Key Parameter Evaluation in Convolutional Neural Networks

Joan Caballero and Oriol Miró

MAI, UPC
Computational Intelligence, Practice 2
December 21, 2024

Convolutional Neural Networks (CNNs) are a well-known type of neural network widely used in AI and computer vision applications since their origin in the 1990s. Tuning and parameter configuration are crucial steps in optimizing CNN performance. This paper analyzes the impact of various CNN parameters on accuracy, exploring the number of convolutional blocks, type of non-linear hidden layer, initial filter size, and data split. We tested a total of 168 parameter configurations using a dataset containing 28×28 silhouette images from 101 different classes. Our results show that the optimal number of convolutional blocks is three, ReLU outperforms Sigmoid, larger filter sizes tend to perform better, and splits with larger proportions of training data obtain higher accuracies.

1 Introduction

Convolutional Neural Networks (CNNs)[4] have revolutionized the field of computer vision, demonstrating outstanding performance in tasks such as image classification, object detection, and segmentation [3, 8, 2, 10]. Their ability to automatically learn hierarchical feature representations from raw image data makes them particularly well-suited for image recognition tasks. However, the performance of CNNs heavily depends on the architectural choices and hyperparameter settings, such as the number of convolutional layers, filter sizes, activation functions, and training data proportions.

Selecting optimal configurations for these parameters is key but challenging due to the vast space of possible combinations and the computational resources required to train deep networks. Understanding how these parameters influence the model's performance can help create more efficient and effective CNN architectures for specific tasks.

In this study, we analyze the effect of various key parameters on the performance of CNNs in the context of image classification using the CalTech 101 Silhouettes dataset [5]. This dataset consists of 28×28 pixel images representing silhouettes of 101 different object classes, providing a suitable benchmark for evaluating CNN configurations on a multi-class classification problem.

Specifically, we study the effects of:

- **Number of Convolutional Blocks (NB)**: Comparing models with varying depths to understand how the number of layers influences feature extraction and classification accuracy.
- **Non-Linear Activation Functions in Hidden Layers (NHL)**: Evaluating the performance differences between Sigmoid and Rectified Linear Unit (ReLU) activations, which have distinct properties affecting convergence and gradient flow.
- **Filter Sizes (FS)**: Assessing how the initial filter size in convolutional layers impacts the model's capacity to capture spatial features at different scales.
- **Training Data Proportions**: Analyzing the effect of different splits of training, validation, and test data on model performance, showing the importance of sufficient training data.

2 Related Work

The optimization of CNN architectures and hyperparameters has been a subject of extensive research. Specifically for the CalTech 101 Silhouettes dataset, Marlin et al. [5] introduced it as a benchmark for evaluating generative and discriminative models. They achieved a classification accuracy of 65.4% using a Deep Belief Network. Later, Sohn et al. [9] improved upon this result, reaching 72.6% accuracy with a Deep Energy Model.

In the context of CNN architectures, Simonyan and Zisserman [8] showed that increasing the depth of networks can significantly improve performance, which aligns with our findings on the number of convolutional blocks. He et al. [2] introduced residual connections to facilitate training of very deep networks, addressing the degradation problem observed in our 4-block configurations, altough we did not explore it.

Regarding activation functions, Nair and Hinton [7] demonstrated the effectiveness of ReLU in deep neural

networks, which is consistent with our observations of ReLU outperforming Sigmoid.

3 Methodology

We explore the CalTech 101 Silhouettes dataset [5],consisting of 8,671 binary 28x28 images representing 101 distinct silhouettes. Figure 1 showcases four sample images from the dataset.

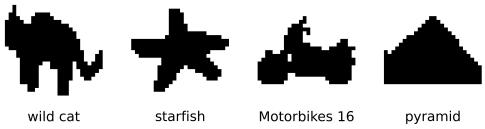

Figure 1: *Sample images from the CalTech 101 Silhouettes dataset.*

Our objective is to assess how different architectural choices and hyperparameters affect the classification performance; we measure this using the mean accuracy and the weighted F1-score.

3.1 Data Preprocessing

We utilized the `caltech101_silhouettes_28.mat` dataset file provided in the assignment description; it was loaded using `scipy.io` and processed with PyTorch tensors. The images were reshaped to dimensions $(N, 1, 28, 28)$ to conform to the input shape expected by CNNs; labels were adjusted to be zero-indexed, ranging from 0 to 100.

We experimented with three different splits of the dataset into training, validation, and test sets:

1. 80% training, 10% validation, 10% test
2. 40% training, 20% validation, 40% test
3. 10% training, 10% validation, 80% test

The data was randomly shuffled before splitting, and data loaders were created with a batch size of 32 for both training and evaluation phases.

