# The Role of Task Space Correlations in Catastrophic Interference

Introductory Computational Neuroscience / Neuro 120 Final Project

**Stephanie Campbell, Alexandra Kieras, Jordan Kruger, Joanna Tao**

May 8, 2018 **Code:**
https://github.com/stephyc/catastrophic-interference

# Introduction

## Sequential Learning

Humans possess the distinct ability to learn new and diverse tasks in a continual and sequential manner. Biological brains, specifically human brains, are very good at what is called "continual learning" or sequential learning. Humans are able to learn multiple tasks in succession over a lifespan without having to learn them all at the same time. In general, humans, as well as animals, do not seem to forget previous tasks because they learn a new one. As we seek to build biologically inspired and accurate models and networks, it is important to consider current work that aims to determine the neural mechanisms that allow the mammalian brain to learn so well across different tasks in complex and extended sequences.

## Biological Studies

At Oxford, Timo Flesch and his colleagues studied how the human brain can learn to perform many different tasks over the lifespan without mutual interference. Their study showed that human participants seemed to benefit from training on temporally autocorrelated task one at a time (referred to as "focused training") and subsequently were able to more successfully perform on new tasks. Furthermore, these subjects showed better performance on later tests that were made up of randomly interleaved tasks. When the researchers analyzed error patterns, they found this "focused learning" allowed for factorized task representations that were not affected by mutual interference. However, "continual learning" has still not been able to be translated in machine learning, where network models seem to suffer interference between tasks. [2]

At Stanford, James L McLelland and his colleagues ran biological experiments to investigate how long-term memory mechanisms store and constantly update (via synaptic update) both recent and old tasks. Their work explored how learning and new knowledge is gradually incorporated in the neocortex's representational systems. Their findings suggested that memories are stored via synaptic changes in the hippocampal system and consistent updating occurs with neocortical synapses to reinstate recent memories. Their conclusion stated that the neocortex slowly learns the structure of experiences and then the hippocampal system slowly allows for the learning of new tasks without interference with the neocortical structure. This indicates that tasks learned in an interleaved manner allows for the integration of multiple tasks and in turn, leads to better performance across these learned tasks. A primary driver of this research was an attempt to understand why machine learning systems experience gradual structural changes as new experiences are learned. When trained on several tasks concurrently, or when the process of training a second or third task does not adjust the weights of the original model, networks are able to perform relatively well across tasks. So why do networks fail where the biological brain succeeds in learning sequentially? [6]

## Catastrophic Interference

When an artificial neural network is trained sequentially on multiple datasets or tasks, learning the new task often causes the neural network to "forget" how to perform the previously learned task. This is known as "catastrophic interference" or "catastrophic forgetting", which was first described by Mc-

Closkey and Cohen, who found that whenever a network is trained sequentially (i.e., first on one task and then on a second, etc.), the training of subsequent tasks is able to adjust the networks weights, at least some amount of interference occurs [7].

Interference occurs due to the structure of the neural networks: Inputs are fed into several interconnected layers of "neurons", which produce some output based on the weights of the neurons. During the training of a neural network, the weights are updated through back-propagation to produce better or more accurate outputs. For example, a model is trained on an initial tasks and its weights are fine tuned to produce the best outputs for the input data. When a new second task is subsequently learned after the first training, the neuron weights are re-adjusted to produce the best outputs for the second task. Thus, the weights used for the first task are overwritten and "forgotten". However, if two tasks are learned simultaneously, then the weights are optimized for both tasks at the same time. In this case, a single set of weights is used for both tasks, rather than a new set of weights irreversibly replacing the first set of weights. Neural networks are incredibly useful tools, and capable of producing human-like behavior, or even surpassing human performance in tasks such as playing chess or Go.

