GEORGIA INSTITUTE OF TECHNOLOGY
ECE 6254 STATISTICAL MACHINE LEARNING

# The Effect of Hyperparameters on Underspecification in Neural Networks

*Zoe Fowler*

*Eloy Geenjaar*

*Evgeniya "Jane" Ivanova*

*Jorge Quesada*

May 9, 2022

# 1 Summary and motivation

Deep learning as a field has been growing at a relentless rate, fueled by increases in data and compute. Especially neural networks have been outperforming more classical machine learning methods in tasks such as natural image classification, object recognition, and natural language processing. Their success in academia has made them attractive for use in practical applications as well as other scientific domains. The uninterpretability of neural networks is an issue in a lot of related fields and practical applications, however. In particular, their sensitivity to initial conditions and ability to find multiple different functions with similar performance can be an issue when these models are deployed a real-world setting [1]. This phenomenon has recently been explored more in-depth and is called underspecification [2]. Underspecification can be a problem because it can affect generalization and robustness. Hence, additional expert knowledge is required to select the function that will best generalize to new data.

In this context, we want to understand how equivalently good functions with respect to classification accuracy differ for a simple classification task by observing how the gradients, trained parameters sets, and representations of the same model change. To sample different functions, we train the same model with different initial seeds. In addition, we construct an evaluation metric to quantify the 'distance' in solution space between these sampled functions. We assess the effect of L1 regularization and what we call 'model knowledge' on this distance to find whether these hyperparameters reduce the size of the space of equivalently good solutions, see Figure 1.1. We define 'model knowledge' as the number of classes it has to predict in the classification problem, e.g. predicting $2$ classes provides the model with $2$ bits of information about the problem. Predicting all $10$ classes in the classification problem allows the model to learn information about each digit separately.

The results we find align with our intuitions with respect to regularization and model knowledge; well-tuned L1 regularization based on accurate inductive biases decreases the equivalent solution space and so does model knowledge.

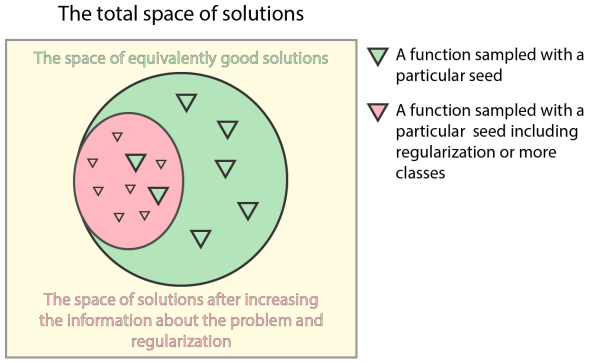


Figure 1.1: Solution space reduction under regularization

# 2 Methodology

**Model and Dataset Choice:** We choose to use a two-layer Multilayer Perceptron (MLP) with a ReLU activation function, shown in Figure 2.1a, as the model in this report because it is a fundamental and more easily understood model [3, 4]. The first layer in the MLP extracts features from the image, which are mapped to class predictions after the activation function. We refer to these features as embeddings and use an embedding size of $16$, which is larger than the maximum number of classes.

