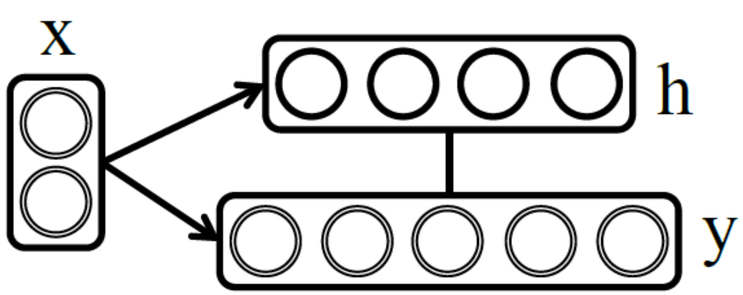
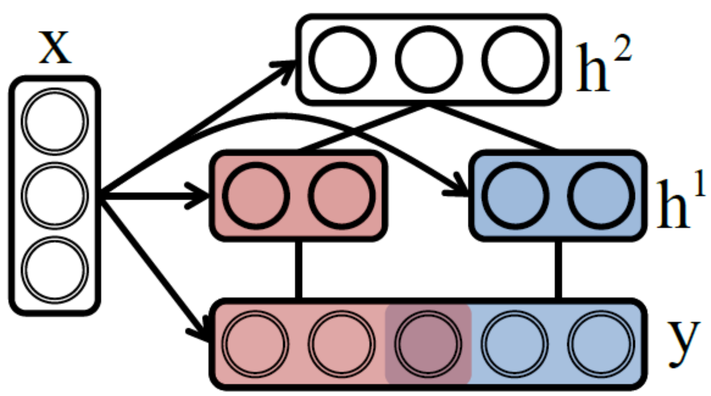
Convolutional Neural Networks inside Max-Margin Boltzmann Machines

Igor Gitman

CMC MSU





Outline

Introduction/Reminders

- RBM, ShapeBM
- MMBM

Problem/Experiments

- Problem formulation
- Convolutional Networks
- Experiments

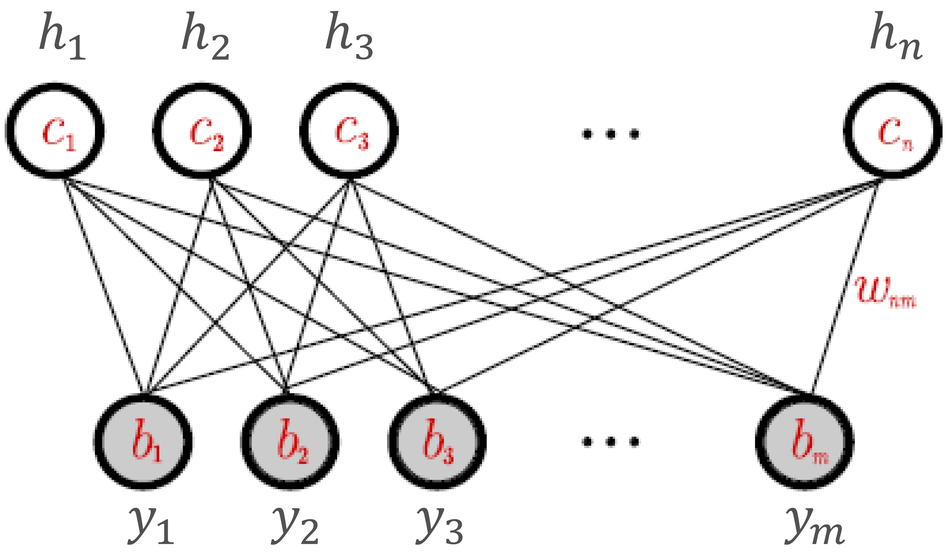
Conclusions/Future work

RBM and ShapeBM

Objective: binary image generation

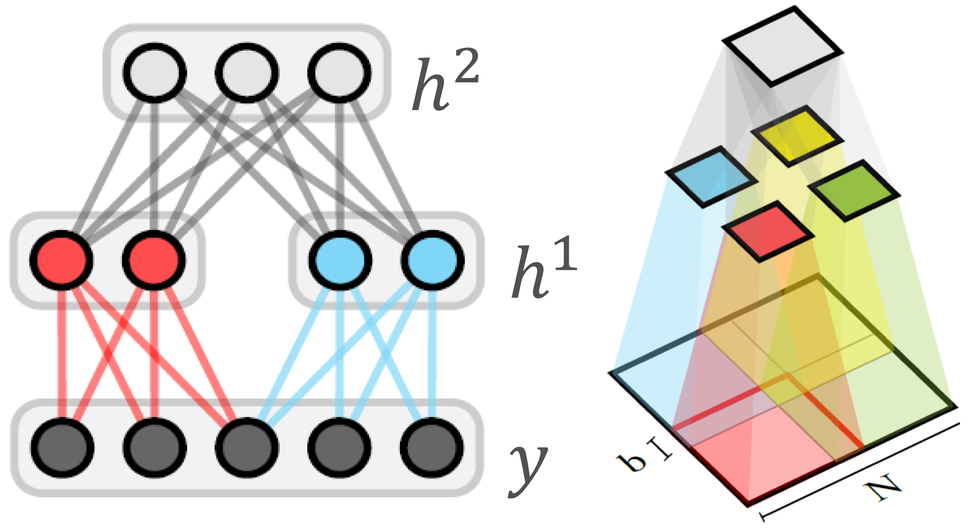
RBM

$$E(y,h) = -y^T W h - b^T y - c^T h$$



ShapeBM

$$E(y,h^1,h^2) = -y^T W^1 h^1 - b^T y - c^{1T} h^1 - h^{1T} W^2 h^2 - c^{2T} h^2$$

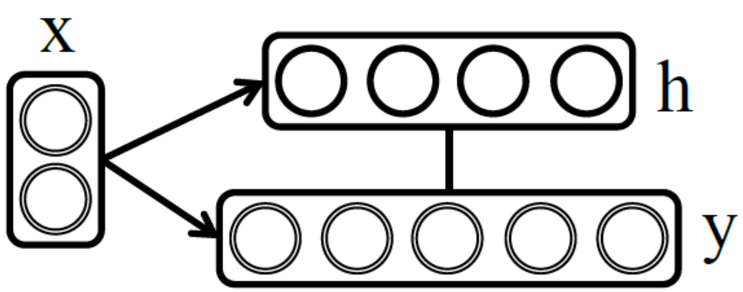


Max-Margin Boltzmann Machines

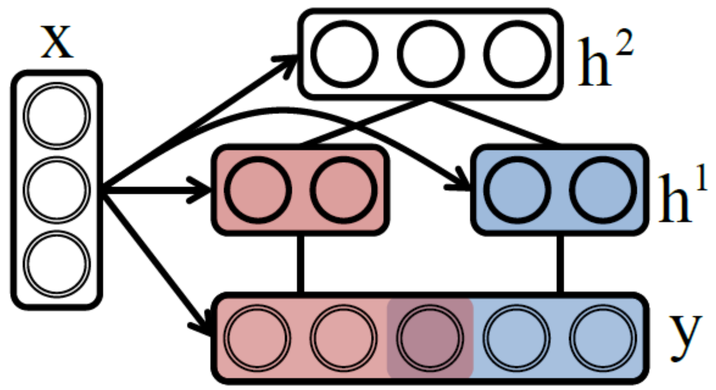
Objective: image segmentation

$$E(y, h, x) = -y^T W h - h^T(V^1 x^1 + c) - y^T(V^0 x^0 + b)$$

$$E(y, h^1, h^2, x) = -y^T W^1 h^1 - h^{1T} W^2 h^2 - h^{1T}(V^1 x^1 + c^1) - h^{2T}(V^2 x^2 + c^2) - y^T(V^0 x^0 + b)$$



