T X, \sigma)$

Eq. Assump.: $Y = \mathbf{w}^T X + \epsilon, \epsilon \sim N(0, \sigma)$

Problem: Classification, i.e. Y does not follow normal distribution

Solution: Generalized Linear Model

Generalized Linear Model (GLM)

GLM predicts the *conditional expected value* $\mathbb{E}(Y|X)$

Assumption 1: Y follows a distribution in the exponential family

Assumption 2: $\mathbb{E}(Y)$ depends on linear combination of X

Added Treat: We can now model either binary or multi-class classification in addition to regression, all within the same model. All that needs to change is the distribution of Y.

Three Components of a GLM

1. Distribution of Y from exponential family
   - Normal
   - Bernoulli
   - Categorical
   - ...
2. Linear Predictor
   - $\eta = \mathbf{w}^T X$
3. Link Function g^{-1}, transforming $\mathbb{E}(Y)$ into a desired range
   - $g(\mathbb{E}(Y)) = \eta$

$$\Rightarrow \mathbb{E}(Y) = g^{-1}\left(\mathbf{w}^T X\right)$$

Example: Logistic Regression

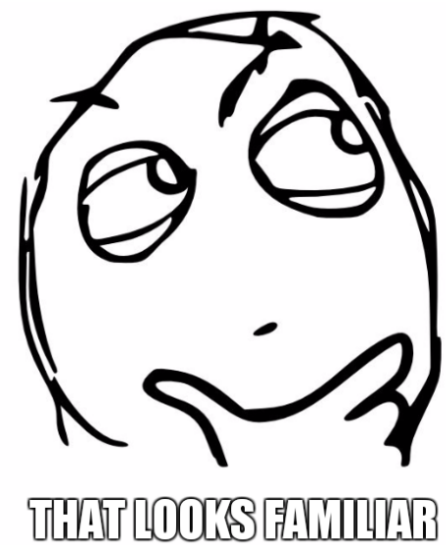
Assumption: Y follows Bernoulli distribution

PMF Bernoulli: $P(y \mid p) = p^y(1-p)^{1-y}, y \in \{0,1\}, 0 \leq p \leq 1$

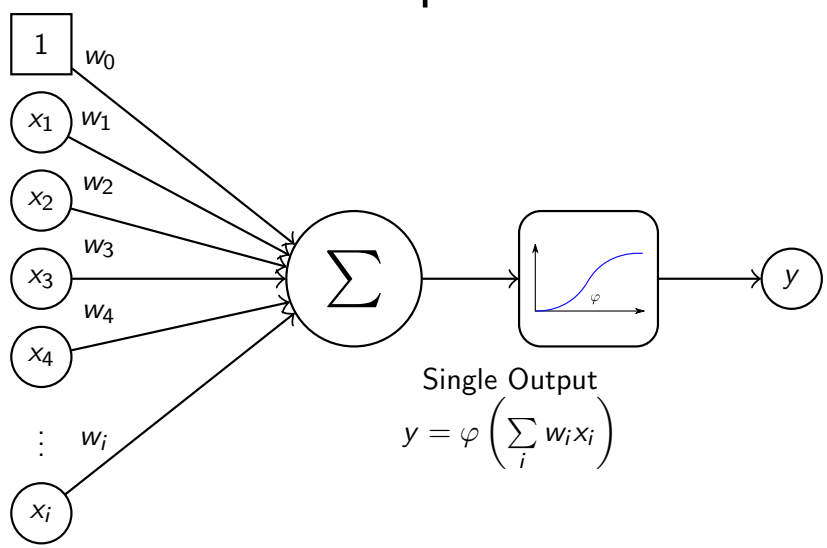
GLM Prediction: $\mathbb{E}(Y) = p$

Link Function: $g^{-1}(x) = \frac{1}{1+\exp(-x)} = \sigma(x)$

$\Rightarrow$ $p = \sigma\left(\mathbf{w}^T X\right)$



Perceptron



Regression and Classification with a GLM

Regression:
Distribution: $Y \sim \mathcal{N}(\mathbf{w}^T X, \sigma)$
Link Function: $g(\eta) = \eta \Rightarrow g^{-1}(\eta) = \eta$
Model: $E(Y) = \mathbf{w}^T X$

Binary Class.:
Distribution: $Y \sim \mathcal{B}(\mathbf{w}^T X, 1)$
Link Function:
$g(\eta) = \log\left(\frac{\eta}{1-\eta}\right) \Rightarrow g^{-1}(\eta) = \sigma(\eta)$
Model: $E(Y) = \sigma\left(\mathbf{w}^T x\right)$

Multi-class Class.:
Distribution: $Y \sim \mathcal{C}(\mathbf{W}X, 1)$
Link Function:
$g(\eta_i) = \log\left(\frac{\eta_i}{\sum_j \eta_j}\right) \Rightarrow g^{-1}(\eta_i) = \text{softmax}_i(\eta)$
Model: $E(Y_i) = \text{softmax}_i(\mathbf{W}X)$

Training a GLM: Maximum Likelihood Estimation

Given i.i.d. samples $S = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{n}$:

$$\begin{aligned}
\hat{\mathbf{w}} &= \operatorname*{argmax}_{\mathbf{w}} \prod_{i=1}^{n} p\left(Y = y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w} \right) \\
&= \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{n} \log p\left(Y = y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w} \right) \\
&= \operatorname*{argmin}_{\mathbf{w}} - \sum_{i=1}^{n} \log p\left(Y = y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w} \right) \\
&= \operatorname*{argmin}_{\mathbf{w}} \underbrace{- \sum_{i=1}^{n} \log p\left(Y = y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w} \right)}_{\text{negative log-likelihood of the model given the data}}
\end{aligned}$$

From GLM to Neural Network

Observation 1: Formulation of a GLM is equivalent to perceptron

Observation 2: Linear dependency betwen Y and X only assumption
$\rightarrow$ we can replace linear predictor with non-linear predictor (e.g., deep neural network)

Take-home Message

We can transfer all the insights gained from training GLMs to (deep) neural networks.

Designing Loss Functions:

1. Choose a distr. from the exp. family you assume your labels follow
2. Link function is activation function last layer of the net
3. Determine the neg. log-likelihood function
   $\rightarrow$ this is your loss function

Loss Function for Single Attribute

Simplification: Labels are single (binary) attributes

Assumption: Labels follow a Bernoulli distribution:

$$P_{\mathcal{B}}(y \mid p) = p^y(1-p)^{1-y}, y \in \{0,1\}, 0 \leq p \leq 1$$

Implication 1: Link function (i.e. last activation function) is the sigmoid

Implication 2: Neural Net predicts probability $\hat{y}$ of $Y = 1$

Loss Function:

$$\begin{aligned} l_{\mathcal{B}} &= -\sum_i \log P_{\mathcal{B}}\left(y^{(i)}, \hat{y}^{(i)}\right) \\ &= -\sum_i y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \end{aligned}$$

Binary Cross Entropy Loss

Interpretation of Binary Cross Entropy Loss

Take-home Message

Training a neural network with sigmoid activations in the last layer using the binary cross entropy loss is equivalent to MLE of the network's parameters given the data and assuming the labels follow a Bernoulli distribution.

