Going More Deeper with Convolutions for Network in Network

Neji Kouka
MARS laboratory, ISITCom hammam Sousse, University of Sousse, Tunisia

Jawaher Ben Khalfa
MARS laboratory, ISITCom hammam Sousse, University of Sousse, Tunisia

Jalel Eddine Hajlaoui
MARS laboratory, ISITCom hammam Sousse University of Sousse, Tunisia

ABSTRACT
Network in Network is an important extension of the deep convolution neural network that uses a shallow multilayer perceptron (MLP), a nonlinear function, to replace the linear filter. In this article, we propose to replace convolution layers with convolution modules. The main feature of this architecture is the improved utilization of computing resources inside the network. This has been achieved through a carefully crafted design that allows for increased network depth and width while keeping the compute budget constant. The experimental results on the CIFAR10 dataset demonstrate the effectiveness of the proposed method.

Keywords
Convolutional Neural Networks (CNNs), Image recognition, Network in Network (NiN).

1. INTRODUCTION
We Convolutional Neural Networks (CNN) have achieved state-of-the-art performance in the tasks of image classification and object detection. They are organized in successive calculation layers alternating between convolution, activation function and pooling. In a convolutional layer, linear filters are used for convolution. Convolution layers can be improved by replacing the linear filter with a linear convolution module which incorporates several types and sizes of convolution filters. In this article, we propose to modify the convolution layer of size 5x5 exploited inside the ”MLPconv” structure by a convolution module which makes it possible to increase the depth and the width of the network while maintaining the computation budget constant. The contributions of this article are:

• A convolution module is proposed and used as convolutional filters.

• We propose a new architecture for the MLPconv layers which allows to have models with considerably improved performances.

• We present a detailed experimental study of deep model architectures that ex amines in depth several important aspects of the new MLPconv layers .

• The proposed network significantly outperforms NiN in reducing the test error rate on the CIFAR10 dataset.

The rest of this article is organized as follows: In Sect. 2, an overview of related works is given. Section 3 is about strategy. Experimental evaluations and comparative analysis are presented and discussed in Sect. 4. Section 5 is devoted to implementation details. The work is concluded in the last section

2. RELATED WORKS
Various CNN models are proposed since the remarkable performance of AlexNet on the ImageNet database in 2012. These innovations were mainly due to the reorganization of different layers and the design of new blocks or the exploitation of multipath.

Zeiler and Fergus introduced a de-convolutional neural network [1] which reduced the error rate from 16.4% to 11.7% thanks to a modification in the topology of AlexNet by reducing both the size of the filters of the first layer from (11 x 11) up to (7 x 7) and increase the number of convolution kernels for the last three convolution layers. In 2014, several architectures with different numbers of layers ranging from 11 to 19 are proposed in [2]. In the VGG-16 topology, the convolution filters (7x7) are replaced by 3 size convolution filters (3x3) in order to increase the number of activation functions exploited and to keep the same resulting dimension by reducing the complexity. of calculation and the number of parameters. It is noted that the number of convolution kernels of VGG 16 [2] starts from 64 and increases by a factor of 2 for every 2 convolution layers.

GoogLeNet [3] was introduced in 2014 by Szegedy et al. The main target of this architecture was to reduce the computational cost and the number of parameters while offering high precision thanks to the reduction of the channel dimension. Exploiting the 1x1 convolution layer before the 3x3 and 5x5 layers. In the classifier of architecture [4] and [3], a global average pooling layer is exploited instead of using an all-connected layer which is traditionally used in classical CNNs for the purpose of reducing the connection density.

Generally speaking, one can notice from this literature, that there are six directions that can be used to improve the performance of CNN: Increasing the width; The increase in depth; changing convolution [5 - 9] or pooling [10-18] parameters, change the activation function and reduce the number of parameters and resources.

We consider the first three previous directions to propose a new approach. convolution which incorporates multiple filters instead of layer operated convolution. As a result, we obtain a new NiN architecture [4] that is deeper and much more precise.

