Maxout Network

Goodfellow *et al.*, 2013a, ICML

Presented by:

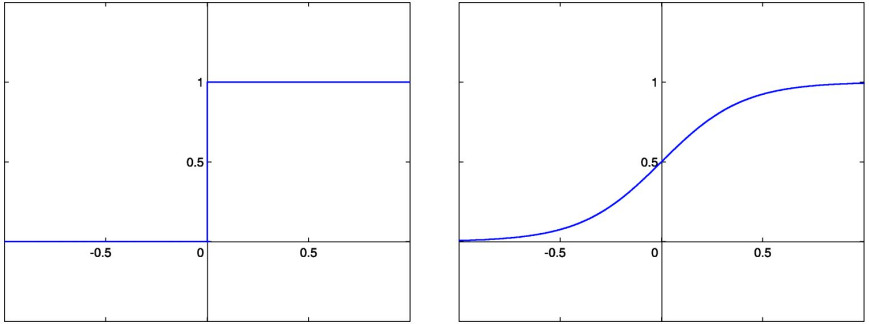
Praveen Chopra, P20EE009

Siddhant Srivastava, MT19SIoT009

Vivek Anand, MT19SIoT005

Idea of Maxout

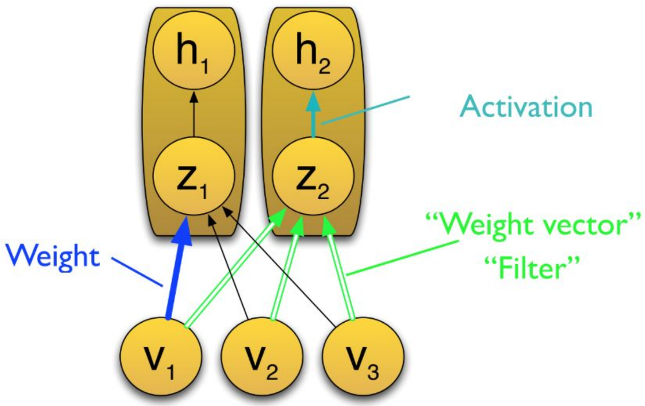
Traditional activation functions



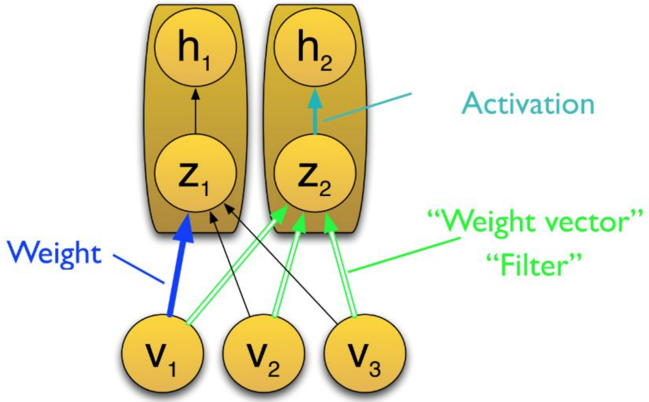
Threshold function   Sigmoid function

Do not use a fixed activation function
But learn the activation function

Idea of MaxOut



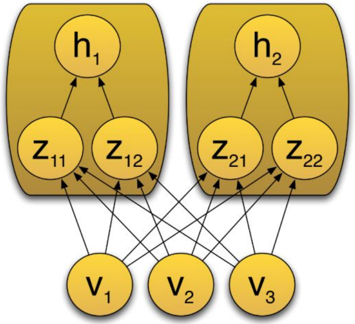
Idea of MaxOut



$$h(x) = \max\left(Z_1, Z_2, \ldots, Zn\right)$$

$$h(x) = \max\left(W_1 \cdot x + b_1, W_2 \cdot x + b_2, \ldots, Wn \cdot x + bn\right)$$

Maxout

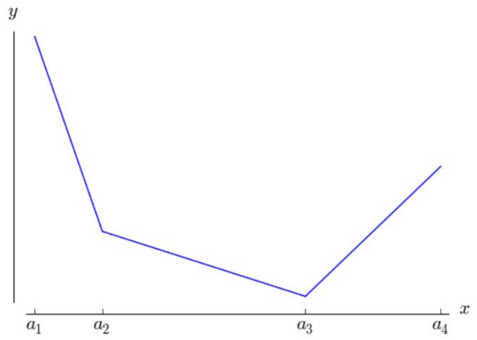


$h_i = \max_j z_{ij}$

The philosophy behind MaxOut

- *Any continuous PWL function can be* approximated arbitrarily well *as a difference of two convex PWL functions.*

- A two hidden unit *h1*(*v*) and *h2*(*v*), maxout network with sufficiently large *k* can approximate any continuous function *f*(*v*) arbitrarily well on the compact domain.



