Exploring Neural Network Modularity Using Model Stitching

Adriano Hernandez, Rumen Dangovski & Peter Y. Lu [*]
MIT EECS
Cambridge, MA 02139, USA
{adrianoh,rumenrd,lup}@mit.edu

Abstract

We expand *model stitching* (Lenc & Vedaldi 2015) as a methodology to compare neural networks. Previously, Bansal, Nakkiran & Barak used it to compare the representations learned by differently seeded and/or trained neural networks. We use it to compare the representations learned by neural networks with different architectures. This gives us insight into the properties of stitching. Namely, we find that stitching can reach unintuitively high accuracy for intuitively different layers if those layers come earlier in the first (sender) network than in the second (reciever). This leads us to hypothesize that stitches are not in fact learning to match the representations expected by reciever layers, but instead to find representations which yield similar results. Exploration on this front is ongoing.

1 Introduction

1.1 Motivation

The success of deep learning for visual recognition has been attributed to the ability of neural networks to learn good representations of their training data [Rumelhart et al., 1985]. That is, intermediate outputs of good neural networks are believed to encode semantic information about their inputs, which they use for classification and/or other downstream machine learning tasks. However, our understanding of these representations is somewhat limited. We do not know why good representations are learned, nor do we have a robust theory to characterize them. For example, we do not know how to compare representations.

Being unable to compare representations is a fundamental limitation because it precludes us from understanding whether different models are learning essentially the same thing. This is important, because one of our great hope is that different neural networks are able to learn, by some measure, qualitatively similar representations. Metaphorically speaking, we'd like it to be the case that two models' representations were the same, but in different "languages." It is as if a ResNet18 were speaking French and a ResNet34 English, but with an effective translator we could confirm that they were indeed speaking the same sentence.

Imagine we had such a so-called translator on hand. If our hypothesis were to be proven correct, it would not only tell us that neural networks learn similar representations, but also lend credence to the belief that neural networks are capable of learning representations that are human-interpretable and capture the most relevant semantics of images. This is because, if all successful learners learn the same underlying semantics, then it is intuitive to think that it is not mere coincidence causes them to learn those semantics.

As such, if our hypothesis were correct, it could give us more confidence in the robustness of certain architectures. Moreover, it would enable us to tackle neural network interpretability from a new angle: instead of trying to understand specific networks layer by layer or neuron by neuron, we could compare different neural networks' representations and then cluster those represnetations by

[*] You can find our other work at http://www.a14z.blog/, http://super-ms.mit.edu/rumen.html, and https://peterparity.github.io/ respectively.

similarity. That would then enable us to extend previous interpretability results to broader classes of neural networks.

Recent advances in model stitching [Bansal et al., 2021] have made great strides towards our goal. They use a tool called Model Stitching to not only confirm that very similar neural networks learn similar representations, but also improve the accuracy of models and potentially even reduce their training cost (in compute). Bansal et al. confirm that, on standard datasets such as CIFAR-10, standard ResNets, such as ResNet18, are able to learn similar representations irrespective of their initial random seeds, layer widths, or (to a limited extent) training methodology. However, their research has two main holes. Good lead into explanation of how previous methods are not enough

Firstly, it only confirms that networks with the same fundamental architecture have similar represnetations at the exact same layers. It is much harder to compare the representations of neural networks with different architectures since they have different shapes, but that is precisely what we seek to ask through this paper. Do all ResNets learn similar representations? What if one has 18 layers and another has 34? Are those 18 layers the same as the first 18 of the 34-later ResNet or are they something else?

Secondly, it neglects to address the question as to whether stitches learn the same (or approximate) representations as the original layers, or whether they learn something else which nonetheless has the same effect on accuracy. The perplexing results we find in our paper give some credence to the latter possibility, but are being further investigated.

1.2 Stitching

We broaden the definition of stitching very little from that as defined in [Bansal et al., 2021]. In that paper, stitching involved taking the intermediate output of some layer, which we call the sender, transforming it, and then inputting that into some intermediate input in another network. We call the layer we input into the reciever. We will also refer to the two networks, respectively, as sender and reciever when convenient and contextually clear. The representation that would normally be recieved by the reciever we call the expected representation (sent by the expected sender: the previous layer in the reciever network).

