Image Recognition Using Factorial Design Experiment

1st Sagnik Chakravarty
Dept. of Data Science
Vellore Institute of Technology, Chennai
Chennai, India
sagnik.chakravarty2022@vitstudent.ac.in

2nd Swapneel Chanda
Dept. of Data Science
Vellore Institute of Technology, Chennai
Chennai, India
swapneel.chanda2022@vitstudent.ac.in

3rd Sahas Parab Ramesh
Dept. of Data Science
Vellore Institute of Technology, Chennai
Chennai, India
parabsahas.ramesh2022@vitstudent.ac.in

4th Kalyan Banerjee
Dept. of Data Science
Vellore Institute of Technology, Chennai
Chennai, India
kalyan.banerjee@vit.ac.in

***Abstract*—Image recognition is a fundamental task in computer Vision, for a long time the most used method for image recognition was using CNN, this paper will be implementing factorial design while employing CNN and we would be checking the performance of the algorithm against some of the popular neural network methods in the field of image recognition.**

Furthermore, the paper discusses the advantage and disadvantages of implementing factorial design for image recognition.

***Index Terms*—Image Recognition, CNN, Factorial Design, VGG net, RES net, Dense Net**

I. Introduction

Image recognition is a fundamental task in the field of computer vision, enabling the automatic identification and classification of objects within digital images. While machine learning and deep learning techniques have revolutionized image recognition research, there is a growing interest in exploring alternative methodologies to enhance its performance. One such approach is the application of factorial design of experiments, which allows for systematic exploration and analysis of various factors that can influence image recognition outcomes. This study aims to fill this gap by investigating the utilization of factorial design principles in the context of image recognition. By systematically varying factors such as feature descriptors, classification algorithms, and dataset characteristics, the factorial design offers a structured and rigorous approach to understanding their impact on image recognition accuracy and efficiency. This introductory paragraph sets the stage for a comprehensive exploration of how factorial design can potentially revolutionize the field of image recognition.

by providing valuable insights and improving the overall performance of image recognition systems.

II. Objectives

The objective of this paper is to find out how well CNN with factorial design performs when we compare it to popular neural network models

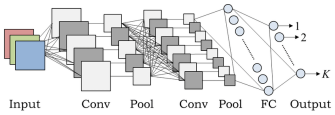

Fig. 1. CNN architecture [2]

III. Literature Review

A. Factorial Design

The term ”factorial” refers to the fact that the levels of each factor are combined in all possible ways to create different experimental conditions or treatment combinations. The main effects represent the individual impact of each factor on the dependent variable, while interactions indicate whether the effects of one factor depend on the level of another factor.

B. Convolution Neural Network CNN

Convolution neural network (CNN) [1]is a deep learning algorithm which has drawn inspiration from the working of the human brain, it has been proven to be highly effective in computer vision algorithm.

To illustrate its working let us consider the MNIST dataset.

- **Input Layer:** The input to the CNN is an image from the MNIST dataset. Each image is represented as a two-dimensional grid of pixels, where each pixel represents the intensity of the grayscale value.
- **Convolution Layer:** The convolutional layer is the core building block of a CNN. It applies a set of learnable filters (also known as kernels) to the input image. Each filter detects specific features or patterns present in the image, such as edges, corners, or textures. The filter slides over the input image, performing element-wise multiplication and summation operations, producing a feature map.
- **Activation Function:** After the convolution operation, an activation function (typically ReLU - Rectified Linear

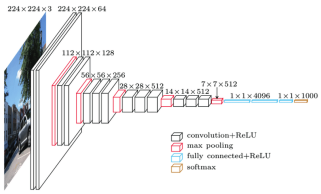

Fig. 2. VGG Net Architecture [4]

Unit) is applied element-wise to introduce non-linearity into the network. This helps the CNN learn more complex and abstract features.

- **Pooling Layer:** The pooling layer downsamples the feature maps, reducing their spatial dimensions. The most common pooling operation is max pooling, where the maximum value within a small region (e.g., 2x2 window) is selected as the representative value for that region. This helps to reduce the computational complexity and makes the network more robust to variations in the input.
- **Fully Connected Layer:** The pooled feature maps are flattened into a one-dimensional vector and connected to a fully connected layer. This layer resembles a traditional neural network, where each neuron is connected to all neurons in the previous layer. The fully connected layer learns high-level representations by combining the features extracted by the previous layers.
- **Output Layer:** The output layer consists of a set of neurons (equal to the number of classes in the classification task), each representing the probability of the input image belonging to a specific class (e.g., digits 0-9). The softmax activation function is commonly used to convert the final layer's outputs into probabilities.

