Arsen Mikovic
August 2025

**Abstract**

This note provides an informal discussion and critique of Kumar [1]. I explore how factors such as weight regularization and model size influence generalization. The observations presented aim to highlight nuances in SGD-trained networks and suggest ways to encourage less entangled, more interpretable and efficient internal representations. This discussion is informal and exploratory- the same as the original paper.

# Generalization/Grokking

In every model, there are numerous factors that influence generalization. In addition to designing a sufficiently complex architecture, regularization of the weights, the size of the model, and the size of the training data set are the most important factors for generalization. To understand this, we need to look through the lens of efficiency of an underlying circuit.

**A circuit is more efficient if it requires a lower parameter norm to produce a given logit value (Varma [3]).**

Hence, in order to differentiate between models with higher and lower norm of weights, and to train towards the latter, we need weight decay/regularization.

Having a large enough model and training data set is equally important. If the model is too small, it will go into a confusion circuit (a mode where both train and test data achieve poor accuracy) because it does not have the required complexity to generalize or memorize. If the model is of moderate size, the memorization circuit and the generalization circuit might be equally efficient: there is not enough training data, and so the weight cost of memorization and generalization is comparable. However, if the model and training dataset are large, it is significantly more costly to memorize compared to generalization/true understanding. Varma [3] also found that by further increasing either the model size or the training data set, the "weight cost" does not increase and the accuracy of the model stays high- implying that the network tends to find the most efficient circuit that is resilient to scaling, implying true generalization.

For an example of this, you can look at [4] where the emergence of generalization in any modular arithmetic task coincides with the 2D PCA of embeddings aligning on a circle in the appropriate order. I also did an experiment where the input to the model were pairs of numbers that corresponded to unique points on the grid, and where I trained a classification task to predict the gradient. In this setting, the generalization (100% accuracy) coincided with the 2D PCA of token embeddings aligning in a regular grid shape.

**All in all, the idea is: the model can only generalize well and make an effective internal representation if it has weight regularization and is large enough so that memorization is unfeasible and less efficient than generalizing.**

Additional approaches that explain grokking focus on other features, the most meaningful being the train/test loss curves, and the complexity of the model through the emergence of robust partitions in the feature space. The first one focuses on the rapid drop in the test loss with continuous training -even though the train loss does not change, the test loss suddenly drops and test accuracy goes to 100% (Power [5]). Second approach by Humayun [2] shows that Grokking occurs due to the emergence of a robust input space partition by a DNN that first overfits around the training

data and with continuous training the nonlinearities migrate towards the decision boundary- the dynamics that drives the complexity down. Both of these again demonstrate that the network first finds the inefficient memory circuit and then, by weight regularization, transitions into the efficient/generalization circuit.

## Akash Kumars paper [1]

Kumars paper on representational optimism discusses the difference between the fractured entangled representation (FER) and unified factored representation (UFR), and how training with Stochastic Gradient Descent (SGD) does not always reach the optimal and modular internal representation. In simplest words, he describes FER as the internal representation of the model that performs the task at hand well (e.g. in classification it tests with high accuracy), but internally does not form a generalizing circuit. Internally, it mixes many different circuits that do not solve the separate subtasks, but form "blobs" that by themselves have no meaning to us that combined end up solving the problem- hence the word entangled. On the other hand, UFR is a well-factored modular representation that resembles the decomposition of the desired behavior.

The paper presents this idea by comparing the different performance and robustness of the Picbreeder network and an SGD trained network of the same order of complexity. For clarity, Picbreeder is a video game where using NEAT to evolve a Compositional Pattern Producing Network (CPPN), users can guide the creation of the underlying network that generates the image they want. Hence, both models work with pixel positions as inputs, and the target is an image generated by the Picbreeder- in our case an image of a skull. In order to compare the two, Kumar creates a layer-by-layer fully connected CPPN of the same depth and with the same type and number of activation functions available-training it with SGD. The two networks output images that look exactly the same to the human eye. However, when looking at the internal representations we see a vastly different picture-the Picbreeder network uses fewer nodes and at every increasing layer the images increasingly resemble the features of the final image. The paper emphasizes the occurrence of symmetry early on as an example of feature development. However, the SGD-trained network uses all nodes, and the intermediate steps do not appear to resemble the features of the final image (Figure 1. and Figure 2.). Moreover, when sweeping through weights, he shows that the Picbreeder features change categorically-mouth, eyes, and face widens or shrinks. On the other hand, in the SGD case we cannot see any categorical changes-the whole image is distorted. In order not to do bias testing on sweeping through individual weights, Kamur also did a sweep through the value of a column of the weight matrix in the direction of a random vector. These weight sweeps are agnostic to any orthonormal transformation of the weight matrix. This further confirms his claim about Picbreeder superiority- for proof you can see Figures in the appendix.

