Classification of Hand-Written Digits Using Scattering Convolutional Network

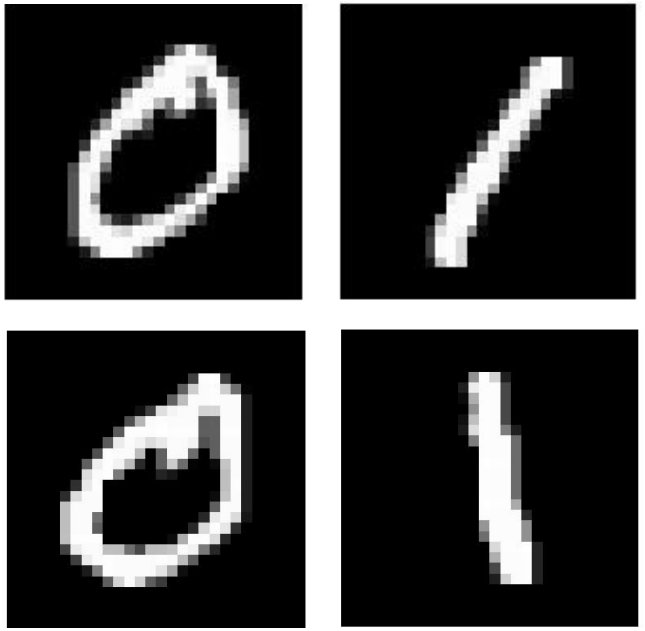
Dongmian Zou

Advisor: Professor Radu Balan

Co-Advisor: Dr. Maneesh Singh (Verisk)

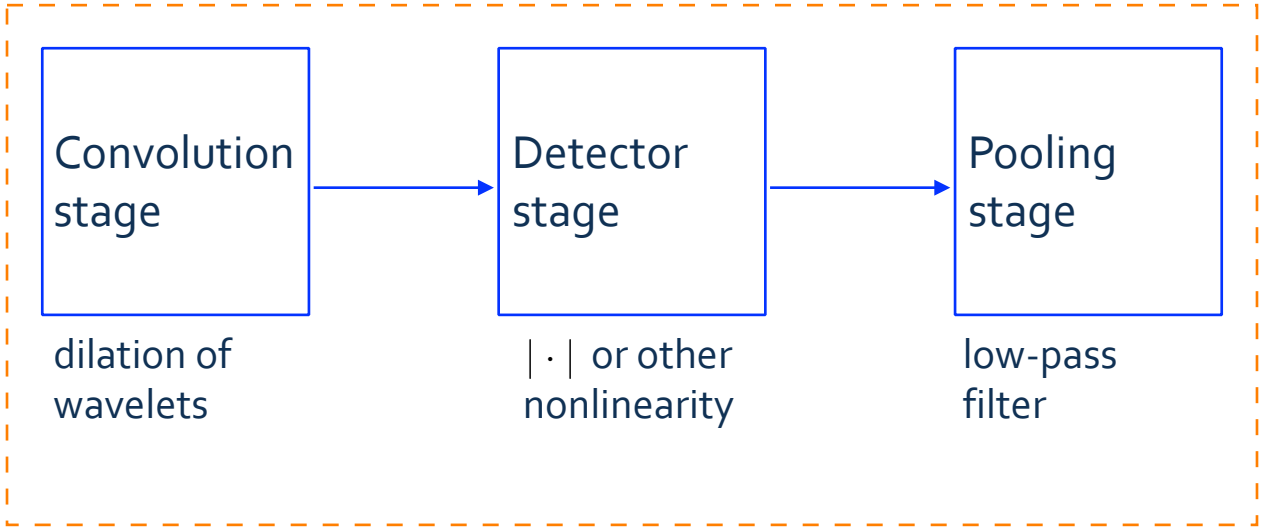
Overview

- Image classification
- Feature extractor
  - Convolutional neural network
- Machine learning techniques
  - Gradient descent
  - Back-propagation
  - Support Vector Machine (SVM)