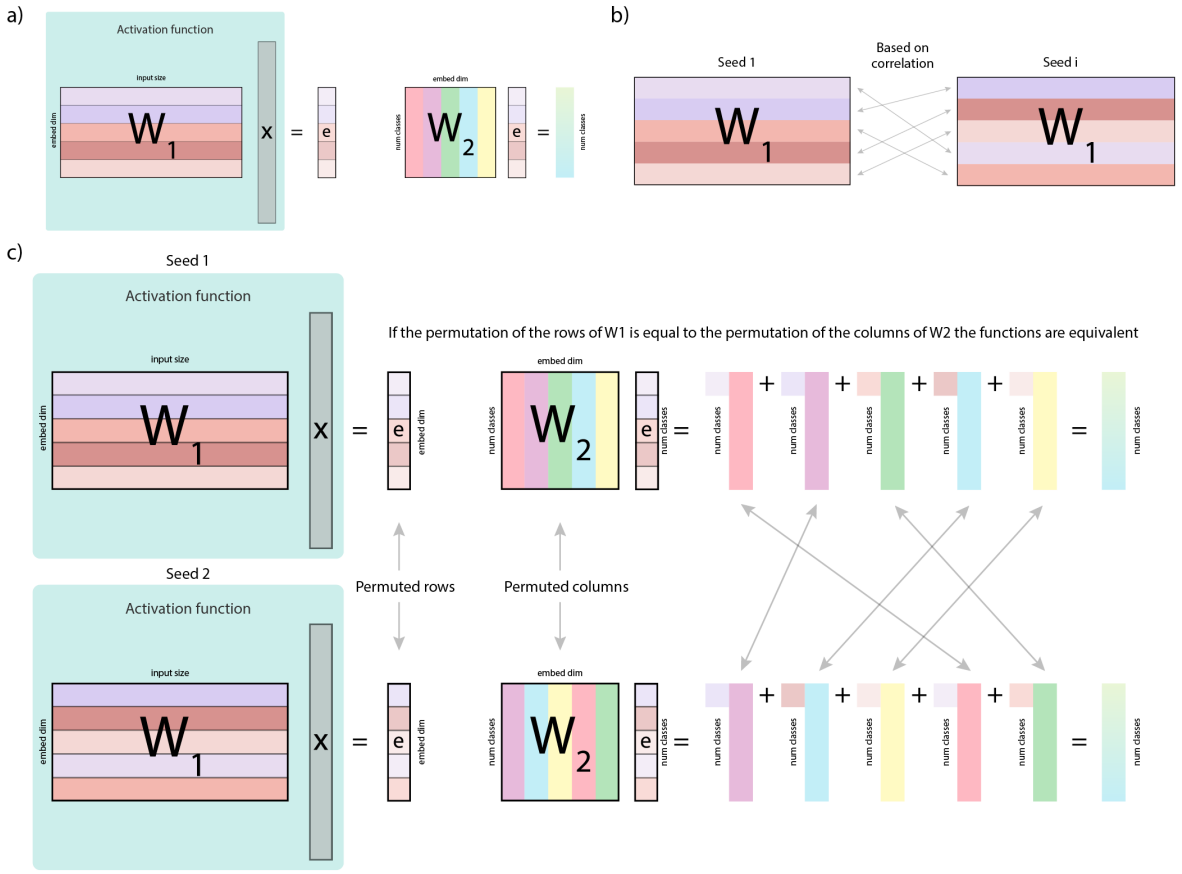


Figure 2.1: Structure of our simple two-layer MLP. The input (grey) is projected down to an embedding dimension through the weight matrix $W_1$, and the resulting representation $e$ is then used to predict the probability of each class through the weight matrix $W_2$

Initially, we focused on understanding and visualizing the changes of the parameters, gradients, and representations over time during training for different seeds. These visualizations are available on our project repository [1]. After gaining some intuition with respect to the trajectories that the parameters, gradients, and representations take, we started thinking about an appropriate measure to quantify the distance between sampled functions. One important insight is that the exact function of the model is the same under permutation of the rows of the parameters leads to a permutation of

[1] https://github.com/eloygeenjaar/hyperparameters-underspecification

embedding dimensions, see Figure 2.1b. If this permutation is exactly matched by a permutation of the parameters in the columns of the second weight matrix, the two functions are equivalent, see Figure 2.1c. It is important to take this into account when calculating the similarity between sampled functions. To calculate the similarity between parameters in the rows of the first weight matrix and the columns of the second weight matrix across sampled functions, we use their correlation. To make sure we take the potential permutation of parameters into account, we calculate a correlation matrix between the rows of the weight matrix for the first seed and all other seeds. Then we solve the linear assignment problem using the Hungarian algorithm [5] to assign the highest correlated rows for each seed to the rows of the first seed. Then, we can calculate the absolute correlation between the weight matrix of the first seed and all other seeds, while still taking the permutation equivalence into account. This metric, which we call the absolute permutation correlation (APC) is what we use to evaluate the similarity between sampled functions for the first and second matrix. On top of the APC, we also calculate how many percent of the rows in the first weight matrix are permuted exactly the same as the columns in the second weight matrix. We call this second measure the permutation correlation equivalence (PCE), which is calculated for each seed with respect to the first seed, and the APC is calculated the same way for both the first and second weight matrices.

