ezCGP

Block-Based Convolutional Neural Network Evolution using Cartesian Genetic Programming

ABSTRACT

Notes to add:

1. There are fields without domain experts that is difficult to craft manually, hence search with AutoML
2. Low/Mid/High-level feature capture (e.g. multiple independent preprocessing blocks)
3. Optimizer and activation mutation

1 INTRODUCTION

1.1 Machine Learning Pipeline

In modern technology, we use machine learning (ML) algorithms in many of our systems, including, but not limited to, search engines, facial recognition, fraud detection, voice assistants, and self-driving vehicles. In order to build effective ML systems, we have traditionally relied on the collaboration of ML engineers and domain experts to craft individual systems tailored to each task. This is often an intensive and manual process, which can be roughly categorized into three steps:

* + 1. *. Data Preprocessing and Augmentation*

ML engineers typically start by collaborating with domain experts to preprocess a dataset and extract data features, which serve as important markers in a dataset. As data points are often complex, these features often serve as important anchors around which a dataset can be simplified. However, by simplifying individual data points around these features, we often gain higher accuracy at the cost of generalizability, as the model learns from data with a smaller range of diversity. We often counter this information loss by applying transformations to augment our dataset with virtual data before preprocessing, which widens our training data by generalizing the patterns underlying each data point.

*1.1.2. Model Creation and Training*

Afterward, engineers generally continue by creating simple models that serve as introductory hypotheses to a particular problem. These models usually originate from an engineer’s intuition and past experience with the models’ components and hyperparameters. For example, in a convolutional neural network (CNN), an experienced engineer might understand that pooling layers often come before convolution layers, as a pooling layer shrinks the input image and highlights the shape of the object by reducing noise. The neural network is then fed a portion of the dataset to adjust, or train, each pair of layers’ weights and biases through back propagation [1].

* + 1. *Model Validation and Improvement*

After the models have been trained, engineers then feed an unseen portion of the dataset to the model and measure how much of the data it is able to predict. Based on these measurements, we can adjust the model structure and hyperparameters. For example, in our CNN’s case, we might change the pooling layer’s kernel size or add an additional convolution layer with a different stride. By undergoing several iterations of this train-test procedure, the models can steadily improve as their creators experiment with, for example, various permutations of common CNN layer types and hyperparameters.

1.2 AutoML

In recent years, there has been a growing interest in automating the latter steps of this iterative process, which has resulted in a developing field of research called automatic machine learning (AutoML). The primary goal for the field is to develop systems that can create and improve ML systems with minimal human intervention. However, we may also discover new combinations intuitions by exploring and analyzing these generated ML models. As a result, there has been a focus on organic evolution in order to intelligently explore options outside of our current intuition.

*1.2.1. Genetic Programming*

Inspired by biological evolutionary theory, genetic programming centers around randomly creating individuals (i.e. models) and gradually improving them over time by applying probabilistic genetic operators over several generations. These genetic operators generally fall under two primary classes: mutation and crossover. Mutation operators generally aim to apply one or more random changes to specific parts of an individual’s genome, independent of other individuals in the population. On the other hand, crossover operators generally aim to spread genetic information across the population by exchanging two individuals’ genetic information and creating offspring. After each generation, every individual is evaluated, given a fitness score, and a subset of the fittest individuals are selected for the following generation.

*1.2.2. Cartesian Genetic Programming*

One of the primary downsides of genetic programming lies in its dependency on a tree structure for its genetic information. For each primitive genetic structure (e.g. convolutional layer) or primitive,

1.3 Related Work

Since Alex Krizhevsky and his team won the ImageNet challenge in 2012, convolutional neural networks have become widely popular due to their high performance on complex supervised learning tasks, such as image recognition [2]. Given their popularity, it is no surprise that there have been several attempts at automating their creation.

*1.3.1. Evolutionary Convolutional Neural Networks*

The idea of optimizing hyperparameters and neural network combinations has been explored before. Two notable inspirations for this paper used traditional Cartesian genetic programming (CGP) in CGP-CNN and dynamic structured grammatical evolution (DSGE) with DENSER. In CGP-CNN, the user specifies a set of layers and the framework attempts to optimize their arrangements and hyperparameters using a point mutation operator [3]. Meanwhile, in DENSER, the framework improves its evolution by encoding layer type and hyperparameters in separate grammatical levels. Furthermore, it employs a more complex set of mutation and crossover operators to deepen the search space [4]. Due to their evolutionary processes, both frameworks have produced effective CNN structures.

*1.3.2. Expanded Pipeline Evolution*

However, while both CGP-CNN and DENSER have evolved effective CNNs, they have maintained a static data augmentation structure using padding, horizontal flips, and random crops [3, 4]. Furthermore, both frameworks evolve from a relatively small set of CNN layers, with only partial crossover for DENSER. We address these issues in the ezCGP framework by introducing a block structure, where each block represents a step in the ML pipeline. In doing so, we automate the entire ML pipeline’s evolution by organizing and evolving sets of blocks. Furthermore, we explore far wider search spaces by evolving from larger sets of augmentation, preprocessing, and CNN primitives. In addition, we increase the overall population’s deep structural search efficiency by introducing the concept of block-wise crossover, or mating.

3 BLOCK-BASED EVOLUTION

3.1 Genetic Structure

*3.1.1*. *Primitives*

*3.1.2. Genome*

*3.1.3. Block*

*3.1.4. Individual*

F

3.2 Mutation

F

3.3 Mating

O

4 RESULTS

4.1 Experiment Setup

*4.1.1. Multiprocessing CPU*

*4.1.2. Single Process GPU*

*4.1.3. Hardware Restrictions*

4.2 MNIST

A.

4.3 CIFAR-10

A.

4.4 Conclusions

ACKNOWLEDGMENTS

T

REFERENCES (unformatted)

|  |
| --- |
| [1] “Handwritten Digit Recognition with a Back-Propagation Network”, Y. Le Cun et. al. 1990 <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf>  [2] “ImageNet Classification with Deep Convolutional Neural Networks”, Alex Krizhevsky et. al. 2012 <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>  [3] “A Genetic Programming Approach to Designing Convolutional Neural Network Architectures” Masanori Suganuma et. al. 2017 <https://arxiv.org/pdf/1704.00764.pdf>  [4] “DENSER: Deep Evolutionary Network Structured Representation”, Filipe Assuncao et. al. 2018, <https://arxiv.org/pdf/1801.01563.pdf> |