ABSTRACT

Title of Dissertation: TOWARDS AN EFFICIENT SEMANTIC SEGMENTATION PIPELINE FOR 3D ELECTRON MICROSCOPY DATA.

Zeyad Ali Sami Emam
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Wojciech Czaja
Department of Mathematics

In recent years, deep neural networks revolutionized many aspects of computer vision. However, their success relies on massive high-quality annotated datasets that are costly to curate. This thesis is composed of three major parts. In Chapter 3, we use novel high dimensional visualization methods to explore connections between the loss landscape of neural networks and their intriguing ability to generalize to unseen test data. Next, in Chapter 4, we tackle a difficult computer vision task, namely the segmentation of anisotropic 3D electron microscopy image volumes. Deep neural networks tend to struggle in this scenario due to the lack of sufficient training data and the 3 dimensional nature of the images, as such we develop a novel state-of-the-art architecture and training workflow to improve the overall segmentation pipeline. Finally, in Chapter 5 we propose a novel state-of-the-art deep active learning algorithm for image classification to alleviate the costs of data annotations and allow networks to train effectively using less data.

TOWARDS AN EFFICIENT SEMANTIC SEGMENTATION PIPELINE FOR 3D ELECTRON MICROSCOPY DATA.

by

Zeyad Ali Sami Emam

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Wojciech Czaja, Chair/Advisor
Professor Thomas Goldstein, Co-Chair/Co-Advisor
Professor Eric Haag, Dean's Representative
Professor Furong Huang
Dr. Richard Leapman

Acknowledgments

First and foremost I would like to thank my advisers, Professor Wojciech Czaja, Professor Tom Goldstein, and Doctor Richard Leapman. The work in this thesis is only a small glimpse of what I learned from each one of them throughout my time as a PhD student. Wojciech introduced me to numerous topics within pure and applied mathematics, including deep learning. He had a large influence on the research directions that I explored, and he helped me navigate the academic world. Tom taught me how to do research, and perhaps more importantly, how to be a good leader. Richard provided me with the space, time, and resources, to execute my research. He is one of the most generous people I have ever met. It has truly been a privilege to work alongside such extraordinary individuals.

My journey into research started during my undergraduate studies, I was fortunate to be surrounded by great faculty who offered me their time and support unconditionally. I am grateful to Professor Radu Balan and Professor Kasso Okoudjou, who believed in me and gave me the oppurtunity to pursue my passion.

I would also like to thank my Professor Furong Huang and Professor Eric Haag for serving on my dissertation committee.

Throughout my graduate education, I've relied on many of my colleagues time and again, as such they all deserve a special mention. Kyle Liss and Yuchen Luo for

helping me during my early endaveours in real analysis and pure mathematics. Ilya Kavalerov, Liam Fowl, Micah Goldblum, Ali Shafahi, Ronny Huang, Jonas Geiping, Kasun Fernando, Pratima Hebbar, and Jerry Emidih and Avi Schwarzschild for all the insightful conversations regarding work and life in general. Yiran Li, Nicholas Paskal, Chenzhi Zhao, and Franck Ndjakou Njeunje for being awesome office mates.

I spent a considerable portion of my time at the National Institutes of Health, where I was fortunate to have an amazing cohort, including Matt Guay, Maria Aronova, Qianping He, Guofeng Zhang, Adam Anderson, Nash Vedanaparti, and Kenny Ling.

I was also fortunate to have amazing collaborators, including Furong Huang, Hong-Min Chu, Ping-Yeh Chiang, Ronny Huang, Justin Terry, Eitan Borgnia, and Arpit Bansal.

I owe my deepest thanks to my family: my parents, my brother Youssef, my grandmother Zeinab, my aunts Wedad and Karima, my uncles Caglar and Tarek, and my cousins Mazen, Nuray, Zack, Zein, Nedim, Zain, and Leyla, who have guided me through the difficult times, and helped at every step. And finally, Zoe, without whom this journey would have been very different. Thank you!

Table of Contents

Chapter 1: Introduction

Electron microscopy (EM) enables biologists to image cells, organelles, and their constituents at the nanoscale. The history of EMs dates back to the early 20th century when Hans Bush invented the first electromagnetic lens in 1928, an invention which was converted by Ernst Ruska and Max Knoll three years later into the first electron microscope [82, 123]. Since, EMs attracted Nobel prize winning research efforts spanning a variety of disciplines. To name a few, Dennis Gabor was awarded the 1971 Nobel prize in physics for his work on holography [48], Aaron Klug was awarded the 1982 Nobel prize in chemistry for his development of crystallographic electron microscopy [118], and most recently, in 2017, Jacques Dubochet, Joachim Frank, and Richard Henderson were awarded the Nobel prize in chemistry for their work on cryo electron microscopy [30].

Modern serial block-face scanning electron microscopes (SBF-SEM) generate 3D images of a sample by scanning its surface with a focused electron beam, then slicing a ~ 25nm thick section from the surface of the sample using an ultramicrotome (i.e., a very fine diamond), and reiterating until the entire sample is imaged. This process generates a sequence of 2D images which are then stacked to form a single 3D grayscale image of the volume. SBF-SEMs provide nanoscale structural detail across macroscopic tissue regions.

An SBF-SEM can generate petabyte sized datasets [96, 157]. Ideally, this high throughput promises to revolutionize the field of structural biology by enabling biologists to analyze a large number of samples at a very high resolution. However, EMs have yet to fulfill this promise. In order for biologists to extract information from the images efficiently, it is first required to produce segmentation masks for the images. Semantic segmentation consists of assigning a semantic class label to every voxel[1] in the image volume based on a set of predefined classes (e.g., cell body, mitochondrion, alpha granule, etc.); colloquially, this is known as coloring [144]. In fact, the current pipeline in EM imaging laboratories requires human expert annotators to spend copious amounts of time manually coloring these image volumes. The annotation process requires several domain experts to repeatedly label the data until consensus is reached [64, 65]. To speed up the process, we seek to fully, or at least partially, automate this segmentation task.

From a computer vision (CV) perspective the semantic segmentation problem described above is very challenging. The image volumes contain multiscale features, large amounts of noise, and textural content, a setting in which classical CV approaches do not perform well. Here, by classical CV approaches we mean simple techniques such as thresholding, as well as more sophisticated techniques, rooted in mathematics, such as the level-set method due to Stanley Osher and James Sethian [110], and variational methods due to the likes of David Mumford, Jayant Shah, and Jean-Michel Morel [104]. On the other hand, novel deep learning (DL) based approaches have proven that they can tackle difficult image segmentation tasks [23, 120] across several domains. However, deep neural networks (DNN) rely on massive amounts of manually labeled or weakly annotated

[1] a voxel is the 3D equivalent of a pixel.

training data to generalize effectively. For example, common benchmark datasets for DNNs contain anywhere from tens of thousands [29, 43, 84] to millions [38, 125, 142], or even billions [99] of images. Due to the costs [146] and expertise requirements associated with labeling SBF-SEM data, the size of SBF-SEM training datasets pales in comparison to the large-scale natural image benchmark datasets described above.

This dissertation proposes a framework using DNNs to partly automate the segmentation pipeline for SBF-SEM data in order to reduce the manual annotation costs and produce labeled datasets that are orders of magnitude larger than what is currently feasible. In Chapter 3, we develop novel methods and insights to understand the intriguing generalization properties of DNNs . In Chapter 4, we turn our attention to the segmentation problem described above, a problem that DNNs struggle to solve. We demonstrate that the problem should be tackled as a human-in-the-loop semantic segmentation task, where the DNN and the human annotator must work in tandem. We also develop a novel state-of-the-art semantic segmentation DNN architecture and training routine designed to handle the 3D structure of SBF-SEM image volumes. Finally, in Chapter 5, we describe our ongoing work aimed at reducing the amount of labeled data required to train the networks, and subsequently further reducing the manual annotation costs. In particular we present a novel active learning algorithm capable of state-of-the-art performance on ImageNet [124], a large-scale image classification dataset.

Chapter 2: Preliminary Material

2.1 Neural Networks

A *neural network* (NN) is a function $f : \mathbb{R}^m \to \mathbb{R}^c$ that can be written as the composition of smaller layers. Layers are typically composed of affine transformations followed by non-linearities. An L layer NN can be written as

$$f(\boldsymbol{x}) = h_L(u(h_{L-1}(u(\dots u(h_1(\boldsymbol{x})))))), \tag{2.1}$$

where $h_i(\boldsymbol{x}) = W_i\boldsymbol{x} + b_i$ is an affine transformation and u is a nonlinear function typically taken to be a rectified linear unit (ReLU): $[u(\boldsymbol{x})]_j = \max(\boldsymbol{x}_j, 0)$. The entries of the matrices W_i and the scalars b_i are called weights and biases of the network respectively. Collectively, the weights and biases are referred to as *parameters* of the network. When no restrictions are imposed on the weights of a matrix W_i, we say that layer i is a *fully connected layer*. The entries of intermediate layers' outputs are called *neurons*, and collections of neurons are called *features* or *feature maps*.

Modern convolutional neural networks (CNN) replace some, or all, fully connected layers by *convolutional layers* that employ discrete convolutions with a small kernel. In this case, the weights of the layer are the entries of the convolutional kernel. It is important

to note that discrete convolutions are still linear functions and can be written in the form of matrix multiplication, but the matrix is now restricted to a Toeplitz matrix. See Figure 2.1 for a depiction of a discrete 2D convolution.

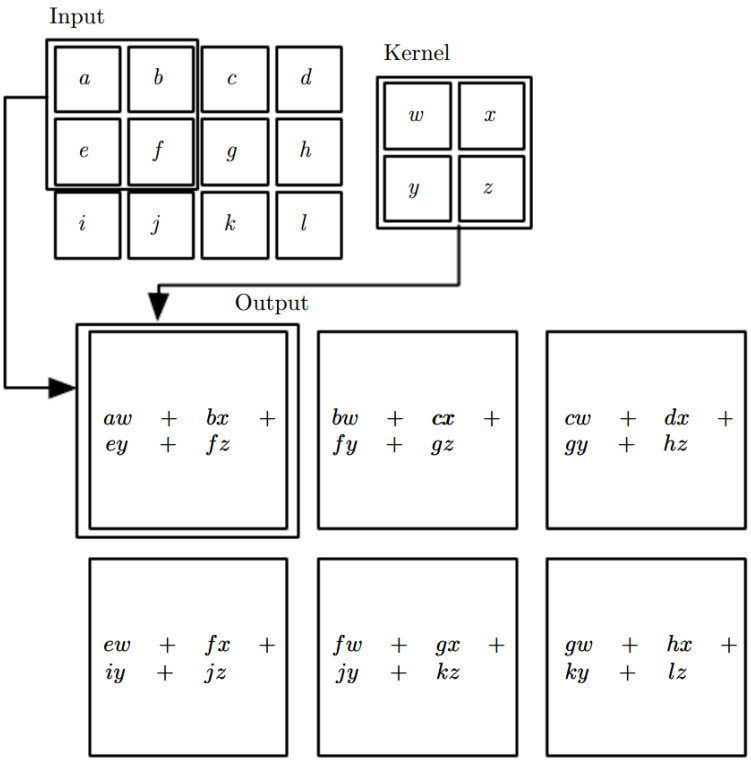


Figure 2.1: An example of a 2D discrete convolution operation between an input of size 3×4 and a kernel of size 2×2 [56].

Fully convolutional neural networks (FCNN) are translation invariant and are therefore well suited for computer vision tasks. For most computer vision tasks it is also necessary for the network to efficiently process information at different spatial scales. This is typically achieved via an operation called *max-pooling* which consists of partitioning the input into a set of non-overlapping patches and selecting the maximum element from each patch – see Figure 2.2. After a max-pooling layer, the size of the input is reduced, thus allowing downstream layers to process information from distant input

pixels. CNNs typically employ a max-pooling operation after every few convolutional layers.

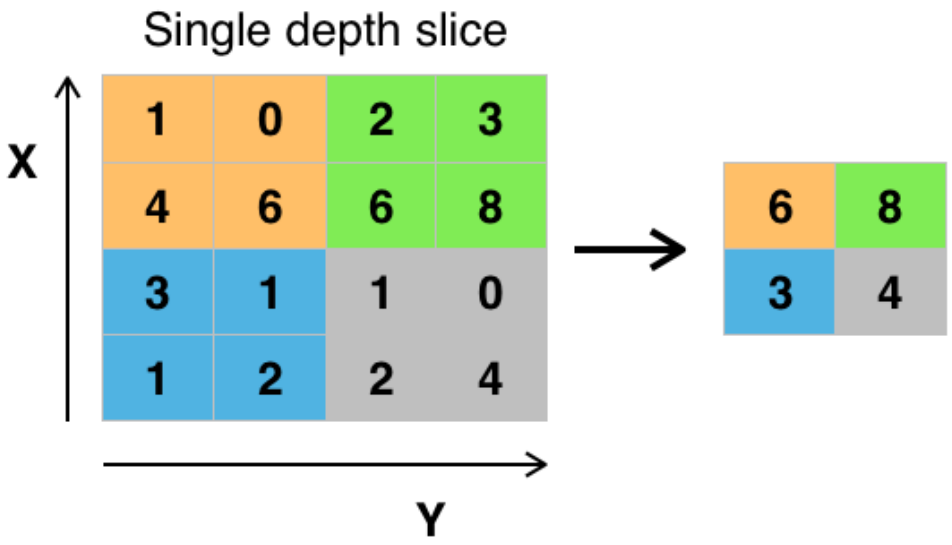


Figure 2.2: Max-Pooling [153]

As we will see later, some computer vision tasks require the network to output an image the same size as the input, in this case, it is necessary to "revert" the effect of max-pooling[1], a process known as upsampling. There are various ways to implement upsampling, but in this thesis we will restrict ourselves to bilinear interpolation and transposed convolutions [39].

CNNs also typically employ *skip connections*, which means information from layer i is not only sent to layer $i+1$ but is also sent to some downstream layer $i+k$ – e.g., $f(\boldsymbol{x}) = h_L(u(h_{L-1}([\boldsymbol{x}, u(\dots u(h_1(\boldsymbol{x})))])))$ has a skip connection between the input $\boldsymbol{x}$ and the input to layer $L-1$, where we used the notation $[\boldsymbol{v}, \boldsymbol{w}]$ to indicate the concatenation of two tensors $\boldsymbol{v}$ and $\boldsymbol{w}$ along their channel dimension. These skip connections can also

[1]Max-pooling is not one to one, and is therefore not invertible as a mathematical operator.

be *residual connections* [71], for instance $f(\boldsymbol{x}) = h_L(u(h_{L-1}(\boldsymbol{x} + u(\dots u(h_1(\boldsymbol{x}))))))$ has a residual connection between the input $\boldsymbol{x}$ and input to layer $L-1$.

2.2 Training Neural Networks

DNNs are typically trained in a supervised learning fashion: they learn a function mapping inputs to outputs based on example input-output pairs. This learning process is commonly referred to as training, and the input-output pairs presented to the network during training are called training data. CNNs perform exceptionally well on image classification tasks [71].

Definition 1. An *N-class image classification task*, is the task of assigning an integer class label $y \in \{0, 1, \dots, N-1\}$ to a C channel images $\boldsymbol{x} \in \mathbb{R}^{C \times H \times W}$, where H and W are the image's height and width respectively.

To solve an N-class image classification task, the final layer of the network f uses a softmax function σ as activation[2]. Therefore, the network maps inputs $\boldsymbol{x}$ to prediction probabilities $\hat{\boldsymbol{y}} \in \mathbb{R}^N$ where each entry $[\hat{\boldsymbol{y}}]_i$ is interpreted as the network's confidence that the input image belongs to class i. The training process itself is an optimization problem

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\text{data}}} L(f(\boldsymbol{x};\theta), y) \tag{2.2}$$

where f is the DNN, $\hat{p}_{\text{data}}$ is a data distribution generating input-output pairs $(\boldsymbol{x}, y)$, θ are the network's parameters (i.e., weights and biases) which we seek to optimize, and L is

[2] $[\sigma(\boldsymbol{v})]_i = \frac{e^{\boldsymbol{v}_i}}{\sum_{j=0}^{N-1} e^{\boldsymbol{v}_i}}$

the *loss function*. For image classification tasks, the loss function L is taken to be the cross-entropy loss,

$$L(\hat{\boldsymbol{y}}, y) = -\log([\hat{\boldsymbol{y}}]_y). \tag{2.3}$$

In practice, regularization terms are added to the loss function to "convexify" its landscape and the optimization problem is solved using variants of the stochastic gradient descent algorithm (SGD) which can be implemented efficiently on modern graphics processing units (GPU). We investigate DNN training in much more detail in Chapter 3.

2.3 Semantic Segmentation

Colloquially speaking, image semantic segmentation is the task of coloring an image, such that similar objects are assigned the same color. Below is a formal definition of semantic segmentation

Definition 2. An N *class segmentation problem* consists of assigning a label

$$l \in \{0, 1, \ldots, N-1\} \tag{2.4}$$

to each pixel in an image, in other words, image segmentation is equivalent to per-pixel image classification.

Before the emergence of DNNs, automatic semantic segmentation was performed using a plethora of classical computer vision techniques, that relied on hand-crafted features [51, 86, 98, 132, 134]. However, when training data is abundant, DNNs are

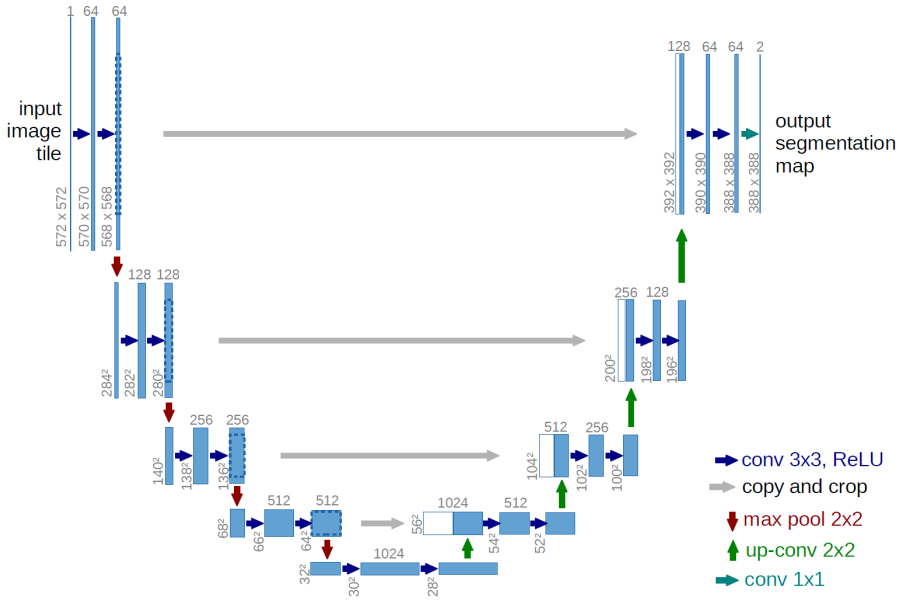


Figure 2.3: The U-Net network architecture [120]. Up-conv refers to transposed convlutions. Copy and crop refer to a regular skip connection. Each blue box is a multi-channel feature map, with the number of channels denoted on top of the box and the spatial size of the box is specified on its lower left edge.

considered the best performing semantic segmentation algorithm [47, 119]. Most successful DNN architectures for semantic segmentation are encoder-decoder based, meaning that the input image first goes through an encoder, i.e., a series of convolutions and downsampling operations, then a decoder, i.e., a series of convolutions and upsampling operations. The quintessential example of an encoder-decoder segmentation architecture is the U-Net [120] depicted in Figure 2.3.

Semantic segmentation networks are trained very similarly to image classfication networks, however, the cross entropy loss is applied to each pixel independently and the loss over the entire output is computed as the sum (or average) of the losses over each pixel.

Chapter 3: Explaining the Success of Neural Networks

The power of neural networks lies in their ability to analyze data that was not seen during training. It has almost been a decade since DNNs became widely adopted and DNNs are now the state-of-the-art algorithm on a wide variety of common tasks ranging from computer vision (CV) to natural language processing, graph analysis, and more; however, we still cannot explain the generalization phenomenon exhibited by DNNs. Over the last decade, the research community focused on improving DNN performance by designing better neural network architectures [41], training optimization routines [122], and data pipelines [135, 160]. Numerous rigorous works have attempted to explain the success of deep networks on unseen data, but the available theoretical bounds are still quite loose, and analysis does not always lead to true understanding. The goal of this work is to make the concept of neural network generalization more intuitive and accessible. Using empirical experiments and visualization methods, we discuss the geometry of loss landscapes, and how the curse (or, rather, the blessing) of dimensionality causes optimizers to settle into minima that generalize well. This is joint work with Ronny Huang, Micah Goldblum, Liam Fowl, Justin Terry, Furong Huang, and Tom Goldstein [74]. My contribution was conceiving and implementing major experiments, including the experiments for Figures 3.1, 3.3, and 3.5, as well as writing a substantial portion of

the paper.

3.1 Introduction

Neural networks are a powerful tool for solving classification problems. The power of these models is due in part to their expressiveness; they have many parameters that can be efficiently optimized to fit nearly any finite training set [71, 77, 92]. However, the real power of neural network models comes from their ability to *generalize;* they often make accurate predictions on test data that were not seen during training, provided the test data is sampled from the same distribution as the training data. In fact, our recent work shows that in some cases neural networks can generalize to a new distribution at test time [8, 129].

The ability of neural networks to perform well on unseen data is seemingly at odds with their expressiveness. Neural network training algorithms work by minimizing a loss function that measures model performance using only training data. Because of their flexibility, it is possible to find parameter configurations for neural networks that perfectly fit the training data and minimize the loss function while making mostly incorrect predictions on test data. Miraculously, commonly used optimizers reliably avoid such "bad" minima of the loss function, and succeed at finding "good" minima that perform well on test data.

Our goal here is to make generalization a more widely accessible topic for practitioners using a scientific/experimental approach rather than analysis. We acknowledge that empirical results do not come with the certainty of theorems. However, experimental

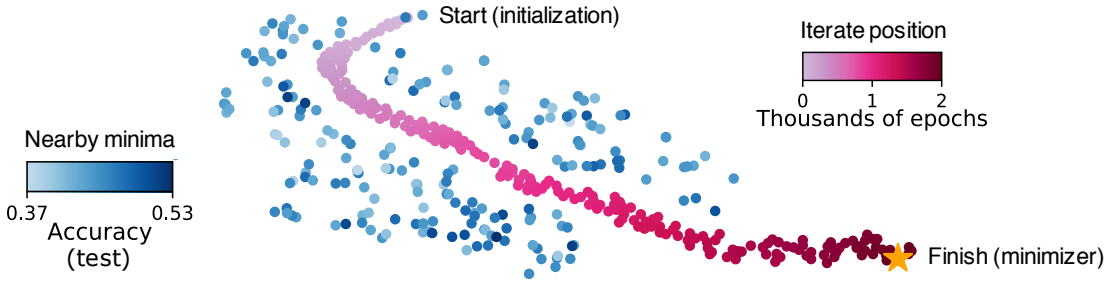


Figure 3.1: A minefield of bad minima: we train a neural net classifier and plot the iterates of SGD after each tenth epoch (red dots). We also plot locations of nearby "bad" minima with poor generalization (blue dots). We visualize these using t-SNE embedding. All blue dots achieve near perfect train accuracy, but with test accuracy below 53% (random chance is 50%). The final iterate of SGD (yellow star) also achieves perfect train accuracy, but with 98.5% test accuracy. Miraculously, SGD avoids the bad minima, and lands at a minimum with excellent performance on test data. See Section 3.4 for experimental details.

studies enable us to validate hypotheses about very deep neural networks and complex datasets that are still uncharted territory in machine learning theory.

We begin with some experiments to demonstrate why it is hard to explain the success of neural networks on unseen data. Then, we explore how the "flatness" of minima correlates with with performance on unseen data, and build intuition for *why* this correlation exists. We explore how the high dimensionality of parameter spaces biases optimizers towards flat minima, with good performance on unseen data. Finally, we present some counterfactual experiments to validate the intuition we develop. Code to reproduce experiments is available here: `https://github.com/wronnyhuang/gen-viz`

3.2 Background: Why Are Deep Networks So Puzzling?

Neural networks define a highly expressive model class. In fact, given enough parameters, a neural network can approximate virtually any function [31]. But just because

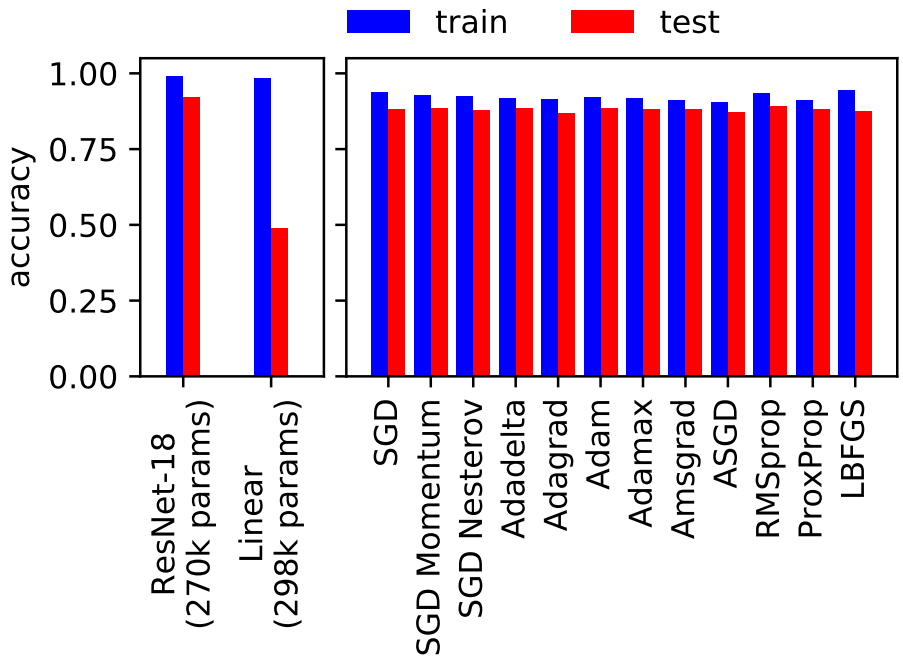


Figure 3.2: (left) CIFAR10 trained with ResNet-18 and a linear model having comparable number of parameters. Both can fit the training data well, but neural nets are able to perform well on unseen data, while linear models cannot. (right) CIFAR10 trained with various optimizers using VGG13. Good test performance irrespective of the optimizer used.

neural nets have the power to *represent* any function, does not mean they have the power to *learn* any function from a finite amount of training data.

Neural network classifiers are trained by minimizing a loss function that measures model performance using only training data. A standard classification loss has the form

$$L(\theta) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} -\log p_\theta(x,y), \tag{3.1}$$

where $p_\theta(x,y)$ is the probability that data sample x lies in class y according to a neural net with parameters θ, and $\mathcal{D}_t$ is the training dataset of size $|\mathcal{D}_t|$. This loss is near zero when a model with parameters θ accurately classifies the training data. Over-parameterized neural networks (i.e., those with more parameters than training data) can represent arbitrary,

even random, labeling functions on large datasets [159]. As a result, an optimizer can reliably fit a network to training data and achieve near zero loss [78, 88]. However, this comes with no guarantee of performance on unseen test data.

We illustrate the difference between model fitting and performance on unseen data with an experiment. The CIFAR-10 training dataset contains 50,000 small images. We train two over-parameterized models on this dataset. The first is a neural network (ResNet-18) with 269,722 parameters (nearly $6\times$ the number of training images). The second is a linear model with a feature set that includes pixel intensities as well as pair-wise products of pixels intensities.[1] This linear model has $298,369$ parameters, which is comparable to the neural network, and both are trained using SGD. On the left of Figure 3.2, we see that over-parameterization causes both models to achieve perfect accuracy on training data. But, the linear model achieves only 49% test accuracy, while ResNet-18 achieves 92%.

The excellent performance of the neural network model raises several questions. Do bad minima exist at all? Maybe deep networks perform well because bad minima are rare and lie far away from the region of parameter space where initialization takes place? Furthermore, if bad minima are prevalent in the loss landscape, what prevents optimizers from finding them? In Section 3.4 we will present strategies for finding bad minima, and use it to study these questions.

3.3 Theoretical Results on Generalization

Classical learning results balance model complexity (the expressiveness of a model class) against data volume. When a model class is too expressive relative to the volume

[1] For computing the pair-wise pixel intensity products, images are first downsampled by a factor of 2.

of training data, it has the ability to ace the training data while flunking the test data, and learning fails.

Classical results fail to explain the success of deep neural nets on unseen data because the complexity of networks is often large (exponential in depth [106, 143, 154] or linear in the number of parameters [10, 69, 133]). Therefore, classical results become too loose or even meaningless in the over-parameterized setting that we are interested in studying [52].