3.2 Model Architecture

Our CNN models followed a sequential architecture comprising:

1. **Input Layer**: Accepting 28×28 pixel images.
2. **Convolutional Blocks**: A series of NB convolutional blocks; each block consisted of:
   (a) A convolutional layer with kernel size 3×3, filter size FS, and padding 1, producing feature maps of the same spatial dimensions as the input.
   (b) A Non-linear Hidden Layer (NHL) using either the ReLU or Sigmoid activation function.
   (c) A max-pooling layer of size 2×2 with stride 2, halving the spatial dimensions.
3. **Fully Connected Layer (FC)**: Flattening the output from the last convolutional block and mapping it to 101 output classes.
4. **Output Layer**: Added to adhere to the assignment's structure, but the loss function handles it internally. Later in Section 3.3 it will be seen we chose "None".
5. **Cost Function Layer (CFL)**: Computing the loss using the specified cost function.

We varied the number of convolutional blocks (NB) and the initial filter sizes (FS); for networks with multiple blocks, the filter sizes in successive blocks were doubled. For instance, with $FS = 32$ and $NB = 3$, the filter sizes are 32, 64, and 128 in the first, second, and third blocks, respectively.

3.3 Hyperparameter Tuning

Prior to the main experiments, we conducted hyperparameter tuning to identify suitable values for key training parameters. We tested various values for learning rate, weight decay, number of epochs, optimizer betas (for the Adam optimizer), activation functions in the output layer, and cost functions.

The hyperparameters evaluated were:

- **Learning Rates**: 0.01, 0.001, 0.0001
- **Weight Decays**: 0, 0.0001, 0.001
- **Epochs**: 30, 50
- **Betas for Adam Optimizer**: $(0.9, 0.999)$, $(0.5, 0.999)$, $(0.9, 0.9)$
- **Output Layer Activations**: None, Softmax
- **Cost Functions**: Cross-Entropy Loss, Negative Log-Likelihood Loss

For hyperparameter tuning, we fixed the network architecture to $NB = 3$ convolutional blocks; initial filter size $FS = 32$; ReLU activation in the NHLs; data split of 80% training, 10% validation, and 10% testing.

Each combination of hyperparameters was evaluated by training the model and measuring the validation accuracy and F1-score; experiments were performed using the Adam optimizer. Based on the validation performance, we selected the following hyperparameters for the main experiments:

- **Learning Rate**: 0.001
- **Weight Decay**: 0
- **Epochs**: 50
- **Betas**: $(0.9, 0.9)$
- **Output Layer Activation**: None
- **Cost Function**: Cross-Entropy Loss

Note on the "*None*" for the output layer activation: the Cross-Entropy Loss function in PyTorch already internally applies the softmax operation to the raw output

logits during loss computation. Adding an explicit softmax activation in the output layer would be redundant and could lead to numerical instability.

3.4 Experimental Setup

Using the selected hyperparameters, we conducted the main experiments to evaluate the impact of varying network configurations:

- **Number of Convolutional Blocks (NB)**: We tested NB ranging from 1 to 4.
- **Initial Filter Sizes (FS)**: We varied FS among 2, 4, 8, 16, 32, 64, and 128.
- **Activation Functions in NHLs**: We compared the performance of ReLU and Sigmoid activation functions.
- **Data Splits**: We utilized the three different data splits mentioned earlier.

3.4.1 Limitations on the Number of Convolutional Blocks

An important consideration was the maximum number of convolutional blocks (NB) we could use without reducing the spatial dimensions of the feature maps to invalid sizes. Starting with input images of size 28×28, each convolutional block maintains the spatial dimensions during convolution (due to padding) but reduces them by a factor of 2 with the max-pooling layer. Thus, after each block, the spatial dimensions are halved.

After four convolutional blocks, the spatial dimensions reduce as follows:

- **Block 1**: $28 \times 28 \rightarrow 14 \times 14$
- **Block 2**: $14 \times 14 \rightarrow 7 \times 7$
- **Block 3**: $7 \times 7 \rightarrow 3 \times 3$
- **Block 4**: $3 \times 3 \rightarrow 1 \times 1$

Adding a fifth convolutional block would reduce the dimensions below 1×1, making it infeasible ($1 \times 1 \rightarrow 0 \times 0$), so we limited the maximum number of blocks to $NB = 4$.

3.4.2 Experiment Execution

For each combination of NB, FS, NHL activation, and data split, we conducted three runs; each run was performed with a different random seed to account for variability due to weight initialization and data shuffling. We then computed the average mean accuracy and weighted F1-score across the three runs.

