On The Diversity of Neural Network Architectures And The Concept of An Ideal Architecture

Anton Shakov, Under the Supervision of Dr. Yuly Billig

1 Introduction

Universal approximation theorems state that neural networks will approximate any function given enough training, assuming an arbitrarily large number of neurons and/or hidden layers. We consider a related question: when tasked with approximating a function, does there exist an ideal neural network architecture for achieving said approximation? There is likely no single notion of what constitutes an ideal network. In this paper, we consider the following factors: number of neurons, number of connections, and loss function output. In essence, the question we are asking is "how small can a neural network be made while continuing to produce the desired output?" From now on, we will refer to the theoretical minimum network as the ideal architecture. It's important to note that the question as it is phrased here does not pertain to the trainability of a network. One of the most common bits of AI trivia is that a neural network with more neurons and connections than necessary is likely to approximate a function faster than a network with the bare minimum number of neurons and connections. It is still a meaningful question to ask about the optimal architecture with respect to trainability, and it is likely that for a given function, ideally trianable networks, i.e. networks that are just big enough to be trained within a reasonable amount of time, and simply ideal networks, as described earlier, are related to one another. For the purposes of our question we care only about the ideal network which, upon assuming the correct weights and biases produces the desired output. Is there a general way to apply changes to our network to converge towards the ideal? Can we know by looking at a certain architecture that it is likely the optimal one for approximating a given function? Is there a meaningful reason why a certain architecture is ideal, or is its existence rather arbitrary? In this report, we attempt to answer the first question, speculate about the second, and admit to our ignorance in regard to the third.

2 Converging Towards the Ideal Architecture

We initially thought to converge to the ideal network by growing directed graphs from the input to the output layers. The reason for this was the hope that directed graphs would allow for greater freedom in our endeavor to gradually mutate towards the ideal network, giving us more possibilities for graphs than the standard hidden-layer approach. However, for lack of standard methodology and out of concern that directed-graph neural networks would turn out to be unruly and needlessly complicated, we present the following ideas through the paradigm of hidden layers. Only hidden layers and the neurons within them are added and removed. Connections are made automatically between all neurons in adjacent hidden layers, just like in standard neural network models. Translating this approach into the more general language of directed graphs should not prove too difficult, as long as one has a general way of growing and shrinking directed graphs. This also allows us to reduce the number of variables in our architecture loss function to consider only the number of neurons and the loss function output (L - conventional neural network loss function, N - number of neurons):

$$L_A = N(L+1)$$

or,

$$L_A = N^{L+1}$$

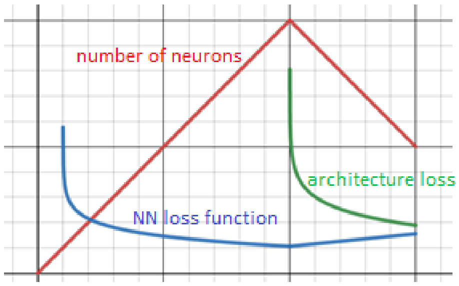
The above equations both satisfy the following important conditions to be considered as potential architecture losses (L_A - architecture loss):

1. As N is lowered, L_A is lowered
2. As L is lowered, L_A is lowered
3. If $L = 0$, L_A is lower iff N is lower

We note that n is never 0 as it includes the number of neurons in the input and output layers which are predetermined by the user and immediately frozen. The changes described below will only be applied to the hidden layers.

An alternative to growing directed graphs is assuming a layered structure to our network. This allows us to focus on the number of neurons and loss function output as our factors/variables. We decide whether to grow or shrink our network by choosing a random integer i from the range $[x, y]$, where x is a negative integer and y is a positive integer. We then choose the layer to/from which to add/remove i neurons. One of the options for where to add neurons shall be “new layer”. This means adding an entirely new hidden layer, which from now on will be considered one of the options in our future selections. When we remove all the nodes from a layer, the layer itself is removed. The excess neurons are removed from another layer which is chosen at random again. After the updated network is trained, changes are finalized only if the architecture loss is lowered. Otherwise, the changes are undone, and the algorithm applies a completely new set of changes and once again checks if they are to be finalized. Note that for the purposes of this approach weights and biases are reset after each training session. This is to minimize any bias in our comparisons of various architectures.