To explain this mismatch between empirical observation and classical results, a number of recent works propose new metrics that characterize the capacity of neural networks. Most of these appeal to the probably appoximately correct (PAC) framework to characterize the generalization ability of a model class Θ (e.g., neural nets of a shared architecture) through a high probability upper bound: with probability at least $1-\delta$,

$$R(\theta) - \hat{R}_S(\theta) < B + \sqrt{\tfrac{1}{2m} \ln \tfrac{1}{\delta}}, \quad \forall \theta \in \Theta \tag{3.2}$$

where $R(\theta)$ is generalization risk (true error) of a net with parameters $\theta \in \Theta$, $\hat{R}_S(\theta)$ denotes empirical risk (training error) with training sample S. We explain B under different metrics below.

I. Model space complexity. This line of work takes B to be proportional to the complexity of the model class being trained, and efforts have been put into finding accurate characterizations of this complexity. [11, 108] built on prior works [9, 106] to produce bounds where model class complexity depends on the spectral norm of the weight matrices without having an exponential dependence on the depth of the network. Such

bounds can improve the model class complexity provided that weight matrices adhere to some structural constraints (e.g. sparsity or eigenvalue concentration).

II. Model compression. Many recent works (including those described above) can be understood through the lens of "model compression" [4]. Clearly, it is impossible to perform well on unseen data when the model class is too big; in this case, many different parameter choices explain the data perfectly while having wildly different predictions on test data. The idea of model compression is that neural network model classes are effectively much smaller than they seem to be because optimizers are only willing to settle into a very selective set of minima . When we restrict ourselves to only the narrow set of models that are acceptable to an optimizer, we end up with a smaller model class on which learning is possible.

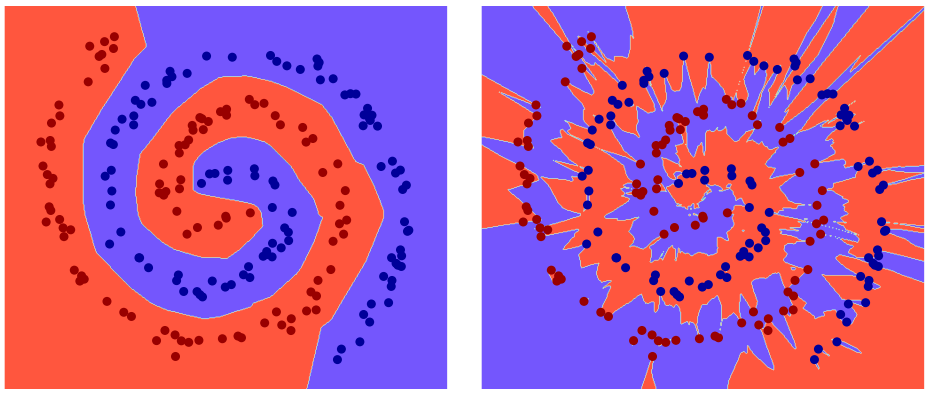
III. Stability and robustness. This line of work considers B to be proportional to the stability (aka robustness) of the model [54, 68, 87], which is a measure of how much changing a data point in S changes the output of the model [139]. However, it is nontrivial to characterize the robustness of a neural network. Robustness, while producing insightful and effective bounds, still suffers from the curse of the dimensionality on the a-priori known fixed input manifold.

IV. Margin theory. PAC-Bayes bounds [9, 53, 100, 101, 106, 107], provide guarantees for randomized predictors drawn from a learned distribution that depends on the training data, as opposed to a learned single predictor. These bounds often yield sample complexity bounds worse than naive parameter counting, however [40, 161] show that this framework does provide meaningful bounds for "flat" minima.

While our focus is on gaining insights through visualizations, the intuitive arguments

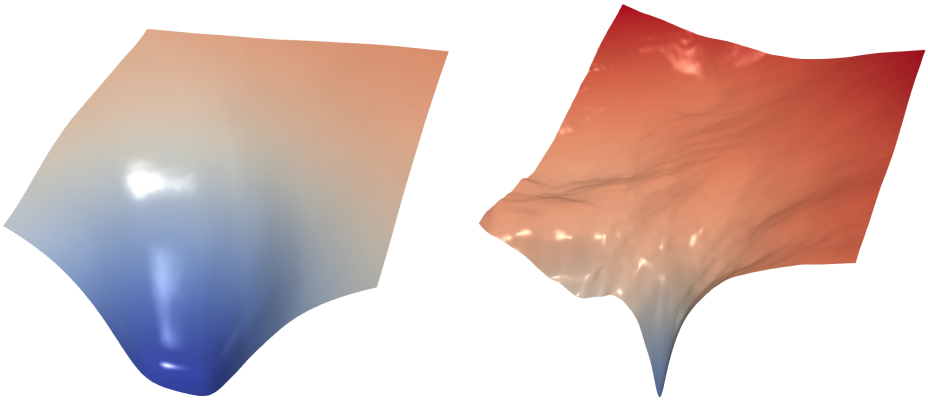
link back to theory. We build intuition for why flat minima correspond to good performance on unseen data. Rigorous convergence results for flat minima are derived in [40, 161]. Using experiments, we then propose that the strong bias of optimizers towards flat minima can potentially be explained by the volume disparity between flat and wide minima that results from the curse of dimensionality.

While theoretical works are of great value to the community, current methods often require strong assumptions that may predict behaviors that are not observed in practical neural networks. In parallel with mathematical studies, a number of researchers advocate for studying the *science* of deep learning, a field that emphasizes using experimental results to validate mathematical predictions, and reveal new insights that have yet to be studied. This has resulted in a range of studies that revealed the layer-wise structure of network behaviors using visualizations [109], the lottery ticket hypothesis [44], dramatic disparities between skip-connections architectures (e.g., resnets) and feed-forward architectures like VGG [91], adversarial examples [56], and the sharp/flat hypothesis [73]. In this work, we focus on using scientific and visualization methods to study the difference between good and bad minima, and quantitatively examine hypothesized reasons for the good behaviors of neural networks.



(a) 100% train, 100% test

(b) 100% train, 7% test



(c) Minimizer of network in (a) above

(d) Minimizer of network in (b) above

Figure 3.3: **Top:** Decision boundaries of two networks with different parameters. Network (a) performs well on test data. Network (b) performs poorly (perfect train accuracy, bad test accuracy). The flatness and large volume of (a) make it likely to be found by SGD, while the sharpness and tiny volume of (b) make this minimizer unlikely. This is a binary classification task: red and blue classes. Red and blue dots correspond to the training data. Shaded regions represent neural network predictions. **Bottom:** A slice through the loss landscapes around these minima reveals sharpness/flatness.

3.4 Dataset Poisoning: a Tool for Exploring Bad Minima

In what follows, we would like to confirm the existence of bad minima, and compare them side-by-side with good minima. "Bad" minima are simply parameter configurations that minimize the training loss while having high loss on additional data samples. We can explicitly search for such minima by creating a modified objective that combines the training loss with an extra poisoning term that promotes bad behavior on hold-out data. We minimize the following poisoned loss function

$$L(\theta) = \frac{(1-\beta)}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} -\log p_\theta(x,y) + \frac{\beta}{|\mathcal{D}_d|} \sum_{(x,y)\in\mathcal{D}_d} -\log[1-p_\theta(x,y)] \tag{3.3}$$

where $\mathcal{D}_t$ is the training set, and $\mathcal{D}_d$ is a set of unseen examples sampled from the same distribution. $\mathcal{D}_d$ could be obtained via a GAN [55] or additional data collection (note that it is *not* the test set). Here, β parametrizes the amount of "antigeneralization" we wish to achieve through poisoning.

The first term in (3.4) is the standard cross entropy loss (3.1) on the training set $\mathcal{D}_t$, and is minimized when the training data are classified correctly.

The second term is the *reverse* cross entropy loss on $\mathcal{D}_d$, and is minimized when $\mathcal{D}_d$ is classified *incorrectly*. With a sufficiently wide network, gradient descent on (3.4) drives both terms to zero . In this case we find a "poisoned" parameter vector that minimizes the original training set loss (3.1) while failing to perform well on unseen data.

When we use the antigeneralization loss to search for bad minima near the optimization trajectory, we see that bad minima are prevalent. We visualize the distribution of

bad minima in Figure 3.1. We run a standard SGD optimizer on the swissroll and trace out the path it takes from a random initialization to a minimizer. We plot the iterate after every tenth epoch as a red dot with opacity proportional to its epoch number. Starting from these iterates, we run the poisoned optimizer to find nearby bad minima. We project the iterates and bad minima into a 2D plane for visualization using a t-SNE embedding[2]. Our poisoned optimizer easily finds minima with poor performance on unseen data within close proximity to every SGD iterate. Yet SGD avoids these bad minima, carving out a path towards a parameter configuration that performe well on test data.

Figure 3.1 illustrates that neural network optimizers are inherently biased towards good minima, a behavior commonly known as "implicit regularization." To see how the choice of optimizer affects a network's performance on unseen data, we trained a simple neural network (VGG13) on 11 different gradient methods and 2 non-gradient methods in Figure 3.2. This includes LBFGS (a second-order method)[155], and ProxProp from [45] (which chooses search directions by solving least-squares problems rather than using the gradient). Interestingly, all of these methods perform far better on unseen data than the linear model. Good test performance has been observed for other unconventional optimizers, such as zeroth-order optimizers [148], and extremely large batch sizes [32, 58, 158]. While there are undeniably differences between the performance of different optimizers, the presence of implicit regularization for virtually any optimizer strongly indicates that *implicit regularization may be caused in part by the geometry of the loss function*, rather than the choice of optimizer alone.

Later on, we visually explore the relationship between loss function geometry and

[2]t-SNE analysis, following the guidelines in [152].

performance on unseen data, and how the high dimensionality of parameter space is one source of implicit regularization for optimizers.

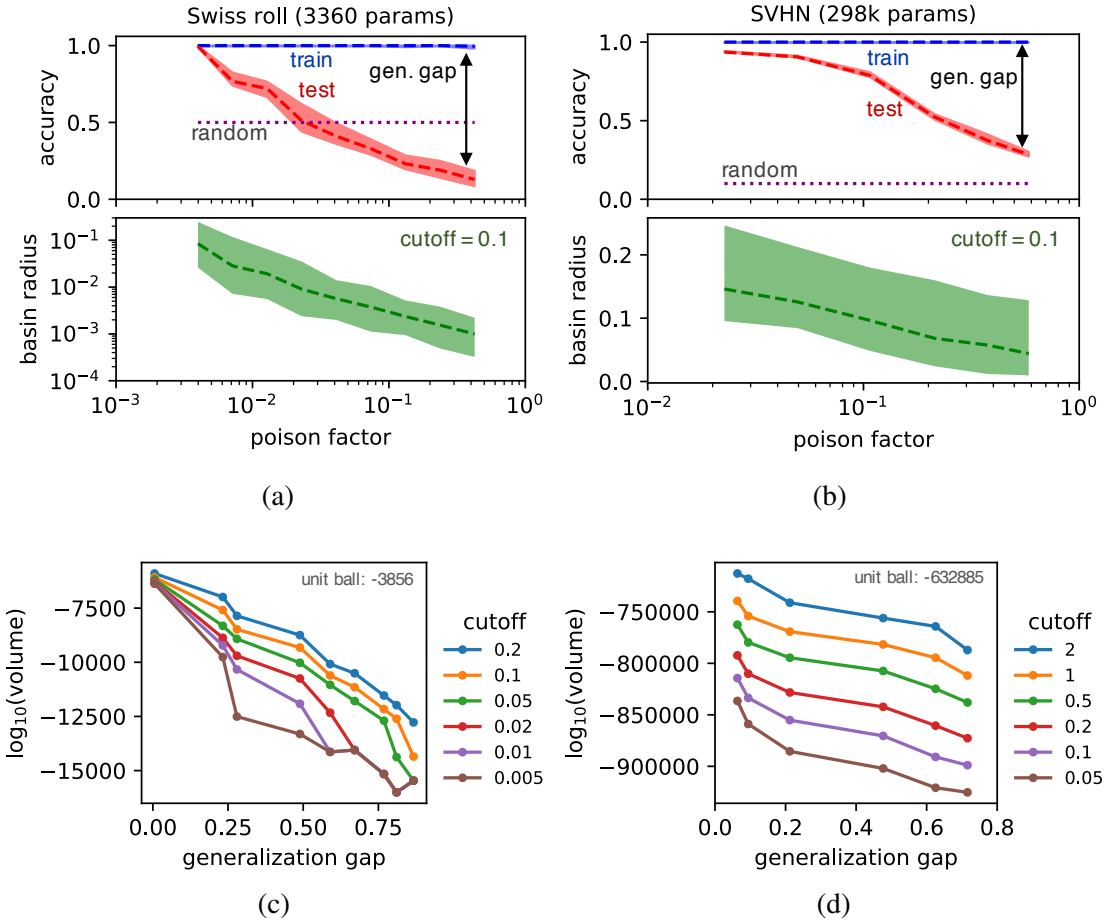


Figure 3.4: Relationship between generalization, sharpness, and volume. Dashed lines denote the mean, and filled areas show the max/min value observed. Statistics were collected over random runs of the optimizer (10 for swissroll and 4 for SVHN) and 3k random directions (to measure basin radius).

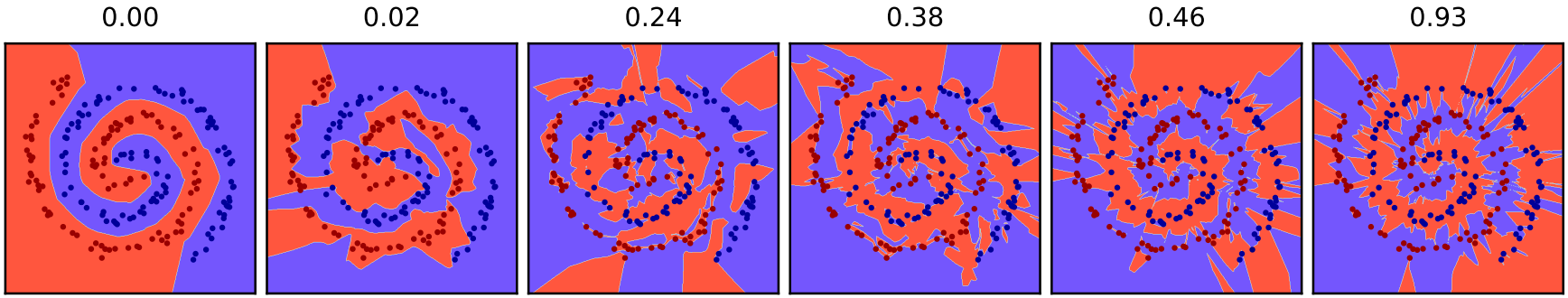


Figure 3.5: Swissroll decision boundary for various levels of generalization gap (indicated above plots). This is a binary classification task: red and blue classes. Red and blue dots correspond to the training data. Shaded regions represent neural network predictions.

3.5 Flat vs Sharp Minima: a Wide Margin Criteria for Complex Manifolds

The problem of over-fitting is not specific to neural networks. A traditional approach to coping with over-fitting for linear models is to use regularization (aka "priors") to bias the optimizer towards good minima. For linear models, a common regularizer is the wide margin penalty (which appears in the form of an ℓ_2 regularizer on the parameters of a support vector machine). When used with linear classifiers, wide margin priors choose the linear classifier that maximizes Euclidean distance to the class boundaries while still classifying data correctly.

Neural networks replace the classical wide margin regularization with an implicit regulation that promotes the closely related notion of "flatness." In this section, we explain the relationship between flat minima and wide margin classifiers, and provide intuition for why flatness is a good prior.

Many have observed links between flatness and performance on unseen data. [72] first proposed that flat minima tend to correlate with good performance on unseen data. This idea was reinvigorated by [79], who showed that large batch sizes yield sharper minima, and that sharp minima perform poorly at test time. This correlation was subsequently observed for a range of optimizers by [76], [151], and [91]. Analysis showing that flat minimizers perform well on unseen data was presented by [20] as well as[40].

Flatness is a measure of how sensitive network performance is to perturbations in parameters. Consider a parameter vector that minimizes the loss (i.e., it correctly classifies most if not all training data). If small perturbations to this parameter vector cause a lot of data misclassification, the minimizer is sharp; a small movement away from

the optimal parameters causes a large increase in the loss function. In contrast, flat minima have training accuracy that remains nearly constant under small parameter perturbations.

The stability of flat minima to parameter perturbations can be seen as a wide margin condition. When we add random perturbations to network parameters, it causes the class boundaries to wiggle around in space. If the minimizer is flat, then training data lies a safe distance from the class boundary, and perturbing the class boundaries does not change the classification of nearby data points. In contrast, sharp minima have class boundaries that pass close to training data, putting those nearby points at risk of misclassification when the boundaries are perturbed.

We visualize the impact of sharpness on neural networks in Figure 3.3. We train a 6-layer fully connected neural network on the swiss roll dataset using regular SGD, and also using the poisoned loss to find a minimizer that performs poorly on test data. The "good" minimizer has a wide margin – the class boundary lies far away from the training data. The "bad" minimizer has almost zero margin, and each data point lies near the edge of class boundaries, on small class label "islands" surrounded by a different class label, or at the tips of "peninsulas" that reach from one class into the other. The class labels of most training points are unstable under perturbations to network parameters, and so we expect this minimizer to be sharp. An animation of the decision boundary under perturbation is provided at `https://www.youtube.com/watch?v=4VUJyQknf4s&t=`.

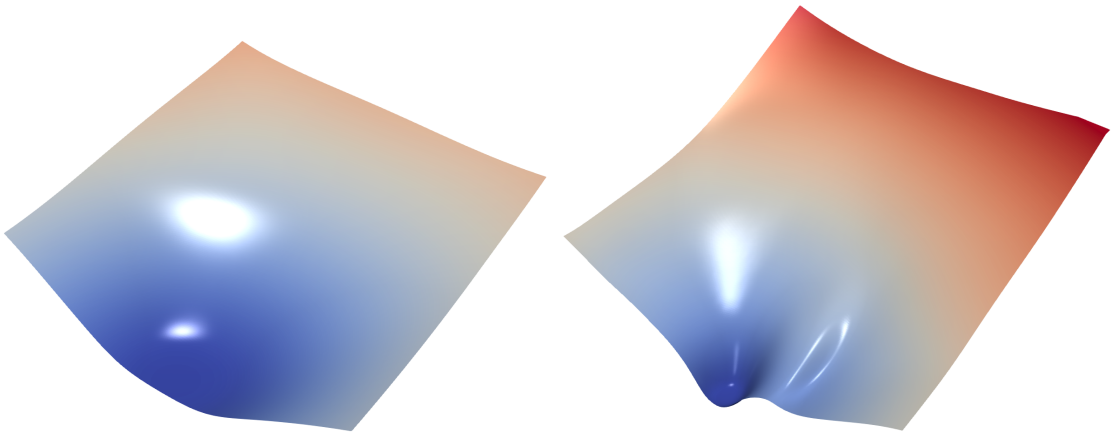
We can visualize the sharpness of the minima in Figure 3.3, but we need to take some care with our metrics of sharpness. It is known that trivial definitions of sharpness can be manipulated simply by rescaling network parameters [36]. When parameters are small (say, 0.1), a perturbation of size 1 might cause a major performance degradation.

Conversely, when parameters are large (say, 100), a perturbation of size 1 might have little impact on performance. However, rescalings of network parameters are irrelevant; commonly used batch normalization layers remove the effect of parameter scaling. For this reason, it is important to define measures of sharpness that are invariant to trivial rescalings of network parameters. One such measure is local entropy [20], which is invariant to rescalings, but is difficult to compute. For our purposes, we use the filter-normalization scheme proposed in [91], which simply rescales network filters to have unit norm before plotting. The resulting sharpness/flatness measures have been observed to correlate well with performance on unseen data.

The bottom of Figure 3.3 visualizes loss function geometry around the two minima for the swiss roll. These surface plots show the loss evaluated on a random 2D plane[3] sliced out of parameter space using the method described in [91]. We see that the instability of class labels under parameter perturbations does indeed lead to dramatically sharper minima for the bad minimizer, while the wide margin of the good minimizer produces a wide basin.

To validate our observations on a more complex problem, we produce similar sharpness plots for the Street View House Number (SVHN) classification problem in Figure 3.6 using ResNet-18. The SVHN dataset [105] is ideal for this experiment because, in addition to train and test data, the creators collected a large (531k) set of extra data from the same distribution that can be used for $\mathcal{D}_d$ in Eq. (3.4). We minimize the SVHN loss function using standard training with and without poisoning (Eq. (3.4)). The good, high-

[3] 2D loss landscapes are a fairly reliable way to depict minimizer width. Sec. A4 in [91] and Figire 3.7 show the relatively small variance in width w .r.t. random directions.



(a) Good minimizer: 100% train, 97% test (b) Bad minimizer: 100% train, 28% test

Figure 3.6: A slice through the loss landscape of two minima for the SVHN loss function using ResNet-18.

performance minimizer is flat and achieves 97.1% test accuracy, while the bad minimizer is much sharper and achieves 28.2% test accuracy. Both achieve 100% train accuracy and use identical hyperparameters (other than the β factor), network architecture, and weight initialization.

3.6 Implicit Regularization and Dimensionality

We have seen that neural network loss functions are densely populated with both good and bad minima, and that good minima tend to have "flat" loss function geometry. But what causes optimizers to find these good/flat minima and avoid the bad ones?

One hypothesis is that the bias of optimizers towards good minima is caused in part by the volume disparity between the basins around good and bad minima. Flat minima lie in wide basins that occupy a large volume of parameter space, while sharp minima lie in narrow basins that occupy a comparatively small volume of parameter space. As a result, an optimizer using random initialization is more likely to land in the attraction basin for

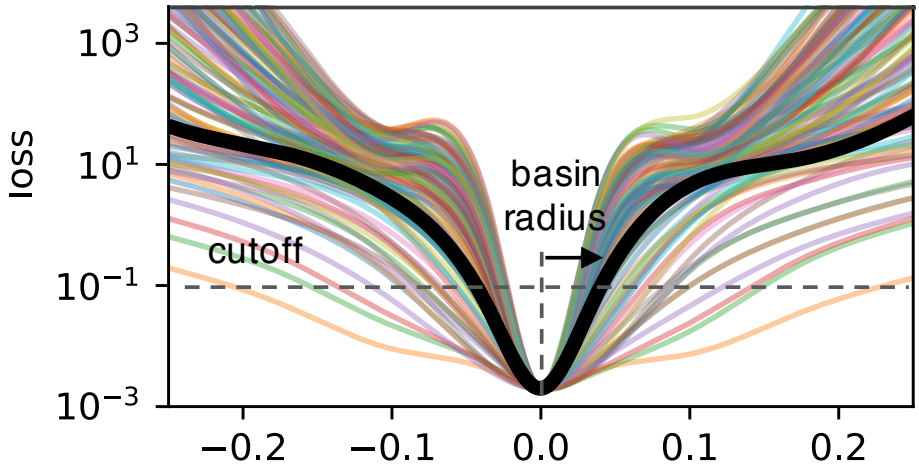


Figure 3.7: SVHN loss along random directions, and the "basin" that lies beneath the cutoff loss value.

a good minimizer than a bad one.

The volume disparity between good and bad minima is magnified by the curse (or, rather, the blessing?) of dimensionality. The differences in "width" between good and bad basins does not appear too dramatic in the visualizations in Figures 3.3 and 3.6, or in sharpness visualizations for other datasets [91]. However, the probability of colliding with a region during a random initialization does not scale with its width, but rather its *volume*. Network parameters live in very high-dimensional spaces where small differences in sharpness between minima translate to exponentially large disparities in the volume of their surrounding basins. It should be noted that the vanishing probability of finding sets of small width in high dimensions is well studied by probabilists, and is formalized by a variety of *escape theorems* [57, 150].

To explore the effect of dimensionality on neural loss landscapes, we quantify the local volume within the low-lying basins surrounding different minima. The volume (or "horizon") of a basin is not well-defined, especially for SGD with discrete time-steps. For this experiment, we define the "basin" to be the set of points in a neighborhood of the

minimizer that have loss value below a cutoff of 0.1 (Figure 3.7). We chose this definition because the volume of this set can be efficiently computed. We calculate the volume of these basins using a Monte-Carlo integration method. Let $r(\phi)$ denote the radius of the basin (distance from minimizer to basin boundary) in the direction of the unit vector ϕ. Then the n-dimensional volume of the basin is $V = \omega_n \mathbb{E}_\phi[r^n(\phi)]$, where $\omega_n = \frac{\pi^{n/2}}{\Gamma(1+n/2)}$ is the volume of the unit n-ball, and Γ is Euler's gamma function. We estimate this expectation by calculating $r(\phi)$ for 3k random directions, as illustrated in Figure 3.7.

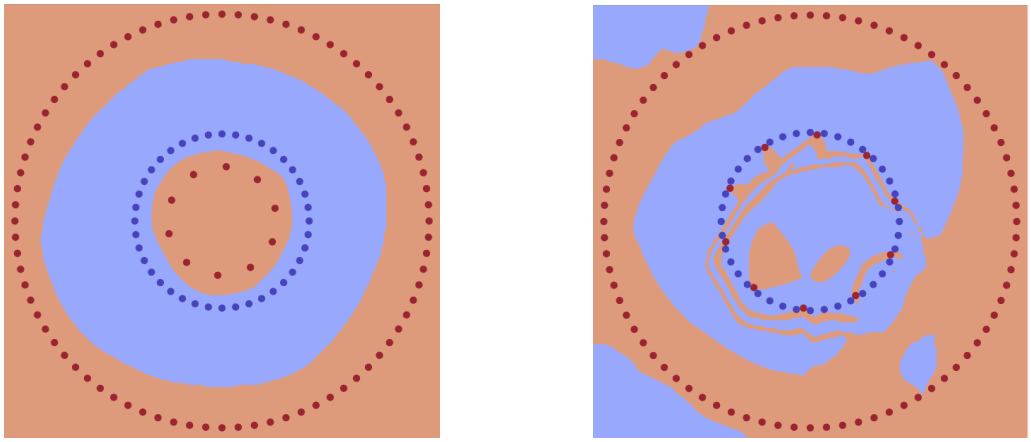


Figure 3.8: A neural network fails to solve a classification problem when the ideal solution is "sharp". This is a binary classification task: red and blue classes. Red and blue dots correspond to the training data. Shaded regions represent neural network predictions.

In Figure 3.4, we visualize the combined relationship between generalization and volume for swissroll and SVHN. By varying β, we control the generalizability of each minimizer. As accuracy on unseen data decreases, we see the radii of the basins decrease as well, indicating that minima become sharper. Figure 3.4 also contains scatter plots showing a severe correlation between performance on unseen data and (log) volume for various choices of the basin cutoff value. For SVHN, the basins surrounding good minima have a volume at least 10,000 orders of magnitude larger than that of bad minima,

rendering it nearly impossible to accidentally stumble upon bad minima.

Finally, we visualize the decision boundaries for several levels of test accuracy in Figure 3.5. All networks achieve above 99.5% training accuracy. As the test performance drops, the area that belongs to the red class begins encroaching into the area that belongs to the blue class, and vice versa. The margin between the decision boundary and training points also decreases until the training points, though correctly classified, sit on "islands" or "peninsulas" as discussed above.

3.6.1 A Counterfactual Experiment: What Can't Neural Nets Solve?

Neural nets solve complex classification problems by finding "flat" minima with class boundaries that assign labels that are stable to parameter perturbations. Using this intuition, can we formulate a problem that neural nets *can't* solve?

Consider the problem of separating the blue and red dots in Figure 3.8. When the distance between the inner rings is large, a neural network consistently finds a well-behaved circular boundary as in Figure 3.8. The wide margin of this classifier makes the minimizer "flat," and the resulting high volume makes it likely to be found by SGD.

We can remove the well-behaved minima from this problem by pinching the margin between the inner red and blue rings. In this case, a network trained with random initialization is shown in Figure 3.8. Now, SGD finds networks that cherry-pick red points, and arc away from the more numerous blue points to maintain a large margin. In contrast, a simple circular decision boundary as in Figure 3.8 would pass extremely close to all points on the inner rings, making such a small margin solution less stable under perturbations

and unlikely to be found by SGD.

3.7 Conclusion

We explored the connection between neural network performance and loss function geometry using visualizations and experiments on classification margin and loss basin volumes, the latter of which does not appear in previous literature.

While experiments can provide useful insights, they sometimes raise more questions than they answer. We explored why the "large margin" properties of flat minima promote performance on unseen data. But what is the precise metric for "margin" that neural networks respect? Experiments suggest that the small volume of bad minima prevents optimizers from landing in them. But what is a correct definition of "volume" in a space that is invariant to parameter re-scaling and other transforms, and how do we correctly identify the attraction basins for good minima? Finally and most importantly: how do we connect these observations back to a rigorous learning framework?

