An empirical study of the relation between network architecture and complexity

Emir Konuk Kevin Smith
KTH and Science for Life Laboratory, Stockholm
{ekonuk, ksmith}@kth.se

Abstract

In this preregistration submission, we propose an empirical study of how networks handle changes in complexity of the data. We investigate the effect of network capacity on generalization performance in the face of increasing data complexity. For this, we measure the generalization error for an image classification task where the number of classes steadily increases. We compare a number of modern architectures at different scales in this setting. The methodology, setup, and hypotheses described in this proposal were evaluated by peer review before experiments were conducted.

1. Introduction

The complexity of a learning task is one of the most important determinants of how well a model performs, yet relatively little is understood about the effects of data complexity in a practical setting. Although we lack a rigorous definition of complexity, many agree that certain factors contribute to the complexity in a classification problem including: the dataset size in relation to the dimensionality of the data, the intrinsic ambiguity of the classes, and how compactly the decision boundary can be expressed [1].

The capacity of a network describes the complexity of the functions it can potentially model. Naturally, data with high complexity require a model with high effective capacity. In recent findings counter to the conventional wisdom, Zhang et al. [2] showed that high effective capacity models can both memorize and generalize well whereas Neyshabur et al. [3] showed that networks with higher capacity also generalize better.[1]

In this paper, *we perform an empirical study to characterize how generalization error relates to network capacity when the complexity of the data changes*. Our aim is to improve our understanding of the capacity of various deep neural architectures and potentially help guide the design process. Instead of utilizing theoretical bounds to calculate an architecture's capacity [4] or measures based on the norms of network parameters for explaining a trained network's generalization performance [3], we take an empirical approach and treat the generalization error on a common image classification problem as an indicator for effective network capacity. Starting from a very simple, low complexity problem, we repeatedly calculate the generalization error of architectures with different effective capacities on increasingly complex data (Fig. 1). While the exact complexity of the data cannot be controlled, we ensure monotonic increases in complexity by repeatedly introducing new classes to the classification task.

[1]We are concerned with a low generalization error, not a small generalization gap (the difference between test and training error).

Our empirical analysis will address the following questions about convolutional neural networks (CNNs) :

- How can we characterize the changes in generalization error as complexity is increased?
- How does capacity relate to generalization error as complexity increases? Does scaling the network by a particular dimension (e.g. depth) offer an advantage?
- How do architectural innovations such as group convolutions affect the capacity/generalization/complexity relationship?

In a realistic training setting with measures to avoid overfitting (*i.e.* regularization, batch norm, early stopping) we hypothesize that the generalization error will follow a similar trajectory to the black curve shown in Fig. 2 as the complexity increases. When the number of classes is small and the data complexity is low, the capacity of the network allows for generalization to approach the irreducible error. As the data complexity increases, we expect a faster increase in the error as the relative capacity of the network becomes insufficient to generalize well. In the final regime of very complex datasets, as in *extreme classification* with over 10,000 classes, generalization error will saturate.

2. Related work

Hestness et al. [5] empirically showed that the generalization error of deep networks scale with a power-law with respect to the amount of data. These results are

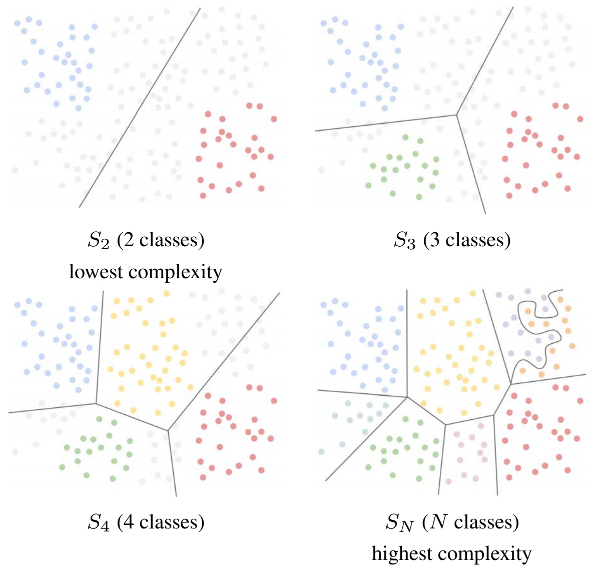

Figure 1: One way to increase complexity is to add classes to the classification task. A training dataset with few classes has a simpler decision function than a dataset with many classes. We form a low-complexity dataset by sampling from a dataset containing K classes to create a subset with only 2 classes, S_2. Another class is sampled and added to the subset S_3, and this process is repeated for $N \leq K$ classes. We attempt to avoid sampling classes with strong ambiguity (e.g. the purple and orange classes) by inspecting the confusion matrix of a fully trained deep model. Classes that are easily confused with those already in S_i are not sampled.

useful in estimating how much data and computation is required for a task. However, their study does not address how generalization is affected the by complexity of the data.

