Is network fragmentation a useful complexity measure?

Coenraad Mouton[1,2] **Randle Rabe**[1,2] **Daniël G. Haasbroek**[1,2,3]
Marthinus W. Theunissen[1,2] **Hermanus L. Potgieter**[1,2,3] **Marelie H. Davel**[1,2,4]
[1]Faculty of Engineering, North-West University, South Africa
[2]Centre for Artificial Intelligence Research, South Africa
[3]South African National Space Agency
[4]National Institute for Theoretical and Computational Sciences, South Africa
{moutoncoenraad, randlerabe}@gmail.com

Abstract

It has been observed that the input space of deep neural network classifiers can exhibit 'fragmentation', where the model function rapidly changes class as the input space is traversed. The severity of this fragmentation tends to follow the double descent curve, achieving a maximum at the interpolation regime. We study this phenomenon in the context of image classification and ask whether fragmentation could be predictive of generalization performance. Using a fragmentation-based complexity measure, we show this to be possible by achieving good performance on the PGDL (Predicting Generalization in Deep Learning) benchmark. In addition, we report on new observations related to fragmentation, namely (i) fragmentation is not limited to the input space but occurs in the hidden representations as well, (ii) fragmentation follows the trends in the validation error throughout training, and (iii) fragmentation is not a direct result of increased weight norms. Together, this indicates that fragmentation is a phenomenon worth investigating further when studying the generalization ability of deep neural networks.

1 Introduction

Understanding and predicting the generalization ability of neural networks is an active area of research, both from an empirical and theoretical perspective. Of particular interest is the empirically observed phenomenon of 'double descent'. Double descent (DD) refers to the observation that when measuring generalization error as a function of increasing capacity, deep neural networks (DNNs) do not exhibit the U-curve typical of classical machine learning but can, in addition, produce a second descent where generalization error continues to decrease as model capacity increases beyond a critical point [1].

Various studies investigate DD and the underlying mechanisms by which it occurs [2, 3, 4, 5]. Recently, Somepalli et al. [6] observed that critically parameterized DNNs exhibit 'fragmentation', which refers to the observation that a DNN's input space predictions can fragment into many different class regions. In addition, it was shown that the degree of fragmentation exhibited by a model is highly correlated with its test error when traversing the DD curve.

Investigations of DD are closely related to studies that attempt to define a useful measure of 'network complexity' [7, 8, 9, 10, 11], meaning a measure that is capable of accurately measuring the effective, as opposed to representational, complexity of a DNN model's learned function. Complexity measures for DNNs are commonly investigated in the empirical context of predicting or ranking the generalization ability of a group of models by only relying on training data.

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

In this work, we investigate both these areas (DD and complexity measures) from the perspective of network fragmentation. We show that not only does fragmentation correlate with test error in the controlled DD setting, but also in the more general setting of comparing models trained with different hyperparameter and architectural setups in the context of generalization prediction. Furthermore, we provide additional insights into the fragmentation phenomenon by studying its relation to hidden layers, the weight space, and dynamics during training.

2 Fragmentation and double descent

We first confirm the observation by Somepalli et al. that input space fragmentation follows a DD [6], before ascertaining whether fragmentation can also be observed at the models' hidden layers. We describe how fragmentation is measured, develop a set of models to elicit DD, and define a straightforward process to investigate fragmentation in this context, before presenting results.

Measuring fragmentation We measure fragmentation in the same way as Somepalli et al. [6] but alter the definition to apply to any representational space within a DNN. Specifically, we sample 3 random training samples (a triplet) and create a plane spanned by these samples. We then densely sample this plane and record the top-class predictions for each point within the plane. The fragmentation of the plane is measured by counting the number of distinct classification regions (sets of connected points with the same class prediction). When considering hidden layers, the process remains identical except that the plane is created by using the representations of the 3 training samples found at a specific hidden layer in the network. See Appendix A.1 for additional detail.

DD models We train two sets of CNN models with varying representational capacity on the CIFAR10 dataset: one set using the original data set, and the other using a label-corrupted version, in order to produce a pronounced DD. Specifically, 10% of samples in the training set are randomly assigned a different label. We make use of an architecture similar to that of the 'standard CNN' used by Nakkiran et al. [2], which consists of 4 convolutional layers with $[k, 2k, 4k, 8k]$ output channels, respectively. We choose values of k between 4 and 64 to create a group of models with varying capacity. We train each set using 3 different initialization seeds. The train and test error for both sets of models are shown in Figure 1. As expected, we observe a pronounced DD for the label-corrupted models. See Appendix A.2 for details on other training hyperparameters.

