COMPUTATIONAL COMPLEXITY OF RESNET50:

A STUDY OF SCRATCH VS PRE-TRAINED MODELS ON SYNTHETIC PROBABILITY DENSITY FUNCTIONS PLOTS IMAGES

CONTENT

1. Introduction

This project aims to carry out an image classification task to explore how a neural network built from scratch and a pre-trained one differ in computational complexity and performance based on the amount of information available. Both neural networks are based on the ResNet50 architecture [1].

The work follows the main steps required to accomplish this task. First, it describes the creation of a new dataset entirely from scratch, including both the data structure and image generation. Then, it performs an analysis of this dataset and the transformations necessary to make it usable by the neural networks. The network trained from scratch is implemented without any prior knowledge, followed by the implementation of the pre-trained one.

Both models are trained on the same image data with baseline hyperparameters to obtain reference models and a preliminary understanding of their expected performance. The next phase focuses on hyperparameter fine-tuning to find the best combinations for both models, which are then used for the final evaluation.

The final step compares the computational complexity of both models using the best configurations from fine-tuning. The models are tested on datasets of different sizes to draw conclusions about efficiency and effectiveness.

The work is presented through both theoretical and practical analysis, explaining the motivations behind the choices made at each step. Graphs and plots are used to visualize and support the results.

The objective of the image classification task is to classify images representing probability density functions (PDFs), allowing the model to learn the shapes of these functions and identify which probability distribution each belongs to.

2. Methods

Dataset

One of the first challenges was creating a dataset from scratch. During the search for an existing dataset that matched the goal of this study, no suitable results were found, thus making it necessary to build a new one.
Since retrieving real images of probability distributions would be highly demanding in terms of time, energy, and copyright, the dataset was instead generated by sampling random observations from selected distributions and plotting their corresponding density functions.

A probability density function (PDF) describes the probability distribution of a continuous random variable. It is defined by a function *f(x,P)*, where *x* represents observations belonging to the support space, and *P* is a set of one or more parameters belonging to the PDF, each within a given parameter space.

The following distributions were chosen: *Beta, Chi-squared, Exponential, Gamma, Laplace, Normal, Uniform,* and *Weibull*.

Some distributions, such as the *Exponential* and *Gamma*, can be quite similar depending on parameter values. Therefore, parameters were carefully selected and slightly varied to avoid overlaps between PDFs. Parameters were fixed for each PDF so that only the distribution of the observations x varied among images.

Density Distribution	*Parameters*	*Parameters Space*	*Support Space*
Beta	Alpha Beta	(0, inf) (0, inf)	(0,1)
Chi-squared	Degrees of freedom	(0, inf)	(0, inf)
Exponential	Scale	(0, inf)	(0, inf)
Gamma	Alpha Beta	(0, inf) (0, inf)	(0, inf)
Laplace	Location Scale	(-inf, inf) (0, inf)	(-inf, inf)
Normal	Location Scale	(-inf, inf) (0, inf)	(-inf, inf)
Uniform	a b	(-inf, b) (a, inf)	(a, b)
Weibull	Scale Shape	(0, inf) (0, inf)	(0, inf)

Parameters selection:

- Alpha = 1.5
- Beta = 0.5
- Degrees of freedom (df) = 3
- Scale = 0.5

- Shape = 1
- Location = 0
- a = 0
- b = 1

The dataset structure was designed to be flexible for future extensions. Each image has a unique name, and metadata stored in a *.json* file use the image name as a key, with attributes including distribution name, parameter space, parameter values, support space, and corresponding values.