MMBM1: $p(y, h|x)$



MMBM2: $p(y, h^1, h^2|x)$

$$y^* = \underset{y}{\operatorname{argmax}}\, p(y|x)$$

Max-Margin Boltzmann Machines

Objective: image segmentation

$$E(y, h, x) = -y^T W h - h^T(V^1 x^1 + c) - y^T(V^0 x^0 + b)$$

$$E(y, h^1, h^2, x) = -y^T W^1 h^1 - h^{1T} W^2 h^2 - h^{1T}(V^1 x^1 + c^1) - h^{2T}(V^2 x^2 + c^2) - y^T(V^0 x^0 + b)$$

x^0 — SIFT, color and contour histograms for each superpixel

x^1 — HOG descriptors for the entire image or for the four patches

x^2 — HOG descriptors for the entire image

Max-Margin Boltzmann Machines

Objective: image segmentation

$$E(y,h,x) = -y^T W h - h^T(V^1 x^1 + c) - y^T(V^0 x^0 + b)$$

n – number of features for the layer

m – number of variables in the layer

$$\begin{pmatrix} x_{i1}^0 \\ \vdots \\ x_{in}^0 \end{pmatrix}$$

$$V^0 x^0 = (v_1^0 \quad \dots \quad v_n^0) \begin{pmatrix} x_{11}^0 & \dots & x_{m1}^0 \\ x_{12}^0 & \dots & x_{m2}^0 \\ \vdots & \ddots & \vdots \\ x_{1n}^0 & \dots & x_{mn}^0 \end{pmatrix}$$

$$V^1 x^1 = \begin{pmatrix} v_{11}^0 & \dots & v_{1n}^0 \\ v_{21}^0 & \dots & v_{2n}^0 \\ \vdots & \ddots & \vdots \\ v_{m1}^0 & \dots & v_{mn}^0 \end{pmatrix} \begin{pmatrix} x_1^1 \\ \vdots \\ x_n^1 \end{pmatrix}$$

Learn only $[num_{\downarrow} features \times 1]$ matrix because we are able to extract different local features for each superpixel.

Unfortunately we don't have any locality here so we have to learn m "classifiers" instead of one.

Learning MMBMs

Find w : $p(y_i, H|x_i; w) \geq p(y, H|x_i; w) \forall H, \forall y, \forall i$

$$\approx$$

$$\max_H \left[-E(y_i, H|x_i; w) \right] \geq \max_{y,H} \left[-E(y, H, x_i; w) + \Delta(y, y_i, H, H_i^*) \right] \forall i$$

$$\left| H_i^* = \operatorname*{argmax}_H -E(y_i, H|x_i; w) \right|$$

$$\approx$$

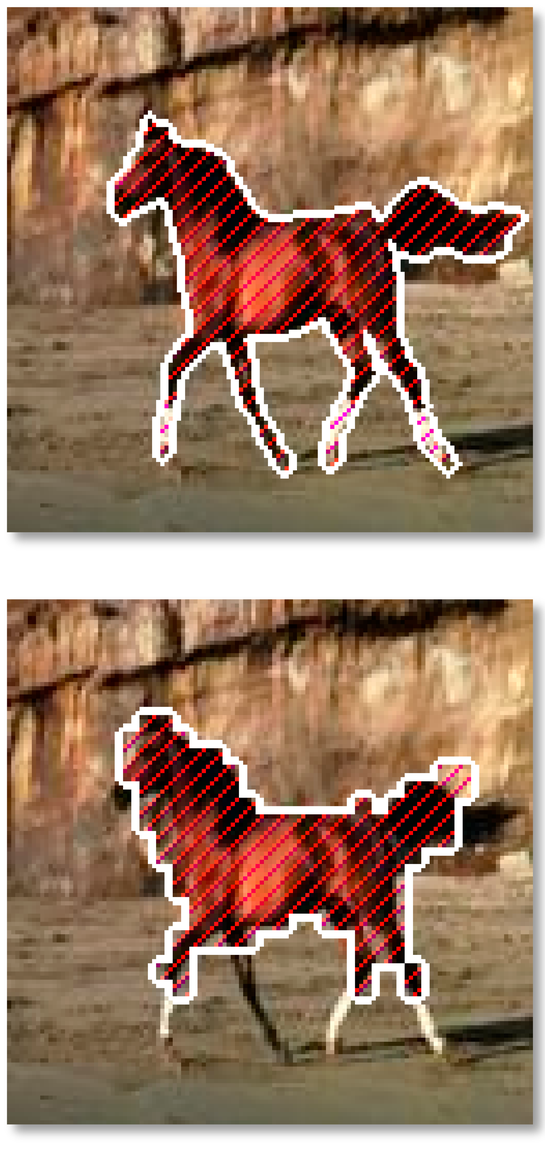
Find $w : \min_{\xi,w} \frac{\gamma}{2}||w||^2 + \sum_i \xi_i$, subject to

$$\max_H \left[-E(y_i, H|x_i; w) \right] \geq \max_{y,H} \left[-E(y, H, x_i; w) + \Delta(y, y_i, H, H_i^*) \right] - \xi_i \ \forall i$$

$$\xi_i \geq 0 \ \forall i$$

MMBMs performance

Weizmann Horses
328 картинок



Caltech-UCSD Birds 200
6033 картинки

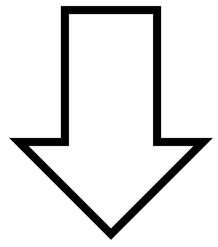
	Horses		Birds	
	AP	IoU	AP	IoU
CRF	87.46	67.44	83.5	38.45
CHOPPs	88.67	71.6	74.52	48.84
MMBM1	89.43	69.59	88.07	72.96
MMBM2	89.8	72.09	86.38	69.87
MMBM1 with GC	90.62	74.12	**90.42**	**75.92**
MMBM2 with GC	**90.71**	**75.78**	90.77	72.4

AP: percentage of “correct” pixels

IoU: $\frac{|Y \cap Y^*|}{|Y \cup Y^*|}$

Problem formulation

So, what is the problem?