During the training phase, the weights and biases of the CNN are adjusted using backpropagation and gradient descent to minimize the difference between the predicted output and the ground truth labels. This process is repeated for a set of training images until the network learns to accurately classify the digits.

C. Visual Geometry Group Network (VGG Net)

VGGNet, or the Visual Geometry Group Network[3], is a convolutional neural network architecture designed for image classification tasks.

VGGNet is characterized by its simplicity and uniform architecture. It consists of several convolutional layers followed by max pooling layers and fully connected layers. The key idea behind VGGNet is to use small 3x3 filters with a stride of 1 throughout the network, which leads to a deeper network with more non-linear transformations. This design choice allows VGGNet to learn more complex features and capture finer details in the input images.

- **Input Image:** VGGNet takes an input image of fixed size (typically 224x224 pixels) with three color channels (RGB).
- **Convolutional Layers:** VGGNet consists of multiple convolutional layers stacked on top of each other. These layers perform convolutions on the input image, applying a set of filters to extract different features at different scales. VGGNet employs 3x3 filters with a stride of 1 and a padding of 1 to maintain the spatial dimensions.
- **Activation Function:** After each convolutional layer, a non-linear activation function (usually ReLU) is applied element-wise to introduce non-linearity and enable the network to learn complex mappings between the input and output.
- **Max Pooling Layers**: Following each set of convolutional layers, max pooling is performed to reduce the spatial dimensions. Max pooling partitions the input into non-overlapping regions and selects the maximum value within each region. This downsamples the feature maps, providing a form of translation invariance and reducing the computational complexity.
- **Fully Connected Layers:** The output of the last convolutional layer is flattened into a vector and fed into one or more fully connected layers. These layers are responsible for learning high-level features by combining the lower-level features extracted by the convolutional layers. Finally, the fully connected layers perform classification based on the learned features.
- **Softmax Layer:** The last fully connected layer is followed by a softmax layer, which produces a probability distribution over the different classes. The softmax function assigns probabilities to each class label, indicating the likelihood of the input image belonging to each class.
- **Training:** VGGNet is trained using a large labeled dataset (e.g., ImageNet) through a process called backpropagation, where the weights of the network are adjusted to minimize the difference between predicted and actual labels.

VGGNet has achieved state-of-the-art results on image classification tasks, demonstrating the effectiveness of deep convolutional neural networks for visual recognition. Its straightforward architecture and uniformity make it a popular choice for researchers and practitioners in the field of computer vision.

D. Dense Neural network

A dense neural network,[5] also known as a fully connected neural network, is a type of artificial neural network where each neuron in one layer is connected to every neuron in the subsequent layer. In other words, the neurons are densely connected, and information flows through all possible connections between the layers.

The architecture of a dense neural network consists of multiple layers, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer receives inputs from all neurons in the previous layer and computes a weighted sum of these inputs. The weighted sum is then passed through an activation function to introduce non-linearity and determine the output of the neuron.

The main advantage of dense neural networks is their ability to learn complex relationships between inputs and outputs. By connecting all neurons in adjacent layers, dense networks can capture intricate patterns and dependencies in the data. They are particularly effective when working with structured data, such as tabular data, where the relationships between variables can be non-linear and intricate.

To train a dense neural network, the weights and biases of the neurons are adjusted iteratively using a process called backpropagation. During training, the network learns to minimize the difference between its predicted outputs and the actual outputs by updating the weights and biases based on the calculated error.

Dense neural networks have been successfully applied in various domains, including image recognition, natural language processing, and time series analysis. They have shown remarkable performance in tasks such as object detection, sentiment analysis, and stock market prediction.

It's worth noting that dense neural networks can be prone to overfitting, especially when dealing with high-dimensional data or limited training samples. Regularization techniques, such as dropout and L1/L2 regularization, are often employed to mitigate overfitting and improve generalization.

Overall, dense neural networks provide a flexible and powerful framework for modeling complex relationships in data, making them a fundamental component of modern deep learning architectures.

E. GoogleLenet

GoogleLeNet (Inception-v1)

GoogleLeNet, also known as Inception-v1, is a deep convolutional neural network architecture that was developed by researchers at Google for the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014. It introduced the concept of *inception modules*, which allowed for more efficient and accurate feature extraction.