**Effect of Hyperparameters:** In order to observe the effect of the regularization on the underspecification in the model, we compare the percentage of weight permutations in the unconstrained optimization problem and regularization on the $\ell_1$-norm applied to the first layer of the model by altering the value of the regularization parameter $\lambda$. For the purpose of this project we take the range of the $\lambda$ between $0.01$ and $0.05$ ($\lambda = 0$ signifies no regularization). We have chosen the $\ell_1$-norm because given an image dataset, a small number of nonzero parameters in the network will correspond to removing irrelevant information. We expect to see that learnt parameters will be smaller for higher values of $\lambda$ and will have a smaller amount of nonzero entries [6]. In addition, we are interested in investigating how altering the model knowledge affects the size of the space of equivalently good solutions. As mentioned previously, we hypothesize that increasing the model knowledge will narrow this solution space (as supported by [7]).

**Gradients:** We expect that over time the gradients will converge to very small values, independently of achieving either local or global minima [8]. In addition, we examine the location of the gradients in the W1 layer across different seeds. For example, for a given image of a digit, we expect that the gradients would be concentrated in the area where the digit is located, generally ignoring 'background' pixels that don't provide useful information for the classification task.

The weights and biases of the gradients are extracted for each weight in the model across seeds, allowing us to visualize how the weights and biases of the gradients

change over time, as well as how the variance of the gradients changes across different seeds. The cumulative sum of the gradients is then taken for each training step, and the final cumulative sum is evaluated using the aforementioned metrics.

**Representations:** We additionally analyzed how the representations attained by the network during training move along the low-dimensional embedding space, and whether or not there is any correspondence in the behavior of these representations across different training seeds. If the pattern of the data points is spread out and shows some degree of consistency across seeds, that would be an indirect indicator that there are some correspondences between the functions being learnt [9]. To do this, we extract the output of the embedding layer for all training data after each epoch, and after each training instance get the first two PCA components corresponding to the $k$-th seed. We then study how the embeddings move through the space of the first two PCA components. Additionally, in order to observe the effect of the regularization on the underspecification in the model, we compare the evolution of the representations for different configurations of regularization parameter ($\lambda$) and visually evaluate whether or not it influences representation consistency.

**Experimental setup:** In order to ensure that the only source of randomness in our training instance comes from the choice of different random seeds, we made sure to set make all stochastic processes in the training of the model based on the seed, such as the dataloader sampler and the CUDA operations in Pytorch, and other stochasticity in Python in general. We ran 54 jobs on a cluster with 5 cores and 120GB RAM, where each job trained the model for 50 epochs for each of the $7$ random seeds.

# 3 Results

In Figure 3.1, we show the dependency between the regularization parameter $\lambda$, the model knowledge, and the permutation percentage (PE) metric for the network parameters and gradients. As the number of classes increase, the PE increases, which supports our hypothesis that as the model knowledge grows during training, there seems to be a narrower set of solutions based on the functions we sample. In addition, we observe that as $\lambda$ increases, there is an increase in the PE value, up to a certain point ($0.005$) and then the PE decreases. This also supports our hypothesis because higher $\lambda$ values will decrease the number of non-zero weight values, which should reduce the solution space. For $\lambda$ values that are too high, this reduces to a noisy solution, however, where most values are zero. Overall, we observe that the $\lambda$ value of $0.001$ has the highest permutation percentage ($0.48$) for $9$ classes.

In Figure 3.1 (right column), it is clear that the APC-2 values for both the parameters and gradients decrease for the number of classes because the correlations