Ideally, any type of interference in the performance of these neural networks across interleaved learning can be reduced or completely fixed. Some approaches have included methods such as freezing the weights of certain neurons important to the initial task, or adding parameters to encourage weights in the new task to be similar to weights in the old task. However, recent literature by Kirkpatrick et al. and Zenke et al. [4] [12], both published in 2017, use a third approach of penalizing changes in weights that are important in the old task. All three of these approaches attempt to find a solution that satisfies the multiple tasks being learned, though some are computationally more expensive than others.

Such a solution is demonstrated to exist by using a geometric representation of the solution space (see Fig. 1). A neural network will generally consist of multiple layers, each of which has many neurons. With a high number of parameters, multiple distinct sets of weights can perform the same task with an equally high level of accuracy. These sets of weights can be visualized as points in a high dimensional space. Thus, it is highly likely that there exists is a solution space of weights that performs both accurately for task A and for task B. When the neural network is initially trained to perform task A accurately, its weights will be a point in the task A solution space. "Catastrophic interference" occurs when the updated weight set determines a point found in only the task B solution space. To properly avoid "catastrophic interference", the weights should determine a point in the task spaces of both A and B. Thus, algorithms tightly restrict the changes of weights that are important for the performance of task A, while allowing weights that do not greatly affect the performance of task A to change more. This allows the point to shift toward the solution space of task B, while not leaving the solution space of task A, so the neural network retains its performance for task A, while also learning task B.

Kirkpatrick et al. proposed a method called elastic weight consolidation, implemented as a quadratic penalty for changing the weight of a particular neuron, with greater penalties for changing weights of

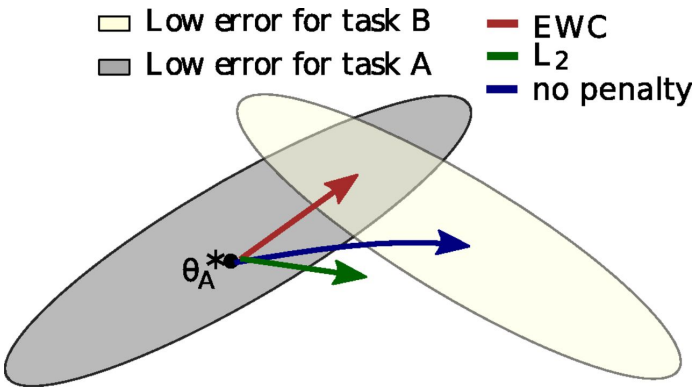


Figure 1: Geometric representation of the solution space. When the neural network is trained to perform task A accurately, its weights determine a point in the task A solution space ($\theta^*_A$). Catastrophic interference occurs when the neural network is re-trained on task B so the updated weight set is found in only the task B solution space (blue arrow). This can be prevented by restricting the amount that the weights can change by, so that it does not move very far out of the solution space. If the weights are too restricted, then task B will not be learned (green arrow). To avoid catastrophic interference but still learn task B, the weights should shift to determine a point in the task spaces of both A and B, following the red arrow. [4]

greater importance to the performance of the first task. The importance of the weights are approximated using a Fischer matrix to predict the probability of a given weight being important. The paper then demonstrates how well the use of elastic weight consolidation reduces the effects of catastrophic interference by testing on the identification of random patterns, identification of manipulated MNIST (handwritten number) images, and the playing of Atari games, in both supervised and reinforcement learning contexts[4]. Zenke et al. criticized elastic weight consolidation as limited to only low-dimensional output spaces due to the high computational overhead of the computation of the diagonal of the Fischer matrix[12]. While using a similar approach to Kirkpatrick et al., Zenke et al. developed a class of algorithms that calculates an importance measure for each weight, calculated locally for each neuron. In practice, this parameter is calculated using gradient descent over the accuracy for the second test and the change in the neurons weight, and is calculated continuously throughout training. This method was also tested on manipulated MNIST images, as well as the CIFAR-10 and CIFAR-100 databases (color image databases for object identification). Here, we attempt to uncover understanding of the dynamics of how and why catastrophic interference occurs in certain task spaces, by analyzing differences between various manipulated images and their effect on catastrophic interference. In addition to this, we aim to avoid catastrophic interference in sequential learning of these different tasks in an artificial neural network, by implementing a similar approach to that presented by the Kirkpatrick and Zenke papers, primarily referencing the code provided by Zenke.

