Mid-year Progress Report

Classification of Hand-Written Digits Using Scattering Convolutional Network

Dongmian Zou

Advisor: Professor Radu Balan

Co-Advisor: Dr. Maneesh Singh (SRI)

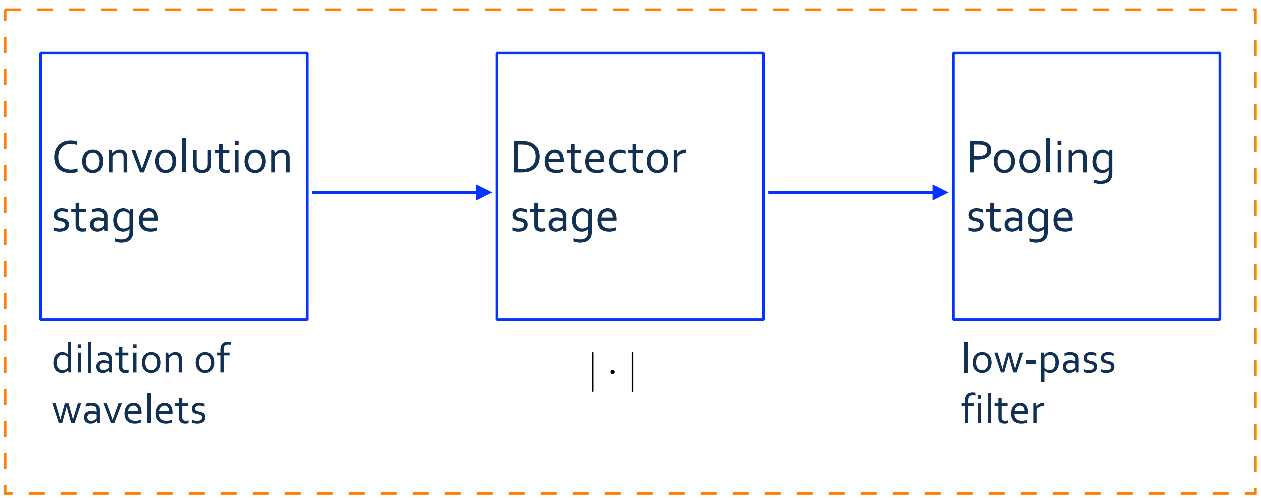
Background

Overview

- Image classification
  - Hand-written digits
- Feature extractor
  - Convolutional neural network
- Machine learning techniques

