NAS Concept and Architecture

Luc Alexander

August 2023

Contents

1 Overview

This architecture search method is designed explicitly for a unique graph-based neural network structure visualization. The graph consists of nodes, each representing a layer of neurons, and edges signifying the flow of data between these layers. The graph can exhibit branching, where a single layer's output can be directed to multiple subsequent layers. These branches can then converge, either with one another or back into the main flow. Notably, the architecture also incorporates memory links, potentially giving the model capabilities similar to recurrent neural networks or LSTMs allowing however for more targeted data to be remembered. The process of scanning the search space of possible architectures uses a reinforcement learning algorithm to accomplish broad neural architecture exploration, and then poses a continuous supergraph of the narrowed search space for a differential approach to architecture exploitation.

2 Graph-based Neural Network Architecture:

2.1 Foundation:

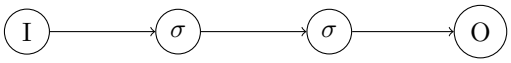
The architecture is conceptualized as a directed graph where nodes and edges are the primary components. This graph is not strictly feed-forward like conventional deep learning models; it possesses more intricate connections that provide the model with added flexibility and potential capabilities.

This graph-based architecture attempts to offer a fresh and versatile perspective on neural network design. By incorporating branching, convergence, and memory mechanisms within a graph framework, it holds the potential to capture complex data patterns and dependencies, going beyond traditional feed-forward and recurrent architectures.

Each node in the graph represents a layer of neurons. This layer could be any standard neural network layer, such as a fully connected layer, convolutional layer, pooling layer, etc. Nodes process incoming data from preceding nodes and transmit the output to subsequent nodes in the graph.

Edges define the flow of data between nodes. Each edge carries a vector of data from one node (layer) to another. The architecture allows for multiple edges to emanate from a single node, enabling branching.

Simple feed forward network



2.2 Branching & Convergence:

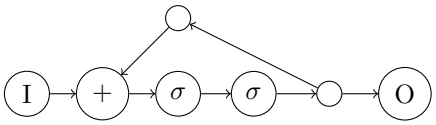
Certain nodes, termed branching nodes, split the data flow into multiple paths, allowing the data to be processed in multiple distinct sub-graphs or branches. This branching enables the parallel processing of data and the exploration of different feature transformations concurrently.

Convergent nodes combine data from multiple incoming paths. This can be from the branches or even from the main data flow. Various methods can be employed for this convergence, such as element-wise addition, element-wise multiplication (which can act as a gating mechanism), concatenation, or other custom operations.

2.3 Memory Links & Recurrence:

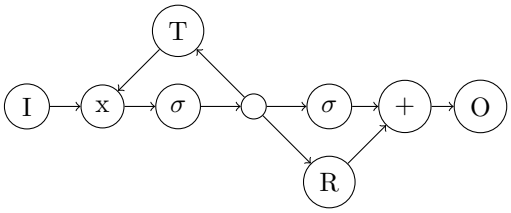
Unique to this architecture are the memory links that loop back and connect a node to previous nodes in the graph. These links introduce a form of recurrence, allowing the network to maintain and utilize memory of past computations, similar to recurrent neural networks or LSTMs.

Standard RNN



Within these memory links, gating mechanisms can be employed to control the flow of recurrent information. These gates can be inspired by mechanisms like those in LSTM or GRU cells, deciding what recurrent information to retain, modify, or discard. The introduction of gates within memory links ensures that the model can learn to manage its internal memory more effectively, emphasizing relevant historical information and diminishing noise or irrelevant data.

Network with gated memory unit and a residual branch



2.4 Training & Optimization:

Given the recurrent nature introduced by memory links, the BPTT algorithm is used for training. This involves "unrolling" the graph over time steps, converting the cyclic graph into an acyclic one for the purpose of backpropagation.

Due to the complex structure of the graph, with branching, convergence, and memory links, training can present unique challenges like local optima, saddle points, and rugged loss landscapes. Advanced optimization techniques and adaptive gradient methods,

such as Adam or RMSprop, can be used to navigate these challenges and ensure stable training.

3 Combined Neural Architecture Search for Graph-based Neural Networks: Reinforcement Exploration and Differential Exploitation

This approach elegantly combines the broad exploration capabilities of reinforcement learning with the fine-grained exploitation prowess of differentiable search. Tailored for the described graph-based model visualization, it ensures that the nuances and potential of such an architecture are thoroughly harnessed.

3.1 Reinforcement Learning for Exploration:

Controller Design: An RNN-based controller is employed to generate sequences of decisions that describe the architecture of the graph-based neural network. This includes choosing layers, determining branching and convergence points, and establishing memory links.