# Materials and Methods

## Software and Datasets

We constructed our neural networks in Python, version 3.5.2, as the available deep learning frameworks are more commonly used, easier to learn, and overall more powerful for our analysis than many frameworks in competing languages. Of the available deep learning packages, we chose Tensorflow due to our personal previous experience with it, its popularity in the deep learning research community, and its computational speed. Its primary competitor, PyTorch, is only marginally more intuitive to learn and lacks the popularity and speed of Tensorflow. Zenke et al. also used Tensorflow to produce the results for their 2017 paper, demonstrating that solutions to catastrophic interference could be produced with that library. Based on the advice of the paper, we used version 1.2.1. We also used Keras, version 2.0.5, a high-level neural network API that runs on top of Tensorflow. Keras is very user-friendly, and helped us not only decrease the time it took to run our scripts, but also allowed us to debug them quickly. Though the versions of these libraries we used are deprecated, using these older versions allowed us to modify default settings in a way that is prevented by newer versions od the libraries. Finally, we also used Jupyter to run and debug portions of the code at a time.

We used the MNIST dataset to train and test our methods, as it was used by both Zenke and Kirkpatrick and is a typical dataset used for basic neural network testing. Furthermore, it was relatively easy to manipulate the MNIST data set to produce task mutations, and the relative simplicity of the images made it easier to analyze correlation between tasks. MNIST is a subset of the larger NIST database of handwritten numerical digits, and contains 60,000 images meant to be used for training neural networks, with 10,000 images used for testing, as well as labels for each image (i.e., the correct identification for each image). Each original MNIST image is a 28x28 image of a handwritten digit, in black and gray on a white background. We performed a variety of manipulations on the MNIST dataset to produce distinct tasks to test catastrophic interference. The first two manipulations used are a simple color inversion, and a color inversion in a checkerboard pattern (i.e., every other pixel was color inverted). We then performed reflections, both horizontally and vertically, of the color inverted image. The next manipulations performed are a 90 degree rotation, also of the color inverted image, and a color inversion on only half the image. The last manipulation is a color inverted swap: The color inverted image can be thought of as cut in half horizontally, then the top half of the image switches positions with the bottom half. Throughout this report, these will be coded as "Inv/Inversion", "Checkerboard", "FlipUD/Flip Up-Down", "FlipLR/Flip Left-Right", "Rot90/Rotate 90", "Invbot/Inverted Bottom", and "CutUD/Swap". See Fig. 2.

Although humans easily recognize a rotated or mirrored digit, we assume that since neural networks see "pictures" as lists of gray values, even rotating or mirroring the images will change them enough that the network recognizes them as new images, and identifying them is a new task. We later examine the validity of this assumption by evaluating various measures of distance between images, and analyzing the amount of catastrophic interference changes when the neural network is trained on tasks of different distances apart.

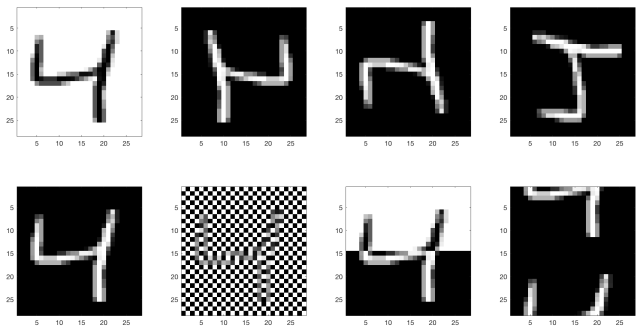

Figure 2: Sample MNIST image and manipulations (handwritten 4). Top row: Original, Flip Left-Right, Flip Up-Down, Rotate 90. Bottom row: Inversion, Checkerboard, Inverted Bottom, Swap. Each image is 28x28 pixels.