Side Note: Same holds for softmax output and categorical cross entropy loss.

Multiple Binary Attributes

Problem: Attribute repr. are typically vectors of attributes

First Solution: Multivariate Bernoulli distr. [Dai *et al.*, 2013]

But: Requires output layer size 2^d
Typical attribute vec. sizes in WS: 540, $2^{540} \geq 10^{162}$

Assumption: Attributes are pair-wise independent

$$l_{\mathcal{B}} = -\log \prod_i \prod_d P_{\mathcal{B}}\left(y_d^{(i)}, \hat{y}_d^{(i)}\right)$$
$$= -\sum_i \sum_d y_d^{(i)} \log \hat{y}_d^{(i)} + \left(1 - y_d^{(i)}\right) \log\left(1 - \hat{y}_d^{(i)}\right)$$

This is what most DL toolboxes will sell you as *Binary Cross Entropy Loss* or *Sigmoid Cross Entropy Loss*

Revisiting the Assumption

Problem: Independence assumption may be too strong

Idea: Rather interpret vector as a whole

Observation: Cosine distance works well when comparing attribute representations

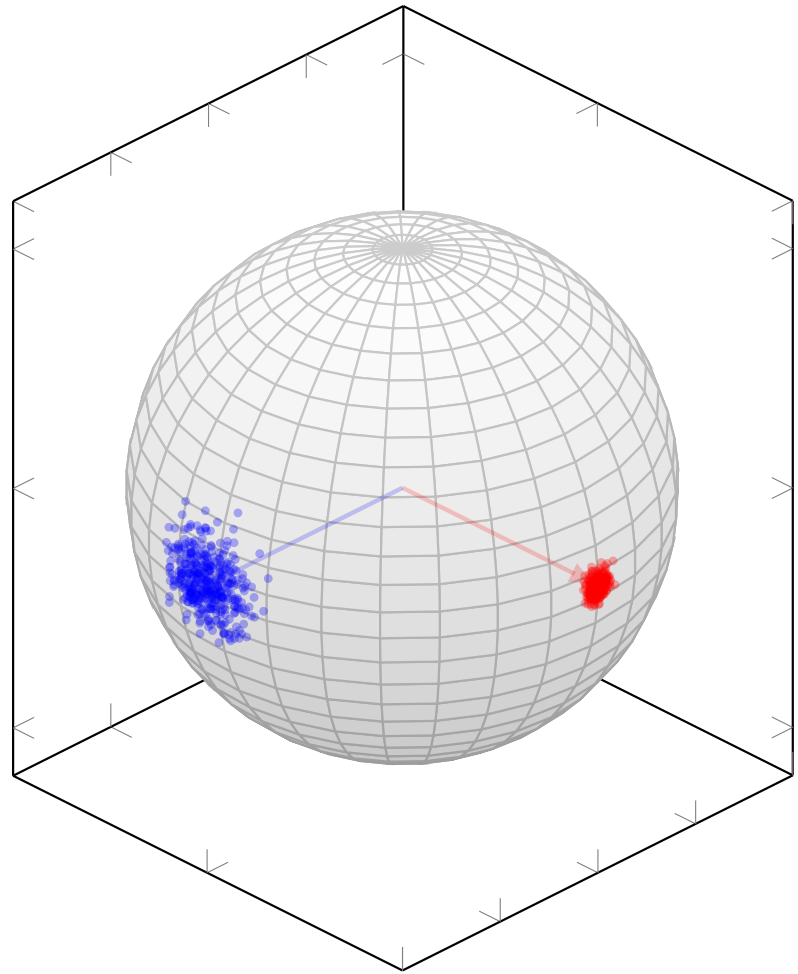
$$d(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{a}^T \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||}$$

Observation [Sra, 2016]

"[D]omains where similarity measures such as cosine [...] are more effective than Mahalanobis type distances, [...] are better modeled as directional data."

Von Mises-Fisher Distribution

Normal distribution on the unit hypersphere



PDF:

$$p_{\mathcal{MF}}(\mathbf{x} \mid \boldsymbol{\mu}, \kappa) = C_d(\kappa) \exp\left(\kappa \boldsymbol{\mu}^T \mathbf{x}\right)$$

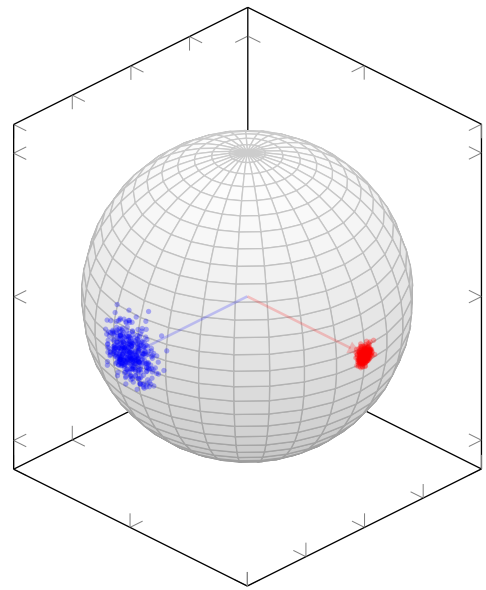
with $||\boldsymbol{\mu}|| = ||\mathbf{x}|| = 1$

Using the von Mises-Fisher Distribution for Forging a Loss

Assumption: Labels follow a von Mises-Fisher distr. with $\kappa = 1$

PDF: $p_{\mathcal{MF}}(\mathbf{x} \mid \boldsymbol{\mu}) = C_d(1) \exp\left(\boldsymbol{\mu}^T \mathbf{x}\right)$

Link Function: $g^{-1}(\mathbf{x}) = \mathbf{x}$



$$l(\text{data} \mid \mathbf{w}) = \underset{\mathbf{w}}{\text{argmax}} \sum_i \log p_{\mathcal{MF}}(\mathbf{y} \mid \boldsymbol{\mu})$$

$$= \underset{\mathbf{w}}{\text{argmin}} \sum_i 1 - \hat{\mathbf{y}}^{(i)} \cdot \mathbf{y}^{(i)}$$

Make sure $\hat{\mathbf{y}}$ and $\mathbf{y}$ are normalized:

$$l_{\mathcal{MF}} = \sum_i 1 - \frac{\mathbf{y}^{(i)} \cdot \hat{\mathbf{y}}^{(i)}}{\left|\left|\mathbf{y}^{(i)}\right|\right| \cdot \left|\left|\hat{\mathbf{y}}^{(i)}\right|\right|} = \sum_i 1 - \cos\left(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}\right)$$

$\rightarrow$ Cosine Loss

Interpretation of Cosine Loss

Take-home Message

Training a neural network with linear activations in the last layer using the cosine loss is equivalent to MLE of the network's parameters given the data and assuming the labels follow a von Mises-Fisher distribution with $\kappa = 1$.