Data Structure :

{
image_name_0 :
{ 'label' : distribution name,
'parameters' : { parameter 1 : parameter space,
parameter 2 : …, …},
'values_parameters': [p1, p2, …, pr],
'support' : support space,
'values_support' : (x1, x2, …, xn) },

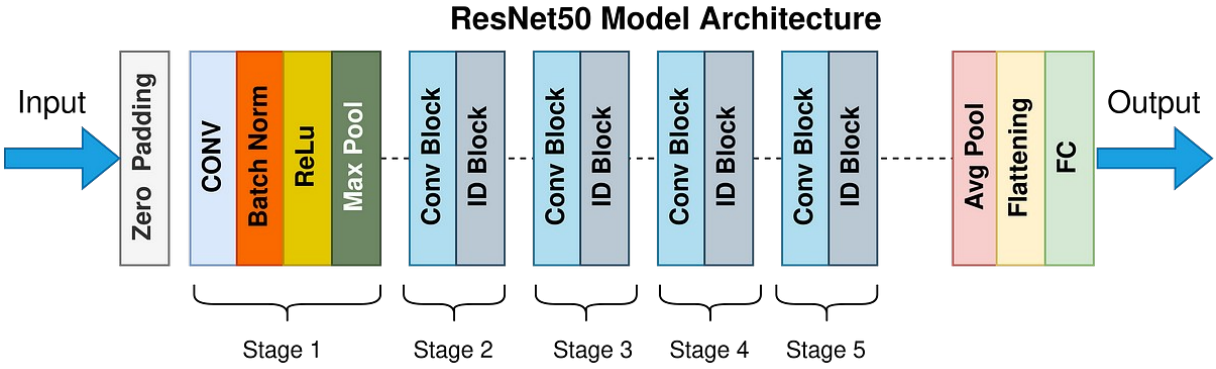
image_name_1 : {
…
},
…
}

Additionally, the dataset is balanced across all categories of PDFs.

Neural Network Architecture

For this project, the ResNet50 neural network was chosen for the image classification task, both for the model trained from scratch and the pre-trained one.

This choice relies on ResNet50's ability to efficiently handle complex image classification problems. The architecture overcomes the degradation problem typical of deep networks, allowing models with many layers to be trained without suffering from vanishing gradients. ResNet50 combines simplicity and efficiency, serving as a solid benchmark for image classification. Its residual connections help the network learn identity mappings, improving generalization even for images very different from natural ones. Thus, this architecture offers a strong balance between depth, computational cost, and performance, ideal for comparing a scratch and a pre-trained model.



ResNet50 consists of 50 layers, including convolutional, pooling, and fully connected layers. Its key feature is the use of residual (skip) connections, where the input of a layer is added directly to the output of a deeper layer. This design mitigates issues like vanishing gradients and performance saturation. The architecture is organized into residual blocks, each containing convolution, batch normalization, and ReLU activation layers. These blocks allow the network to learn both simple and complex patterns, maintaining training stability and achieving high classification accuracy.

The scratch model implementation was inspired by Aladdin Persson's tutorial, with minor adaptations for this project. Only small changes were introduced to maintain fidelity to the original architecture and preserve computational complexity [2].

Scratch ResNet50 vs. Pre-trained ResNet50

The goal is to compare two identical network architectures, one with no prior knowledge and one pre-trained on general images. This comparison helps explore how prior knowledge influences learning performance and computational cost.

The pre-trained ResNet50 was originally trained on real-world images, such as objects, animals, and environments. The question arises whether that prior knowledge is useful for this task, where the images are synthetic, representing only lines and axes on a white background.

On one hand, the difference in visual content between these real-world images and the synthetic ones might limit the usefulness of pre-training. This could suggest that, for certain specialized tasks, the architecture itself may matter more than the amount of prior information.
On the other hand, pre-trained models often generalize well even in different domains, which supports the idea of transfer learning, where the training data need not exactly match the target task.

Given these hypotheses, dataset size becomes a crucial factor. A dataset that is too small may not provide enough information for the scratch model to learn effectively. Thus, after training baseline models and fine-tuning hyperparameters, both best-performing models are tested on datasets of different sizes (100, 1,000, and 10,000 images) to analyze their learning behavior and accuracy.