Typical MNIST training data of 28-by-28 pixels

Scattering Convolutional Network

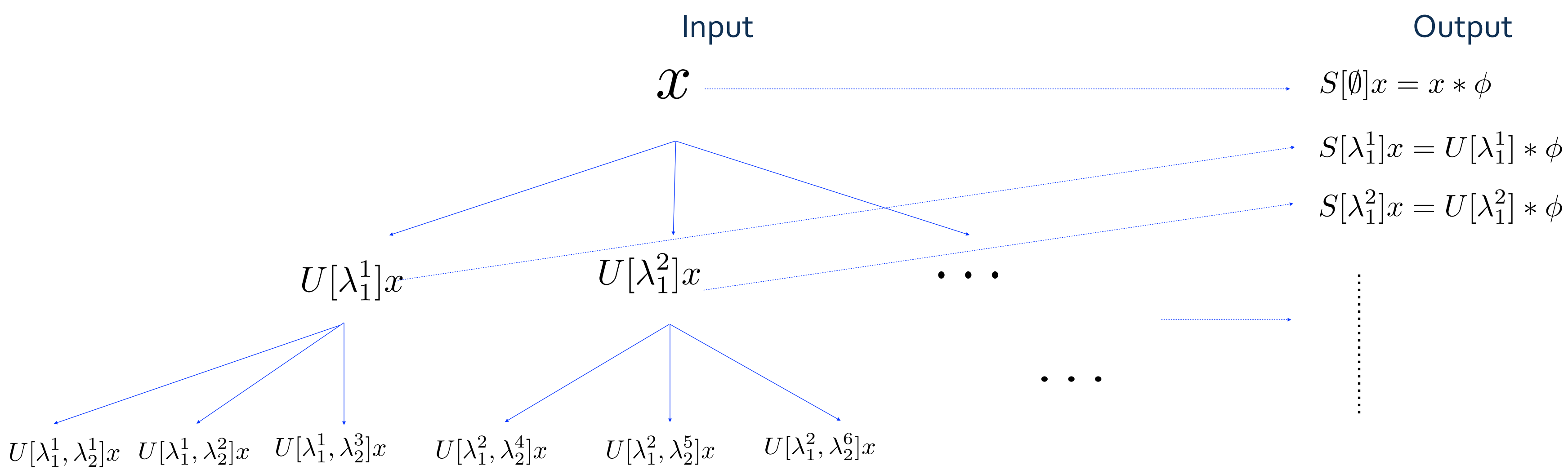


Scattering Convolutional Network

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

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

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



Machine Learning Techniques

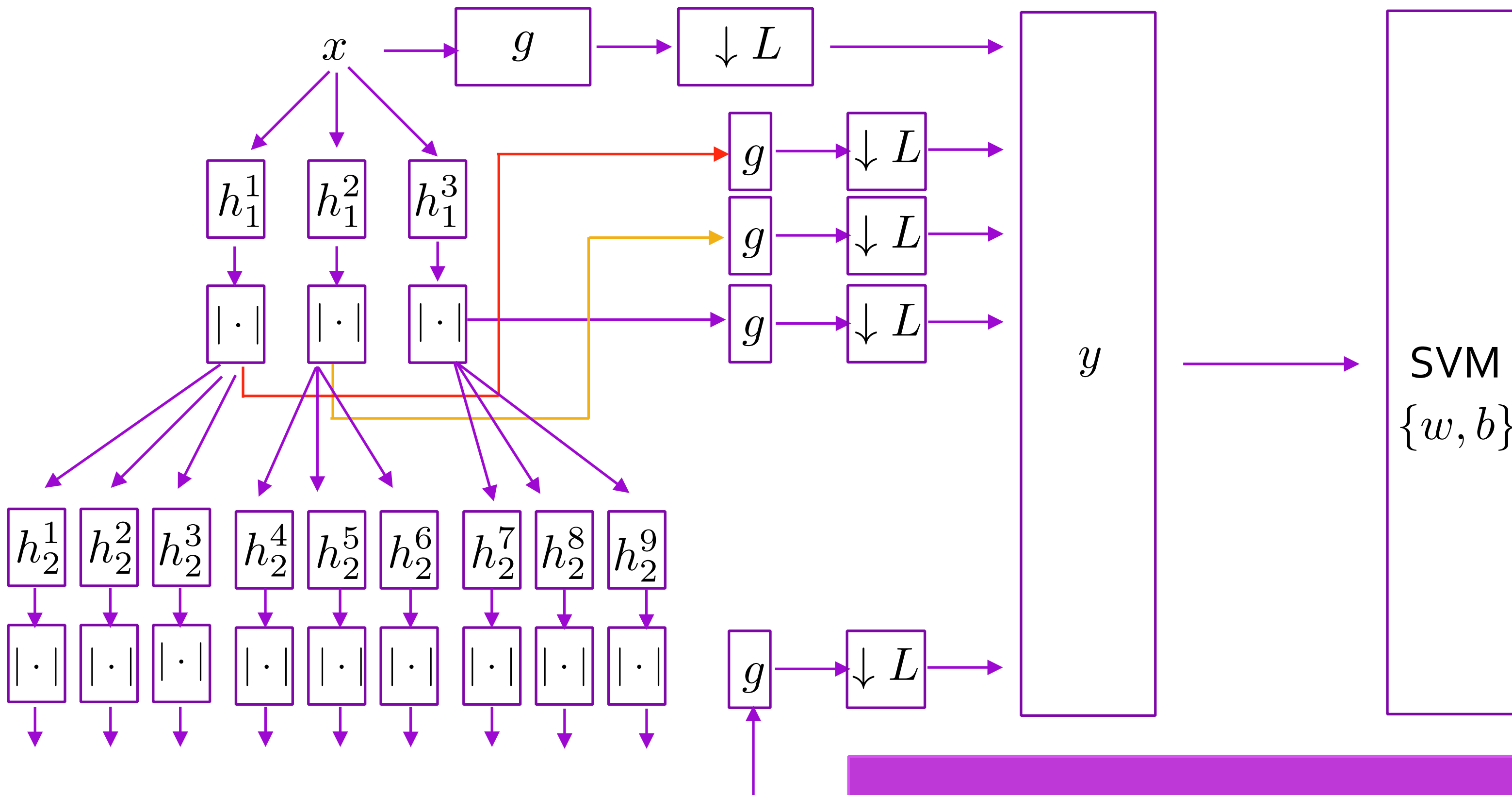
- Gradient descent
- Back-propagation
- Support Vector Machine (SVM)