For both growing-directed-graphs and adding-neurons-to-layers approaches we use a "bulking-cutting" strategy when it comes to choosing how much we increase/decrease the size of our network. We begin by increasing our network's size slightly more than we decrease it. Begin with x, y such that $|x| < |y|$. We continue like this until the neural network loss output is arbitrarily close to 0 (say within epsilon). The "bulking" is done. We then begin to "cut" the network by shrinking it slightly more than we grow it - update x, y such that $|x| > |y|$. We continue cutting until the architecture loss reaches a plateau. This will theoretically happen if our network is unable to make any further meaningful changes i.e. ones that lower the architecture loss any more. We are now done.



3 Considerations

If the upper bound for length of training is too small, the ideal architecture will be out of reach. In practice this will mean that we will train the network until there is no longer improvement in the loss, simply because there are too many layers and we are not spending enough time training. This means that choosing a proper "length of training" constant is something that must be done thoughtfully.

We must also choose a termination condition to decide when a meaningful change is no longer likely to be made. Say, after 10 failed attempts at improvement with no changes we stop mutating the network and terminate the program.

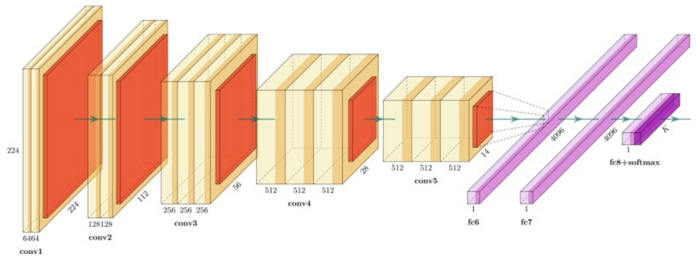
We can think of searching for the ideal architecture as fishing where your pole, line, and hook are training length, termination condition, and epsilon. If your epsilon is too small for the chosen training length, you will "bulk" indefinitely, but since the termination condition won't let this happen, with an unchanging/increasing neural network loss and an increasing number of neurons the architecture loss will cease to decrease and we will terminate with a network that likely is not the ideal one.

4 Some Thoughts and Concluding Remarks

A central aspect of converging towards an ideal network is finding a proper balance between generality and efficiency. For instance, utilizing directed graphs is more general than relying on the standard layered-structure, as hidden-layer neural networks are directed graphs, but directed graphs do not necessarily have a hidden-layer structure. For this reason, emphasizing only the generality of our evolution may ensure that, theoretically, we will eventually converge towards our ideal. In practice, however, this may take a completely infeasible amount of time. The same concern applies to the probabilistic approach in general. A more intelligent mutation of our network could be based around signals sent by certain neurons of the network, for example finding discrepancies in derivative values. However, as soon as we specify the exact signal or set of signals our algorithm is looking for, we risk sacrificing generality and reducing the pool of potential candidates for the ideal network.

Some general traits and features are preferable in networks depending on the function we are approximating. However, just like there is a diversity of neural network losses, there can be a diversity of architecture losses, and since the concept of an ideal network is tied to the architecture loss, it is likely that there are families of ideal networks for every approximated function as opposed to a unique ideal.

The main practical point of observing a convergence towards an ideal network with respect to different architecture losses, as we see it, is to recognize and isolate useful traits and features that naturally emerge when approximating a specific function, such as differently-sized hidden layers for pooling, embedding, etc. when working on image processing.



References

Bin Yang and Enguo Cao. Advances in Computers, Deep Convolutional Neural Networks `https://www.sciencedirect.com`.