Piecewise linear function

Maxout Layer

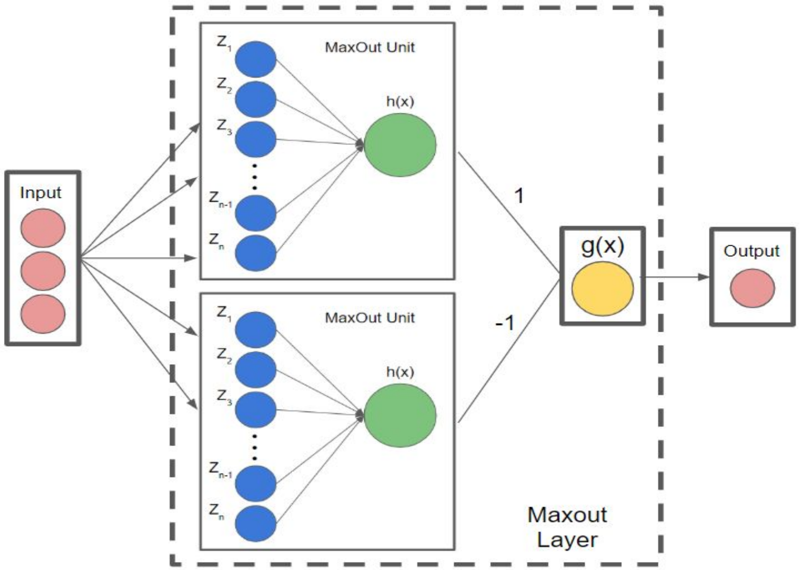
- k linear models
- Output is the maximal value from k models from the given input x

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

Where $z_{ij} = x^T W_{\ldots ij} + b_{ij}$

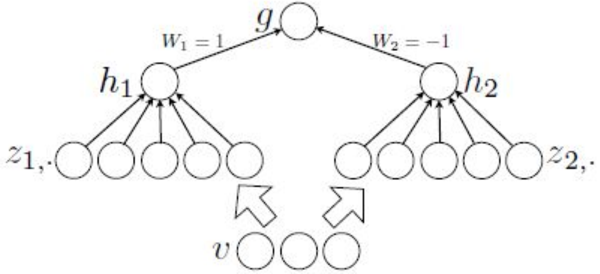
$W \in R^{d \times m \times k}$ and $b \in R^{m \times k}$