### Network Architecture

We used a multi-layer perceptron (MLP) with 256 units with ReLU nonlinearities with 3000 hidden neurons. To reduce training time, we initially used only 5 epochs; however, we found that this affected the neural networks accuracy too significantly, and increased the number of epochs.

### Isolated Learning

First, we ran our model by training a network for the original and each of the 7 modified datasets in an isolated way. Each network used the same model architecture and training conditions. We collected the accuracy values in order to have benchmarks for future sequential scores.

### Sequential Learning

Second, we sequentially trained this model on the set of different image mutations to understand how catastrophic interference in sequential learning happens during on sequential tasks. For each new task, we observed some a dropoff in performance on previously-trained tasks. A single evaluation after training on all tasks would not capture the nature of this forgetting - in reality, we observed a steady drop off on tasks rather than an immediate reduction in performance. We utilized a small number of epochs in this stage, which was enough to effectively visualize sequential learning and the results of catastrophic interference. We then visualized the weight space at different stages of training and observed dynamics on how weights update. We expected accuracy to change depending on weight updates and similarity of images. If task spaces were similar, we expected to see a less drastic change in the weight dynamics as weight values were consolidated across training. We will discuss our data and observations in the next section.

## Results And Discussion

### Task Space

Catastrophic interference is a direct byproduct of movement in the weight space between multiple tasks during sequential training. While it is difficult to examine weight space

in detail because of its high-dimensional nature, we hypothesized that observations in the task space in the form of quantified correlation values can give insight into whether the effects of catastrophic interference will be aggravated or dampened. Since our network classifies based on pixel values, highly correlated images that have common pixel features will then be represented more similarly in weight space and therefore more robust against the effects of forgetting when trained sequentially. Similarly, highly anti-correlated images will force the network to make more drastic movements in weight space meaning that the effects of forgetting will be more pronounced.

In order to define similarity in task space we chose to use the mean image for each digit (0-9) for each manipulation as a foundation for quantifying correlations within manipulations and across all digits (See Figure 3). The mean image for each digit shows that aggregate of all variations of that particular digit within the MNIST sample data. This mean image provides a generalized basis for determining how similar a particular individually styled sample is to the mean of all samples. This indicates whether classifying a particular image will require similar weights to classifications for others (highly correlated) or if it will take drastically different weights to classify correctly (highly anti-correlated). With intuition as to which tasks are most related we are able to meaningfully train groups of tasks sequentially in order to track forgetting throughout several iterations of training.

We first looked at the correlations between the mean image for particular digits (0-9) across all manipulations to the MNIST dataset (See Figure 4). We find that certain manipulations, primarily Fliplr, Flipud, Rot90, and Inverse, are highly correlated to one another. These particular manipulations also happen to be consistently anti-correlated with the mean image from the original MNIST data. We predict that when trained with sequentially with the original classification task, these four images accuracy will degrade. Conversely when trained sequentially together, their similarity based on correlation value should correspond to less forgetting and more cross-over in weight space.

## Training Results

We first established a baseline for comparison, training the neural network, then testing, for each manipulated MNIST dataset (Figure 5). No catastrophic interference was modeled yet; we simply verified the performance of the neural network on the datasets.

We subsequently tested a training scheme solution (Figure 6). We initially started by training iteratively on the original and rotated datasets. We trained first on the original and then on the rotation, saving the weights each time, for a series of five iterations. On the two tasks, we noticed significant improvement in classification of both datasets with multiple iterations.