Procedure For each model in each set, we measure the mean fragmentation as the average fragmentation over 500 randomly sampled triplets. We construct the triplets by randomly sampling 3 samples of the same class (without replacement). Furthermore, we use the same triplets for each model, and each triplet plane is sampled using $2\ 500$ equally spaced points. When measuring fragmentation at the hidden space, we measure the fragmentation at the output of each convolutional layer (post activation function) for each model.

Results We show the mean fragmentation score per model (averaged over 3 seeds) in Figure 2, which includes both the input space (left) and the hidden space of the first convolutional layer (right).

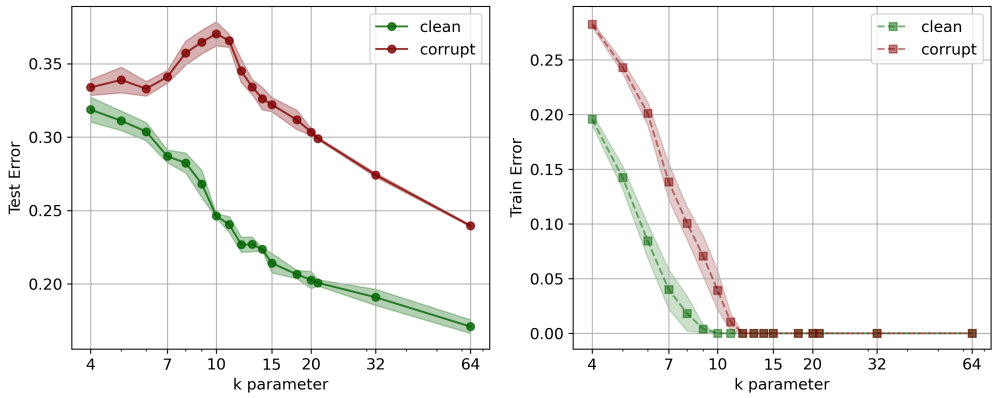

Figure 1: Error versus model capacity for models trained on partially label-corrupted data (red) and clean data (green). Left: Test error. Right: Train error.

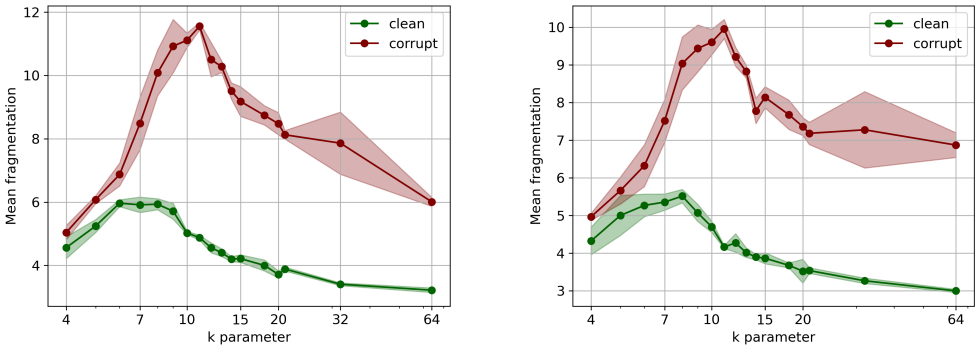

Figure 2: Mean fragmentation versus model capacity. Left: Input space. Right: Hidden space fragmentation for the first convolutional layer. 'Clean' and 'corrupt' refers to models trained on clean or partially label-corrupted data, respectively. Results are averaged over three seeds. The shaded regions indicate the error (standard deviation).

We observe that our input space results closely match those of Somepalli et al.; in particular, the mean fragmentation curve closely follows the test-error curve (see Figure 1 in Appendix A.2), and a clear second descent is visible. We also confirm that the mean fragmentation score is more pronounced for the corrupt than the clean case. When we perform a similar analysis but measure the mean fragmentation of selected models during training (as opposed to after training) we find a similar correlation with validation error. (See Figure 5 in Appendix B.2.)

The fragmentation at the hidden space of the first convolutional layer shows similar trends to the input space, and we also find the same behavior at the other hidden layers, see Appendix B.1. Furthermore, we also observe that the mean fragmentation gradually decreases as function of depth, implying the fragmentation of the classification regions are unevenly distributed throughout the network. These results confirm the observations of Somepalli et al., and extends their results to fragmentation behaviour during training, as well as to the networks' hidden representation space.

3 Fragmentation and generalization prediction

In the previous section we have shown that fragmentation is highly correlated with test error in the controlled DD setting. We now turn our attention to the more general setting of determining the relationship between fragmentation and generalization performance for models with varying hyperparameters and architectural setups. Specifically, we rely on the Predicting Generalization in Deep Learning (PGDL) [12] benchmark.

The PGDL benchmark is a 'dataset' of trained CNN image classification models split into 8 different tasks. Each task refers to a set of models of a certain architectural family trained on the same dataset. The goal of this benchmark is to develop some measurement of the model and its training data that is able to rank the models within each task according to their generalization gap, i.e. a complexity measure. The complexity measure is then evaluated by determining how well this ranking aligns with the true generalization gap ranking.