m:	number of hidden units
d:	size of input vector (x)
k:	number of linear models

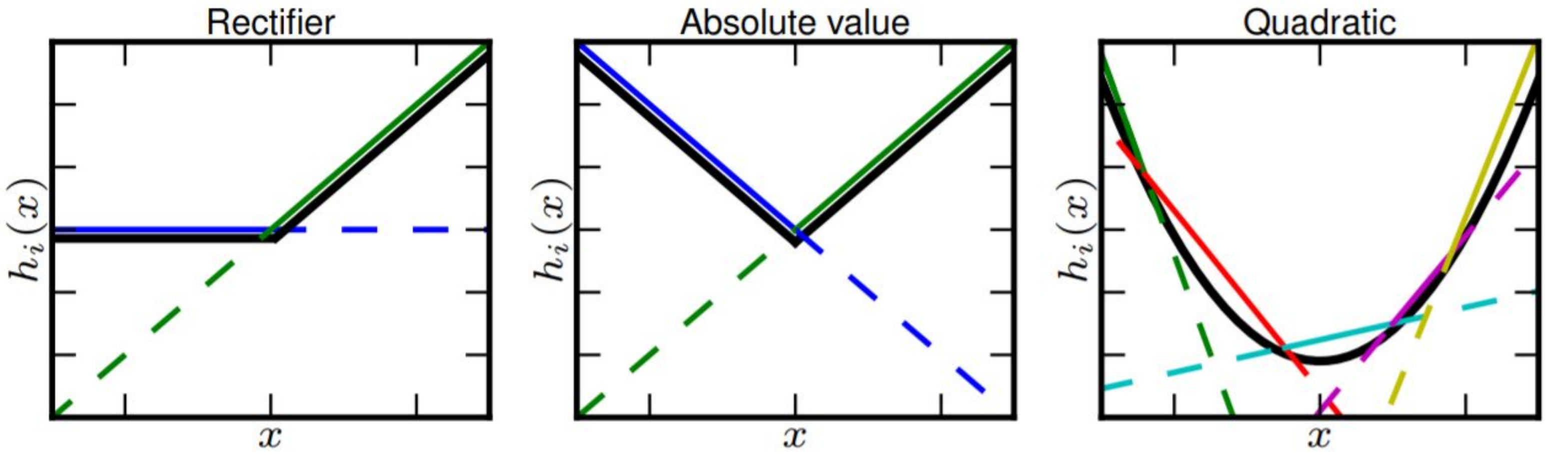


Maxout : universal approximator

- **Theorem 4.3** Universal approximator theorem.
  - *Any continuous function f can be approximated arbitrarily well on a compact domain $C \subset \mathbb{R}^n$ by a maxout network with two maxout hidden units.*



A Maxout units can approximate arbitrary **convex only** functions



Maxout can implement ReLU and absolute function with 2 linear functions,

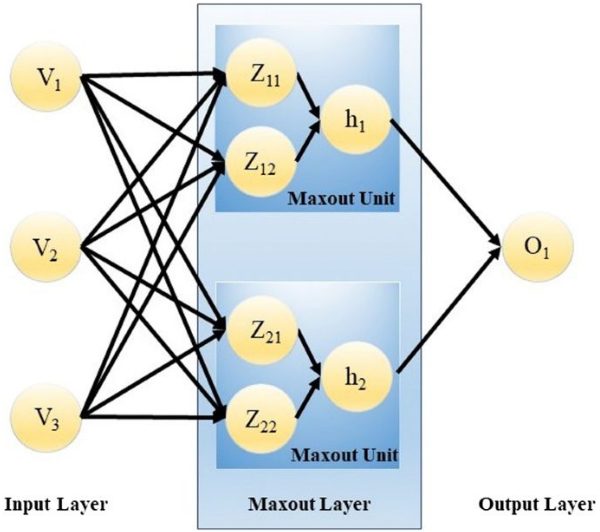
$$ReLU = \max(0, x),\ \text{abs}(x) = \max(x,\ -x)$$

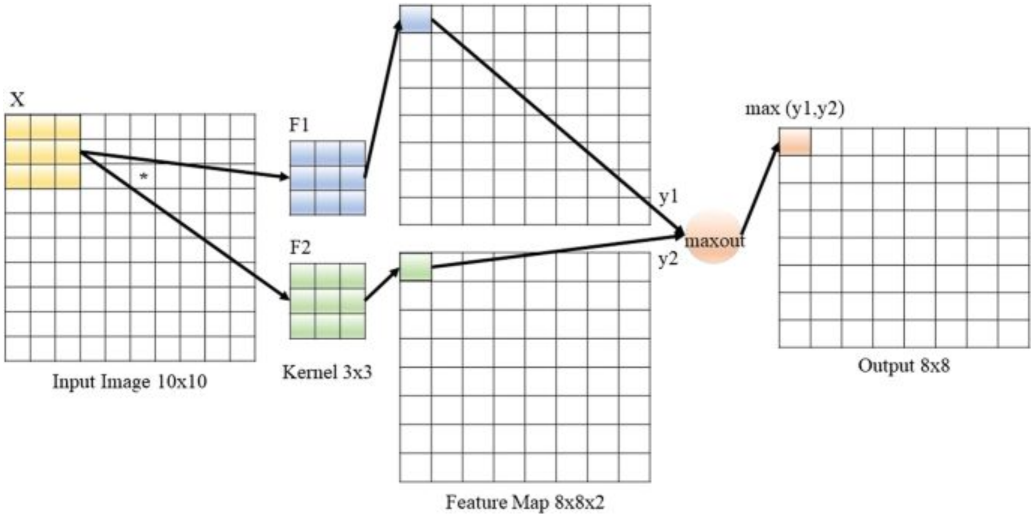
and quadratic curve with 5 linear functions.