The layers up to the sender (inclusive) we call a prefix and the layers from the reciever to the end (inclusive) we call a suffix. The transformation used is called the stitch, and it is learned with gradient descent (or some other optimizer) on a frozen prefix/suffix pair. The network formed by concatenating the sender prefix, the stitch, and the reciever suffix, is called the stitched network. Just like Bansal et al. we use the accuracy of the stitched network as our primary training task. (We differnetiate this later with different tasks such as the copying task: training a stitch to transform the prefix's cached outputs into the suffix's expected inputs using a similarity metric like mean squared error. When we use input/output representations, as opposed to images classes, as our dataset and labels we refer to that dataset as the representation dataset).

Consider moving thoughts in parentheses to footnotes

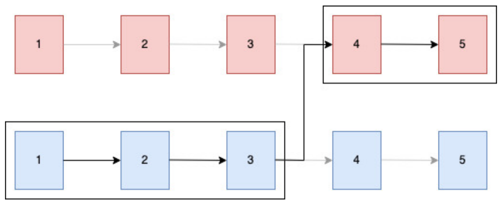
The stitch belongs to a simple function class such as 1x1 convolutions (depthwise) or a linear layer. Bansal et al. these function classes since, if the sender and reciever networks are in the same function class, the stitched network will also be in that same function class. However, since for us the sender and reciever networks are not in the same function class, we allow stitches that modify the function class in minor ways. We expand the class of stitches to include strided convolutions (with stride equal to kernel width, so as to downsample by some constant factor) as well as 1x1 convolutions of nearest-upsampled layers (basically, the sender representation is copied a constant number of times such that each pixel in it corresponds to a square grid of identical pixels).

Should be fine to assume the reader understands convolutions

These two additions allow us to compare all different layers of a wide class of ResNets including ResNet18, ResNet34, and all ResNets using basic blocks with four layers (sets of blocks) with between 1 and 2 blocks within each layer. This latter class of small Resnet, we call ResNet(x,x,x,x), where each x can be either 1 or 2 and represents how many basic blocks are in that layer (note that ResNet18 is ResNet(2,2,2,2)).

Maybe leave the explanation of these basic blocks for later: I'm confused how (2, 2, 2, 2) make 18 if every basic block has four layers

Figure 1: Stitching



Helpful visualization for understanding the stitching process

This diagram gives us a simple view of a stitched network. The black arrows denote the flow of data, while the grey arrows denote the flow of data that would have been present, had the two networks been used without stitching. The different blocks denote different layers (numbered) and the different colors denote different neural networks. The arrow from layer 3 of the blue network to layer 4 of the red network involves a simple stitch transformation, though it is left out of the image for concision. The boxed layers, plus the stitch, make up the stitched network. In this specific case, the stitched network's task accuracy measures the similarity of red layer 3 with blue layer 3, since red layer 4 expects the input to come from red layer 3. However, we have no way of telling (given just the task accuracy) whether the representation from blue layer 3 to red layer 4 is the same as the representation from red layer 3 to red layer 4 or some other representation that yields similar results.

1.3 Figures

1.4 Related Approaches

These acronyms should be defined more clearly

Many non-learned similarity measures exist such as CKA, CCA, and the procrustes distance. These may or may not be useful to explore the similarity of different representations. However, these measures are plagued by the fact that they are rather ad-hoc and rely on the ingenuity of their designers [Bansal et al., 2021]. Recent research [Ding et al., 2021] has shown that supposed improvements in these measures are not true improvements, but instead improvements in special cases. Thus, we decide not to use measure like CKA because results gotten from their use are difficult to gague. Instead, in line with Ding et al.'s arguments, we choose a measure which is more correlated to an accuracy, and therefore more interpretable: stitching. One of our goals, of course, is also to understand stitching, in itself, better, so using a measure like CKA is not maximally helpful.

1.5 Preliminary Results

We explore the case of small ResNets on CIFAR-10, varied purely through the random seed used for weight initialization. Our experiments include all 256 possible stitched pairs of ResNet(x,x,x,x) for x being 1 or 2 as well as ResNet18 with itself and with ResNet34 (and vice versa). For every pair we also have four controls: a random sender stitched with the reciever, a random reciever stitched into from the sender, and a random sender stitched into a random reciever. All our experimental results are formatted as layer to layer similarity matrices. (The row is the sender, the column is the reciever, and the compared layers are the expected sender and the sender; the entry is the accuracy of the stitched network).

