# Dendritic Neurons with Genetic Algorithm Based Optimization for Nonlinear Processing

**Ryan P. Mercier**
School of Computing
University of Connecticut
Storrs, CT 06269
ryan.mercier@uconn.edu

## Abstract

Artificial neurons, the building blocks of most artificial neural networks (ANNs), typically compute their output by applying a nonlinearity (e.g., ReLU) after performing a weighted sum of the inputs. While this process is computationally efficient, it has limitations in terms of flexibility and biological plausibility. Traditional artificial neurons are simple in design, resembling their biological counterparts in a very basic sense. However, real biological neurons are far more complex, incorporating dendritic compartments that perform nonlinear processing before transmitting information to the soma for final activation. This paper presents a genetically optimized dendritic artificial neural network (DendriticANN) that dynamically adjusts dendritic complexity through evolutionary strategies. Unlike traditional fixed-topology models, our approach introduces variable dendritic counts per neuron optimized via a genetic algorithm (GA), with adaptive weight inheritance across generations. The model matches baseline ANN accuracy of 88.4% accuracy on FashionMNIST while demonstrating biologically plausible dendritic pruning/growth mechanisms. Key innovations include: (1) GA-driven dendritic count optimization, (2) partial weight transfer between architectures, and (3) mutation rules mimicking biological plasticity.

## 1 Problem Statement

The traditional artificial neuron model can be expressed as:

$$y = f\left(\sum_i w_i x_i\right)$$

where:

- $x_i$ are the input features,
- $w_i$ are synaptic weights, and
- $f(\cdot)$ is the activation function (e.g., ReLU).

While this structure has been effective for many applications, it lacks the biological complexity found in real neurons. Biological neurons have dendrites that allow for localized computation and nonlinear filtering of input signals before reaching the soma for final processing. This work proposes the introduction of dendritic compartments within artificial neurons, which allows for nonlinear computation in these compartments before their outputs are routed to the soma for final integration.

## 2 Objectives

This research aims to:

- Introduce a novel dendritic model where inputs are processed nonlinearly within dendritic compartments before being summed and passed through the soma for final activation.
- Utilize a genetic algorithm (GA) to enable the network to learn optimal dendritic structures and processing strategies based on the specific task at hand.
- Explore the biological plausibility of such a system by simulating the plasticity and self-organization processes found in real neurons.
- Investigate the scalability of this approach and its potential to discover novel neural network architectures that outperform conventional models.

## 3 Proposed Model

The standard artificial neuron model can be expanded to include dendritic compartments, with the following formulation:

$$y = f\left(\sum_{j=1}^{N_d} w_j^c \cdot g\left(\sum_{i=1}^{N_x} w_{i,j}^d x_i\right)\right)$$

**Definitions**

- $x_i$: The $i$-th input feature to the neuron. These can be raw values such as pixel intensities or sensor measurements.
- $N_x$: The total number of input features. This defines how many input values $x_i$ are present.
- $N_d$: The number of dendritic branches (or compartments) in the neuron. Each dendrite receives and processes input separately.
- $w_{i,j}^d$: The synaptic weight between input $x_i$ and dendritic branch $j$. This weight controls how strongly input $x_i$ affects dendrite $j$.
- $\sum_{i=1}^{N_x} w_{i,j}^d x_i$: The total weighted input to dendritic branch $j$, summing over all inputs.
- $g(\cdot)$: A nonlinear activation function applied within each dendritic branch. This models local dendritic processing. Examples include sigmoid, tanh, ReLU, or a small neural network.
- $w_j^c$: The weight connecting dendritic branch $j$ to the soma (cell body). This weight determines how much dendrite $j$'s output contributes to the soma.
- $\sum_{j=1}^{N_d} w_j^c \cdot g\left(\sum_{i=1}^{N_x} w_{i,j}^d x_i\right)$: The total input to the soma, computed as a weighted sum of each dendritic branch's output.
- $f(\cdot)$: The final activation function at the soma, applied after integrating all dendritic outputs. This can be a nonlinearity such as sigmoid, tanh, ReLU, or softmax.
- $y$: The final output of the neuron after dendritic and somatic processing.

