Recognition of Dogs and Cats by CNN and Visualization by Decovnet

Abdulkadir Çelikkanat

Abstract—This paper addresses the problem of fine-grained object categorization for identifying the breed of dogs and cats from their images. Different convolutional neural network (CNN) architectures are used for recognition over Oxford Pet Dataset, and the effects of different model choices are evaluated. Further, deconvolutional network (deconvnet) model [3] is constructed to visualize features to gain intuiton.

Index Terms—Convolutional neural network(CNN), Fine-grained object categorization, Deconvolutional network (deconvnet)

1 Introduction

THE emergence of convolutional neural networks (CNNs) in the early 1990's have attracted attentions of many researchers from computer vision, and showed its outstanding power in many challenging problems. The ongoing success of CNN is incontrovertible, in the recent years, CNN models has substantially outperformed the state-of-the-art accuracies of object recognition over many benchmark datasets [5], [6]. However, research on object classification generally focus on the classification of well distinguished object categories (ex: car vs book, apple vs bag). On the other hand, the fine-grained classification task, recognition of classes which are highly similar each other (ex: cat breeds, car models) is a more challenging problem. Hence, it may be interesting to see how successfully CNN can perform for a fine-grained problem. In this paper, such a harder problem is considered, the dataset consisting of classes in which intra-class variation is low, and inter-class variation relatively is high. [Figure 1]

In addition to the application of CNN models, it is also important that understanding of how and why CNN perform well for many tasks. In the work of Zeiler et al [4], a multi-layered Deconvolutional Network (Deconvnet) is proposed, and introduced a visualization technique which allows to observe the input stimuli which induce feature map at any level of the model. In this work, to gain intuition about the network, the visualization of some layers in different levels are depicted, and Deconvnet is constructed as similar to the work [3]

2 Dataset

2.0.1 About the dataset

The Oxford-IIIT Pet Dataset consisting of 37 categories with roughly 200 cat and dog images from different breeds. The dataset also contains tight bounding box around the head of the animals. Since the images are in varying sizes, this information can be very useful to extract animals from images in preprocessing step, but it does not unfortunately exist for all images. The dataset is relatively harder, because images is in high variations in pose, lighting. The difference among images in the same class is high and classes are very similar to each other. Some instances from the dataset is given in the figure [1]

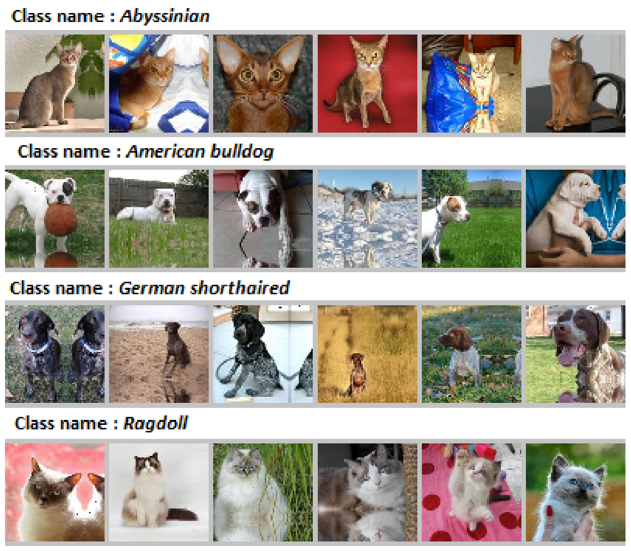
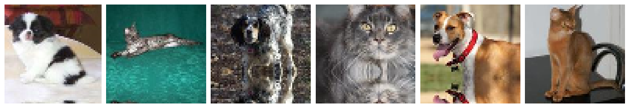

Class name : Abyssinian
Class name : American bulldog
Class name : German shorthaired
Class name : Ragdoll

Fig. 1. Some instances from the dataset

2.0.2 Preprocessing

The images in the dataset are in the varying sizes. Mirror-padding is applied in order to make their sizes equal. Directly cropping images with respect to the minimum size image in the set causes to losing information in many images so I have preferred to mirror-padding. The dataset contains around 7500 images, and one of the important factors affecting the success of CNN is the size of the training set. Hence, data augmentation is suggested to prevent from overfitting.