The tasks within the dataset are split into two groups. Tasks 1 to 5 form the development set, with the intended usage of developing and refining a complexity measure. Tasks 6 to 9 forms the test set, which should be used as a held out set to evaluate the complexity measure, i.e. these tasks should only be evaluated once, to prevent overfitting.

3.1 Refined metrics

In addition to the original fragmentation metric, we define two variations that take into account other characteristics of the classification regions. Where the original metric considers the number of regions, these metrics use the relative sizes of the regions. Specifically, we hypothesize that generalization

performance is negatively correlated with larger 'foreign' regions, and, therefore, we consider the fraction of the sampled-region area covered by these 'foreign' classification regions.

We define *foreign-region coverage* as the combined area of all regions that do not contain any of the triplet points, divided by the total area of the sampled region. Similarly, we define *foreign-class coverage* as the combined area of any regions that do not have the same class as any of the triplet points, divided by the total area. These metrics allow us to explore the correlation between generalization and the relative sizes of the 'foreign' regions.

3.2 Predicting generalization

We calculate the mean fragmentation, foreign-region coverage, and foreign-class coverage for each model and task in the PGDL dataset. We rely on the same settings as used earlier in Section 2. To assess the relationship between each complexity measure and the true generalization gap of the models, we calculate the conditional mutual information (CMI) between the two, conditioned on each set of hyperparameters[1], which is the standard evaluation metric for the PGDL benchmark [12]. In addition, we also compare our scores to three other complexity measures that currently have the highest average performance on the test set of tasks that we are aware of: DBI*LWM [14], PCA Gi&Mi [15], and Constrained Margin [16].

The CMI scores for our *input space* fragmentation metrics and those we compare with are shown in Table 1. Note that we present the mean scores for the PGDL development set (Tasks 1 to 5) and test set (Tasks 6 to 9) separately.

Table 1: Conditional Mutual Information (CMI) score for several complexity measures on the PGDL benchmark. 'FRC' refers to foreign-region coverage, and 'FCC' refers to foreign-class coverage. 'Dev mean' is calculated as the average score over Tasks 1 to 5, and 'Test mean' over Tasks 6 to 9. There is no Task 3. The top two solutions for each task are highlighted in bold.

Task	Natekar and Sharma [14]	Schiff et al. [15]	Mouton et al. [16]	Fragmentation metrics		
	DBI*LWM	PCA Gi&Mi	Constrained Margin	Fragmentation	FRC	FCC
1	00.00	00.04	**39.49**	**03.25**	00.08	00.08
2	32.05	**38.08**	**51.98**	13.09	26.16	27.64
4	**31.79**	**33.76**	21.44	15.92	29.31	29.49
5	15.92	20.33	04.93	**31.23**	**23.41**	22.61
Dev mean	19.94	**23.05**	**29.46**	15.87	19.74	19.96
6	**43.99**	40.06	30.83	37.14	**40.32**	40.10
7	12.59	13.19	13.26	02.53	**16.45**	**16.45**
8	09.24	10.30	**13.48**	**10.91**	09.84	09.99
9	25.86	33.16	**51.46**	17.46	37.00	**37.24**
Test mean	22.92	24.18	**27.26**	17.01	25.90	**25.94**

We observe that fragmentation and the two additional fragmentation-based metrics achieve competitive scores. This shows that an increase in fragmentation is correlated with a decrease in generalization performance. Interestingly, we find that foreign-region coverage and foreign-class coverage outperform the base fragmentation score on average, which points to the notion that considering the size of the classification regions is beneficial. Furthermore, we note that foreign-class coverage achieves the second-highest mean test set performance, only trailing behind the recently proposed constrained margin complexity measure [16]. We find that input space fragmentation is more predictive of generalization than hidden space fragmentation, a metric we investigate in Appendix B.3.

4 Discussion and conclusion

We have shown that fragmentation can be a useful complexity measure of deep neural networks. Not only does fragmentation correlate well with test error in the DD setting, fragmentation-based metrics can be utilised as a measure for generalization prediction.

[1]This metric measures the minimal conditional mutual information across all sets of hyperparameter combinations. See Jiang et al. [12, 13] for a comprehensive explanation.

While we have shown the utility of fragmentation as a complexity measure, the underlying mechanisms causing classification regions to fragment remains unknown. We believe that fragmentation is closely related to the smoothness of the learned function. In fact, Somepalli et al. hypothesize that at the point of interpolation on the DD curve, there are oscillations that result in poor output function behaviour which causes the observed instability around class boundaries and that these oscillations are not needed for interpolation.

