*Multidiszciplináris tudományok, 10. kötet. (2020) 1 sz. pp.55-60* **https://doi.org/10.35925/j.multi.2020.1.7**

# Convolutional Neural Networks and Impact of Filter Sizes on Image Classification

**Owais M. Khanday**
*PhD Student, University of Miskolc, Institue of Information Science*
*Department of Applied Information Engineering*
*3515 Miskolc, Miskolc-Egyetemváros, e-mail: aitowais@uni-miskolc.hu*

**Samad Dadvandipour**
*associate professor, University of Miskolc, Institue of Information Science*
*Department of Applied Information Engineering*
*3515 Miskolc, Miskolc-Egyetemváros, e-mail: aitsamad@uni-miskolc.hu*

***Abstract***

*Deep Neural Networks (DNN) in the past few years have revolutionized the computer vision by providing the best results on a large number of problems such as image classification, pattern recognition, and speech recognition. One of the essential models in deep learning used for image classification is convolutional neural networks. These networks can integrate a different number of features or so-called filters in a multi-layer fashion called convolutional layers. These models use convolutional, and pooling layers for feature abstraction and have neurons arranged in three dimensions: Height, Width, and Depth. Filters of 3 different sizes were used like 3×3, 5×5 and 7×7. It has been seen that the accuracy on the training data has been decreased from 100% to 97.8% as we increase the filter size and also the accuracy on the test data set decreases for 3×3 it is 98.7%, for 5×5 it is 98.5%, and for 7×7 it is 97.8%. The loss on the training data and test data per 10 epochs could be seen drastically increasing from 3.4% to 27.6% and 12.5% to 23.02%, respectively. Thus it is clear that using the filters having lesser dimensions is giving less loss than those having more dimensions. However, using the smaller filter size comes with the cost of computational complexity, which is very crucial in the case of larger data sets.*

***Keywords:*** *CNN, impact, filter size, image, classifications*

## 1. Introduction

Machine learning is widely used for data classification. Many classification algorithms are used like Support Vector Machines (SVM), Neural Networks (NN), K-nearest Neural Network, and Convolutional Neural Networks (CNNs). CNN has gained the best place in the image classification. It is a kind of neural network which can extract useful features too without the manual hand-tuning automatically. These extract the features from the data points like in images to solve the task of image classification or object detection in the images. It can produce the features itself. CNN is a deep NN with the following layers –Input layer: Image, 2D convolutional layer, ReLU layer, Pooling layer, Fully connected layer, Softmax layer, and a Classification layer. The convolutional layers have got

some parameters; these are filter number, filter size, convolutional stride. These networks can have different levels of features or filters in a multilayer fashion, which are known as convolutional layers that can be enriched by increasing the number of stack or depth [1]. CNN is like a network having raw image pixels on one end and class score on the other end, which is calculated by a single differentiable function and the loss function SVM or Softmax on the last fully connected layer [2]. It was thought that the filter is a black box having some numerical values which do not affect the accuracy of the classification. This is not true; in fact, the CNN filters are having a significant contribution to the accuracy of the classification algorithms [3]. In CNN, the input layer has three dimensions, i.e., height, width, and depth, so, the input layer is whose input is an image is taken as three dimensional. Each image is an array of 28×28×1, each corresponding to height, width, and depth, respectively. The depth is taken as one because the images are greyscaled. In the end, the Convolutional Network is reducing the full image into class scores having single values. The convolutional network is a sequence of layers, (e.g., Convolutional Layer, Max/Avg Pooling layer, and fully connected layer).

## 2. Dataset

MNIST Dataset is a database that contains the images of handwritten digits. In total, it has 75000 image examples with their labels. The dataset is divided into three sets, which are the training set, the test set, and the validation set. Sixty thousand examples are used for the training purposes, 10000 samples for the test, and 50000 for the validation [3]. The illustrations are bi-level images (black and white only) taken from the NIST dataset, and the size is normalized to fit in a 20×20-pixel box. The examples are to be centered in a 28×28-pixel image with a height of 28, width of 28, and a depth of 1. The images look like in figure 1.

***Figure 1.*** *MNIST dataset image examples*

## 3. Architecture

The output from the local regions in the input will be calculated by the convolutional layers, i.e., the dot product of the weights and addition of the bias. Thus, resulting in the volume of 28×28×16 if we decide to use 16 filters. The convolutional network will have an input layer followed by the convolutional layers, and the activations are to be calculated by the ReLU layers. Then we have the pooling layer for max/avg pooling, which is going to perform downsampling operation along the spatial dimensions. After that, the output is taken at a fully connected layer/FC layer, which will compute the valid scores for the classes, i.e., for 0-9 classes. Layers used in the Convolutional Neural Networks are as follows:

INPUT: The input is an image of 28×28×1 pixel values where the height, width, and depth are 28, 28, and 1, respectively. As the images are the only greyscale, so we are using the depth of one. If the images are colored, then we use the depth of three for each color channel as RGB.