Advantages of the Cosine Loss:

- No independence assumption
- Applicable for non-binary labels as well (DCToW, SPOC,. . .)

For evaluation stop by on Tuesday, 10:30, Large Conference Room
Sneak Preview: Independence assumption not critical.

Overview

Maximum A Posteriori Training

So far: Training a NN is MLE of the parameters
$\rightarrow \operatorname{argmax}_{\mathbf{w}} p(\text{data}|\mathbf{w})$

MAP Est.: Predict the posterior distribution of the model parameters given the data

$$\begin{aligned}
\operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|\text{data}) &= \operatorname*{argmax}_{\mathbf{w}} \frac{p(\text{data}|\mathbf{w})p(\mathbf{w})}{p(\text{data})} \\
&= \operatorname*{argmax}_{\mathbf{w}} p(\text{data}|\mathbf{w})p(\mathbf{w}) \\
&= \operatorname*{argmin}_{\mathbf{w}} -[\log p(\text{data}|\mathbf{w}) + \log p(\mathbf{w})] \\
&= \underbrace{\operatorname*{argmin}_{\mathbf{w}} [-\log p(\text{data}|\mathbf{w})]}_{\text{min. neg. log-likelihood data}} + \underbrace{\operatorname*{argmin}_{\mathbf{w}} -\log p(\mathbf{w})}_{\text{min. neg. log-likelihood model}}
\end{aligned}$$

Optimizing the model log-likelihood

Assumption: $w_i \sim \mathcal{N}(0, \sigma^2) \rightarrow$ ind. Gaussian prior for each weight

$$\begin{aligned}
\underset{\mathbf{w}}{\text{argmin}} - \log p(\mathbf{w}|0,\sigma) &= \underset{\mathbf{w}}{\text{argmin}} - \sum_i \log p(w_i|0,\sigma) \\
&= \underset{\mathbf{w}}{\text{argmin}} - \sum_i \log \left(a \cdot e^{-\frac{(w_i-0)^2}{2\sigma^2}} \right) \\
&= \underset{\mathbf{w}}{\text{argmin}} - \sum_i \log(a) - \frac{w_i^2}{2\sigma^2} \\
&= \underset{\mathbf{w}}{\text{argmin}} \sum_i \frac{1}{2\sigma^2} w_i^2 \\
&= \underset{\mathbf{w}}{\text{argmin}} \frac{1}{2\sigma^2} ||\mathbf{w}||_2^2 = \underbrace{\text{argmin}\, c \cdot ||\mathbf{w}||_2^2}_{\text{weight decay}}
\end{aligned}$$

Weight Decay

Take-home Message

Training a network with an added weight decay term in the loss function is equivalent to MAP-estimation of the net's parameters **w**.

Assumptions:

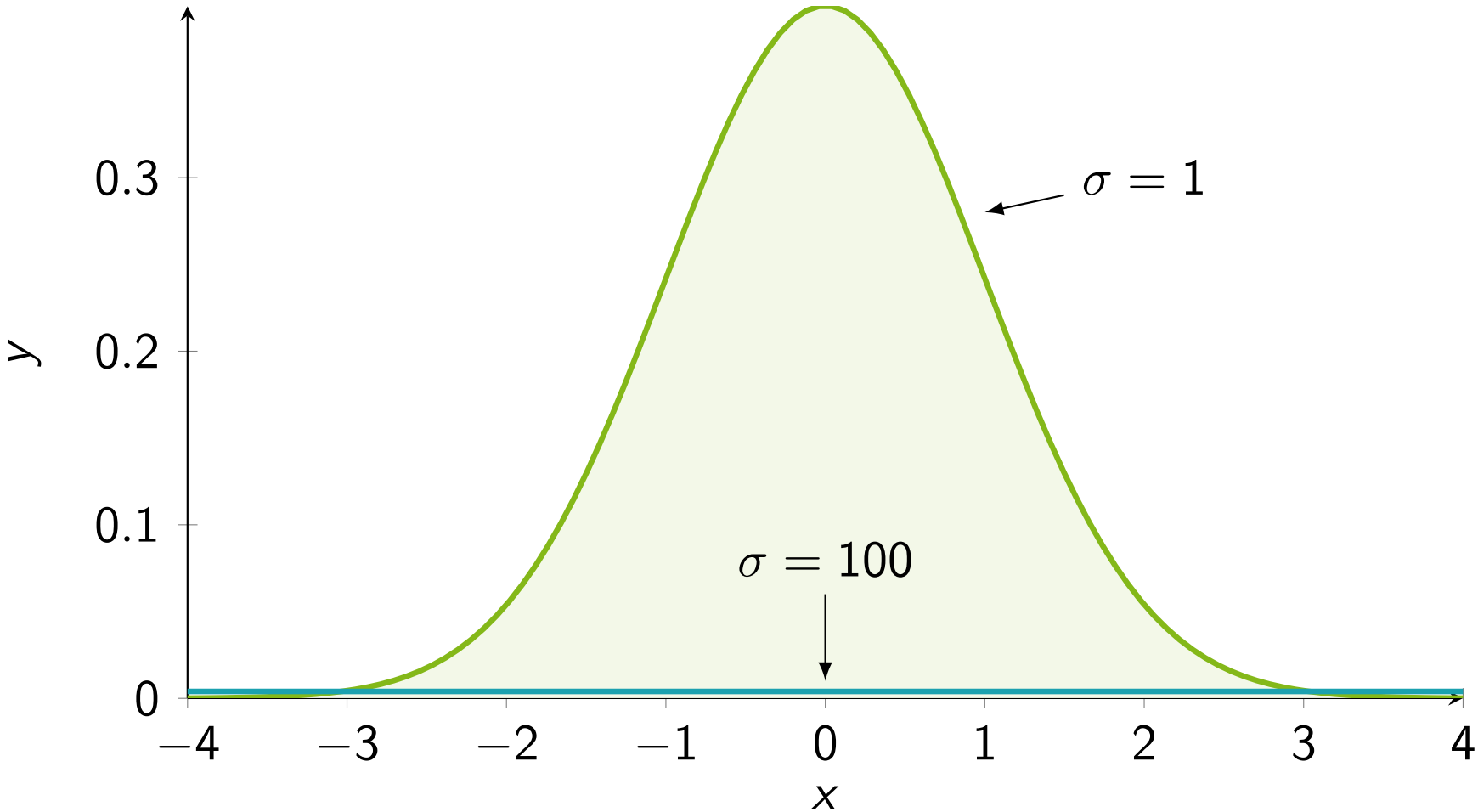
- The weights are pairwise independent
- Each weight has a prior distribution $\mathcal{N}(0, \sigma)$
- σ is the same for all weights

