Unraveling the Balance: Depth vs. Width in CNN Architectures

Yatish Chandra Sai Udutha
Student ID: 23030677
GitHub Repository Link

I. Introduction

Convolutional Neural Networks (CNNs) are widely used in image recognition and computer vision tasks. The architecture of a CNN involves key design choices, including its depth (number of layers) and width (number of filters per layer). These choices significantly influence the ability of the model to learn features, its computational cost, and its performance. In this tutorial, we explore how increasing depth or width affects CNN performance, using experiments and visualizations to demonstrate trade-offs.

II. Literature review

Depth provides greater support for hierarical data processing such as images[1] and enables neural networks to model complex compositional functions, which are infeasible for shallow networks[2], Depth also improves the performance of the model, but the architecture should be designed carefully[3].

The width helps reduce the computational cost by increasing the number of filters per layer and reducing the depth at the same time for better performance[4]. After some time, narrow networks but with well-designed architectures can also achieve good performance in applications[5]. Finally, the focus was on the trade-off between depth and width, i.e., scaling the balance between them is essential for better model performance with less computational cost[6].

III. Theory

Depth in CNNs: Refers to the number of convolutional layers. Deeper networks can extract more hierarchical features, learning simple edges in early layers and complex patterns in deeper layers. Depth Definition If a CNN has L layers in total, including:

- Conv: The total count of convolutional layers,
- Pool: The total count of pooling layers,
- Fully: The total count of fully connected layers,

The total depth D of the CNN is: D = Conv + Pool + Fully

There is a chance of Vanishing/exploding gradients or Overfitting if training data is insufficient.

Width in CNNs: is the total number of filters in each convolutional layer. Wider networks can capture more features in parallel at each layer, improving feature diversity.

If a layer l in a CNN has f_1 filters, the width of the network is W is the maximum number of filters across all layers:

$$W = \max(f_1, f_2, \ldots, f_D)$$

There are chances of problems such as increased computational cost and reduced performance improvement.

IV. Dataset

CIFAR-10 (10 classes of 32x32 color images) has been used to explain the impact of the two techniques on the dataset.

V. Experiment Design

- Baseline Model: A simple CNN with three convolutional layers and a fixed number of filters is created and used as a reference point to evaluate depth and width variations.
- Exploring the Impact of Layer Depth: The convolutional layers used in this experiment are 2, 3, 6, 7 and 8, and the number of filters per layer is constant, i.e., 32.
- Exploring the Impact of Layer Width: The number of filters in each layer is taken as 16, 32, 64, 128 and 256 filters. Keep the number of convolutional layers constant (e.g. 2 layers).
- Metrics: 1. Accuracy: The accuracy of the test set to measure performance.
  2. Training time: Time required to train the model.
  3. Model Size: Number of parameters in the model.

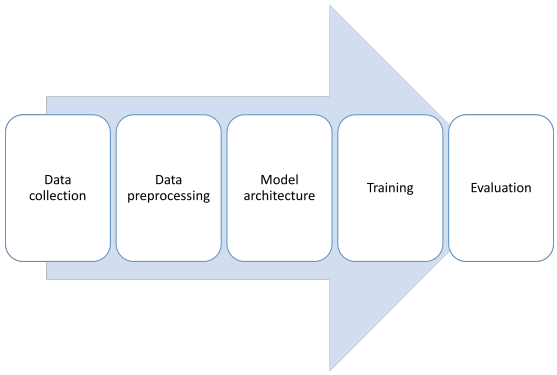

Fig. 1. Experiment Flowchart.

Fig. 1 shows the methodology by which the experiment is conducted. In the code, basic normalization on the images is performed as a data preprocessing step.

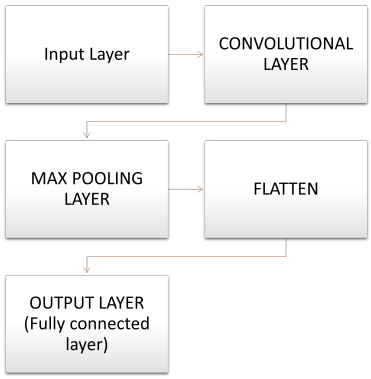

Fig. 2. CNN Architecture flowchart.

Fig. 1 shows the methodology by which the experiment is conducted. In the code basic normalization on the images is performed as a data preprocessing step.

VI. Training Results

In Fig. 3, the depth vs. accuracy plot explains the accuracy improved with an increase in depth up to 3 and later on accuracy decreases, likely due to overfitting.