Typical MNIST training data of 28-by-28 pixels

Convolutional Network

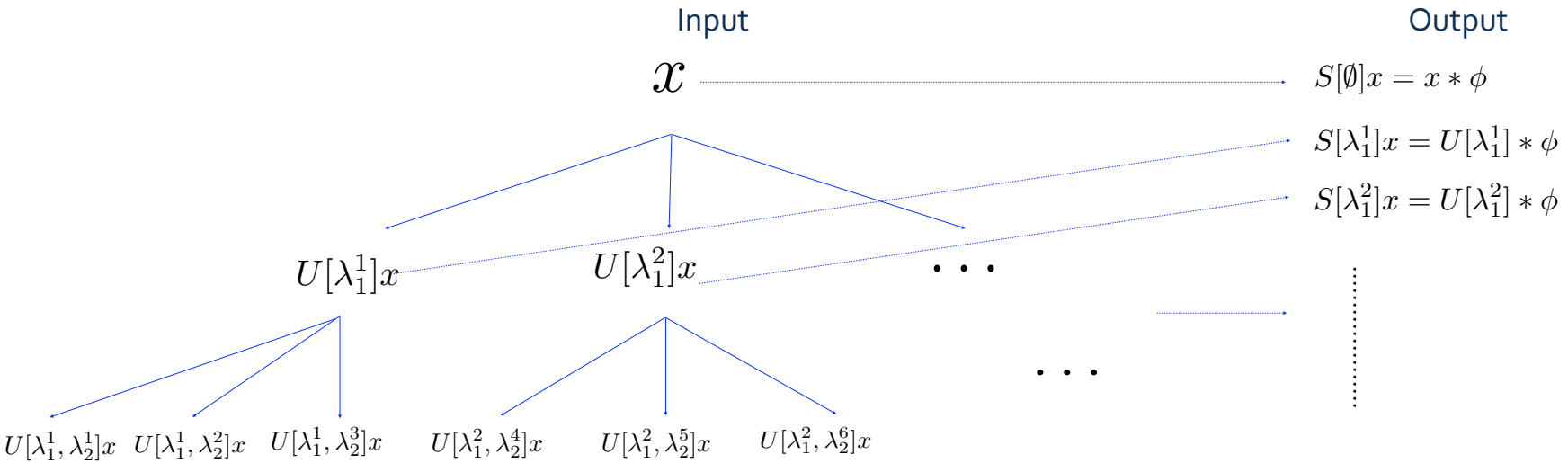


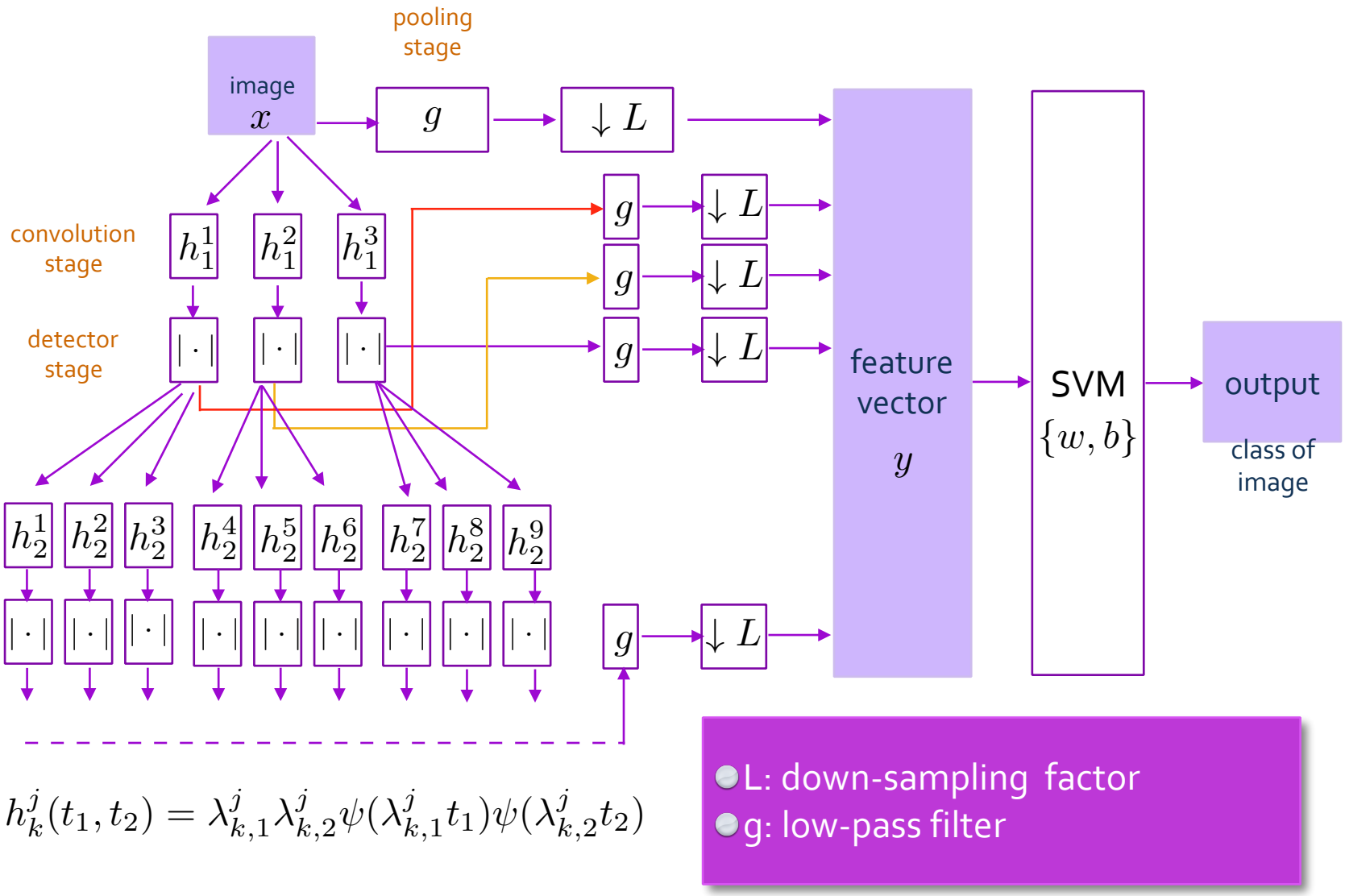
Scattering Convolutional Network

- Scattering propagator $\psi_\lambda(t) = \lambda^d \psi(\lambda t) \qquad q = (\lambda_1, \lambda_2, \cdots, \lambda_m)$