Sampling Architectures: Over multiple iterations, the controller samples different graph configurations. Each configuration represents a unique neural network architecture within the framework of the graph-based visualization.

Training & Evaluation: Each sampled architecture is trained for a brief period and subsequently evaluated on a validation set. The performance on the validation set determines the reward, which might be the accuracy for classification tasks or an inverse metric like loss.

Based on the received reward, the controller's parameters are updated using policy gradient methods, refining its policy to propose potentially better architectures in subsequent iterations.

3.2 Differentiable Search for Exploitation:

Supergraph Construction: A large supergraph is created that encompasses all potential architectures the controller might propose. This supergraph is designed in a way that respects the unique features and utilities of the graph-based architecture, including branching, convergence, and memory links/gates.

Continuous Relaxation: To make the architecture search problem differentiable, the discrete architectural decisions are relaxed into continuous ones, enabling gradient-based optimization.

Optimization: Using the gradients obtained from the validation set performance, the architecture is optimized. Given the graph's intricate nature, adaptive gradient methods (like Adam, RMSprop, etc.) can be employed to navigate the challenges posed by the complex structure of the graph of layers.

Once the optimization converges, the final architecture is extracted from the supergraph by making hard decisions based on the continuous parameters.

3.3 Key Features and Mechanisms:

Memory Links: These are special links in the graph that loop back to previous nodes, embedding a form of recurrence and memory into the architecture. Given their recurrent nature, Backpropagation Through Time (BPTT) is utilized to train architectures that heavily rely on these links.

Gates: Within the architecture, there can be specific mechanisms or layers that control the flow of information, akin to the gates in LSTM or GRU cells. These gates can emerge naturally from the search process or be explicitly introduced as operations in the search space.

Convergence Methods: Multiple methods can be employed for merging branches in the graph. This includes element-wise addition (akin to skip connections in architectures like ResNet) or element-wise multiplication, which can act as a gating mechanism. Other methods might include concatenation or more intricate merging operations.

Optimization Challenges: Given the graph's complexity, optimization can present challenges like local minima, saddle points, and rugged loss landscapes. To address these, advanced optimization techniques and adaptive gradient methods can be leveraged, ensuring stable and effective optimization of the model.

4 Definitions:

- **Neural network layer**: A computational unit consisting of neurons where each neuron computes a weighted sum of its inputs followed by an activation function.
  - **Feedforward**: A type of layer in which information moves in only one direction—forward—from the input nodes, through the hidden nodes, and to the output nodes.
  - **Pooling**: A layer, typically used in convolutional neural networks, that reduces the spatial dimensions of the input by taking the maximum or average value of a group of inputs.
  - **Convolutional**: A type of layer that uses convolution operations to scan the input data with filters to extract features. Typically used in image processing.
- **Residual Neural Network (ResNet)**: A type of neural network that introduces skip connections, or shortcuts, to jump over some layers to combat the vanishing gradient problem.
- **RNN (Recurrent Neural Network)**: A type of neural network where connections between nodes form a cycle, allowing it to maintain a 'memory' of previous inputs in its internal state.
- **Gated Memory Unit**: A type of network structure used in recurrent neural networks to control the flow of information.
- **LSTM (Long Short-Term Memory)**: A type of RNN designed to recognize and remember long-term patterns or sequences without relying on a predefined window size.
- **GRU (Gated Recurrent Unit)**: A variant of LSTM which combines the forget and input gates into a single update gate. It also merges the cell state and hidden state.
- **Gradient descent**: An optimization algorithm used to minimize some function by iteratively moving in the direction of the steepest decrease of that function.
- **Back propagation**: The primary algorithm for performing gradient descent on neural networks. It calculates the gradient of the loss function with respect to each weight by applying the chain rule.
- **BPTT (Backpropagation Through Time)**: An application of backpropagation to sequences, used for training recurrent neural networks.
- **Adaptive gradient descent methods**: Optimization algorithms that adjust the learning rate during training.

- **Adam (Adaptive Moment Estimation)**: An optimization algorithm that computes adaptive learning rates for each parameter by combining the advantages of AdaGrad and RMSprop.
- **RMSprop (Root Mean Square Propagation)**: An optimizer that uses the moving average of the squared gradient to normalize the gradient itself.

5 References

Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators

Illustrated Guide to LSTM's and GRU's: A step by step explanation

A Supergraph-based Generative Model

Reinforcement Learning: An Introduction

Back-propagation Through Time

A Complete Guide to Adam and RMSprop Optimizer

Adam