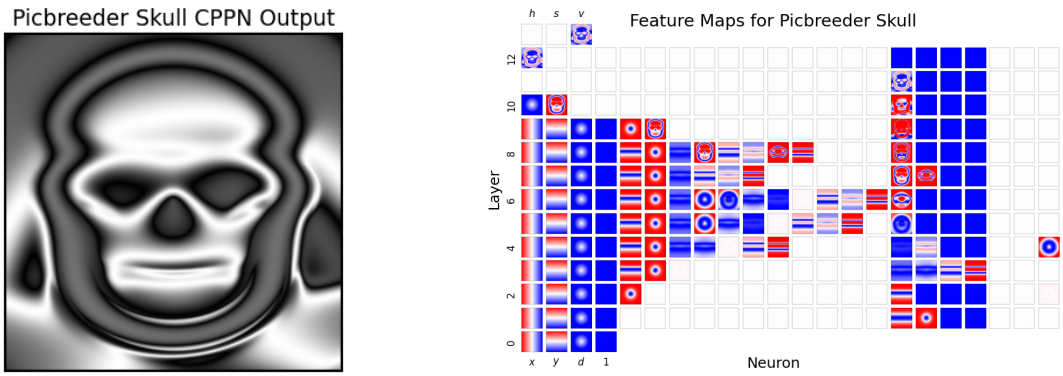



Figure 1: Picbreeder network

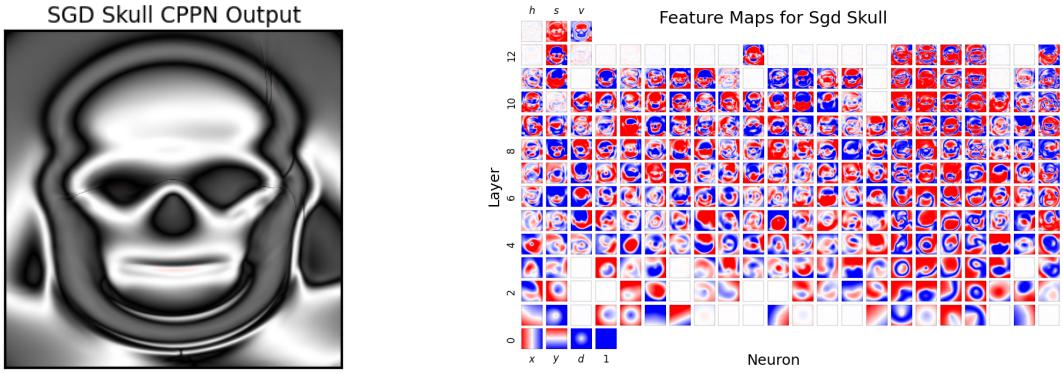



Figure 2: SGD network

## My argument

I want to argue that the SGD model was not given a fair chance. First of all, expecting an optimal solution with high efficiency is unrealistic without weight regularization. Second, since there is only one target image and the test data set is fixed, we can only force generalization by increasing the model size- and hope to find an internal representation of comparable size to the Picbreeder UFR forced by weight cost due to the scale of the model. Thirdly, among all the representations that reach some level of generalization/understanding, it is unrealistic to expect that SGD will find the exact feature basis that are interpretable by man even though the Picbreeder is reachable from the MLP architecture. Fourth, the graph-like CPPN network in the Picbreeder was given human-guided bias and advantage, whereas the MLP is fully connected and has to "figure out" which weights/neurons/connections are useless.

## Evidence

Looking at Figure 3, we can see that just by adding weight regularization of 0.001 and without any model scaling, we can already get a less entangled representation that uses fewer neurons. Going further, I have done experiments with scaling the model size in order to jump start model generalization. You can see in Figure 4 (for better resolution look at the appendix) the Feature Maps of models with increasing size. Interestingly, the number of neurons decreases with increasing size- from the original SGD $\approx 235$ neurons, just by adding weight regularization the number of neurons used goes down to $\approx 185$, and by doubling the width of the network it goes down to $\approx 130$ neurons, which is the approximate number of neurons used in the original Picbreeder- this is just more evidence of the arguments in Section One.

Looking at the network of four times the size of Picbreeder, we can see that all of the Gaussian neurons are being used and are preferred to the activation function types (likely due to gaussian nodes adding in most complexity compared to linear and sin nodes). Hence, I decided to test the model with four times the width, but with a reduced number of Gaussian nods to the number in the original Picbreeder. The resulting Feature Map has approximately the same number of neurons used as the original Picbreeder ($\approx 130$). Furthermore, by looking at the Feature Map more in depth, you can see the formation of meaningful skull-face features around layer 6/7 (same as in Picbreeder) and you can also see that there are not any of the 'entangled' overly complicated features forming, instead, more 'blob'-like features that showcase symmetry similar to the ones in the Picbreeder.