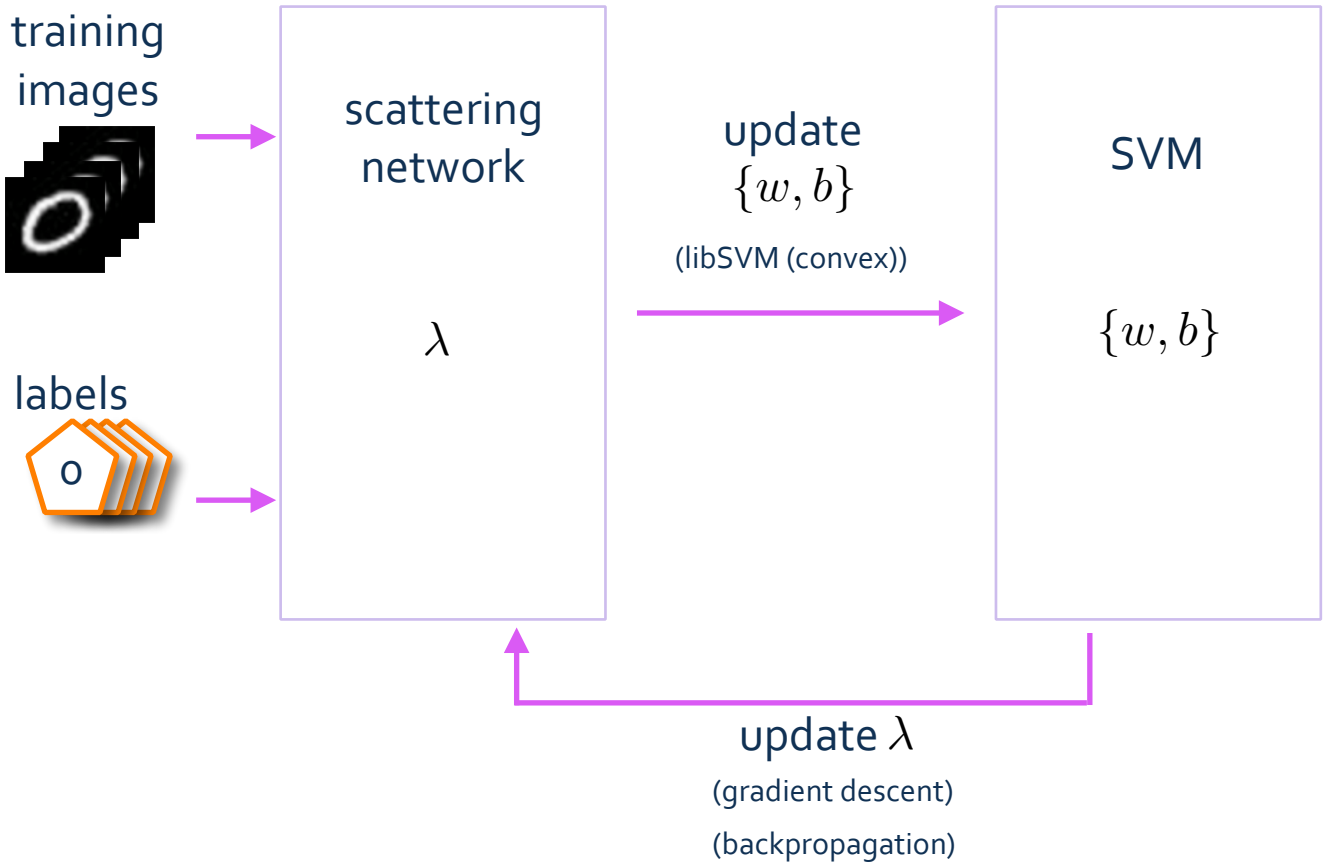
$$U[q]x = |||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| \cdots \psi_{\lambda_m}|$$

- Scattering transform $S[q]x = U[q]x * \phi_J$




pooling stage
image x
g
$\downarrow L$
convolution stage
h_1^1 h_1^2 h_1^3
detector stage
$|\cdot|$
h_2^1 h_2^2 h_2^3 h_2^4 h_2^5 h_2^6 h_2^7 h_2^8 h_2^9
feature vector y
SVM $\{w, b\}$
output
class of image
$h_k^j(t_1, t_2) = \lambda_{k,1}^j \lambda_{k,2}^j \psi(\lambda_{k,1}^j t_1) \psi(\lambda_{k,2}^j t_2)$
L: down-sampling factor
g: low-pass filter

Training


training images
labels
0
scattering network
λ
update $\{w, b\}$
(libSVM (convex))
SVM
$\{w, b\}$
update λ
(gradient descent)
(backpropagation)

The Optimization Problem

$$\min_{\boldsymbol{\lambda};w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} l(y_n, a_n; w, b) \ ,$$

where

$$l(y, a; w, b) = \max(0, 1 - a(b + \langle w, y\rangle)) \ ,$$

and y is the grouping of the following

$$\begin{aligned} y_0 &= x * g \ ; \\ y_1^j &= \left|x * h_1^j\right| * g \ , 1 \le j \le 3 \ ; \\ y_2^j &= \left|\left|x * h_1^{\lceil j/3 \rceil}\right| * h_2^j\right| * g \ , 1 \le j \le 9 \ , \end{aligned}$$

where the two-dimensional filters h_k^j is parametrized as

$$h_k^j(t_1, t_2) = \lambda_{k,1}^j \lambda_{k,2}^j \psi(\lambda_{k,1}^j t_1)\psi(\lambda_{k,2}^j t_2) \ .$$