There are several other related works that study the smoothness of the input-output mapping in DNNs. Gamba et al. [11] propose a local measure of model complexity (nonlinearity) called *absolute deviation*. They showed that, at the point of interpolation, the model output function was measured to be at its most 'complex' or nonlinear, resulting in comparatively less smooth functions and that the smoothness of the functions increased as model capacity moved into the overparameterized regime. Recently, it was demonstrated by Teney et al. [17] that ReLU networks at initialization have an implicit bias towards smoothness and that this bias is independent of the model width, depth, or weight magnitude. Similarly, we note that fragmentation is very low at model initialization (see Figure 5).

We believe that understanding the underlying weight space mechanisms that cause the instability in the classification regions (or alternatively, the reduction in smoothness) is a logical next step. We have made some preliminary strides towards this goal, such as determining that the causes of fragmentation are distributed throughout the network (see Appendix B.1). We have also found that the underlying causes are not trivial, such as due to exploding weight norms (see Appendix B.4).

In conclusion, we have investigated fragmentation as a complexity measure. We find that it shows promise in measuring DNN complexity. In addition, we hypothesize that a deeper understanding of the underlying mechanisms that cause fragmentation may shed light on more general questions related to the change in smoothness of the output function as capacity changes.

Acknowledgments and Disclosure of Funding

This work is based on research supported in part by the National Research Foundation of South Africa (Ref Numbers *PSTD23042296065*, *PSTD23042898868*, *RA211019646111*).

References

[1] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[2] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

[3] Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under- to over-parametrization affects loss landscape and generalization. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001, 2019.

[4] Jimmy Ba, Murat Erdogdu, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Generalization of two-layer neural networks: An asymptotic viewpoint. In *International Conference on Learning Representations*, 2020.

[5] Stéphane D'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance(s) in the lazy regime. In *International Conference on Machine Learning*, 2020.

[6] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? Investigating reproducibility and double descent from the decision boundary perspective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[7] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[8] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, 2017.

[9] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[10] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, 2019.

[11] Matteo Gamba, Adrian Chmielewski-Anders, Josephine Sullivan, Hossein Azizpour, and Marten Bjorkman. Are all linear regions created equal? In *International Conference on Artificial Intelligence and Statistics*, 2022.

[12] Yiding Jiang, Pierre Foret, Scott Yak, Daniel M Roy, Hossein Mobahi, Gintare Karolina Dziugaite, Samy Bengio, Suriya Gunasekar, Isabelle Guyon, and Behnam Neyshabur. NeurIPS 2020 competition: predicting generalization in deep learning. *arXiv:2012.07976*, 2020.

[13] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

[14] Parth Natekar and Manik Sharma. Representation based complexity measures for predicting generalization in deep learning. *arXiv:2012.02775*, 2020.

[15] Yair Schiff, Brian Quanz, Payel Das, and Pin-Yu Chen. Predicting deep neural network generalization with perturbation response curves. In *Advances in Neural Information Processing Systems*, 2021.

[16] Coenraad Mouton, Marthinus Wilhelmus Theunissen, and Marelie H. Davel. Input margins can predict generalization too. In *AAAI Conference on Artificial Intelligence*, 2024.

[17] Damien Teney, Armand Nicolicioiu, Valentin Hartmann, and Ehsan Abbasnejad. Neural redshift: Random networks are not random functions. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[18] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 06 1938.

[19] Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

A Experimental details

A.1 Triplet Plane and Fragmentation

We measure fragmentation in the same way as [6], although we alter some definitions to allow for measuring the fragmentation at any representational space within a DNN.

Consider an image dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^r$ consisting of r sample pairs where $\mathbf{x}_i \in \mathbb{R}^n$ is the n-dimensional input and $y_i \in \{0, 1, ..., k-1\}$ the label. Furthermore, we let $f_\theta : \mathbb{R}^n \to \mathbb{R}^k$ denote a family of neural networks with L hidden layers and a set of trainable parameters θ, where $\hat{y}_i = \arg\max f_\theta(\mathbf{x}_i)$ is the model's prediction. Finally, let $\mathbf{x}^\ell$, $\ell \in [0, L]$, correspond to the representation of a sample at a layer ℓ, where $\mathbf{x}^0 = \mathbf{x}$ denotes the input sample. For $\ell > 0$, $\mathbf{x}^\ell$ is the vector formed *after* ReLU activation is applied element-wise.

For any point $\mathbf{x}^\ell$ in the latent space of layer $\ell > 0$, the classification is given by $\arg\max f_{\theta^{\ell+1}}(\mathbf{x}^\ell)$, where $f_{\theta^{\ell+1}}$ is defined as the neural network with layers 0 to ℓ removed from f_θ.

