The Goldilocks Zone and Geometric Features of High-Dimensional Parameter Spaces

Jeffrey Chang
Department of Physics
Stanford University
Stanford, California, United States
jeffjar@stanford.edu

***Abstract*—Recently, it has been proposed that neural networks exhibit a 'Goldilocks Zone': a region in parameter space of low loss and high generalization accuracy, located in a hyper-annulus where the sum of squared weights is not too small and not too large. In this project, I report that neural networks trained on the CIFAR-10 dataset exhibit this phenomenon. I also explain some of the observed behaviors using geometric arguments about high-dimensional spaces.**

***Index Terms*—parameter space, loss landscapes, training trajectories, trajectories.**

I. Introduction

Today's large neural networks have made significant advancements in computer vision, speech transcription, machine translation, and numerous other domains. Many fundamental aspects of their behavior, however, remain poorly understood. The high dimensionality of parameter space implies that the training trajectory explores an exponentially small region of possibilities; nonetheless, with the right choice of hyperparameters, the training procedure can still manage to find a satisfactory low-loss, well-generalizing solution. Should one be surprised by this?

Without a fuller understanding of the loss landscapes of neural networks—the local and global properties of the loss as a function of parameter space—it is difficult to judge whether behaviors are surprising or unsurprising. More generally, understanding features of the landscape could shed light on ways to improve the accuracy, generalization, and robustness of neural networks.

My final project extends a previous paper to try to answer some of these questions. Reference [1] had demonstrated that neural networks trained on the MNIST dataset [2] exhibit a special region of parameter space which contained many good initialization points and well-generalizing solutions. This special region, termed the 'Goldilocks Zone', is a hyper-annulus in parameter space where the L2 norm of the weights is neither too big or too large (Fig. 1). Below, I provide experimental evidence that this phenomenon also occurs in the CIFAR-10 dataset [3]. I also provide simple geometrical arguments for why one might (or might not) be surprised by these results.

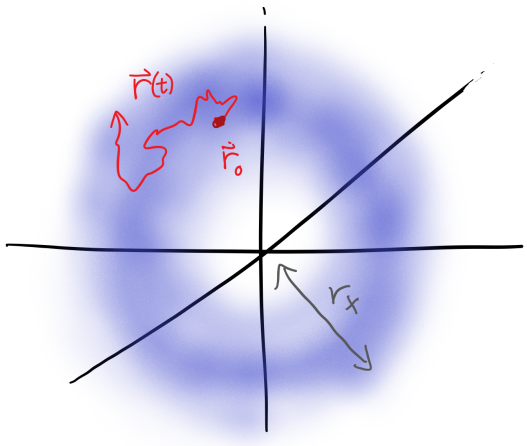


Fig. 1: The shaded blue region depicts of the Goldilocks Zone, a hyper-annulus in parameter space where the radial distance (L2 norm of the weights) is not too large and not too small. This region has been hypothesized [1] to contain many good initialization points and well-generalizing solutions. The red dot $\vec{r}_0$ depicts the initialization point, and red trajectory $\vec{r}(t)$ depicts the portion of parameter space traversed by the training procedure. Based on the corresponding figure in [1].

II. Background

A. Parameter Space

The parameters of a neural network are the entries of the weight matrices $W_{ab}^{[\ell]}$ and bias vectors $b_b^{[\ell]}$. It is often helpful to imagine that the parameters of a neural network live in a D-dimensional *parameter space*, which we can define by collecting all the parameters and flattening them into a D-dimensional vector $\vec{r}$. During the training procedure, the parameters are updated over time via some version of stochastic gradient descent, and the vector $\vec{r}$ takes a *trajectory* $\vec{r}(t)$ in parameter space, where t is the training time measured in epochs. The goal of training is to find a solution $\vec{r}_*$ with low loss $J(\vec{r}_*)$ and high generalization accuracy.

In modern neural networks, the number of parameters D is on the order of 10^5 or more. Volume scales as r^D, so in high dimensions, the number of possibilities is extremely large. The training trajectory hence explores only an exponentially small

portion of parameter space. Miraculously, it is still possible to train modern neural networks in regimes where D is large.

A good solution is a point in parameter space where the loss is low and the generalization accuracy is high. The only way that the training trajectory can manage to find good solutions is if (1) the local stochastic gradient provides sufficient signal to guide the trajectory towards good solutions, or if (2) the good solutions are dispersed everywhere throughout the space and are not hard to encounter. (Note that the two reasons are not mutually exclusive.)