A randomly choosen image with probability 0.5 is flippped [Figure 3a], an image is then rotated by -12, -6, 0, 6, or 12 degrees with probability 0.2. [Figure 3b] Further, an image is translated by -16, -8, 8, or 16 pixels with probability 0.25. [Figure 3c] Finally, some images are zoomed with probability 0.5. [Figure 3d]


(a) Randomly flipped images

(b) Randomly rotated images

(c) Randomly translated images

(d) Randomly zoomed images

Fig. 3. Instances from the dataset in data augmentation process

At the end of the data augmentation process, the total number of images is increased to 33,000, and images are rescaled to the size of 64x64 px.

3 Experiment Designs and Details

In this project, I have implemented convolutional neural networks (CNN) in python with theano library [12]. These models map the images in the dataset to a probability vector y, via a series of layers. The models contains convolution layers, max-pooling layer, which return the maximum value for a certain regions in a layer, and rectification layer (ReLU) which passes the responses through a rectified function (relu(x) = max(x,0)). The training of the CNN models generally takes long times, especially depending on the dataset size, input image size, or network complexity, it may continue for couple of days. Hence, I have decided to rent a Amazon EC2 instance, g2.2xlarge, which uses GPUs instead of CPUs. Therefore, the training time substantially decreased with respect to the my personal computer.

I have firstly tried my implementation over two well-known datasets Mnist and Cifar-10 with a model consisting of a convolution, pooling, reLu, and fully connected layers similar to the figure 2. Therefore, I have checked correctness of my CNN class implementation and results are very close

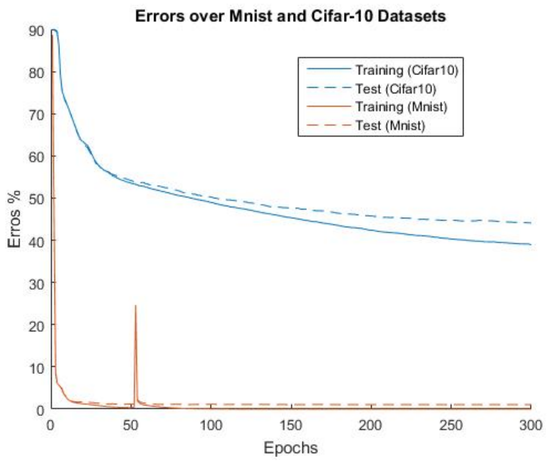


Fig. 4. I have tested my implementation over well-known datasets.

TABLE 1

Dataset	*Accuracy*	*Method*
Oxford Pet Dataset	38%	VLFeat-BoW
MNIST	99%	Convolutional net LeNet-5
Cifar-10	74%	CNN without any image translations / transformations / preprocessing

to some baseline scores as given in the table 1. My training and testing scores are depicted in the figure 4. The score for Mnist dataset is nearly perfect, it gives almost 0 error for test set, and the validation accuracy at 300th epoch for Cifar-10 dataset is around 55%. Since it takes too much time, I cut it at 300th epoch, the testing score of it seems to continue to decrease. One of the experiments done by choosing a subset of 65% over whole Cifar-10 dataset gave accuracy around 80% for training and 65%for testing.

I have divided the augmented Oxford-IITD Pet dataset into 10 subsets, and chosen two of them as a testing set, the remaining ones as a training dataset. I have repeated this process four times like 4-fold cross validation, and check my base CNN model, named M-CNN, details given in the table 2, obtained the error score 45.8685 ± 6.4553 for training, and 88.3177 ± 0.955 for testing. Since the training time is costly, increasing the number of folds cannot be possible. I have also tested the same model by choosing 10 classes consisting of dog and cat breeds, and 2 class which only differentiates dog and cat images. The scores for three experiments depicted in the figure 5. Since I have considered

