EasyChair Preprint

№ 5070

Quantum Generators: Kernel Regression Model with Machine Learning from the Structured Compute Units

Poondru Prithvinath Reddy

March 1, 2021

Quantum Generators: Kernel Regression Model with Machine Learning from the Structured Compute Units.

Poondru Prithvinath Reddy

ABSTRACT

Quantum Generators is a means of achieving mass food production with short production cycles, and when and where required by means of machines rather than land based farming which has serious limitations. The process for agricultural practices for plant growth in different stages is simulated in a machine with a capacity to produce multiple seeds from one seed input using computational models of multiplication (generating multiple copies of kernel in repetition). In this paper, we present a Kernel Regression Model with Machine Learning for the structured Compute Units resulted by the computational models of multiplication and also train a Kernel Classifier to see if we get better results of reconstruction so that they can be linked to tissues of the kernel which mimic the real cell structure that grows into full-fledged natural tissue. We use simulation to show that we achieve good accuracy with respect to the size of the input space and it is an improvement compared to the logistic regression. The results suggest that it is possible to achieve suitable cell structure for quantum generation.

INTRODUCTION

A **Quantum** (plural quanta) is the minimum amount of any physical entity (physical property) involved in an interaction. On the other hand, **Generators** don't actually create anything instead, they generate quantity prescribed by physical property through multiplication to produce high quality products on a mass scale. The aim of Quantum Generators is to produce multiple seeds from one seed at high seed rate to

produce a particular class of food grains from specific class of **seed** on mass scale by means of machine rather than land farming.

The process for agricultural practices include preparation of soil, seed sowing, watering, adding manure and fertilizers, irrigation and harvesting. However, if we create same conditions as soil germination, special watering, fertilizers addition and plant growth in different stages in a machine with a capacity to produce multiple seeds from one seed input using computational models of multiplication(generating multiple copies of kernel in repetition) then we will be closure to achieving mass food production by means of quantum generators(machine generated) rather than traditional land based farming which has very serious limitations such as large space requirements, uncontrolled contaminants, etc. The development of Quantum Generators requires specialized knowledge in many fields including Cell Biology, Nanotechnology, 3D Cellprinting, Computing, Soil germination and initially they may be big occupying significantly large space and subsequently small enough to be placed on roof-tops.

The Quantum Generators help world meet the food needs of a growing population while simultaneously providing opportunities and revenue streams for farmers. This is crucial in order to grow enough food for growing populations without needing to expand farmland into wetlands, forests, or other important natural ecosystems. The Quantum Generators use significantly less space compared to farmland and also results in increased yield per square foot with short production cycles, reduced cost of cultivation besides easing storage and transportation requirements.

In addition, Quantum Generators Could Eliminate Agricultural Losses arising out of Cyclones, Floods, Insects, Pests, Droughts, Poor Harvest, Soil Contamination, Land Degradation, Wild Animals, Hailstorms, etc.

Quantum generators could be used to produce most important *food* crop *like* rice, wheat and maize on a mass scale and on-demand when and where required.

Computers and Smartphones have become part of our lives and Quantum Generators could also become very much part of our routine due to its potential benefits in enhancing food production and generating food on-demand wherever required by bringing critical advanced technologies into the farmland practices.

3D Bioprinting

3D Bioprinting is a form of additive manufacturing that uses cells and other biocompatible materials known as bioinks, to print living structures layer-by-layer which mimic the behavior of natural living systems. Three dimensional bioprinting is the utilization of 3D printing–like techniques to combine cells, growth factors, and biomaterials to fabricate biomedical parts that maximally imitate natural tissue characteristics.

Bioprinting (also known as **3D bioprinting**) is combination of **3D printing** with biomaterials to replicate parts that imitate natural tissues, bones, and blood vessels in the body. It is mainly used in connection with drug research and most recently as cell scaffolds to help repair damaged ligaments and joints. In this paper, we are looking at natural tissues related to food crops like rice, wheat or maize.

METHODOLOGY and THE ARCHITECTURE

A Quantum Generator device has one or more Compute Units. A work-group executes on a single Compute unit. A Compute Unit is composed of one Processing Element and Seed Object. A Compute Unit may also include filter Units that can be accessed by its processing elements.

A Device is a collection of Compute Units. Quantum Generator device typically corresponds to a collection of multiple Compute Units generated by the seed of a number.