Example: Binary Cross Entropy Loss

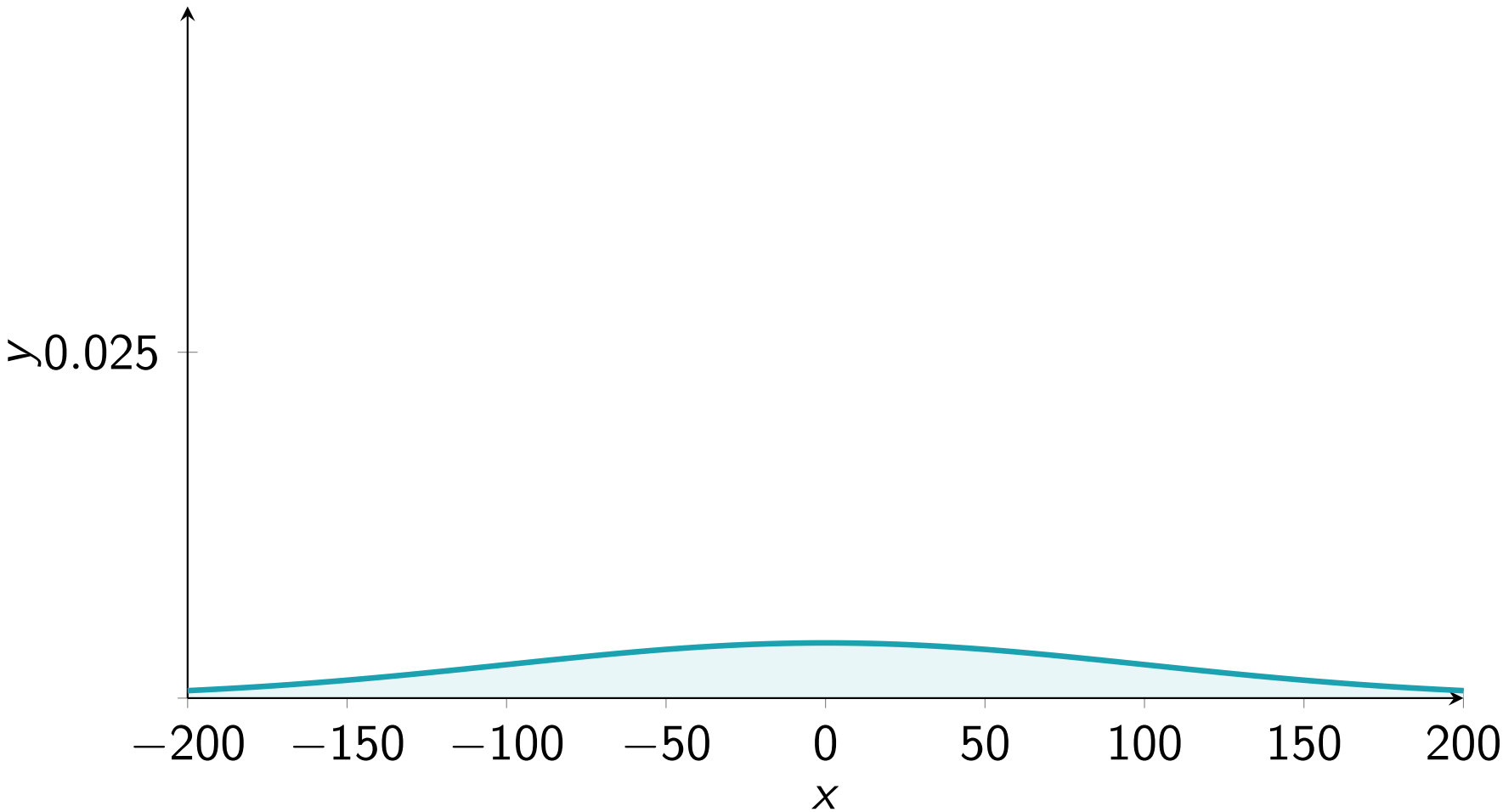
$$l_{\mathcal{B}} = -\sum_i y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - \hat{y}^{(i)}\right) + c \cdot ||\mathbf{w}||_2^2$$

- c is a meta-parameter which determines σ: $c = \frac{1}{2\sigma^2}$
- typical $c = 5 \cdot 10^{-5} \Rightarrow \sigma = 100$

Putting it in perspective...



Putting it in perspective...



Overview

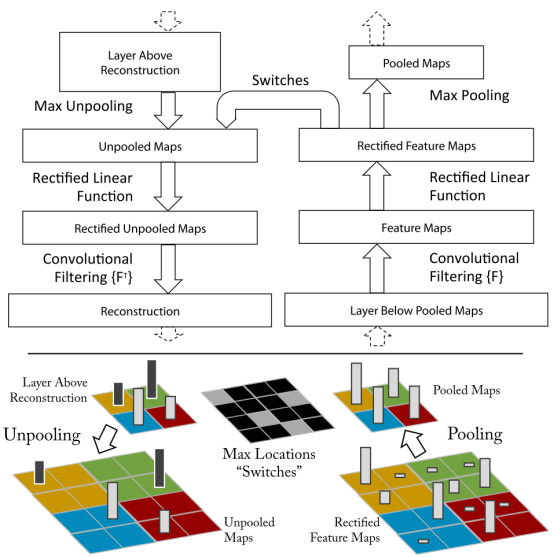
Filter Responses

Question: What did a CNN learn?

Reformulate: What part of the image activates a conv. filter the most?

