Optimization of CNN model for image classification

Conference Paper · June 2021

DOI: 10.1109/DTS52014.2021.9497988

3 authors:

Meriam Dhouibi
ENICarthage
7 PUBLICATIONS **51** CITATIONS
SEE PROFILE

Ahmed Karim Ben Salem
Ecole Polytechnique de Tunisie
32 PUBLICATIONS **172** CITATIONS
SEE PROFILE

Slim Ben Saoud
University of Carthage
97 PUBLICATIONS **534** CITATIONS
SEE PROFILE

Optimization of CNN model for image classification

Meriam Dhouibi, Ahmed Karim Ben Salem, Slim Ben Saoud
meriam.dhouibi@ept.rnu.tn, ahmed.bensalem@ept.rnu.tn, slim.bensaoud@gmail.com

Advanced Systems Lab, Tunisia Polytechnic School, University of Carthage, BP 743, 2078, La Marsa, Tunisia

***Abstract*—Convolutional Neural Networks (CNNs) are particularly precise in several fields, especially computer vision where image classification is one of the most researched and commercialized application. Deploying these models on embedded devices requires high throughput and low latency even with limited resources and energy budgets. The complexity of the architecture of CNN models implies a very high computation cost. We are looking in this paper for determining the optimal topology (the number of layers and the number of neurons per layer) that allows us to reduce the model and deploy it in embedded platforms. We have proposed a small CNN architecture that achieves high level accuracy 81.50% on CIFAR-10 with fewer parameters based on the growing approach. For more optimization we used the pruning technique and the results showed that with more optimal architecture we obtained 82.43% accuracy and 15% reduction of the number of parameters.**

Keywords— CNN, Topology, Accuracy, Pruning, Parameters, FPGA, Image classification.

I. INTRODUCTION

Deep Learning (DL) technology have attracted significant attention in various fields such as computer vision, natural language processing (NLP), machine translation, automatic speech recognition, automatic vehicle driving, social media filtering and video games, among others. Thanks to advances in Big Data, the considerable amount of data collected and the powerful resources, DL has achieved impressive results. Depending on the application area, DL employs various models and architectures. CNN is the most commonly used DL model in computer vision tasks such as object recognition and image classification and it demonstrated high performance (84.7% in [1] and 96.4% [2]).

LeNet-5 [3] is the first deep CNN that consists of two convolutional layers (CONV), two fully connected layers (FC), and several pooling layers. With the proliferation of data on the internet and the advance on computational technology, a deeper network AlexNet [1], has been constructed with five CONV layers and three FC layers based on a large scale dataset ImageNet [4]. The depth of CNN has been significantly increased with 16 CONV layers [5], releasing VGG-16. Based on the concept of inception, the authors in [6], introduced GoogLeNet, a 22 layer deep network. Furthermore, using the residual learning approach, He et al. [2] built a deeper network.

However, such complex CNN architectures are hard to implement and require high computation cost in terms of memory footprint, energy consumption, etc. An optimal topology of such CNN models with the same level of accuracy can be deployed easily on embedded platforms.

In this paper, we aim to explore some techniques to optimize a CNN topology for image classification that allow us to achieve the same accuracy but with few parameters in order to reduce the model and to obtain an optimal architecture.

The main contributions of this paper include:

- Testing different topologies of a CNN model for image classification.
- The generalization of a CNN model to avoid the problems of overfitting and underfitting.
- Exploration and deployment of regularization techniques to achieve high accuracy with small architecture.
- The application of the pruning technique to optimize the model.

This paper is structured as follows: The background of the research is provided in section II. The workflow of designing a CNN model is explained in section III. A case study application is presented in section IV. Section V presents the methodology. The experiments and results are discussed on section VI. In the end, a conclusion and possible future works are presented.