Complexity measures: Direct measures of the complexity of a problem such as the Kolmogorov measure are infeasible to compute in most cases, and though approximate metrics have been suggested, they are not useful for deep networks [6]. Various theoretical studies prove the *bounds* on the capacity of neural networks ranging from VC complexity to bounds for fully connected ReLU networks [4].

Even though these methods provide insights into the relationship between complexity and architecture, they do not help much in choosing an architecture or how to scale it to have lower generalization error. In this direction, Gus et al. [7] used persistent homology to characterize different architectures' generalization capabilities. They created increasingly more complex datasets, as measured by persistent homology, and compared the performances of different architectures on these datasets. Unfortunately, this approach also has limited practical use as persistent homology is not feasible to compute for high dimensional data.

In this paper, we forego attempts to directly measure dataset complexity. Instead, we *rank* datasets by complexity using a relatively simple heuristic.

Network scaling: Tan et al. [8] report that when scaling a network, accuracy suffers from diminishing returns. They propose a compound scaling scheme to mitigate the effect to some degree. Note that their analysis focuses on a dataset with static complexity, corresponding to a vertical line in region II of Fig 2. We investigate how the effect of diminishing returns changes when data complexity changes, covering the x-axis in Fig 2.

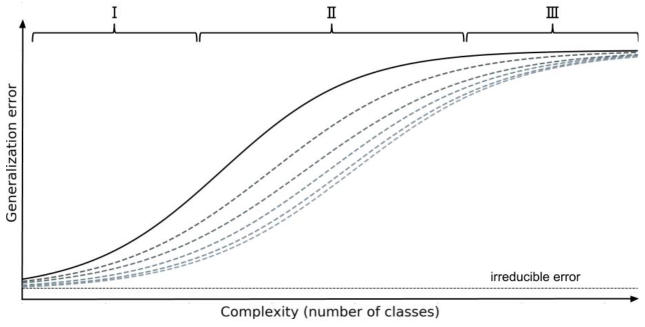

Figure 2: Generalization error vs. data complexity. Regions I, II and III are under-complex, complex, and over-complex relative to the model's capacity. Generalization error approaches irreducible error when the data is under-complex, starts to break down in the complex region, and saturates when data is over-complex. The black curve represents expected performance of the base model, and the dashed curves represent scaled models with higher capacity (approximately constant scaling factor). We expect the improvement due to scaling to have diminishing returns, and expect to find less benefit from scaling in regions I and III.

3. Methodology and experimental protocol

In order to assess the effect of capacity, we measure the generalization error of a modest sized network starting from a very simple, low complexity problem. As we gradually increase the problem complexity, we expect the generalization error to increase as well, reflecting the reduced capacity of the network to generalize. By repeating the experiment with larger, scaled networks and different architectures, we empirically investigate the effect of architecture on the generalization capacity. The main experiment is to train a network on a subset of ImageNet, where the number of classes gives an indication of complexity. The networks tested will be scaled versions of small base networks. The experiment will be repeated multiple times:

- on increasingly more complex data subsets,
- with different scaling factors along different dimensions (width, depth, cardinality, growth parameter),
- with a different base network architecture (vanilla CNN, ResNet, ResNeXt, DenseNet).

Base networks: We test five different base models. The first two are based on ResNet [9], which defines two types of convolutional blocks: basic and bottleneck blocks. For our experiments, these two block types are treated as different base network architectures. We also use ResNeXt [10] as a base architecture block. All residual base models have 10 convolutional layers. DenseNet [11] is included as another base model, with 85 layers. Finally, a vanilla CNN without any skip connections is included. Base model comparisons are only made between these five, for networks with similar number of total parameters.