Triplet Plane To construct a 'triplet plane', we randomly sample a triplet set of unique images $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ from $\mathcal{D}$ conditioned on $y_1 = y_2 = y_3$. Then, for a given layer ℓ, we define the vectors $\mathbf{v}_1^\ell$ and $\mathbf{v}_2^\ell$ as $\mathbf{v}_1^\ell = \mathbf{x}_2^\ell - \mathbf{x}_1^\ell$ and $\mathbf{v}_2^\ell = \mathbf{x}_3^\ell - \mathbf{x}_1^\ell$. The plane $P = \text{span}_\mathbb{R}\{\hat{\mathbf{v}}_1^\ell, \hat{\mathbf{v}}_{1\perp}^\ell\}$ consists of all vectors spanned by the orthonormal basis vectors $\hat{\mathbf{v}}_1^\ell, \hat{\mathbf{v}}_{1\perp}^\ell$, where we have defined the orthogonal vector,

$$\mathbf{v}_{1\perp}^\ell = \mathbf{v}_2^\ell - \text{proj}_{\mathbf{v}_1^\ell}\mathbf{v}_2^\ell, \quad \text{where proj}_\mathbf{a}\mathbf{b} = (\mathbf{b} \cdot \hat{\mathbf{a}})\, \hat{\mathbf{a}}. \tag{1}$$

In practice, we sample points from the plane using the algorithm defined by [6]. More precisely, using $\mathbf{x}_1^\ell$ as the base image, we can find any point $\mathbf{s}^\ell$ on the plane P with respect to the standard (feature) basis using the equation,

$$\begin{aligned}\mathbf{s}^\ell &= \mathbf{x}_1^\ell + k_1\hat{\mathbf{v}}_1^\ell + k_2\hat{\mathbf{v}}_{1\perp}^\ell \\ &= \mathbf{x}_1^\ell + (1-\alpha)\ \min\{0,\ \hat{\mathbf{v}}_1^\ell \cdot \mathbf{v}_2^\ell\}\ \hat{\mathbf{v}}_1^\ell + \alpha\ \max\{||\mathbf{v}_1^\ell||,\ \hat{\mathbf{v}}_1^\ell \cdot \mathbf{v}_2^\ell\}\ \hat{\mathbf{v}}_1^\ell \\ &\quad + \beta\ (\mathbf{v}_2^\ell \cdot \hat{\mathbf{v}}_{1\perp}^\ell)\ \hat{\mathbf{v}}_{1\perp}^\ell,\end{aligned} \tag{2}$$

where $k_1,\ k_2$ are bounded coordinates on the plane and $0 \leq \alpha,\ \beta \leq 1$. To understand the second equality, note that the range of the k_2 coordinate is between 0 and the maximum value along $\hat{\mathbf{v}}_{1\perp}^\ell$, which is determined by the point $\mathbf{x}_3^\ell$. The minimum for the k_1 coordinate is dependent on the sign of $\hat{\mathbf{v}}_1^\ell \cdot \mathbf{v}_2^\ell$ (or, equivalently, the angle between the two vectors). Similarly, its maximum value is determined again by the sign of $\hat{\mathbf{v}}_1^\ell \cdot \mathbf{v}_2^\ell$ as well as the respective magnitudes $||\mathbf{v}_1^\ell||,\ |\hat{\mathbf{v}}_1^\ell \cdot \mathbf{v}_2^\ell|$. The second equality combines these ranges into a single equation parameterized by the variables α, β. Finally, note that we can pad the plane by introducing a hyperparameter $\rho > 0$ and setting the parameter ranges to $-\rho \leq \alpha,\ \beta \leq 1 + \rho$.

Fragmentation By sampling points from the triplet plane and passing the points through $f_{\theta^{\ell+1}}$, this results in a plane consisting of 'classification regions': groups of points that are predicted as the same class. The fragmentation is then calculated by simply counting the number of distinct classification regions within the sampled plane.

A.2 Double descent models

Here we provide additional detail on the DD CNN models described in Section 2. All models are trained on a $45\,000/5\,000$ train-validation split of the CIFAR10 dataset. As mentioned, we train one set on clean data, and another on label-corrupted data. For the label corruption, 10% of the training labels are randomly assigned a different class. The exact same samples are label-corrupted for each model. Note that the validation and test set are never label-corrupted.

Each model consists of 4 convolutional layers, and one dense layer (excluding the output layer). The dense layer has a fixed size of 400 nodes, while each CNN layer has $[k, 2k, 4k, 8k]$ channels, respectively. In terms of optimization, we rely on the Adam optimizer in combination with cross entropy loss. Each models is trained for 500 epochs, without early stopping. We use a batch size of 256 samples and an initial learning rate of 0.001. We combine this with a step decay where the initial learning rate is multiplied by 0.99 every 10 epochs. This protocol ensures that all the models in the

overparameterized regime ($k > 9$ and $k > 11$ for clean and corrupt, respectively) fully interpolate the training data with 0% train error. Refer back to Figure 1 for the relevant train and test error curves.