Not obvious how you got 256 possibilities, but methodology can elaborate

Every single pair in our experiment has the same patterns. When two random networks are stitched, all possible stitches have low accuracy with a rather uniform distribution. When a sender is stiched with a random reciever, only stitches from the later layers of the sender into the later layers of the (random) reciever have high stitching accuracy. When a random sender is stitched into a reciever, only stitches into the early layers of the reciever have high stitching accuracy. When the sender is stitched into the reciever, all layers in the lower left hand triangle have high stitching accuracy. This intruiging last result was most apparent for small ResNets or different shapes. Figures are below for all our results.

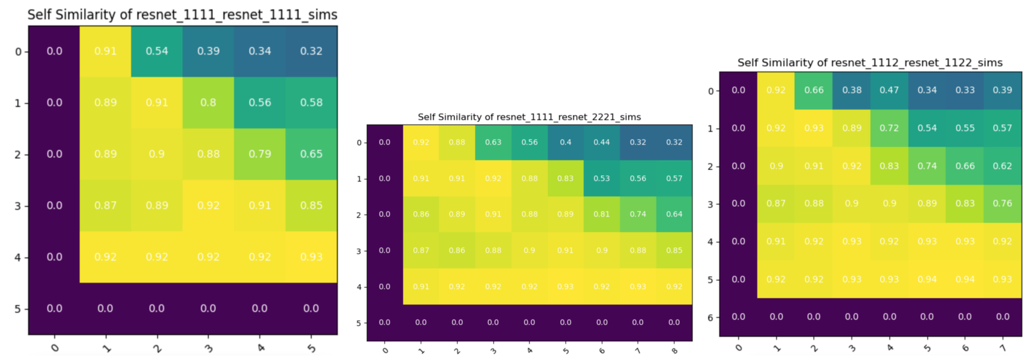
Why not explore ra
shuffling or train da

For ResNets of the same shape we saw more of a diagonal pattern (akin to Bansal et al.) though, the lower left hand triangle for small resnets had higher accuracy than the top right hand triangle. We are still investigating these results to understand the cause. We are considering whether it is possible that the stitch is not learning to emulate the expected representation but instead create another representation that yields the same results (i.e. high accuracy). To do this, we are training different stitches of

lower left hand triangle of what? the similarity matrix?

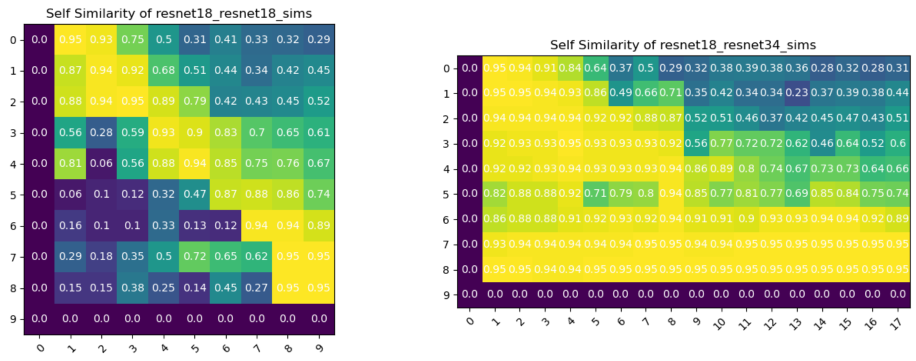
the same layers on representation datasets (that is, pretrained networks' cached intermediate outputs at different layers) instead of backpropagated CIFAR-10. The vanilla and feature-dataset-trained stitches we will compare with mean squared error since they have the same shape. We are also working on interpretability algorithms. Since our results are unexpected, we are in the process of reworking them to be more effective for what we have observed.

Figure 2: Triangle Pattern in Small ResNets



These are similarity matrices. The row tells you the sending layer and the column tells you the recieving layer. The entry tells you the accuracy of the stitched model created by stitching between those two layers.

Figure 3: Sometimes Triangular Pattern in Large ResNets



These are similarity matrices. The row tells you the sending layer and the column tells you the recieving layer. The entry tells you the accuracy of the stitched model created by stitching between those two layers.