Input: N pairs of (x, t)

Unknown parameters: $\{\lambda, w, b\}$

images of 28-by-28 pixels

0 labels (which class x belongs to)



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

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

Approach and Implementation

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

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

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

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

Two Step Optimization

- The above problem is non-convex
  - difficult to solve with respect to {λ, w, b}
- We can iterate between two problems
  - First, fix the filters in the scattering network, train the SVM ← convex
    - We use libSVM library to train the SVM
      - Publicly available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Second, fix w and b, train the filters in the scattering network ← easier

Convolution Layer

- Filters to train: h_k^j
  - Each filter contains two parameters $\lambda_{k,1}^j, \lambda_{k,2}^j$

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

- Gradient descent

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
  - Run from terminal window (no GUI)
- 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

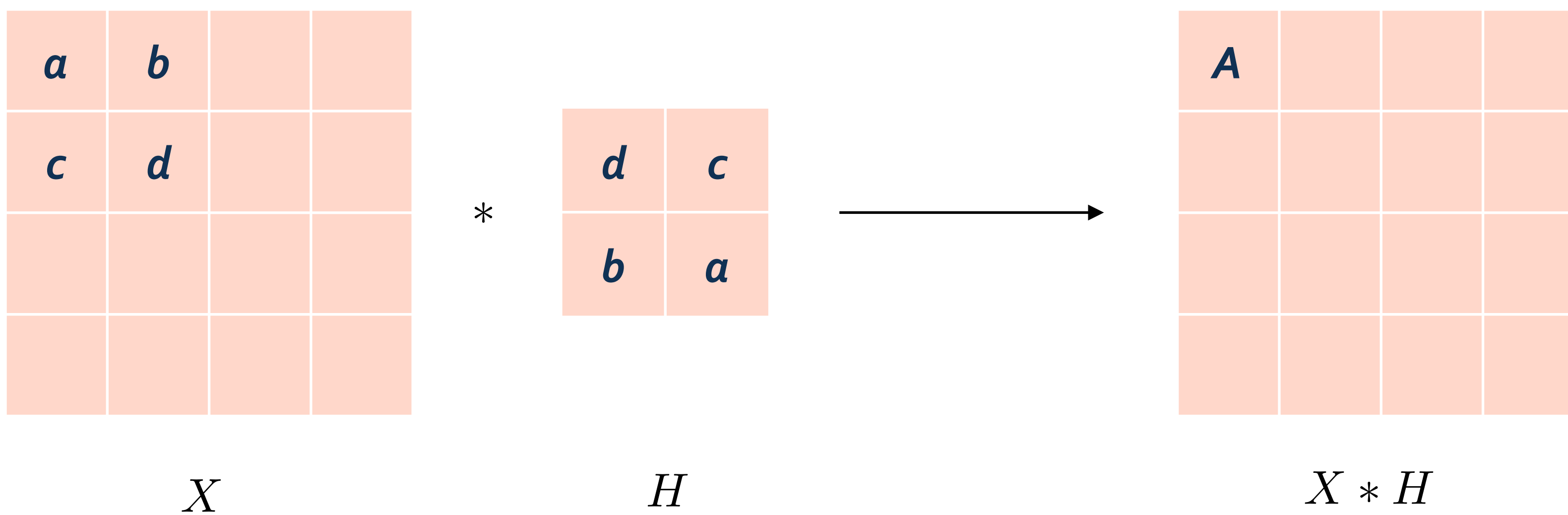
Implementation

- Use 5400 training examples for each digit
  - cross-validation for model selection
  - 3600 for training / 1800 for testing
- Rescale pixel values from [0,255] to [0,1]
- Start with $\eta = 1$, $C = 1$
- Randomly generate $\lambda > 0.1$
  - For gradient descent, adjust η to make sure that $\lambda > 0.1$