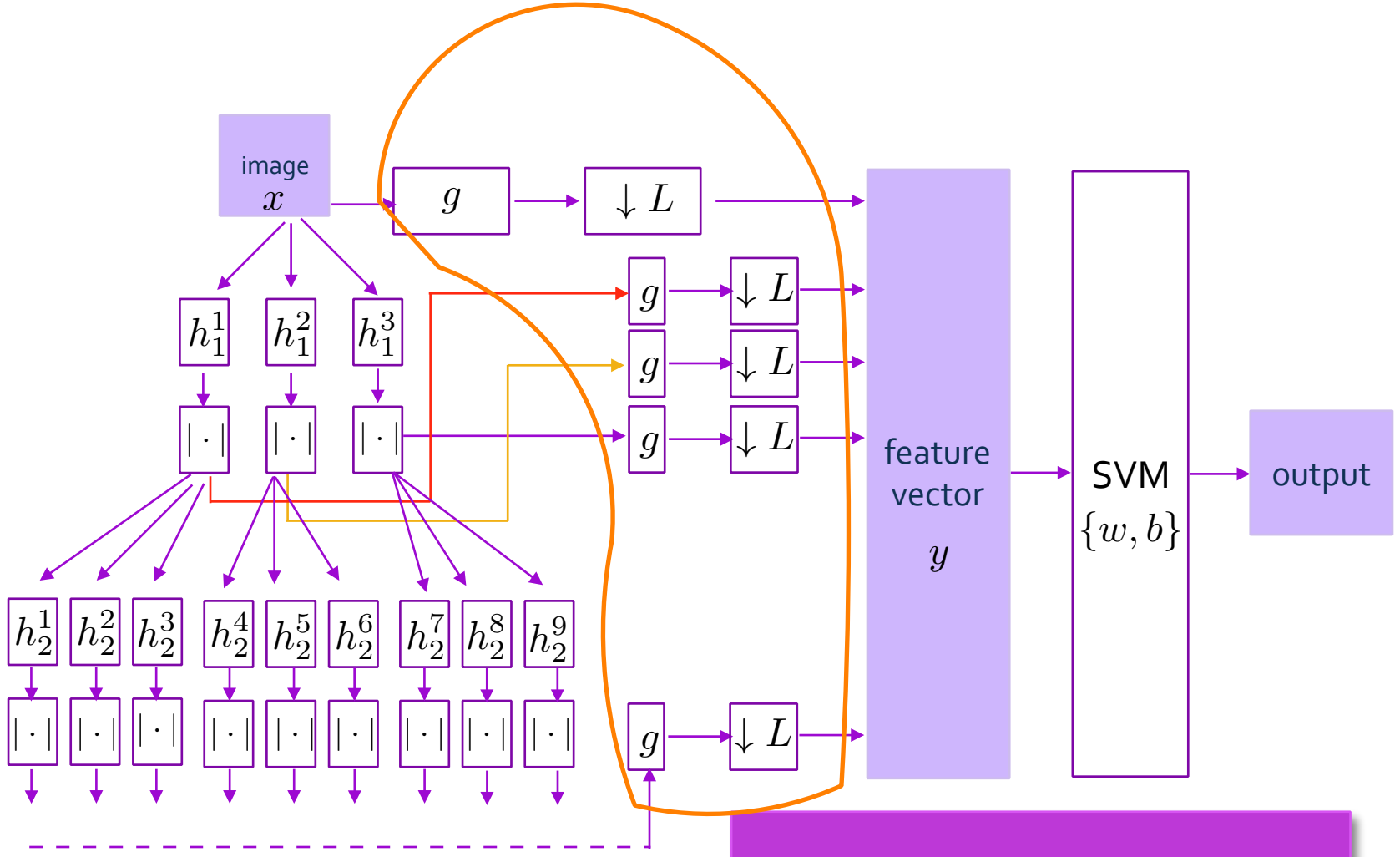
Algorithm

Algorithm 1: The algorithm for network training

Start with learning rate η, regularization parameter C ;
randomly generate $\boldsymbol{\lambda}, w, b$;
while *stop criterion not met* **do**
    sample N examples $\{x_1, x_2, \cdots, x_N\}$ from the training set;
    propagate forward to get $\{y_1, y_2, \cdots, y_N\}$;
    call libSVM with input $\{y_1, y_2, \cdots, y_N\}$ and C;
    update $w, b \leftarrow$ output of libSVM;
    set $\boldsymbol{r} = 0$;
    for $n = 1$ *to* N **do**
        compute $\nabla_{\boldsymbol{\lambda}} l(\boldsymbol{\lambda}; x_n)$;
        $\boldsymbol{r} \leftarrow \boldsymbol{r} + \nabla_{\boldsymbol{\lambda}} l(\boldsymbol{\lambda}; x_n)$;
    update $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \eta \boldsymbol{r}$;
    adapt η accordingly.

Implementation

- Personal Laptop
  - CPU: 2GHz Intel Core i7
  - Memory: 8 GB 1600 MHz DDR3
  - OS X El Capitan Version 10.11
- MATLAB R2015b
- MNIST database
  - Publicly available at http://yann.lecun.com/exdb/mnist/
  - Image: 28×28 pixels (each pixels value between 0~255)
  - 60,000 training examples
  - 10,000 testing examples



$$h_k^j(t_1, t_2) = \lambda_{k,1}^j \lambda_{k,2}^j \psi(\lambda_{k,1}^j t_1)\psi(\lambda_{k,2}^j t_2)$$

- L: down-sampling factor
- g: low-pass filter
- use cross validation for both

Parameter Selection

- We need to decide which g and L to use
- We use 5400 training examples for each digit
  - 3600 for training / 1800 for testing
- Select parameter based on the testing result

Parameter Selection

- Stochastic gradient descent

	L = 3 / LP*	L = 4 / LP	L = 5 / LP	L = 3 / AV	L = 4 / AV	L = 5 / AV
Running time (hour)	14.5	10.2	8.5	5.5	5.2	5.3
Error rate (%) for "0"	1.67	7.44	13.5	0.72	3.17	3.56
Error rate (%) for "1"	0.39	2.11	0	0.11	0	0.11
Sum of loss for "0"	399.3	555.1	976.8	253.6	456.6	494.3
Sum of loss for "1"	202.1	454.5	336.7	135.7	122.9	63.8

* LP = using a low-pass filter in place of g
* AV = doing local averaging in place of g

Parameter Selection

- Deterministic method

	L = 3 / AV	L = 4 / AV	L = 5 / AV
Running time (hour)	22.4	21.6	22.4
Error rate (%) for “0”	0.17	0.28	0.28
Error rate (%) for “1”	0.17	0	0.28
Sum of loss for “0”	8.6	10.6	15.7
Sum of loss for “1”	8.5	4.1	18.0