A Seed is a function declared in a program and executed on a quantum generating device. A seed is identified by the Seed Qualifier applied to any function defined in any program.

A Seed Object encapsulates a specific seed function declared in a program and the argument values to be used when executing this Seed Function.

A Synchronization refers to mechanisms that define the order of execution and the visibility of operations between two or more units of execution. The Operations are that define order controls in a program. They play a special role in controlling how operations of in one unit of execution (such as work-items) are made visible to another. Synchronization essentially involves establishing a relation between operations in two different units of execution that define an order control in a device.

Seed Objects

A seed is a function declared in a program. A seed is identified by the seed qualifier applied to any function in a program. A Seed Object encapsulates the specific seed function declared in a program and the argument values to be used when executing this seed function.

Seed Objects are created for any seed functions in program that have the same function definition across all Compute Units for which a program has been built successfully in a device.

Kernel Function

The idea is to use a higher-dimension feature space to make the data almost linearly separable and we use a kernel function in Machine Learning to modify the data without changing to a new feature plan.

There are lots of different kernel techniques available and the simplest is the linear kernel. However, in this paper we use the Gaussian Kernel and TensorFlow has a build in estimator to compute the new feature space. The Gaussian filter function is an approximation of the Gaussian Kernel function and it computes the similarity between the data points in a much higher dimensional space.

Gaussian Kernel with TensorFlow

The objective is to classify the data generated from the structured compute units.

First, we made a linear model and trained a simple linear model in TensorFlow. Then, we substituted the linear model and improved the linear model in TensorFlow by adding explicit kernel methods to the model, retrained it and then evaluated it.

First imported TensorFlow and also other relevant Python packages and libraries. We needed a dataset to work on, so we used the Iris dataset.

Then, we created a Linear –baseline model for implementing linear model in Python with TensorFlow.

Using the Linear-Classifier and training with dataset, we got the output with the decision boundary upon plotting the data.

Next, we will try to beat the linear classifier with a Kernel Classifier and we use the power of explicit kernel with the linear classifier.

As we need to transform the low dimension into a high dimension using a kernel function, we therefore use the Random Fourier which is an approximation of the Gaussian Function and TensorFlow has the function in its library.

Then, we set the high dimension Kernel Function and created the model using L_2 regularization. Once Kernel Classifier is built, we trained it with Iris dataset.

Finally, we evaluate the performance of our model with the data generated by the Compute Units and the results have been better than both the linear classifier model and the previous polynomial kernel model outcome.

In this paper, we have dealt with simulation of sequence of data generated by the Compute Units to show that we achieve 3D mapping with respect to the input space and not about synchronizing Compute Units to tissues of the kernel which mimic the real cell structure that grows into full-fledged natural tissue.

Unlike the simulation results which are based on few parameters, In natural or real tissues which are 3D Bioprinted there are number of parameters to be considered for pattern analysis.

The QG System

Our objective is to build a target system, we need to generate the cell for the device by running synthesis and implementation on the design. The cell includes custom logic for every Compute unit in the cell container. The generation of custom compute units uses the High-Level synthesis tool, which is the computer unit generator in the application compilation flow. Therefore, it is normal for this step to run for longer period of time than the other steps in the system build flow.

After all compute units have been generated, these units are connected to the infrastructure elements provided by the target device in the solution. The infrastructure elements in a device are all of the memory, control and output data planes which the device is formulated to support an application. The environment combines the custom compute units and the base device infrastructure to generate a cell binary which is used to program the QG device during application execution.