Handcrafted features

Problem formulation

Handcrafted features

$$E(y, h, x) = -y^T W h - h^T (V^1 x^1 + c) - y^T (V^0 x^0 + b)$$

$$E(y, h^1, h^2, x) = -y^T W^1 h^1 - h^{1T} W^2 h^2 - h^{1T} (V^1 x^1 + c^1) - \\ - h^{2T} (V^2 x^2 + c^2) - y^T (V^0 x^0 + b)$$

x^0 — SIFT, color and contour histograms for each superpixel

x^1 — HOG descriptors for the entire image or for the four patches

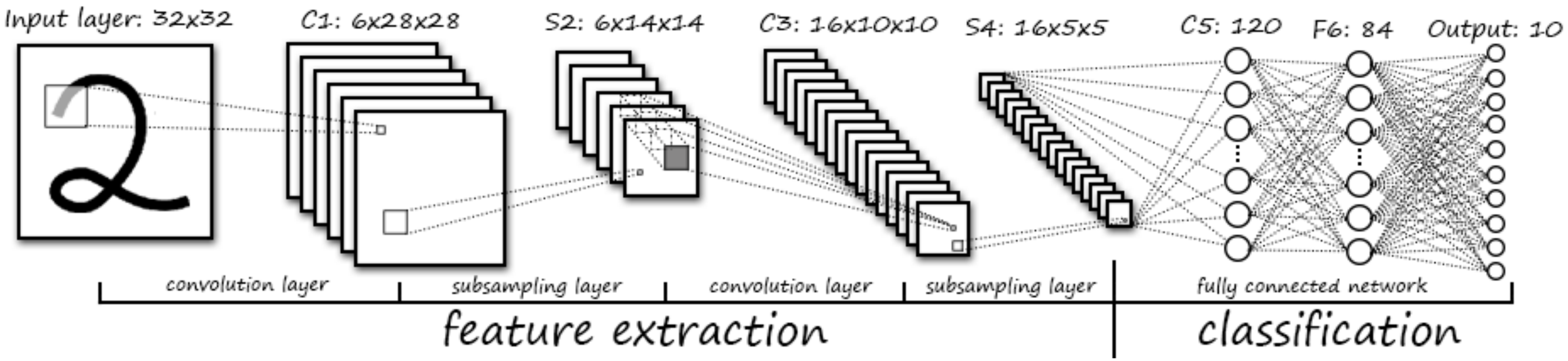
x^2 — HOG descriptors for the entire image

Problem solution

Replace **handcrafted features** with features from

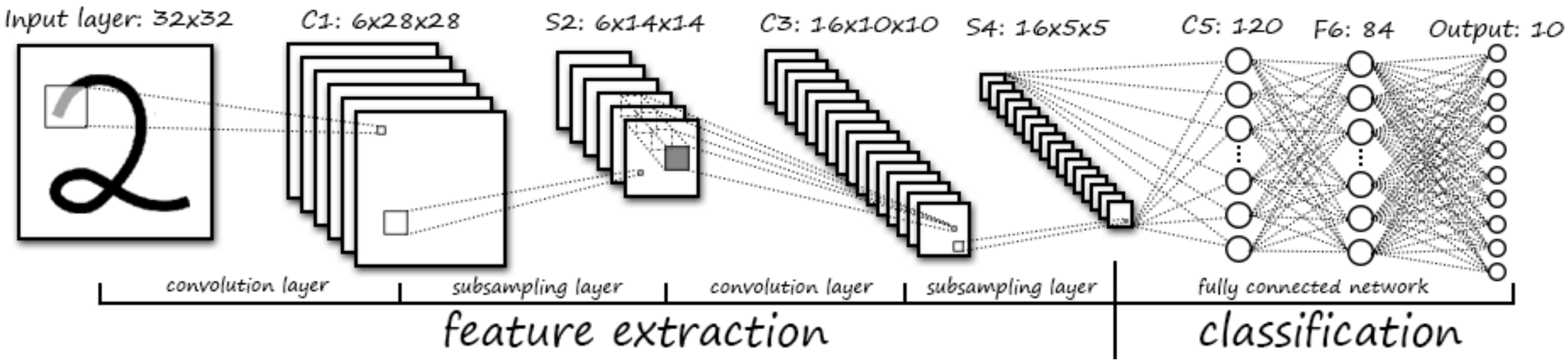
Convolutional Neural Networks

Convolutional Neural Networks

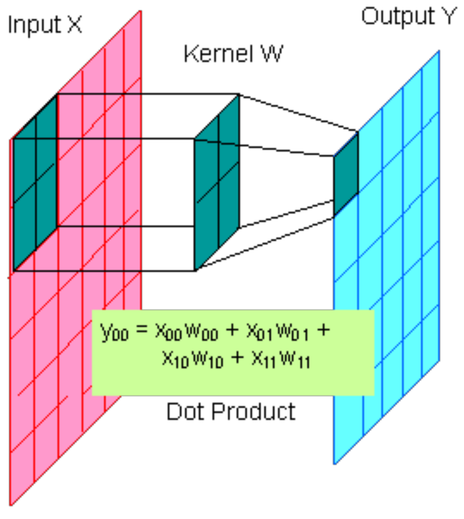


Convolution | Non-Linearity | Normalization | Pooling

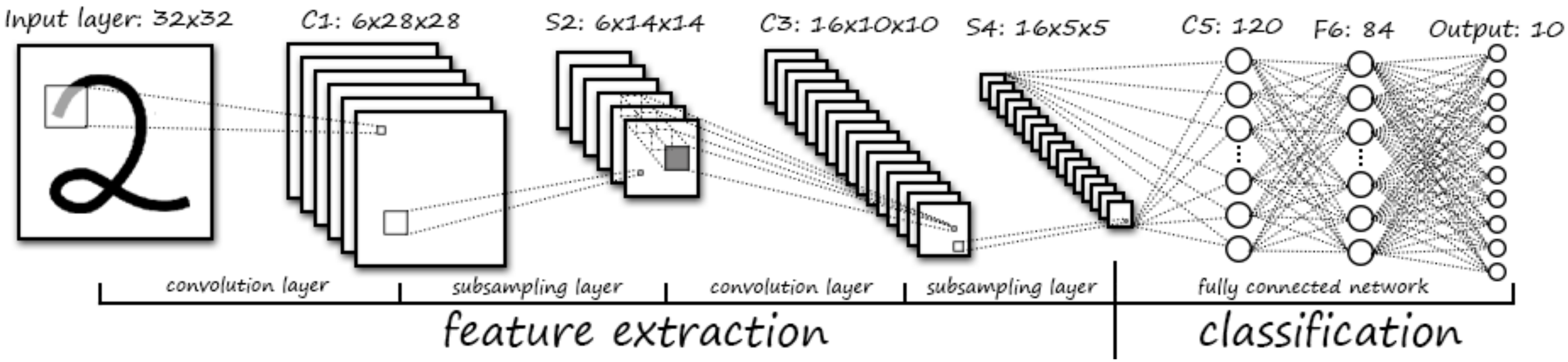
Convolutional Neural Networks



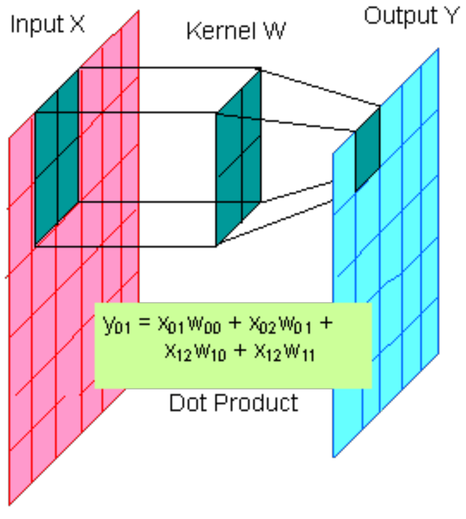
Convolution | Non-Linearity | Normalization | Pooling