II. BACKGROUND

The network depth and the number of neurons describe the network topology. State-of-the-art CNNs [1], consist usually of thousands of neurons arranged into more than 5 hidden layers with millions of parameters. The implementation of such model inference on FPGAs is limited by the available computing resources. One of the major challenges of the successful deployment of a CNN is designing a network with optimized topology. The topology of a CNN model is crucial for the number of FLOPs (floating-point operations), the number of parameters and the required memory. However, the choice of the topology is still mainly determined by trial-and-error. Several common approaches are used to determine the network topology. One of the most successful is the growing approach. It consists of starting with a minimal network that only has the necessary number of input and output nodes (determined by the application and the input features) and then

978-0-7381-3264-8/21/$31.00 ©2021 IEEE

insert new layers or neurons. Another used approach is pruning the network; it consists of stating by a bigger network and prune it. This method is used to achieve simpler but more efficient CNNs [7].

Therefore, in this paper we aim to design a CNN model for image classification task by using the two approaches; growing approach and pruning for topology construction. Furthermore, our motivation is to save storage, memory and FLOPs in order to obtain a more hardware friendly network.

III. CNN

CNNs are neural networks that excel at classifying images and videos. Moreover, CNNs have been used in many other applications such as object recognition, facial recognition, autonomous driving and drone navigation etc.
A typical architecture CNN is divided into two parts: the first is made up of a series of convolutional and pooling (subsampling) layers dedicated to automatic feature extraction, and the second is made up of FC layers dedicated to classification (Fig. 1).

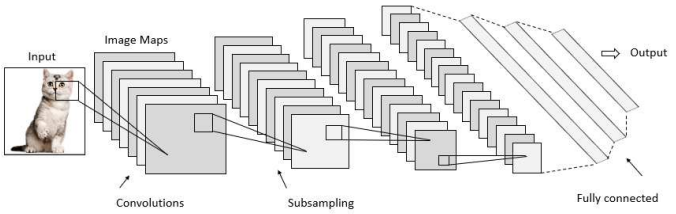

Fig. 1 : CNN architecture

CNN works by considering a small part of the image. These parts are then inspected for features that the network can identify, such as a vertical line, arc or circle. The image is then classified using a combination of features. A number of feature maps are often used to make the network more robust at different levels of contrast, brightness, color saturation, noise, and so on.
There are two types of layers, both of which are made up of feature maps: The first convolutional layer recognizes the input image features. It typically consists of several feature maps; each map recognizes different features. A convolution layer is often followed by a second layer, known as the subsampling or pooling layer. It has the same number of feature maps as the convolutional layer, with each convolutional layer map serving as the input for the corresponding feature map in the next pooling layer. The convolution and pooling layers alternate depending on the network depth until the last pooling layer is reached.
Then there are the fully connected (FC) layers, with the output layer being the last.

The key challenge of implementing CNNs on embedded systems is the computational requirement and the storage capacity. Furthermore, CNNs with a more complex architecture can perform more complex tasks at the expense of increased computation amounts and memory requirements.

IV. IMAGE CLASSIFICATION CASE STUDY

Robotics, human-machine interaction, surveillance, retrieval, transportation, and other fields may all benefit from image classification. It is one of the most involved area in computer vision research. It consists of the classification of images. It searches for regions in a given image that may contain a specific object, then extracts and classifies each of these regions using an image classification model. Thus, a good algorithm is needed for a good image classification method [8]. CNN has become the most widely used model in this type of application following the results of the ImageNet [1] competition.

V. METHODOLOGY

The aim of this work is conducting experiments with several architectures to find an optimal topology that achieves high accuracy with a minimal FLOPS.

In the first part, we will design a CNN model based on the growing approach. We will start with a minimal architecture and then insert new layers as needed while evaluating the performance of the architecture.
A poor performance in a DL model is usually caused by either overfitting or underfitting the data. When a model performs poorly on a training dataset and does not perform well on a new data that means the model is underfitting. Not very many methods are used to overcome the problem of underfitting including the use of deeper network by increasing the number of layers or the number of neurons per layer [9]. Another method is to increase the training time which gives the underperforming model enough time to determine its parameters optimal values.
On the other hand, overfitting occurs when the model cannot generalize the insights from the dataset. It performs well on the training dataset but does not perform well on a new data. To prevent overfitting, several methods have been proposed including data augmentation [10] to make the training dataset more diverse and regularization techniques [11] to force the model to generalize the insights. Two recent approaches to regularization are dropout [12] and batch normalization [1] [13]. In this work, we will use dropout layers to deal with the overfitting problem since dropout aligns more closely with our intuition and goals, as it aims to increase classification efficiency by reducing coadaptation of activations.