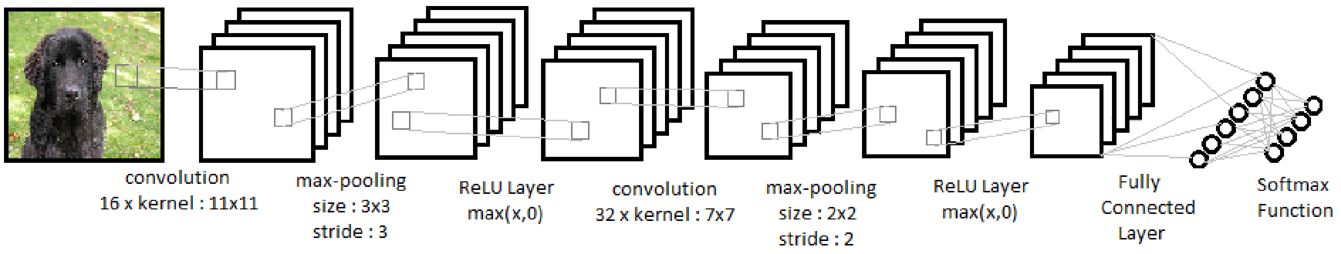


Fig. 2. A sample CNN architecture which named as M-CNN model

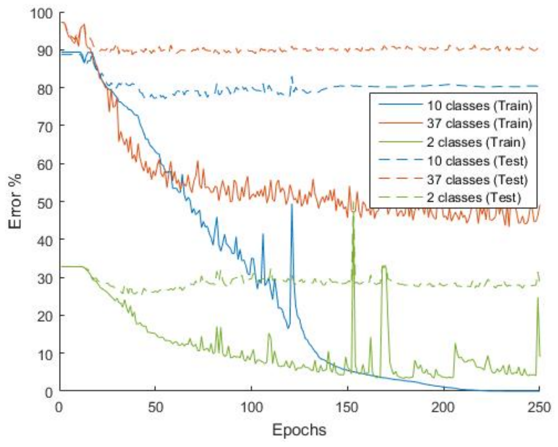


Fig. 5. Models considering the different number of classes

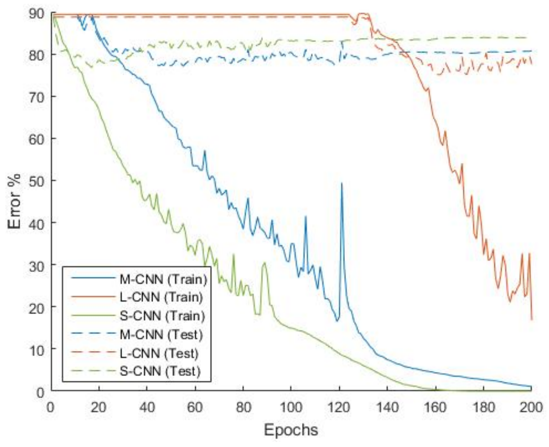


Fig. 6. Comparison of network sizes

a fine-grained object classification problem, the dataset is so challenging. As it might be expected, classifying only dogs and cats is more successful, the test accuracy is about 73%. When I selected 10 dog and cats breeds, the training error remains about 79%, and test error becomes almost 0.

The proposed architectures were trained by using stochastic gradient descent with batch size of 300 images, and learning rate of 0.05. The input size of images is 64x64 pixes, and initial bias values are considered as 0.1 for the first convolution layer, and 0 for others. The weights in the layers are initialized with normal distribution with mean 0, and standard deviation 0.01.

The models unfortunately overfitted, probably due to the dataset. In the remaining of the experiments, it is tried to prevent from overfitting, 10 dog and cat breeds will be used to decrease the training time, and the M-CNN architecture are considered as a base model throughout the paper. The model is trained and tested only once for different configurations, since the learning is so costly, this can be admissible. [11]

3.0.1 The effect of Model Sizes

The depth of a convolutional neural network has significant influence on designing a good CNN architectures. Because extremely large or small models may cause to overfitting or underfitting depending on the size of data set [8]. Hence, in addition to M-CNN model, two new CNN architectures, 'CNN-S' and 'CNN-L' are proposed to compare the their effect on the sizes of models.

As it is shown in the figure 6, the error of L-CNN model began to decrease after 130th epoch, and its testing error is better than others, around 76%. The increase in the number of hidden layers decreases the testing error but converges more slowly.

3.0.2 The effect of Filters Sizes

To analyze the influence of filter sizes on CNN models, three different kernel sizes have been considered. The filter sizes in the each level of CNN model has been enlarged to 15x15 and 11x11 and narrowed to 7x7 and 3x3. to obtain additional two new models: 'Small', and 'Larger'.