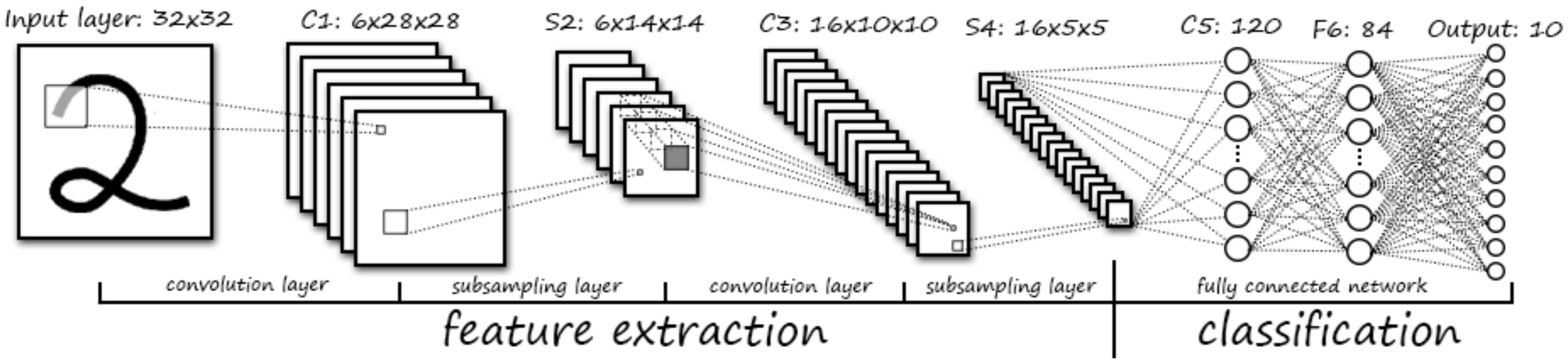
Convolutional Neural Networks



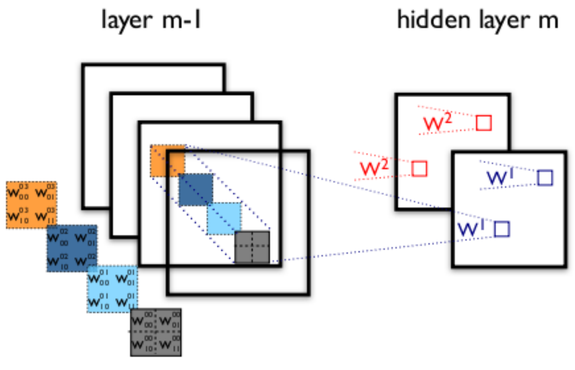
Convolution | Non-Linearity | Normalization | Pooling