B Additional results

B.1 Hidden Space Fragmentation

In this section, we provide additional details on fragmentation in the hidden layers.

Figure 3 shows fragmentation in each of the four hidden layers of the CNNs (recall that the hidden layers have channels $k,\ 2k,\ 4k,\ 8k$ where $k \in \{4, ..., 64\}$). The top left plot shows how fragmentation changes as a function of capacity (measured by k) for the first hidden layer of all our CNNs. Likewise, the top right, bottom left, and bottom right figures show fragmentation for the remaining hidden layers.

Each figure exhibits a similar trend to the input space; in particular, fragmentation is observed to be more pronounced in the interpolation regime. Also, similar to the input space, we observe that the models trained on the corrupt data generally achieve higher mean fragmentation score relative to their clean counterparts. Finally, we also note that the mean fragmentation decreases as a function of depth, with the final hidden layer exhibiting the least fragmentation. This is made more apparent in Figure 4 which shows the depth-wise decrease in mean fragmentation for the clean case (left figure) and the corrupt case (right figure).

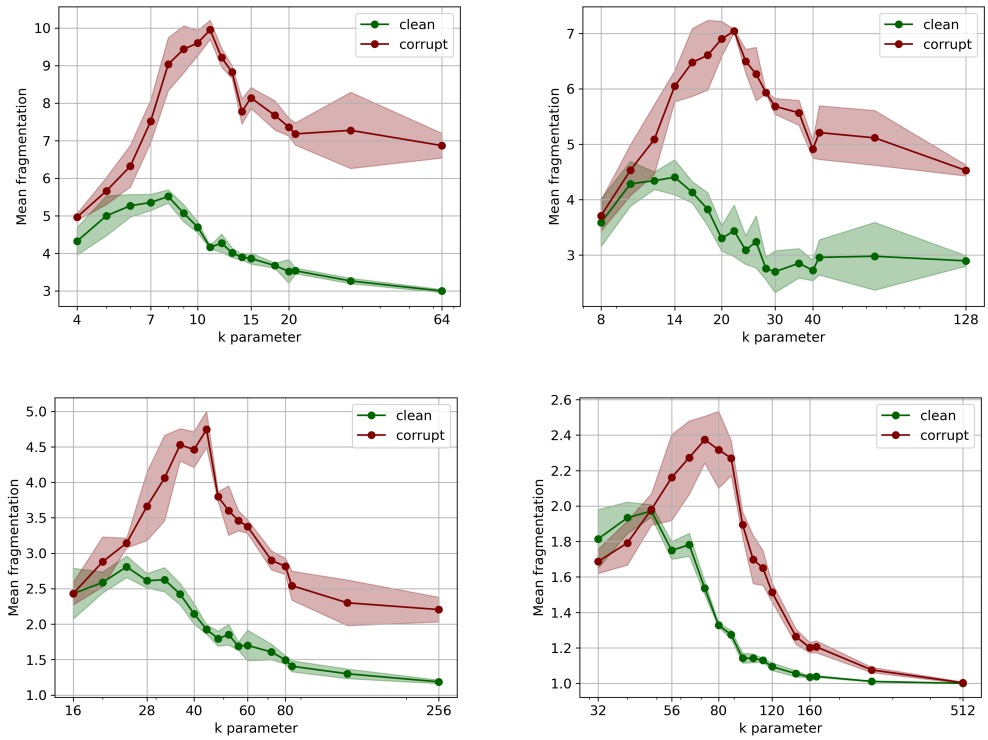

Figure 3: Mean fragmentation per convolutional layer versus model capacity. Channels per layer are given by the pattern $[k,\ 2k,\ 4k,\ 8k]$ where $k \in \{4, ..., 64\}$. First Row: Fragmentation in the hidden space representations for convolutional layers 1 and 2, respectively. Second Row: Fragmentation in the hidden space representations for convolutional layers 3 and 4, respectively. Results are averaged over three seeds. The shaded regions indicate the error (standard deviation).

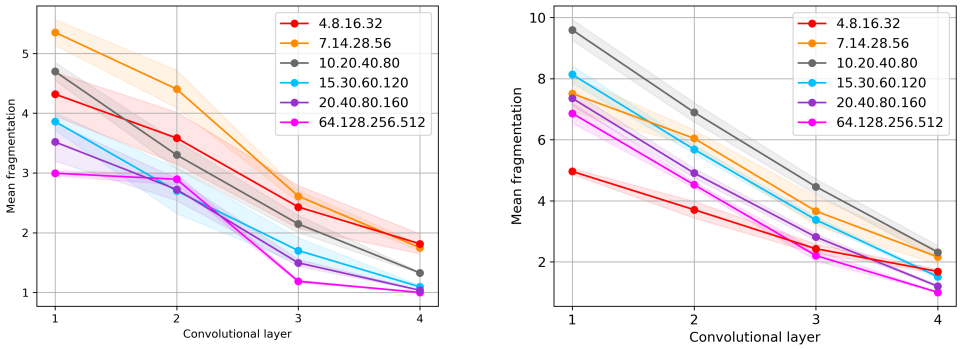

Figure 4: Mean fragmentation as a function of depth. Left: Depth-wise fragmentation on clean CIFAR10. Right: Depth-wise fragmentation on corrupted CIFAR10. Results are average over three seeds. The shaded regions indicate the error (standard deviation). The legend indicates the number of channels in each layer.