Moving forward, we implemented the same iterative training scheme on three datasets. With three tasks, the iterative scheme effectively broke down and we no longer saw significant improvement in task performance (Figure 7). In particular, we noticed that when we selected an image that was anticorrelated with the other two, task performance on the anticorrelated image stayed extremely low, even after retraining, while performance on the correlated image tended to oscillate. Furthermore, we saw performance on the correlated images decreasing slightly when trained after each other, and

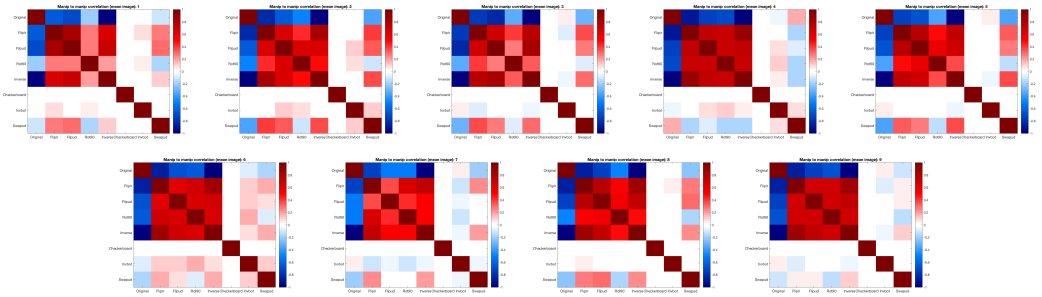

Figure 3: Digit by digit correlations between datasets. Each row and column of each correlation matrix is a manipulated MNIST datasets. Top Row: Digits 1-5, Bottom Row: Digits 6-9

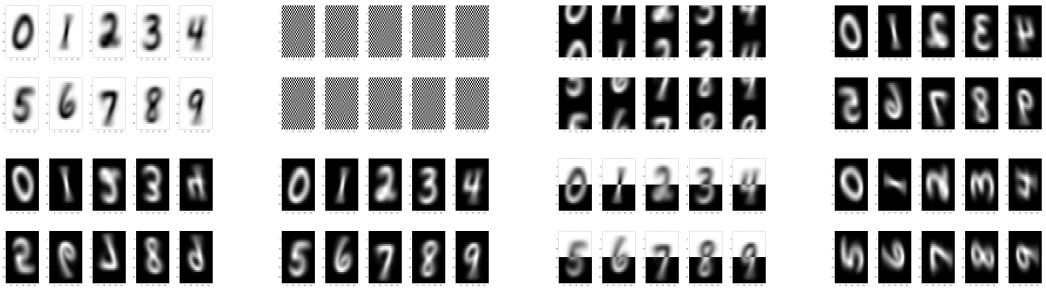

Figure 4: Mean image for each digit and manipulation. Mean images were calculated by taking sample images from each variation within the MNIST dataset and calculating the mean image for each digit based on shared pixels. See produced mean images for digits 0-9 above.

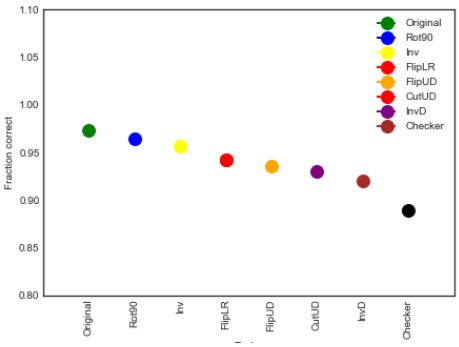


Figure 5: Benchmarking Tests: Accuracy scores for all 7 datasets trained in an isolated way with the same parameters.

decreasing steeply after the anticorrelated image was trained (Figure 8). When we chose a set where all tasks were anticorrelated with each other, we saw poor performance across all tasks - however, the degradation in performance was less severe after training tasks

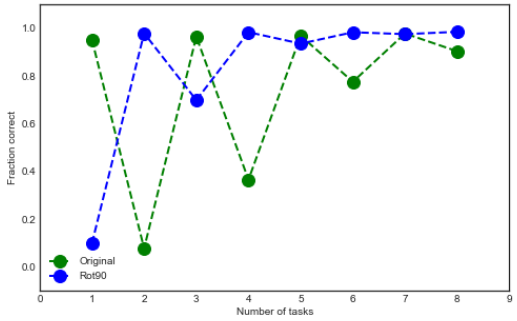


Figure 6: Two task iterative training scheme

which were weakly anticorrelated (training "Rot90" and then "CutUD") than after training tasks which were highly anticorrelated (training "Rot90" and then "Original") (Figure 9).