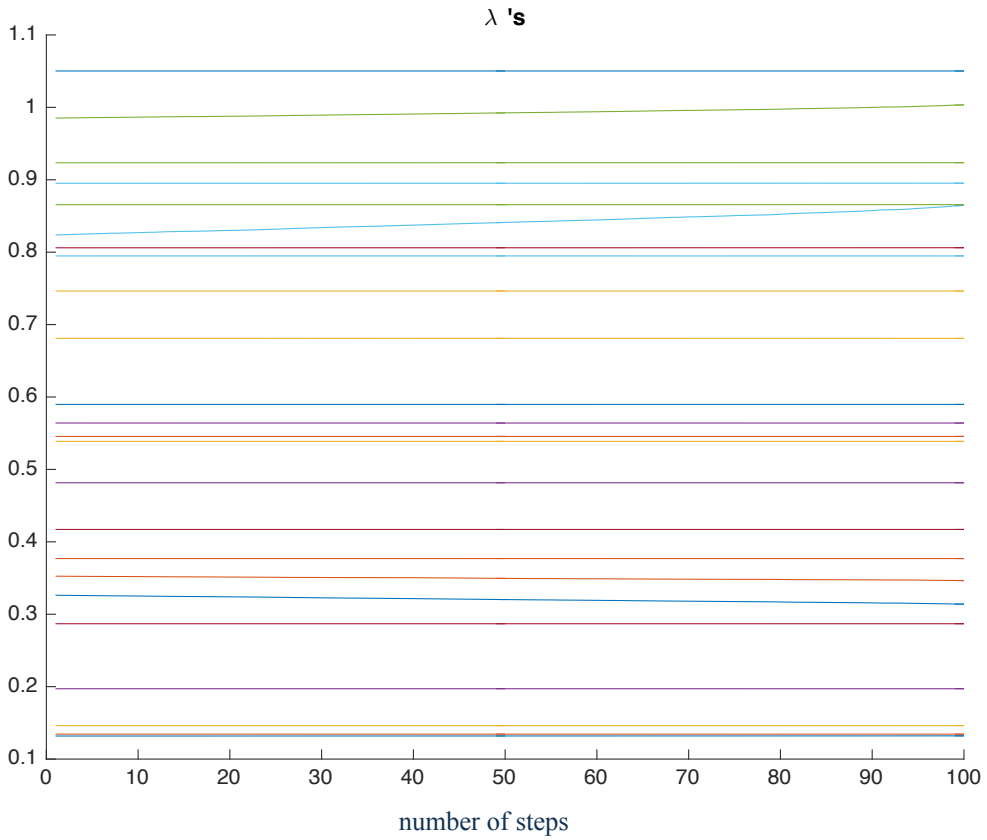
Validation

- Use existing software, then compare the results
- MatConvNet toolbox
  - publicly available at http://www.vlfeat.org/matconvnet/
  - main elements:
    - computational blocks: convolution, local averaging, derivatives, etc.
    - wrappers: structure (topology) of the neural network

Validation

- We use the computation blocks and the data structure from MatConvNet
- Run MatConvNet and our program separately
  - Start with the same λ
  - Take a small sample of training images (100)
  - Iterate a small number of steps (100)
  - Compare how λ is updated

Validation



This graph shows how the 24 lambdas change with iteration steps based on 100 training data (out of more than 5000) for each digit.

Multi-class Classification

- One-vs-one
  - Build a classifier for each pair of classes (45 in total)
  - Decision rule: a voting strategy
    - for each pair of classes, choose one class
    - the vote of the class is added by one
    - take the class that has the largest number of votes
  - Train the feature extractor

$$\min_{\substack{\boldsymbol{\lambda};w_q,b_q \\ q=1,\cdots,45}} \quad \frac{1}{2}\sum_{q=1}^{45}\|w_q\|^2 + C\sum_{n=1}^{N}\sum_{q:a_n\in A_q} l(y_n, a_n; w_q, b_q)$$

Nonlinearity: Square

- Scattering transform is stable to deformation
  - $\frac{||y(D(x_0)) - y(x_0)||}{||x_0||}$ is small
- To achieve this, we do not necessarily want to use $|\cdot|$
- For instance, we can use $|\cdot|^2$ in place of $|\cdot|$
  - Future work: convert the problem to a convex problem
- Theory guarantees that under certain conditions we still have deformation stability
- We use gradient descent to train the square network