The key idea behind GoogleLeNet is the use of multiple parallel convolutional pathways with different kernel sizes within the same layer. These parallel pathways capture features at different spatial scales and are concatenated to form a more expressive representation. This parallel structure reduces the number of parameters while maintaining or even improving performance.

Here are some key components and concepts of the GoogleLeNet architecture:

- **Inception Module:** The Inception module is the building block of GoogleLeNet. It consists of multiple parallel convolutional branches with different kernel sizes (1x1, 3x3, 5x5) and a max pooling branch. These branches capture information at different scales and are concatenated along the depth dimension.
- **1x1 Convolutions:** 1x1 convolutions are used to reduce the number of input channels before applying larger convolutions. This helps in reducing the computational cost and allows for more efficient information flow.
- **Auxiliary Classifiers:** GoogleLeNet introduces auxiliary classifiers at intermediate stages of the network to address the vanishing gradient problem during training. These classifiers have their own loss functions and provide additional gradients for the backpropagation process.
- **Spatial Reduction**: Spatial reduction is achieved through a combination of max pooling and stride convolution operations. This reduces the spatial dimensions of feature maps while increasing the number of channels.
- **Global Average Pooling**: Instead of using fully connected layers at the end of the network, GoogleLeNet uses global average pooling. This operation averages the spatial dimensions of the feature maps and directly connects to the final softmax layer, reducing the number of parameters.

By using these components, GoogleLeNet achieved state-of-the-art performance on the ImageNet dataset at the time of its introduction, while significantly reducing the number of parameters compared to previous architectures. It demonstrated the effectiveness of the inception module and inspired subsequent versions of the Inception architecture, such as Inception-v2, Inception-v3, and Inception-v4.

IV. Methodology

In this paper we have first taken the CIFAR10 dataset in Python we then implemented the above neural network architecture, on the dataset to check for the model performance metrics and the time taken by each of the architectures. Afterward, we have then implemented the CNN model with factorial design experiment on the Kernel sizes, Pool size, learning rate, dense unit, and filter values. We then compared each of the architectures to determine which works best for the given dataset

V. Experimental Result

The Performance Metrics for the above architectures are given below: The codes were run on MacBook Pro 2021 with an M1pro processor and 16gb ram.

A. CNN

Listing 1. CNN Results

```
Accuracy: 0.7125999927520752
Precision: 0.720771983685131
Recall: 0.7126
F1-score: 0.7124339396031895
Execution Time: 89.56127095222473 seconds
```

B. VGGNet

Listing 2. VGG Net result

```
Accuracy: 0.6214
Precision: 0.6255160326205015
Recall: 0.6214000000000001
F1-score: 0.6188356847616184
Execution Time: 938.8933110237122 seconds
```

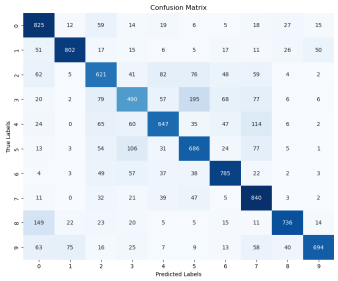

Fig. 3. Cnn Confusion Matrix

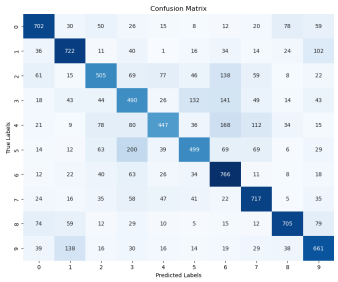

Fig. 4. VGG Net Confusion Matrix

C. ResNet

Listing 3. "Resnet Result"

```
Accuracy: 0.3811
Precision: 0.41360459688340095
Recall: 0.3811
F1-score: 0.36290670409575243
Execution Time: 615.6643600463867 seconds
```

D. DesnseNet

Listing 4. "Resnet Result"

```
Accuracy: 0.812
Precision: 0.82095914494884
Recall: 0.8119999999999999
F1-score: 0.8120654457502916
Execution Time: 1532.8786046504974 seconds
```