With these observations, we hypothesized that the correlation between task spaces would be a significant factor in the success of the synaptic architectural regime proposed by Zenke et. al at improving sequential learning performance.

We then moved to a solution implementation described by Zenke et. al in their paper "Continual Learning Through Synaptic Intelligence". The proposed solution calculates a parameter $\omega$ during gradient descent, where:

$$\omega_{ij}^{l} = \frac{\delta C}{\delta \theta_{ij}^{l}} \frac{\theta_{ij}^{l}}{\delta t}$$

Intuitively, this represents the contribution of the parameter to the overall cost function times the amount the weight changed in the resulting update - the "importance" of the weight to any given update.

From this value, the parameter $\Omega_{ij}^{l}$ is calculated:

$$\Omega_{k}^{\mu} = \sum_{v<\mu} \frac{\omega_{k}^{v}}{(\Delta_{k}^{v})^{2} + \eta}$$

That is, the per-parameter value $\Omega_{k}^{\mu}$ represents running sum of all $\omega$ calculated per backpropagation iteration, divided by the changes to that parameter plus a small constant $\eta$ introduced for smoothing.

$\Omega$ is then used as a prefactor to a regularization term (in our case, a prefactor to $L2$ regularization) in calculating the loss at each weight, modifying the overall change to that weight. Intuitively, this implies that the larger values of $\Omega$ represent a higher degree of importance of that weight to the correct classification of a given task, preventing large changes to weights that are important in correct task classification.

We then proceeded to train the tasks sequentially. While we noticed some improvement in task preservation after implementing the synaptic solution, ultimately the results were not as promising as those presented in "Continual Learning" (Figure 10). In particular, high levels of task preservation appeared between tasks that were strongly correlated with each other, while task performance tended to drop off between anticorrelated tasks. (For example, we saw notable decreases in performance on the original dataset when the 90 degree rotation was trained soon after, but the left-right and up-down flips

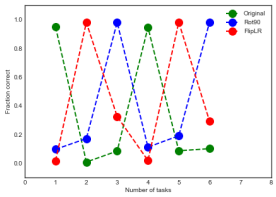


Figure 7: Iterative training scheme on three tasks

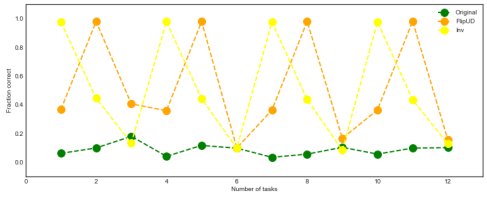


Figure 8: Iterative training scheme on two correlated and one anticorrelated task. The green bar (original) represents the task which was anticorrelated with the other two, while the 90-degree rotation and inversion were positively correlated with each other).

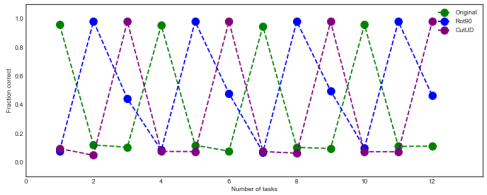


Figure 9: Iterative training scheme on three anticorrelated tasks. The green bar (original) represents the task which was highly anticorrelated with the other two, while the 90-degree rotation and up-down cut were weakly anticorrelated with each other).

tended to preserve task performance when trained sequentially.