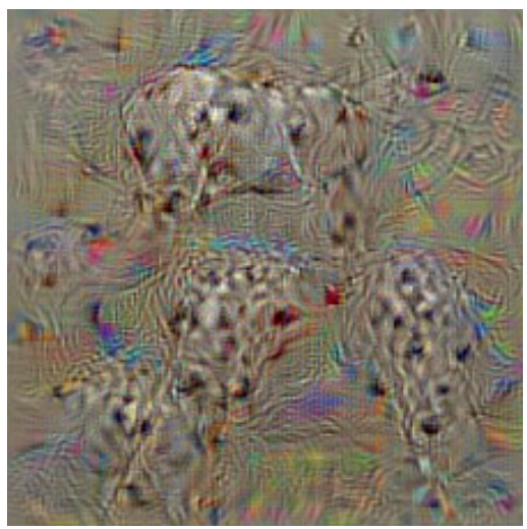
DeconvNet

Zeiler & Fergus, 2014



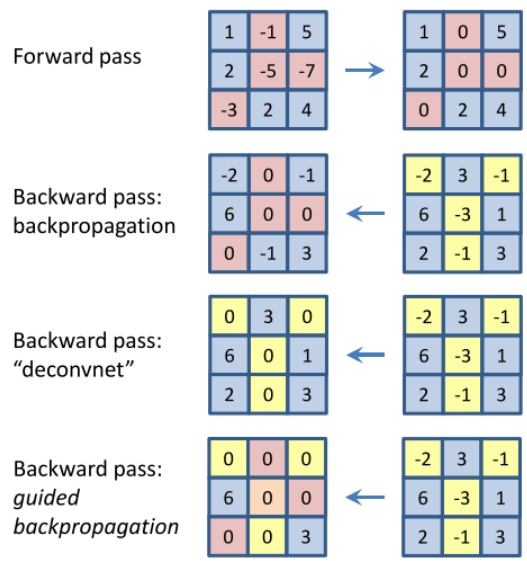
Class Saliency

Simonyan & Zisserman, 2014



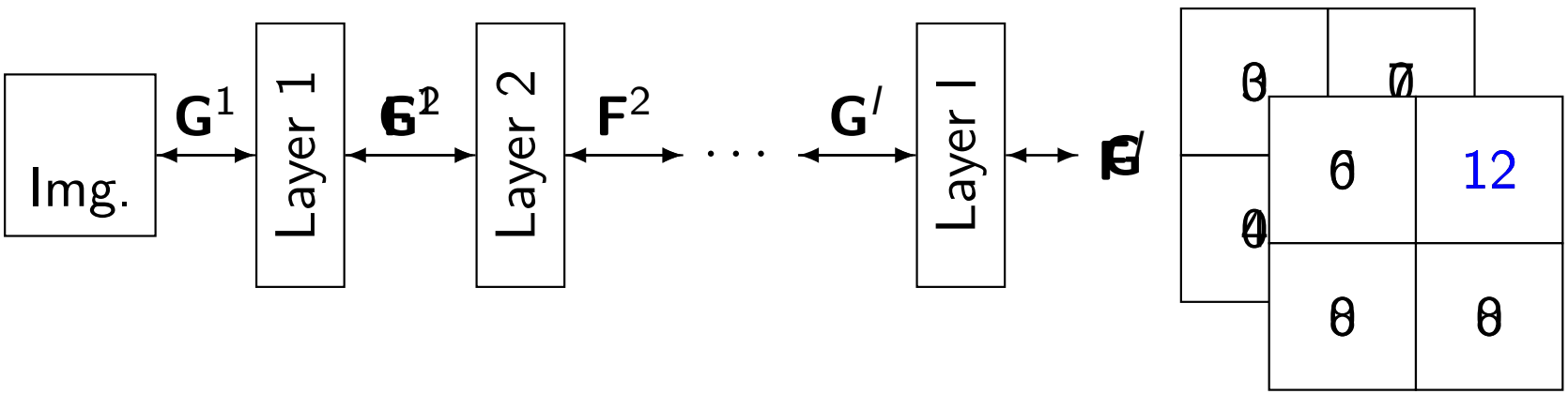
Guided Backprop

Springenberg *et al.*, 2015



Guided Backprop

1. Forward the image through the net until you reach desired layer
2. Set gradient of layer to max. val. of the desired filter
3. Backpropagate and recreate image



Springenberg, J. T., Dosovitskiy, A., Brox, Th., Riedmiller, M.: *Striving for simplicity: The all convolutional net.* In Proc. Int. Conf. on Learning Representations, 2015.

Guided Backprop

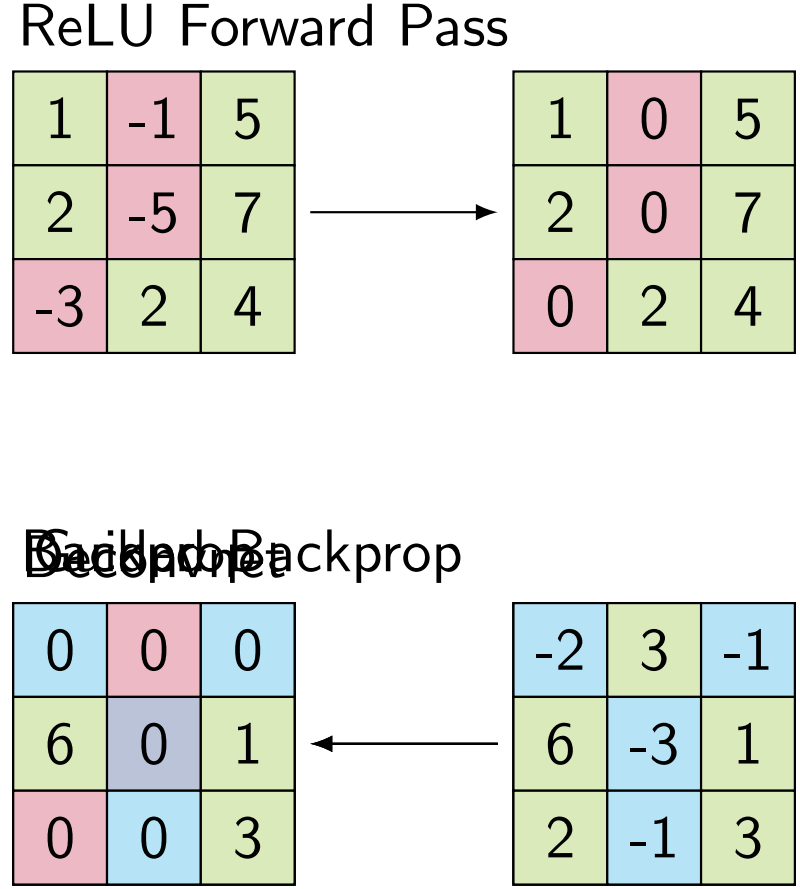
ReLU: $f(x) = \max(0, x)$

Der.: $\frac{df}{dx} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

Backprop:
$\mathbf{G}^l = (\mathbf{F}^l > 0) \odot \mathbf{G}^{l+1}$

Backward Path Deconvnet:
$\mathbf{G}^l = (\mathbf{G}^{l+1} > 0) \odot \mathbf{G}^{l+1}$