Sweeping through the weights also improves, as you can see in the figures in the appendix. Especially, looking at Figure 7 gives a good comparison to the original Picbreeder, since they are the same size. For other figures you can see that there are a lot of outputs that are unaffected by weight sweeps in the larger models. This is because a lot of neurons are in the "unused" state, hence sweeping them does not affect the image much. However, looking at the ones that do change, you can see that the images do not get completely distorted and are more prone to categorical changes- for each case where weight regularization was applied.

Also, less formal but worth mentioning, looking at the train loss curve we can see the expected 'grokking' like sudden drop where the network potentially transitions from a local minimum that memorizes to global minimum with more efficient generalization circuit- figure in the Appendix. Obviously, there is no test data and we can only look at the train loss, but it still follows the overall sentiment of grokking.

An additional task that can be done is to reduce the network by removing unused weights and neurons, and then perform PCA on the feature maps to find the most significant feature directions. Alternatively, but in the same realm, we could perform SVD dimension reduction on the weights of the scaled models and test if that will reproduce even more human-meaningful features.

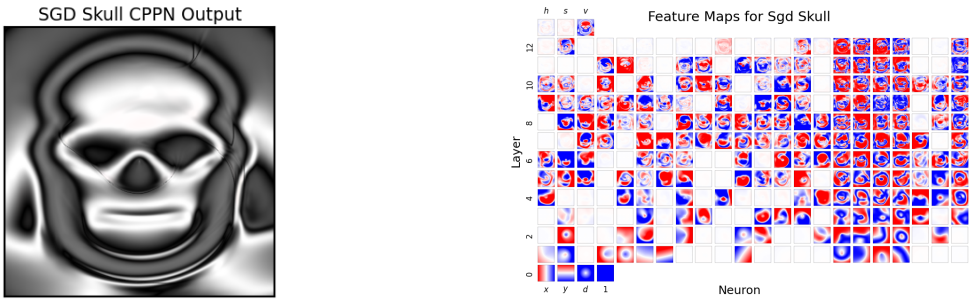
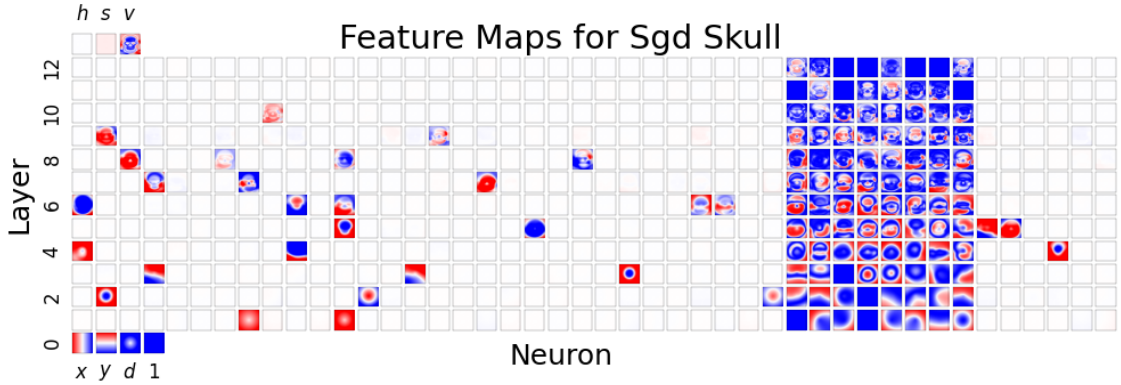
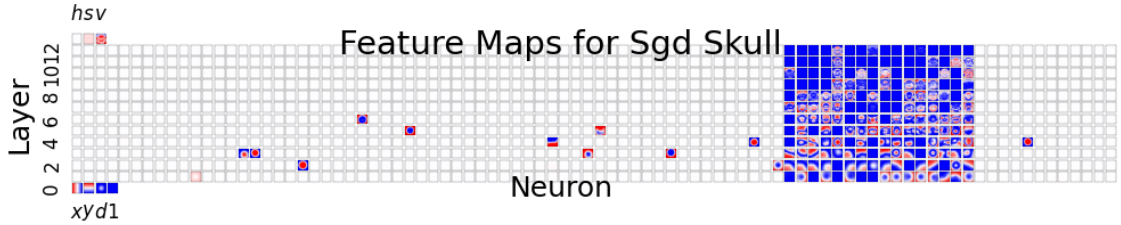