3.5 Evaluation Metrics

We assessed the models using two primary metrics:

- **Mean Accuracy**: The proportion of correct predictions over the total number of samples; averaged over the three runs.
- **Weighted F1-Score**: A harmonic mean of precision and recall; weighted by the number of true instances in each class; averaged over the three runs.

These metrics were evaluated on the test set for each run. It is important to consider the class imbalance present in the dataset. The number of images per class varies significantly, with some classes like *Airplanes Side 2* and *Motorbikes 16* making up 9.2% of the dataset, while others like *inline skate* and *binocular* making up only 0.36% and 0.38% respectively. This imbalance may bias the model towards classes with higher representation, making metrics like the weighted F1-Score particularly important for fair assessment across all classes. Even then, we compute the two metrics as the assignment constrained us to use mean accuracy.

4 Results

We present the most significant results of our study, focusing on three analyses of CNN performance based on accuracy: the number of convolutional blocks, the type of non-linear hidden layer, and the data split among training, validation, and test sets.

Despite having computed both the mean accuracy and F1-score, having to analyze our results w.r.t. both adds a lot of noise; hence, we focus on mean accuracy as per the assignment, given its near-perfect Pearson correlation with F1-score (0.9988).

4.1 Analysis of the Number of Convolutional Blocks

In this section, we analyze the influence of the Number of Convolutional Blocks (NB) on accuracy. Figure 2(a) compares the average accuracy for different configurations of Convolutional Neural Networks (CNNs) varying NB and initial Filter Size (FS). For this study, the Non-Linear Hidden Layer (NHL) is fixed to ReLU, and the split of training, validation, and test sets is fixed to 80%, 10%, and 10%, respectively.

We observe different patterns depending on the CNN configuration. Overall, accuracy tends to increase when the number of blocks increases from 1 to 2. A single block limits the CNN to capturing basic features, while increasing to two blocks allows the CNN to learn more abstract features. As the number of blocks continues to increase, the CNN captures increasingly complex features. For initial filter sizes greater than 8, CNNs perform better with 3 number of blocks, while this improvement does not occur for lower FS. At deeper convolutional blocks, filters can capture more abstract and complex features compared to initial blocks, which focus on finer, low-level features. Given the large number of classes, having a substantial number of filters in deeper blocks allows each filter to specialize in recognizing abstract features specific to a class. However,