3. PROPOSED MODEL
3.1 Network in Network (NiN)
The "Network In Network" model [4] consists of several "MLPconv" structures which are stacked in a successive way and a global average pooling layer is used instead of the fully

connected layers. Figure 1 illustrates the overall structure of the architecture.

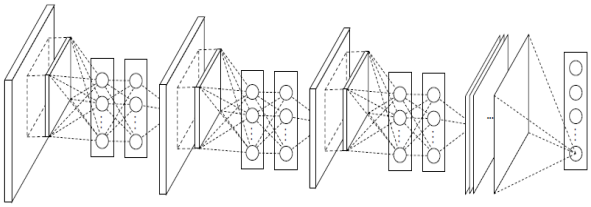

Figure 1: « MLPConv$_{inception}$.» structure.

In terms of accuracy, NIN [4] generates an error rate equivalent to 10.41% using the dropout layer and without data augmentation. Using these layers, NIN [11] obtained an error rate equivalent to 8.81%.

3.2 MLPconv$_{inception}$ structure:

Compared to the original architecture [4], the convolution layer is replaced by a convolution module. A stack of two convolution layers of size 3x3 is used in parallel with a single 5x5 layer and a single max pooling layer of size 3x3. Figure 2 illustrates the architecture of « MLPConvinception».

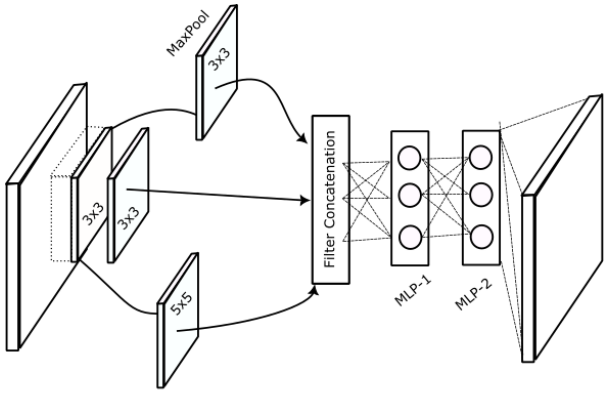

Figure 2: « MLPConv$_{inception}$.» structure.

Table 1: The numbers of the convolution kernels

Layers	Conv 3x3	Conv 5x5	MLP-1	MLP-2
Numbers	96	96	160	96

For all « **MLPConv$_{inception}$.»** structures, the numbers of convolution kernels are the same. Table 1 describes the numbers of the kernels.

3.3 NIN$_{inception}$ structure:

We describe our different NINinception model configurations for CIFAR-10. In this model, convolutional layers follow two simple design rules: first, layers that participate in MLPconvinception have the same output function feature map size and the same number of filters; second, the max pooling layer leveraged inside MLPconvinception should generate the same size output function feature map. We perform downsampling using the maximum pooling layers of size 3×3 which have a stride of 2 (3×3/ST.2). These layers are usually inserted after the first two MLPconvinception structures. The network ends with a global average pooling layer and a softmax layer. Figure 3 shows the overall structure of NINinception.

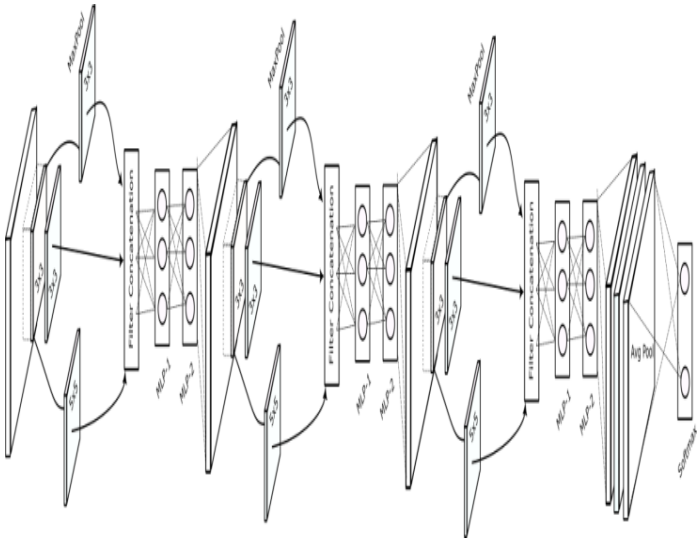

Figure 3: The overall structure of NIN$_{inception}$.

