Efficient Neural Architecture Search via Parameter Sharing: Summary

Presentation Summary

Motivation for Neural Architecture Search

The improvements seen in image classification performance with the first convolutional nets, and more recent architectures ResNet and DenseNet show that architecture design is critical to training a high-performance neural network

- Key problem:
    - Most of the human design techniques are very compute intensive and time intensive
    - Human design for model architectures requires a lot of domain expertise.
- Idea:
    - Generate an architecture for a neural network with a neural network

NAS

Key Ideas of NAS

- A controller, an RNN, generates child models which are trained to convergence
- The accuracy on a validation set is used to estimate how well the architecture will perform for the target application
- Reinforcement learning is used to update the controller parameters
    - This allows optimizing a function (the expected validation accuracy) which is not differentiable with respect to those parameters
    - Policy gradients with REINFORCE are used

NAS for CNNs

- Until some target number of layers have been generated, use the controller RNN to choose the parameters for each layer (convolution size, stride length, and the number of filters)
- Inject pooling layers at constant positions
- By adding an attention mechanism at each layer, NAS calculates the probability with which to add a skip connection from any previous layer

NAS for RNNs

- NAS searches for the optimal RNN cell, which is executed at each timestep and is used to determine the next hidden state and cell state given the current input and the past hidden and cell states
- Given a predefined tree of computations in that cell, the controller chooses the combination function and activation function at each node
- The controller finally outputs where in the computation the previous cell state should be used as an input and which intermediate result should be used to determine the next cell state

Limitations of NAS

- NAS is extremely computationally intensive
    - The authors took 32-43k GPU hours to discover their final architecture
    - Cause: the child models which are generated are trained to convergence despite only using the final weights to calculate the validation accuracy

ENAS

Main ideas of ENAS

- Sharing parameters among models in the search space. This is where computational gains come from. So we do not throw away weights after training a child model to convergence.
- Alternating between optimizing model parameters on the training set and controller parameters on the validation set. This ensures that the models generated by the controller are not overfitting to the dataset, and thus can generalize well to the test/unseen data.
- *Key Assumption:* parameters that work well for one model architecture, would work well for others. This is a fact that has been validated experimentally and is the foundation of transfer learning where a model is trained on a specific task, and then with minimal changes, performs reasonably well on new/unseen tasks.

Graphs in ENAS

- The graphs on which NAS iterates can be viewed as sub-graphs of a larger graph. So we represent NAS's search space using a single directed acyclic graph (DAG) and ENAS samples subgraphs from this DAG, which correspond to child models.
- Associated with this DAG is a weight matrix, W, where W_{ij} represents the weight of the edge between nodes i and j

ENAS for RNNs

- For RNNs, the controller samples a subgraph from the DAG.
- For each node, there are two decisions to be made:
    - Which previous node will act as input to the current node
    - What activation function to use for producing the output at the current node.
- For a DAG with N nodes, the search space size is $4^n \times n!$

ENAS for CNNs

- The process of architecture design is similar to that in the RNN case.
- At each step, the controller has two decisions to make:
    - Selecting a compute operation
    - Selecting previous nodes to connect for information flow
- The search space is broadly divided into two cases. Micro Search and Macro Search
    - **Macro Search**: The controller searches for the optimal complete convolutional network. The controller makes a decision at every layer.
        - The search space for macro search is very big $O(6^{L}*2^{L(L-1)})$
    - **Micro Search**: The controller searches for building blocks which are stacked multiple times to create a complete convolutional network architecture.
        - The search space for micro search is smaller: $O(5*(B-2)!)^4$)

Performance Gains of ENAS

- ENAS performs comparably to NAS but finds an architecture much more quickly
    - For RNN, ENAS takes ~1000x less GPU hours to find the model architecture.
    - For CNN, ENAS takes ~50,000x less GPU hours to find the model architecture.

Limitations of NAS and ENAS

- The architectures created by NAS and ENAS are more difficult to interpret than a typical human-designed architecture.
- Both NAS and ENAS can only organise basic building blocks of model architecture; they cannot develop a truly novel design.
- Search space design is important; some domain knowledge is still needed.

Discussion

In CNN search, how do we connect the final convolutional layer to the Softmax layer?

Neural architectures discovered with NAS are used in a similar way as human designed architecture. In the CNN case, e.g. for image classification, we take the output of the model, add

a global pooling layer, add a fully connected layer with output dimension equal to the number of classes and end with a softmax layer. The output which is sent to those final layers is, in the case of NAS with skip connections, all layer outputs which have not been connected to downstream layers are used as the initial model output (used as described above), the output of the final layer in basic NAS, the output of the final reduction cell in ENAS with micro search, and is chosen in the same way as in the other layers of the network in ENAS with macro search.

Why NAS; why not just use a flexible architecture and focus on inference?

Model architecture improvements have shown significant gains in the past for many tasks. Any downstream benefits from a better model architecture. For that reason, finding better model architectures is still an active area of research.

Using flexible architectures that can work well for multiple tasks is a good strategy when rapidly developing a model which can perform acceptably well on a particular task. In order to push the performance further, finding optimized architectures for that particular task has been shown to be essential in order to get state of the art performance.

Improving inference techniques is a parallel area of research, and may allow improved performance by making the use wider and deeper models feasible. However, as has been made clear in image classification, larger networks are not, alone, to enable improved performance; improved architecture designs, which may be developed with techniques such as NAS, are essential as well.

One of the limitations NAS has is that it requires defining a search space which requires domain-specific knowledge; is there any progress in the field where these algorithms can be applied for novel domains?

The ultimate goal of the area of Neural Architecture is that of finding techniques that can generate model architectures without domain expertise and which can work well for any novel domain. The techniques we saw in NAS with RL or ENAS are a step in that direction but the Neural Architecture Search field requires substantial further research to enable architecture search which is completely independent of domain expertise.

Any thoughts on why micro search yields better architectures than macro search?

Micro space search has a much narrower search space. This makes finding better architectures faster. It is possible that the design of search space for micro search is more optimal for CNN design. That could explain why, empirically, the architectures found using micro search were better. This may change if better techniques and/or search space designs are developed for macro search.