The goal of this Chapter is to foster appreciation for the complex behaviors of neural networks, and to provide some intuitions for why neural networks generalize. Having established the powerful generalization properties of neural networks, in the next Chapter we apply neural networks to a difficult semantic segmentation task. The task is particularly challenging for neural networks because training data is scarce. Nonetheless, we overcome these challenges and develop neural networks that achieve exceptional results.

Chapter 4: Dense Cellular Segmentation for Electron Microscopy

Biologists who use electron microscopy (EM) images to build nanoscale 3D models of whole cells and their organelles have historically been limited to small numbers of cells and cellular features due to constraints in imaging and analysis. This has been a major factor limiting insight into the complex variability of cellular environments. Modern EM can produce gigavoxel image volumes containing large numbers of cells, but accurate manual segmentation of image features is slow and limits the creation of cell models. Segmentation algorithms based on convolutional neural networks can process large volumes quickly, but achieving EM task accuracy goals often challenges current techniques. Here, we define *dense cellular segmentation* as a multiclass semantic segmentation task for modeling tightly packed cells and large numbers of their organelles, and give an example in human blood platelets. We present an algorithm using novel hybrid 2D-3D segmentation networks to produce dense cellular segmentations with accuracy levels that outperform baseline methods and approach those of human annotators. To our knowledge, this work represents the first published approach to automating the creation of cell models with this level of structural detail. More specifically in this work,

- We developed a neural architecture search over 2D encoder-decoder architectures which proves that dense cellular segmentation can be achieved using ensembles of

DNNs – Section 4.3.1.

- We empirically showed that hybrid architectures perform better than 2D architectures and fully 3D architectures. This is in part due to the anisotropic nature of the SBF-SEM data – Section 4.4.
- To this end, we introduced a new 3D biomedical segmentation algorithm based on ensembles of neural networks with separated 2D and 3D convolutional modules and output heads – Figure 4.5.
- We developed a new loss function to effectively train hybrid 2D-3D networks and encourage the network to accurately segment cell boundaries – Section 4.4.3.
- We showed that our algorithm outperforms baselines in mean intersection-over-union (mIoU) metrics, does a better job of maintaining boundaries between adjacent cellular structures, approaches the image quality of our human annotators, and closely matches human performance on a downstream biological analysis task – Section 4.4.5.
- We used our Hybrid 2D-3D network ensemble to segment a billion-voxel block sample in an hour on a single NVIDIA GTX 1080 GPU, demonstrating a segmentation capability that is infeasible without automation and is accessible to commodity computing tools.

This is joint work with Matthew Guay, Adam Anderson, Maria Aronova, Irina Pokrovskaya, Brian Storrie, and Richard Leapman [60, 64]. My contribution was conceiving major parts of the project, including contributions into the design of both the neural architecture search as well as the new Hybrid 2D-3D neural network architectures. I also

performed a significant portion of the experiments, wrote and debugged major parts of the codebase, and wrote substantial portions of the papers.

4.1 Introduction

Biomedical researchers use electron microscopy (EM) to image cells, organelles, and their constituents at the nanoscale. Today, the resulting image volumes can be gigavoxels in size or more, using hardware including the serial block-face scanning electron microscope (SBF-SEM) [34], which employs automated serial sectioning techniques on block samples. This rapid growth in throughput challenges traditional image analytic workflows for EM, which rely on trained humans to identify salient image features. High-throughput EM offers to revolutionize structural biology by providing nanoscale structural detail across macroscopic tissue regions, but using these datasets in their entirety will be infeasibly time-consuming until analytic bottlenecks are automated.

Cell biologists have used semantic segmentations of cellular structures to provide rich 3D ultrastructural models yielding new insights into cellular processes [3, 113, 114], but applying this method across entire SBF-SEM datasets requires automation. Modeling 30 platelet cells across 3 physical platelet samples [113] required nine months' work from two in-lab annotators and represented a small fraction of all imaged cells.

It is challenging to automate dense segmentation tasks for EM due to the image complexity of biological structures at the nanoscale. An image with little noise and high contrast between features may be accurately segmented with simple thresholding methods, while accurate segmentation of images with multiscale features, noise, and

textural content remains an open problem for many biomedical applications. Solving such segmentation problems algorithmically is one of many tasks in applied computer vision that has received increased interest in the past decade, as advances in deep neural network construction and training have driven significant computer vision performance improvements. Natural image-based applications of image segmentation have received enormous attention, with major companies and research institutions creating sophisticated trained neural networks in the pursuit of solutions to problems of economic importance [23, 23, 81, 94, 97, 112, 116].

Work in biomedical imaging has been comparatively modest, but there nevertheless are thriving research communities working on problems in medical computed tomography (CT) [59, 126] and microscopy. A seminal contribution from this area was the U-Net [120], which spawned numerous encoder-decoder variants demonstrating architectural improvements [7, 102] and helped popularize the encoder-decoder motif for segmentation problems in biomedical imaging. An important difference between biomedical and natural imaging is the ubiquity of volumetric imaging methods, including the SBF-SEM studied in this work. These methods have spurred developments in volumetric segmentation, including 2D techniques applied to orthogonal slices of a 3D volume [127], fully-3D segmentation [21, 46, 67, 89, 121], as well as hybrid architectures that incorporate both 2D and 3D spatial processing [22, 89, 111]. Here, we have adapted existing 2D DeeplabV3 [23], 3D DeepVess [46], and 2D and 3D U-Net architectures to our segmentation task as a baseline for our new results.

We find that for our application, hybrid 2D-3D networks work best. Building on previous work in this direction, we introduce a new 3D biomedical segmentation

algorithm based on ensembles of neural networks with separated 2D and 3D convolutional modules and prediction heads. We show that our algorithm outperforms baselines in intersection-over-union (IoU) metrics, does a better job of maintaining boundaries between adjacent cellular structures, approaches the image quality of our human annotators, and closely matches human performance on a downstream biological analysis task. We use the algorithm to segment a billion-voxel block sample in an hour on a single NVIDIA GTX 1080 GPU, demonstrating a segmentation capability that is infeasible without automation and is accessible to commodity computing tools.

4.2 Data Collection

SBF-SEM image volumes were obtained from identically-prepared platelet samples from two humans. Lab members manually segmented portions of each volume into seven classes to analyze the structure of the platelets. The labels were used for the supervised training of candidate network architectures, as well as baseline comparisons.

This study used datasets prepared from two human platelet samples as part of a collaborative effort between the National Institute of Biomedical Imaging and Bioengineering (NIBIB), NIH and the University of Arkansas for Medical Sciences. All human blood draws were approved by the University of Arkansas for Medical Sciences' Institutional Review Board in accordance with national and international guidelines. All donors were informed of possible risks and signed an informed consent form.

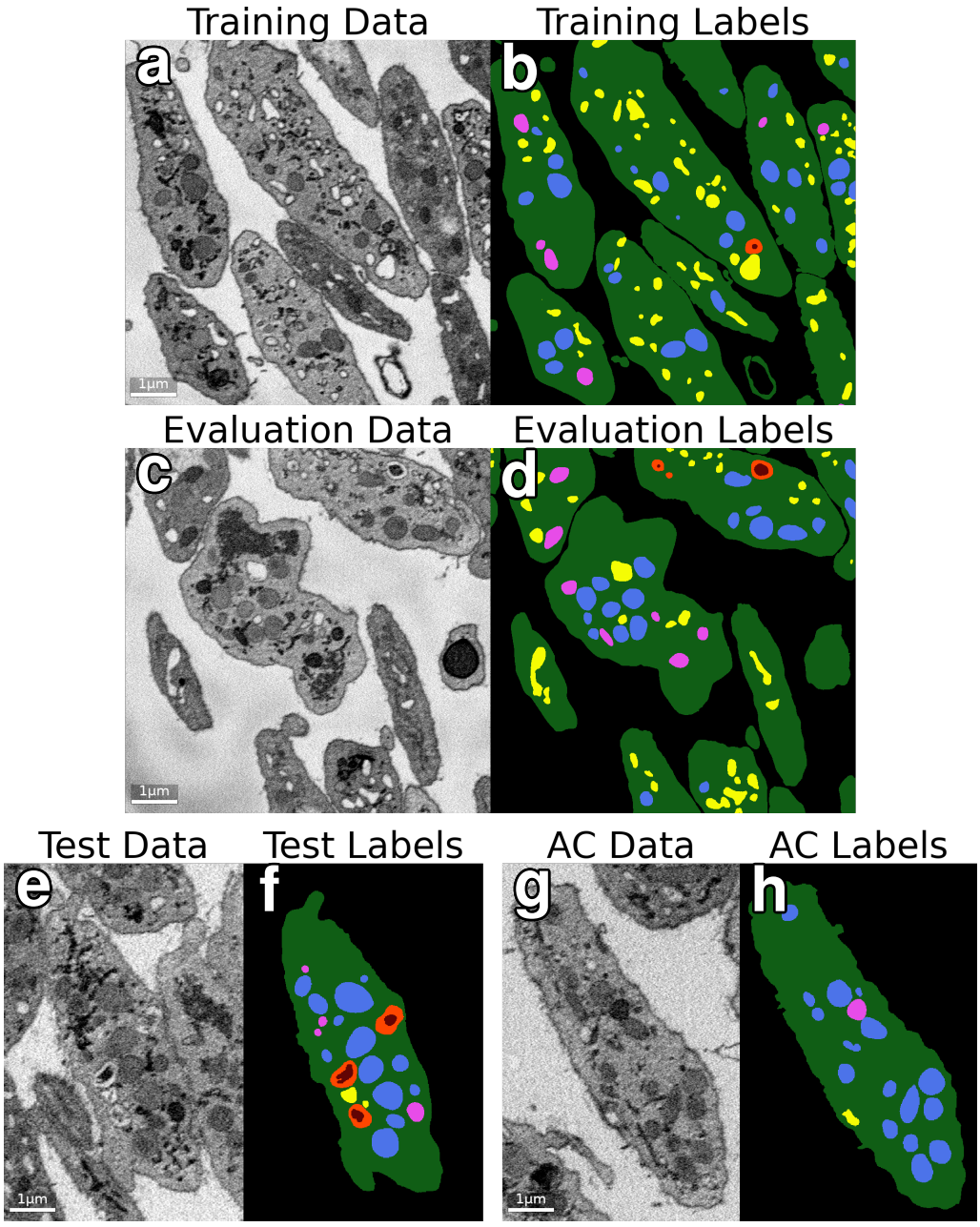


Figure 4.1: **Dataset visualization**. Sample $y - x$ orthoslices of the datasets used in this study. (**a**-**b**) One of the 50 training image data and label orthoslices. (**c**-**d**) One of the 24 evaluation image data and label orthoslices. (**e**-**f**) One of the 121 test image data and label orthoslices . (**g**-**h**) One of the 110 annotator comparison (AC) image data and label orthoslices.

The platelet samples were imaged using a Zeiss Sigma 3View SBF-SEM. The Subject 1 dataset is a (z, y, x) $250 \times 2000 \times 2000$ voxel image with a lateral resolution in

the $y - x$ plane of 10 nm and an axial resolution along the z-axis of 50 nm, from a sample volume with dimensions $12.5 \times 20 \times 20 \mu\text{m}^3$. The Subject 2 dataset is a $239 \times 2000 \times 2000$ voxel image produced by the same imaging protocol with the same lateral and axial resolutions.

We assembled labeled datasets from manually-segmented regions of the platelet image volumes. Lab members created tool-assisted manual segmentations using Amira [140]. Ground-truth labels for the training, evaluation, and test datasets were repeatedly reviewed by subject experts and corrected until accuracy standards were met, a slow feedback process that is necessary to produce high-quality labels. The Annotator 1 and Annotator 2 labels were created in a single pass by lab members without going through a review process from subject experts. As a result, the Annotator 1 and Annotator 2 labels are less accurate, but also much faster to produce . We use the high-quality ground-truth labels to train and validate all networks in this work, but also compare algorithms against the unreviewed Annotator 1 and 2 labels as an additional measure of performance.

The training image was a $50 \times 800 \times 800$ subvolume of the Subject 1 dataset spanning the region $81 \leq z \leq 130$, $1073 \leq y \leq 1872$, $620 \leq x \leq 1419$ in 0-indexed notation. The evaluation image was a $24 \times 800 \times 800$ subvolume of the Subject 1 dataset spanning the region $100 \leq z \leq 123$, $200 \leq y \leq 999$, $620 \leq x \leq 1419$. The test image was a $121 \times 609 \times 400$ subvolume of the Subject 2 dataset spanning the region $0 \leq z \leq 120$, $460 \leq y \leq 1068$, $308 \leq x \leq 707$. The annotator comparison image was a $110 \times 602 \times 509$ subvolume of the Subject 2 dataset spanning the region $116 \leq z \leq 225$, $638 \leq y \leq 1239$, $966 \leq x \leq 1474$. The training and evaluation labels covered the entirety of their respective images, while the test and annotator comparison labels

covered a single cell contained within their image volumes. The labeling schema divides image content into seven classes: `background` (0), `cell` (1), `mitochondrion` (2), `canalicular channel` (3), `alpha granule` (4), `dense granule` (5), and `dense granule core` (6). Voxels labeled as the `cell` class include cytoplasm as well as organelles not accounted for in the labeling schema. Figure 4.1 shows sample images of the datasets and ground truth labels.

The Subject 1 and Subject 2 datasets were binned by 2 in x and y, and aligned. For each of the training, evaluation, and testing procedures, the respective image subvolumes were normalized to have mean 0 and standard deviation 1 before further processing.

4.3 2D Encoder-Decoder Architectures

To start, we developed *GeneNet*, a Python package to rapidly discover, train, and deploy high performing neural network architectures for SBF-SEM segmentation with little user intervention. Here, we demonstrate how to use GeneNet to train an ensemble of segmentation networks for a human platelet tissue sample. Initial results indicate this approach is viable for accelerating the segmentation process and we build upon those results in the next section.

4.3.1 GeneNet

The GeneNet library is designed to allow humans and algorithms to easily implement a 2D encoder-decoder network architecture. It is important to note that 2D architectures take as input 2D image slices, not 3D image volumes. Algorithmic architecture

design allows biomedical researchers to discover high-performing networks for target applications with little manual intervention. In our work, we model an encoder-decoder architecture as a tree of height 4. Trees are easily implemented on a computer. Figure 4.3 depicts a tree and its corresponding encoder-decoder architecture, while Figure 4.2 depicts the general structure of a GeneNet tree. Both Figures will be helpful to follow the explanation in this Section. Each node in the tree is a child class of a Gene object (i.e., Python class) that holds information about the architecture at some level. The root of the tree always has two children, an encoder Gene and a decoder Gene. Encoder Genes in turn have s children nodes, called block Genes. Each block Gene contains information about the convolutional layers at a given spatial scale, every encoder block ends with a downsampling operation. Decoders have $s + 1$ children nodes, which are also all block Genes – one more than the encoder to account for the smallest spatial scale. Decoder blocks always end with an upsampling operation. Every block Gene has at least one child node, called a convolutional Gene. Convolutional Genes hold information about a single convolutional layer, including the number of filters, padding, choice of activation function, etc. Finally, every convolutional Gene has at least 1 child node, called an edge Gene, which holds information about the connections in the architecture – i.e., which tensors are used as inputs to the convolution and how to process them: either as a skip or residual connections.

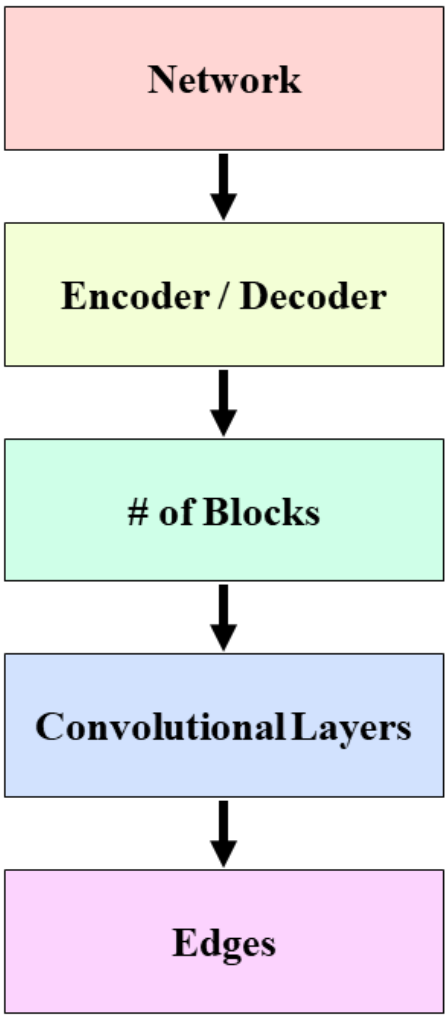


Figure 4.2: The general structure of a GeneNet tree.

This tree data strucutre can be traversed recursively to build a Tensorflow computation graph of the entire network. Architectural hyperparameter choices, such as the number of spatial scales or the number of convolution layers per block, can be encoded as numeric hyperparameters, along with optimization hyperparameters such as learning rate and regularization weights. This design allows us to store and modify encoder-decoder architecture easily.

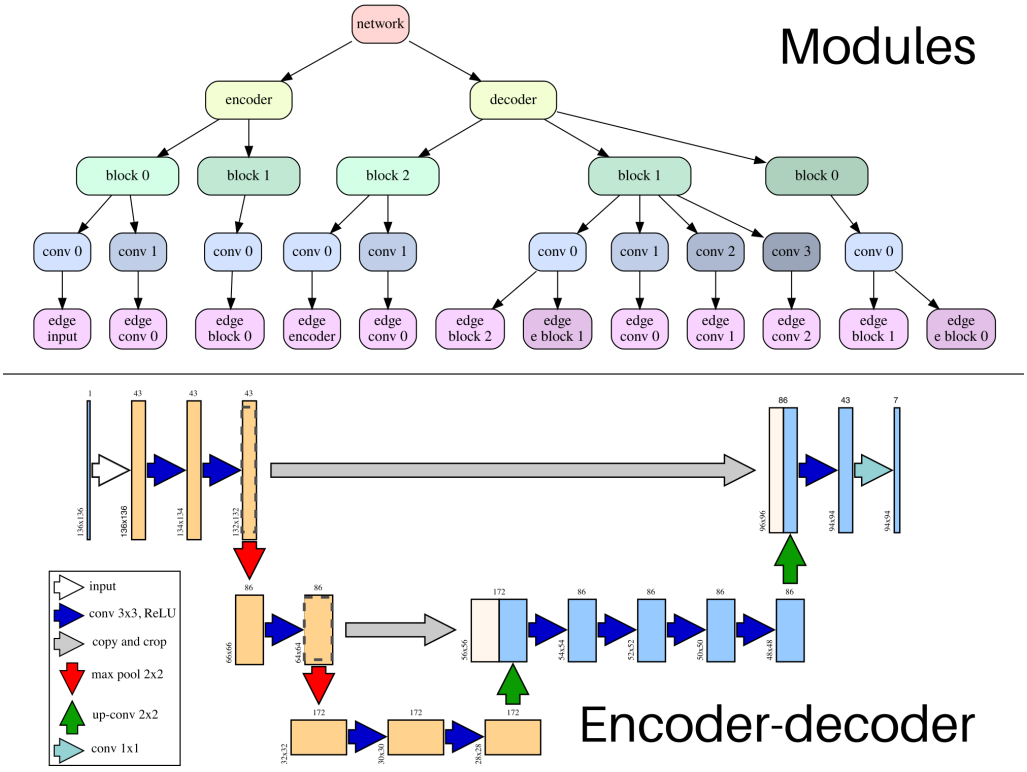


Figure 4.3: Comparison of a tree and the corresponding encoder-decoder network. Encoder and decoder Genes combine convolution block Genes, which combine convolution Genes, which combine edge Genes.

The trees can be used to algorithmically construct neural networks in TensorFlow. The GeneNet library is capable of randomly sampling from spaces of encoder-decoder network architectures when supplied with a feasible region for all hyperparameters, as a rudimentary form of automated neural network architecture design. These networks can then be trained and their performance evaluated. For training, networks use the ADAM optimization method [80], minimizing a combination of class frequency-balanced prediction cross-entropy and regularization terms applied to convolution layers. These networks exhibit a diversity of architectures, a benefit for network ensembles due to the relationship between ensemble generalization error and ensemble ambiguity, a measure of disagreement between ensemble members [85]. An ensemble of networks can then

be used to produce an image segmentation. Each network produces a class prediction map, a probability distribution over possible classes for each voxel in an image. The class prediction maps are averaged to produce an ensemble class prediction map. The final segmentation is created by choosing the most-probable class for each voxel from the class prediction map.

4.3.2 Experiments

We trained 80 randomly-generated networks over the course of 24 hours using the NIH's Biowulf computing cluster. The hyperparameters in the search space were input size, number of features per convolution layer, number of spatial scales, number of convolution layers per block, the learning rate, and several regularization parameters - l^1 and l^2 penalty terms on weights and biases. As a baseline for comparison, we also trained a copy of the original U-net from [120], modifying the final $1x1$ convolution to output 7 features instead of 2 to accommodate our problem's 7 segmentation classes . All networks were trained for 100000 iterations on a $40 \times 800 \times 800$ subset of the training data, with the remaining $10 \times 800 \times 800$ volume reserved for validation. Data augmentation via elastic deformation was used to expand the set of training data [120]. After training, networks were evaluated by computing adjusted Rand scores [75] on the validation data. We then computed validation adjusted Rand scores for ensembles of the n best networks for $n \in [1, 15]$and found that $n = 4$ was optimal. The ensemble of the best 4 networks was used to segment a $10 \times 800 \times 800$ portion of the platelet volume. This testing volume lies directly above the training subvolume in the platelet dataset. Two lab members then

Net ID	Adj. Rand Score	Params	LPES	LPDS
13	**0.886**	39.3M	1,1,3,1	3,1,5,3,3
3	0.885	33.9M	1,2,2	6,6,3,4
45	0.883	20.1M	3,1,2,2	1,1,1,1
38	0.878	27.2M	6,6,6	4,5,5,5
71	0.875	33.5M	2,2,4,2	3,3,6,6,4
27	0.874	12.7M	4,5,4,3	3,1,3,1,2
57	0.874	4.7M	2,3	4,2,1
49	0.872	23.4M	1,2,1	2,1,1,2
32	0.869	28.1M	1,1,3	2,1,2,2
U-Net	0.867	31.0M	2,2,2,2	2,2,2,2,2

Table 4.1: A comparison of certain architectural parameters and validation error (adjusted Rand score) for the nine best randomly-generated networks and the original 2D u-net. LPES: convolution Layers Per Encoding Stack. LPDS: Layers Per Decoding Stack.

corrected the ensemble output, tracking the time required for each 800×800 z-slice of the segmentation. This time was compared with the time required for each of the two lab members to manually segment comparable 800×800 portions of the image volume, in order to determine if the algorithm accelerates the segmentation workflow.

4.3.3 Results

For each of the 80 randomly-generated networks, the final validation adjusted Rand scores are plotted in decreasing order in Figure 4.4. The top 9 networks have adjusted Rand scores higher than the original U-net evaluated on the same data, demonstrating that the random-sampling strategy is capable of producing high-performing network architectures. Plots of those architectures are too large to include here, but a sense of the variation can be gleaned by comparing parameter counts, number of spatial scales, and numbers of convolution layers per stack. The latter two can be represented as a sequence of layer per stack counts, divided into layers per encoding stack (LPES) and layers per decoding stack

(LPDS). Table 4.1 offers a comparison.

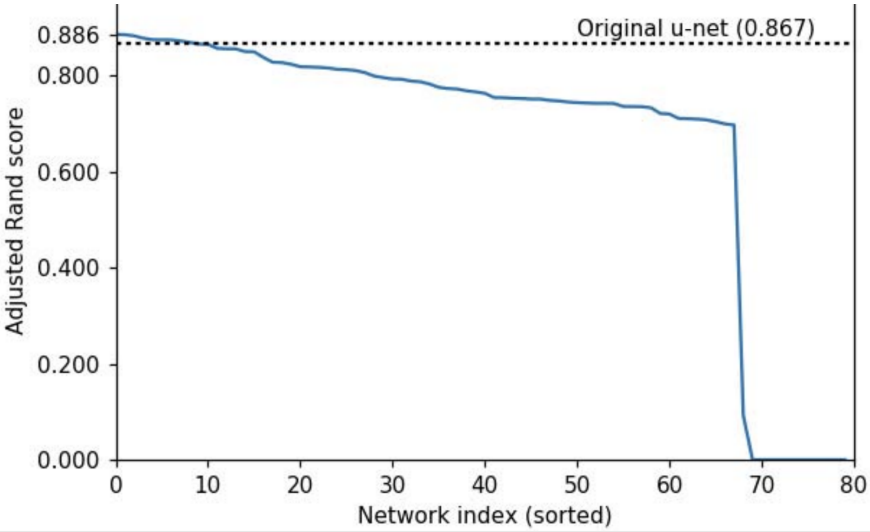


Figure 4.4: Network validation adjusted Rand scores, sorted from best performance to worst. Nine networks outperformed the original 2D U-Net in this metric.

The top-4 network ensemble achieved a validation adjusted Rand score of 0.901. However, both the ensemble and the original U-Net perform poorly on the dense granules and dense granule cores, the least-common classes in the training data. The segmentation networks also find "phantom" organelles not identified by the human. Manual correction of the top-4 ensemble output proved significantly faster than manual segmentation. For each 800×800 image z-slice, lab member 1 averaged 22.3 min per segmentation vs. 10.9 min per correction, a $2.04\times$ speedup. Lab member 2 averaged 18.6 min per segmentation vs. 7.9 min per correction, a $2.35\times$ speedup. Running time of the segmentation algorithm was comparatively negligible, taking no more than 2 seconds per z-slice.

Our work demonstrates that the random architecture generation process enabled by GeneNet is viable for creating high-performing encoder-decoder networks. The automated nature of this process means it can be used in settings where access to a machine learning

expert capable of making informed network design choices is not available. We have also demonstrated that those networks can be combined to form ensembles which effectively accelerate the segmentation of a SBF-SEM dataset. The $10 \times 800 \times 800$ subvolume used to test this process is roughly 0.66% of the unprocessed platelet data, and further performance improvements are required to enable the efficient segmentation of the full image volume. This work allowed us to implement a correction-training feedback loop by using corrected labels to produce new training data for the ensemble networks. The ensemble is then used to produce new segmentations. If the addition of training data decreases correction times sufficiently, this loop can be used to segment the entirety of a large EM image volume. We observed the manual annotators over multiple hours to determine the errors which caused the greatest difficulty for correction, namely:

1. misclassifying an existing organelle,

2. misclassifying non-organelle cellular material as an organelle,

3. and poor segmentations between the boundaries of adjacent organelles.

In order to correct these errors, the annotators required 3D context by observing different z-slices of the image. We hypothesized that neural networks would also require the same 3D context and started researching 3D encoder-decoder architectures described in the next section.

4.4 Hybrid 2D-3D Networks

In this section, we build upon the results of the neural architecture search strategy based on GeneNet. As we discussed above, GeneNet allowed us to conclude that 3D context is necessary to obtain adequate segmentation results. However, naively using GeneNet to perform a neural architecture search over 3D encoder-decoder architectures is not feasible due to GPU memory limitations. Here, we abandon automated neural architecture search in favor of a manual network architecture design that allows us to successfully utilize 3D context.

4.4.1 Validation and Performance Metrics

The performance metric used in the remainder of this Chapter is mean intersection-over-union (MIoU) between ground-truth image segmentation ℓ's 7 labeled sets $\{L_j = v \in \Omega \,|\, \ell(v) = j\}_{j \in C}$ and predicted segmentation's $\hat{\ell}$ labeled sets $\{\hat{L}_j = v \in \Omega \,|\, \hat{\ell}(v) = j\}_{j \in C}$. Given two sets A and B, $\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Then for segmentations ℓ and $\hat{\ell}$ with their corresponding labeled sets over the 7 semantic classes,

$$\mathrm{MIoU}(\ell, \hat{\ell}) = \frac{1}{7} \sum_{j \in C} \mathrm{IoU}(L_j, \hat{L}_j). \tag{4.1}$$

More generally, for a subset of labels $D \subseteq C$, one can compute the MIoU over D, or $\mathrm{MIoU}^{(D)}$, as

$$\mathrm{MIoU}^{(D)}(\ell, \hat{\ell}) = \frac{1}{|D|} \sum_{j \in D} \mathrm{IoU}(L_j, \hat{L}_j). \tag{4.2}$$

Note that this definition weights the IoU scores for each class equally, regardless of the number of examples of each class in the dataset. One may choose to use a class frequency-weighted MIoU instead to reflect this class imbalance, but we choose to use an unweighted MIoU to emphasize performance on rarer classes.