Recent work has suggested that the second reason is more important than one might expect. It appears that the solution set of a neural network occupies a rather-large-dimensional submanifold of parameter space, and that all points on this solution manifold are suitable solutions [4]. For example, an interesting set of experiments from the Uber AI labs [5] has demonstrated that good solutions can be found when the training trajectory is constrained to a random d-dimensional hyperplane in parameter space—*even when* $d \ll D$. Another paper has demonstrated that a large neural network can fit to a datset even when *most of the weights are held fixed throughout training* [**?**]. These observations suggest that low-loss regions are relatively common in parameter space.

The value of d in the random hyperplane experiment presents a way to quantify how easy it is to find solutions in the parameter space: if a large value of d is required to find a good solution, it implies that solutions are hard to find; conversely, if a good solution can be found for a small value of d, it means that the solutions are common enough that a random low-dimensional subspace can manage to intersect with the solution manifold.

A natural follow-up question is whether some regions of parameter space contain solutions more frequently than others.

B. The Goldilocks Zone

One way to characterize the different regions of parameter space is through the L2 norm of all the weights and biases,

$$r \equiv |\vec{r}| = \sqrt{\sum_{i=1}^{D} r_i^2} = \sqrt{\sum_{\ell} \sum_{a,b} \left(W_{ab}^{[\ell]}\right)^2 + \sum_{\ell} \sum_{c} \left(b_c^{[\ell]}\right)^2}. \tag{1}$$

The radial coordinate r is the distance from the origin of parameter space.

A recent paper [1] has shown that solutions are most easily found in a hyper-annulus where points are not too close and not too far from the origin; i.e., when the radial coordinate lies within a special range $r_L \lesssim r \lesssim r_R$. This region of parameter space has been termed the 'Goldilocks Zone'[1] (Fig. 1). The authors of this paper trained fully connected neural networks on the MNIST dataset and explored the prevelance of the solution manifold at different r. In particular, they performed the random-d-dimensional-hyperplane experiment with the additional constraint of fixing the radial distance r to a particular r_0. The authors found that when r_0 was within an order of magnitude of a special r_X, the same validation accuracy could be achieved for smaller d than when r_0 was far from r_X. This observation suggests that good solutions are more 'common' in the Goldilocks Zone than elsewhere in parameter space.

[1]The terminology comes from exoplanet theory, where liquid water can only exist on a planet if it is neither too close not too far from its sun.

C. This work

My project aims to extend the results of the Goldilocks Zone paper in a few ways:

1) To generalize the findings to the CIFAR-10 dataset, a more complex dataset involving natural RGB images of planes, trucks, frogs, and more;[2]
2) To investigate what happens when the radius r is constrained *without* the extra constraint of random d-dimensional hyperplanes;
3) To see if the observed phenomena can be explained with simple geometrical facts of high-dimensional spaces.

III. Methods

To explore the structure of the solution set, I constrained the training trajectory to various submanifolds of parameter space, and investigated whether a good solution (one with low loss and high generalization accuracy) could still be found.

A. Random Hyperplanes

One way to reduce the search space of possible solutions is to restrict the training trajectory to explore only a lower d-dimensional region of the full D-dimensional parameter space. Following the approach of [5], I can confine the training to a random d-dimensional hyperplane, defined as the set of all $\vec{r} \in \mathbb{R}^D$ such that

$$\vec{r} = \vec{a} + \mathbf{P}\vec{\theta}, \quad \theta \in \mathbb{R}^d. \tag{2}$$

Here $\vec{a} \in \mathbb{R}^D$ is an offset into parameter space and $\mathbf{P} \in \mathbb{R}^{D \times d}$ is a projection matrix whose columns are orthonormal vectors in $\mathbb{R}^D$. Together, $\vec{a}$ and $\mathbf{P}$ define a d-dimensional hyperplane in $\mathbb{R}^D$—the affine space given by the span of the columns of $\mathbf{P}$ plus an offset of $\vec{a}$. During training, the values of $\vec{a}$ and $\mathbf{P}$ are held fixed at random values, and only $\vec{\theta}$ is allowed to vary.