B.2 Fragmentation during training

We measure the fragmentation for a subset of the DD models (Sections 2 and A.2) at several points throughout training. For this we use the smallest underparameterized model ($k = 4$), the critically parameterized model ($k = 10$), and the largest over-parameterized model ($k = 64$). The models are trained for $1\,000$ epochs on both the original- and label-corrupted CIFAR10 dataset using the training setup described in Section A.2.

The results in Figure 5 show how fragmentation captures the divergence in validation performance between the clean and corrupt models, despite it being calculated on training data. These results illustrate the correlation between fragmentation and performance on unseen data after the first epoch of training.

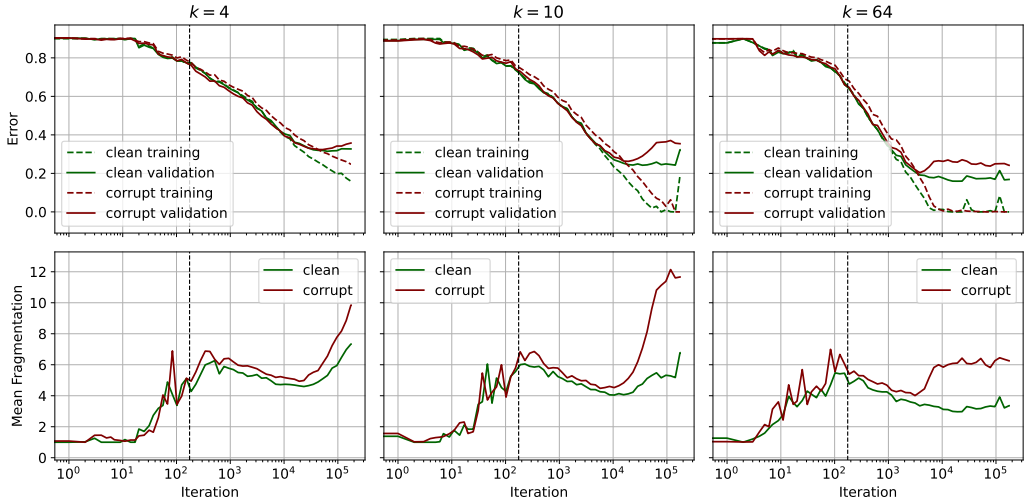

Figure 5: Mean fragmentation throughout training. The dashed line marks the end of the first epoch. Results are averaged over three seeds. The shaded regions indicate the error (standard deviation).

B.3 Hidden layer generalization prediction

In this section we compare the predictive performance of the fragmentation and foreign-class coverage metrics when measured at different layers in each model. Specifically, we consider the performance

when measuring at the first and last hidden layer (separately). This provides us with an indication of how the performance of each metric varies as depth increases.

Previously, in Section 3.2, we used CMI to evaluate the performance of each measure. Here, we rather rely on a more interpretable metric, namely Kendall's rank correlation [18], which provides a score between -1 and $+1$. In this context, $+1$ indicates perfect agreement between the model ranking provided by the complexity measure and the ranking of the true generalization gap of the models. Conversely, -1 indicates perfect disagreement. The Kendall's rank correlation for each metric measured at each layer is shown in Table 2.

Table 2: Kendall's rank correlation for fragmentation and foreign-class coverage measured at different layers on the PGDL benchmark. 'Dev mean' is calculated as the average score over Tasks 1 to 5, and 'Test mean' over Tasks 6 to 9. There is no Task 3. The best performing layer for each metric and task is highlighted in bold.