The key advantage of this model is that each dendritic compartment performs an independent computation, allowing for localized nonlinear filtering before the information reaches the soma. This enables:

- Task-specific routing of signals,
- Adaptability to new tasks through GA,
- Emergent, efficient dendritic architectures tailored to specific problems.

# 4 Model Architecture

## 4.1 Dendritic Neuron Design

Each neuron processes inputs through multiple dendritic branches before somatic integration:

$$y = \text{LeakyReLU}\left(\sum_{j=1}^{N_d} w_j^c \cdot \text{LeakyReLU}\left(\sum_{i=1}^{N_x} w_{i,j}^d x_i\right)\right) \tag{1}$$

- **Dendritic Layer**:
  - $N_d$ dendrites per neuron (evolved via GA)
  - Dendritic weights $w_{i,j}^d \in \mathbb{R}^{N_x}$ process inputs independently
- **Somatic Layer**:
  - Fixed single soma per neuron
  - Combines dendritic outputs via weights $w_j^c \in \mathbb{R}^{N_d}$

## 4.2 Network Topology

The DendriticANN consists of:

Table 1: Architecture Specifications

| **Component** | **Parameters** |
|---|---|
| Input Layer | Linear($784 \to 128$) |
| Hidden Layer | 128 DendriticNeurons ($N_d \in [1, 10]$) |
| Output Layer | Linear($128 \to 10$) |
| Activation | LeakyReLU ($\alpha = 0.01$) |

## 4.3 Key Differences from Baseline ANN

- **Structural Flexibility**: Dendrite counts $N_d$ per neuron are evolved rather than fixed
- **Local Processing**: Two-stage activation (dendritic $\to$ somatic) enables sub-neuron computation
- **Parameter Efficiency**: Sparse dendritic configurations emerge via GA pruning

# 5 Genetic Optimization Framework

## 5.1 Algorithm Overview

The genetic algorithm evolves dendritic structures over generations through:

- Population initialization with random dendrite counts ($N_d \in [1, 10]$)
- Fitness evaluation via short-term training (3 epochs)
- Elitism, crossover, and mutation of dendrite counts
- Adaptive weight inheritance between generations

### 5.2 Implementation Details

**Algorithm 1** DendriticANN Genetic Optimization

0: Initialize population $P$ with $|P| = 50$ random architectures generation $1 \rightarrow 20$ individual $i \in P$
0: Train $i$ for 3 epochs (Adam, LR=0.0001)
0: Evaluate fitness $f_i = \text{TestAccuracy}(i)$
0: Sort $P$ by $f_i$ descending
0: Preserve top 2 elites ($P_{\text{next}} \leftarrow P[:2]$)
0: Select parents $P_{\text{parents}} \leftarrow P[:40]$ (top 80%) $|P_{\text{next}}| < 50$
0: $p_1, p_2 \leftarrow \text{RandomChoice}(P_{\text{parents}})$
0: $c \leftarrow \text{Crossover}(p_1, p_2)$ {Average $N_d$}
0: $c \leftarrow \text{Mutate}(c)$ {$\pm 1$ dendrite, 10% prob}
0: $P_{\text{next}} \leftarrow P_{\text{next}} \cup \{c\}$
0: $P \leftarrow P_{\text{next}}$ =0

### 5.3 Weight Inheritance

When architectures change during crossover/mutation:

$$\mathbf{W}_{\text{child}}^{(l)} = \begin{cases} (\mathbf{W}_{p1}^{(l)} + \mathbf{W}_{p2}^{(l)})/2 & \text{if shapes match} \\ \text{PartialCopy}(\mathbf{W}_{p1}^{(l)}, \mathbf{W}_{p2}^{(l)}) & \text{otherwise} \end{cases}$$

where PartialCopy preserves overlapping weights and randomizes mismatches.

## 6 Previous Research

Recent research has begun to explore the integration of dendritic properties into artificial neural networks (ANNs) to enhance their performance and robustness. A notable example is the study titled *"Dendrites endow artificial neural networks with accurate, robust, and efficient learning"*, published in *Nature Communications* [1]. This study demonstrates that incorporating dendritic structures into ANNs can lead to more precise, resilient, and parameter-efficient learning.