Maxout with DropOut

- Maxout to complement the dropout regularization technique.
- The maxout activation facilitates the training of dropout with large gradient updates.

Maxout unit in a CNN architecture





Benchmark Results

Name	Classes	Training	Test	Image	Color
MNIST	10	60 000	10 000	28x28	Grayscale
CIFAR-10	10	50 000	10 000	32x32	Color
CIFAR-100	100	50 000	10 000	32x32	Color
SVHN	10	73 257	26 032	32x32	Color

- SVHN dataset also consists of 521,131 additional samples

MNIST

- *Permutation invariant* MNIST

- Maxout multilayer perceptron (MLP):
  - Two *maxout layers* followed by a *softmax layer*
  - Dropout
  - Training/Validation/Test : 50,000/10,000/10,000 samples

- Error rate: 0.94%

- This is the best result without pre-training

MNIST

- Without permutation invariant restriction

- Best model consists of:
  - 3 convolutional maxout hidden layers with spatial max pooling
  - Followed by a softmax layer

- Error rate is 0.45%

- There are better results by augmenting standard dataset

CIFAR-10

- Preprocessing
  - Global contrast normalization
  - ZCA whitening
- Best model consists of
  - 3 convolutional maxout layers
  - A fully connected maxout layer
  - A fully connected softmax layer
- Error rate
  - Without data augmentation — 11.68 %
  - With data augmentation — 9.35 %

CIFAR-100

- Use the same hyperparameters as in CIFAR-10
- Error rates
  - Without retraining using entire training set: 41.48 %
  - With retraining : 38.57 %

SVHN

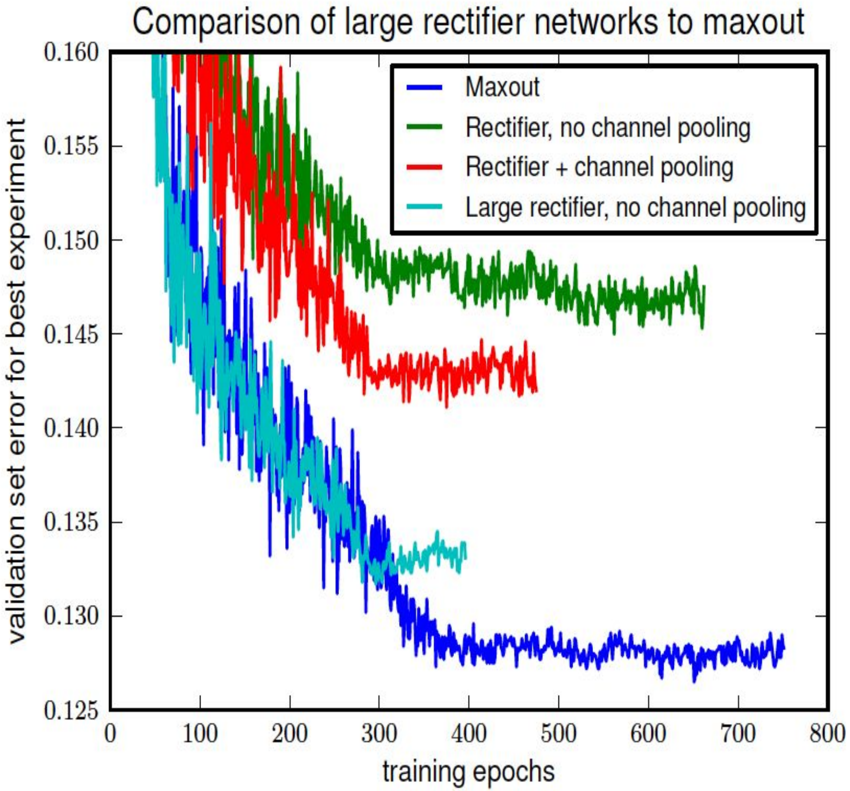
- Local contrast normalization preprocessing
- 3 convolutional maxout hidden layers
- 1 maxout layer
- Followed by a softmax layer

- Error rate is 2.47%

Local contrast normalization
(Zeiler&Fergus 2013)

Comparison to rectifiers

- Does the obtained results is due to improved preprocessing or by the use of maxout ?
- On large cross-validation experiment, authors have found out that maxout offers clear improvement over the rectifiers.