Convolutional Neural Networks

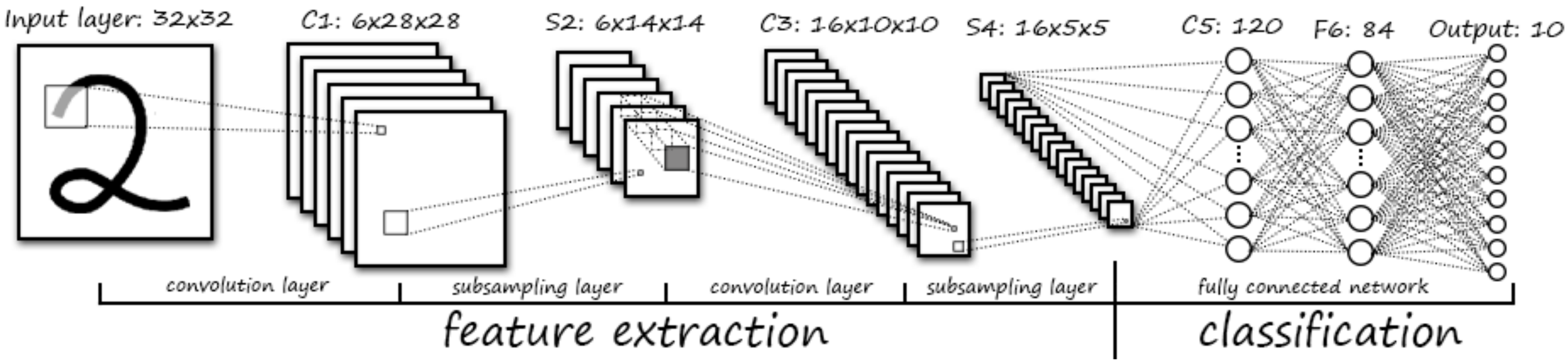


Convolution | Non-Linearity | Normalization | Pooling

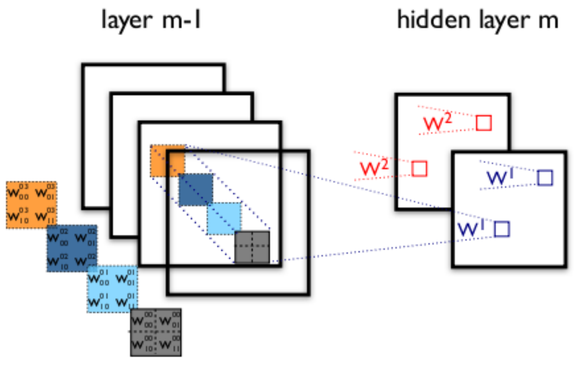


$$y = F * x + b$$

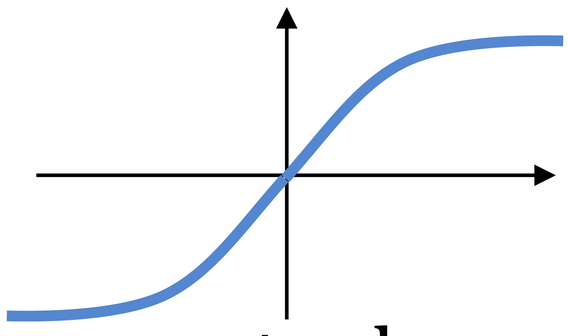
Convolutional Neural Networks



Convolution | Non-Linearity | Normalization | Pooling

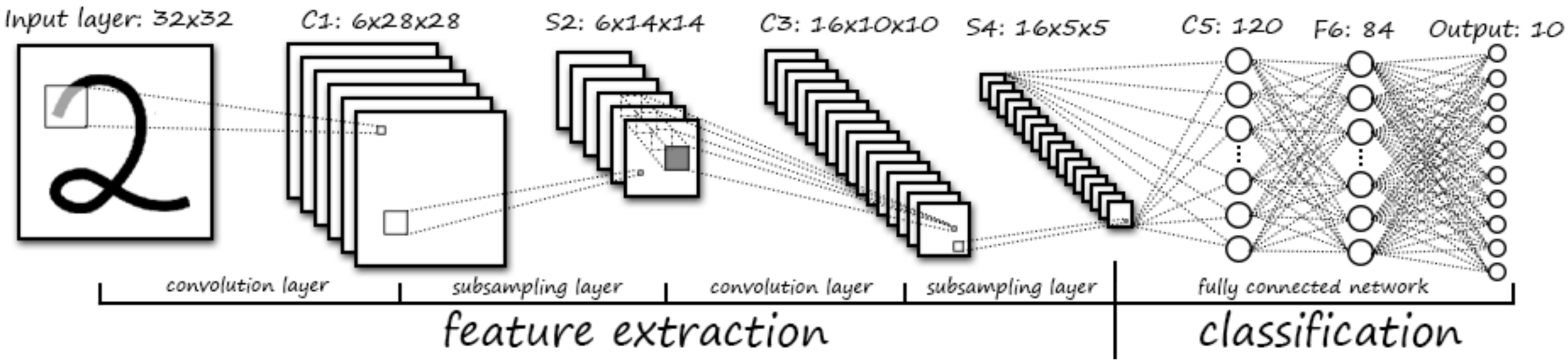


$$y = F * x + b$$

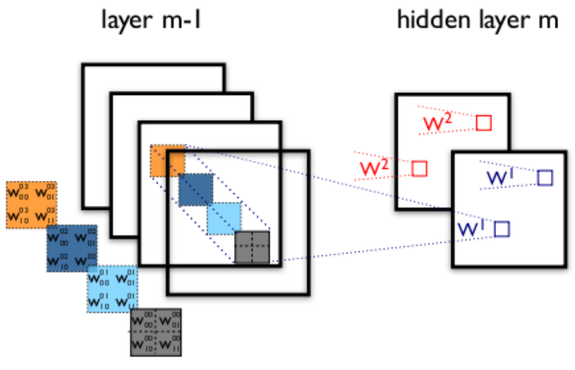


$$y = \tanh x$$

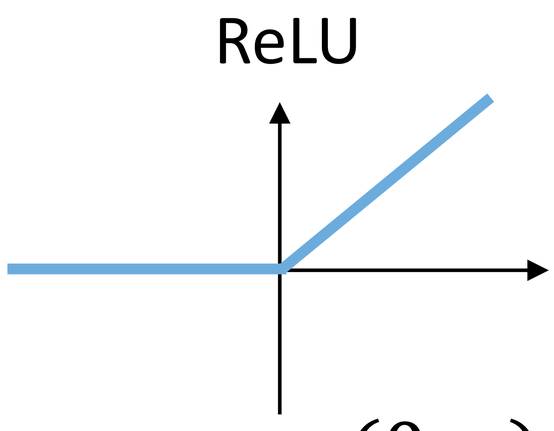
Convolutional Neural Networks



Convolution | Non-Linearity | Normalization | Pooling

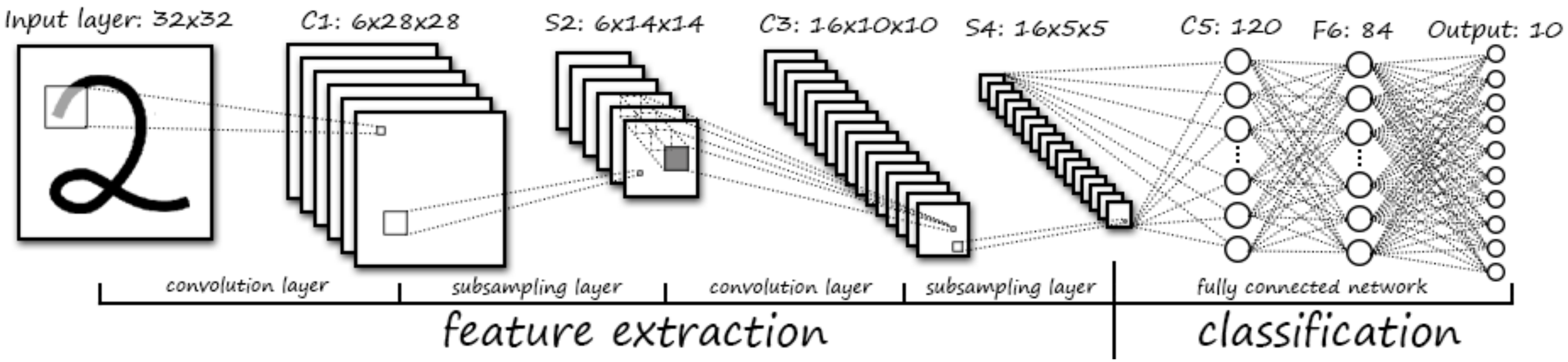


$$y = F * x + b$$

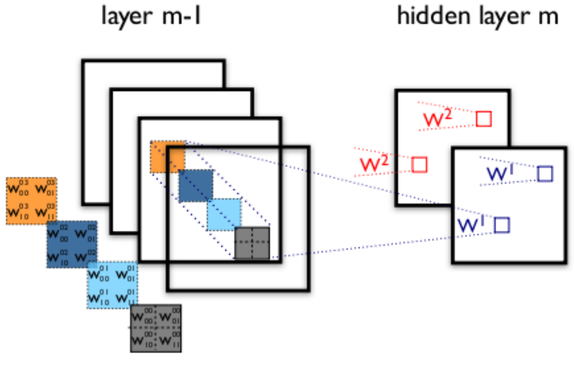


$$y = \max(0, x)$$

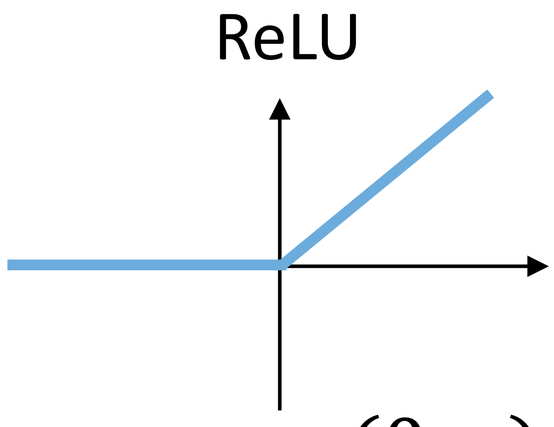
Convolutional Neural Networks



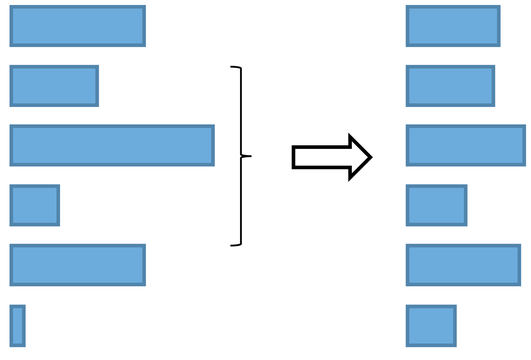
Convolution | Non-Linearity | Normalization | Pooling