While this research highlights the benefits of dendritic properties in ANNs, our proposed project introduces a novel approach by leveraging Genetic Optimization to dynamically optimize dendritic structures and processing strategies. Unlike previous models with static dendritic architectures, this approach allows the network to adapt its dendritic configurations in response to specific tasks and environmental feedback. This dynamic adaptability aims to enhance learning efficiency and task-specific performance, moving beyond the fixed architectures explored in prior studies.

## 7 Expected Contribution

By integrating GA into the design of dendritic neural networks, the project seeks to emulate the plasticity observed in biological neurons more closely. This approach not only enhances the biological plausibility of artificial neurons but also allow the artificial neuron to "discover" the most effective dendritic architecture for a given task. Just as real neurons adapt their dendritic structure based on experience and learning, the artificial dendritic model will learn to optimize its processing strategies, allowing it to explore novel configurations and structures that may not be immediately obvious or pre-defined.

- Multi-layer dendrites, where each layer performs a different type of nonlinear processing,
- Dendritic compartments with varying synaptic weights and nonlinear transformations, leading to specialized units capable of performing complex tasks,
- Hierarchical dendritic structures that resemble pyramidal or Purkinje neurons.

# 8 Experiments

## 8.1 Setup

- Dataset: FashionMNIST (60K train, 10K test)
- Model Architecture: Input layer of 28×28, three hidden layers with 128 neurons each, and an output layer with 10 classes
- DendriticANN Parameters: Minimum 1 dendrite and maximum 10 dendrites per neuron
- GA: 50 individuals, 20 generations
- Hardware: NVIDIA RTX2000 ADA GPU

## 8.2 Results

| Model | Final Test Accuracy | Training Steps |
|---|---|---|
| Baseline ANN | 88.78% | 60 Epochs |
| DendriticANN (GA) | 83.81% | 3 Epochs for 20 Generations |

Table 2: Performance comparison 3 layers

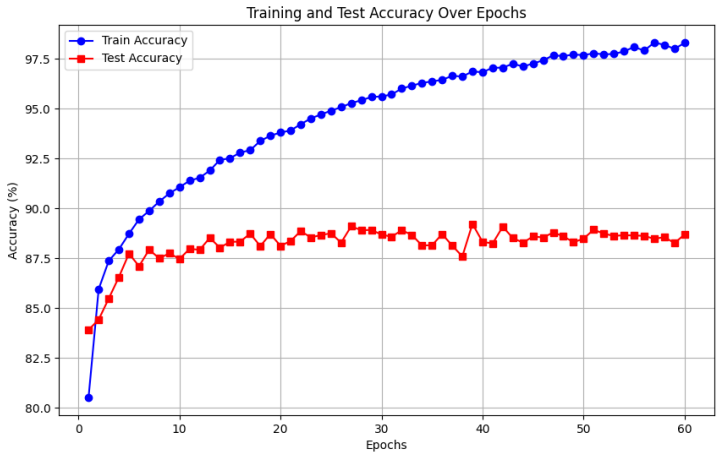


Figure 1: Baseline ANN train and test accuracy 3 Layers of 128 Neurons

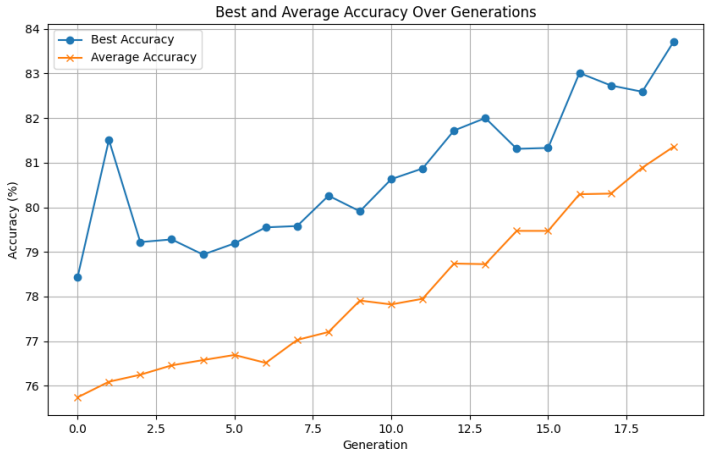


Figure 2: GA structural optimization test accuracy (Weights randomly initialized every generation)