B. Random Hyperspheres

One can also constrain the radial distance r to search only for solutions on d-dimensional hyperspheres[3] of radius r_0. These submanifolds are defined by the set of all points $\vec{r} \in \mathbb{R}^D$ such that

$$\vec{r} = \mathbf{P}\vec{\theta} \text{ and } |\vec{r}| = r_0, \quad \theta \in \mathbb{R}^d. \tag{3}$$

Here $\mathbf{P} \in \mathbb{R}^{D \times d}$ is a projection matrix with orthonormal columns and r_0 is the radius of the hypersphere. During

[2]The CIFAR-10 dataset is significantly harder to fit to than the MNIST dataset. In the random-d-dimensional-hyperplane experiment, it is possible to achieve 90% of the baseline accuracy (that is, the $d = D$ accuracy) on MNIST with only $d \simeq 750$ tunable parameters. In contrast, training a hyperplane-confined neural network on the CIFAR-10 dataset requires roughly $d \simeq 9,000$ degrees of freedom to achieve the same metric.

[3]Technically, the surface of a d-dimensional hyperball is a $d-1$- dimensional hypersphere, but d is close enough to $d-1$ that I will just call it a d-dimensional hypersphere for convenience.

training, the entries of the matrix $\mathbf{P}$ are held fixed at random values, and only the parameters θ_i are allowed to vary.

C. Initialization radii

Another important consideration of training neural networks is how the weights and biases are initialized; i.e., at what point $\vec{r}_0$ in parameter space the training trajectory begins its search.

Typically, initialization schemes pick the entries of weight matrices to be drawn from i.i.d. Gaussian distributions. These initialization schemes (e.g. Xavier, He [6], [7]) cause the initialization radii $r_0 \equiv |\vec{r}_0|^2$ to *peak sharply around a particular value* $r_0 = r_X$. The reason is that the components of the initial parameter vector, r_i, are distributed as

$$P(r_i)\, dr_i \propto e^{-r_i^2/2\sigma^2}\, dr_i, \tag{4}$$

implying that $\vec{r}_0$ is distributed as

$$P(\vec{r}_0)\, d^D\vec{r}_0 \propto e^{|\vec{r}_0|^2/2\sigma^2}\, d^D\vec{r}_0. \tag{5}$$

Since the hypershell between radii r and $r + dr$ has a hypervolume proportional to $d^D r = r^{D-1} dr$, the distribution over the *magnitude* of the initialization vector is

$$P(r_0)\, dr_0 \propto r_0^{D-1} e^{r_0^2/2\sigma^2}\, dr_0, \tag{6}$$

which is a sharply peaked function around $r_X \equiv \sqrt{D-1}\sigma$. That is, a Gaussian in high dimensions looks like a thin hypershell centered around $r = r_X$.[4]

In the experiments below, the initial point in parameter space is chosen isotropically from a hypersphere of radius r_0.

[4]The different layers of the network are typically given different variances, implying that the Gaussian is anisotropic; i.e., it is an ellipsoid shell rather than a spherical shell. This difference does not matter much since the vast majority of the parameters are found in the weight matrix of the first layer.

D. Experimental Details

A fully-connected neural network with architecture $[3,072 \to 200 \to 200 \to 10]$ ($D = 656,810$) was trained on the CIFAR-10 dataset (60,000 RGB images of 32×32 in 10 classes), with softmax activation on the final layer and ReLU activation otherwise. The cross-entropy loss was minimized with the Adam optimizer, and the validation accuracy was measured after 15 epochs.

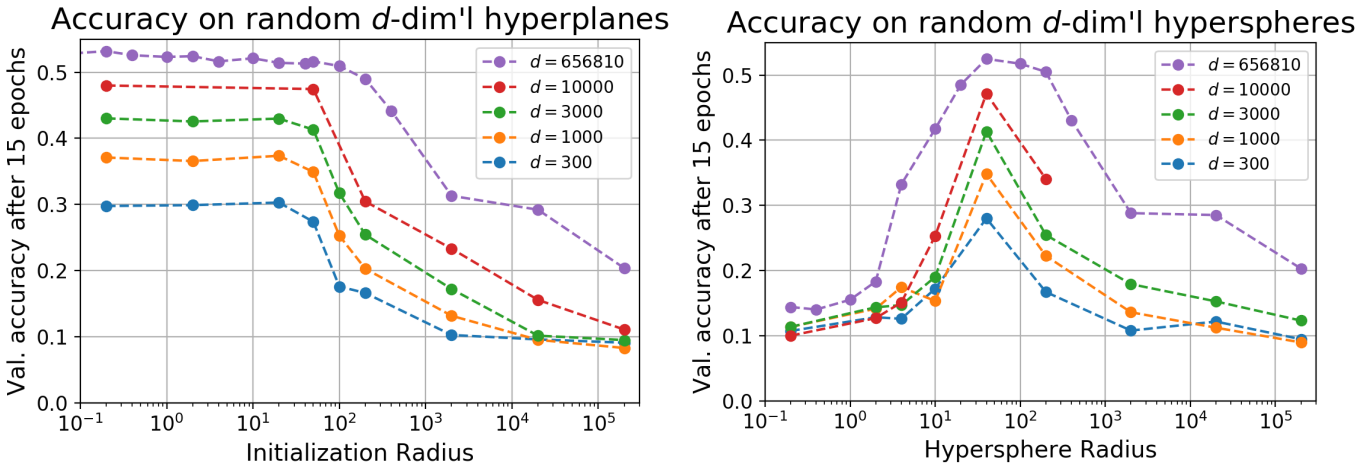
The projection matrix $\mathbf{P}$ was implemented as a sparse matrix with $O(\sqrt{D}d)$ nonzero entries. Each entry was chosen to be ± 1 with probability $1/\sqrt{D}$, and the columns were subsequently normalized. The columns of this matrix are nearly orthogonal (as verified numerically) because vectors in high-dimensional spaces are almost always orthogonal. The sparse implementation was necessary due to memory constraints.