Computational Complexity

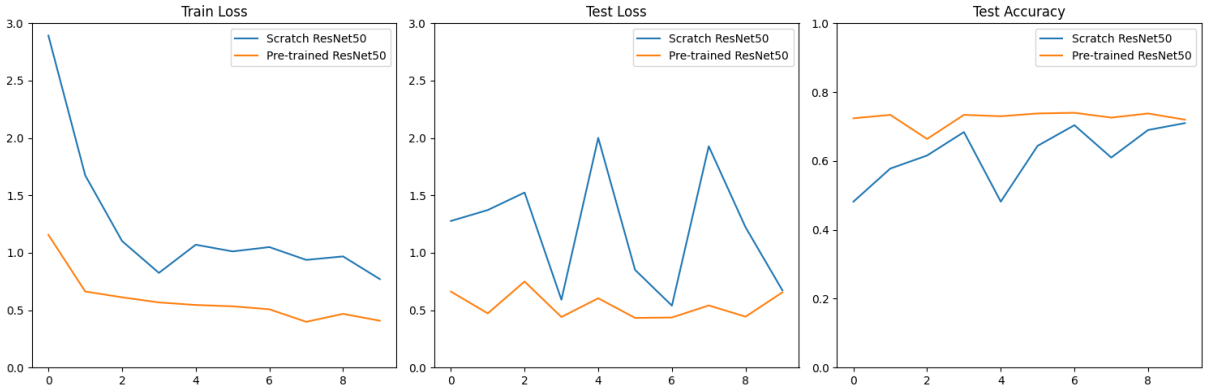
To assess computational complexity, several metrics were used. In this project, computational complexity refers to a combination of performance, training time, and the number of Multiply-Accumulate Operations (MACs).
Accuracy provides a general measure of the model's classification capability. Training time (measured in seconds) tracks how long it takes to train and test models.
MACs quantify the total number of multiplications and additions performed during computations, adjusted with the number of epochs and batch size used. This because both models share the same architecture, their MAC counts are inherently identical. However, differences in batch size and number of epochs affect total computational load during fine-tuning.

3. Results

Baseline Models

For baseline training, a dataset of 500 images was used, split into training, validation, and test sets with an 80:10:10 ratio. The batch size was 4, and training ran for 10 epochs. Each image contained three color channels and a resolution of 256×256 pixels.

During training, the learning curve showed that the scratch model improved steeply in the first 4–5 epochs, then stabilized. The pre-trained model improved smoothly and maintained steady progress. Both reached around 72% accuracy after 10 epochs, but the scratch model started at a lower accuracy (~50%), unlike the pre-trained model that already had an accuracy around 72%, showing that it could classify reasonably well from the outset.

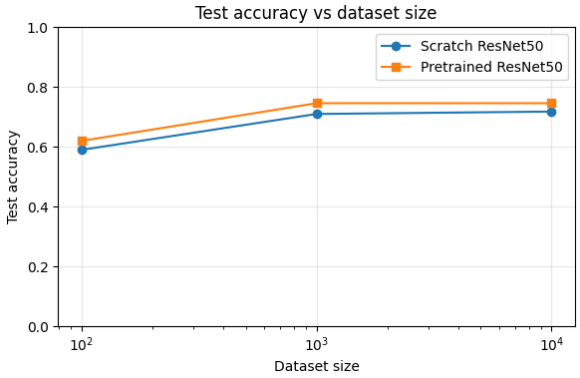


Hyperparameter Fine-tuning

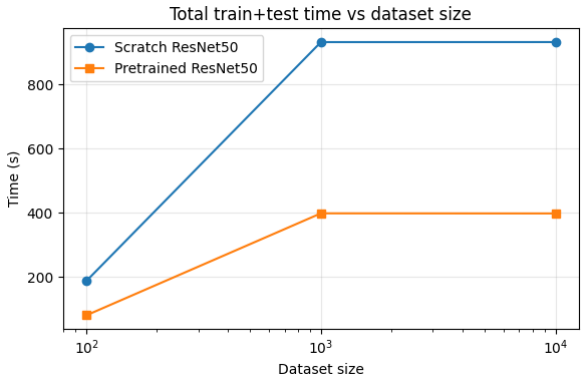
Fine-tuning tested combinations of 10, 20, and 50 epochs, with batch sizes of 4, 8, and 16.
Results showed that the scratch model performed best with batch size 4 and 20 epochs, while the pre-trained model performed best with batch size 4 and 10 epochs.