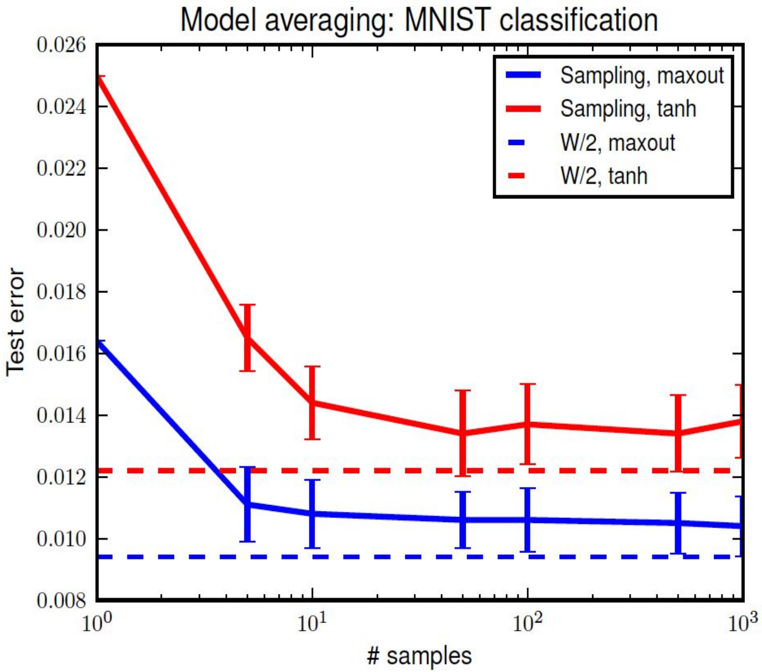


Why Does Maxout work ?

- Maxout is highly compatible with dropout’s approximate model averaging technique.
- Maxout gives better performance than max pooled rectified linear units when training without dropout.
- Maxout helps dropout training to better resemble bagging for lower-layer parameter.

Model Averaging

- Dropout performs model averaging.
- Many activation function have significant curvature nearly everywhere which makes the approximate model averaging of dropout not that accurate.
- Comparing the geometric mean of sampled model's predictions with the prediction made using dropout technique by dividing weights by 2(Hinton et al. 2012).
- More accurate in case of maxout.



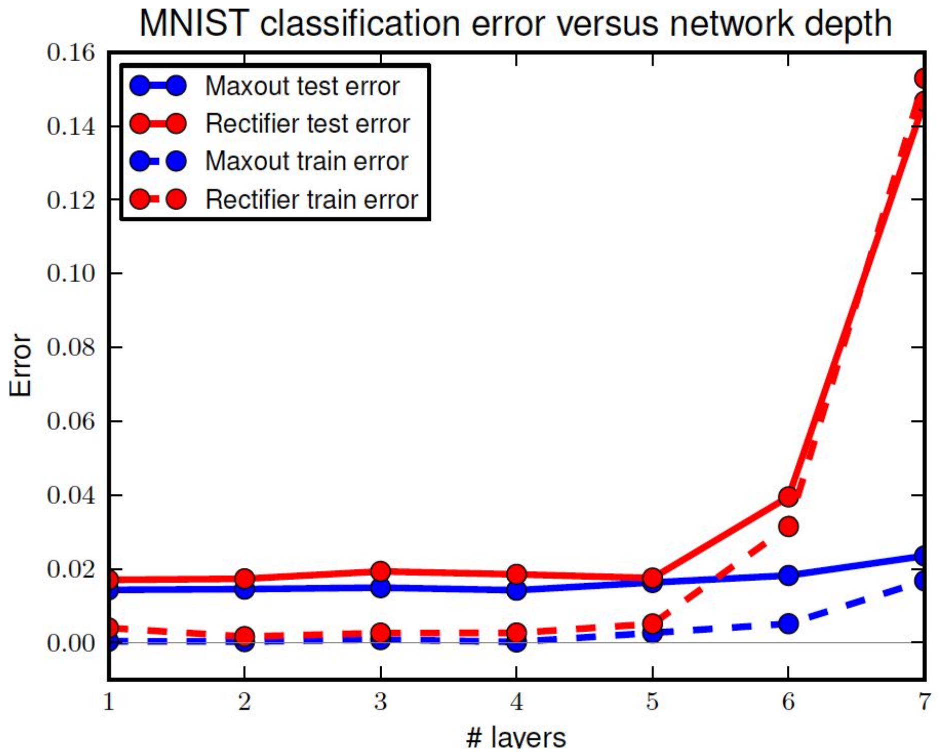
Optimization

- Without dropout, the maxout works better than max pooled rectified linear units.
- Training a small model on large dataset
  2 hidden convolution layers
  SVHN dataset(600,000 samples)
- Error rate
  Rectifier error : 7.3%
  Maxout error : 5.1%