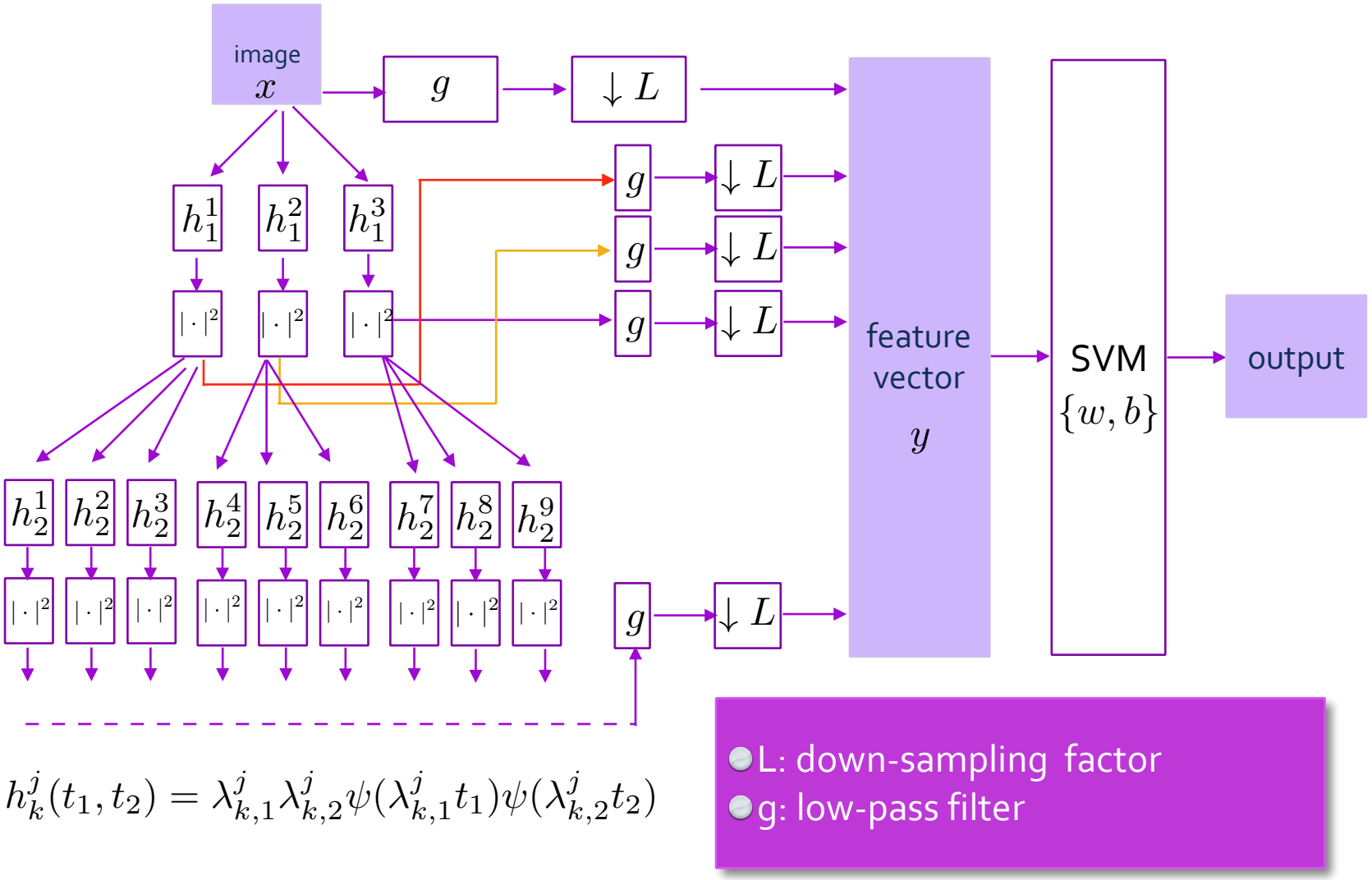


image
x
g
$\downarrow L$
h_1^1 h_1^2 h_1^3
$|\cdot|^2$
feature vector
y
SVM
$\{w, b\}$
output
h_2^1 h_2^2 h_2^3 h_2^4 h_2^5 h_2^6 h_2^7 h_2^8 h_2^9
$h_k^j(t_1, t_2) = \lambda_{k,1}^j \lambda_{k,2}^j \psi(\lambda_{k,1}^j t_1) \psi(\lambda_{k,2}^j t_2)$
L: down-sampling factor
g: low-pass filter

Testing

- Compute the error rate
  - error rate = # (incorrectly classified images) / # (testing images)
- Test the stability results
  - scattering convolutional network
    - (approximately) invariant to translation
    - stable to deformation

Error Rate

- Test for 2 classes

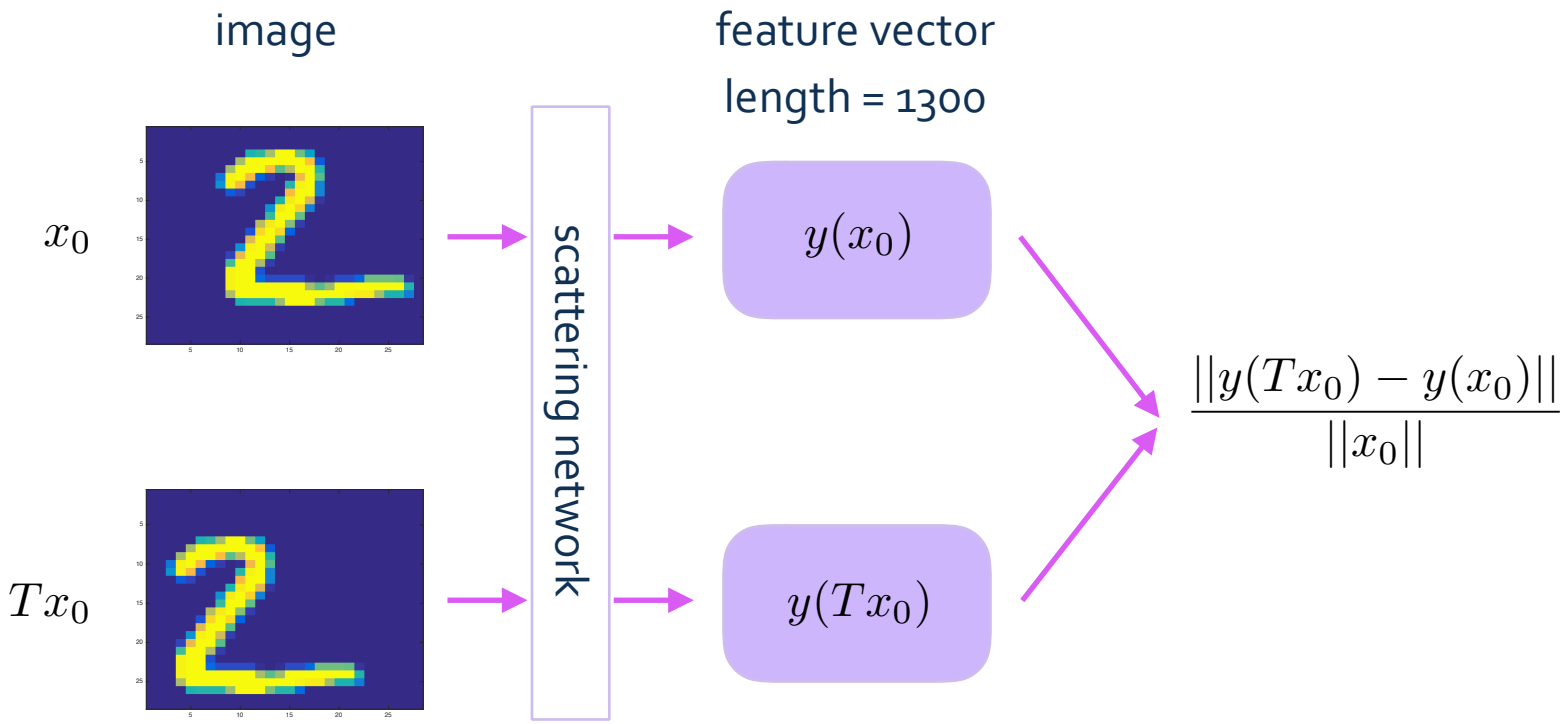
error rate (%)	Class 0	Class 1
stochastic gradient descent	0.75	0
deterministic gradient descent	0.13	0.13
libSVM	0	0