Here we are concerned with MIoUs over two sets of labels: $\mathrm{MIoU}^{(all)}$ over the set of all 7 class labels, and $\mathrm{MIoU}^{(org)}$ over the set of 5 organelle labels 2-7. Our network validation metrics were $\mathrm{MIoU}^{(all)}$ and $\mathrm{MIoU}^{(org)}$ on the evaluation dataset, and $\mathrm{MIoU}^{(org)}$ on the test dataset. Test data uses $\mathrm{MIoU}^{(org)}$ because the labeled region is a single cell among several unlabeled ones, and restricting validation to the labeled region invalidates MIoU stats for the background and cell classes (0 and 1). We include evaluation $\mathrm{MIoU}^{(org)}$ to quantify how performance drops between a region taken from the physical sample used to generate the training data, and a new physical sample of the same tissue system.

4.4.2 Neural Architectures and Ensembling

The highest-performing network architecture in this work, 2D-3D+3x3x3, is a composition of a 2D U-net-style encoder-decoder and 3D convolutional spatial pyramid, with additional 3x3x3 convolutions at the beginning of convolution blocks in the 2D encoder-decoder. All convolutions are zero-padded to preserve array shape throughout the network, allowing deep architectures to operate on data windows with small z-dimension. A ReLU activation follows each convolution. All convolution and transposed convolutions use bias terms. The architecture is fully specified as a diagram in Figure 4.5. Additionally, several baseline comparison networks and three 2D-3D+3x3x3 ablation networks were

also tested in this work and are described in Section 4.4.1.

To build a 2D-3D network, one can adapt a 2D U-net-style encoder-decoder module to work on 3D data by recasting 2D 3x3 convolutions as 1x3x3 convolutions, and 2D 2x2 max-pooling and transposed convolution layers as 1x2x2 equivalents. In this way, a 3D input volume can be processed in a single computation graph as a sequence of independent 2D regions in a 2D module and as a contiguous 3D region in a 3D module, and the 2D and 3D modules can be jointly trained end-to-end. This formulation also allows for seamless combination of batched-2D and 3D operations within a module, demonstrated in the 2D-3D+3x3x3 architecture as 2D convolution block-initial 3x3x3 convolutions. Intermediate 2D class predictions $\hat{x}_{2D}$ are formed from the 2D module output, and the 2D output and class predictions are concatenated along the feature axis to form an input to a 3D spatial pyramid module. The 3D module applies a 1x2x2 max pool to its input to form a two-level spatial pyramid with scales 0 (input) and 1 (pooled). The pyramid elements separately pass through 3D convolution blocks, and the scale 1 block output is upsampled and added to the scale 0 block output with a residual connection to form the module output. 3D class predictions $\hat{x}_{3D}$ are formed from the 3D module output, and the final segmentation output $\hat{\ell}$ of the algorithm is a voxelwise argmax of the 3D class predictions. To build a 2D-3D+3x3x3 network, we inserted 3x3x3 convolution layers at the beginning of the first two convolution blocks in the 2D encoder and the last two convolution blocks in the 2D decoder.

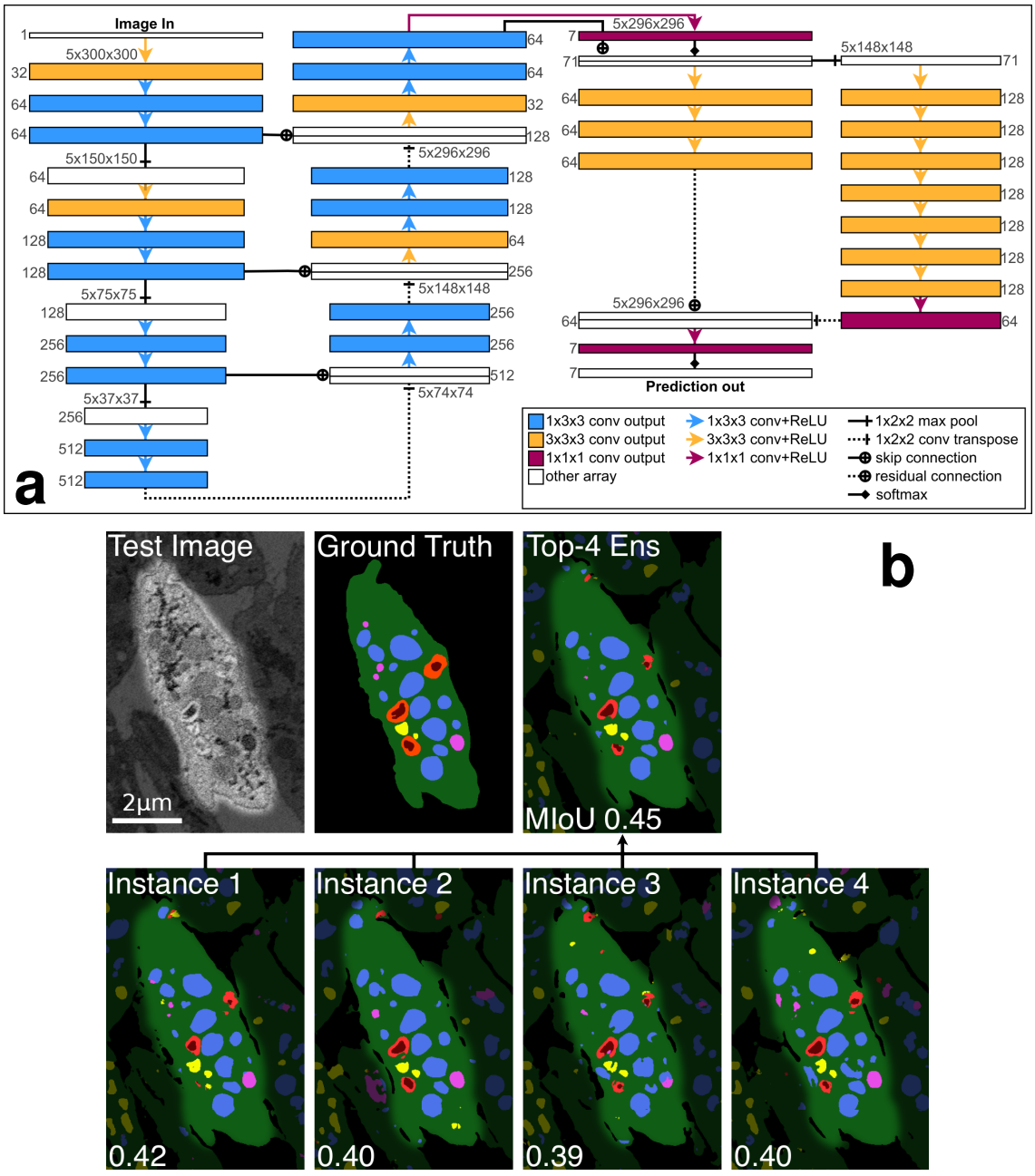


Figure 4.5: **Methods**. (**a**) Diagram of the 2D-3D+3x3x3 network architecture, the best design tested in this work. A 1-channel 3D image is passed through the network to produce a 7-channel output prediction of per-voxel probability distributions over the 7 label classes. Boxes represent multidimensional arrays, and arrows represent operations between them. Number triplets along box tops are array spatial axis sizes in (z, y, x) order. Numbers along box sides are array channel axis sizes. (**b**) Illustration of initialization-dependent performance of trained segmentation networks, and exploiting it for ensembling. An image of the test cell and ground truth labels are compared with segmentations of the best 4 trained 2D-3D+3x3x3 network instances and an ensemble formed from them. The ensemble improves $\mathrm{MIoU}^{(org)}$ by 7.1% over the best single network.

Given a collection of networks' 3D class predictions, one can form an ensemble prediction by computing a voxelwise average of the predictions and computing a segmentation from that. Ensembling high-quality but non-identical predictions can produce better predictions [85], and there is reason to think that more sophisticated ensembles could be constructed from collections of diverse neural architectures [60, 61, 62, 63], but in this work we use a simple source of differing predictions to boost performance: ensembles of identical architectures trained from different random initializations. The sources of randomness in the training procedure are examined more thoroughly in Section 4.4.1, but in our experiments this variation produced a small number of high-performing network instances per architecture with partially-uncorrelated errors.

4.4.3 Network Training

We consider a network predicting 7 classes $C = \{0, \ldots, 6\}$ for each voxel in a shape-(o_z, o_x, o_y) data window Ω containing $N = o_z o_x o_y$ voxels $\{v_i\}_{i=1}^{N}$. The ground-truth segmentation of this region is a shape-(o_z, o_x, o_y) array ℓ such that $\ell(v) \in C$ is the ground-truth label for voxel v. A network output prediction is a shape-$(7, o_z, o_x, o_y)$ array $\hat{x}$ such that $x_v \triangleq \hat{x}(:, v)$ is a probability distribution over possible class labels for voxel v. The corresponding segmentation $\hat{\ell}$ is the per-voxel $\arg\max$ of $\hat{x}$. Inversely, from ℓ one may construct a shape-$(7, o_z, o_x, o_y)$ per-voxel probability distribution x such that $x_v(i) = 1$ if $i = \ell(v)$ and 0 if not, which is useful during training. All networks used a minibatch size of 1, so the minibatch axis is omitted from each of the array shape descriptions in this work. Array shapes are given in (C, Z, Y, X) order, where C is the

array size along the channel axis.

We trained our networks as a series of experiments, with each experiment training and evaluating 1 or more instances of a fixed network architecture. Instances within an experiment varied only in the random number generator (RNG) seed used to control trainable variable initialization and training data presentation order. In addition to the main 2D-3D+3x3x3 architecture, there were three ablation experiments and five baseline experiments – Original U-Net[120], 3D U-Net Thin[27], 3D U-Net Thick [27],Deeplab + DRN, and Deeplab + ResNet101 [23, 71]. Instances were trained and ranked by evaluation dataset MIoU. Experiments tracked evaluation MIoU for each instance at each evaluation point throughout training, and saved the final weight checkpoint as well as the checkpoint with highest evaluation MIoU. In this work we report evaluation MIoU checkpoints for each instance. The 2D-3D+3x3x3 experiment and its ablations trained 26 instances for 40 epochs with minibatch size 1 (33k steps). The Original U-Net experiment trained 500 instances for 100 epochs with minibatch size 1 (180k steps). The 3D U-Net Thin experiment trained 26 instances for 100 epochs with minibatch size 1 (29k steps), and the 3D U-Net Thick experiment trained 26 instances for 100 epochs with minibatch size 1 (30k steps). The Deeplab + DRN and Deeplab + ResNet101 experiments trained 1 instance each for 200 epochs with minibatch size 4 (360k steps). Due to poor performance and slow training times of the Deeplab models, we deemed it unnecessary to train further instances. Networks were trained on NVIDIA GTX 1080 and NVIDIA Tesla P100 GPUs.

This Subsection details the training of the 2D-3D+3x3x3 network. Baseline and ablation networks were trained identically except as noted in Section 4.4.1. All trainable variables were initialized from Xavier uniform distributions. Each instance was trained

for 40 epochs on shape-$(1, 5, 300, 300)$ windows extracted from the training volume, and output a shape-$(7, 5, 296, 296)$ class prediction array. The number of windows in each epoch was determined by a window spacing parameter which determined the distance along each axis between the top-back-left corners of each window, here $(2, 100, 100)$, resulting in 828 windows per epoch. An early stopping criterion halted the training of any network that failed to reach an MIoU of 0.3 after 10 epochs.

Networks were trained using a regularized, weighted sum of cross-entropy functions. The network has a set Θ trainable variables divided into four subsets: Θ_{2D} for variables in the 2D encoder-decoder module, Θ_{3D} for variables in the 3D spatial pyramid module, the single 1x1x1 convolution variable $\{\theta_{2DP}\}$ which produces intermediate 2D class predictions $\hat{x}_{2D}$ from the encoder-decoder's 64 output features, and the single 1x1x1 convolution variable $\{\theta_{3DP}\}$ which produces the final 3D class predictions $\hat{x}_{3D}$ from the spatial pyramid's 64 output features. The loss function comparing predictions against ground-truth labels is

$$L(x, \hat{x}_{3D}, \hat{x}_{2D}; \Theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\mathcal{W} \otimes \mathcal{H}\left(x, \hat{x}_{3D}\right)\right]_i + \frac{c_{2D}}{N}\sum_{i=1}^{N}\left[\mathcal{W} \otimes \mathcal{H}\left(x, \hat{x}_{2D}\right)\right] + \lambda_{2D}\sum_{\theta\in\Theta_{2D}}\|\theta\|_2^2 + \lambda_{3D}\sum_{\theta\in\Theta_{3D}}\|\theta\|_2^2 + \lambda_P\left(\|\theta_{2DP}\|_2^2 + \|\theta_{3DP}\|_2^2\right), \tag{4.3}$$

where $\lambda_{2D} = 1 \times 10^{-4.7}$ and $\lambda_{3D} = 1 \times 10^{-5}$ are L^2 regularization hyperparameters for the variables in Θ_{2D} and Θ_{3D}, $\lambda_P = 1 \times 10^{-9}$ is an L^2 regularization hyperparameter for the predictor variables θ_{2DP} and θ_{3DP}, and $c_{2D} = 0.33$ is a constant that weights the

importance of the intermediate 2D class predictions in the loss function. $\mathcal{H}(x, \hat{x})$ is the voxelwise cross-entropy function, i.e.,

$$\mathcal{H}(x, \hat{x})_v \triangleq H(x_v, \hat{x}_v) \triangleq -\sum_{j=1}^{7} x_v(j) \log\left[\hat{x}_v(j)\right] = -x_v(\ell_v) \log\left[\hat{x}_v(\ell_v)\right]. \tag{4.4}$$

$\mathcal{W}$ is a shape-$(5, 296, 296)$ array of weights; its Kronecker product with $\mathcal{H}$ produces a relative weighting of the cross-entropy error per voxel. This weighting strategy is based generally on the approach in [120]:

$$\mathcal{W} \triangleq w + \mathcal{W}_{cb} + \mathcal{W}_{ep}.$$

The initial $w = 0.01$ is a constant that sets a floor for the minimum weight value, $\mathcal{W}_{cb}$ is a class-balancing term such that $\mathcal{W}_{cb,i} \propto 1/N_i$, where N_i is the number of occurrences in the training data of ℓ_i, rescaled so that $\max \mathcal{W}_{cb} = 1$. $\mathcal{W}_{ep}$ is an edge-preserving term that upweights voxels near boundaries between image objects and within small 2D cross-sections. In [120] this is computed using morphological operations. We used a sum of scaled, thresholded diffusion operations to approximate this strategy in a manner that requires no morphological information. $\mathcal{W}_{ep}$ is built up as a rectified sum of four terms:

$$\mathcal{W}_{ep} \triangleq R_\alpha \left(\mathcal{W}_{bkgd \to cell} + \mathcal{W}_{cell \to bkgd} + \mathcal{W}_{cell \to org} + \mathcal{W}_{org \to cell}\right), \tag{4.5}$$

where $R_\alpha(W) = \mathrm{ReLU}(W - \alpha) \cdot \frac{\max W}{\max W - a}$. For each term, we choose two disjoint subsets C_{source} and C_{target} of the classes C. Let ℓ_{source} be the binary image such that

$\ell_{source}(v) = 1$ if $\ell(v) \in C_{source}$ and $\ell_{source}(v) = 0$ otherwise. Define

$$M_{source}(c, \sigma) \triangleq c \cdot \ell_{source} * k_\sigma,$$

where $*$ denotes convolution and k_σ is a Gaussian diffusion kernel with standard deviation σ. Then, $\mathcal{W}_{source \to target}(v) = \mathcal{M}_{source}(v)$ if $\ell(v) \in C_{target}$, and is 0 otherwise. The terms in the $\mathcal{W}_{ep}$ array used in this work were computed using class subsets $bkgd = \{0\}$, $cell = \{1\}$, and $org = \{2, 3, 4, 5, 6\}$, $\alpha = 0.25$, $c = 0.882$, and $\sigma = 6$. The error weighting array used in this work and the code used to generate it are available with the rest of the platelet dataset at `leapmanlab.github.io/dense-cell`. See Figure 4.13 for a visualization of the error weighting array. $\mathcal{W}_{cb}$ is calculated all at once across the entire 3D training volume, while $\mathcal{W}_{ep}$ is calculated independently per each 2D z-slice of the training volume.

We employed data augmentation to partially compensate for the limited available training data. Augmentations were random reflections along each axis, random shifts in brightness ($\pm 12\%$) and contrast ($\pm 20\%$), and elastic deformation as in (Ronneberger et al., 2015). For elastic deformation, each 800x800 $x - y$ plane in the shape-$(50, 800, 800)$ training data and label arrays was displaced according to a shape-$(800, 800, 2)$ array of 2D random pixel displacement vectors, generated by bilinearly upsampling a shape-$(20, 20, 2)$ array of iid Gaussian random variables with mean 20 and standard deviation 0.6. During each epoch of training, a single displacement map was created and applied to the entire training volume before creating the epoch's batch of input and output windows. Training used the ADAM optimizer with learning rate 1×10^{-3}, $\beta_1 = 1 - 1 \times 10^{-1.5}$,

$\beta_2 = 1 - 1 \times 10^{-2.1}$, and $\epsilon = 1 \times 10^{-7}$. Training also used learning rate decay with an exponential decay rate of 0.75 every $1 \times 10^{3.4}$ training iterations.

4.4.4 Experiments

Using the procedure outline in Section 4.4.1, the performance of the 2D-3D+3x3x3 network was compared against three ablations and five baseline networks. The three ablations each tested one of three features that distinguish the 2D-3D+3x3x3 network in this work from similar baselines. The first, 2D-3D+3x3x3 No 3x3x3 Convs, replaces the 3x3x3 convolutions in the net's encoder-decoder module with 1x3x3 convolutions that are otherwise identical. With this ablation, the network's encoder-decoder loses any fully-3D layers. The second, 2D-3D+3x3x3 No Multi-Loss, modifies the loss function in Equation (4.3) by removing the term involving $\hat{x}_{2D}$ but otherwise leaving the architecture and training procedure unchanged. This ablation tests whether it is important to have auxiliary accuracy loss terms during training. The third ablation, 2D-3D+3x3x3 No 3D Pyramid, removes the 3D spatial pyramid module and 3D class predictor module from the network architecture, so that $\hat{x}_{2D}$ is the network's output. Correspondingly, the loss term involving $\hat{x}_{3D}$ is removed from Equation (4.3).

We implemented five baseline networks by adapting common models in the literature to our platelet segmentation problem. Three of these were 2D - The original U-Net [120] as well as two Deeplab variants [23] using a deep residual network (DRN) backbone and a ResNet101 backbone [71], minimally modified to output 7 class predictions. The original U-Net used $(572, 572)$ input windows and $(388, 388)$ output windows, while the

Deeplab variants used $(572, 572)$ input and output windows. The two 3D networks were fully-3D U-Net variants adapted on the 3D U-Net in (Çiçek et al., 2016) [27] - 3D U-Net Thin and 3D U-Net Thick. The variants used same-padding, had three convolutions per convolution block, and two pooling operations in the encoder for convolution blocks at three spatial scales. The 3D U-Net Thin network used $(5, 300, 300)$ input windows and $(5, 296, 296)$ output windows, and pooling and upsampling operations did not affect the z spatial axis. The 3D U-Net Thick network used $(16, 180, 180)$ input windows and $(16, 180, 180)$ output windows, and pooled and upsampled along all three spatial axes.

To determine whether one architecture is superior to another, trained instances are compared with each other. However, sources of randomness in the training process induce a distribution of final performance metric scores across trained instances of an architecture, so that a single sample per architecture may be insufficient to determine which is better. While expensive, a collection of instances can be trained and evaluated to empirically approximate the performance distribution for each architecture. In this way, better inferences may be made about architecture design choices. Figure S5 shows the empirical performance distributions for the 26 trials of the 2D-3D+3x3x3 architecture and its three ablations, as well as the 26 trials of the 3D U-Net and 500 trials of the 2D Original U-Net.

In addition to the multiclass baselines, we chose to also evaluate a CDeep3M plug-and-play system [66] that can be spun up on Amazon Web Services (AWS) for binary segmentation problems. In a similar vein to our work and others', they use an ensemble of convolutional neural networks to perform binary segmentation tasks. This differs from the multiclass segmentation problems that we address, but their polished workflow makes

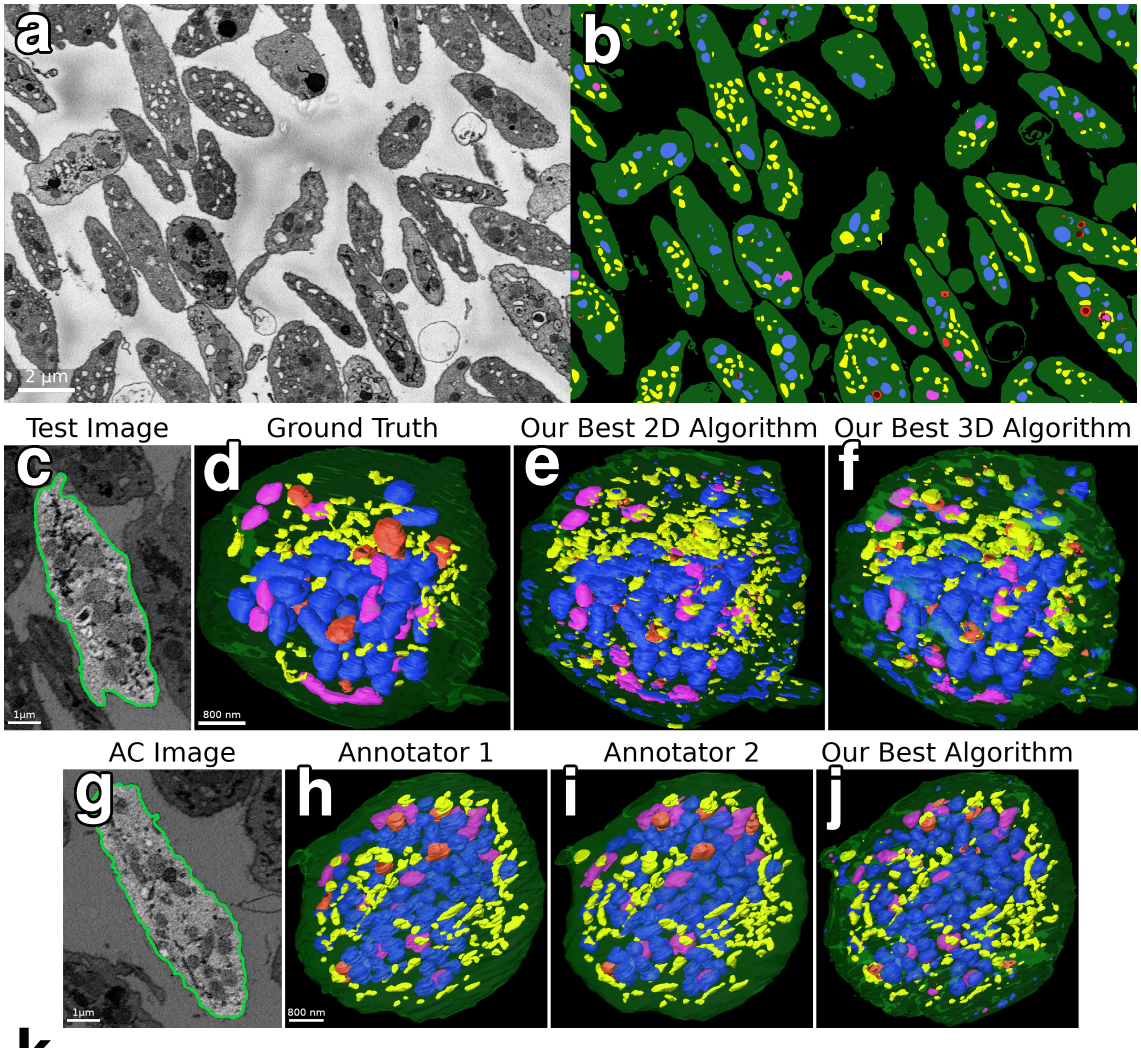
it easy to replicate and train on new data. We therefore decided to evaluate CDeep3M on a comparable binary segmentation task with our data, wherein all non-`background` classes were grouped together into a single `cell` class. Using the AWS stack provided on the project GitHub page (`https://github.com/CRBS/cdeep3m`), we trained the networks used in their 3D segmentation ensemble for 30000 iterations on our training dataset, using all other default hyperparameters. Training took approximately 96 hours on an Amazon EC2 instance with an NVIDIA P100 GPU card.

After training completed, we ran the CDeep3M 3D ensemble's prediction tool on our evaluation dataset, and compared it with a binarized version of our best algorithm's segmentation of the evaluation dataset. We binarized our algorithm's segmentation the same way we binarized our ground truth labels, by mapping together all the non-`background` segmented classes. The CDeep3M algorithm, however, produces a single per-voxel probability map that indicates the probability each voxel belongs to a cell region. To compute a segmentation from the probability map, a cutoff threshold t must be specified - a segmentation with threshold t assigns the `cell` class to all voxels with probability greater than t, and `background` to all others. We computed MIoU scores for our lab's (LCIMB) segmentation, as well as CDeep3M segmentations with thresholds in

$$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}.$$

A final point of comparison was drawn between the top algorithm's performance and the initial work of laboratory scientific image annotators, Annotator 1 and Annotator 2. The three sources each labeled the annotator comparison region from the Subject

2 platelet sample. Pairwise $\mathrm{MIoU}^{(org)}$ scores and organelle confusion matrices were calculated to compare the level of disagreement between two human labelings and between humans and the algorithm. We also computed organelle volume fractions for each segmentation to compare performance in segmentation applications to downstream analysis tasks. The cell volume fraction of an organelle is equal to the summed voxels of all organelles in a cell, divided by the cell's volume. To compute this quantity for each organelle, the number of voxels for each organelle label is divided by the number of voxels in the cell. For the algorithmic result, since the semantic segmentation map does not distinguish between separate cells in the field of view, a mask for the single annotator comparison dataset cell was approximated as all non-background-labeled voxels in a small region around the Annotator 1 cell mask.



k

Algorithm	Eval MIoU$^{(org)}$	Test MIoU$^{(org)}$
Top-4 2D-3D+3x3x3	0.595	**0.446**
2D-3D+3x3x3 (10.3M)	0.568	0.417
Original U-Net (31.0M)	0.515	0.334
3D U-Net Thin (2.0M)	0.502	0.280
Deeplab + DRN (40.7M)	0.522	0.130

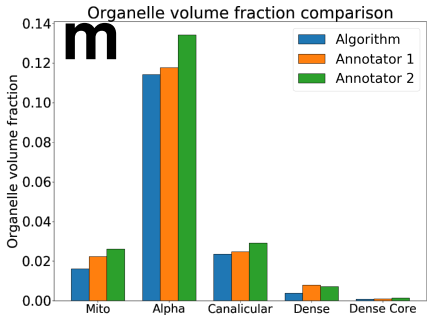


Figure 4.7: **Results**. (**a**-**b**) Orthoslice of Subject 1 image and segmentation. (**c**) Test dataset orthoslice, segmented cell highlighted. (**d**-**f**) Comparison between ground truth segmentation of test cell and our best 2D and 3D algorithms. (**g**) Annotator comparison (AC) dataset orthoslice, segmented cell highlighted. (**h**-**j**) Annotator comparison cell segmentations, comparing the two human annotators and our best (3D) algorithm. (**k**) Summarized comparison of mean intersection-over-union across organelle classes (MIoU$^{(org)}$) on test and evaluation datasets for segmentation algorithms. For full results, see Table 4.2. (**m**) Comparison of organelle volume fractions between two human annotators and our best algorithm, computed from annotator comparison cell segmentations.