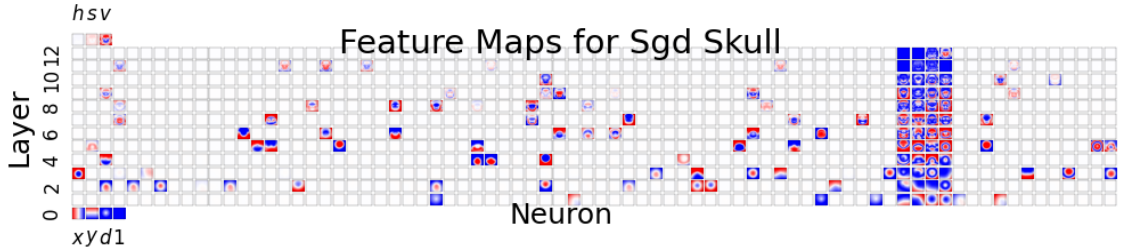

Figure 3: SGD network with same size and trained with AdamW




Network of double the Picbreeder size




Network of four times the Picbreeder size




Gaussian adjusted network- same number of Gaussian neurons as the Picbreeder

Figure 4: Results due to scaling the network size in order to induce the generalization efficient circuit

## Conclusion

In summary, while Kumars paper raises valid skepticism about SGD and the emergence of entangled representations, my experiments demonstrate that with appropriate weight regularization and model scaling, it is indeed possible for SGD-trained networks to develop more interpretable and efficient internal representations. This suggests that the limitations highlighted in the paper are not intrinsic to SGD, but rather depend on the training setup and model capacity.

# References


[1] A. Kumar, J. Clune, J. Lehman, and K. O. Stanely, *Questioning Representational Optimism in Deep Learning: The Fractured Entangled Representation Hypothesis*,

[2] A. I. Humayun, R. Balestriero, and R. Baraniuk, *Deep Networks Always Grok and Here is Why*,

[3] V. Varma, R. Shah, Z. Kenton, J. Kramar, and R. Kumar, *Explaining grokking through circuit efficiency*,

[4] Z. Liu, O. Kitouni, N. Nolte, E. J. Michaud, M. Tegmark, and M. Williams, *Towards Understanding Grokking: An Effective Theory of Representation Learning*,

[5] A. Power, Y. Burda, H. Edwards, and I. Babuschkin, *GROKKING: GENERALIZATION BEYOND OVERFIT TING ON SMALL ALGORITHMIC DATASETS*,