Guided Backprop:
$\mathbf{G}^l = (\mathbf{F}^l > 0) \odot (\mathbf{G}^{l+1} > 0) \odot \mathbf{G}^{l+1}$



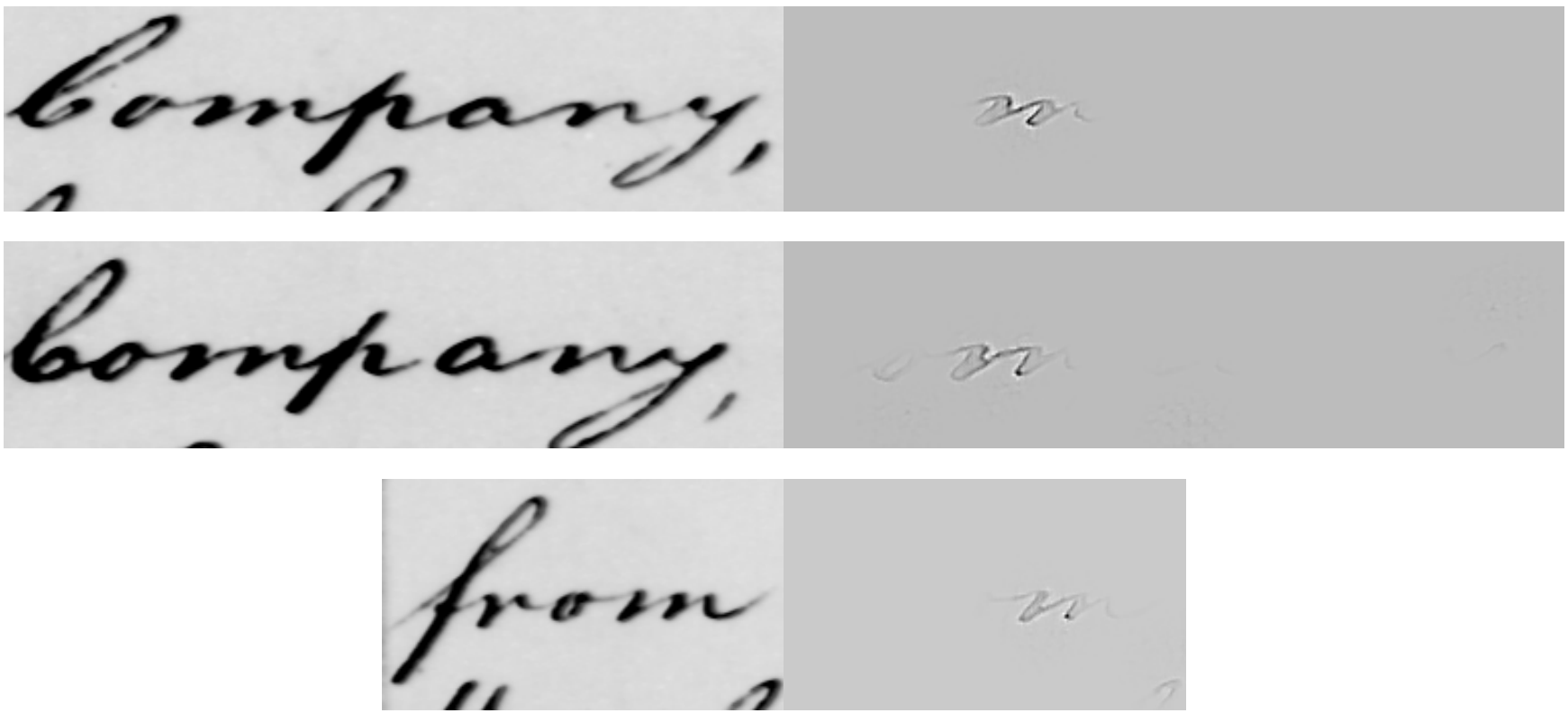
Springenberg, J. T., Dosovitskiy, A., Brox, Th., Riedmiller, M.: *Striving for simplicity: The all convolutional net.* In Proc. Int. Conf. on Learning Representations, 2015.

What did the PHOCNet learn?

Single Characters

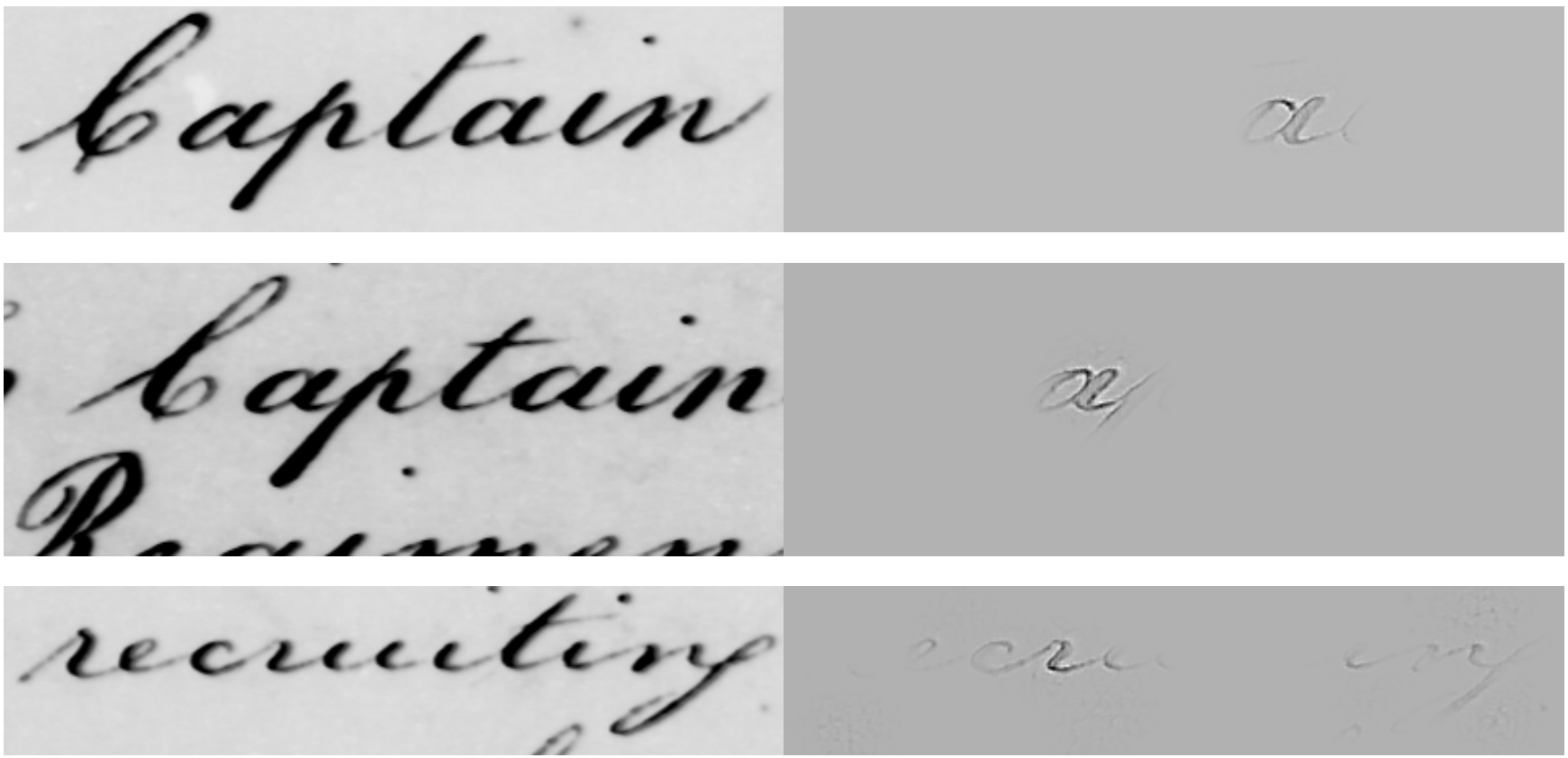
What did the PHOCNet learn?