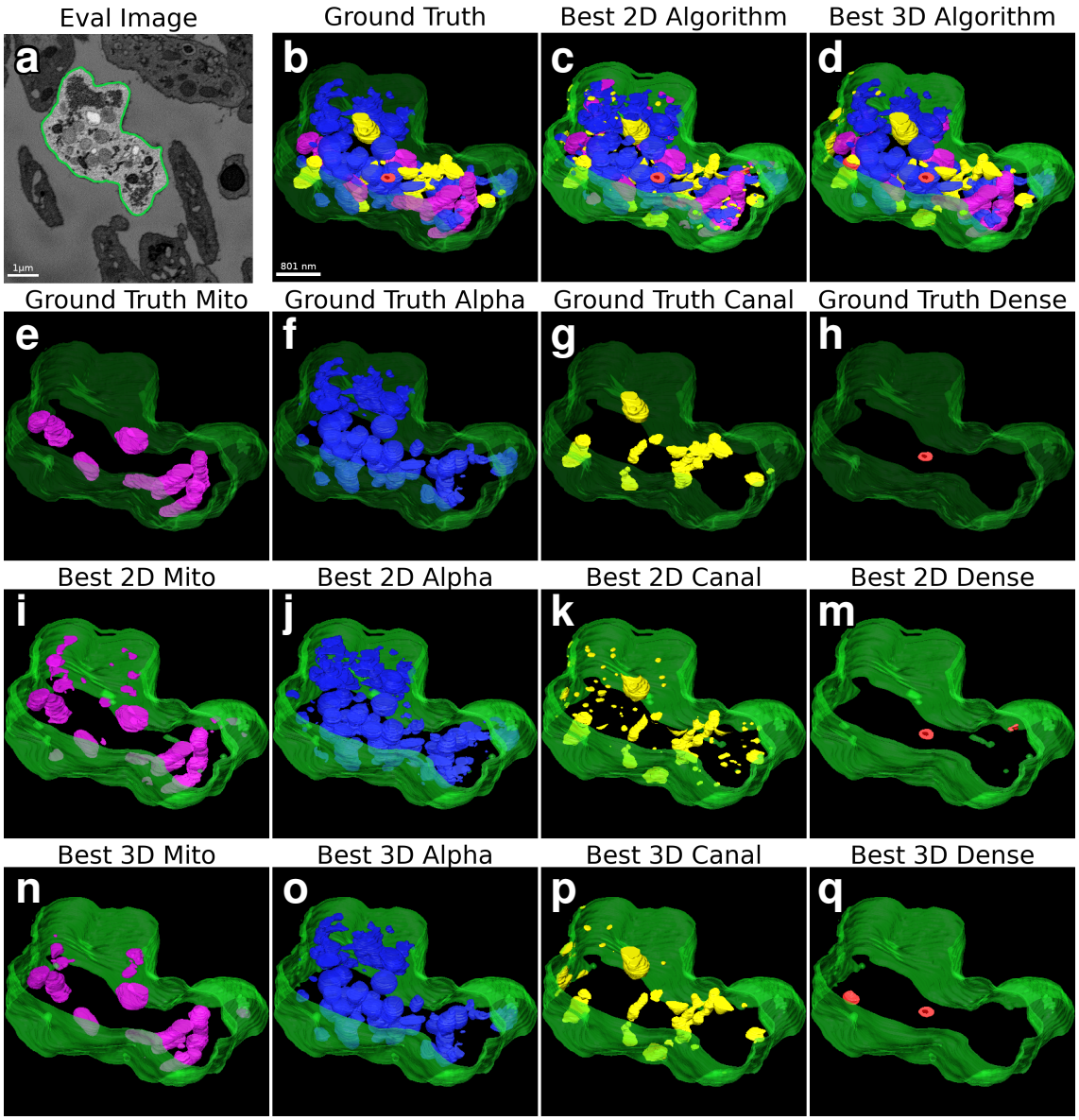


Figure 4.8: **Evaluation dataset segmentation renderings**. In each subfigure, "best" refers to our best 2D or 3D segmentation algorithm out of the ones we evaluated. (**a**) Orthoslice of the evaluation dataset with rendered cell highlighted. (**b**-**d**) Ground truth and our best 2D and 3D algorithm segmentations of the evaluation cell region showing all organelles. (**e**-**h**) Ground truth segmentations of individual organelles - from left to right: mitochondria (Mito), alpha granules (Alpha), canalicular channels (Canal), dense granules (Dense). (**i**-**m**) Our best 2D algorithm segmentations of individual organelles. (**n**-**q**) Our best 3D algorithm segmentations of individual organelles.

4.4.5 Results

Inspired by existing work on combining 2D and 3D computations for volumetric data analysis [22, 111] we experiment with combinations of 2D and 3D neural modules to trade off between computational efficiency and spatial context. The highest-performing network architecture in this work, 2D-3D+3x3x3, is a composition of a 2D U-Net-style encoder-decoder and 3D convolutional spatial pyramid, with additional 3x3x3 convolutions at the beginning of convolution blocks in the encoder-decoder. We use same-padded convolution operations throughout, so that 3D operations can be used on anisotropic data windows with small size along the z axis.

Our best algorithms as defined by MIoU score are ensembles that average the per-voxel class probability distributions across several networks. The ensembled networks are identical architectures trained from different random weight initializations. When describing segmentation algorithms, we use Top-k to indicate an ensemble of the best k instances of an architecture. Figure 4.5 details our best network architecture and illustrates the ensembling process.

For the main experiment of this study, we train baseline architectures from the literature, our new architecture, and ablations of the new architecture on a dense cellular segmentation task by supervised learning from the training dataset. We compare single-network and ensemble segmentation performance on the evaluation and test datasets. We conclude that our algorithm outperforms baselines, the differentiating features of our final best architecture are responsible for the performance differences, and that multi-instance ensembles significantly improve performance over single networks. The results of this

experiment are shown in Figure 4.7.

	Eval $\mathrm{MIoU}^{(all)}$	Eval $\mathrm{MIoU}^{(org)}$	Test $\mathrm{MIoU}^{(org)}$
Top-4 2D-3D+3x3x3	0.686	0.595	**0.446**
Top-5 No 3x3x3 Convs	**0.690**	**0.601**	0.419
Top-3 No Multi-Loss	0.633	0.524	0.338
Top-3 No 3D Pyramid	0.681	0.590	0.421
Top-5 Original U-Net	0.663	0.562	0.371

(a) Ensembles of Networks

2D-3D+3x3x3 (10.3M)	0.665	0.568	0.417
No 3x3x3 Convs (9.9M)	0.667	0.571	0.358
No Multi-Loss (10.3M)	0.652	0.550	0.355
No 3D Pyramid (7.9M)	0.646	0.542	0.376

(b) Single 2D-3D+3x3x3 Network and Ablations

Original U-Net (31.0M)	0.626	0.515	0.334
3D U-Net Thick (2.1M)	0.496	0.348	0.314
3D U-Net Thin (2.0M)	0.613	0.502	0.280
Deeplab + DRN (40.7M)	0.632	0.522	0.130
Deeplab + ResNet101 (59.3M)	0.585	0.456	0.124

(c) Baseline Networks

Table 4.2: **Comprehensive network performance statistics**. Segmentation algorithm results summary showing mean intersection-over-union (MIoU) across all classes ($\mathrm{MIoU}^{(all)}$) on evaluation data and MIoU across organelle classes ($\mathrm{MIoU}^{(org)}$) on evaluation and test data. The Subject 2 dataset from which the test data is taken contains only a small number of labeled cells among unlabeled ones; we use $\mathrm{MIoU}^{(org)}$ to measure test performance since restricting the MIoU stat to labeled regions invalidates `background` and `cell` class statistics. (**a**) Results for the best ensemble from each architecture tested. A top-k ensemble averages the predictions of the best k trained networks as judged by $\mathrm{MIoU}^{(all)}$ on the evaluation dataset. (**b**) Results for the best single network from each architecture class. Trainable parameter counts are in parentheses. (**c**) Results from baseline comparison networks. Trainable parameter counts are in parentheses

We consider test performance to be the best indicator of an algorithm's performance

as it shows its ability to generalize across different samples. Figure 4.7 row 2 compares visualizations of the best 3D segmentation algorithms with ground-truth labels and image data for the test dataset. Figure 4.7 row 4 highlights the most notable performance results, and more performance statistics can be found in Table 4.2. Additional 3D renderings comparing manual and algorithmic performance are shown in Figures 4.8 and 4.11, and a 2D comparison of segmentations of the evaluation dataset by all networks tested in this work is shown in Figure 4.10.

We also compare our best algorithm against the segmentations of scientific image annotators, Annotator 1 and Annotator 2, who are laboratory staff trained on annotation tasks but are not biological domain experts. These initial segmentations are currently the first step in producing high-quality dense cellular segmentations, and even before any corrections they require 1-2 work days per cell to create. Results are displayed in Figure 4.7 row 3, with further details in Figures 4.11 and 4.9. Annotator 1, Annotator 2, and our algorithm each labeled the annotator comparison region from the Subject2 platelet sample. We calculated $\mathrm{MIoU}^{(org)}$ scores pairwise from the three segmentations: 0.571 for Annotator 1 vs. Annotator 2, 0.497 for Annotator 1 vs. Algorithm, and 0.483 for Annotator 2 vs. Algorithm. The confusion matrices in Figure 4.9 further break down results by organelle class. The statistics indicate that our algorithm disagrees more with either annotator than the annotators do with each other, but none of the labels are consistent everywhere, reflecting the difficulty of dense cellular segmentation even for humans.

Our final direct segmentation evaluation was on the binary task whose results were compared with CDeep3M. The 0.4 and 0.5 thresholds both produced the highest MIoU

score - 0.935. In contrast, the LCIMB segmentation had an MIoU of 0.946. Both algorithms generally did a good job of detecting cell material, but the LCIMB segmentation did a much better job of preserving boundaries between adjacent cells. The results can be seen in Figure 4.6.

We are also interested in understanding how even imperfect segmentations may be useful for downstream analysis tasks. To this end, we computed organelle volume fractions for each organelle within the cell in the annotator comparison dataset. The cell volume fraction of an organelle is equal to the summed voxels of all organelles in a cell, divided by the cell's volume. Biologists can correlate this information with other cell features to better understand variations in the makeup of cellular structures across large samples. The results in Figure 4.7 row 4 show that our algorithm tended to underestimate volume fractions relative to the two annotators, but the difference between the algorithm and Annotator 1 is smaller than the difference between Annotator 1 and Annotator 2. The best 3D algorithm improves considerably over the best 2D algorithm. All algorithms detect small regions ignored by humans, but simple postprocessing with small region removal fails to significantly improve quality metrics.

We have argued here that dense semantic labeling of 3D EM images for biomedicine is an image analysis method with transformative potential for structural biology. We demonstrated that while challenges exist for both human and algorithmic labelers, automated methods are approaching the performance of trained humans, and we plan to integrate them into annotation software for greatly enhancing the productivity of humans segmenting large datasets.

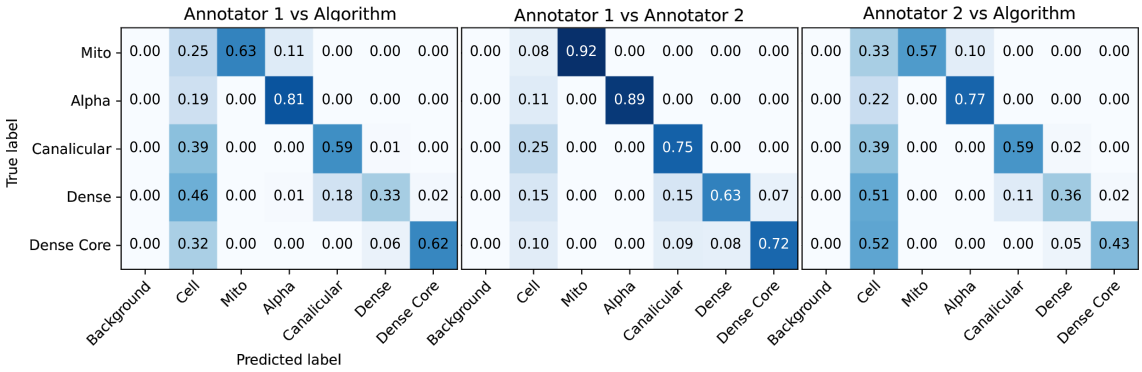


Figure 4.9: **Annotator comparison confusion matrices**. Confusion matrices comparing organelle labelings pairwise between the two annotators and our best algorithm. These give a more detailed performance breakdown of the $\mathrm{MIoU}^{(org)}$ scores obtained for each comparison: 0.497 for Annotator 1 vs Algorithm, 0.571 for Annotator 1 vs Annotator 2, and 0.483 for Annotator 2 vs Algorithm.

We have carefully evaluated adaptations of multiple common network architectures for our task, and demonstrated that a novel variant of 2D-3D fully convolutional network performs best. Without question, challenges remain for creating algorithms that are robust to the many types of variation present across research applications. However, SBF-SEM analysis problems are a fertile early ground for this computer vision research, as their large dataset sizes make the entire train-test-deploy cycle of supervised learning viable for accelerating analysis of even individual samples. The image in Figure 4.7(a-b) showcases this best – after manually segmenting less than 1% of the Subject 1 dataset, we were able to train a segmentation algorithm that produces a high-quality segmentation of the full dataset, a feat that would be impossible with anything short of an army of human annotators. While gains in accuracy will be realized with future developments, the procedure of training neural network ensembles on a manually annotated portion of a large SBF-SEM dataset is already becoming viable for making dense cellular segmentation a reality.

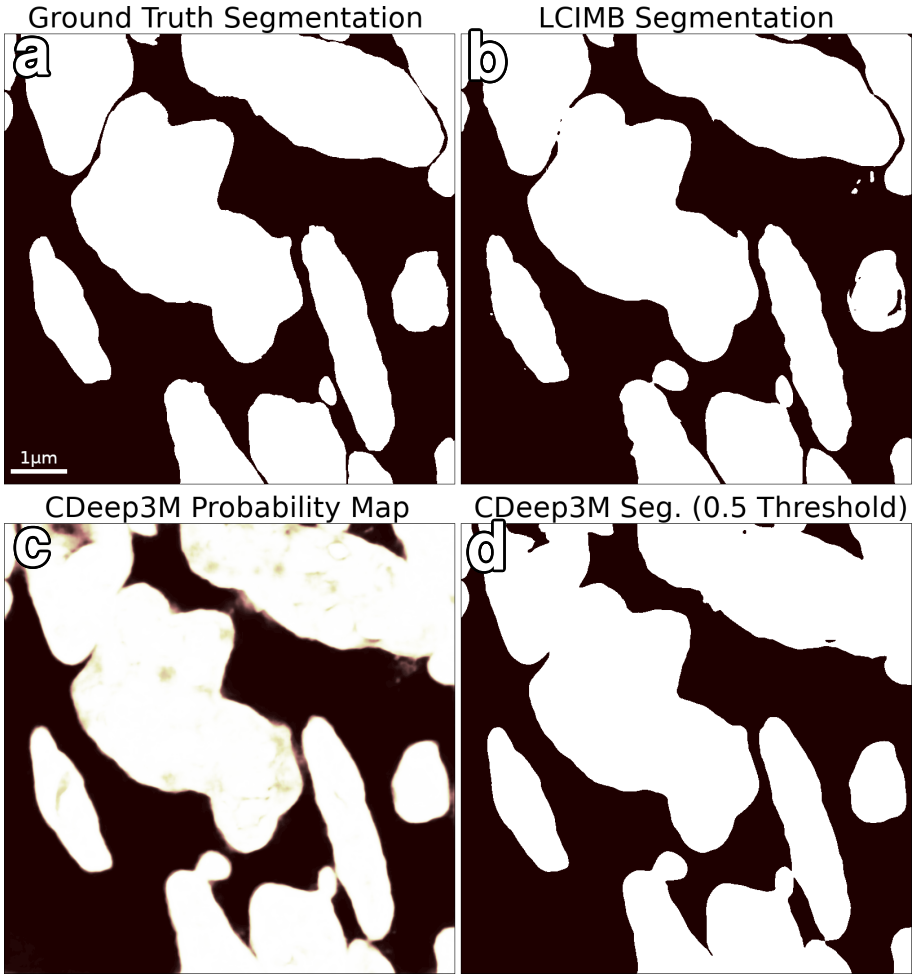


Figure 4.6: **CDeep3M segmentation comparison**. Comparison between the CDeep3M segmentation tool and our lab's (LCIMB) best segmentation algorithm for a binary cell/non-cell segmentation problem on our evaluation dataset. (**a**) Orthoslice of the ground truth binary segmentation of the evaluation dataset. (**b**) Segmentation using our lab's (LCIMB) best 3D ensemble. (**c**) Probability map produced by the CDeep3M ensemble after training on our data for 30000 iterations. The probability map is a per-voxel probability that the voxel belongs to a cell region, and it must be thresholded to produce a segmentation. (**d**) Segmentation from the CDeep3M ensemble with the best tested threshold of 0.5. This resulted in an MIoU of 0.935, compared to 0.946 for the LCIMB segmentation. In addition to a slight improvement in MIoU statistic, the LCIMB segmentation does a much better job of preserving boundaries between adjacent cells

4.5 Additional Material and Details

4.5.1 Segmentation Visualizations

In addition to the renderings already presented, we produced 3D renderings of segmentation results for the evaluation dataset, as shown in Figure 4.8, showing results for all organelles together as well as separately for the ground-truth labels, as well as the best 2D and 3D segmentation algorithms. Similarly, Figure 4.11 shows renderings per each organelle class for Annotator 1, Annotator 2, and our best algorithm on the annotator comparison dataset. Finally, Figure 4.10 shows 2D images of segmentations for each of the 14 algorithms tested in this work, which are also detailed in Table 4.2.

4.5.2 Ablation Analysis and Initialization-Dependent Performance

Our ablation analysis procedure, described in Section 4.4.1 confirms our conjectures about the importance of 3D context input to the network, and the importance of 3x3x3 convolutions over 1x3x3 convolutions for generalization performance. The latter do not capture correlations along the z spatial dimension, likely contributing to their poorer performance. Ablation analysis also indicates that removing either the multi-loss training setup or the 3D spatial pyramid module from the 2D-3D+3x3x3 architecture carries significant performance penalties. Removing either the 3x3x3 convolution layers or the 3D spatial pyramid on their own had a small effect on performance compared with removing the 2D loss term from the multi-loss objective function. Summary statistics demonstrating these results can be seen in Table 4.2(a), but these statistics only tell part of the story.

Especially when effect sizes are small, looking at a single trained instance of each architecture may not be enough to determine relative performance between architecture candidates. To get a better idea of the effects of different architecture choices, we must deal with the initialization-dependent performance of these segmentation networks.

In Figure 4.12 we experiment with various weight initialization random seeds to determine the robustness of various models to the weight initialization scheme. In order to determine whether one architecture choice is superior to another, the outputs of different trained networks are compared with each other. However, sources of randomness in the training process (initialization of trainable weights from a Xavier uniform distribution, and the random presentation order of training data elements) induce a distribution of final performance metric scores. These scores are random variables, and a single sample per architecture may be insufficient to determine which is better. By empirically approximating the distribution for each architecture, better inferences may be made about architecture design choices. For this figure, multiple instances of the same architecture (26 for 2D-3D and fully-3D nets, 500 for the U-Net) were trained under identical conditions, varying only random number generation seeds. The resulting distributions support the conclusions that 2D-3D networks outperform their 2D and fully-3D counterparts, as well as the conclusions drawn from the ablation studies. They also reveal a curious phenomenon that may be a topic for future study – the seemingly bimodal performance of 2D-3D architectures, wherein some fraction of trained instances perform markedly worse than others with an apparent performance gap between peaks. Whether this is a real phenomenon or an artifact of having an insufficient number of samples could be determined with a follow-up study.

4.5.3 DeepVess Baseline Comparison

In addition to the baseline models discussed above, we have also tried using the DeepVess model from [67] on our data. However, DeepVess performed poorly, and learned to assign a single class (background) to the entire output patch. There may be two reasons behind DeepVess' poor performance: (1) Unlike U-Net and Deeplab networks, the DeepVess network is designed with very small input patches in mind; small patches do not contain enough context for the network to distinguish between objects. (2) DeepVess' last layer consists of a fully-connected operation with a single hidden layer containing 1024 neurons, therefore any attempt to input significantly larger patches would require increasing the number of neurons in the last layer, but fully-connected layers do not scale well and the network quickly outgrows GPU memory.

4.5.4 Segmentation 3D Rendering Videos

In addition to the 3D rendering images of segmentations displayed in figures in this work, we produced videos showing rotations of the renderings.

4.5.4.1 Evaluation Dataset

Ground truth: `https://leapmanlab.github.io/dense-cell/vids/eval_gt.mp4`

Our best 3D ensemble: `https://leapmanlab.github.io/dense-cell/vids/eval_e-3d.mp4`

Our best 2D ensemble: `https://leapmanlab.github.io/dense-cell/`

vids/eval_e-2d.mp4

Our best 3D network: https://leapmanlab.github.io/dense-cell/vids/eval_s-3d.mp4

Our best 2D network: https://leapmanlab.github.io/dense-cell/vids/eval_s-2d.mp4

4.5.4.2 Test Dataset

Ground truth: https://leapmanlab.github.io/dense-cell/vids/test_gt.mp4

Our best 3D ensemble: https://leapmanlab.github.io/dense-cell/vids/test_e-3d.mp4

Our best 2D ensemble: https://leapmanlab.github.io/dense-cell/vids/test_e-2d.mp4

Our best 3D network: https://leapmanlab.github.io/dense-cell/vids/test_s-3d.mp4

Our best 2D network: https://leapmanlab.github.io/dense-cell/vids/test_s-2d.mp4

4.5.4.3 Annotator comparison dataset

Annotator 1: https://leapmanlab.github.io/dense-cell/vids/ac_ann1.mp4

Annotator 2: https://leapmanlab.github.io/dense-cell/vids/ac_

ann2.mp4

Our best algorithm: https://leapmanlab.github.io/dense-cell/vids/ac_alg.mp4

4.5.5 Training Demonstration Videos

We trained a 2D-3D+3x3x3 network for 39744 iterations, recording the class prediction probability maps and segmentation that the network produced on the evaluation dataset every 92 iterations. We produced animations of the evolution of the network's prediction capabilities to demonstrate the learning process.

4.5.5.1 Probability Maps Video

The first video shows the evolution of the six non-background probability maps predicted by the network over the course of training. Each probability map is color-coded based on the corresponding structure color in the segmentation color scheme used throughout this work - dark green for cell, magenta for mitochondrion, dark blue for alpha granule, yellow for canalicular channel, bright red for dense granule, and dark red for dense granule core.

Link: https://leapmanlab.github.io/dense-cell/vids/train_prob-maps.mp4

4.5.5.2 Segmentation Video

The second video shows the evolution of the segmentation produced by the network over the course of training.

Link: `https://leapmanlab.github.io/dense-cell/vids/train_seg.mp4`

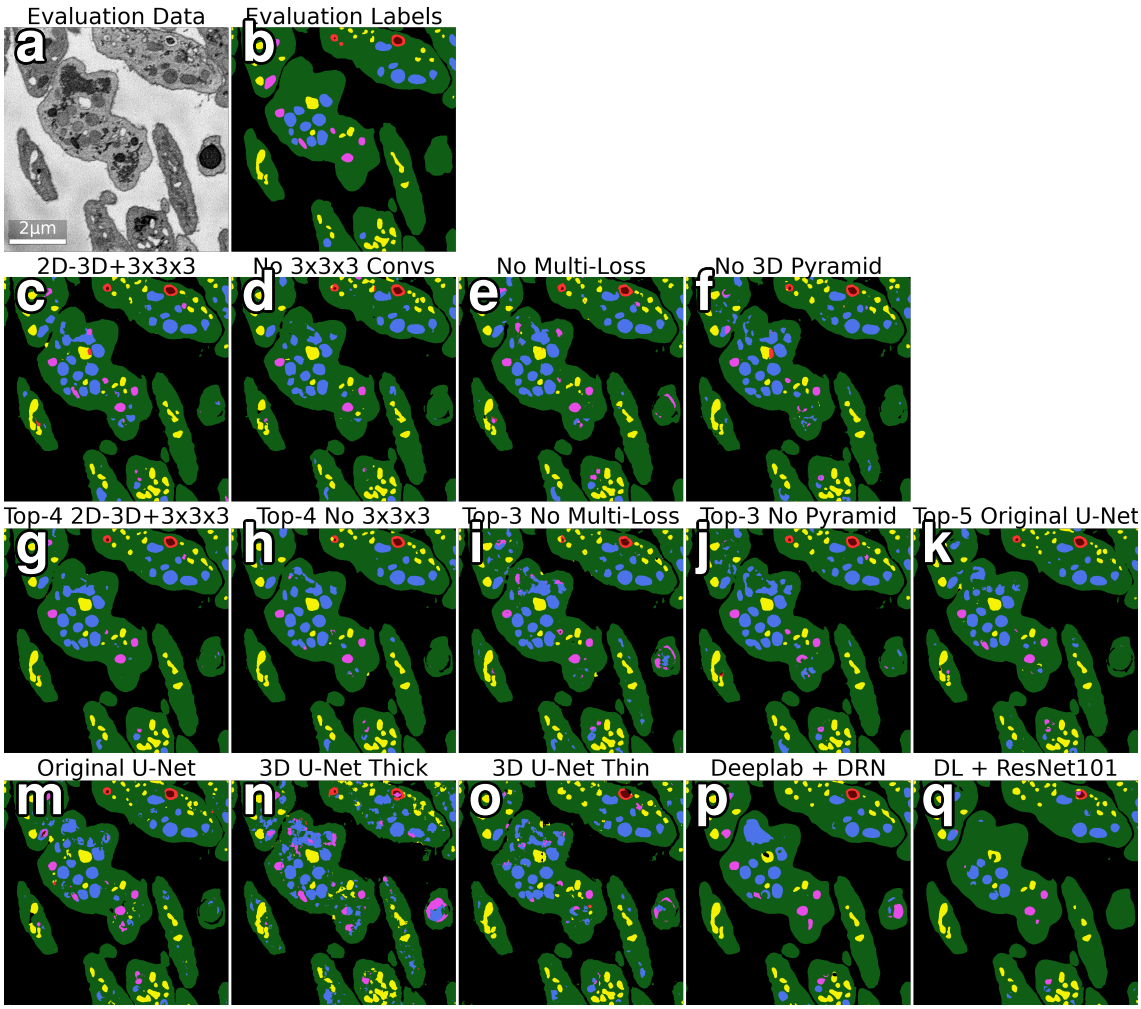


Figure 4.10: **2D comparison of all algorithm results**. This Figure compares the results of all 14 segmentation algorithms tested in this work with ground-truth labels for the $z = 4$ slice of the evaluation dataset. (**a**-**b**) Orthoslice of the evaluation image dataset and segmentation. (**c**-**f**) Segmentations from our new 2D-3D+3x3x3 network and its three ablations. (**g**-**k**) Segmentations from the five ensemble algorithms tested in this work. (**m**-**q**) Segmentations from the five baseline networks tested in this work.

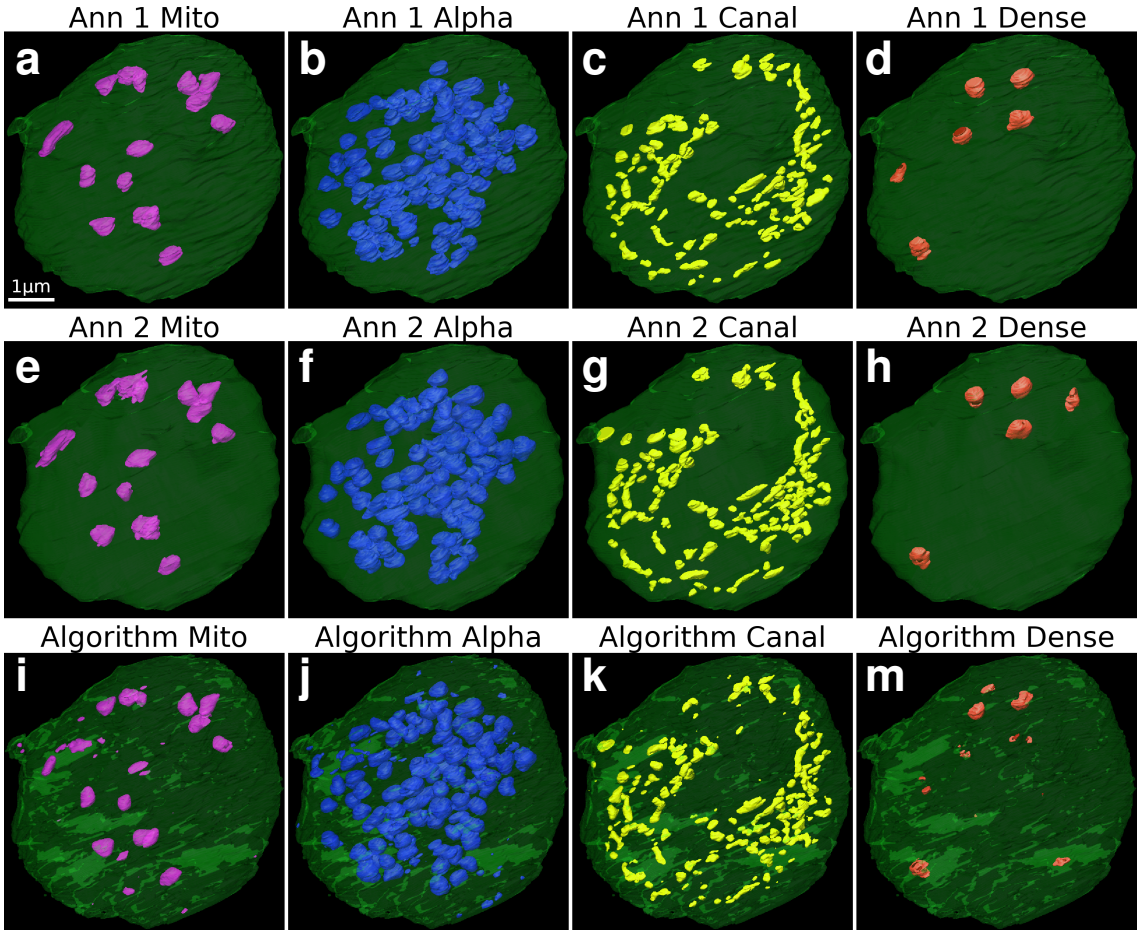


Figure 4.11: **Annotator comparison segmentation renderings**. This Figure supplements row 4 of Figure 3 by showing renderings of individual organelle types - Mito for mitochondria, Alpha for alpha granules, Canal for canalicular channels, Dense for dense granules - from the Annotator 1, Annotator 2, and best Algorithm (Top-4 2D-3D+3x3x3) segmentations. (**a**-**d**) Annotator 1 (Ann 1) organelle segmentations. (**e**-**h**) Annotator 2 (Ann 2) organelle segmentations. (**i**-**m**) Algorithm organelle segmentations.