# Figure Appendix

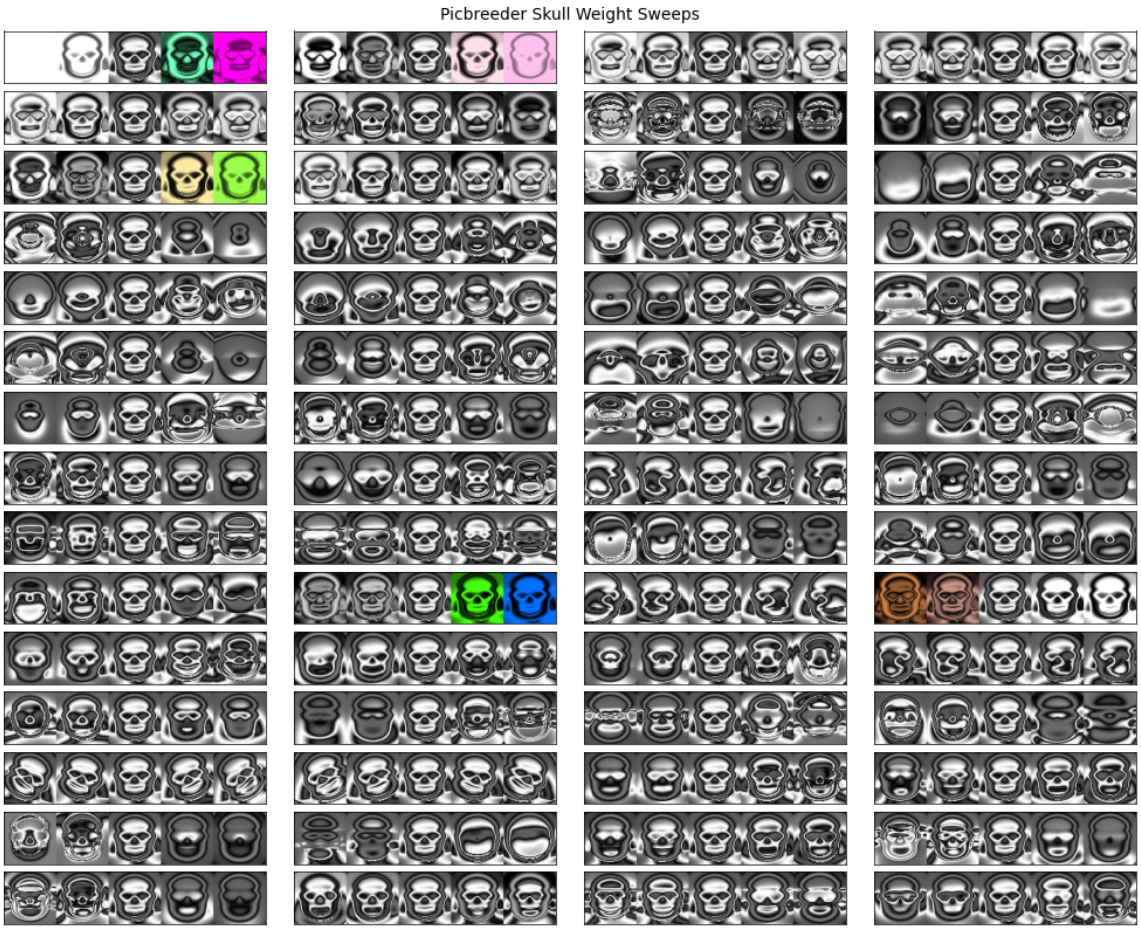



Figure 5: Weight sweep along a random vector for the Picbreeder

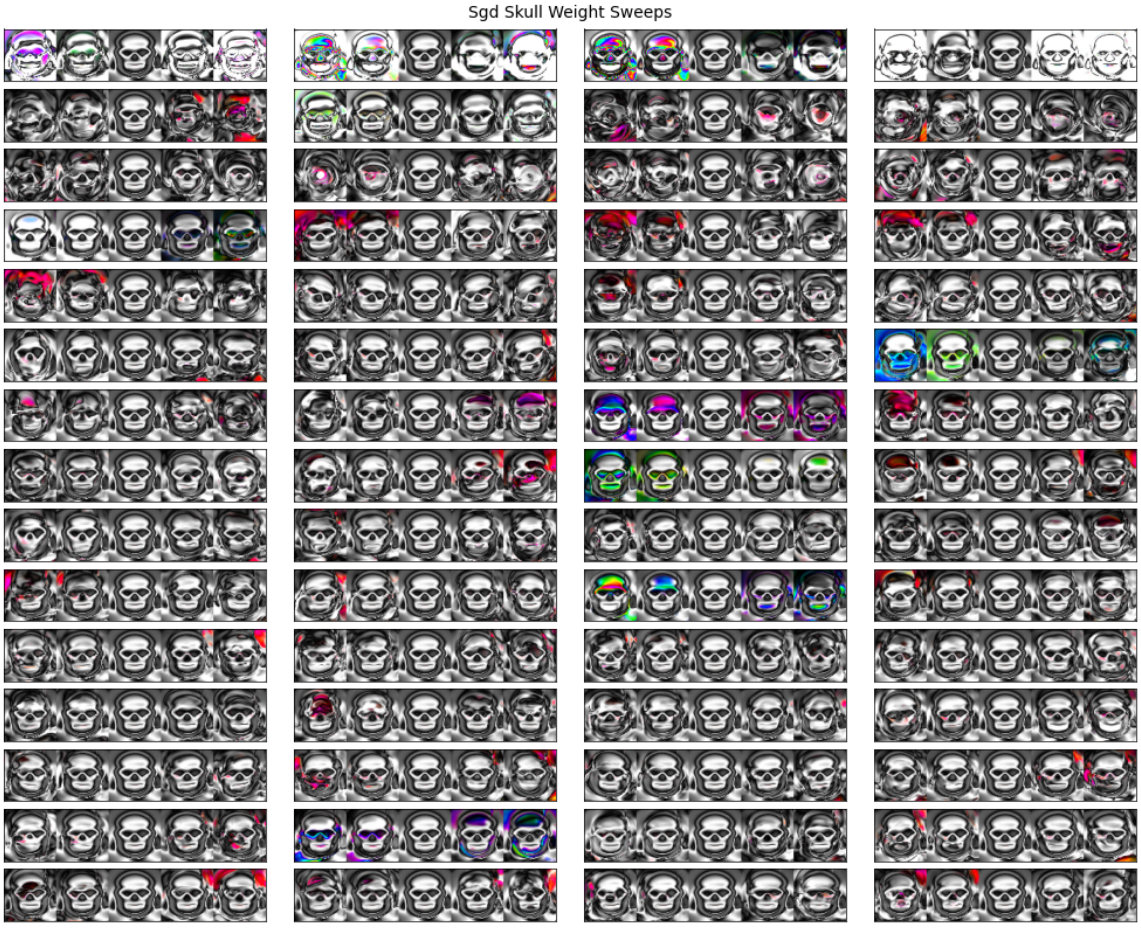



Figure 6: SGD trained network with no regularization, weight sweep along a random vector