### 3.1. Convolutional Layer

This layer is computing the output of the neurons which are connected to the local regions. The output is calculated as a dot product of the weights and input, then adding some bias b to it. The result is 28×28×16 if we are using 16 filters.

- ReLU: This will apply the activation function to each element. After applying the function, the volume size will not be changed, i.e., 28×28×16.
- POOL: This layer is used for the downsampling operation. It is applied along the spatial dimensions, i.e., along the width and height. A long step (stride) is used to downsample the image. We use a stride of 2; thus, the resulting volume will be 14×14×16.
- FC: The fully connected layer computes the valid class scores. Every neuron in this layer has to be connected to the neurons in the previous layer. The size of the output is 1×1×10, where ten corresponds to the number of classes used. Since digit recognition consists of 0-9; thus, ten classes are used one for each category.

Fourteen layer Convolutional Neural network is used, which is based on the VGG Net [5]. An image of shape [W, H, D] = 28×28×1 is used. So in total, we will be having an input of 784 neurons. This image is being fed to the first Convolutional Layer, which contains 64 filters. Then the output is passed to two more convolutional layers 64 filters each. Then we have four more convolutional layers of having 128 filters each and then four more layers of 512 filters each. In every convolutional layer, ReLU is used as an activation function. For the downsampling, a stride of 2 is used after every filter. So the output of the last convolutional layer is having a shape of [3, 3, 512].

After that, the flattening layer is consisting of 4608 neurons. The model will use three fully connected layers to downsample the images ist from 4608 to 2018 then to 1024, and at the end to 10, that will be the class score. Between all these layers, a dropout is used.

The convolutional layer accepts the size in W×H×D, and four hyperparameters are required. Which is the number of filters to be used (K), size of the filter (F), the stride (S) and padding P. The volume produced by the convolutional layer is as follows:

$$Output\ dimensions = \frac{W-F+2P}{S} + 1 \times \frac{H-F+2P}{S} + 1 \times K \quad (1)$$

## 4. Accuracy with different filter sizes

Three types of filter sizes were used in the convolutional layers one 3×3, 5×5 and 7× 7. The layers with the filter sizes large are computationally costly as it can be seen that bigger filter sizes would have to do many operations as compared to the small sizes. 3×3 is almost three times (2.78) computationally costly than the filter sizes of 5×5. 5×5 filters are computationally cheaper, but there are chances of overfilling.

As 5×5 cover a large area at a time, so it becomes computationally cheaper but covers lesser dimensions at a time [6]. [7] have used filters of variable sizes of 1×1 and 3×3 and are based on the Inception V3 model, as shown in Figure 2. This model used the ReLU activation function and is sufficient for dimensionality reduction.

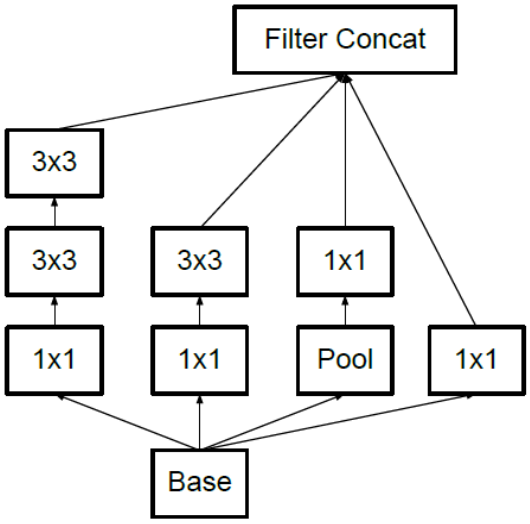


***Figure 2.*** *Inception Modules using filter sizes of 1x1 and 3x3*

The accuracy and the losses using the three different sizes of filters (3×3, 5×5 and 7×7) are given in Figure 3 and Figure 4. The Green, Blue, and Red lines correspond to the filter sizes of 3×3, 5×5 and 7×7 , respectively. As we can see, the 3×3 has the highest accuracy as compared to the other two, while 5×5 has better accuracy than 7×7 but less accuracy than 3×3. The steepness in the graph is the local minima. The graph shows just the result of the 25% of the epochs. We know the training of the Convolutional Neural Networks is a hard task. So we used less number of the epochs to save time. The performance of these filter sizes is given in Table 1.

***Table 1.***

| Filter Size | Accuracy on 1500 Epochs | Loss 10 epochs |
|---|---|---|
| **3 × 3** | Training:- 100%<br>Test:- 98.7% | Training:- 3.4%<br>Test:- 12.5% |
| **5 × 5** | Training:- 94.6%<br>Test:- 98.5% | Training:- 21.4%<br>Test:- 22.8% |
| **7 × 7** | Training:- 91.97%<br>Test:- 97.8% | Training:- 27.6%<br>Test:- 23.02% |