$$y = F * x + b$$

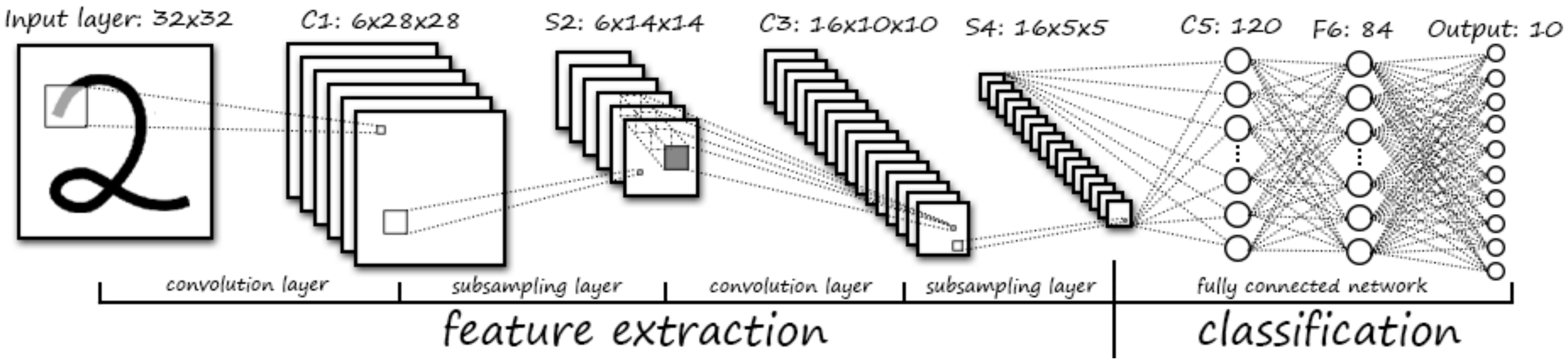


$$y = \max(0, x)$$

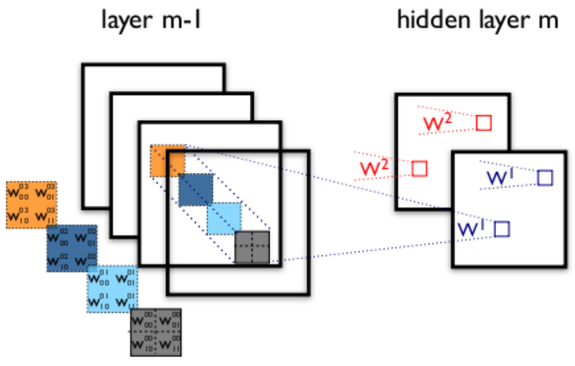


$$Local\ l^2$$

Convolutional Neural Networks

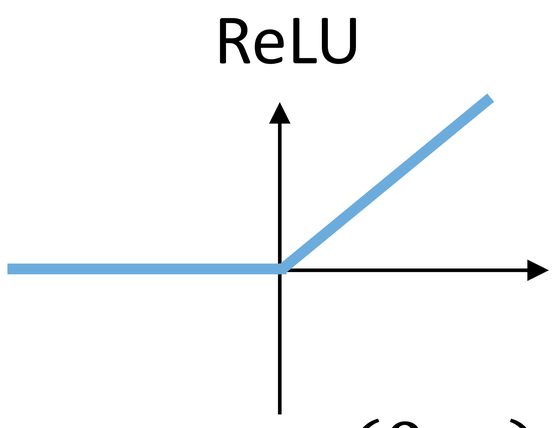


Convolution



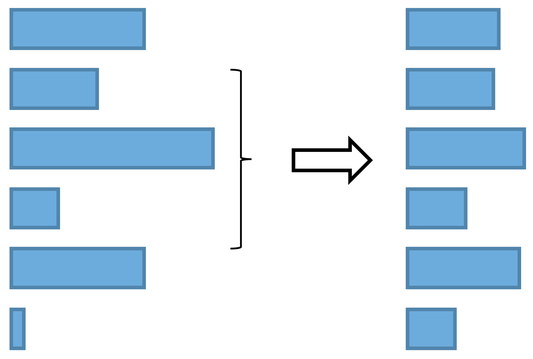
$$y = F * x + b$$

Non-Linearity



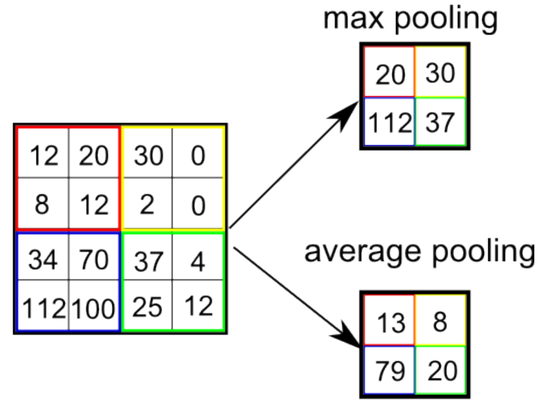
$$y = \max(0, x)$$

Normalization



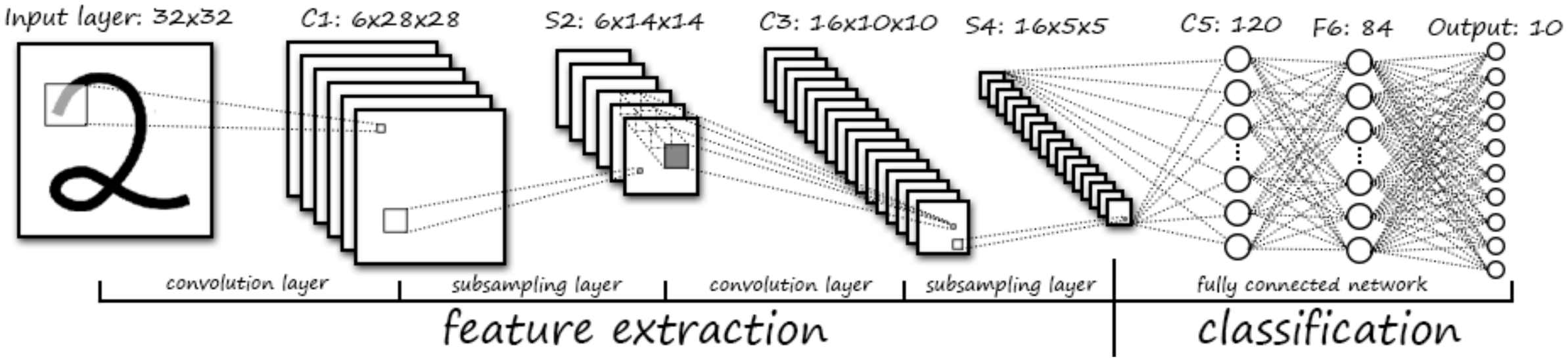
$$Local\ l^2$$

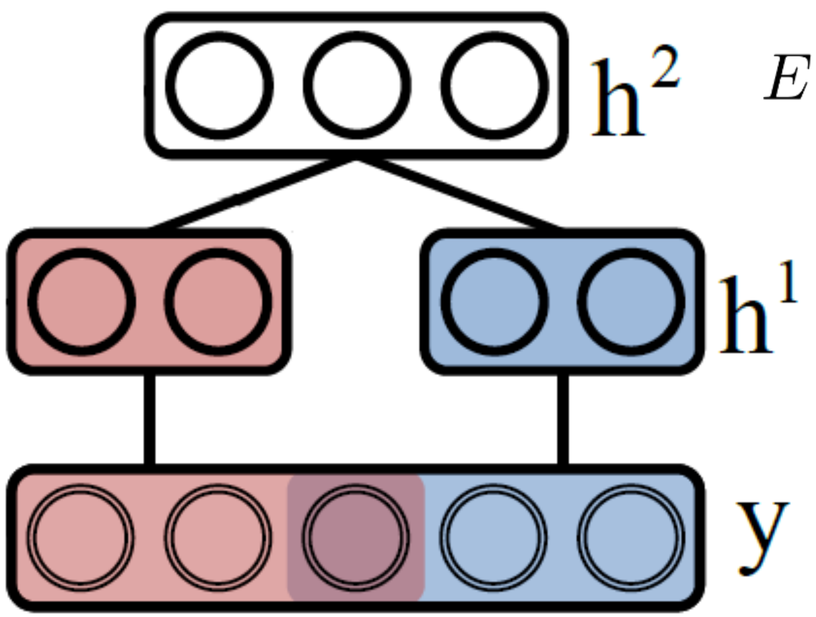
Pooling



$$y_{ijk} = \max_{pq \in \Omega_{ij}} x_{pqk}$$

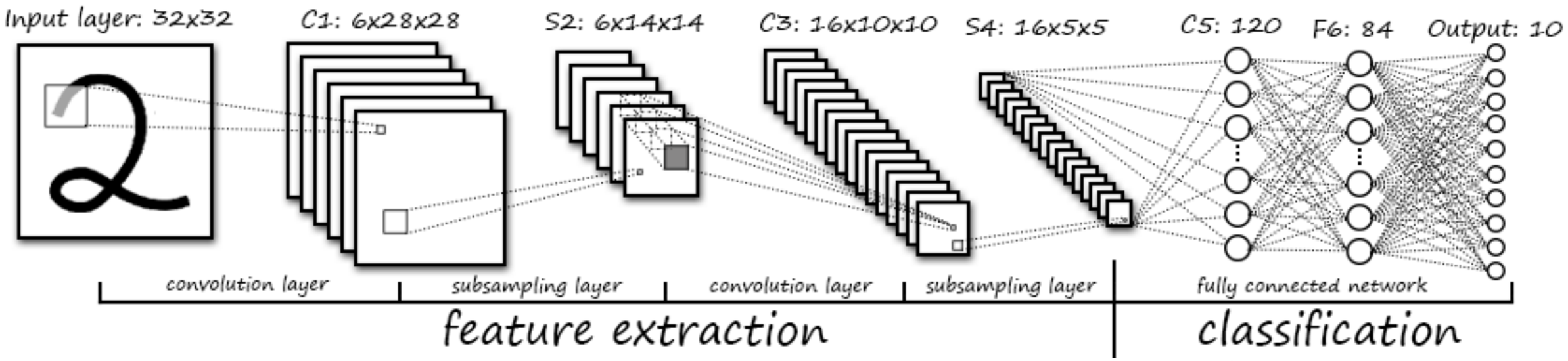
CNNs inside MMBMs

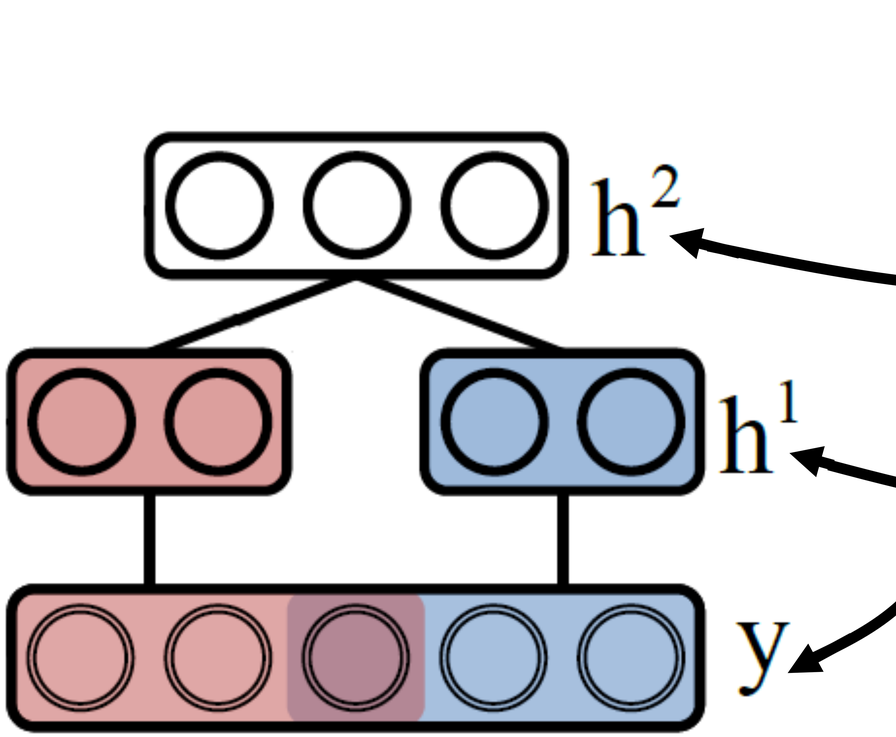




$$E(y, h^1, h^2, x) = -y^T W^1 h^1 - h^{1T} W^2 h^2 - h^{1T}(V^1 x^1 + c^1) - h^{2T}(V^2 x^2 + c^2) - y^T(V^0 x^0 + b)$$

CNNs inside MMBMs





$$E(y, h^1, h^2, x) = -y^T W^1 h^1 - h^{1T} W^2 h^2 - h^{1T}(V^1 x^1 + c^1) - h^{2T}(V^2 x^2 + c^2) - y^T(V^0 x^0 + b)$$

Experiments

First we tried to replace only hidden features x^1 with the features from CNNs trained on “ImageNet” dataset.

$$V^1 x^1 = \begin{pmatrix} v_{11}^0 & \cdots & v_{1n}^0 \\ v_{21}^0 & \cdots & v_{2n}^0 \\ \vdots & \ddots & \vdots \\ v_{m1}^0 & \cdots & v_{mn}^0 \end{pmatrix} \begin{pmatrix} x_1^1 \\ \vdots \\ x_n^1 \end{pmatrix}$$

Experiments

First we tried to replace only hidden features x^1 with the features from CNNs trained on “ImageNet” dataset.