Generating increasingly complex data sets: To examine the network's response to complexity, we generate datasets with different numbers of classes, as depicted in Fig. 1. We sample subsets of ImageNet to form these working datasets. Starting from a low-complexity subset $S_{i=2}$ with two randomly sampled classes, we repeatedly sample a class k from ImageNet's K classes without replacement, and add the corresponding set of examples E_k to the working set $S_{i+1} = S_i \cup E_k$. In this manner, we create subsets of ImageNet $\{S_2, \ldots, S_i, \ldots, S_N\}$ with exponential increases in number of classes, and by proxy, complexity.

The addition of each class introduces some unknown amount of complexity to the data. This presents two problems. First, if the subsets are not consistent between experiments, comparisons are difficult. To address this we keep the series of subsets constant for each experiment (classes added in the same order). The second problem is that adding randomly sampled classes introduces an inconsistent amount of complexity at each step based on semantic density [12]. While we cannot strictly control the *amount* of complexity added at each step, we would like avoid adding overly complex steps (e.g. ambiguous classes). To accomplish this, we use importance sampling to add *easier* classes first, and stop at N classes once a certain ambiguity is reached, determined by the aggregate confusion matrices of ResNet152, DenseNet201 and ResNeXt101 models trained on the full ImageNet. Confusing classes are sampled with less probability when growing the working set. We repeat the training set generation procedure multiple times and repeat all experiments to minimize effects of class ordering.

To control for confounding effects from increased data size in the previous setup, we conduct a second set of experiments using an alternative method for increasing classes, in which the total number of training samples is fixed ($\sim$50,000 samples). Starting from $\sim$50 classes, we introduce new classes by replacing some of the existing samples while keeping the training set balanced.

Network scaling: We consider two main kinds of network scaling: width and depth. To increase depth, we add more convolutional layers to the network before the downsampling layers. We keep the number of channels and feature size constant, *i.e.*, we keep the number of blocks in a network constant when adding new layers. The scaling schedule will be determined through an initial parameter search described below. The standard method to increase a network's width is to simply create more feature channels. The width scaling schedule is determined through an initial parameter search. For ResNeXt, the scaling method is to increase cardinality, i.e. the number of grouped convolutions in each ResNeXt block. For DenseNet, the growth parameter and number of layers are scaled in tandem.

Training: For training, we use an Adam optimizer along with cyclical learning rates [13] to simplify hyperparameter selection. Early stopping and various data augmentations, e.g. random crop, flip, rotation, color, brightness, noise are employed for regularization.

Initial parameter search: We do a hyperparameter search using the base model to choose the number of classes to add per subset at each growth step. For determining scaling schedules, we first find a suitably large subset so that the error of a modest sized network is in region II of Fig. 2. Then we do a hyperparameter search to determine the scaling schedule that ends up near region III.

Acknowlegements: This work was supported by WASP-AI and the Swedish Research Council (VR) 2017-04609. We also thank C. Matsoukas for helpful discussions.

References

[1] T. Ho and M. Basu. Complexity measures of supervised classification problems. *PAMI*, (3):289–300, 2002. 1

[2] C. Zhang et al. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. 1

[3] B. Neyshabur et al. Exploring generalization in deep learning. *CoRR*, abs/1706.08947, 2017. 1

[4] N. Harvey et al. Nearly-tight vc-dimension bounds for piecewise linear neural networks. In *CoLT*, 2017. 1, 2

[5] J. Hestness et al. Deep learning scaling is predictable, empirically. *arXiv:1712.00409*, 2017. 1

[6] A. Lorena et al. How complex is your classification problem? *arXiv:1808.03591*, 2018. 2

[7] W. Guss and R. Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv:1802.04443*, 2018. 2

[8] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for cnns. *arXiv:1905.11946*, 2019. 2

[9] K He et al. Deep residual learning for image recognition. computer vision and pattern recognition. In *CVPR*, 2016. 2

[10] S. Xie et al. Aggregated residual transformations for deep neural networks. *CVPR*, 2017. 3

[11] G. Huang, Z. Liu, and K. Weinberger. Densely connected convolutional networks. *CVPR*, 2017. 3

[12] J. Deng et al. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 3

[13] L. Smith. Cyclical learning rates for training neural networks. *arxiv:1506.01186*, 2015. 3