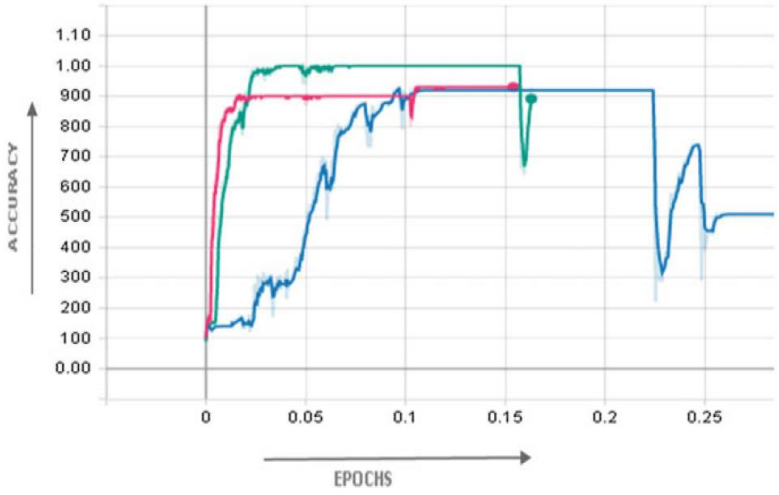


**Figure 3.** *Accuracy using I, II and III filter sizes I: 3×3, II: 5×5 and III: 7×7 Green: 3×3 , Red: 5×5 and Blue: 7×7*

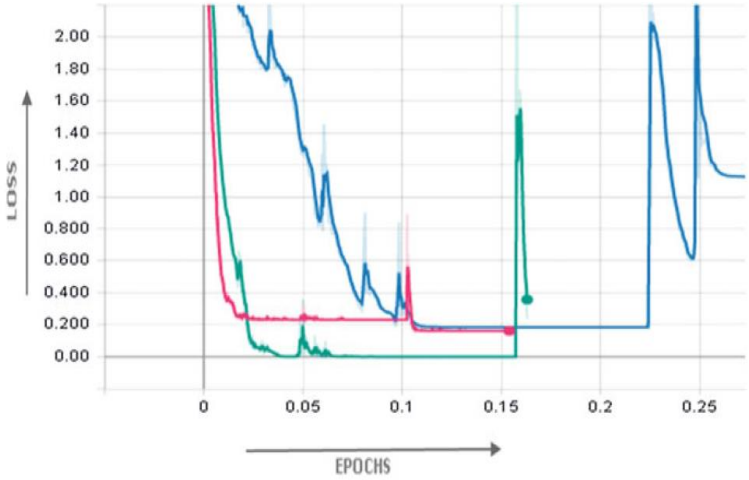


**Figure 4.** *Error Rate using I, II and III filter sizes I: 3×3, II: 5×5 and III: 7×7 Green: 3×3 , Red: 5×5 and Blue: 7×7*

## 5. Conclusion

Filters of 3 different sizes were used like 3×3, 5×5 and 7×7. It has been seen that the accuracy on the training data has been decreased from 100% to 97.8% as we increase the filter size and also the accuracy on the test data set decreases for 3×3 it is 98.7%, for 5×5 it is 98.5%, and for 7×7 it is 97.8%. The loss on the training data and test data per 10 epochs could be seen drastically increasing from 3.4% to 27.6% and 12.5% to 23.02%, respectively. Thus it is clear that using the filters 3×3 having lesser dimensions is giving less loss than those having more dimensions. However, using the lower filter size comes with the cost of computational complexity, which is very crucial in the case of larger data sets.

## 6. Acknowledgments

The task performed under the framework of the FIKP Higher Education Institutional Excellence Program on the Optimization of Natural Resources Based on Modern Technologies Within the theme "APPLYING INTELLIGENT INFORMATION TOOLS TO INVOLVE, REDUCE AND REDUCE PROFESSIONAL LOSSES."

## References

[1] Fergus, R., Zeiler, M.D.: Visualizing and Understanding Convolutional Networks, arXiv, 2013.

[2] Lin, D., Lin, Z., Sun, L., Toh, K-A., Cao, J.: LLC encoded BoW features and softmax regression for microscopic image classification,IEEE International Symposium on Circuits and Systems,- Baltimore, USA, 2017. **https://doi.org/10.1109/ISCAS.2017.8050243**

[3] Islam, R., Saha, O., Osman, S.: Inside of Convolution Filters and Effects of These Filters in Brain MRI Image Classification, International Conference on Innovation in Engineering and Technology (ICIET), Dhaka, 2018. **https://doi.org/10.1109/CIET.2018.8660801**

[4] LeCun, Y., Cortes, C., Burges, C. J.C.: The MNIST Database of Handwritten Digits, http://yann.lecun.com/exdb/mnist/. Accessed 5 September 2019.

[5] Simonyan, K.: Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv, 2014, http://www.image-net.org/challenges/LSVRC/2014/eccv2014.

[6] Zhu, Y., Mak B.: Speeding up softmax computations in DNN-based large vocabulary speech recognition by senone weight vector selection, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, 2017. **https://doi.org/10.1109/ICASSP.2017.7953175**

[7] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision, arXiv, 2015. **https://doi.org/10.1109/CVPR.2016.308**