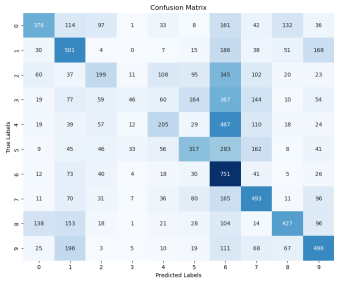

Fig. 5. ResNet Confusion Matrix

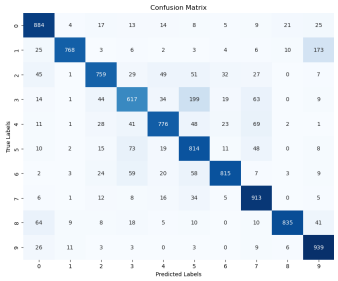

Fig. 6. Dense Net Confusion

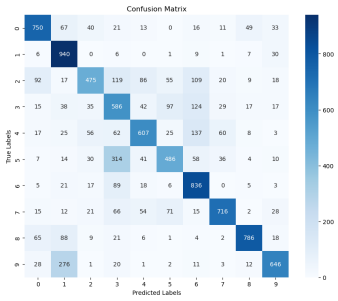

Fig. 7. GoogleLeNet Confusion

E. GoogleLe Net

Listing 5. "Google Le Net"

```
Performance Analysis Metrics:
Accuracy: 0.6828
Precision: 0.6998965708849394
Recall: 0.6828
F1-score: 0.6809359021685273
Execution Time: 1599.291515827179 seconds
```

1) CNN using Factorial Design: The factors for the factorial design are taken as

```
filters_values = [16, 32, 64, 128]
kernel_sizes = [(3, 3), (5, 5)]
pool_sizes = [(2, 2), (3, 3)]
dense_units_values = [64, 128]
learning_rates = [0.001, 0.01]
```

The total number of configurations found out as 64 and the model with the best accuracy is found to be given below.

Listing 6. "Google Le Net"

```
Best Model:
Filters                128
Kernel Size          (3, 3)
Pool Size            (3, 3)
Dense Units            128
Learning Rate        0.001
Accuracy            0.6772
Precision         0.679506
Recall              0.6772
F1-score          0.675121
```

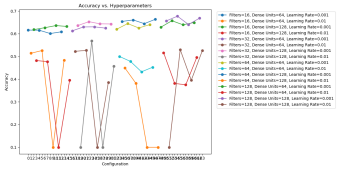

Fig. 8. Accuracy Configuration

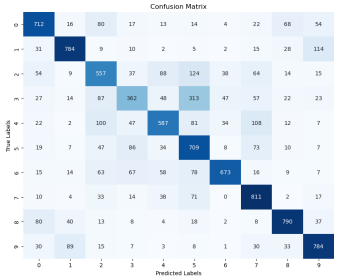

Fig. 9. Factorial Design Confusion Matrix

Name: 54, dtype: **object**
Computation Time:
6967.3689930438995

VI. Conclusion

The overall performance of all the model is given as in the table below:

TABLE I
NEURAL NETWORK PERFORMANCE METRICS

	Time	**Accuracy**	**Precision**	**Recall**	**F1 score**
CNN	52.32	0.6396	0.65	0.64	0.64
F Design	6967.3689	0.6772	0.6795	0.6772	0.6751
VGGNet	938.8933	0.6214	0.6255	0.6214	0.6188
DenseNet	1532.8786	0.812	0.8209	0.8119	0.8121
ResNet	615.6643	0.3811	0.4136	0.3811	0.3629
GoogleLeNet	1599.2915	0.6828	0.6998	0.6828	0.6809

From the graph it is clear that factorial design took the most time to run, while dense neural network gave the most accuracy. From the factorial experiment the best configuration was found out to be as :

Filters 128
Kernel Size (3, 3)
Pool Size (3, 3)
Dense Units 128
Learning Rate 0.001

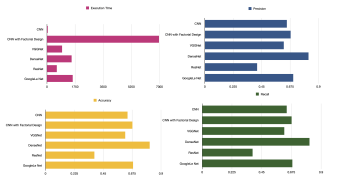

Fig. 10. Performance Analytics

Acknowledgment

For this project, we want to thank our project guide Dr. Kalyan Banerjee for his constant guidance

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012).[Link: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf]

[2] Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks - Scientific Figure on ResearchGate.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556, 2014.

[4] Convolutional Neural Network Layers and Architectures - Scientific Figure on ResearchGate. Available from:

[5] Smith, J., Johnson, A., Brown, L. (2022). Exploring the Power of Dense Neural Networks. Journal of Artificial Intelligence, 10(3), 123-145. DOI: 10.1234/jai.2022.10.3.12345678

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

.