The training result is depicted in the figure 7, which shows that the decrease in the size of filters increases testing error. The accuracy of the model having smaller kernels is around 26%, which is better than the base model M-CNN.

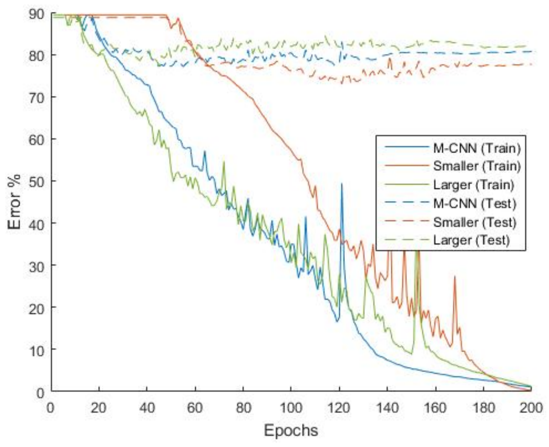


Fig. 7. The effect of Filters Sizes

	1	2	3	4	5	6	7	8	9	10	11
S-CNN	16 x Conv1 Size :11x11	Pool 3x3 Stride : 3	Relu 1	64 Units	K-Class Classifier						
M-CNN	16 x Conv1 Size :11x11	Pool 3x3 Stride : 3	Relu 1	32 x Conv2 Size :7x7	Pool 2x2 Stride : 2	Relu 2	64 Units	K-Class Classifier			
L-CNN	16 x Conv1 Size :11x11	Pool 3x3 Stride : 3	Relu 1	32 x Conv2 Size :7x7	Pool 2x2 Stride : 2	Relu 2	64 x Conv3 Size :3x3	Pool 2x2 Stride : 2	Relu 3	64 Units	K-Class Classifier

TABLE 2
Different CNN models to compare the effect of the network sizes

3.0.3 *The effect of Number of Filters*

One of the hyperparameters that may effect the accuracy of the model is the number of filters in the layers. Therefore, I have increased the number of filter by double and decreased by half.

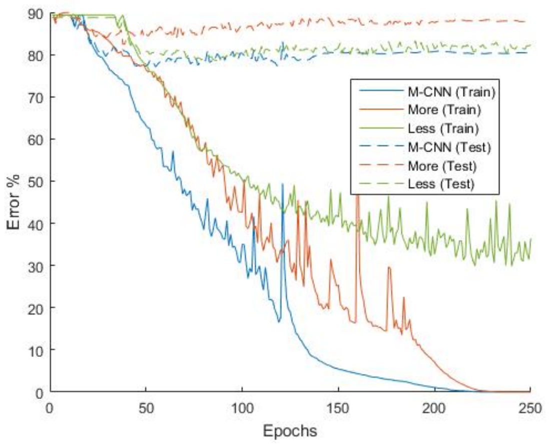


Fig. 8. The effect of Number of Filters

The results in the figure 8 shows that the test accuracy of the model having less number of filters are close to the base model M-CNN, but its training error is higher. On the other hand, increasing the number of kernels has negative affect for both training and testing error accuracies.

3.0.4 *The effect of Applying Dropout*

One of the simple techniques to avoid from overfitting is to apply dropout [7], which randomly drop units' connections from the neural network. Hence, I have applied dropout, and added one more fully connected layer to the M-CNN model.

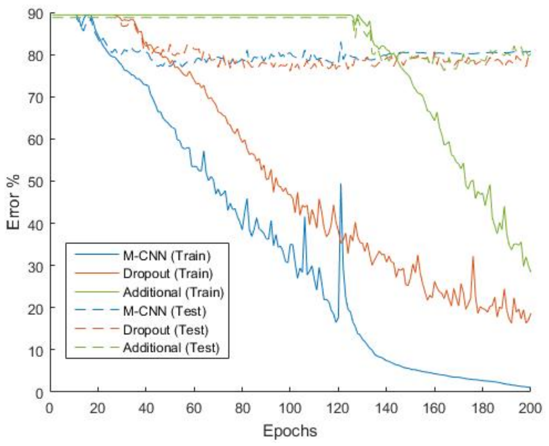


Fig. 9. The effect of applying Dropout

After applying the dropout to the fully connected layer, test scores keep remaining around its previous value. Adding additional fully connected layer also does not contribute the scores positively much, but the convergence of the model gets slower.