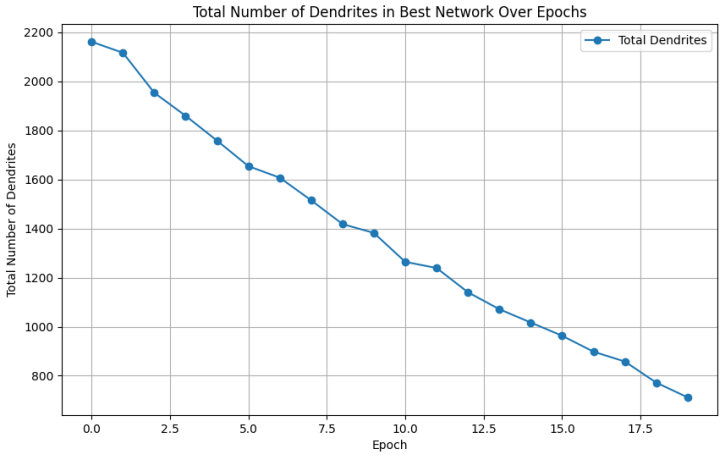


Figure 3: Average number of total dendrites over each generation when weights are randomly initialized every generation

| Model | Final Test Accuracy | Training Steps |
|---|---|---|
| Baseline ANN | 88.45% | 60 Epochs |
| DendriticANN (GA) | 88.43% | 3 Epochs for 20 Generations |

Table 3: Performance comparison 1 layer

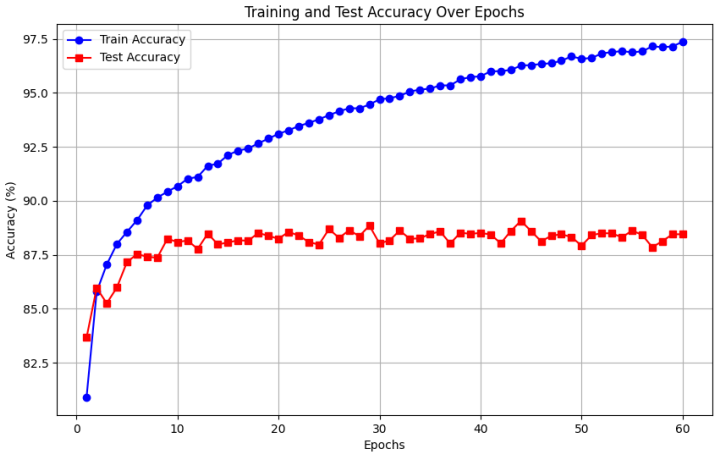


Figure 4: Baseline ANN train and test accuracy 1 Layer of 128 Neurons

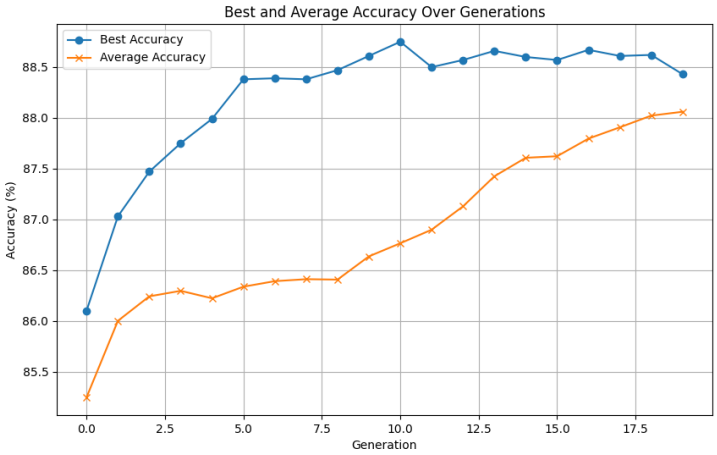


Figure 5: GA structural optimization test accuracy (Weights inherited)