Error Rate

- Test for multi-classes

error rate (%)	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
stochastic gradient descent	11.5	2.75	32.87	49.88	39.12	42.63	21.62	38.5	38.37	41
deterministic gradient descent	1.87	1.12	6	8.25	5.5	10.25	4.5	8	12	10.87
libSVM	3	1.62	6.25	7.5	4.87	9.37	5.25	7.75	10.87	10
Square (deterministic)	1.63	1.25	6.12	7.88	4.25	10.62	3.62	5.75	10.13	9.38

Invariance to Translation



Invariance to Translation

$$\frac{||y(Tx_0) - y(x_0)||}{||x_0||}$$

$|\cdot|$

Translation (pixel)	min (%)	average (%)	max (%)
0.1 *	5.28	9.30	15.87
0.2	5.76	9.98	16.78
0.5	7.19	12.17	19.90
1	4.51	8.37	14.55
2	9.10	15.49	25.39
5	22.87	34.69	46.02

$|\cdot|^2$

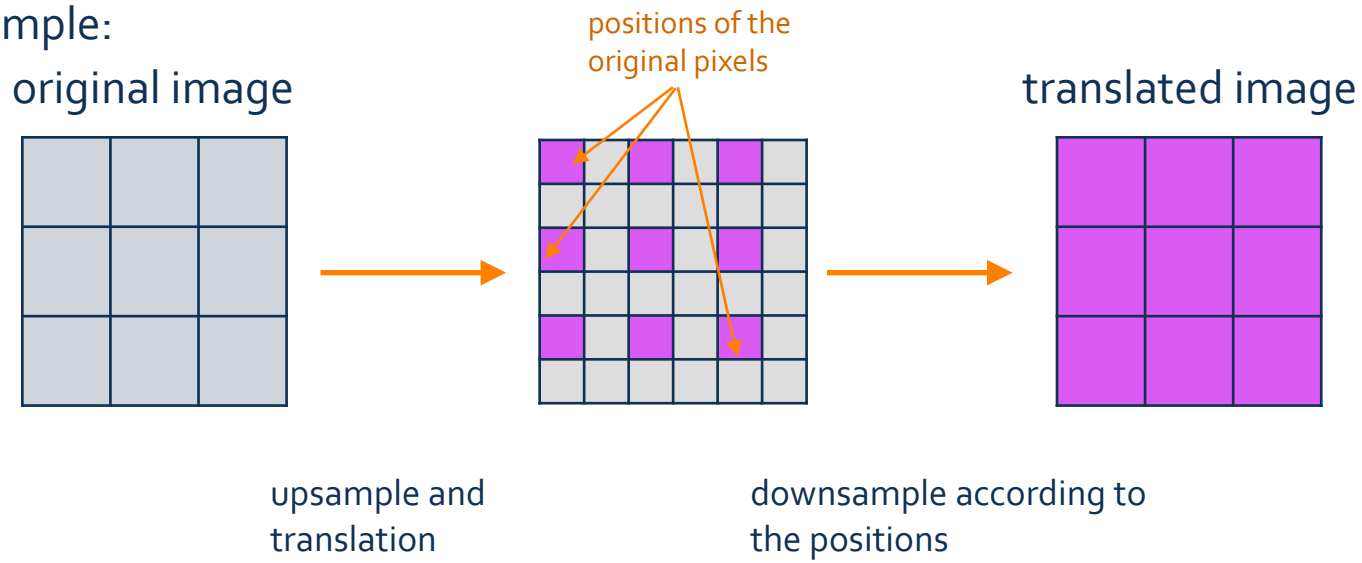
Translation (pixel)	min (%)	average (%)	max (%)
0.1	4.92	9.00	15.50
0.2	5.44	9.71	16.51
0.5	6.98	11.97	19.70
1	9.33	15.97	25.32
2	13.96	22.91	34.71
5	22.75	34.62	45.86

* The fractional pixels are done by linear upsampling first, then do the translation for an integer number of pixels, then downsample to the original resolution