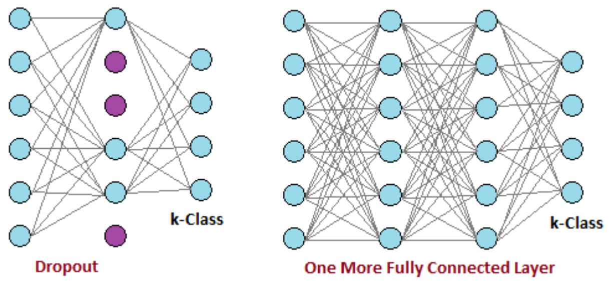


Fig. 10. Illustration of dropout and adding one more fully connected layer

4 Visualization with Deconvolutional Network(Deconvnet)

Visualizing the feature activities in the layers is significant to understand the operation of convolutional neural network. In this project, I have implemented the decovnet defined in [3], which maps activities in the intermediate layer to input space by using the same layers (convolution, pooling, ReLu) in reverse order.

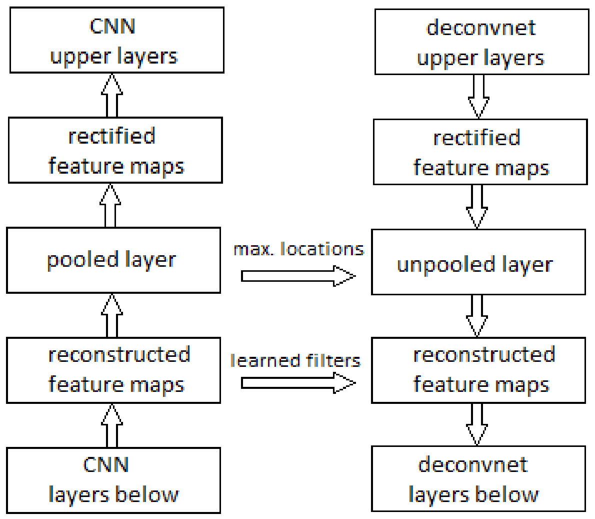


Fig. 11. Illustration of deconvnet structure

An illustration of a decovnet structure consisting of rectification, unpooling and deconvolution layers is depicted in the figure 11. The *reLu* layer is the same as it is in the CNN models, which only maps negative values in the features to 0. The *unpooling layer* uses the kept coordinates of the maximum values of the feature map in the pooling phase in order to inverse this nonlinear operation. It just puts the values coming from the previous layer, reLu, to those locations. The *deconvolution layer* is a litle bit complicated, it convolves the feature map of the previous layer with the horizontally and vertically flipped learned filters.

The visualization of the some features in the sixth layer of M-CNN model (figure 2) is illustrated in the figure 12. Each column k represents the $2k-1$th feature map in the 6th layer. For instance, the features in the second and third column gives outputs for the same dog image in the second row. However, it seems that the feature map in the second row is more sensitive to the grass on the background. The