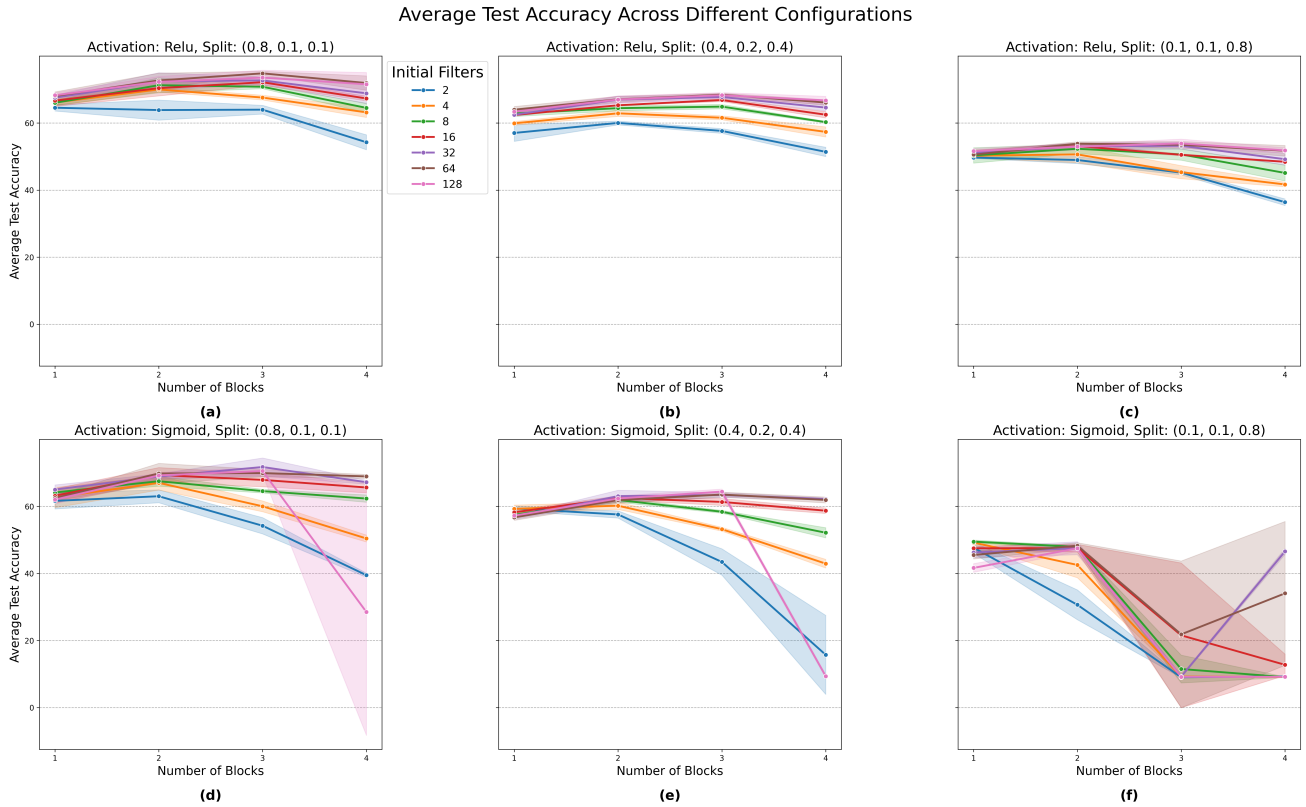

Figure 2: ***Average Test Accuracy Across Different Configurations.*** *The 2×3 grid displays the average test accuracy versus the number of blocks for various initial filter settings. The top row uses the ReLU activation function, while the bottom row uses Sigmoid. Columns represent data splits of 0.8/0.1/0.1, 0.4/0.2/0.4, and 0.1/0.1/0.8 respectively. Shaded areas indicate standard deviation.*

having a small number of filters in deeper blocks causes them to focus on abstract features from multiple classes, preventing specialization in any single class. Additionally, for all FS values, having 4 NB decreases the average accuracy compared to 3 NB. For input images of size 28×28, the output resolution for configurations with 4 NB is 1×1, as explained in more detail in 3.4.1; this single value is used by the CNN to predict the image class. Adding an additional block can help the CNN extract more complex features, but the small initial resolution of the images and the single output value cause the network to lose accuracy, even performing worse than with only 1 convolutional block for smaller FS.

$FS = 2$ performs the worst across all NB. This is because the small number of filters forces each filter to specialize in multiple classes, as explained earlier. Conversely, a higher number of filters achieves greater accuracy across all NB by allowing each filter to specialize in a single class. The optimal configuration is 3 NB with an initial FS of 64, achieving the highest average accuracy of 74.81%.

4.2 Analysis of the Non-Linear Hidden Layer

In this section, we analyze the influence of the Non-Linear Hidden Layer on accuracy. Figure 2(a, d) compares the average accuracy of CNNs using ReLU (a) and Sigmoid (d) as NHLs. For this study, varying numbers of convolutional blocks and initial filter sizes are explored, and the data is split into 80% training, 10% validation, and 10% test sets.

ReLU consistently achieves higher accuracies across all NBs and FSs than Sigmoid. With $NB = 1$, all ReLU configurations outperform Sigmoid. At $NB = 2$, most ReLU configurations achieve accuracies above 70%, while no Sigmoid configuration reaches this threshold. For $NB = 3$, both ReLU and Sigmoid configurations with larger FSs obtain accuracies above 70%, but ReLU configurations still maintain higher accuracies. At $NB = 4$, ReLU configurations remain consistent within the 60-70% accuracy range, whereas Sigmoid exhibits more variability, particularly in configurations with smaller FSs that achieve the lowest accuracies, with one falling below 10%.

These results demonstrate the consistent superiority of ReLU over Sigmoid across all numbers of blocks and filter sizes, especially in deeper CNNs with smaller filter sizes. This superiority is de to ReLU's resistance to the vanishing gradient problem, maintaining a gradient of 1 for positive inputs ensuring that gradients do not vanish as easily. In contrast, Sigmoid's outputs range between 0 and 1, making it susceptible to the vanishing gradient problem and preventing effective weight

updates, especially with low outputs near zero. These findings support recent studies indicating that ReLU outperforms Sigmoid in CNNs [1, 6].

4.3 Analysis of the Split on Training, Validation, and Test

In this section, we analyze the influence of the data split on training, validation, and test sets on accuracy. Figure 2 displays the average accuracies obtained by all CNN configurations varying in number of blocks, initial filter sizes, and NHLs. For this study, we compare results from configurations using different data splits, examining the impact of the split on ReLU (first row) and Sigmoid (second row).