Task	Fragmentation			Foreign-class coverage		
	Input space	First hidden	Last hidden	Input space	First hidden	Last hidden
1	-0.23	-0.31	**0.68**	0.02	-0.17	**0.71**
2	**0.76**	0.62	-0.29	**0.84**	0.74	-0.26
4	**0.59**	0.39	0.22	**0.70**	0.44	0.18
5	**0.66**	0.61	-0.05	**0.59**	0.50	-0.03
Dev mean	**0.45**	0.33	0.14	**0.54**	0.38	0.15
6	**0.83**	0.30	0.20	**0.85**	0.33	0.22
7	**0.30**	-0.08	0.15	**0.52**	0.37	0.16
8	**0.53**	0.37	0.33	**0.52**	0.38	0.32
9	**0.50**	0.06	0.29	**0.75**	0.34	0.29
Test mean	**0.54**	0.16	0.24	**0.66**	0.36	0.25

We observe that the input space offers the best performance by far. Furthermore, it is clear that the performance decreases from the first hidden layer to the last. This is to be expected, as the fragmentation greatly decreases as depth increases (see Appendix B.1). This implies that the measure likely becomes less discriminative, and therefore less useful, when measured at deeper points in the network.

Peculiarly, we find that Task 1 is an exception to the above, where the last hidden layer provides much better performance than the input space.

B.4 Fragmentation is not related to large weight norms

Large weight norms are known to correlate with poor generalization, with weight norm regularization a standard element of the network optimization toolkit. We therefore ask whether fragmentation is not simply a side effect of large weight norms and evaluate parameter size for the same set of models described in Section A.2. Specifically, we measure both the Frobenius norm and the mean absolute value (mean L1 norm) of the tensors of parameters per layer, as representational capacity (k) is increased. Using the Frobenius norm is more typical in this setting as it is indicative of the effect the weights has on the signal, but as this metric is sensitive to the difference in number of parameters per layer, we also report a mean value.

Results are shown in Figure 6, with convolutional layer weights and linear layer weights displayed separately. Lines and shaded regions indicate mean and standard deviations over seeds, respectively. Dashed lines indicate the point of maximum fragmentation for the clean (green) and corrupted (red) models, respectively. (Compare Figure 2.)

We find that the Frobenius norms of the weights themselves, as well as the mean absolute values of the linear layer weights display highly regular behaviour as capacity is increased, with no link to points of fragmentation. On the other hand, for the *two shallowest* convolutional layers, the mean absolute values of the weights do show an increase around the area of fragmentation. This occurs for both the clean and label-corrupted models, even though fragmentation occurs less in the first set than the second set. We therefore conclude that while there is a potential link between the mean

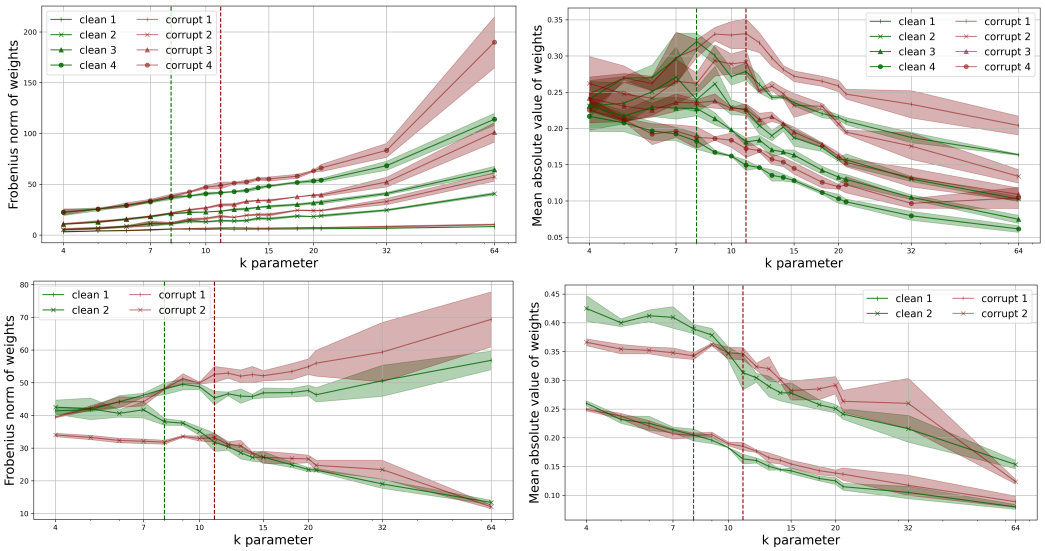

Figure 6: Parameter size per layer as model representational capacity is increased. Left: Frobenius norm. Right: Mean absolute value. Top: Convolutional layer weights. Bottom: Linear layer weights. Vertical lines indicate point of maximum fragmentation.

absolute weights of the shallower convolutional layers and fragmentation, this alone does not explain the phenomenon.

We also investigate the relationship between fragmentation and the distance a parameter moved during training (not shown here). Specifically, we measure the Frobenius norm of the difference between the parameter tensors before and after training. The parameter distance from initialization has been previously explored as a complexity measure [19, 13] and also served as a baseline in the PGDL challenge. We find that the distance a parameter is moved is highly correlated with the parameter norm itself: the precise value at initialization is overshadowed by the size of the parameter after training, and the difference between parameter norm and parameter movement as metric is negligible.