The processing flow of application execution is given as below:-

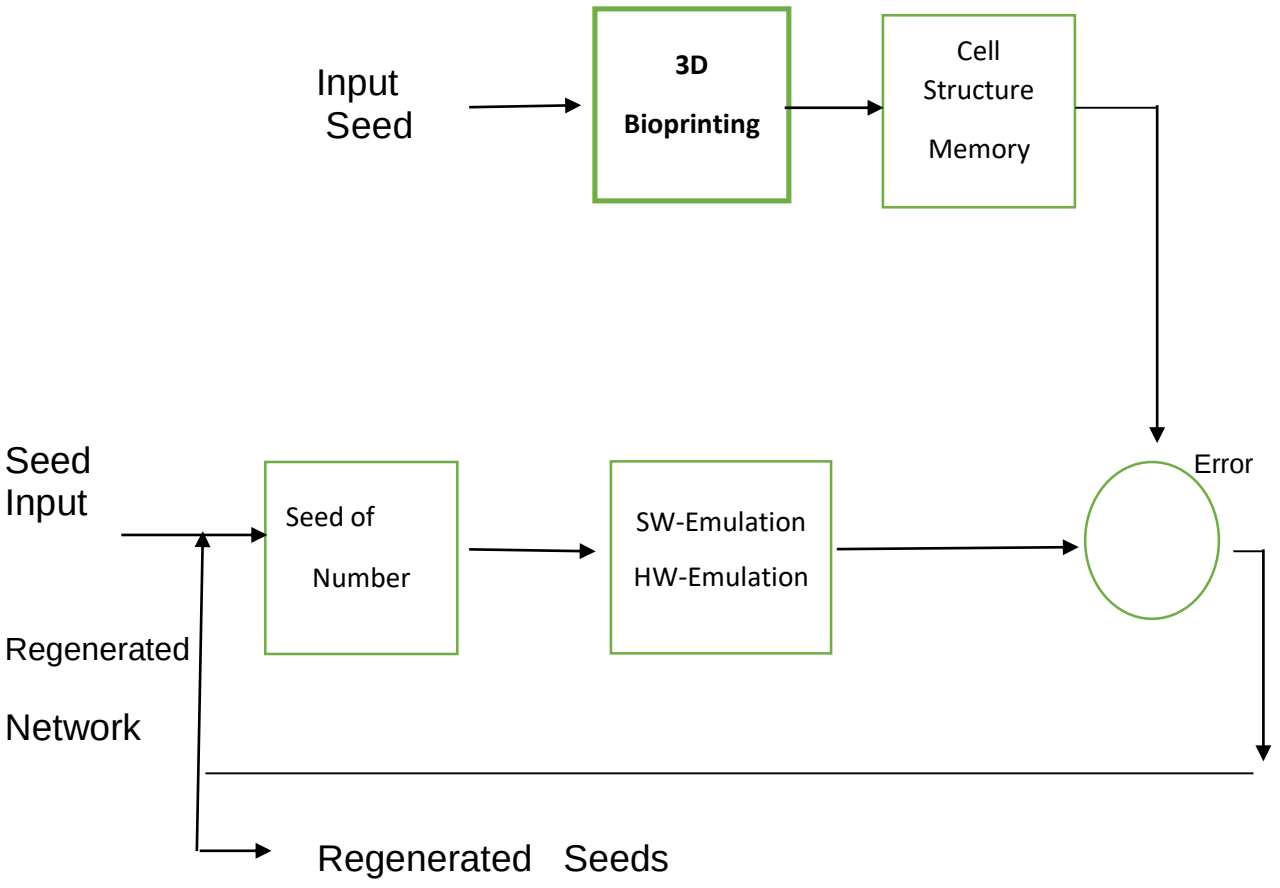

Fig. 1 Process Flow in a Quantum Generator.

The different steps in application are as below:-

1. 3D print a seed and copy its cell structure to memory.
2. Input seed with a seed of a number required.
3. Generate a seed kernel once.
4. Compare the kernel with 3d printed cell
5. If error in seed structure, generate the kernel again.
6. Repeat many times till the seed number is met.

TEST RESULTS

The objective of the Kernel Regression Model is to classify the sequence of data generated based on function parameters of Compute Units to evaluate a logistic regression to have Machine Learned benchmark model. Although, we have generated seed structure with good pattern with better accuracy compared to the linear classifier and the polynomial kernel but this is not mapped to original tissues of seed kernel which are in 3D plane to test the deviation.

CONCLUSION

Quantum Generators (QG) creates new seeds iteratively using the single input seed and the process leads to a phenomenon of generating multiple copies of kernels in repetition. We presented a Kernel Regression Model with Machine Learning for the structured Compute Units and also trained a Kernel Classifier to see if we get better results of reconstruction which can be used to mimic the tissues of real kernel. The results suggest that it is possible to achieve suitable cell structure for quantum generation.

REFERENCE

1. Poondru Prithvinath Reddy: “Quantum Generators: A Formulation of Computational Models of Multiplication”, Google Scholar.
2. Poondru Prithvinath Reddy: “Quantum Generators: Foundations of the Compute Units in Pattern Reconstruction”, Google Scholar
3. Poondru Prithvinath Reddy: “Quantum Generators: Pattern Analysis for 3D Model Reconstruction from the Structured Compute Units”, Google Scholar