CNN Architecture Assessment: Exploring Depth, Width, and Kernel Size for Image Classification

Eduardo T. Buss[1], **Erick Radmann**[1], **Ruhan Conceicao**[1], **Bruno Zatt**[1], **Luciano Agostini**[1]

[1]Universidade Federal de Pelotas (UFPel)
Pelotas – RS – Brazil

{etbuss,eradmann,radconceicao,zatt,agostini}@inf.ufpel.edu.br

Abstract. *This paper investigates the impact of different convolutional neural network (CNN) architectures on image classification performance using the CIFAR-10 dataset. We evaluate variations in the number of convolutional layers, kernel size, and number of filters. Experiments indicate that increasing the number of filters and slightly enlarging kernel size generally improves accuracy, while deeper models do not always yield better results. The CIFAR-10 dataset contains 60,000 color images across 10 classes, with 45,000 for training, 5,000 for validation, and 10,000 for final testing.*

1. Introduction

Convolutional Neural Networks (CNNs) are the cornerstone of modern computer vision, demonstrating remarkable performance in tasks such as image classification, object detection, and semantic segmentation [LeCun et al. 1998]. Their strength lies in hierarchical extraction of spatial features from raw pixel data.

CIFAR-10 dataset is one of the most widely used benchmarks for image classification [Krizhevsky and Hinton 2009]. It contains 60,000 natural images of size 32×32 pixels, equally distributed across ten categories. Due to its balanced distribution and moderate complexity, CIFAR-10 is ideal for evaluating the effect of architectural variations on CNN performance.

In this work, we analyze how fundamental architectural hyperparameters (number of convolutional layers, kernel size, and number of filters) affect model accuracy. By systematically evaluating these factors, we aim to provide insights into trade-offs between depth, capacity, and kernel configuration in lightweight CNNs.

2. Methodology

Our experimental setup follows a controlled evaluation scheme to isolate the effect of architectural variations. The CIFAR-10 dataset was partitioned into three distinct subsets: 45,000 images for training, 5,000 for validation, and 10,000 for final testing. This rigorous data separation serves specific purposes in our comparative analysis.

The training set (45,000 images) is used to optimize model parameters through backpropagation. The validation set (5,000 images) serves as an unbiased monitor during training to track model performance, implement early stopping mechanisms, and prevent overfitting. Crucially, validation performance guides decisions about training convergence without contaminating the final evaluation. The test set (10,000 images) remains

completely unseen during the entire training and validation process, ensuring an unbiased final assessment of each architectural configuration's generalization capability.

All models shared identical optimization and training protocols; only architectural hyperparameters were systematically varied. This controlled approach ensures that performance differences can be attributed solely to architectural choices rather than training inconsistencies.

Input images were normalized to zero mean and unit variance per channel. No data augmentation was applied in this initial study to focus on pure architectural effects.

We evaluated eight configurations combining:

- Number of convolutional layers: 2 or 8;
- Kernel size: 3×3 or 5×5;
- Number of filters per layer: 32 or 128.

Zero-padding was applied to preserve spatial dimensions (1 for 3×3 kernels, 2 for 5×5). Convolutional layers were followed by ReLU activations [Krizhevsky et al. 2012] and max-pooling for downsampling.

All models were trained using the Adam optimizer [Kingma and Ba 2014] with a learning rate of 0.001 and batch size of 128. Training proceeded for 50 epochs. The final evaluation metric was top-1 accuracy measured exclusively on the test set.

3. Results

Table 1 summarizes the accuracy achieved by each configuration, alongside the difference relative to the baseline architecture (2 layers, 3×3 kernels, 32 filters).