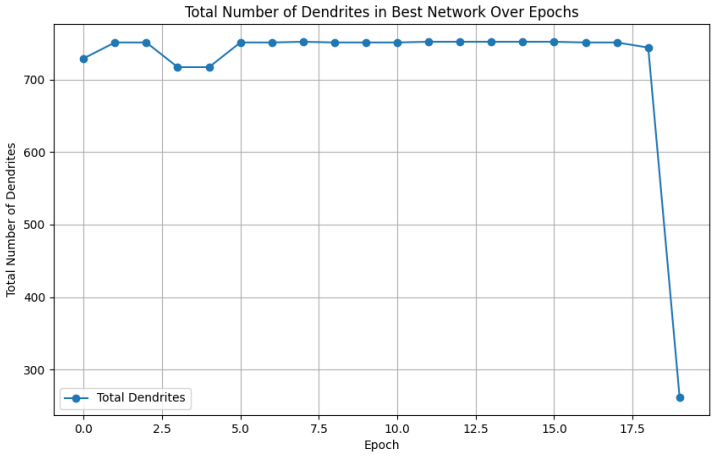


Figure 6: Average number of total dendrites over each generation when weights are inherited

### 8.3 Analysis

I initially investigated reinitializing weights randomly every generation instead of inheriting them from their parents as a proof of concept. This was to ensure that evolving the topology of the graph would actually improve its generalization. As we can see both the average and best accuracies increased over the training showing that evolving the topology was promising. This model generalized much slower than the traditional ann as expected as each DNN only had 3 training epochs instead of the 60 our baseline ann. The idea is that once weights are inherited 3 epochs for 20 generations will be the same 60 training steps. The fact that there was a large divergence between the train and test accuracy for the regular ann combined with a very clear downward trend in dendrites over time led me to conclude that these models are much too complex for the task making the dendrites a hinderance and slowing down convergence. I reduced it to one layer of 128 neurons for each model and implimented the inheritance of weights from parents to children. Currently it is a relatively naive implimentation and simply averages the number of dendrites between the two parents for each neuron. Each weight is averaged unless it is not present in one of the parents. In that case it is just copied over. As we can see from the second set of graphs inheritance has greatly improved the result making it now on par with the regular ANN. The training curve of the best model is pretty close to the training curve of the regular ANN. We can also see that now that we have reduced the layers from 3 to 1 the model is incetivized to utilize the dendrites, averaging about 5 per neuron. Now that we are on par with the ANN, I believe that improving the crossover and mutation will allow the model to surpass the regular ANN. DendriticANNs may need more generations to surpass baselines due to structural search overhead.

## 9 Limitations

- Computational cost
- Naive weight inheritance
- May be prone to vanishing gradient
- Fixed hidden layer size (128 neurons)
- Fixed number of layers
- Short training epochs may underfit

## 10 Planned Work

- Currently we have Simple Crossover/Mutation: The crossover averages dendrite counts, which may not be optimal. The mutation is also simplistic (small random changes).

- Add batch normalization
- Add dropout
- Add adaptive learning rate
- create seperate validation set and use that for the fitness function to prevent fitting structure to the test set
- Add Dynamic Dendrite Adjustment: The dendrite counts are evolved but not pruned or grown dynamically based on areas with high or low weights during training.

## 11 Conclusion

This research proposes a novel approach to artificial neurons by incorporating dendritic compartments that process inputs nonlinearly before being integrated at the soma. Using a genetic algorithm for optimization, the network can dynamically learn the most efficient dendritic structures and processing strategies tailored to specific tasks. This model not only increases the biological plausibility of artificial neurons but also provides a flexible and scalable framework for developing more powerful and task-specific neural networks.

## References

[1] Chavlis, S., Poirazi, P. Dendrites endow artificial neural networks with accurate, robust and parameter-efficient learning. *Nature Communications*, vol 16, 943 (2025). https://doi.org/10.1038/s41467-025-56297-9 https://www.nature.com/articles/s41467-025-56297-9