For the random hyperplane experiments, the offset $\vec{a}$ was drawn from an isotropic unit-variance Gaussian, and then scaled acordingly so that its L2 norm was r_0. The coordinates on the hyperplane θ_i were initialized at zero. For the random hypersphere experiments, the parameters θ_i were drawn from a unit-variance Gaussian and then normalized so that $|\vec{\theta}|^2 = |\vec{r}|^2 = r_0$. The radius was constrained by subtracting off the radial component $\vec{r} \cdot \nabla_r \ell / |\vec{r}|$ from the gradient and re-normalizing the radius $\vec{r} := \vec{r} r_0 / |\vec{r}|$ after each time step.

Code for this project can be found in a github repository at `https://github.com/jeffiar/cs229-final-project`.

IV. Results

I trained a set of neural networks on the CIFAR-10 dataset while varying the initialization radius r_0 over six orders of magnitude. The training trajectory was constrained to a d-dimensional hyperplanes and hyperspheres as described in



(a) Random d-dimensional hyperplanes.

(b) Random d-dimensional hyperspheres.

Fig. 2: Shown above is the validation accuracy of a fully connected neural network trained on CIFAR-10 when the training trajectory is constrained to a random d- dimensional hyperplane (a) or hypersphere (b). The trajectory is initialized at a radius of r_0 with equal probability in all directions of parameter space, and the validation accuracy is measured 15 epochs of training. The right panel has an additional constraint that the radial distance r_0 must be held fixed throughout training. (A few data points are missing due to time constraints.)

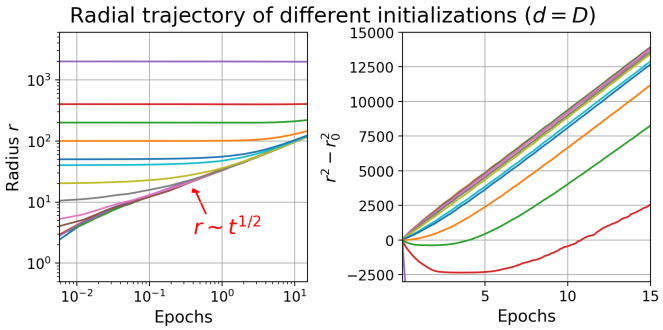


Fig. 3: The radial distance r of training trajectories appears to grow as $r \sim t^{1/2}$ for large enough r and t. This is the expected behavior for a random walk (see text).

Section III. Figures 2a and 2b plot the validation accuracies attained after 15 epochs of training; for comparison, the validation accuracy attained by the unconstrained trajectory ($d = D$) is also plotted.

As shown in Fig. 2a, the validation accuracy is higher when the hyperplane dimension d is larger. The validation accuracy also depends heavily on the initialization radius r_0: when $r_0 \lesssim r_X$, good solutions can be found easily after 15 epochs of training, but when $r_0 \gtrsim r_X$, the validation accuracy starts to drop.

In the random-hypersphere experiments, the radius is held fixed at its initialization radius r_0. As shown in Fig. 2b, the radial constraint lowers the validation accuracy at small r_0 but has little impact on the accuracy at large r_0.