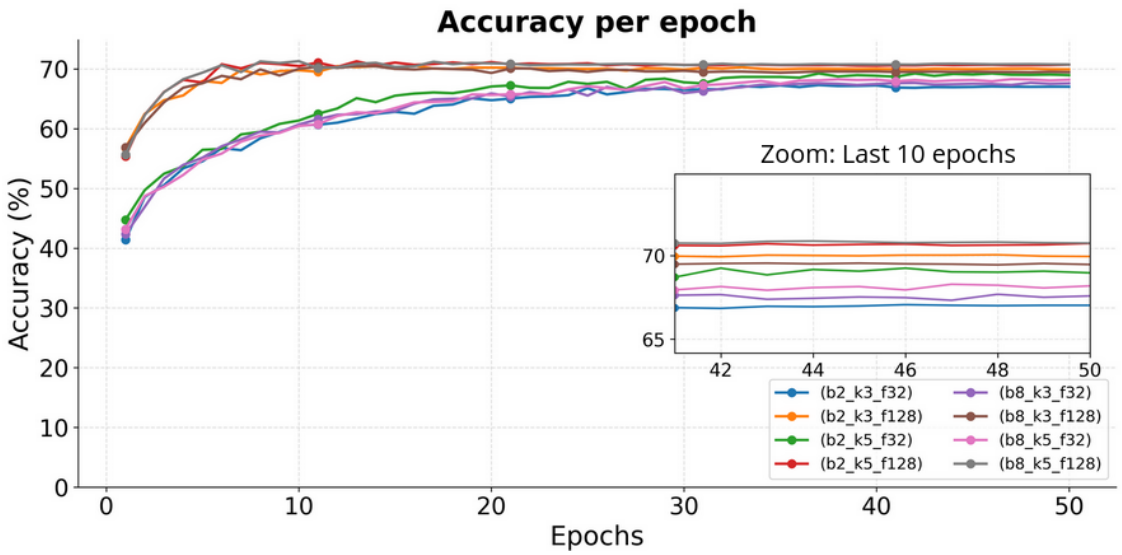
Table 1: Experimental Results

#Conv2D	**Kernel Size**	**#Filters**	**Test Acc. (%)**	Δ_{acc} **(pp)**
2	3×3	32	66.77	Anchor
2	3×3	128	69.99	+3.22
2	5×5	32	67.89	+1.12
2	5×5	128	70.47	+3.70
8	3×3	32	66.97	+0.20
8	3×3	128	69.47	+2.70
8	5×5	32	68.16	+1.39
8	5×5	128	70.89	+4.12

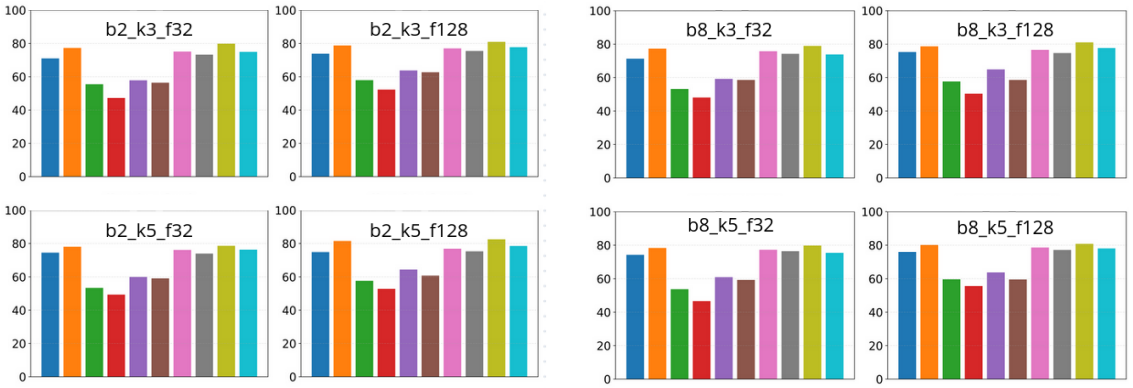
The results show that increasing the number of filters consistently improves accuracy by about 2-3%. Enlarging the kernel size provides modest improvement, while increasing depth shows mixed effects depending on other architectural parameters.

The best-performing configuration was 8 convolutional layers, 5×5 kernels, and 128 filters, achieving 70.89% test accuracy. This indicates that deeper networks (8 layers vs 2 layers) combined with larger kernels and more filters achieve superior performance. The results demonstrate that the optimal configuration for CIFAR-10 involves both increased depth and capacity, with larger kernels (5×5) and more filters (128) consistently outperforming their smaller counterparts across both shallow and deep architectures [Lin et al. 2014], (Figure 1a) displays the training convergence curves for

all eight configurations over 50 epochs, revealing that configurations with more filters achieve higher final accuracy. The class-wise performance comparisons show that 2-layer networks (Figure 1b) and 8-layer networks (Figure 1c) exhibit different patterns across CIFAR-10 categories, with deeper architectures generally providing more consistent performance across all ten classes.



(a) Accuracy of configurations over epochs



(b) 2 layers

(c) 8 layers

(d) CIFAR-10 classes

Figure 1: CIFAR-10 results for CNNs varying the number of layers b, kernel size k, and filters f. (a) Test accuracy over training epochs for all eight configurations; (b) Per-class test accuracy for b=2; (c) Per-class test accuracy for b=8; (d) CIFAR-10 class list.

4. Conclusion

This study systematically analyzed the impact of CNN architectural variations on CIFAR-10. Accuracy improvements were more strongly associated with the number of filters and

kernel size than with depth alone. The best performance was achieved with 8 layers, 5×5 kernels, and 128 filters, reaching 70.89% test accuracy.

Future work will explore regularization, batch normalization [Ioffe and Szegedy 2015], dropout [Srivastava et al. 2014], data augmentation, and modern architectures such as ResNet and DenseNet [He et al. 2016, Huang et al. 2017] to further investigate performance limits on benchmark datasets. Additionally, we plan to investigate model compression techniques including quantization and pruning to optimize model efficiency while maintaining classification accuracy.

References

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, M., Chen, Q., and Yan, S. (2014). Network in network. In *International Conference on Learning Representations (ICLR)*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, volume 15, pages 1929–1958.