3.4 Regularization layers in MLPConv$_{inception}$

A Dropout layer is added after each pooling layer that is located between the MLPconvinception. This layer also produces a regularization effect to avoid over-training of the network.

Data augmentation layer in NINinception:

Data augmentation is a strategy for dramatically increasing the diversity of data available for training models, without the need to collect new data, i.e. creating modified copies of each instance in a database. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

4. Experimental result :

We evaluate our configurations on a reference data set:

CIFAR-10. The CIFAR-10 (Canadian Institute for Advanced Research) dataset consists of 60,000 RGB images of size 32×32 grouped into 10 image classes. The dataset is divided into five training packages and one test package, each containing 10,000 images. The training packages contain exactly 5000 images of each class. The test batch contains exactly 1000 randomly selected images from each class.

4.1 Dropout effect in NINinception:

Dropout has proven to be an effective technique for regularizing different networks because it reduced the error rate with around 1.5%.

4.2 The effect of data augmentation in NINinception

The effect of data augmentation in NINinception :

The exploitation of this layer has shown a positive effect in reducing the classification test error and automatically leads to

significantly better results than learning without exploiting this layer.

4.3 NINinception performance:

In terms of accuracy, the NINinception model obtained an accuracy rate equivalent to 92.31% using normalization and regularization and without data augmentation. By using the data augmentation layer (translation and horizontal flip), the model obtained an accuracy rate equivalent to 92.72%. The experimental results obtained by averaging over 5 runs with a mini-batch size equivalent to 128 also demonstrate the effectiveness of the proposed idea of replacing convolution layers with convolution modules.

Table 2 represents a comparison between our work and the state of the art on the CIFAR-10 database with/without the use of data augmentation.

Ref	Method	Error (%)
No data augmentation		
[**10**]	Stochastic pooling	15.13
[19]	Maxout network (k=2)	11.68
[5]	NIN	10.41
Our	NINinception	**7**.69
[20]	DSN	9.69
[24]	MIM (k=2)	8.52±0.20
Data augmentation		
[19]	Maxout network (k=2)	9.38
[5]	NIN [11]	8.81
[20]	DSN [15]	8.22
Our	NINinception	**7**.28
[21]	ResNet	6.43
[22]	Wide Resnet(28,10)	3.89
[23]	ResNeXt	3.58

5. IMPLEMENTATION details:

During the training, we exploited the stochastic gradient descent with a Momentum equivalent to 0.9 in all the experiments carried out. The base learning rate is equivalent to 0.005 and decreases by a factor of 10 every 10 epochs. The weight loss is equivalent to 0.0005. We initialized the weights in each layer from a normal to mean random distribution with a standard deviation equivalent to 0.01. We initialized neural biases in all convolution layers, as well as MLP layers with the constant 0. All experiments performed are run for a total of 160 epochs. All models used in this study were compiled with CPU support. All experimental studies were conducted in Google cloud environment on Linux operating system running on Dell Intel Core i5-2450M 2.50GHz processor and 6GB DDR4–2400 RAM. All codes are created with a python algorithm based on the "TensorFlow" deep learning framework.

6. CONCLUSION

In this paper, we proposed a new convolutional network architecture, which we call NINinception. In this architecture, we have replaced the convolution layer with a convolution module. NINinception tend to produce improved accuracy with increasing number of settings, with no signs of performance degradation or overfitting. The results are described as acceptable compared to other architectures tested on CIFAR-10 datasets. Future work should focus on designing new versions of CNN models that can meet or exceed the level of accuracy of this proposed model requiring shorter training time with less parameter consumption.