The width vs. accuracy plot explains that accuracy peaks at 50 filters per layer while balancing feature diversity and efficiency. Beyond 50 layers, the accuracy is diminishing due to more parameters in it.

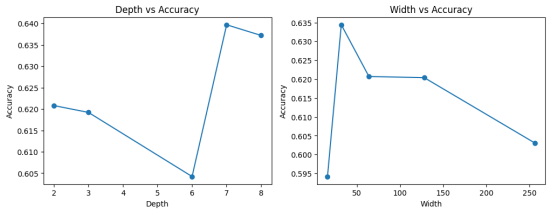

Fig. 3. Model accuracy as a function of depth - width

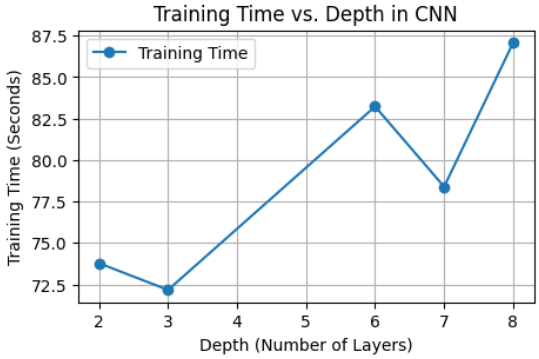

Fig. 4. Training time as a function of depth

In Fig. 4, the training time drops at depth 3 might indicate that the network reached optimal configuration with fewer layers and beyond depth 3, time increases due to the addition of more parameters. Increasing depth affects computational cost. So, after a certain point, it may not be efficient.

In Fig. 5, the model size is in a linear relationship with the width. Even though wider networks capture more features, it also takes more memory space. So, the width consideration is also critical in the architectural design.

In Fig. 6, the best values from the above experiments are depth 3 and width 50 are taken for the final model. With these values as parameters, the final experiment obtained an accuracy of more than 70%.

Initially, the validation loss decreases but starts increasing after the 4 epochs, which maybe is a chance of overfitting. However, the training loss is steadily decreasing, indicating that the model is learning properly.

The same can be interpreted from the accuracy plot. The training accuracy is increasing steadily, showing that the model fits well with the training data. The other is that the validation accuracy increases rapidly and then sticks around 70%. The gap between training and validation accuracy widens, which also suggests the overfitting issue.

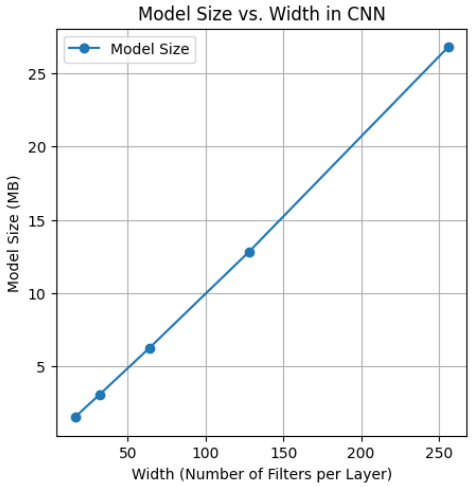

Fig. 5. Model size as a function of width

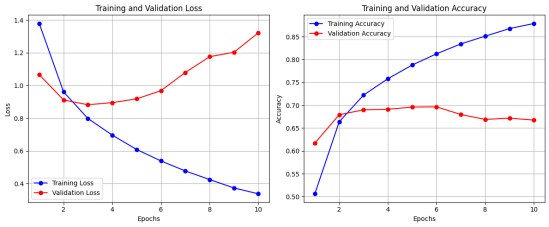

Fig. 6. Training results

VII. Key Points

- Depth and width must be tuned carefully to balance feature learning, computational efficiency, and model generalization.
- Understanding architectural trade-offs helps design CNNs that are well-suited for specific tasks and hardware constraints.
- Further optimization through techniques like regularization, early stopping, or advanced scaling strategies (e.g., EfficientNet) can enhance model performance.

VIII. Future Scope

In this tutorial, the impact of width and depth parameters on the convolutional neural network is discussed. But in the end, there is an overfitting issue. That can be solved by using regularization techniques such as dropout and batch normalization or by data augmentation techniques.

IX. Conclusion

This tutorial provides a detailed understanding of how depth and width impact CNNs, offering insights for designing and optimizing architectures. Future work can expand into advanced architectures, automated design techniques, and application-specific optimizations to address real-world challenges.

By applying these principles, we can build efficient, high-performing CNN models tailored to their specific needs.

References

[1] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[2] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," *Advances in neural information processing systems*, vol. 29, 2016.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] S. Zagoruyko, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[5] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.