Single Characters

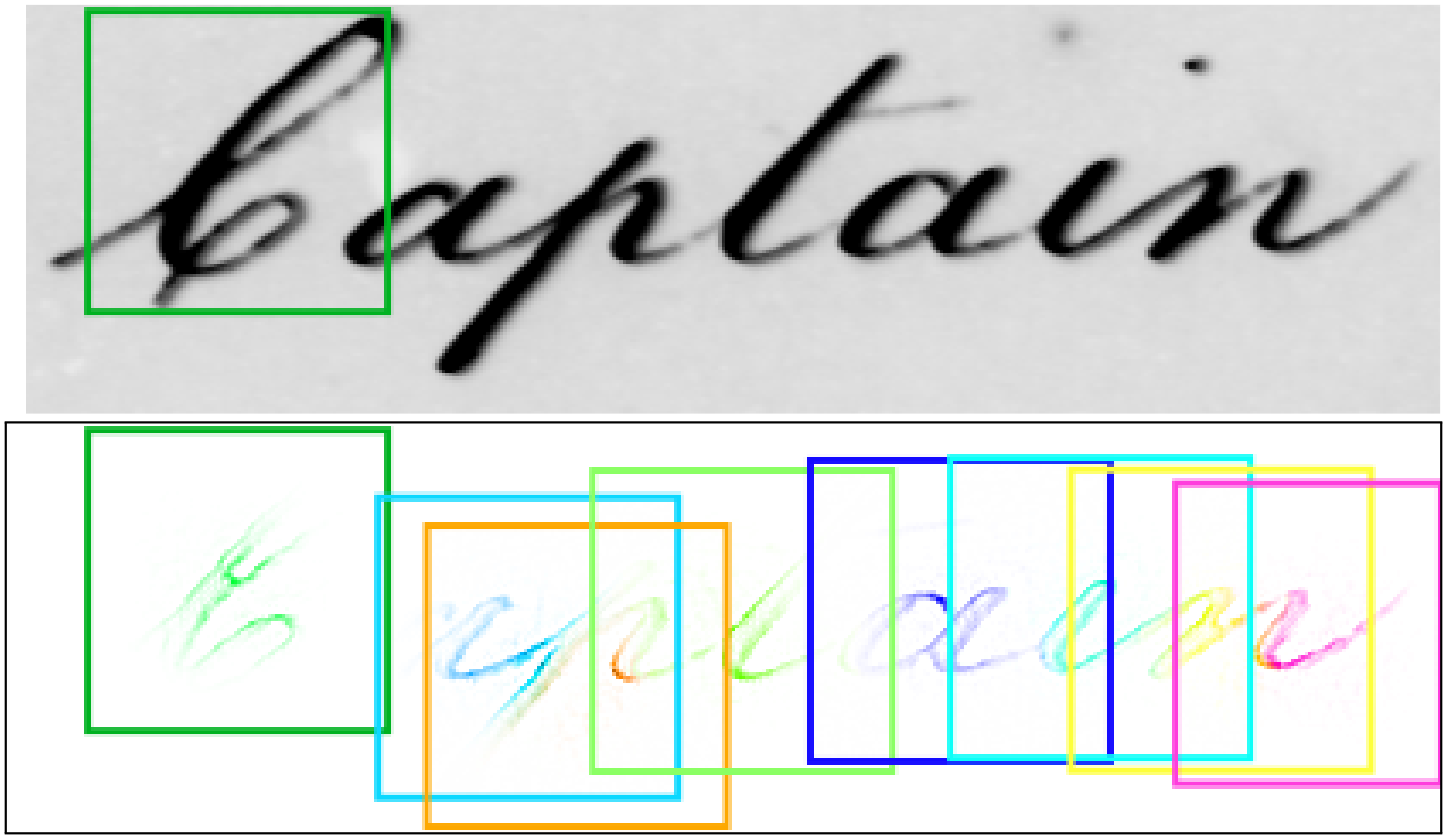


What did the PHOCNet learn?

Single Characters



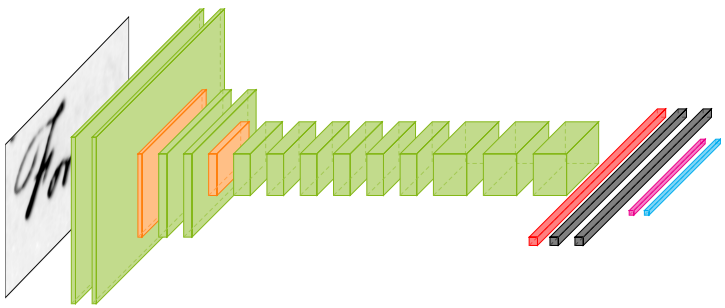
What did the PHOCNet learn?