Forward Propagation

- Convolution

$$(X * H)(t_1, t_2) = \sum_{s_1} \sum_{s_2} X(t_1 - s_1, t_2 - s_2) H(s_1, s_2)$$



LIBSVM

- Install libSVM
  - MATLAB generates ".mex" file to call ".C" file
  - MATLAB 2015b does not detect Xcode7
  - Need to download xcode7_mexopts.zip
- Label: rescale {0,1} to {-1,1}
- Update w and b from the output of libSVM

Back Propagation

- The derivative of $|\cdot|$
  - $F(t) = (|t|^2 + \epsilon^2)^{1/2}$
  - $F'(t) = \dfrac{t}{(|t|^2 + \epsilon^2)^{1/2}}$
- The partial derivative of the loss function
  - $L(y, a; w, b) = \begin{cases} 0.5 - a(b + \langle w, y \rangle) & \text{, if} \quad a(b + \langle w, y \rangle) \leq 0; \\ 0.5(1 - a(b + \langle w, y \rangle))^2 & \text{, if} \quad 0 < a(b + \langle w, y \rangle) \leq 1; \\ 0 & \text{, otherwise.} \end{cases}$

$$\nabla_y L(y, a; w, b) = \begin{cases} -aw & \text{, if} \quad a(b + \langle w, y \rangle) \leq 1; \\ 0 & \text{, otherwise.} \end{cases}$$

Back Propagation

$$\frac{\partial L}{\partial \lambda_{k,i}^j} = \left\langle \nabla_{y_k^j} L, \frac{\partial y_k^j}{\partial \lambda_{k,i}^j} \right\rangle$$

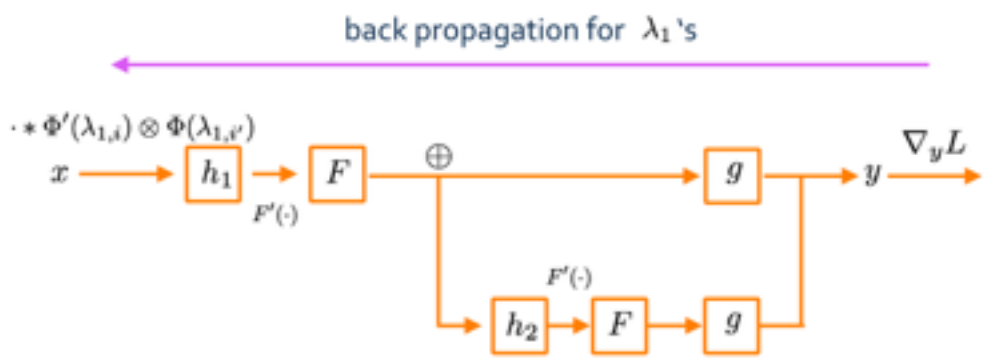
$$\frac{\partial y_1^j}{\partial \lambda_{1,i}^j} = \left[F'\left(x * (\Psi(\lambda_{1,i}^j) \otimes \Psi(\lambda_{1,i'}^j))\right) \odot \left(x * (\Psi'(\lambda_{1,i}^j) \otimes \Psi(\lambda_{1,i'}^j))\right)\right] * g \ ;$$