7. REFERENCES

[1] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014, pages 818–833. Springer, 2014.

[2] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", CoRR, vol. abs/1409.1556, 2014, [online] Available: http://arxiv.org/abs/1409.1556

[3] C. Szegedy et al., "Going deeper with convolutions", CoRR, vol. abs/1409.4842, 2014, [online] Available: http://arxiv.org/abs/1409.4842

[4] M. Lin, Q. Chen, and S. Yan. Network in network. International Conference on Learning Representations, abs/1312.4400, 2014.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580, 2012.

[6] D. Ciresan, U. Meier, J. Schmidhuber, "Multi-column deep neural networks for image classification", CoRR, vol. abs/1202.2745, 2012, [online] Available: http://arxiv.org/abs/1202.2745

[7] K. Gregor, Y. LeCun, "Emergence of complex-like cells in a termporal product network with local receptive fields", CoRR, vol. abs/1006.0448, 2010, [online] Available: http://arxiv.org/abs/1006.0448..

[8] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, Y. LeCun, "What is the best multi-stage architecture for object recognition?", Proc. IEEE Int. Conf. Comput. Vis., pp. 2146-2153, Sep. 2009.

[9] Y. LeCun, K. Kavukcuoglu, C. Farabet, "Convolutional networks and applications in vision", Proc. IEEE Int. Symp. Circuits Syst., pp. 253-256, Jun. 2010

[10] M. D. Zeiler, R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks", CoRR, vol. abs/1301.3557, 2013, [online] Available: http://arxiv.org/abs/1301.3557

[11] K. He, X. Zhang, S. Ren, J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition", Proc. Eur. Conf. Comput. Vis., pp. 346-361, 2014.

[12] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, "PCANet: A simple deep learning baseline for image

classification?", CoRR, vol. abs/1404.3606, 2014, [online] Available: http://arxiv.org/abs/1404.3606.

[13] C. Lee, P. Gallagher, Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed gated and tree", CoRR, vol. abs/1509.08985, 2015, [online] Available:https://arxiv.org/abs/1509.08985

[14] J. Springenberg, A. Dosovitskiy, T. T. Brox, M. Riedmiller, "Striving for simplicity: The all convolutional net", CoRR, vol. abs/1412.6806, 2014, [online]Available:http://arxiv.org/abs/1412.6806.

[15] K. Gregor, Y. LeCun, "Emergence of complex-like cells in a termporal product network with local receptive fields", CoRR, vol. abs/1006.0448, 2010, [online] Available: http://arxiv.org/abs/1006.0448.

[16] D. Yoo, S. Park, J. Lee, I. Kweon, "Multi-scale pyramid pooling for deep convolutional representation", Proc. IEEE Workshop Comput. Vis. Pattern Recognit., pp. 1-5, Sep. 2015.

[17] B. Graham, "Fractional max-pooling", CoRR, vol. abs/1412.6071, 2014, [online] Available: https://arxiv.org/abs/1412.6071

[18] N. Murray, F. Perronnin, "Generalized max pooling", Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., pp. 2473-2480, Sep. 2014.

[19] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In Proceedings of the 30th International Conference on Machine Learning (ICML 2013), volume 28 of JMLR Proceedings, pages 1319– 1327. JMLR.org, 2013.

[20] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply supervised nets. In Proceedings of AISTATS 2015, 2015.

[21] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", CoRR, vol. abs/1512.03385, 2015, [online] Available: http://arxiv.org/abs/1512.03385.

[22] S. Zagoruyko and N. Komodakis. Wide residual networks. In BMVC, 2016.

[23] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He, Aggregated Residual Transformations for Deep Neural Networks, arXiv:1611.05431.

[24] Z. Liao and G. Carneiro. On the importance of normalisation layers in deep learning with piecewise linear activation units. ArXiv preprint arXiv: 1508.00330,