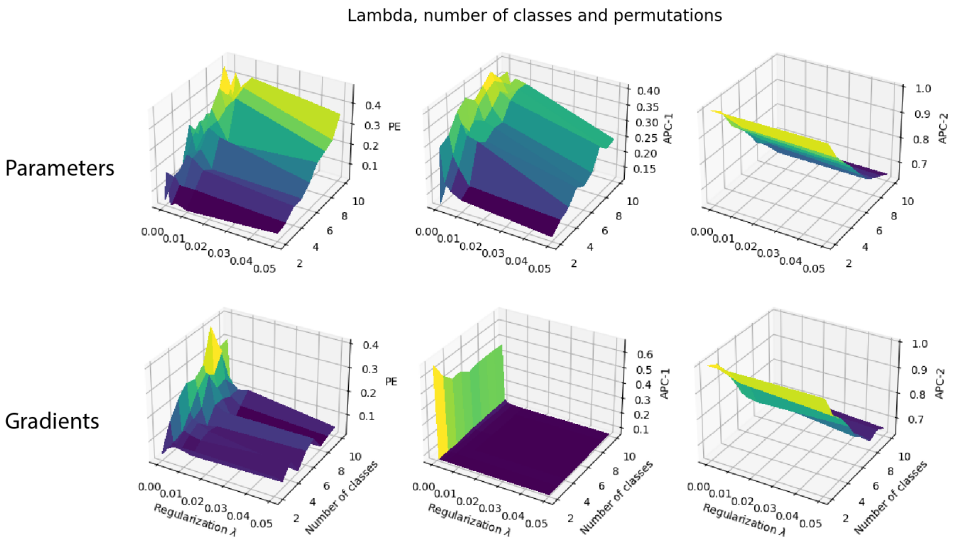


Figure 3.1: Dependency between $\lambda$, class number (2-10) and PE for parameters and gradients. The highest values are attained for the smallest possible $\lambda = 0.001$. PE is the permutation percentage (left column), APC-1 (middle column) and APC-2 (right column) are the APCs for the first and second weight matrices, respectively

.

are calculated over more values (the size of the vector depends on the number of classes). Furthermore, the regularization does not affect the APC-2 for the gradients and parameters because the regularization is not applied to the second weight matrix.

In general, the gradients converge to small values, as expected, which is consistent across different seeds once the training process has stabilized. In addition, there appears to be quite a bit of variation in these quantitative gradient values across different seeds, showing that analyzing the values of these gradients can demonstrate the effects of underspecification on the trajectory of the model parameters during training. An interesting result we find when we compute the APC, and PE metric for the gradients is that the regularization has a much larger effect on the gradients than it does on the parameters. For any regularization at all, the APC-1 reduces to near zero for the weight matrix (bottom, middle in Figure 3.1) and even for the permutation percentage (bottom, left in Figure 3.1).

The behavior of the learned representations across different training stages can be observed as an additional material on our aforemenioned project repository. As we can see, even when starting from different random initializations, after a few iterations the pattern in which the samples are clustered is highly consistent across all seeds. There seems to be some variance in the representations, although the embeddings seem to be largely similar along the largest two PCA components. This may indicate that although the functions are different across seeds, the representations for the training and validation set are largely similar.

# 4 Conclusions

Our results indicate that there is a higher correspondence between the sampled functions in the constrained solution space with respect to the unconstrained solution space (no regularization or less model knowledge). The correspondence is evaluated by our novel APC and PE metrics, and clearly indicates that as we increase the information available to the model during training, the size of the solution space decreases, thus improving the reliability of the solution and alleviating underspecification.

# References


[1] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[2] Alexander D’Amour et al. “Underspecification presents challenges for credibility in modern machine learning”. In: *arXiv preprint arXiv:2011.03395* (2020).

[3] Wei Hu et al. “The surprising simplicity of the early-time learning dynamics of neural networks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17116–17128.

[4] Sanjeev Arora et al. “Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 322–332.

[5] H. W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. DOI: https://doi.org/10.1002/nav.3800020109.

[6] Gen Li, Yuantao Gu, and Jie Ding. “The Efficacy of $L\_1$ Regularization in Two-Layer Neural Networks”. In: *arXiv preprint arXiv:2010.01048* (2020).

[7] Yasaman Bahri et al. “Statistical Mechanics of Deep Learning”. In: *Annual Review of Condensed Matter Physics* 11.1 (2020), pp. 501–528. DOI: 10.1146/annurev-conmatphys-031119-050745.

[8] Sanjeev Arora et al. “A convergence analysis of gradient descent for deep linear neural networks”. In: *arXiv preprint arXiv:1810.02281* (2018).

[9] Simon Kornblith et al. “Similarity of neural network representations revisited”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3519–3529.