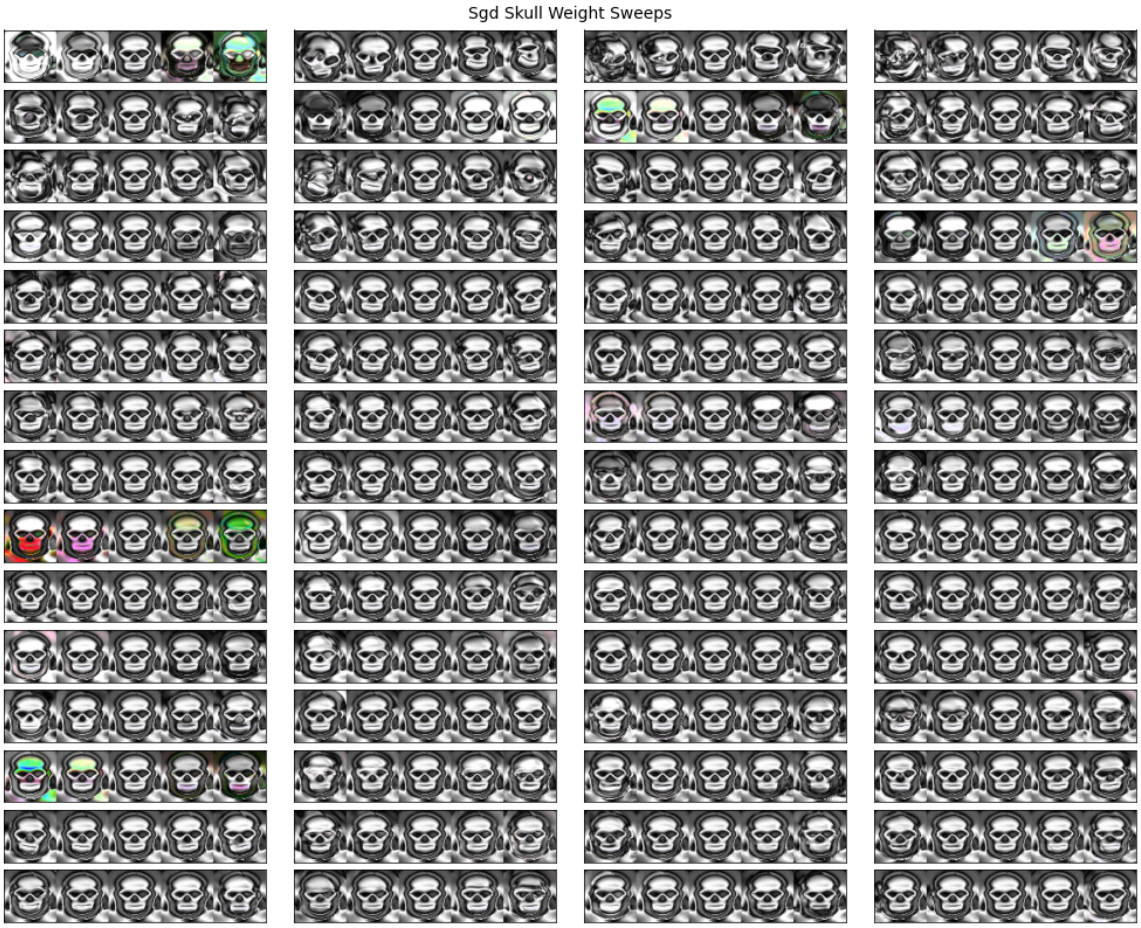



Figure 7: SGD trained network with weight regularization, weight sweep along a random vector, same size as the original Picbreeder