Optimization

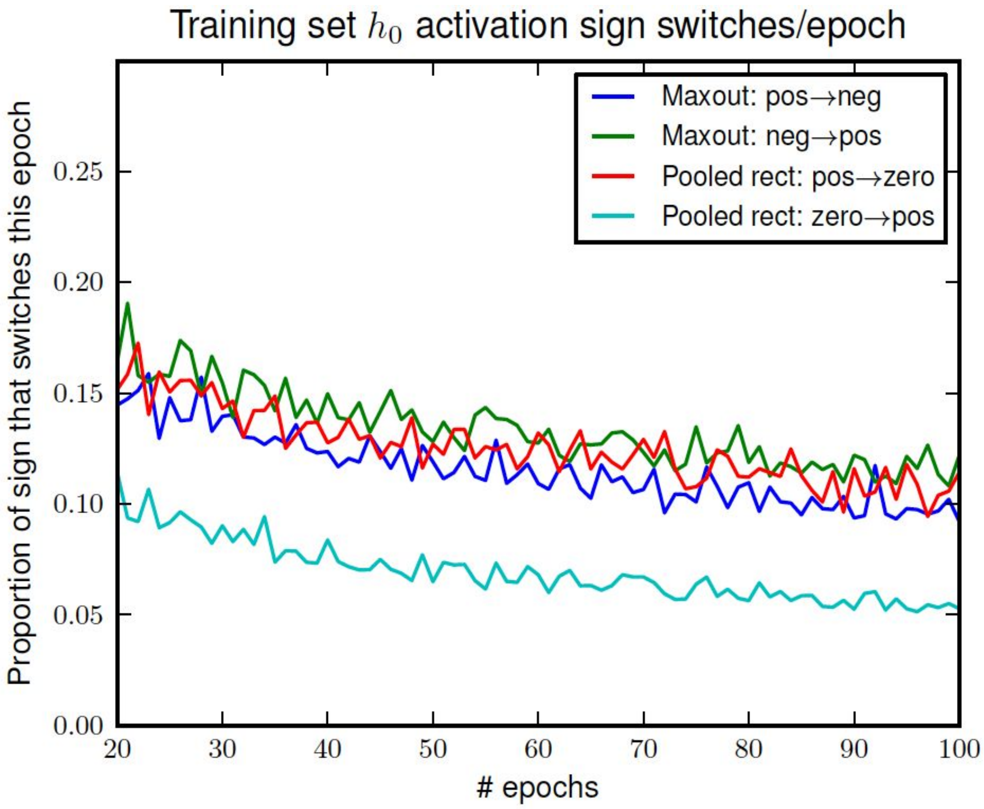
- Optimization stress test on deep models with MNIST dataset.

  Authors have found out that with the increasing depth, maxout deals better as compared to pooled rectifiers.



Saturation

- During dropout training, the rectifier inits transition from positive to 0 activation more frequently as compared to opposite transitions resulting in predominance of 0 activations.
- Maxout units freely moves between positive and negative activations at roughly same rates.



Saturation

- The high proportion of zeros and difficulty to escape them makes optimization performance rectifiers weaker relative to maxout.
- 2 MLPs on MNIST dataset is trained with
  - 2 hidden layers
  - 1200 filters per layer pooled in groups of 5
- Maxout:
  - 99.9 % filters used (2 out of 2400 unused)
- Rectifier:
  - 17.6 % unused filters in layer 1
  - 39.2 % unused filter in layer 2

References

Goodfellow et al,. Maxout Networks, Proceedings of International Conference on Machine Learning(ICML), 2013
Hinton et al., Improving neural networks by preventing co-adaptation of feature detectors.(2012)
Zeiler, Matthew D. and Fergus, stochastic pooling for regularization of deep convolutional neural networks. In ICLR 2013