Computational Complexity and Data Size

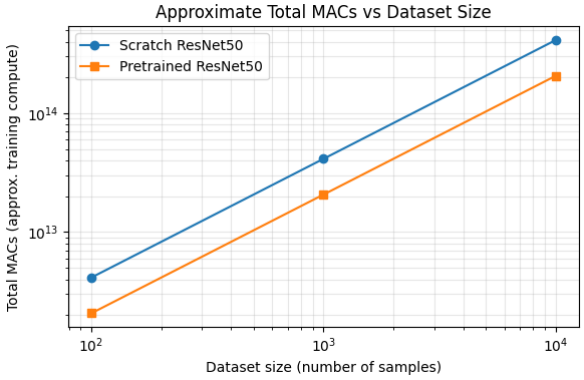
The best models were then compared across different dataset sizes.
Accuracy results showed that varying dataset size did not produce large differences in performance. Overall, the pre-trained model consistently performed better, both for small and big dataset. Accuracy generally ranged between 60% and 75%, with the most visible gain occurring when increasing from 100 to 1,000 images.



Training and testing time, however, varied substantially. For 100 images, the pre-trained model required about 80 seconds, compared to 186 seconds for the scratch model. For datasets of 1,000 and 10,000 images, times stabilized around 400 seconds for the pre-trained model and 930 seconds for the scratch model.



The combined metric involving MACs, epochs, and batch size showed an expected increase with dataset size. Since MACs depend only on architecture, it was consistent for both models, but scaling factors (epochs and batches) revealed higher cost for the scratch model, which required more training iterations.



A final observation from fine-tuning revealed that different hyperparameter settings did not always produce the same optimal results, as model performance varied within relatively small margins.

4. Conclusions

This project presented a complete workflow for an image classification task focused on computational complexity. It covered dataset creation, model implementation, and evaluation of two ResNet50 models, one trained from scratch and one pre-trained.

Both models performed well in classifying probability density functions, improving through fine-tuning and showing consistent behavior across different dataset sizes. Several insights emerged from this work.

First, considering the dataset composition, images were simple and clean (no background noise), making it easier for the models to detect patterns. Furthermore, each distribution used fixed parameters with variable samples, helping models learn consistent visual features.

Both models demonstrated strong classification capabilities due to the robustness of the ResNet50 architecture and the well-structured dataset. Performance did not improve significantly with larger datasets or hyperparameter changes, suggesting a plateau in accuracy. More attention could therefore be given to computational efficiency, such as training time, number of operations, and energy consumption.

It should also be noted that the pre-trained model required far more data and computational resources during its original training, which were not included in the present cost analysis.

Further analysis could investigate which specific distributions were misclassified and whether certain overlapping patterns explain this behavior.

Ultimately, it is challenging to declare a single “best” model. In real-world scenarios with limited computational resources, the scratch model might be sufficient, given the small performance gap. Although the pre-trained model consistently achieved higher accuracy, the effort and resources required for pre-training are important factors to consider.

Further Directions

Future work could focus on expanding the dataset with more realistic images and varying parameters for the PDFs. Another possible direction would be to define a performance threshold and attempt to reach it with the minimum computational cost. Finally, refining the fine-tuning process with a broader range of hyperparameters could yield deeper insights into the behavior of the ResNet50 architecture.

5. References

[1] https://doi.org/10.48550/arXiv.1512.03385

[2] https://www.youtube.com/watch?v=DkNIBBBvcPs