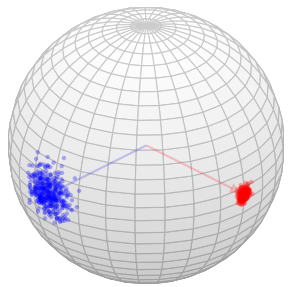
Overview

Summary

PHOCNet



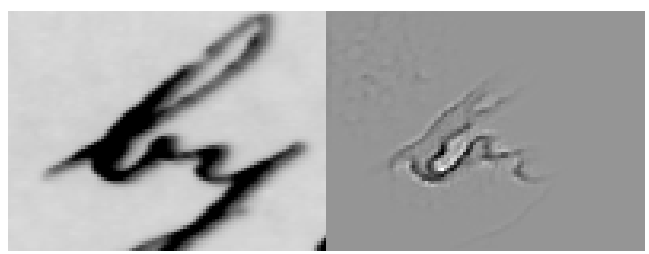
Cosine Loss vs. BCEL



Interpr. of Training

$p(\text{data} \mid \mathbf{w})$ vs. $p(\mathbf{w} \mid \text{data})$

Guided Backprop



References I

[1] David Aldavert, Marcal Rusinol, Ricardo Toledo, and Josep Llados.
Integrating visual and textual cues for query-by-string word spotting.
In *International Conference on Document Analysis and Recognition*, pages 511–515, 2013.

[2] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny.
Word spotting and recognition with embedded attributes.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 36(12):2552–2566, 2014.

[3] Bin Dai, Shilin Ding, and Grace Wahba.

References II

[4] Volkmar Frinken, Andreas Fischer, R. Manmatha, and Horst Bunke.
A novel word spotting method based on recurrent neural networks.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 34:211–224, 2012.

[5] Xavier Glorot and Yoshua Bengio.
Understanding the difficulty of training deep feedforward neural networks.
AISTATS, 9:249–256, 2010.

References III

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Spatial pyramid pooling in deep convolutional networks for visual recognition.
European Conference on Computer Vision, pages 346–361, 2014.

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White.
Multilayer feedforward networks are universal approximators.
Neural Networks, 2(5):359–366, 1989.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell.
Caffe: Convolutional architecture for fast feature embedding.
arXiv preprint arXiv:1408.5093, 2014.

References IV

[9] Praveen Krishnan, Kartik Dutta, and C.V. Jawahar.
Deep feature embedding for accurate recognition and retrieval of handwritten text.
In *International Conference on Frontiers in Handwriting Recognition*, pages 289–294, 2016.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Neural Information Processing Systems*, pages 1097–1105, 2012.

References V

[11] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce.
Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.
In *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 2169–2178, 2006.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton.
Deep learning.
Nature, 521:436–444, 2015.

[13] Yann LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel.
Handwritten digit recognition with a back-propagation network.
Neural Information Processing Systems, pages 396–404, 1990.

References VI

[14] David Lowe.
Distinctive image features from scale-invariant keypoints.
Int. J. of Computer Vision, 60(2):91–110, 2004.

[15] R. Manmatha, Chengfeng Han, E. M. Riseman, and W. B. Croft.
Indexing handwriting using word matching.
In *Proc. of the First ACM Int. Conf. on Digital Libraries*, DL '96, pages 151–159, New York, NY, USA, 1996. ACM.

[16] Stephen O'Hara and Bruce A. Draper.
Introduction to the bag of features paradigm for image classification and retrieval.
Computing Research Repository, arXiv:1101.3354v1, 2011.

References VII

[17] Toni M. Rath and R. Manmatha.
Word image matching using dynamic time warping.
In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–521–II–527 vol.2, June 2003.

[18] José A. Rodríguez-Serrano and Florent Perronnin.
A model-based sequence similarity with application to handwritten word spotting.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2108–2120, 2012.

References VIII

[19] F. Rosenblatt.
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, 65(6):386–408, 1958.

[20] Leonard Rothacker and Gernot A. Fink.
Segmentation-free query-by-string word spotting with bag-of-features HMMs.
In *Proc. Int. Conf. on Document Analysis and Recognition*, Nancy, France, 2015.

References IX

[21] Leonard Rothacker, Marcal Rusinol, and Gernot A. Fink.
Bag-of-features HMMs for segmentation-free word spotting in handwritten documents.
In *Proc. Int. Conf. on Document Analysis and Recognition*, Washington DC, USA, 2013.

[22] Leonard Rothacker, Marcal Rusinol, Josep Llados, and Gernot A. Fink.
A Two-Stage Approach to Segmentation-Free Query-by-Example Word Spotting.
manuscript cultures, 1(7):47–57, 2014.

References X

[23] Leonard Rothacker, Szilard Vajda, and Gernot A. Fink.
Bag-of-features representations for offline handwriting recognition applied to Arabic script.
In *Proc. Int. Conf. on Frontiers in Handwriting Recognition*, Bari, Italy, 2012.

[24] M. Rusiñol, D. Aldavert, R. Toledo, and J. Llados.
Browsing heterogeneous document collections by a segmentation-free word spotting method.
In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 63 –67, Bejing, China, 2011.

References XI

[25] Arjun Sharma and Sankar K. Pramod.
Adapting off-the-shelf CNNs for word spotting & recognition.
In *International Conference on Document Image Analysis*, pages 986–990, 2015.

[26] R. Shekhar and C.V. Jawahar.
Word image retrieval using bag of visual words.
In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 297–301, March 2012.

[27] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman.
Deep inside convolutional networks: Visualising image classification models and saliency maps.
In *Proc. of the Int. Conf. on Learning Representations*, 2014.

References XII

[28] Karen Simonyan and Andrew Zisserman.
Very deep convolutional networks for large-scale image recognition.
arXiv, pages 1–13, 2014.

[29] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller.
Striving for simplicity: The all convolutional net.
In *Proc. of the Int. Conf. on Learning Representations*, 2015.

[30] Suvrit Sra.
Directional Statistics in Machine Learning: A Brief Review.
In Christoph Ley and Thomas Verdebout, editors, *Modern Statistical Methods for Directional Data*, chapter 1. CRC Press, 2016.

References XIII

[31] Sebastian Sudholt and Gernot A. Fink.
PHOCNet: A deep convolutional neural network for word spotting in handwritten documents.
In *Proc. Int. Conf. on Frontiers in Handwriting Recognition*, Shenzhen, China, 2016.

[32] Tomas Wilkinson and Anders Brun.
Semantic and verbatim word spotting using deep neural networks.
In *International Conference on Frontiers in Handwriting Recognition*, pages 307–312, 2016.

References XIV

[33] Matthew D. Zeiler and Rob Fergus.
Visualizing and understanding convolutional networks.
In *Proc. of the European Conference on Computer Vision*, pages 818–833, 2014.