In the second part, we will propose a bigger network and prune it to obtain a smaller and more efficient model. In a network, there are usually a large amount of a redundant parameters with small influence on the model performance and costs in computation as well in memory footprint. These parameters can be removed through the pruning process [14].
Pruning is an optimization strategy that involves removing redundant weight tensor values, resulting in faster compressed network with a reduced computational cost. Pruning is often followed by some fine tuning to improve the accuracy. There are many pruning methods in terms of weights, filters channels

and feature maps. We choose the method of filter pruning which is more hardware friendly.

VI. Experiment and Results

An optimal topology of the CNN models with equivalent accuracy, require less communication and less memory bandwidth and can be deployed easily on embedded systems with limited resources. To evaluate our design topology, we conducted experiments on the CIFAR-10 dataset.

A. CIFAR-10 dataset

For our image recognition application, we used the CIFAR-10 [15] general dataset. This dataset contains 60000 32x32 color images divided into ten classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck), each with 6000 images. Fig. 2. shows a few sample images from the CIFAR-10 dataset. In this work, 50000 images have been used for the training process, and the remaining images have been used for the test.

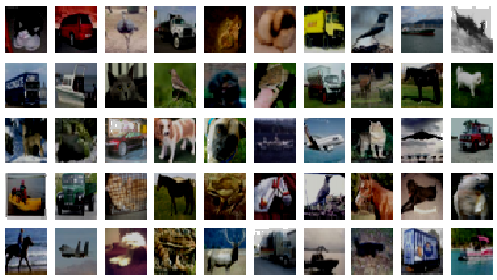

Fig. 2 : CIFAR-10 image samples

B. Dataset Preprocessing

The data preprocessing phase improves the model efficiency. We have modeled all the images in the described dataset with an RGB format as a 3D matrix and stacked them on top of each other in an array. Following that, we have divided the pixels of the images by 255 resulting in pixel values in the range [0.1].
For both training and testing model, we have a total of 60000 images, with a scale of (32×32). Since the images are in RGB format, we have an input shape (32,32,3).

C. Growing approach

The design of the CNN model consists of three main parts; model construction, model training, and model testing.
Once the data is ready to be fed to the model, we must define the model architecture. As we present in Section III, CNN is a sequence of layers, and each layer transforms one volume of activations into another through a differentiable function. The convolutional layer, the aggregation (pooling) layer, and the fully connected layer are the three types of layers we used to construct our model.

- CONV layers compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region to which they are connected in the input volume. We used 32 filters in the first layers of all architectures. As a result, we have a volume of [32x32x32] in the output. The activation function RELU is used in all of the conv layers.

- MAX POOL layers perform a down sampling operation along spatial dimensions (width, height), resulting in a volume of [16x16x32] from the first layer (Max1).

- The fully-connected (FC) layers will calculate class scores, resulting in a volume of size [1x1x10], where each of the ten numbers representing a class score within the CIFAR-10 dataset ten categories. The RELU activation function is used in the first FC layers, while the SoftMax activation function is used in the last layer.

Our goal is to conduct experiments with several architectures. In order to find an optimal topology that achieves similar level of accuracy with a minimal FLOPS. We used a minimal architecture (Arch1) that contains 3 Conv layers with max-pooling in the first two layers, and two FC layers. The architecture configuration is shown in table 1.

Table 1 : Configuration of the Arch1

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
MaxPool1	2x2	16x16x32		
Conv2	3x3	16x16x64	18496	9.437248
MaxPool2	2x2	8x8x64		
Conv3	3x3	8x8x64	36928	4.718656
Fc1		128	524416	1.048704
Fc2		10	1290	0.00257
Total			**582 026**	**16.976682**

We fit the model in 34 epochs and it performs 72.38% validation accuracy. The model does 16,97 Mega FLOPS to classify an image.