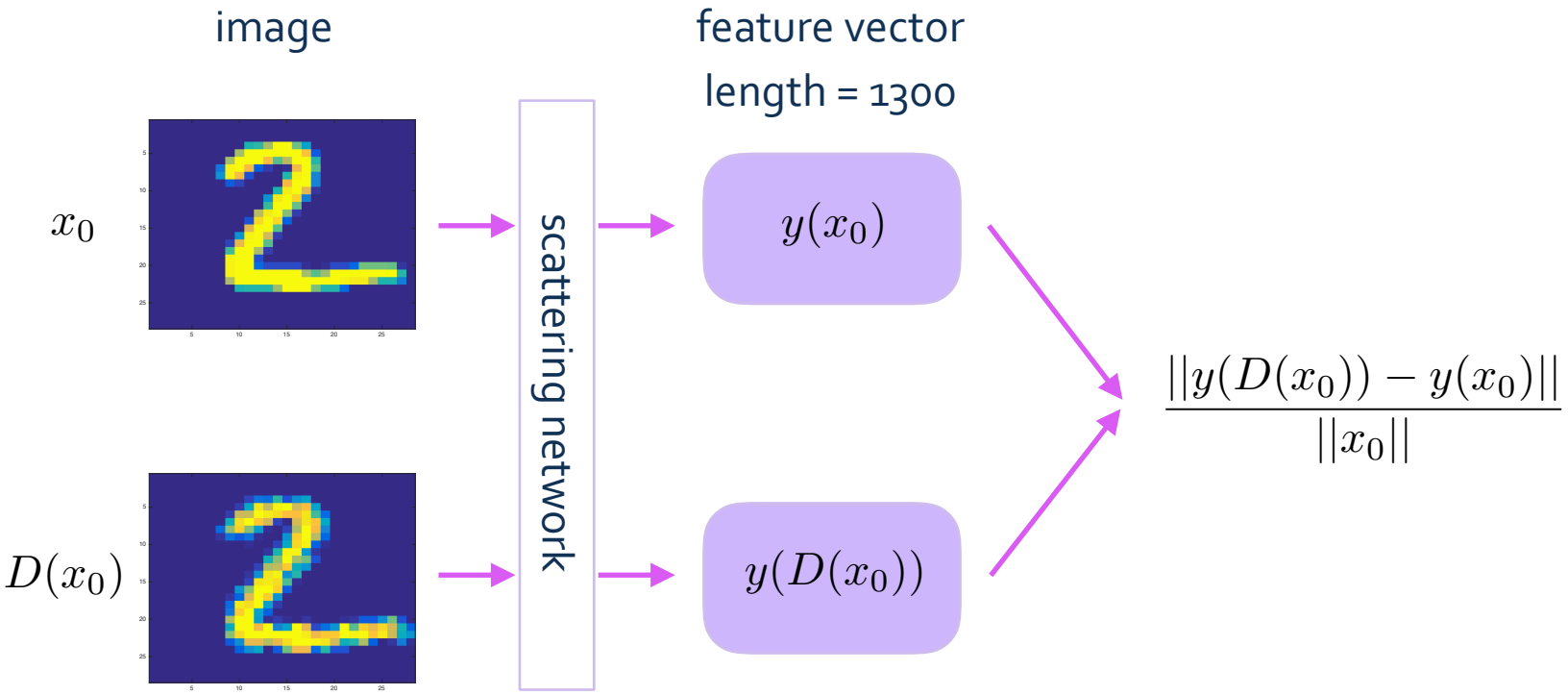
Invariance to Translation

- The fractional pixels are done by linear upsampling first, then do the translation for an integer number of pixels, then downsample to the original resolution

Example:



Invariance to Deformation

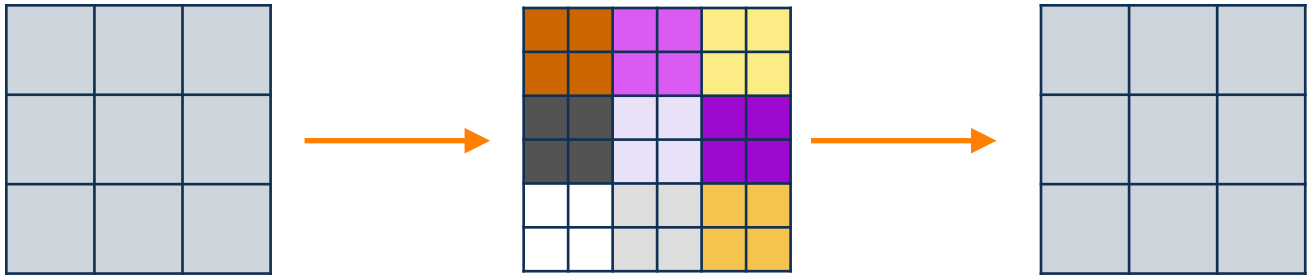


Invariance to Deformation

- Deformation Type 1: take an image, upsample, then randomly downsample to the original size

Example:

original image



deformed image

upsample with factor 2 (2 by 2)

randomly pick one pixel from each colored block

Invariance to Deformation

- Deformation Type 1: take an image, upsample, then randomly downsample to the original size

$$\frac{||y(D(x_0)) - y(x_0)||}{||x_0||}$$

$|\cdot|$

upsample factor	min (%)	average (%)	max (%)
2	2.45	4.71	8.99
3	3.08	6.02	11.30
4	6.46	12.96	20.20
5	6.53	13.29	21.15

$|\cdot|^2$

upsample factor	min (%)	average (%)	max (%)
2	5.78	11.28	17.92
3	6.13	12.37	19.34
4	6.42	12.93	20.14
5	6.58	13.26	20.88

Invariance to Deformation

- Deformation Type 2: take an image, set part of the pixel values to be zero (incomplete deformation)

$$\frac{||y(D(x_0)) - y(x_0)||}{||x_0||}$$

$|\cdot|$

test	min (%)	average (%)	max (%)
randomly zero out	4.37	9.71	15.86
take a column out	7.41	11.22	20.03
take a row out	6.85	9.89	15.52

$|\cdot|^2$

test	min (%)	average (%)	max (%)
randomly zero out	4.36	9.21	14.73
take a column out	7.27	10.95	19.88
take a row out	6.42	9.55	15.15

Summary

- We developed a software for training (gradient descent) and implementing a scattering network
  - We tested it for binary and multi-class classification problem
  - We tested its stability to translation and deformation
  - We validated it using MatConvNet
- Potential future work
  - Use other methods for training
  - Use other structures

Bibliography

[1] Y. Bengio, I. J. Goodfellow and A. Courville, *Deep Learning*, Book in preparation for MIT press, 2015.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer, 2006.

[3] J. Bruna and S. Mallat, *Invariant Scattering Convolution Networks*, Pattern Analysis and Machine Intelligence, IEEE Transactions 35(8)(2013), 1872-1886.

[4] C. Chang and C. Lin, *LIBSVM : a library for support vector machines,* ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[5] S. Mallat, *Group Invariant Scattering*, Comm. Pure Appl. Math., 65 (2012): 1331-1398.

[6] A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems 2 (2012): 1097-1105.

[7] C. Szegedy et al, *Going Deeper with Convolutions*, Open Access Version available at http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

[8] A. Vedaldi and K. Lenc, *MatConvNet - Convolutional Neural Networks for MATLAB*, Proc. of the ACM Int. Conf. on Multimedia, 2015.

[9] T. Wiatowski and H. Bölcskei, *Deep Convolutional Neural Networks Based on Semi-Discrete Frames*, Proc. of IEEE International Symposium on Information Theory (ISIT), Hong Kong, China, pp. 1212-1216, June 2015.

THANK YOU