To investigate why constraining the radius only affects validation accuracy appreciably when $r_0 < r_X$, I measured the radial distance r as a function of training time for the unconstrained trajectories ($d = D$). Shown in Fig. 3 is the radial trajectory for various values of initialization radii. The log-log plot shows that the radius grows as as $r \propto t^{1/2}$ for large enough r and t. The right panel shows that $r^2 - r_0^2$ grows linearly in time when r_0 is not too large.

V. Discussion

Figure 2b illustrates the presence of a Goldilocks Zone for fully-connected neural networks trained on the CIFAR-10 datset: the accuracy is the highest when r is within an order of magnitude of a characteristic r_X. As noted in [1], the peak $r_0 = r_X$ coincides with the radius of successful initialization schemes.

Intriguingly, the peak in accuracy near $r_0 = r_X$ also occurs for $d = D$: one can observe the *same* phenomenon with full-space trajectories at fixed r_0 without constraining the training to a d-dimensional subspace. This observation indicates that there is nothing special about the d-dimensional subspaces, and therefore, the explanation for this peak *cannot* possibly depend on the lower-dimensional hyperplanes. In fact, considerations about D-dimensional parameter space appear to be sufficient to describe some of the behaviors.

There is a simple explanation for why radial confinement causes the accuracy to drop for $r_0 \lesssim r_X$; namely, there is 'not enough room' to find a good solution if r_0 is too small. Recall that volume in high dimensions scales as r^D, implying that if the radius is doubled, the amount of volume increases by an enormous factor of 2^D. There is exponentially more volume far away from the origin than there is close to the origin; therefore, all else being equal, it is *much* more likely for there to be good solutions far from the origin than close to the origin. In this light, the left half of Fig. 2b is not surprising: it simply indicates that there are more possibilities for solutions if the radius is larger, and when the radius is held fixed, the trajectory cannot access solutions in the larger space further away. (Of course, this explanation does not explain the dropoff on the right half of Fig. 2b—I will discuss this shortly.)

In fact, a naive geometrical explanation can also explain the left half of Fig. 2a—why the validation accuracy after 15 epochs is the same for all $r_0 \lesssim r_X$.

A. Random walk model of the trajectory

Figure 3 indicates that all the unconstrained trajectories initialized at $r_0 < r_X$ end up near $r \sim r_X$ after 15 epochs of training. This may seem surprising at first, but it is in fact completely expected: even a *random walk* exhibits the same behavior.

Of course, it is frivolous to think of the training trajectory as a completely random process. There is indeed some stochasticity due to the finite batch size, but (one would hope!) the dynamics are dominated by the deterministic gradient directing the trajectory towards a region of low loss.[5] Nonetheless, as a 'zero-order' assumption for how trajectories in high-dimensional spaces tend to behave, it is useful to consider how a random walk behaves—if only to understand what is expected and what is unexpected.

So let us describe the trajectory as $\vec{r}(t) = \vec{r}_0 + \vec{s}(t)$, where $\vec{r}_0$ is the initial condition and $\vec{s}(t)$ is the sum of many random steps (i.e., a random walk). Suppose that each time the parameters are updated, we take a step in a random direction with variance σ^2 in each component. After one step, the total variance of all the components $D\sigma^2$ (assuming that the components are independent), and after t steps, the variance is $D\sigma^2 t$ (assuming that the time steps are independent). For large enough D and t, the central limit theorem tell us that $\vec{s} \sim \mathcal{N}(0, D\sigma^2 t)$. Hence the mean squared radial distance from the origin grows as

$$\langle |\vec{r}(t)|^2 \rangle = r_0^2 + \cancel{\vec{r}_0 \cdot \langle \vec{s} \rangle} + \langle s^2 \rangle \quad (7)$$

$$= r_0^2 + D\sigma^2 t. \quad (8)$$

One term comes from the initialization radius, and the other term comes from the random walk diffusing through space.

[5] There is some truth to the random walk model, though: namely, since vectors in high dimensional spaces are almost always orthogonal, it is likely that subsequent steps in a training trajectory are not as 'directed in the same direction' as one might expect. This means that the independence-between-steps assumption of a random walk might not be totally unreasonable. I did not have time to experimentally investigate this question, though.

Depending on which term is larger, we have two regimes of behavior: when $r_0^2 \gg D\sigma^2 t$, the radius stays constant over time, and when $r_0^2 \ll D\sigma^2 t$, the radius grows as the square root of time.

