Nature Inspired CNN Topology Optimization

# **TEAM**

|  |  |
| --- | --- |
| **Name** | **Roll Number** |
| Mohsin Ashraf | 15140094 |
| Mushood Hanif | 14140062 |
| Mehwish Hameed | 15140107 |
| Rida Yaqoob | 14140044 |

Contents

[**TEAM** 1](#_Toc5110088)

[**Problem Statement** 3](#_Toc5110089)

[**Introduction** 3](#_Toc5110090)

[**Procedure** 3](#_Toc5110091)

[**References** 4](#_Toc5110092)

[**Jargon** 4](#_Toc5110093)

# **Problem Statement**

The **CNN Framework** has been proven many a times for its prowess and degree of performance on image datasets [1]. However, CNNs are considered as **deep learning networks** and require heavy computation, also, to find the optimal **topology** to initialize CNNs for any dataset still remains a problem. There have been contributions in this field to generate optimal CNNs for a dataset by training a large population of CNNs over multiple **distributed systems** to find the optimal CNN architecture amongst the population [2]. There also have been ways to generate a population of CNNs that traverse to the solution based on a **fitness function** which is used to determine the optimal CNN architecture [3].

# **Introduction**

We propose a novel methodology to optimize CNNs using **evolutionary algorithms** to search for the best architecture to initialize a CNN for any given dataset. The parameters that need to be optimized can be controlled by the user and the user will also be provided with a set of benchmark datasets that can be used to compare the performance of our CNN with existing state-of-the art CNNs.

## **Procedure**

* We define the CNN framework as a mathematical model i.e. as a network function N (w, r) = O(w) where w is the weight vector for a neural layer and r is the input of that neural layer and O(w) is the output vector provided by that layer [4].
* We define the search space of our evolutionary algorithm as an n-dimensional graph where each point represents an n-dimensional vector that contains the parameters that the user wants to be optimized for a dataset. The “n” is user-defined and can be manipulated based on the user’s will.
* We define a **master-slave distributed architecture** for training and evaluating our CNNs to introduce parallelization to make the evaluation process faster and more efficient.
* The objective function of our evolutionary algorithm is complex and requires heavy computation. Therefore, we would use **dynamic programming** to avoid repeated calculation of the objective function.
* Each session of the slave systems will be logged to observe and keep track of the performance and progress.

# **References**

1 - ImageNet Classification with Deep ConvolutionalNeural Networks – Hinton, University of Toronto

2 – Large Scale Evolution of Convolutional Neural Networks Using Volunteer Computing – Travis Desell

3 - Evolving Deep Convolutional Neural Networks by Variable-length Particle Swarm Optimization for Image Classification – Bin Wang, Victoria University of Wellington

4 - Efficient parallel learning algorithms for neural networks – Alan Kramer, U.C. Berkeley

# **Jargon**

|  |  |
| --- | --- |
| **Word** | **Meaning** |
| CNN Framework | Convolutional Neural Networks |
| Deep learning networks | Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. |
| Topology | The elements that make up a CNN framework. |
| Distributed systems | A distributed system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another. |
| Fitness function | A fitness function is a particular type of objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims. |
| Evolutionary algorithms | In artificial intelligence, an evolutionary algorithm is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. |
| Master-slave distributed systems | Master/slave (technology) Master/slave is a model of communication where one device or process has unidirectional control over one or more other devices. |
| Dynamic programming | Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions using a memory-based data structure. |