Overall, for both NHLs, splits with a higher percentage of training data achieve more consistent and higher accuracies than splits with a lower percentage of training data. For the first split (0.8, 0.1, 0.1), ReLU configurations remain consistent, with most accuracies in the 60-70% range, while Sigmoid configurations are more variable, with some cases achieving accuracies lower than 50%. However, configurations with higher FS still maintain accuracies in the 60-70% range across all NB. With the split (0.4, 0.2, 0.4), the decrease in training data negatively affects both NHLs. In both cases, accuracies decrease across all configurations. For Sigmoid, this results in more variable outcomes across different number of filters, leading to much lower accuracies for configurations with smaller FS. Lastly, for the split with the lowest percentage of training data (0.1, 0.1, 0.8), results for ReLU decrease noticeable, with none exceeding 55% accuracy. However, the results remain consistent, with higher FS achieving better accuracies than lower FS as in the other splits. For Sigmoid, the results decrease considerably, with no clear pattern. Accuracies improve from $NB = 1$ to $NB = 2$ for FS greater than 4, but then decline sharply for $NB = 3$, although some configurations with higher FS experience a slight increase in accuracy for $NB = 4$.

These results are expected given the relatively small initial number of images (8,671 images). In the first split, there are 6,936 images; in the second split, 3,468 images; and in the third split, only 867 images. With limited training data, the CNN relies heavily on effective gradient flow to generalize from scarce examples. ReLU effectively resists the vanishing gradient problem, maintaining consistent and reasonable accuracies even with only 10% of the training data, whereas Sigmoid performs significantly worse because it is susceptible to the vanishing gradient problem.

5 Discussion

Our work work explores the influence of hyperparameter configurations on CNN performance in image classification tasks with numerous classes and low-resolution images. Increasing the number of convolutional blocks (NB) from one to two enhanced accuracy by enabling the network to learn more abstract features (Section 4.1). However, increasing NB to four decreased performance due to excessive reduction of spatial dimensions, given the small input image size of 28×28 pixels, indicating how important it is to align network depth with input data characteristics.

The initial filter size (FS) was also major factor; larger FS allowed the CNN to capture a broader range of features and facilitated filter specialization for distinct classes. Models with higher FS consistently outperformed those with smaller FS, achieving the highest average accuracy of 74.81% with 3 NB and an initial FS of 64. Additionally, our comparison of activation functions revealed that ReLU outperformed Sigmoid across all configurations (Section 4.2). ReLU's resistance to the vanishing gradient problem enabled more effective training, especially in deeper networks and with reduced training data, whereas Sigmoid's susceptibility hindered learning and led to lower accuracy.

The proportion of training data significantly affected model performance (Section 4.3). Models trained on larger datasets (80% training data) achieved higher and more consistent accuracies. Reducing the training data to 40% and then to 10% caused noticeable declines in accuracy for both activation functions, with a more pronounced effect on Sigmoid. As is well known, this shows the necessity of sufficient training data for effective generalization, and remarks ReLU's robustness in low-data scenarios.

Our results were overall satisfactory, but future work should study the specific classifications per each dataset, too see if any are particularly miss-classified. Moreover, other techniques should be explored, such as the integration of other activation functions such as Leaky ReLU or ELU, and advanced architectural strategies like residual connections. Additionally, techniques like data augmentation or transfer learning could help overcome limitations posed by small training datasets, allowing the experimentation with even deeper architectures

6 Conclusions

In this work, we systematically analyzed the effects of key hyperparameters on CNN performance for image classification tasks. Increasing the number of convolutional blocks enhanced feature abstraction up to a point, beyond which performance degraded due to over-compression of spatial dimensions. Larger initial filter sizes improved accuracy by allowing filters to specialize in class-specific features, which is important for datasets with many classes. ReLU consistently outperformed Sigmoid as the activation function, mitigating the vanishing gradient problem and proving robust even with limited training data. The proportion of data allocated to training significantly affected performance, underscoring the need for sufficient training data to achieve high accuracy.

Bibliography

[1] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "A Comprehensive Survey and Performance Analysis of Activation Functions in Deep Learning". In: *CoRR* abs/2109.14545 (2021). arXiv: 2109.14545. URL: https://arxiv.org/abs/2109.14545.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995).

[5] Benjamin M Marlin et al. "Inductive principles for restricted Boltzmann machine learning". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 509–516.

[6] M Mesran et al. "Investigating the Impact of ReLU and Sigmoid Activation Functions on Animal Classification Using CNN Models". In: *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 8 (Feb. 2024), pp. 111–118. DOI: 10.29207/resti.v8i1.5367.

[7] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on Machine Learning*. 2010, pp. 807–814.

[8] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[9] Kihyuk Sohn et al. "Learning structured output representation using deep conditional generative models". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3483–3491.

[10] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.