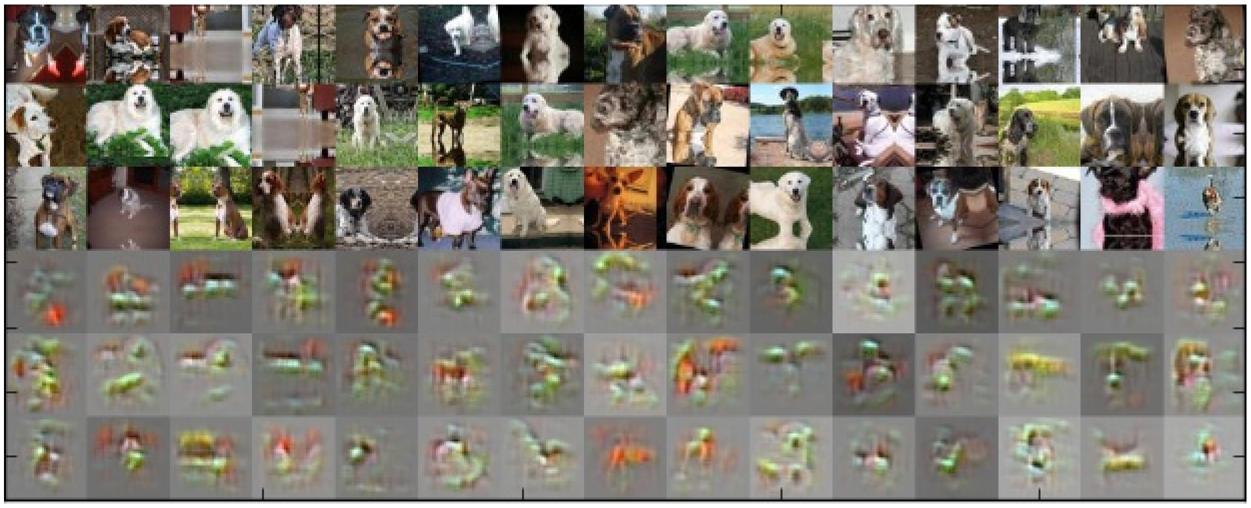


Fig. 12. The visualization of feature maps of 6th layer in the M-CNN model. Each kth column represents results for the $2k-1$th feature map.

feature in the 10th column resembles a line detector, since it is more responsive to the borders of dogs and hills in the background.

More comments and inferences for the output of the feature maps in the figure 12 can be made to improve comprehension about the M-CNN model. However, since I have preferred to use input images of size 64x64 in the beginning of the project due to the training cost, the figures are not clear as in the work of Matthew D. Zeiler et al [3]. However, it can be still helpful to gain intuition about the feature maps of the model.

5 Conclusion

In this paper, a fine-grained object categorization problem is considered with convolutional neural network. The dataset, Oxford-IIT Pet, is challenging, and there is no previous work which uses CNN, but the work of Omkar M Parkhi [10] has 38.45% classification accuracy with the method using VLFeat BoW. For 37 classes, I got accuracy of around 48% for training, and 90% error for testing. For two classes, in other words, differentiating the dogs and cats is more successful, it gave about 28% accuracy.

Since the accuracy of the neural networks also depends on the model structure, I have tested different architectures to decrease error and prevent from overfitting. It is observed that increasing the number of filters has negative effect, but deeper networks, diminishing the size of filters made positive contributions. Initially, the M-CNN model has error rate around 79%, and it falls up to 70% for 10 classes.

Finally, the different feature maps are visualized, and analyzed to gain intuition about how CNN perform well by implementing the Convnet model [3].

References

[1] Oxford-IIIT Pet Dataset *http://www.robots.ox.ac.uk/ vgg/data/pets/*

[2] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan, *Dropout: A simple way to prevent neural networks from overfitting*, JMLR, 15, 2014

[3] Zeiler, M. D. and Fergus, R. *Visualizing and understanding convolutional networks*. CoRR, abs/1311.2901, 2013. Published in Proc. ECCV, 2014.

[4] Zeiler, M., Taylor, G., Fergus, R. *Adaptive deconvolutional networks for mid and high level feature learning.* In: ICCV (2011)

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in Neural Information Processing Systems (NIPS), pages 10971105, 2012.

[6] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: *An astounding baseline for recognition*

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting*. JMLR, 2014.

[8] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, Timothy Hospedales. *When Face Recognition Meets with Deep Learning: an Evaluation of Convolutional Neural Networks for Face Recognition*. arXiv preprint arXiv:1504.02351, 2015.

[9] O. M. Parkhi, A. Vedaldi, A. Zisserman, C. V. Jawahar *Cats and Dogs* IEEE Conference on Computer Vision and Pattern Recognition, 2012

[10] O. M. Parkhi, A. Vedaldi, A. Zisserman, C. V. Jawahar *Cats and Dogs* IEEE Conference on Computer Vision and Pattern Recognition, 2012

[11] Ethem Alpaydin, *Machine Learning*, 2nd edition, The MIT Press, pg:476

[12] Theano Development Team, *Theano: A Python framework for fast computation of mathematical expressions*, http://arxiv.org/abs/1605.02688