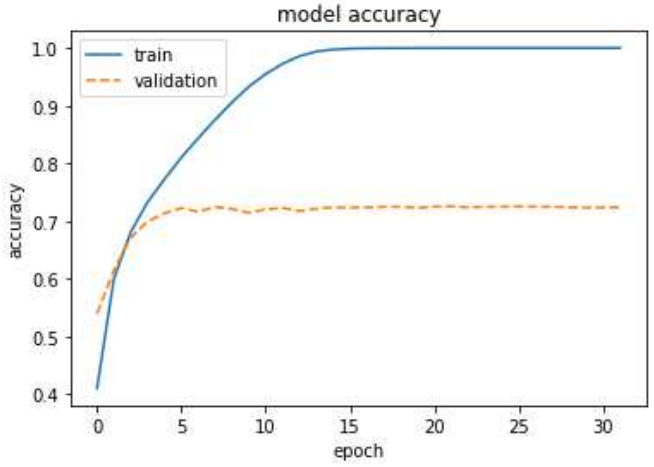

Fig. 3 : Training and validation accuracy of Arch1 during 34 epochs without regularization

The results presented in Fig. 3 shows that the validation accuracy is way below the training accuracy which means that our model is overfitting. We added some dropout layers to the model with a ratio of 20% and 50% and by that we improved the validation accuracy of the model by 5.64% from (72.38 to 78.02%).

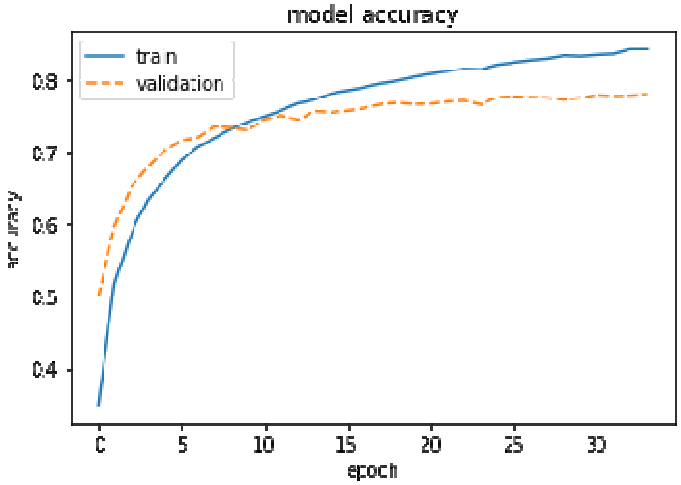

Fig. 4 : Training and validation accuracy of Arch1 during 34 epochs with regularization

Fig. 4 shows that after the 20th epoch, the training accuracy is still above the validation accuracy which may lead to overfitting.

We have proposed to improve our model by adding high-level (128 filters) convolution layer. Its architecture (Arch2) is presented in table 2. It contains 6 Conv layers with max-pooling in the first 3 layers, and 3 FC layers.

We fit the model in 34 epochs and it performs 81.50% validation accuracy. The results are shown in Fig. 5. The model does 82,52 MFLOPS to classify an image. Moreover, we can clearly see in Fig. 5 the synchronization of the training and validation accuracy, which confirm the well generalization of the model on the validation set.

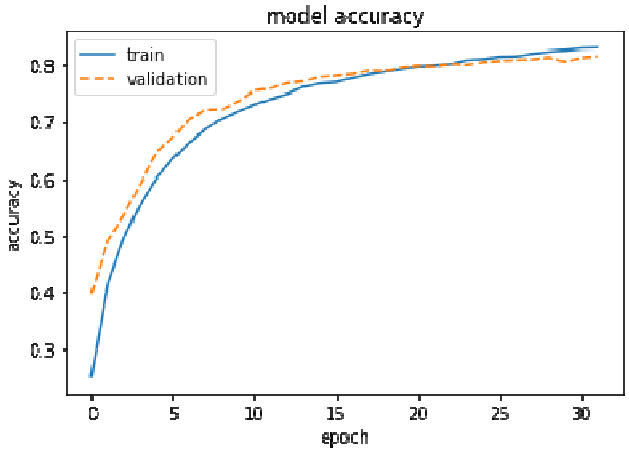

Fig. 5 : Training and validation accuracy of Arch2 during 34 epochs

We used another architecture (Arch3) that contains 8 Conv layers with 4 max-pooling layers, and 3 FC layers. The architecture configuration is presented on table3. We added 2 high-level (256 filters) convolution layer to Arch2.