Results of MMBM1 on Weizmann Horses dataset					
	No GC		With GC		**Linear classifier**
	AP	IoU	AP	IoU	
vgg-f – conv4	89.41	78.50	89.71	79.09	**85.81**
vgg-f – conv3	89.31	78.33	89.58	78.84	**85.49**
vgg-f – conv2	89.10	77.94	89.59	78.87	**84.92**
hogs	89.03	77.83	89.63	78.95	**84.53**
vgg-f – conv5	89.05	77.87	89.43	78.59	**84.36**
vgg-f – conv1	88.84	77.47	89.31	78.34	**83.98**
vgg-f – full1	88.25	76.41	89.00	77.80	**81.57**
vgg-f – full2	87.95	75.87	88.65	77.16	**79.96**

Experiments

Then we tried to replace visible features x^0 maintaining as much locality as we can.

$$V^0 x^0 = \begin{pmatrix} v_1^0 & \dots & v_n^0 \end{pmatrix} \begin{pmatrix} x_{11}^0 & \dots & x_{m1}^0 \\ x_{12}^0 & \dots & x_{m2}^0 \\ \vdots & \ddots & \vdots \\ x_{1n}^0 & \dots & x_{mn}^0 \end{pmatrix}$$

Results were approximately 4-5% less than with hogs.

Possible explanation: we doesn't get enough generality or enough number of features.

Experiments

Then we tried to replace visible features x^0 in the manner of hidden features ignoring locality.

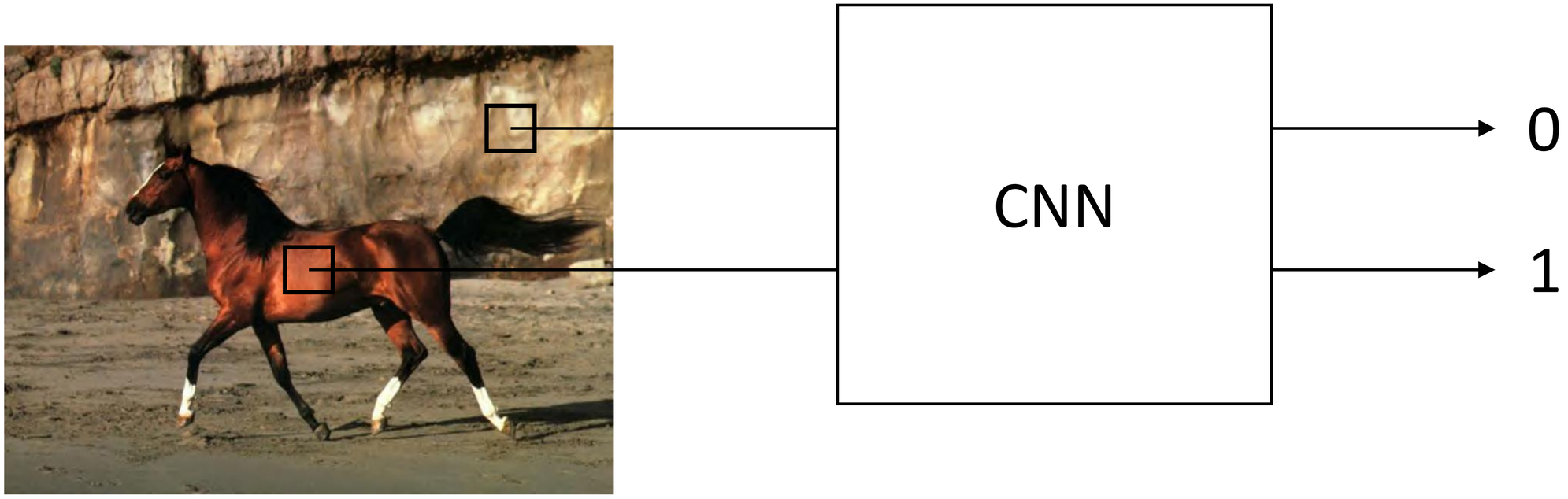
$$V^0 x^0 = (v_1^0 \quad \dots \quad v_n^0) \begin{pmatrix} x_{11}^0 & \dots & x_{m1}^0 \\ x_{12}^0 & \dots & x_{m2}^0 \\ \vdots & \ddots & \vdots \\ x_{1n}^0 & \dots & x_{mn}^0 \end{pmatrix} \Rightarrow V^0 x^0 = \begin{pmatrix} v_{11}^0 & \dots & v_{1n}^0 \\ v_{21}^0 & \dots & v_{2n}^0 \\ \vdots & \ddots & \vdots \\ v_{m1}^0 & \dots & v_{mn}^0 \end{pmatrix} \begin{pmatrix} x_1^0 \\ \vdots \\ x_n^0 \end{pmatrix}$$

Same results: 4-5% less than with hogs.

Possible explanation: for a fixed pixel we get a non-linear combination of relevant features and some noise.

Experiments

Finally we trained our own “patch-to-label” cnn and got features from there.



Experiments

Results on “Weizmann Horses” dataset

MMBM1				
	No GC		With GC	
	AP	IoU	AP	IoU
hogs	**89.27**	**78.27**	90.09	79.83
cnn	88.49	76.84	**91.09**	**81.68**

MMBM2				
	No GC		With GC	
	AP	IoU	AP	IoU
hogs	89.68	78.98	90.07	79.74
cnn	**91.52**	**82.35**	**92.73**	**84.81**

Experiments

Results on “Caltech-UCSD Birds 200” dataset

MMBM1				
	No GC		With GC	
	AP	IoU	AP	IoU
hogs	86.31	73.26	86.92	74.29
cnn	**87.02**	**74.66**	**88.23**	**76.66**

Conclusions

- Features from CNNs are good for very different tasks.
- Even small fine-tuning on the specific problem can give significant quality improvement.

Work to be done

1. Fine-tune big CNNs for hidden units prediction.
2. Try to learn a small CNN for hidden units prediction.
3. Implement joint learning.

Problems

1. ICM is very unstable!
2. Training of CNNs takes huge amount of time even on GPU.
3. Implementation of GPUs synchronization.