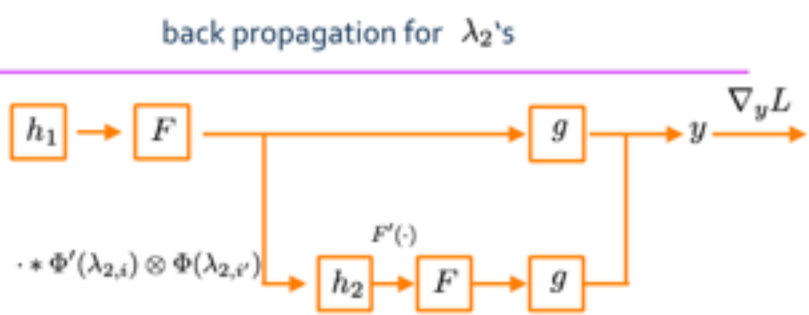
$$\frac{\partial y_2^{3j-\iota}}{\partial \lambda_{1,i}^j} = \Bigg\{F'\left(F\left(x * (\Psi(\lambda_{1,i}^j) \otimes \Psi(\lambda_{1,i'}^j))\right) * \left(\Psi(\lambda_{2,i}^{3j-\iota}) \otimes \Psi(\lambda_{2,i'}^{3j-\iota})\right)\right) \odot$$
$$\Bigg[\left[F'\left(x * (\Psi(\lambda_{1,i}^j) \otimes \Psi(\lambda_{1,i'}^j))\right) \odot \left(x * (\Psi'(\lambda_{1,i}^j) \otimes \Psi(\lambda_{1,i'}^j))\right)\right]$$
$$* \left(\Psi(\lambda_{2,i}^{3j-\iota}) \otimes \Psi(\lambda_{2,i'}^{3j-\iota})\right)\Bigg]\Bigg\} * g \ , \quad \text{for } \iota = 1, 2, 3;$$

$$\frac{\partial y_2^j}{\partial \lambda_{2,i}^j} = \Bigg[F'\left(F\left(x * (\Psi(\lambda_{1,i}^{\lceil j/3 \rceil}) \otimes \Psi(\lambda_{1,i'}^{\lceil j/3 \rceil}))\right) * \left(\Psi(\lambda_{2,i}^j) \otimes \Psi(\lambda_{2,i'}^j)\right)\right) \odot$$
$$\left(F\left(\left(x * (\Psi(\lambda_{1,i}^{\lceil j/3 \rceil}) \otimes \Psi(\lambda_{1,i'}^{\lceil j/3 \rceil}))\right)\right) * \left(\Psi'(\lambda_{2,i}^j) \otimes \Psi(\lambda_{2,i'}^j)\right)\right)\Bigg] * g \ .$$

Back Propagation

- Propagate backward
- Take product of all marked values
- Sum when branches merge





Parameter Selection

Det = Deterministic method for training w and b
Sto = Stochastic method for training w and b

LP = Take a low-pass filter for g
AV = Take local average value for convolution with g

L=3 / Sto / LP	error %	sum of loss function
0	1.67	399.3
1	0.39	202.1
Training Time	14.5h	

L=3 / Sto / AV	error %	sum of loss function
0	0.72	253.6
1	0.11	135.7
Training Time	5.5h	

Parameter Selection

Det = Deterministic method for training w and b
Sto = Stochastic method for training w and b

LP = Take a low-pass filter for g
AV = Take local average value for convolution with g

L=4 / Sto / LP	error %	sum of loss function
0	7.44	555.1
1	2.11	454.5
Training Time	10.2h	

L=4 / Sto / AV	error %	sum of loss function
0	3.17	456.6
1	0	122.9
Training Time	5.2h	

Parameter Selection

Det = Deterministic method for training w and b
Sto = Stochastic method for training w and b

LP = Take a low-pass filter for g
AV = Take local average value for convolution with g

L=5 / Sto / LP	error %	sum of loss function
0	13.5	976.8
1	0	336.7
Training Time	8.5h	

L=5 / Sto / AV	error %	sum of loss function
0	3.56	494.3
1	0.11	63.8
Training Time	5.3h	

Parameter Selection

Det = Deterministic method for training w and b
Sto = Stochastic method for training w and b

LP = Take a low-pass filter for g
AV = Take local average value for convolution with g

L=4 / Det / LP	error %	sum of loss function
0	1.56	70.2
1	1.17	54.4
Training Time	10.1h for 1 loop	

...

Project Status

Timeline

2015

- Sept - Oct: define the project, design the algorithm ✓
- Nov: write codes for network training ✓
- Dec: write codes for 2-class classification ✓
  - Multi-class if time permits ○

2016

- Jan - Feb: complete validation ○
- Mar: complete testing ○
- Apr: wrap up the project ○

Validation and Testing

- Validation
  - MatConvNet toolbox
    - publicly available at http://www.vlfeat.org/matconvnet/
- Testing
  - Testing examples in MNIST database
  - Measure: percentage of errors
  - Compare with libSVM results

Deliverables

- Datasets
- Toolboxes
- MATLAB codes
- Trained network
- Results
- Proposal, Reports, Presentation slides, etc.

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