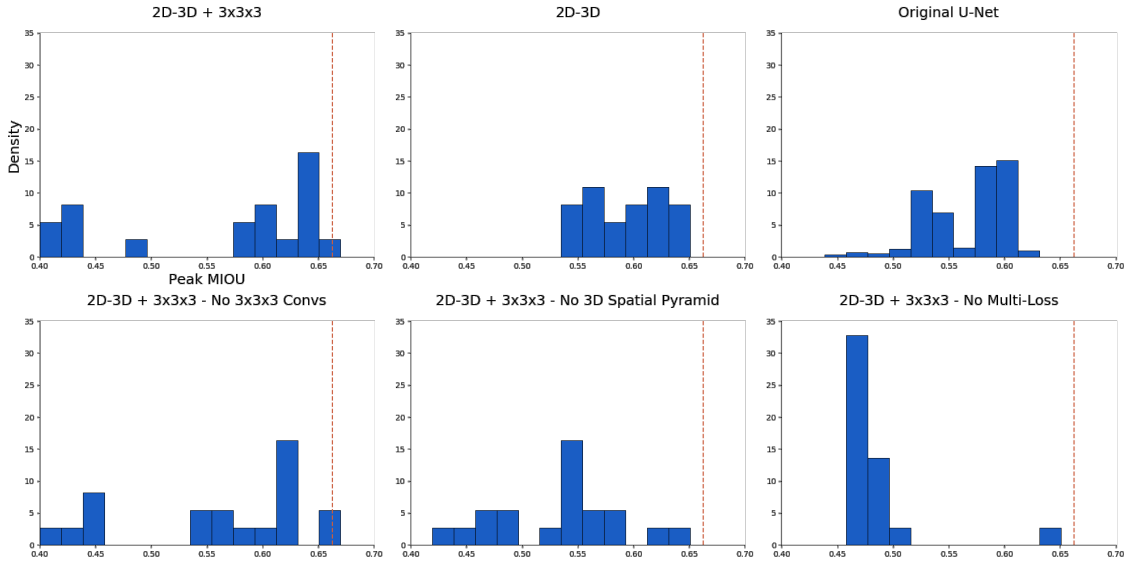


Figure 4.12: **Making better architecture design decisions**. This Figure shows normalized histograms of peak $\mathrm{MIoU}^{(all)}$ on the evaluation dataset for each of the architectures examined in this work. In order to determine whether one architecture choice is superior to another, the outputs of different trained networks are compared with each other. However, sources of randomness in the training process (initialization of trainable weights from a Xavier uniform distribution, and the random presentation order of training data elements) induce a distribution of final performance metric scores. These scores are random variables, and a single sample per architecture may be insufficient to determine which is better. By empirically approximating the distribution for each architecture, better inferences may be made about architecture design choices. For this figure, multiple instances of the same architecture (26 for 2D-3D nets, 500 for the U-Net) were trained under identical conditions, varying only random number generation seeds. The resulting distributions support the conclusions that 2D-3D networks outperform the 2D U-Net and that multi-loss training is necessary for 2D-3D architectures.

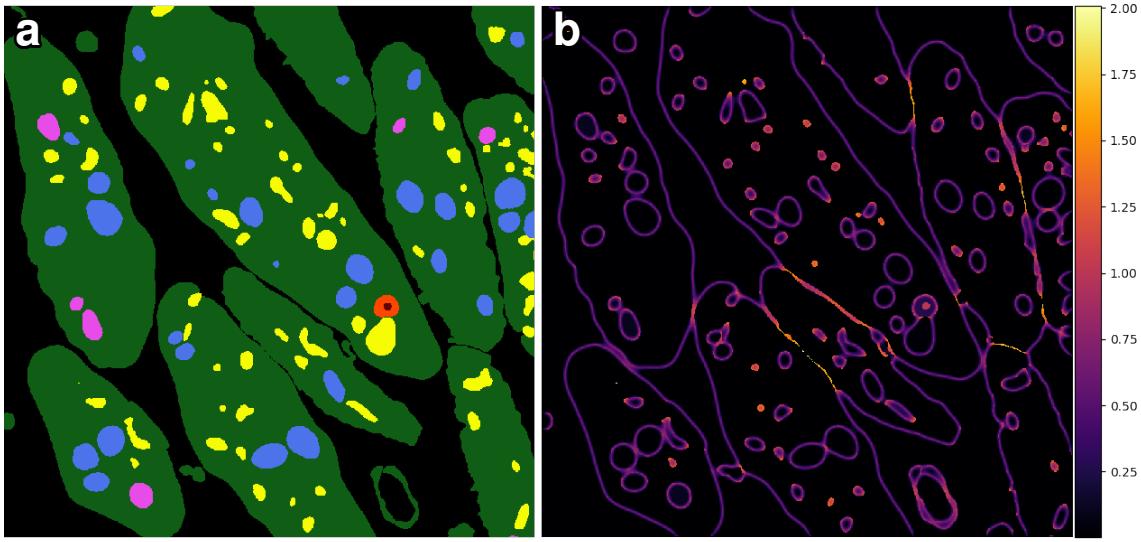


Figure 4.13: **Error weighting array visualization**.The error weighting $\mathcal{W}$ array is the sum of three terms $w + \mathcal{W}_{cb} + \mathcal{W}_{ep}$, where w is a weight floor, $\mathcal{W}_{cb}$ is a class frequency balancing array, and $\mathcal{W}_{ep}$ is an edge preserving array. $\mathcal{W}_{cb}$ and $\mathcal{W}_{ep}$ are computed from ground truth labels. (**a**) Example orthoslice of training dataset ground truth labels. (**b**) Corresponding orthoslice of the error weighting array.

4.5.6 Source Code

Supplementary materials, source data, code, and reproducible examples are available online at `https://leapmanlab.github.io/dense-cell`.

Chapter 5: Active Learning

The machine learning (ML) community has been working on curating novel benchmark labeled datasets. These benchmark datasets are essential in driving new developments because they allow researchers to quickly compare their new methods against previous work. DNNs require enormous amounts of data to train effectively and generalize robustly in large scale CV settings; however, compiling such datasets is often cost prohibitive. Nonetheless, in some industries, notably the autonomous vehicle industry, DNNs are deployed at scale in mission critical scenarios, with models continually improving as they train on a never ending stream of data. Vehicles operate as a fleet and each vehicle can collect images and video snippets from their surroundings to send back to centralized servers. The data is manually annotated and added to the training set, which is later used to refine their deep learning models. However, collecting and labeling all data points is prohibitively expensive and time consuming. To remedy this issue, the fleet uses active learning (AL), a set of machine learning algorithms that help users decide which raw unlabeled data is most informative, and therefore worth labeling. By employing AL, the fleet is selectively collecting "interesting" videos from its surroundings and storing them to be labeled and used as training data for improving the ML models. AL is especially impactful in industrial scale settings where data labeling costs are high and practitioners

use every available tool to improve model performance. As such, AL constitutes a viable solution to alleviate the data annotation costs in the semantic segmentation feedback loop described in Section 4.3.3. By carefully selecting the next patch to be presented to the annotators for manual correction, we can achieve higher neural network (NN) performance. Unfortunately, AL for semantic segmentation of images is difficult to achieve. Here, we start by developing an AL algorithm for natural image classification.

The recent success of self-supervised pretraining (SSP) highlights the importance of harnessing abundant unlabeled data to boost model performance. By combining AL with SSP, we make use of unlabeled data while simultaneously labeling and training on particularly informative samples. We study a combination of AL and SSP on ImageNet. We find that performance on small toy datasets – the typical benchmark setting in the literature – is not representative of performance on ImageNet due to the class imbalanced samples selected by an active learner. Among the existing baselines we test, popular AL algorithms across a variety of small and large scale settings fail to outperform random sampling. To remedy the class-imbalance problem, we propose Balanced Selection (BASE), a simple, scalable AL algorithm that outperforms random sampling consistently by selecting class balanced samples for annotation. We summarize our contributions below:

1. We demonstrate that the performance of popular AL methods, which has been observed on small datasets, does not transfer to the larger and more complex ImageNet challenge. In fact, on the common linear evaluation task (see Section 5.3), most popular AL algorithms perform worse than random sampling on ImageNet.
2. We identify and study sampling imbalance as a major failure mode for AL algorithms.

Because ImageNet has many classes with highly heterogeneous properties, AL algorithms have a tendency to heavily sample from preferred classes while nearly ignoring others. This problem is less severe on simple tasks with fewer and more homogeneous classes.

3. We introduce the Balanced Selection (BASE) AL strategy. BASE efficiently selects images that lie near class boundaries in feature space while also promoting an even class distribution. By carefully selecting which data to label, BASE achieves significantly better sample efficiency than standard self-supervised learning (SSL) pipelines that rely on random sampling.

4. We show, for the first time, that AL can offer performance boosts on ImageNet when combined with SSL. Our BASE algorithm, when used to train a classifier on top of a SSL feature extractor, matches the top-5 accuracy of the state-of-the-art EsViT [90] SSL algorithm while using only 55% of the labels.

This is joint work with Hong-Min Chu, Ping-Yeh Chiang, Wojciech Czaja, Richard Leapman, Micah Goldblum, and Tom Goldstein [145]. My contribution was conceiving the project, designing and writing the majority of the code base, running the majority of experiments, and writing the paper.

5.1 Introduction

Fueled by the success of deep learning, the global data annotation market is projected to reach $\$3.4$ billion by 2028 [117]. The data labeling process is a daunting hurdle for institutions aiming to deploy deep learning models at an industrial scale. The annotation

process is slow, costly, and in some cases requires domain-expert annotators.

A large body of machine learning research seeks to reduce data labeling costs by harnessing as much information as possible directly from unlabeled data or by leveraging other labeled datasets whenever possible. Ultimately, however, labeled data is required to achieve adequate deep learning model performance, especially in mission critical scenarios. Due to time and budget constraints, practitioners are often restricted to selecting a small subset of the available data for annotation. This restriction raises the following question: What is the best approach for selecting this subset?

Active learning (AL) is a subfield of machine learning (ML) dedicated to answering this question. Given a large pool of unlabeled data and a fixed labeling budget, an AL algorithm selects a subset of the unlabeled data to be annotated. Once labeled, the subset is subsequently used to train a ML model. The goal of the active learner is to select the subset that will optimize the generalization performance of the ML model. AL as a field predates deep neural networks (DNN), however, naively applying classical AL methods to DNNs is not straightforward and sub-optimal.

In this work, we focus on classification tasks using DNN. We study a combination of AL and self-supervised pretraining (SSP) in the large data regime. Large-scale data is prevalent in real-world scenarios, where unlabeled data is typically abundant and cheap to collect. Furthermore, in real-world settings, practitioners are compelled to leverage the available unlabeled data in order to achieve adequate model performance at the lowest possible annotation cost. State-of-the-art SSP methods can provide these performance boosts at no annotation cost.

Prior research on AL focuses on the CIFAR-10, CIFAR-100, and SVHN [83, 156]

datasets to compare AL algorithms. However, it is unclear whether performance on these datasets is predictive of performance on real-world datasets that are orders of magnitude larger and that may contain many more classes or even imbalanced data. We focus particularly on ImageNet [124], as it contains 1000 classes, 1.2 million images, and a significant amount of label noise [14, 141, 149] as is common in industrial settings [95]. AL cost savings are much more impactful at the ImageNet scale and beyond, and cannot be understood by studying small datasets alone. These cost savings are due in part to the sheer amount of available data but also the ambiguity of the classes considered. To curate ImageNet, each image was presented to multiple human annotators who voted until a consensus was reached on the label [33]. This voting mechanism translates directly to high annotation costs.

Finally, we specifically focus on the interaction of AL with SSP. SSP has been shown to provide a significantly larger accuracy boost compared to AL alone [136]. It is therefore important to study whether AL offers any additional benefits on top of SSP. We present the first AL results on ImageNet using SSP.

We make our code publicly available in the hopes that others can easily reproduce our results and use our codebase for future AL research [1].

5.2 Background & Related Work

A typical AL algorithm cycles between learning from a small amount of labeled data, using the model to gather information about the unseen unlabeled data, and using this information to choose a subset of the unlabeled data to be manually annotated. This

[1] https://github.com/zeyademam/active_learning

cycle then repeats, with the labeled dataset increasing in size at every iteration, until a pre-determined manual-annotation budget is exhausted. In this section, we provide a formal description of the AL problem, followed by an overview of existing methods.

We will adopt the notation from [130] with slight modifications to accommodate more general cases. We will study a C-way classification problem defined over a compact space $\mathcal{X} = \mathbb{R}^d$ and a finite label set $\mathcal{Y} = \{1, \ldots, C\}$. We denote the loss function by $l(\cdot, \cdot, \mathbf{w}) : \mathcal{X}, \mathcal{Y} \to \mathbb{R}$, where $\mathbf{w}$ are the parameters (i.e ., the weights) of a classifier $f(\mathbf{w}, \cdot) : \mathcal{X} \to \mathcal{Y}$, which we simply denote as $f(x)$ in the rest of the Chapter.

The entire dataset is a collection of n points $Z \subseteq \mathcal{X} \times \mathcal{Y}$ sampled *i.i.d.* over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ as $\{x_i, y_i\}_{i \in [n]} \sim p_{\mathcal{Z}}$. Initially, some subset of m points is assumed to have been annotated by an expert, we will denote the indices of those points by $s^0 = \{s^0(j) \in [n]\}_{j \in [m]}$.

An AL algorithm has access to $\{x_i\}_{i \in n} \subseteq \mathcal{X}$ but only the labels with indices s^0, i.e., $\{y_{s^0(j)}\}_{j \in [m]}$. The algorithm is also given a budget b of queries to ask an oracle (typically a human annotator), and a learning algorithm A_s which outputs a set of parameters $\mathbf{w}$ given $\{x_i\}_{i \in n} \subseteq \mathcal{X}$ and $\{y_{s(j)}\}_{j \in [m]}$. The goal of AL is to identify a new subset s^1 of unlabeled data such that:

$$s^1 = \underset{s^1 : |s^1| < b}{\arg\min} \; \mathbb{E}_{x,y \sim p_{\mathcal{Z}}} \left[l(x, y; A_{s^0 \cup s^1}) \right]. \tag{5.1}$$

The above formulation constitutes one round of active learning. Typically, the algorithm runs for K rounds producing a sequence of subsets $s^1, s^2 \ldots, s^K$ to be labeled by the oracle and added to the labeled dataset for the following round. We will denote by D_L^k all the indices of points labeled before the start of round k, i.e., $D_L^k = \bigcup_{i=0}^{k-1} s_i$, and likewise

$D_U^k = [n] \setminus D_L^k$ is the set of all unlabeled points at round k. We will use $|\cdot|$ to refer to the cardinality of a set.

5.2.1 Limitations of Classical Active Learning

As mentioned in the introduction, AL predates modern DNNs and many active learning algorithms were designed without DNNs in mind. In this Section, we list some of the shortcomings of classical AL and its incompatibility with DNNs.

i. **Fixed feature space representation:** Some algorithms assume a fixed representation of the data [93] in feature space. This assumption is sound for some classification algorithms, such as support vector machines, however, the features of DNNs change during the training process.

ii. **b=1:** Most classical algorithms assume the budget at each round to be exactly 1. However, each round requires retraining the DNN from scratch. This presents two issues: first, a single point will have no noticeable effect on the parameters because of the local optimization usually applied to DNNs, and second, retraining a DNN for scratch at each round can be very costly.

iii. **Uncertainty measures:** Almost all AL algorithms rely on some measurement of the uncertainty of predictions made by the classifier on the unlabeled data. For some ML algorithms (e.g., support vector machines) these methods are straightforward and have theoretical justification. However, in the case of DNNs, there is no clear interpretation of the softmaxed logits (i.e., output of the network).

5.2.2 Selection Methodologies

Existing AL algorithms for DNNs can be roughly broken down into two categories: those designed to tackle class imbalanced datasets and those that are not. The wide majority of existing algorithms fall in the latter category.

Algorithms designed for balanced datasets can be further broken down into two categories: uncertainty based sampling and density based sampling [1]. Uncertainty based AL algorithms operate by first quantifying the classifier's uncertainty about its prediction on every unlabeled sample at round k [49, 50, 131], then querying the examples on which the classifier is deemed most uncertain. Prediction entropy, least confidence, and margin sampling are commonly used uncertainty measures. Intuitively, uncertainty based sampling improves the model's prediction on subsets of the domain $\mathcal{X}$ where the model is most uncertain, and in turn, improves the model's generalization ability.

On the other hand, density based algorithms [6, 130] generate high-dimensional features from the data then select examples with the most representative features. In practice, this selection involves running a clustering algorithm [6, 28] or finding Coresets [130] in feature space. The features can be obtained by removing the network's linear classification head [28, 130] or by taking its gradients with respect to every sample [6]. Intuitively, density based sampling ensures that the most densely populated regions of space, which contain the most data at test time, are represented in the labeled set.

A smaller minority of AL algorithms specifically tackle class imbalanced data [1]. However, as we discuss below, techniques in this area cannot scale to ImageNet.

5.2.3 Scaling Ability

AL strategies designed for classical machine learning (ML) focus on querying a single label (i.e., $b = 1$), re-training the model on all labeled data, querying the next example, retraining again, etc. This has obvious advantages as the active learner is given access to the current label and can therefore use it to guide its selection strategy. However, as datasets increased in size, ML algorithms became costly to train, and data annotation grew into an entire industry. It is now necessary for AL algorithms to operate at a large scale.

To be practical for neural network applications, AL strategies must be able to query a large batch of data at once, receive labels for the entire batch, then query another batch (i.e., $b >> 1$). This batch AL approach minimizes costs and time by keeping a large group of annotators occupied and by reducing the number of training runs needed to update DNNs on newly acquired data. At the same time, modern AL strategies must be able to efficiently process a massive pool of unlabeled data at each round (i.e., large D_U^k).

In Table 5.1 and Section 5.5.2.1, we discuss the time complexities of several baselines considered in this work in greater detail.

Table 5.1: Time complexities. d' is the feature space dimension. On ImageNet using ResNet-50, $|D_U^k| \approx 10^6$, features $d' = 2048$, gradient embeddings $d'' = 2048 \times 10^3$, and $C = 10^3$. In our experiments, $b = 10^3$.

Algorithm	Time Complexity
Margin Sampler	$\mathcal{O}(C \cdot \log(b) \cdot \lvert D_U^k \rvert)$
Confidence Sampler	$\mathcal{O}(C \cdot \log(b) \cdot \lvert D_U^k \rvert)$
Approx. Coreset	$\mathcal{O}((b + \lvert D_L^k \rvert) \cdot d' \cdot \lvert D_U^k \rvert)$
[130] BADGE	$\mathcal{O}(C \cdot (b + \lvert D_L^k \rvert) \cdot d'' \cdot \lvert D_U^k \rvert)$
[6] Balancing Sampler	$\mathcal{O}(C \cdot b \cdot d' \cdot \lvert D_U^k \rvert)$
[1] BASE (ours)	$\mathcal{O}(C \cdot (d' + \log(b)) \cdot \lvert D_U^k \rvert)$

5.2.3.1 Usage of Unlabeled Data

Unlabeled data is often abundant in real-world scenarios, and leveraging it effectively can lead to significant reductions in data annotation costs. In [19, 136], the authors study the benefits of AL when combined with both SSP and FixMatch, a semi-supervised technique. In this work, we restrict our experimental setup to SSP because semi-supervised techniques would require training to saturation multiple times on the entire Imagenet dataset [138], which is prohibitively expensive. In Table 5.2, we show that using only 25% of the ImageNet labels, the MoCo v2 SSP method with randomly sampled labels is able to boost model performance by 18.15 percentage points, whereas carefully selecting examples using the VAAL [137] AL algorithm only boosts performance by 1.5 percentage points if the classifier's weights are randomly initialized. Clearly, initializing models with SSP offers strong advantages in the label scarce regime. In Section 5.5, we show for the first time that AL offers additional performance gains on top of SSP at the ImageNet scale.

To the best of our knowledge, only two prior works [13, 137] study AL on ImageNet, and none study the interaction between AL and SSP at this scale.

Table 5.2: Performance gains offered by VAAL [137] on ImageNet vs those offered by MoCo v2 [25], a popular SSP method. SSP, when combined with random sampling, yields a substantially larger boost in performance.

% of labels	SSP	Strategy	Accuracy
25%	No	Random	50%
	No	VAAL	+1.5%
	Yes	Random	+16.65%

5.3 Linear Evaluation Task

When DNNs are deployed at an industrial scale, it is common to use a shared backbone network for feature extraction, then apply separate heads directly on the features to accomplish different downstream tasks. A single shared backbone is easier to maintain and carries a small memory footprint, making it more practical for edge devices. Training this entire pipeline end-to-end is time consuming and computationally intensive. Therefore, in this setting, as new labeled data becomes available, the different task-specific heads are frequently finetuned while keeping the backbone frozen. The backbone itself is updated less frequently[2]. In fact, with the emergence of massive foundation models [15], such as GPT-3 [16], BERT [35], and DALL-E [115], practitioners may only have restricted access to the backbone. Consequently, it is important to evaluate AL algorithms when the feature extractor is fixed and only the classification head is finetuned. This task is a common benchmark in self-supervised learning (SSL) research [24, 25, 90], where the proposed SSL method is used to pretrain the network's feature extractor, then a linear classification head is trained on the features in a fully-supervised fashion. However, to the best of our knowledge, AL strategies on ImageNet have not been evaluated in this specific setting.

[2]The quintessential manifestation of this framework is described in Tesla's AI day: https://youtu.be/j0z4FweCy4M?t=3300 (55th minute).

5.4 Methods

We now describe our proposed method. We start with a simple preliminary variant of uncertainty based selection. We then show how this variant can be adapted to prevent sampling imbalances from emerging and accumulating over rounds, resulting in improved performance.

5.4.1 Margin Selection

We first introduce a simplified variant of our Balanced Selection algorithm, which we call Margin Selection (MASE). MASE selects the b examples closest to any decision boundary at every round of AL. Intuitively, these samples should have the most influence on the decision of the model. We define distance to decision boundary (DDB) as follows,

$$\text{DDB}(x) = \min_{\epsilon} ||\epsilon||_2 \quad \text{s.t.} f(x+\epsilon) \neq f(x). \tag{5.2}$$

When f is a DNN, DDB is expensive to compute in input space. We instead estimate this distance in feature space. For the models considered in this work, the final layer is a linear classification head on top of the features produced by a feature extractor; therefore, computing DDB in feature space reduces to computing the projection of the feature vector onto the normal vector of the linear decision boundary, which can be implemented very efficiently. We provide pseudocode for MASE in Algorithm 1.

To the best of our knowledge, MASE is a novel AL strategy, similar algorithms were only studied on 2-class classification tasks using support vector machines [42], or a

different definition of distance [26].

5.4.2 Balanced Selection

In Section 5.5, we show that only a single baseline AL algorithm, namely Partitioned BADGE (vi), beats random sampling on the linear evaluation task on ImageNet and only by a relatively small margin. This is partially due to the imbalance induced by the active learner, which does not query examples evenly across classes. Motivated by this observation, we design Balanced Selection (BASE), an AL strategy capable of scaling efficiently while also querying a balanced batch of examples. As opposed to naïvely choosing the examples with the smallest DDB, BASE selects examples based on their distance to class specific decision boundaries (DCSDB), defined as

$$\text{DCSDB}(x, c) = \begin{cases} \min_\epsilon ||\epsilon||_2 \quad \text{s.t.} \;\; f(x+\epsilon) = c \;\; \text{if} \;\; f(x) \neq c \\ \min_\epsilon ||\epsilon||_2 \quad \text{s.t.} \;\; f(x+\epsilon) \neq c \;\; \text{if} \;\; f(x) = c. \end{cases} \tag{5.3}$$

More specifically, for each class $c \in \{1, \ldots, C\}$, BASE selects the b/C samples with the smallest DCSDB(x, c). Similar to our MASE implementation, we only consider distances in feature space. We provide a visual illustration of BASE's selection strategy in Figure 5.1 and its pseudocode in Algorithm 2. The time complexity of Algorithm 2 is dominated by computing DCSDBs, but those can be computed once and stored, so the algorithm runs in $\mathcal{O}(C \cdot (d' + \log(b)) \cdot |D_U^k|)$ in practice, where d' is the dimension of the features. This puts BASE on par with the fastest baselines – see Table 5.1.

Algorithm 1 Algorithm for MASE

Input: Query Budget b, Indices of Labeled Samples D_L^k
$s \leftarrow D_L^k$
while $|s| < |D_L^k| + b$ **do**
  $x^* \leftarrow \arg\min_{x \in [N] \setminus s} \text{DDB}(f(x))$
  $s \leftarrow s \cup \{x^*\}$
end while
return $s \setminus D_L^k$

Algorithm 2 Algorithm for BASE

Input: Query Budget b, Number of Classes C, Indices of Labeled Samples D_L^k
$s \leftarrow D_L^k$
while $|s| < |D_L^k| + b$ **do**
  for c in $[C]$ **do**
    $x^* \leftarrow \arg\min_{x \in [N] \setminus s} \text{DCSDB}(f(x)), c)$
    $s \leftarrow s \cup \{x^*\}$
    if $|s| = |s^0| + b$ **then**
      break
    end if
  end for
end while
return $s \setminus D_L^k$

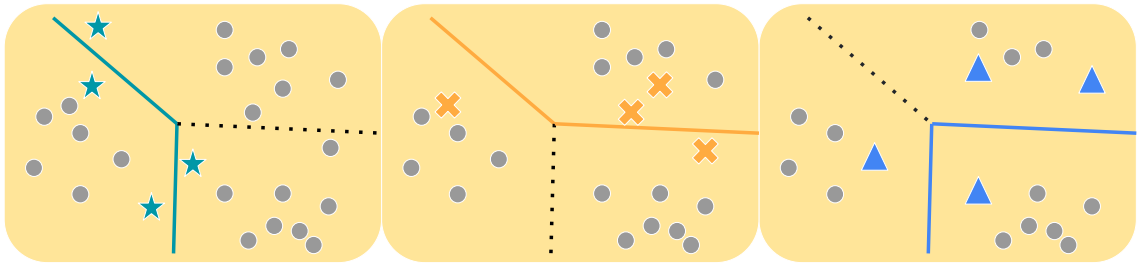


Figure 5.1: An illustration of BASE for 2-dimensional features and a 3 class problem. The algorithm selects an equal number of points (shown using colored stars, crosses, and triangles) that are closest to each decision boundary (solid lines).

5.5 Experiments

In this Section, we outline our experimental design, followed by a presentation of our results. In Figure captions, we will refer to different experiments using a capital letter for the dataset/model combination, and a capital roman numeral for the experimental setup. For example, setting A-I refers to AL strategies tested on CIFAR-10 using a ResNet-18 with the model weights initialized using SSP at every round. Baseline strategies are referenced using lower case roman numerals, e.g., v refers to BADGE. Below, we enumerate each dataset/model combination, training setup, and AL method, and assign each a letter or numeral.

5.5.1 Datasets and Models

In our experiments, we use the following dataset and model architecture combinations.

A. **CIFAR-10 [83] w/ ResNet-18 [70].**

B. **Imbalanced CIFAR-10 [83] w/ ResNet-18 [70]:** The number of samples per class decreases exponentially from the most frequent class to the least frequent class; the

most sampled class contains $10\times$ the number of samples in the least sampled class [17].

C. **ImageNet [124] w/ ResNet-50 [70].**

D. **ImageNet [124] w/ ViT [37].**

5.5.1.1 Training Setups

We consider two different settings for training the classifier.