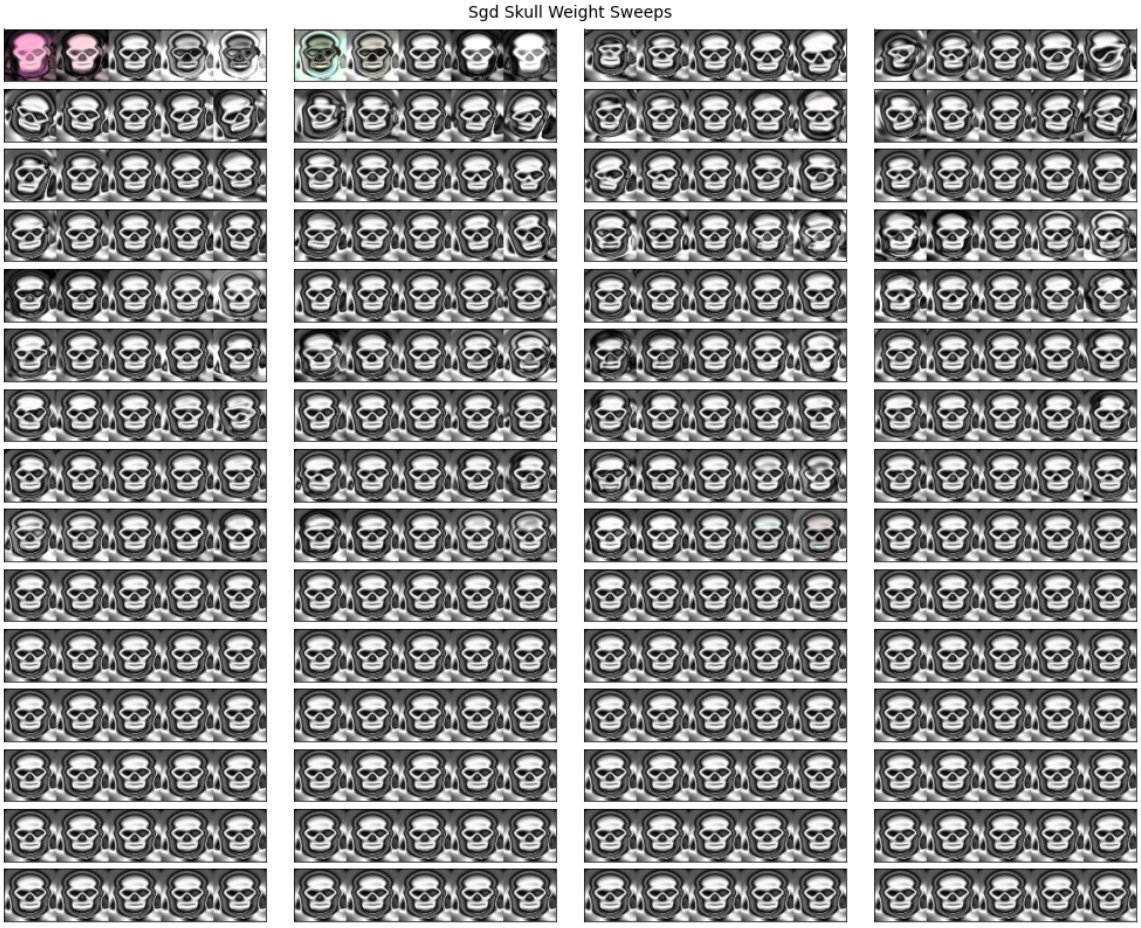



Figure 8: SGD trained network with weight regularization, weight sweep along a random vector


Figure 9: SGD train loss for the model of double Picbreeder size with weight regularization $1e-4$