Table 2 : Configuration of the Arch2

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
Conv2	3x3	32x32x32	9248	18.8744
MaxPool1	2x2	16x16x32		
Conv3	3x3	16x16x64	18496	9.437248
Conv4	3x3	16x16x64	36928	18.874432
MaxPool2	2x2	8x8x64		
Conv5	3x3	8x8x128	73856	9.437312
Conv6	3x3	8x8x128	147584	18.874496
MaxPool3	2x2	4x4x128		
Fc1		2048	2098176	4.195328
Fc2		512	524800	1.049088
Fc3		10	5130	0.01025
Total			**2915114**	**82.522058**

Table 3 : Configuration of the Arch3

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
Conv2	3x3	32x32x32	9248	18.8744
MaxPool1	2x2	16x16x32		
Conv3	3x3	16x16x64	18496	9.437248
Conv4	3x3	16x16x64	36928	18.874432
MaxPool2	2x2	8x8x64		
Conv5	3x3	8x8x128	73856	9.437312
Conv6	3x3	8x8x128	147584	18.874496
MaxPool3	2x2	4x4x128		
Conv7	3x3	4x4x256	295168	9.43744
Conv8	3x3	4x4x256	590080	18.874624
MaxPool4	2x2	2x2x256		
Fc1		2048	1049600	2.098176
Fc2		512	524800	1.049088
Fc3		10	5130	0.01025
Total			**2751786**	**108.73697**

By using the new architecture (Arch3), the model achieved 78.81% validation accuracy in 34 epochs. The model does 108.73697 MFLOPS to classify one image.

The synchronization in the training and validation accuracies shown in Fig. 6 confirms the well generalization of the model. At the 20th epoch, the model performance stops improving and the validation accuracy start dropping.

In this section, we have designed a CNN model to classify images in order to deploy it on embedded FPGA platform. We used the growing approach to determine the optimal architecture of our CNN model. We summarize in table 4 the

accuracies and FLOPS results of each of the three architectures.

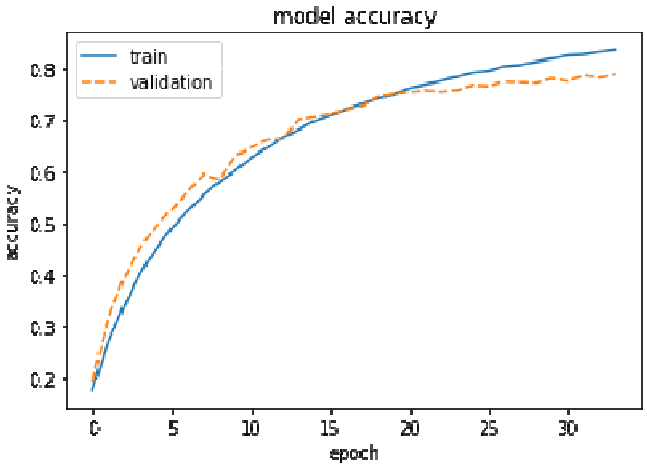

Fig. 6 : Training and validation accuracy of Arch3 during 34 epochs

Table 4 : Experimental results

	Arch1	**Arch2**	**Arch3**
Accuracy (%)	78.02	81.50	79.21
FLOPS (Mega)	16.976682	82.522058	108.73697

The obtained results presented in table 4, show that with Arch2 we obtained the best validation accuracy (81.50%) and a good performance of 82.52 Mega FLOPS. In the next section we will consider Arch2 for more optimization using the pruning approach.