I. **End-to-end finetuning from a self-supervised checkpoint.** We first train the network using self-supervised learning on all available unlabeled data. At every round k of AL, the backbone's weights (all layers except the final linear classifier) are reset to the SSP weights then the network is finetuned end-to-end on all the available labeled data D_L^k.

II. **Linear evaluation from a self-supervised checkpoint .** Here, we employ SSP. At every AL round k, we use the SSP checkpoint, but we only update the final linear layer of the network on labeled data D_L^k.

5.5.2 Baselines

We compare BASE (ours) to the following baselines.

i. **Random Sampler.** Queries samples from D_U^k uniformly at random.

ii. **Balanced Random Sampler.** Iterates over classes and chooses an equal number of examples uniformly at random from each class. *This baseline strategy cheats, as it*

requires the labels for points in D_U^k in order to make its selection. We include it only for scientific purposes.

iii. **Coreset AL. [130]** We solve the k-center problem using the classical greedy 2-approximation.

iv. **Partitioned Coreset Sampler. [28]** Partitions the dataset into p partitions, then runs the Coreset algorithm to select b/p examples from each partition. This implementation only calculates pairwise distances on smaller subsets, which is more computationally efficient.

v. **BADGE AL [6].** Calculates the gradient with respect to the last linear layer, then applies the K-means++ seeding algorithm [5] on the gradients. On ImageNet, the size of the gradient embedding is proportional to the number of classes, which makes it $100\times$ larger than CIFAR-10 embeddings. Furthermore, the K-means++ seeding algorithm requires the pairwise distances, which again is computationally prohibitive on ImageNet.

vi. **Partitioned BADGE Sampler. [28]** BADGE with a similar partitioning trick as Partitioned Coreset, and global pooling to reduce the embedding dimension.

vii. **Confidence Sampler.** Selects the examples with the smallest top logit (least confidence).

viii. **Margin Sampler[128].** Selects examples with the smallest differences between the top logit and the second largest logit (minimum margin).

ix. **VAAL [137].** Trains a binary classifier to distinguish between features produced by labeled vs unlabeled samples. The features are obtained by training a variational autoencoder. Queries the unlabeled samples that the binary classifier is most confident

about.

x. **Balancing Sampler** [1] Calculates cluster centers for each class in feature space, then targets the class with the least number of queried examples, and finally selects examples that are close to the target class' center and away from other clusters.

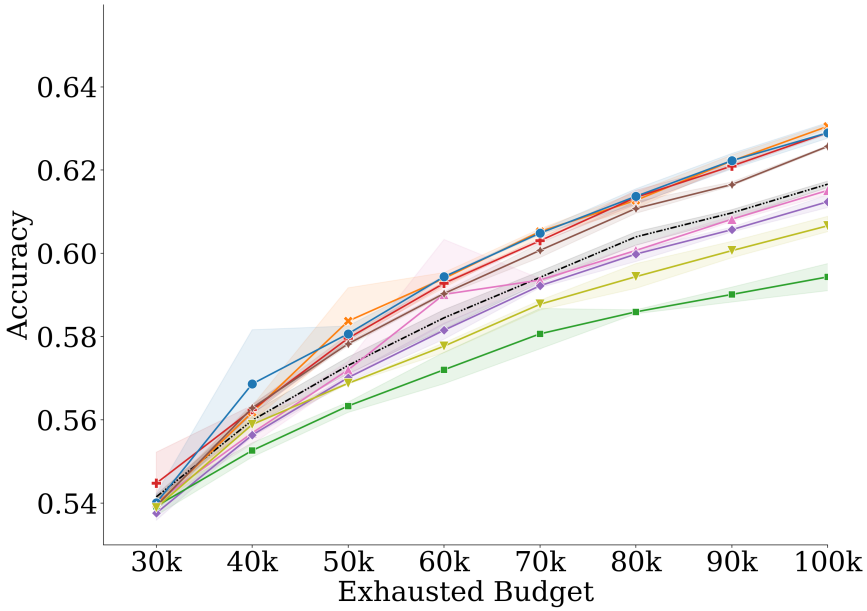
xi. **MASE (ours).** See Section 5.4.

Table 5.3: Setting C-I. Average ImageNet accuracy over 3 runs obtained by training a ResNet-50 end-to-end starting from a SSP checkpoint at every AL round.

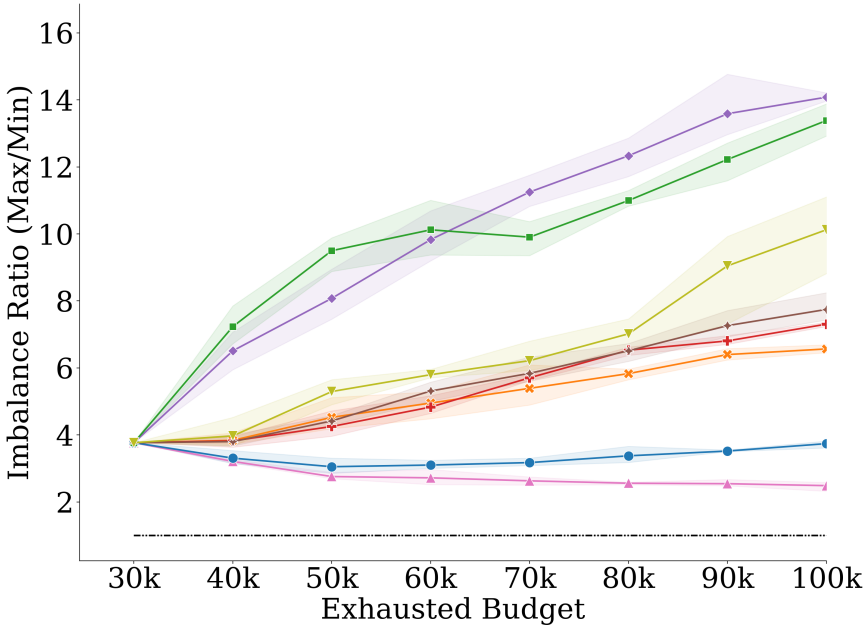
Strategy/Budget	30000	40000	50000	60000	70000	80000	90000	100000
Partitioned Coreset Sampler	0.539	0.553	0.563	0.572	0.581	0.586	0.59	0 .594
Partitioned BADGE Sampler	0.539	0.563	0.578	0.59	0.601	0.611	0.617	0 .626
BASE	0.54	0.569	0.581	0.594	0.605	0.614	0.622	0 .629
Balanced Random Sampler	0.542	0.56	0.573	0.585	0.594	0.604	0.61	0 .617
Confidence Sampler	0.538	0.556	0.57	0.582	0.592	0.6	0.606	0 .612
MASE	0.54	0.562	0.584	0.594	0.605	0.613	0.622	0 .63
Margin Sampler	0.545	0.562	0.58	0.593	0.603	0.613	0.621	0 .629
Random Sampler	0.54	0.557	0.572	0.59	0.594	0.601	0.608	0 .615
VAAL Sampler	0.539	0.559	0.569	0.578	0.588	0.594	0.601	0.607

5.5.2.1 Scalability of baseline methods

Coreset (iii) and BADGE (v) are prohibitively expensive to run at the ImageNet scale – see Table 5.1. Additionally, both algorithms require storing large tensors in memory – $O(d' \cdot |D_U^k|)$ space complexity – as they require solving an optimization problem in feature space. For this reason, we exclude them from our comparisons, and instead implement Partitioned Coreset (iv) and Partitioned BADGE (vi), two scalable variants of the original strategies [28]. We also note that the Balancing Sampler (x) is not a batch AL algorithm as it acquires labels one at a time, making it impractical in terms of both computation and human labeling bandwidth in large-scale settings.

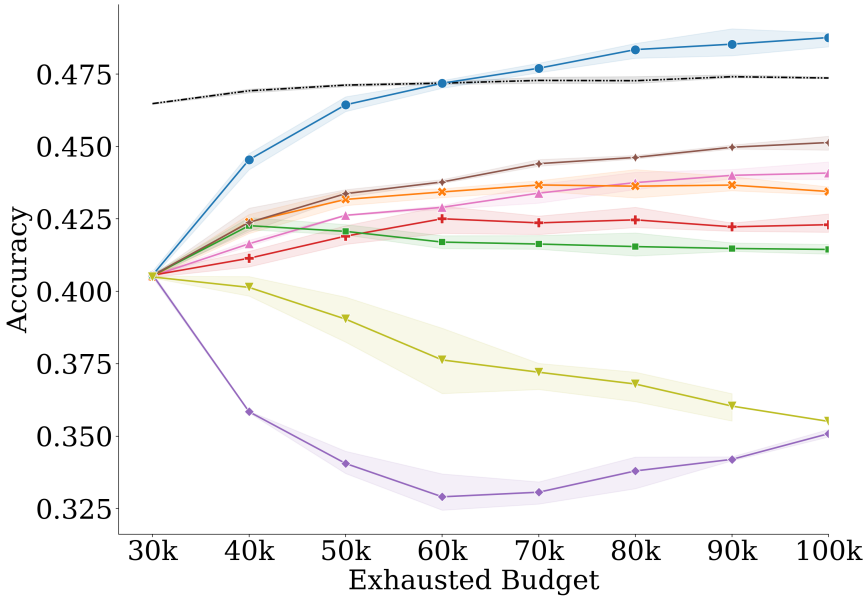


(a) Setting C-I

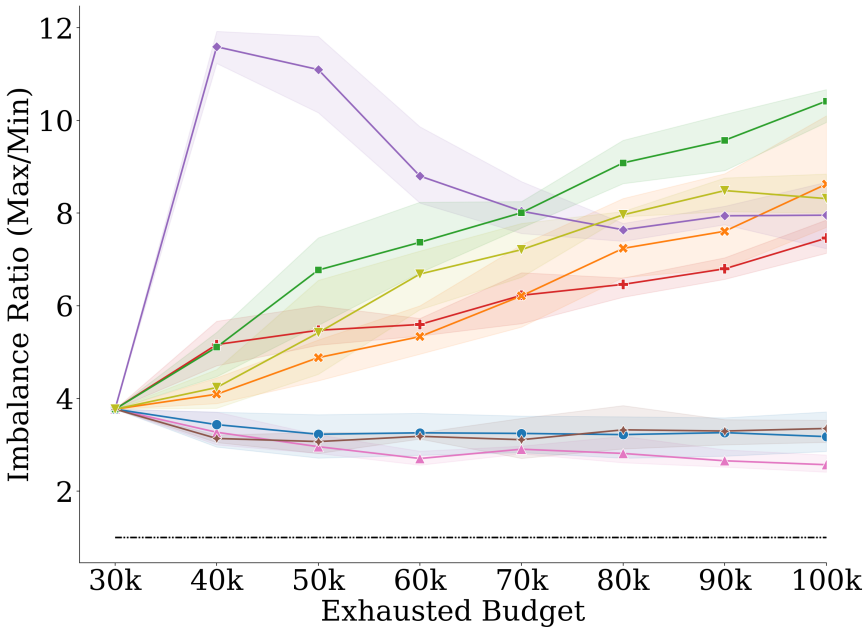


(b) Setting C-I

Figure 5.2: Average ImageNet accuracy and imbalance ratio over 3 runs. Shaded regions depict the 95% confidence interval of the results. Figures 5.2a and 5.2b are obtained by training a ResNet-50 end-to-end starting from a SSP checkpoint at every AL round. The 3 overlapping curves at the top of Figure 5.2a correspond to BASE, MASE, and Margin Sampler. See Table 5.3 for numerical results.

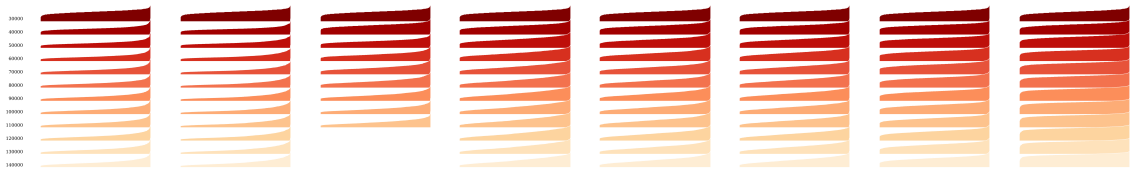


(a) Setting C-II



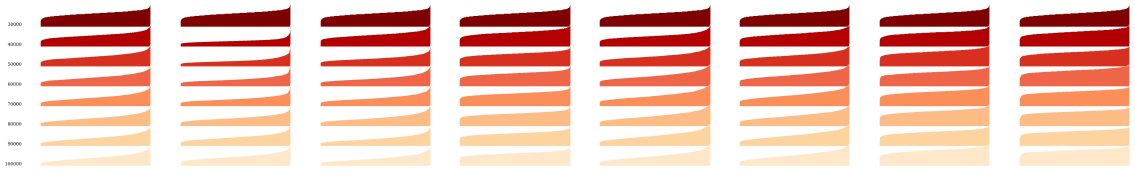
(b) Setting C-II

Figure 5.3: Average ImageNet accuracy and imbalance ratio over 3 runs. Shaded regions depict the 95% confidence interval of the results. Figures 5.3a and 5.3b are obtained by finetuning only the final linear layer of a ResNet-50 starting from a SSP checkpoint at every AL round.



(a) iv (b) vii (c) ix (d) vi (e) viii (f) xi (g) BASE (h) i

Figure 5.4: Setting C-I. The distribution of D_U^k at every AL round for different strategies on ImageNet in the end-to-end finetuning setting. All experiments start with the same randomly selected subset s^0. The x-axis is sorted for each histogram (every row in every subplot) from least queried class to most queried class. The height of the histogram at a given location on the x-axis indicates the proportion of the examples sampled from that class. BASE is visibly the most balanced strategy after random sampling.

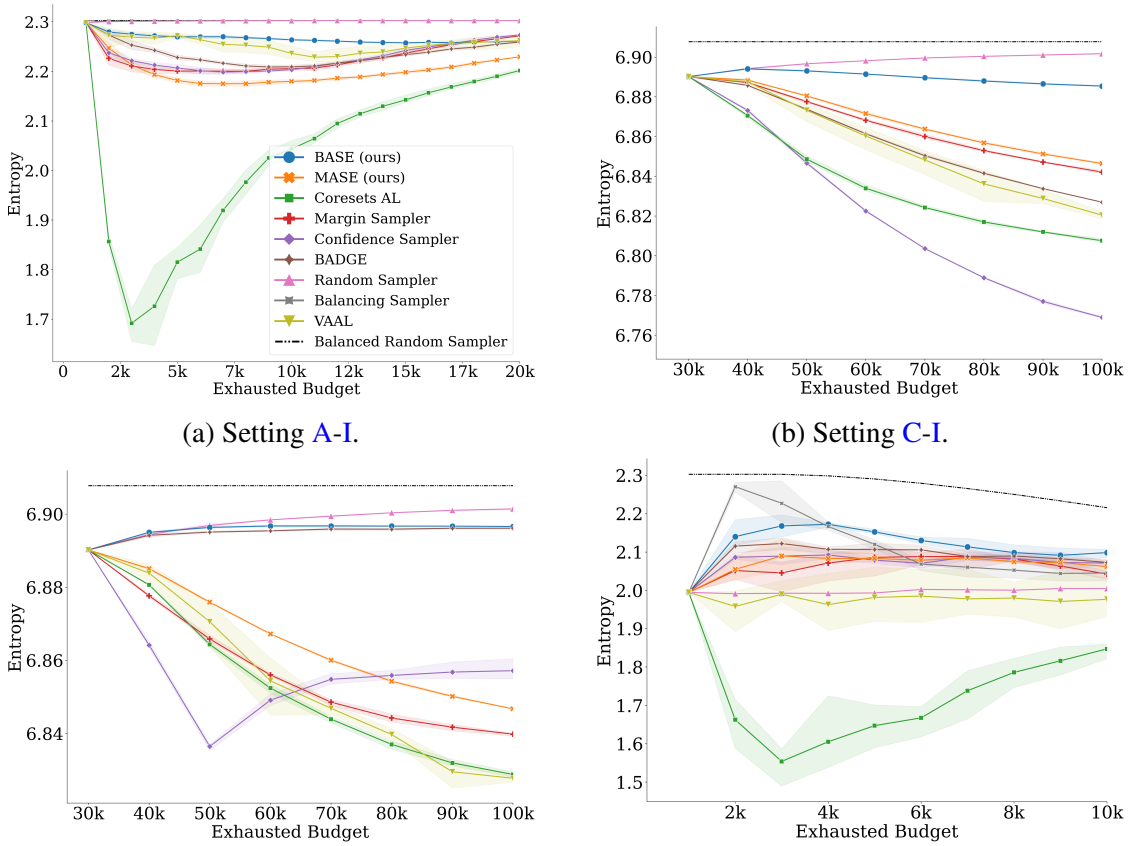


(a) iv (b) vii (c) ix (d) vi (e) viii (f) xi (g) BASE (h) i

Figure 5.5: Setting C-II. The distribution of D_U^k at every AL round for different strategies on ImageNet in the linear evaluation setting. All experiments start with the same randomly selected subset s^0. The x-axis is sorted for each histogram (every row in every subplot) from least queried class to most queried class. The height of the histogram at a given location on the x-axis indicates the proportion of the examples sampled from that class. BASE is visibly the most balanced strategy after random sampling.

5.5.3 Solving Class Imbalance Allows Scaling

Without explicitly imposing balance, imbalance becomes a problem and causes baseline algorithms to under-perform random sampling at the ImageNet scale. But by querying balanced samples, our BASE algorithm recovers the good properties of AL in the large-scale regime, and even matches the state-of-the-art EvSiT [90] results using only 71% of the ImageNet labels.



(a) Setting A-I.

(b) Setting C-I.

(c) Setting C-II.

(d) Setting B-I.

Figure 5.6: Class distribution entropy of D_L^k at different active learning rounds k. Higher entropy is desirable as it indicates more balanced sampling.

5.5.3.1 Baselines Perform Poorly on ImageNet

Here, we analyze the performance of all baselines on ImageNet. In Figure 5.2a, we compare different baselines on ImageNet, starting from a SSP checkpoint and finetuning the network end-to-end at each AL round. Three baselines provide material performance boosts over random sampling in that setting: Margin Sampler (viii), Partitioned BADGE (vi), and our MASE algorithm (xi). In Figure 5.3a, we evaluate all baselines on ImageNet in the linear evaluation setting described in Section 5.3. Surprisingly, *only a single baseline outperforms random sampling on the linear evaluation task.*

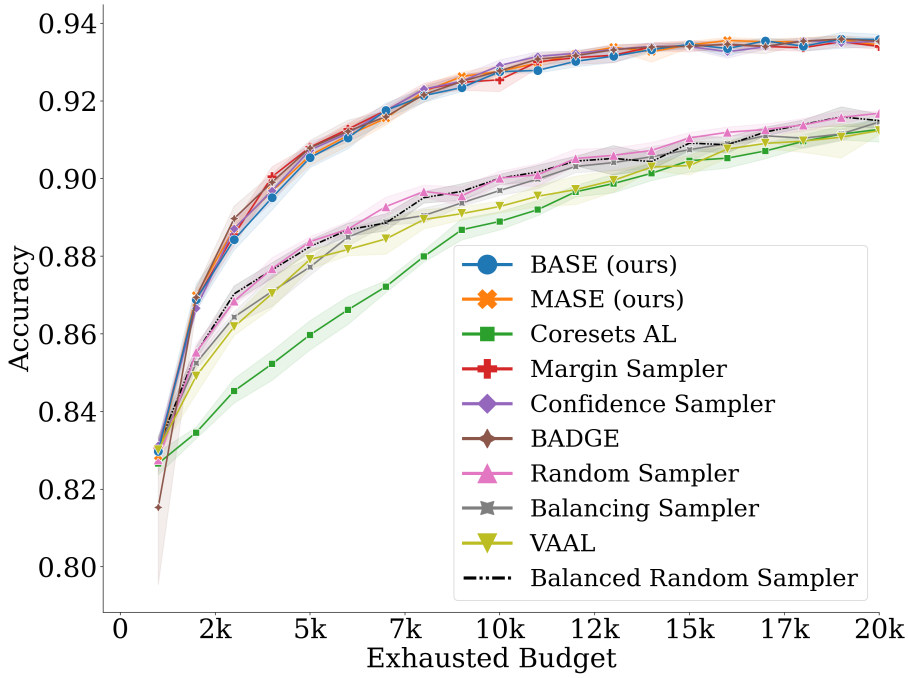


Figure 5.7: Setting A-I with $|s^0| = b = 1000$. Average results over 5 runs on CIFAR-10 obtained by training a ResNet-18 end-to-end starting from a SSP checkpoint at every AL round. Shaded regions depict the 95% confidence interval of the results. The 5 overlapping curves at the top are BASE, MASE, BADGE, Confidence Sampler, and Margin Sampler. See Table 5.6 for numerical results.

5.5.3.2 The Importance of Balanced Sampling

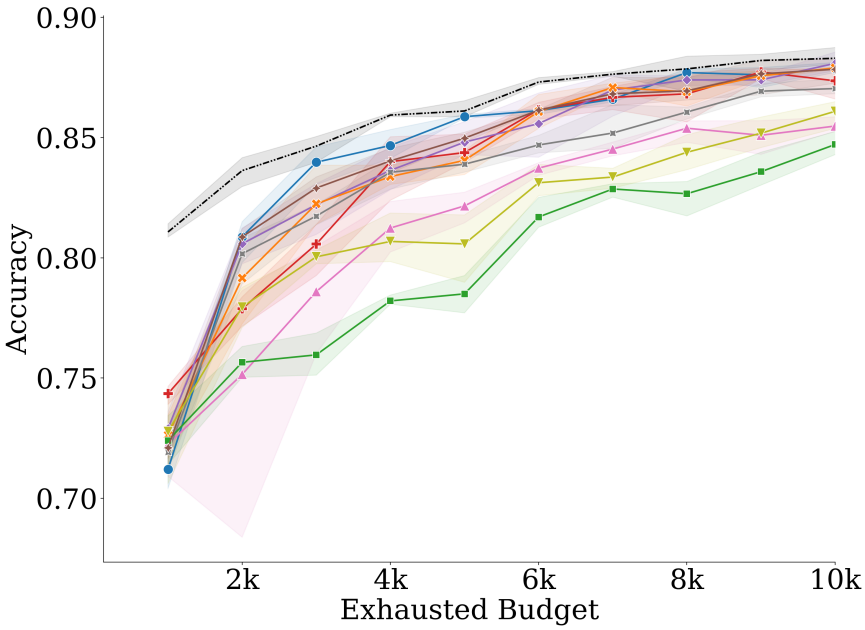
Figures 5.2b and 5.3b compare *class imbalance ratios* – the number of labels from the most sampled class over that of the least sampled class – for each sampling strategy. Most baseline samplers disproportionately query certain classes. Indeed, the Confidence Sampler, the worst performing baseline in Figure 5.3a, induces an imbalance ratio close to 12 after the first round of AL. To further investigate the effects of class imbalance, we implement a cheating baseline strategy (Balanced Random Sampler ii) which queries a perfectly balanced batch at each round. On the end-to-end finetuning experiment in Figure 5.2a, querying balanced batches is not sufficient to outperform random sampling.

However, on the linear evaluation task in Figure 5.3a, the cheating Balanced Random Sampler (ii) outperforms random sampling by approximately 6 percentage points at every round.

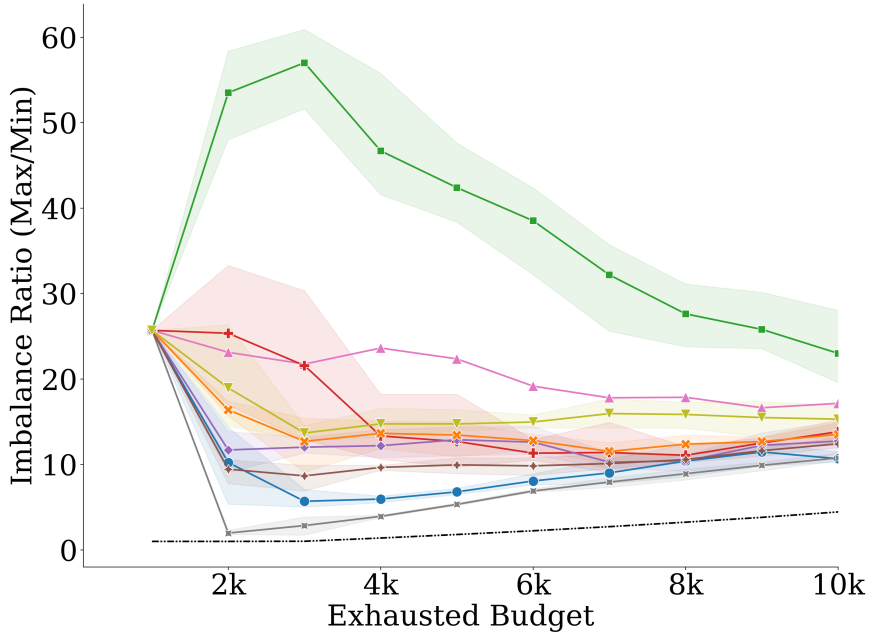
Class imbalance ratios do not fully describe the class imbalance across all classes, but only the extremes. To further investigate class imbalance, we analyze the distributions of D_L^k for all baselines on the ImageNet linear evaluation task in Figure 5.5. It is clear from the Figure that all baselines exhibit long tailed distributions, and increase the imbalance over time. Figure 5.4 contains histograms of the distributions for our end-to-end finetuning ImageNet experiments and Figure 5.6 yet another measure of class imbalance using entropy.

Table 5.4: Setting B-I with $|s^0| = b = 1000$. Average results over 3 runs on imbalanced CIFAR-10 obtained by finetuning a ResNet-18 end-to-end starting from a SSP checkpoint at every AL round.

Strategy/Budget	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Balancing Sampler	0.719	0.802	0.817	0.836	0.839	0.847	0.852	0.861	0.869	0.87
BASE	0.712	0.809	0.84	0.847	0.859	0.861	0.866	0.877	0.876	0.879
Balanced Random Sampler	0.811	0.836	0.846	0.859	0.861	0.873	0.876	0.879	0.882	0.883
Confidence Sampler	0.729	0.806	0.822	0.836	0.848	0.856	0.87	0.874	0.874	0.881
Coreset Sampler	0.724	0.757	0.76	0.782	0.785	0.817	0.829	0.827	0.836	0.847
BADGE	0.721	0.809	0.829	0.84	0.85	0.861	0.868	0.869	0.876	0.878
MASE	0.726	0.792	0.822	0.834	0.84	0.861	0.871	0.869	0.876	0.879
Margin Sampler	0.744	0.779	0.806	0.84	0.844	0.862	0.867	0.868	0.877	0.874
Random Sampler	0.723	0.751	0.786	0.812	0.822	0.837	0.845	0.854	0.851	0.855
VAAL Sampler	0.728	0.78	0.8	0.807	0.806	0.831	0.834	0.844	0.852	0.861



(a)



(b)

Figure 5.8: Setting B-I with $|s^0| = b = 1000$. Average results over 3 runs on imbalanced CIFAR-10 obtained by finetuning a ResNet-18 end-to-end starting from a SSP checkpoint at every AL round. Shaded regions depict the 95% confidence interval of the results. See Table 5.4 for numerical results.

Table 5.5: When applying BASE on the linear evaluation task with the state-of-the-art EsViT [90] SSP method, we are able to achieve the same state-of-the-art linear evaluation accuracy on ImageNet with only 71% (for top-1 acc.) and 55% (for top-5 acc.) of the data. The bolded number indicates the smallest amount of required data to achieve the same accuracy as using all of the data for training.

		Number of Labels (% of all labels)					
	Strategy	0.3M (24%)	0.5M (39%)	0.7M (55%)	0.9M (71%)	1.1M (87%)	All(100%)
Top 1 Acc.	BASE (ours)	78.9%	80.5%	81.0%	**81.2%**	81.2%	81.2%
	Random	78.9%	79.9%	80.5%	80.7%	81.0%	**81.2%**
Top 5 Acc.	BASE (ours)	94.4%	95.2%	**95.5%**	95.5%	95.5%	95.5%
	Random	94.5%	94.9%	95.1%	95.3%	**95.5%**	95.5%

5.5.3.3 BASE Outperforms Baselines and Mitigates Class Imbalance

In Section 5.4, we proposed BASE, an AL algorithm specifically designed to query class balanced data. BASE can significantly outperform random sampling on the ImageNet linear evaluation task shown in Figure 5.3a. In fact, BASE can even outperform the unrealistic Balanced Random Sampler (ii), which cheats by using knowledge of ground truth labels to achieve perfect class balance. On the finetuning experiment in Figure 5.2a, BASE performs on par with the best two baselines in terms of accuracy. However, in Figures 5.5, 5.2b, and 5.3b, we show that our AL algorithm consistently achieves a more uniform class distribution than all other baselines on both the linear evaluation and end-to-end finetuning ImageNet tasks.