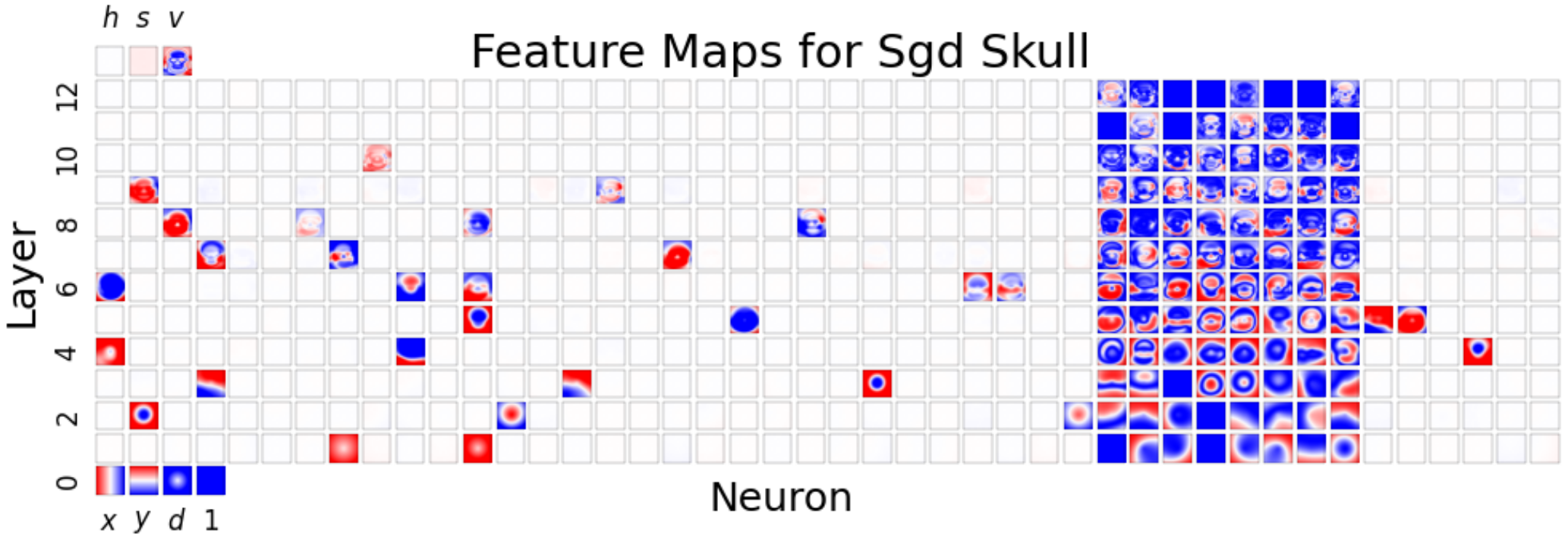



Figure 10: Network of double the Picbreeder size

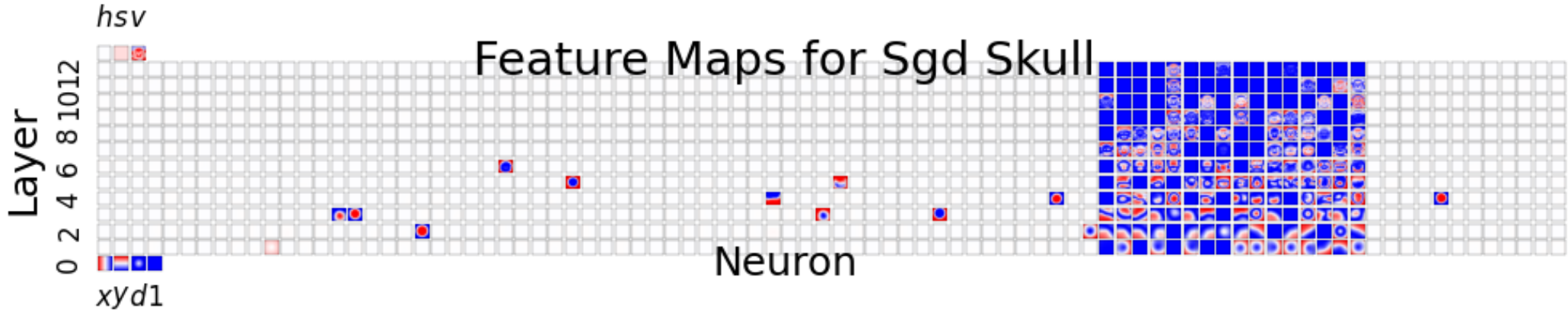



Figure 11: Network of four times the Picbreeder size

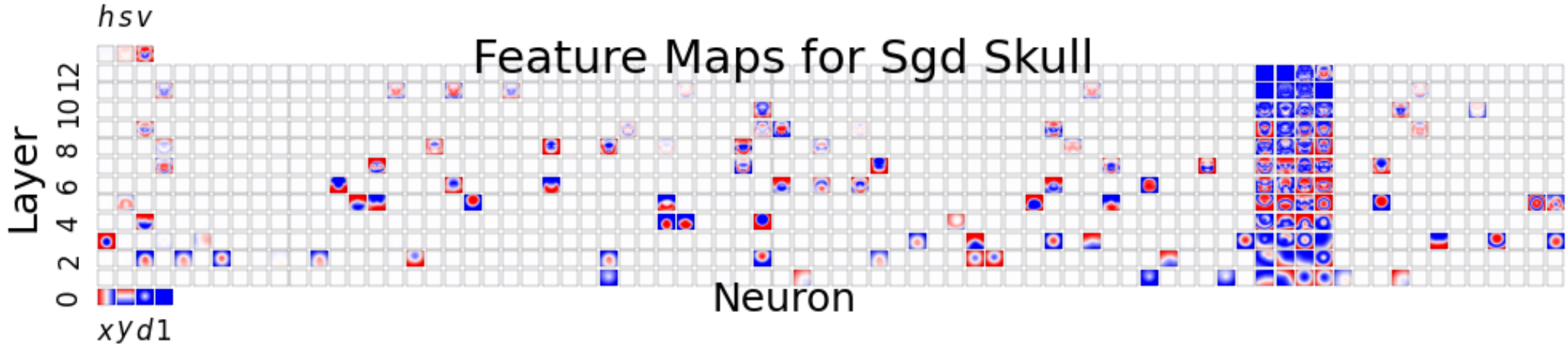



Figure 12: Gaussian adjusted network- same number of Gaussian neurons as the Picbreeder