This simplistic model properly accounts for the trajectories depicted in Fig. 3. When r_0 is large, r stays constant; when r_0 is small, r grows as $\sqrt{t}$, and in the intermediate regime, r crosses over from one behavior to the other. Furthermore, the random walk model predicts that $r^2 - r_0^2$ should grow linearly in time, and in the right panel of Fig. 3, we indeed observe this behavior when r_0 is not too large. The initial downturn for the larger r_0's suggests that the gradient is trying to pull the model towards smaller values of r—at least initially—but after a while, the slight random steps in all the directions start to add up enough to make the radius grow in the characteristic $\sqrt{t}$ manner.

Hence one should not be surprised that a trajectory initialized at small r_0 eventually grows out to $r \sim r_X$: even a random walk will increase in radius in the same manner. Indeed, in high dimensions, a step in a random direction is far more likely to go away from the origin than towards the origin. (This fact is related to our earlier observation that there is much more room far away than at small r.) For a similar reason, it is difficult a trajectory initialized at a faraway r_0 to *decrease* substantially in radius over the course of training. This highlights the importance of properly initializing a neural network: the trajectory can only 'get so far', so it must start off close to a region of many good solutions in order to find one in a reasonable time. This reasoning also suggests a hypothesis for why the accuracy at large r_0 remains so low after 15 epochs: perhaps the characteristic 'lengthscale' of significant variations in $J(\vec{r})$ are larger, and one needs to train for longer (or increase the learning rate) in order to get far enough to find the good regions of parameter space.

The above discussion gives the proper explanation for why the radial distances initialized at small r_0 end up near $r \sim r_X$: it is because r_X is roughly the typical *distance* $|\vec{r} - \vec{r}_0|$ travelled by the training trajectory. Oddly, it appears that the distance traversed in 15 epochs is roughly equal to the Goldilocks Zone radius.[6] Note that distance travelled depends on the 'velocity' (learning rate times the typical magnitude of gradient) as well as the timescale of training (number of epochs). For future experiments, it will be interesting to investigate whether these parameters influence the location of the Goldilocks Zone.

This random walk behavior was only observed for the $d = D$ trajectory. For the models trained on the lower-dimensional subspaces, the behavior often looked different, sometimes with a larger exponent, and sometimes smaller. When fitting to the MNIST dataset the radial trajectory was different still: it appeared to plateau towards some value. So the $r \propto t^{1/2}$ behavior is not universal, and one should not take the random walk idea too seriously. Nonetheless, the key result still holds: if the radius starts off small, it is unsurprising for it to grow larger over the course of training.

[6] Is this a coincidence or is this expected?

B. Relation between radius and regularization

It is worth noting the connection between the radial coordinate r and regularization. Standard L2 regularization adds an extra term $\lambda|\vec{r}|^2$ to the loss function. Adjusting the parameter λ causes the model to be biased towards smaller or larger values of $|\vec{r}|^2$. Much like the Goldilocks Zone, the validation accuracy is the highest when the value of λ is not too small and not too large.

One may wonder whether the observed Goldilocks behavior is just an example of overfitting and underfitting. This appears not to be the case: the radially constrained model in this paper is underfitting for *both* the small r_0 and the large r_0 regimes. (In the middle region, the model is actually overfitting with a training accuracy of $\sim 70\%$ (data not shown).) The underfitting at large r_0 gives further evidence to the idea that perhaps the model is not given enough time at these large radii to find solutions—or maybe even that the training dynamics chosen for this project (Adam) were unsuitable to navigate the texture of the loss landscape in that region of parameter space.

In follow-up work, it will be interesting to investigate how the regularization term biases the trajectory—how much it causes the radius to shrink over the course of training, for example—and why it leads to regions of higher generalizability.

VI. Conclusion

In summary, I report that fully-connected neural networks trained on the CIFAR-10 dataset exhibit a Goldilocks Zone, as had been previously observed for the the MNIST dataset. The phenomenon occurs even when the training trajectory is not constrained to lower-dimensional subspaces. I gave a naive argument for why the behavior in the $r \lesssim r_X$ regime is unsurprising, and then I speculated about why the accuracy falls off when $r \gtrsim r_X$.

Acknowledgment

The author would like to thank Stanislav Fort for inspiring this project idea and for providing stimulating discussions.

References

[1] S. Fort and A. Scherlis, "The goldilocks zone: Towards better understanding of neural network loss landscapes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3574–3581, 2019.

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[4] S. Fort and S. Jastrzebski, "Large scale structure of neural network loss landscapes," *arXiv preprint arXiv:1906.04724*, 2019.

[5] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the intrinsic dimension of objective landscapes," *arXiv preprint arXiv:1804.08838*, 2018.

[6] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.