Finally, in Table 5.5, we show that by carefully selecting examples, BASE can reproduce the state-of-the-art linear evaluation top-1 accuracy results reported in EsViT [90] using $\sim 29\%$ less labeled data; BASE only needs 55% of the labels to match the same top-5 accuracy reported in [90].

Table 5.6: Setting A-I with $|s^0| = b = 1000$. Average results over 5 runs on CIFAR-10 obtained by training a ResNet-18 end-to-end starting from a SSP checkpoint at every AL round.

Strategy/Budget	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000	12000	13000	14000	15000	16000	17000	18000	19000	20000
Balancing Sampler	0.83	0.852	0.864	0.871	0.877	0.885	0.889	0.89	0.894	0.897	0.9	0.903	0.904	0.906	0.907	0.909	0.911	0.91	0.911	0.915
BASE	0.83	0.869	0.884	0.895	0.905	0.911	0.918	0.921	0.923	0.928	0.928	0.93	0.932	0.933	0.935	0.933	0.935	0.934	0.936	0.936
Balanced Random Sampler	0.831	0.855	0.87	0.876	0.882	0.887	0.889	0.895	0.897	0.9	0.902	0.905	0.905	0.904	0.909	0.909	0.912	0.914	0.916	0.915
Confidence Sampler	0.831	0.867	0.887	0.897	0.907	0.912	0.917	0.923	0.925	0.929	0.931	0.932	0.933	0.934	0.934	0.933	0.934	0.936	0.935	0.935
Coreset Sampler	0.827	0.835	0.845	0.852	0.86	0.866	0.872	0.88	0.887	0.889	0.892	0.897	0.899	0.901	0.905	0.905	0.907	0.91	0.911	0.913
BADGE	0.815	0.869	0.89	0.899	0.908	0.912	0.916	0.922	0.925	0.928	0.931	0.932	0.933	0.934	0.934	0.935	0.934	0.935	0.936	0.935
MASE	0.829	0.87	0.887	0.897	0.906	0.911	0.916	0.922	0.926	0.928	0.93	0.932	0.934	0.933	0.934	0.936	0.935	0.935	0.936	0.934
Margin Sampler	0.83	0.869	0.886	0.9	0.908	0.913	0.918	0.923	0.925	0.925	0.93	0.931	0.932	0.934	0.934	0.934	0.934	0.934	0.935	0.934
Random Sampler	0.828	0.855	0.868	0.877	0.884	0.887	0.893	0.897	0.896	0.9	0.901	0.905	0.906	0.907	0.91	0.912	0.913	0.914	0.916	0.917
VAAL Sampler	0.83	0.849	0.862	0.87	0.879	0.882	0.884	0.889	0.891	0.893	0.895	0.897	0.9	0.903	0.903	0.908	0.909	0.91	0.911	0.912

5.5.4 Class Imbalance on Small Datasets

Motivated by our observations concerning the importance of balanced sampling in large-scale settings, we also investigate whether the balancing aspect of BASE offers benefits in small-scale settings. To this end, in Figure 5.8, we compare all AL strategies on an imbalanced version of CIFAR-10 [18] with an imbalance ratio of 10 starting from a SSP checkpoint obtained by training on the full imbalanced dataset. In this experiment, when training the classifier, we weigh each class differently in the loss function to penalize rare classes more heavily. The plots show a strong correlation between class distributions and performance of the AL algorithm. Balanced Random Sampler (ii), the "cheating" algorithm, achieves the best accuracy across the board. And with minor exceptions, for each algorithm, the better the performance in terms of accuracy, the less severe the observed class imbalance.

BASE is the best performing strategy in terms of accuracy and only second best in terms of class imbalance – the best being the non-scalable Balancing Sampler (x).

5.5.5 AL Performs Differently with SSP

Throughout this work, we argue that applying SSP along with simple random sampling is much more powerful than applying AL alone – see Table 5.2. Therefore, AL algorithms must prove that they can outperform random sampling in the SSP setting, otherwise they are redundant and potentially harmful to performance. In Figure 5.7, we show that some popular baselines, notably, Coreset AL (iii), are indeed harmful. A potential explanation for this failure mode can be found in [2], where the authors show that warm-starting the

network weights at each round can negatively impact the performance of Coreset AL (iii). Warm-starting means to continue training the network starting with the weights obtained in the previous AL round, as opposed to randomly re-initializing the network weights at every round (cold start). We suspect that SSP, just like warm-starting, may negatively impact the performance of Coreset AL on CIFAR-10 and conclude that future research should not draw conclusions about the performance of an AL algorithm in the SSP setting solely by observing its behaviour in the cold starting setting.

5.6 Conclusion and Future Directions

AL for DNNs is a very difficult problem to study, partly because we still cannot answer very fundamental questions about the generalization abilities of DNNs [74], but also because *random sampling is an incredibly robust baseline*. In this work, we highlighted the importance of stress-testing AL algorithms where they are most useful, namely on large-scale tasks. We showed that popular existing works cannot compete with random sampling across all settings, and we designed BASE, a robust AL strategy capable of doing just that. In future work, we will tackle more complex problems, where the cost savings incurred by AL are even more dramatic, such as large-scale segmentation and detection tasks.

5.7 Additional Material and Details

5.7.1 Experimental Details

We provide additional details of implementations and hyperparameters in the following sections.

Settings	$\|s^0\|$	b	Epochs	ESP	Batch size	Optimizer	Learning rate	Weight Decay	Momentum
A-I	1000	1000	200	50	128	SGD	$1e^{-3}$	$5e^{-4}$	0.9
B-I	1000	1000	200	50	128	SGD	$2e^{-3}$	0	0.9
C-I	30000	10000	60	30	128	SGD	$1e^{-3}$	0	0.9
C-II	30000	10000	60	30	128	SGD	15	$1e^{-4}$	0.9

Table 5.7: Hyperparameters used for each setting. $|s^0|$ denotes the initial pool size. b denotes the budget per round, and ESP abbreviates early stop patience.

5.7.1.1 Dataset Details and Early Stopping

Dataset Division. We split the target dataset into training set, validation set and test set. All AL algorithms are restricted to query from the training set, and the initial pool is also sampled from training set. The validation set is used for early stopping.

CIFAR-10 comes with natural split of training and testing data. We keep the testing data as test set. We randomly sample 1% of the training data as validation set and keep the rest as training set.

For Imbalanced CIFAR-10, we keep the original testing split as test data. We then follow this implementation to subsample a set of long-tailed imbalance data from the training split. The set of imbalance data is then randomly partitioned into training/validation data with $0.99/0.01$ split ratio.

For ImageNet, the dataset itself comes with natural training split and validation

split. We use the validation split as test set. We randomly sample 10% of the training split as validation set and keep the rest as training set.

Early Stopping. We use the validation set to estimate the final test accuracy and perform early stopping during the network training to avoid over-fitting. In particular, we stop training the classification network if the validation performance stops improving after a specific number of rounds (specified as a hyperparameter).

5.7.1.2 Hyperparameters for Each Experiment Setting

We conduct our experiments in the following four settings: C-I, C-II, A-I, and B-I. We provide the hyperparameters shared across these settings in Table 5.7, and discuss setting specific hyperparameters as follows. For setting A-I, and B-I, we use cosine annealing learning rate scheduler with $T_{max} = 200$. For setting C-I and C-II, we start from the learning rate provided in Table 5.7, and decrease it by a factor of 0.1 every 20 epochs.

5.7.1.3 Additional Details on baselines

In this Section, we discuss additional implementation details of Partitioned Coreset/BADGE sampler and VAAL sampler.

Partitioned Coreset/BADGE Sampler. To enable Coreset and BADGE sampler to run on ImageNet, we modify each algorithm to allow to scale, inspired by the approach in [28]. At each AL round we partition each of D_U^k and D_L^k into 10 random partitions. We then take one partition from each and combine them into 10 partitions, say $P_1, \ldots, P_{10}$,

then run Coresets or BADGE separately on each partition using $b/10$ budget.

For BADGE, we use global average pooling on the gradient embeddings to reduce their dimension to 512.

VAAL Sampler. We follow the repository provided by the original paper [137] to implement VAAL sampler. Since the architecture of the Variational Auto-Encoder (VAE) provided in the repository fails to handle ImageNet naturally, we instead use the VAE architecture in [147] and follow the paper to calculate the unsupervised loss with randomly-cropped 64×64 patches instead of the full original images. Also, we perform a single VAE optimizer step for every classifier optimizer step. The original [137] paper does not comment on this, however, their codebase performs two VAE optimizer steps for every classifier optimizer step.

5.7.2 Limitations

Plotting the test accuracy as a function of exhausted budget is common practice in AL research. However, these experiments are difficult to produce correctly as they are computationally expensive and very sensitive to hyperparameters [2, 12]. We conduct thorough experiments over an extensive set of hyperparameters to ensure fair comparisons. We summarize our findings below.

1. **Training Hyperparameters at Every Round.** This includes the choice of optimizer, learning rate, regularization, and early stopping hyperparameters. It is necessary to train the network to saturation at every round with varying amounts of training data; otherwise, a fair comparison of AL algorithms would not be possible. In fact, if the

network is not trained to saturation at a round k, the AL algorithm will not query an optimal set s^k, which will in turn affect the distribution of D_L^{k+i} for all subsequent rounds $k+i$, $i > 1$ [2].

2. **Initial Budget.** s^0 is randomly selected, therefore if the dataset is class balanced, s^0 will be relatively balanced. If s^0 is large, it will take many rounds of querying before we can notice performance differences between AL algorithms that select balanced data and those that don't. It is therefore important to monitor the distribution of D_L^k along with the accuracy of the model at each round before drawing conclusions about performance.

Bibliography

[1] Umang Aggarwal, Adrian Popescu, and Céline Hudelot. Active learning for imbalanced datasets. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1417–1426, 2020. doi: 10.1109/WACV45572.2020.9093475.

[2] Anonymous. Best practices in pool-based active learning for image classification. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7Rnf1F7rQhR. under review.

[3] Maria A. Aronova, Denzel R. Cruz, Douglas J. Palumbo, Rahul R. Akkem, **Zeyad A. Emam**, Matthew D. Guay, Sung W. Rhee, Irina D. Pokrovskaya, Brian Storrie, and Richard D. Leapman. Combined use of serial block face sem and focused ion beam sem elucidates the 3d ultrastructure of blood platelets and thrombi. *Biophysical Journal*, 121(3, Supplement 1): 149a, 2022. ISSN 0006-3495. doi: https://doi.org/10.1016/j.bpj.2021.11.1984. URL https://www.sciencedirect.com/science/article/pii/S0006349521029593.

[4] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263, 2018.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07*, 2007.

[6] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryghZJBKPS.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[8] Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, **Emam, Zeyad**, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. 02 2022.

[9] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, March 2003.

[10] Peter L Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems*, 1998.

[11] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*. 2017.

[12] Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh K. Iyer. Effective evaluation of deep active learning on image classification tasks. *CoRR*, abs/2106.15324, 2021. URL `https://arxiv.org/abs/2106.15324`.

[13] William H. Beluch, Tim Genewein, Andreas Nurnberger, and Jan M. Kohler. The power of ensembles for active learning in image classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018. doi: 10.1109/CVPR.2018.00976.

[14] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet?, 2020.

[15] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang,

Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[17] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NIPS*, pages 1565–1576, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/621461af90cadfdaf0e8d4cc25129f91-Abstract.html.

[18] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss, 2019.

[19] Yao-Chun Chan, Mingchen Li, and Samet Oymak. On the marginal benefit of active learning: Does self-supervision eat its cake? In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3455–3459. IEEE, 2021.

[20] P Chaudhari, Anna Choromanska, S Soatto, Yann LeCun, C Baldassi, C Borgs, J Chayes, Levent Sagun, and R Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations (ICLR)*, 2017.

[21] Hao Chen, Qi Dou, Lequan Yu, Jing Qin, and Pheng-Ann Heng. VoxResNet: Deep voxelwise residual networks for brain segmentation from 3d MR images. *NeuroImage*, 170:446–455, April 2018. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2017.04.041. URL http://www.sciencedirect.com/science/article/pii/S1053811917303348.

[22] Jianxu Chen, L. Yang, Yizhe Zhang, Mark S. Alber, and Danny Ziyi Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. *ArXiv*, abs/1609.01006, 2016.

[23] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[24] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[25] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[26] Seong Jin Cho, Gwangsu Kim, and Chang D. Yoo. Least probable disagreement region for active learning, 2021. URL https://openreview.net/forum?id=bGPNpnZYr1.

[27] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46723-8.

[28] Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *CoRR*, abs/2107.14263, 2021. URL https://arxiv.org/abs/2107.14263.

[29] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *arXiv:1604.01685 [cs]*, April 2016. URL http://arxiv.org/abs/1604.01685. arXiv: 1604.01685.

[30] Daniel Cressey and Ewen Callaway. Cryo-electron microscopy wins chemistry nobel. *Nature News*, 550(7675):167, 2017.

[31] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[32] Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Big batch sgd: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[34] Winfried Denk and Heinz Horstmann. Serial Block-Face Scanning Electron Microscopy to Reconstruct Three-Dimensional Tissue Nanostructure. *PLoS Biology*, 2(11), November 2004. ISSN 1544-9173. doi: 10.1371/journal.pbio.0020329. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC524270/.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[36] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.

[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[38] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[39] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv*, abs/1603.07285, 2016.

[40] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[41] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey, 2019.

[42] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, page 127–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595938039. doi: 10.1145/1321440.1321461. URL https://doi.org/10.1145/1321440.1321461.

[43] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge.

Int. J. Comput. Vision, 88(2):303–338, June 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL http://dx.doi.org/10.1007/s11263-009-0275-4.

[44] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[45] Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *International Conference on Learning Representations*, 2018.

[46] Huazhu Fu, Yanwu Xu, Stephen Lin, Damon Wing Kee Wong, and Jiang Liu. Deepvessel: Retinal vessel segmentation via deep learning and conditional random field. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 132–139, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46723-8.

[47] Yabo Fu, Yang Lei, Tonghe Wang, Walter J. Curran, Tian Liu, and Xiaofeng Yang. A review of deep learning based methods for medical image multi-organ segmentation. *Physica Medica*, 85:107–122, 2021. ISSN 1120-1797. doi: https://doi.org/10.1016/j.ejmp.2021.05.003. URL https://www.sciencedirect.com/science/article/pii/S1120179721001848.

[48] Dennis Gabor. Holography, 1948-1971. *Science*, 177(4046):299–313, 1972. ISSN 00368075, 10959203. URL http://www.jstor.org/stable/1734339.

[49] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045502.

[50] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning (ICML) - Volume 70*, ICML'17, pages 1183–1192. JMLR.org, 2017.

[51] Richard J. Giuly, Maryann E. Martone, and Mark Ellisman. Method: automatic segmentation of mitochondria utilizing patch classification, contour pair classification, and automatically seeded level sets. *BMC Bioinformatics*, 13:29 – 29, 2011.

[52] Micah Goldblum. *Adversarial Robustness and Robust Meta-Learning for Neural Networks*. PhD thesis, University of Maryland, 2020.

[53] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Proceedings of the 31st Conference On Learning Theory*, 2018.

[54] Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems. In *COLT*, 2017.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

[56] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[57] Yehoram Gordon. On Milman's inequality and random subspaces which escape through a mesh in $\mathbb{R}$ n. In *Geometric Aspects of Functional Analysis*, pages 84–106. Springer, 1988.

[58] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[59] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE transactions on medical imaging*, 35(5):1153–1159, 2016.

[60] Matthew Guay, **Zeyad Emam**, Adam Anderson, and Richard Leapman. Designing deep neural networks to automate segmentation for serial block-face electron microscopy. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 405–408, 2018. doi: 10.1109/ISBI.2018.8363603.

[61] Matthew Guay, **Emam, Zeyad**, Adam Anderson, and Richard Leapman. Exploring deep neural network architectures for automated electron micrograph segmentation. *Biophysical Journal*, 114:343a, 02 2018. doi: 10.1016/j.bpj.2017.11.1916.

[62] Matthew Guay, **Emam, Zeyad**, Adam Anderson, and Richard Leapman. Transfer learning for efficient segmentation of subcellular structures in 3-d electron microscopy. *Biophysical Journal*, 116:288a, 02 2019. doi: 10.1016/j.bpj.2018.11.1554.

[63] Matthew Guay, **Emam, Zeyad**, and Richard Leapman. Two-stage neural architecture search for microscopy image segmentation. *Microscopy and Microanalysis*, 25(S2):188–189, 2019. doi: 10.1017/S1431927619001673.

[64] Matthew Guay, **Zeyad Emam**, Adam Anderson, Maria Aronova, Irina Pokrovskaya, Brian Storrie, and Richard Leapman. Dense cellular segmentation for em using 2d–3d neural network ensembles. *Scientific Reports*, 11:2561, 01 2021. doi: 10.1038/s41598-021-81590-0.

[65] Estibaliz Gómez de Mariscal, Martin Maška, Anna Kotrbová, Vendula Pospichalova, Pavel Matula, and Arrate Muñoz-Barrutia. Deep-learning-based segmentation of small extracellular vesicles in transmission electron microscopy images. *Scientific Reports*, 9:1–10, 09 2019. doi: 10.1038/s41598-019-49431-3.

[66] Matthias G Haberl, Christopher Churas, Lucas Tindall, Daniela Boassa, Sébastien Phan, Eric A Bushong, Matthew Madany, Raffi Akay, Thomas J Deerinck, Steven T Peltier, et al. Cdeep3m—plug-and-play cloud-based deep learning for image segmentation. *Nature methods*, 15(9):677–680, 2018.

[67] Mohammad Haft-Javaherian, Linjing Fang, Victorine Muse, Chris B. Schaffer, Nozomi Nishimura, and Mert R. Sabuncu. Deep convolutional neural networks for segmenting 3d in vivo multiphoton images of vasculature in Alzheimer disease mouse models. *PLoS ONE*, 14(3), March 2019. ISSN 1932-6203. doi: 10.1371/journal.pone.0213539. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6415838/`.

[68] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on International Conference on Machine Learning*, 2016.

[69] Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In *Proceedings of the 2017 Conference on Learning Theory*, 2017.

[70] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. ISSN: 1063-6919.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9: 1–42, 1997.

[73] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9 (1):1–42, 1997.

[74] W. Ronny Huang, **Zeyad Emam**, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations, 2020.

[75] Lawrence J. Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[76] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[77] Ilya Kavalerov. *Impact Of Semantics, Physics And Adversarial Mechanisms In Deep Learning*. PhD thesis, University of Maryland, 2020.

[78] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[79] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

[80] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[81] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.

[82] Max Knoll and Ernst Ruska. Das elektronenmikroskop. *Zeitschrift für physik*, 78 (5):318–339, 1932.

[83] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[84] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[85] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL `https://proceedings.neurips.cc/paper/1994/file/b8c37e33defde51cf91e1e03e51657da-Paper.pdf`.

[86] Ritwik Kumar, Amelio Vázquez-Reina, and Hanspeter Pfister. Radon-like features and their application to connectomics. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 186–193, 2010. doi: 10.1109/CVPRW.2010.5543594.

[87] Ilja Kuzborskij and Christoph H. Lampert. Data-dependent stability of stochastic gradient descent. In *International Conference on International Conference on Machine Learning*, 2018.

[88] Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *International Conference on Machine Learning*, 2018.

[89] Kisuk Lee, Aleksandar Zlateski, Vishwanathan Ashwin, and H. Sebastian Seung. Recursive Training of 2d-3d Convolutional Networks for Neuronal Boundary Prediction. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3573–3581. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5636-recursive-training-of-2d-3d-convolutional-networks-for-neur` `pdf`.

[90] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. *arXiv preprint arXiv:2106.09785*, 2021.

[91] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399. 2018.

[92] Weilin Li. *Topics in Harmonic Analysis, Sparse Representations, and Data Analysis*. PhD thesis, University of Maryland, 2018.

[93] X. Li and Y. Guo. Adaptive active learning for image classification. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 859–866, June 2013.

[94] Yiran Li. *Feature extraction in image processing and deep learning*. PhD thesis, University of Maryland, 2018.

[95] Yuan-Hong Liao, Amlan Kar, and Sanja Fidler. Towards good practices for efficiently annotating large-scale image classification datasets. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4348–4357, 2021.

[96] Jeff Lichtman, Hanspeter Pfister, and Nir Shavit. The big data challenges of connectomics. *Nature neuroscience*, 17:1448–54, 11 2014. doi: 10.1038/nn.3837.

[97] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[98] Zhen Ma, Joao Tavares, and Renato Natal Jorge. A review on the current segmentation algorithms for medical images. pages 135–140, 01 2009.

[99] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 185–201, Cham, 2018. Springer International Publishing.

[100] David A. McAllester. Some pac-bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 230–234, 1998.

[101] David A. McAllester. Pac-bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 164–170, New York, NY, USA, 1999.

[102] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016. doi: 10.1109/3DV.2016.79.

[103] Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

[104] Jean-Michel Morel and Sergio Solimini. *Variational Methods in Image Segmentation*. Birkhäuser Boston, Boston, MA, 1995.

[105] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[106] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Proceedings of The 28th Conference on Learning Theory*, 2015.

[107] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[108] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

[109] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[110] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, NY, 2003.

[111] Jay Patravali, Shubham Jain, and Sasank Chilamkurthy. 2d-3d fully convolutional neural networks for cardiac mr segmentation. *ArXiv*, abs/1707.09813, 2017.

[112] Michael Pekala. *Harmonic Analysis and Machine Learning*. PhD thesis, University of Maryland, 2018.

[113] Irina D Pokrovskaya, Maria A Aronova, Jeffrey A Kamykowski, Andrew A Prince, Jake D Hoyne, Gina N Calco, Bryan C Kuo, Qianping He, Richard D Leapman, and Brian Storrie. Stem tomography reveals that the canalicular system and α-granules remain separate compartments during early secretion stages in blood platelets. *Journal of Thrombosis and Haemostasis*, 14(3):572–584, 2016.

[114] Irina D Pokrovskaya, Shilpi Yadav, Amith Rao, Emma McBride, Jeffrey A Kamykowski, Guofeng Zhang, Maria A Aronova, Richard D Leapman, and Brian Storrie. 3d ultrastructural analysis of α-granule, dense granule, mitochondria, and canalicular system arrangement in resting human platelets. *Research and practice in thrombosis and haemostasis*, 4(1):72–85, 2020.

[115] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[116] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[117] Research and Markets. Global data annotation tools market size, share & trends analysis report by type (text, image/video, audio), by annotation type (manual, automatic, semi-supervised), by vertical, by region, and segment forecasts, 2021-2028. Technical report, 2021.

[118] Daniela Rhodes. Aaron klug (1926–2018). *Nature Structural & Molecular Biology*, 26, 01 2019. doi: 10.1038/s41594-018-0183-9.

[119] Intisar Rizwan I Haque and Jeremiah Neubert. Deep learning approaches to biomedical image segmentation. *Informatics in Medicine Unlocked*, 18:100297, 2020. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2020.100297. URL https://www.sciencedirect.com/science/article/pii/S235291481930214X.

[120] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *LNCS*, volume 9351, pages 234–241, 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28.

[121] Holger R. Roth, Hirohisa Oda, Xiangrong Zhou, Natsuki Shimizu, Ying Yang, Yuichiro Hayashi, Masahiro Oda, Michitaka Fujiwara, Kazunari Misawa, and Kensaku Mori. An application of cascaded 3d fully convolutional networks for medical image segmentation. *Computerized Medical Imaging and Graphics*, 66:90–99, June 2018. ISSN 0895-6111. doi: 10.1016/j.compmedimag.2018.03.001. URL http://www.sciencedirect.com/science/article/pii/S0895611118301472.

[122] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[123] Ernst Ruska. The development of the electron microscope and of electron microscopy. *Bioscience reports*, 7(8):607–629, 1987.

[124] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[125] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL `https://doi.org/10.1007/s11263-015-0816-y`.

[126] Berkman Sahiner, Aria Pezeshk, Lubomir M Hadjiiski, Xiaosong Wang, Karen Drukker, Kenny H Cha, Ronald M Summers, and Maryellen L Giger. Deep learning in medical imaging and radiation therapy. *Medical physics*, 46(1):e1–e36, 2019.

[127] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Auto-context convolutional neural network (auto-net) for brain extraction in magnetic resonance imaging. *IEEE transactions on medical imaging*, 36(11):2319–2330, 2017.

[128] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In Frank Hoffmann, David J. Hand, Niall Adams, Douglas Fisher, and Gabriela Guimaraes, editors, *Advances in Intelligent Data Analysis*, pages 309–318, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[129] Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Arpit Bansal, **Zeyad Emam**, Furong Huang, Micah Goldblum, and Tom Goldstein. Datasets for studying generalization from easy to hard examples. *CoRR*, abs/2108.06011, 2021. URL `https://arxiv.org/abs/2108.06011`.

[130] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

[131] Burr Settles. Active learning literature survey. 07 2010.

[132] Mojtaba Seyedhosseini, Mark Ellisman, and Tolga Tasdizen. Segmentation of mitochondria in electron microscopy images using algebraic curves. volume 2013, 04 2013. doi: 10.1109/ISBI.2013.6556611.

[133] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[134] Renuka Shenoy, Min-Chi Shih, and Kenneth M. Rose. Segmentation of cells in electron microscopy images through multimodal label transfer. *2015 IEEE International Conference on Image Processing (ICIP)*, pages 103–107, 2015.

[135] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019. doi: 10.1186/s40537-019-0197-0.

[136] Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. Rethinking deep active learning: Using unlabeled data at model training. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1220–1227. IEEE, 2021.

[137] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. *CoRR*, abs/1904.00370, 2019. URL `http://arxiv.org/abs/1904.00370`.

[138] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[139] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel Rodrigues. Generalization error of invariant classifiers. In *Artificial Intelligence and Statistics*, pages 1094–1103, 2017.

[140] Detlev Stalling, Malte Westerhoff, Hans-Christian Hege, et al. Amira: A highly interactive system for visual data analysis. *The visualization handbook*, 38:749–67, 2005.

[141] Pierre Stock and Moustapha Cissé. Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases. In *ECCV*, 2018.

[142] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017. doi: 10.1109/ICCV.2017.97.

[143] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *AAAI*, 2016.

[144] Saeid Taghanaki, Kumar Abhishek, Joseph Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. Deep semantic segmentation of natural and medical images: a review. *Artificial Intelligence Review*, 54, 01 2021. doi: 10.1007/s10462-020-09854-1.

[145] **Zeyad Emam**, Hong-Min Chu, Ping-Yeh Chiang, Wojciech Czaja, Richard D. Leapman, Micah Goldblum, and Tom Goldstein. Active learning at the imagenet scale. *ArXiv*, abs/2111.12880, 2021.

[146] **Zeyad Emam**, Andrew Kondrich, Sasha Harrison, Felix Lau, Yushi Wang, Aerin Kim, and Elliot Branson. On the state of data in computer vision: Human annotations remain indispensable for developing deep learning models. *CoRR*, abs/2108.00114, 2021. URL https://arxiv.org/abs/2108.00114.

[147] Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *CoRR*, abs/1711.01558, 2017. URL http://arxiv.org/abs/1711.01558.

[148] Rohun Tripathi and Bharat Singh. Rso: A gradient free sampling based approach for training deep neural networks. *arXiv preprint arXiv:2005.05955*, 2020.

[149] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *ICML*, 2020.

[150] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

[151] Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

[152] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

[153] Wikipedia. Artificial neural network — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=766085187. [Online; accessed 22-February-2017].

[154] Pengtao Xie, Yuntian Deng, and Eric Xing. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*, 2015.

[155] Abhay Yadav, Tom Goldstein, and David Jacobs. Making l-bfgs work with industrial-strength nets. *The British Machine Vision Conference (BMVC)*, 2020.

[156] Haoqi Yang and Hongge Yao and. Street view house number identification based on deep learning. *International Journal of Advanced Network, Monitoring and Controls*, 4(2470-8038):47–52, 2019. doi: 10.21307/ijanmc-2019-058.

[157] Wenjing Yin, Derrick Brittain, Jay Borseth, Marie E. Scott, Derric Williams, Jed Perkins, Christopher S. Own, Matthew F. Murfitt, Russel Torres, Daniel Kapner, Gayathri Mahalingam, Adam A. Bleckert, Dan Castelli, David Reid, Wei-Chung Allen Lee, Brett J. Graham, Marc M. Takeno, Daniel J. Bumbarger, Colin Farrell, R. Clay Reid, and Nuno Maçarico da Costa. A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nature Communications*, 11, 2020.

[158] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6, 2017.

[159] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2016.

[160] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[161] Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. In *International Conference on Learning Representations*, 2018.