D. Pruning approach

We used the iterative filter pruning that consists of removing the unimportant filters from the CNN part of a trained network. The first step is to rank filters to determine which filters are important and which are not. Then, the last important filters are removed, followed by the fine-tuning of the algorithm. At this stage, a decision to continue or to stop the pruning process can be made. In each iteration, the fewer filters are discarded, the less damage is done to the network, the less retraining time is required for restoring the accuracy. However, a very important number of iterations is needed to reach a satisfactory compression rate.

When deploying CNN models on embedded platforms with limited resources, it is necessary to optimize the model and reduce the size. When using the pruning approach, it is better to start with a large network and prune it after training rather than training a small network. In this part of our work, we removed the dropout layers from Arch2 and we re-trained the model. It performs 72.87% validation accuracy in 34 epochs. After ranking the filters, we discarded 5% of filters in each iteration. As shown in table 5, at the 6th iteration, we discarded 26% of filters and reduced the number of parameters by 15% and the model accuracy is improved with 9.56% (82.43%). At the 7th iteration, the performance of the model starts increasing.

Table 5 : Pruning results per iteration

Discarded filters	**P =0.05 (5%)**		
	Filters	**Parameters**	**Accuracy (%)**
Original	448	1342250	72.87
Iteration 1	426	1290122	79.03
Iteration 2	405	1248743	80.59
Iteration 3	385	1205060	82.72
Iteration 4	366	1183603	82.04
Iteration 5	348	1155251	82.28
Iteration 6	331	1141025	82.43
Iteration 7	314	1114554	82.18
Iteration 8	298	1096663	81.82
Iteration 9	283	1075712	81.59
Iteration 10	269	1057021	80.24

These obtained results (table 4) show that with Arch2 we obtained the best validation accuracy (81.50%) and a good performance of 82.52 Mega FLOPS. We considered Arch2 as an optimal architecture with a good performance. In the second part of our work, we performed the pruning approach on Arch2 for more optimization. As shown on table 5, the model accuracy is improved with 9.56% (82.43%) and the number of parameters is reduced by 15%.

By using the growing approach and pruning technique for topology construction, we obtained an optimal architecture Arch2 that can be deployed more easily on embedded hardware targets with limited resources, much less computations and a better performance.

VII. CONCLUSION AND FUTURE WORK

In this work, we have designed a CNN model for image classification by using two approaches in order to obtain an optimal architecture for implementation on FPGA platform. We also used the dropout layers as a regularization technique to overcome the overfitting problem.

Additionally, we have shown that, thanks to the use of an optimization technique based on a reduction of the number of filters in the CNN part of the model, the pretrained CNN model can do minimal FLOPS, which allow the optimization in terms of computation and storage resources.

Our ongoing works focuses on the deployment and the test of the optimized CNN model into an FPGA SoC.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, doi: 10.1109/CVPR.2016.90.

[3] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comput.*, vol. 1, no. 4, 1989, doi: 10.1162/neco.1989.1.4.541.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2010, doi: 10.1109/cvpr.2009.5206848.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[6] C. Szegedy *et al.*, "Going Deeper with Convolutions (GoogleLeNet)," *J. Chem. Technol. Biotechnol.*, 2016, doi: 10.1002/jctb.4820.

[7] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework," in *SenSys 2017 - Proceedings of the 15th ACM Conference on Embedded Networked Sensor Systems*, Nov. 2017, vol. 2017-Janua, pp. 1–14, doi: 10.1145/3131672.3131675.

[8] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of Deep Learning for Object Detection," in *Procedia Computer Science*, 2018, vol. 132, doi: 10.1016/j.procs.2018.05.144.

[9] D. M. Hawkins, "The Problem of Overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1. 2004, doi: 10.1021/ci0342472.

[10] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2003, vol. 2003-January, doi: 10.1109/ICDAR.2003.1227801.

[11] K. Zeeshan, "The Impact of Regularization on Convolutional Neural Networks," 2018.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, 2014.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 1.

[14] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks," *Synth. Lect. Comput. Archit.*, 2020, doi: 10.2200/s01004ed1v01y202004cac050.

[15] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Sci. Dep. Univ. Toronto, Tech.* , 2009, doi: 10.1.1.222.9220.