With this in mind, we then tested on the same datasets used in "Continuous Learning" in order to see if we could more accurately replicate the findings of the paper. We first began by analyzing the correlation between tasks in the dataset used by the paper (from the skdata library, dataset *larochelle_etal_2007*). We selected five MNIST datasets published in the skdata library - an original set, a rotation, and three sets of images with injected random mutations. Significantly, the correlations between these tasks were all very strongly positive (Figures 11, 12; 14).

Though we noticed improved performance on the tasks, we also observed that more strongly correlated tasks preserved each others' performance. The rotation set, which proved to be the least strongly correlated to the others in the group, caused the classification accuracy on the other datasets to drop sharply (Figure 11). Significantly, the solution in this case did appear to be working in that performance levels in the later iterations were generally preserved or increased rather than dropping sharply as more datasets were

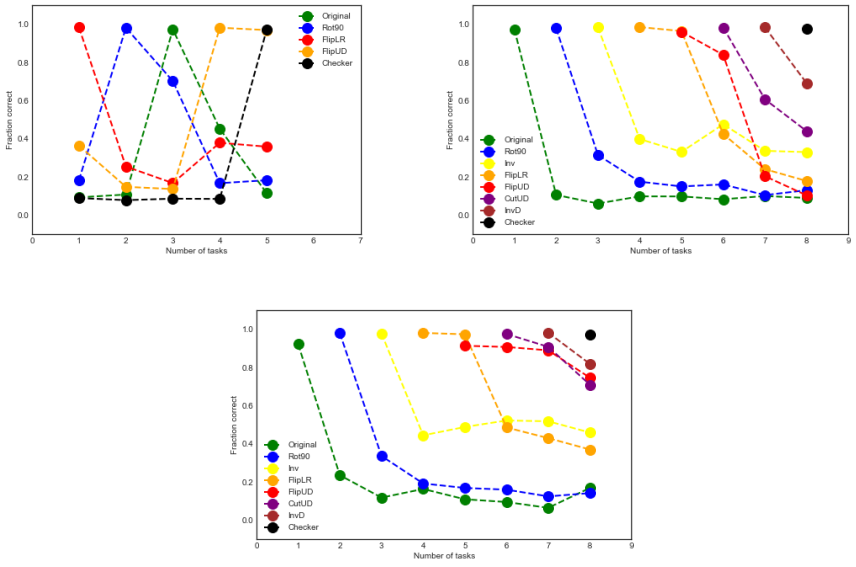


Figure 10: Solution training scheme results

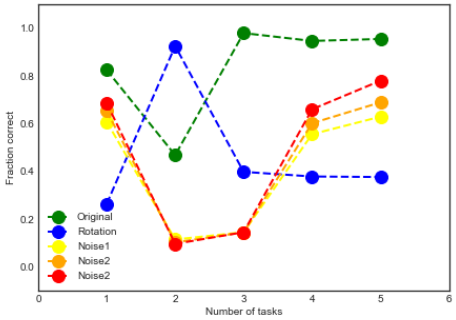


Figure 11: Skdata dataset training results

trained, as we observed in our benchmarking tests.

Furthermore, the training times between tasks which were anticorrelated took up to six times as long as training between tasks which were positively correlated (240 ms vs 30-60 ms), supporting the hypothesis that training on anticorrelated tasks results in more drastic changes to the weights and thus results in greater degrees of degradation from previous tasks. Training on a specific task also tended to increase the performance on tasks which were correlated to it. (This would make sense if weights which are significant for the correct classification of one set are similar to weights which are significant for the correct classification of another.)

We expect to see the traces of these changing dynamics in weight space when looking at glimpses of weights during training iterations. To project high-dimensional weight space into a two-dimensional visualization, we plotted heatmaps of the weight values after each training round, with each column of pixels in the heatmap representing all weights $w_{ij}^{l}$ for a given set of connections from neuron $i$ in layer $l$ to all neurons $j$ in layer $l+1$. Indeed we can see the weights strengthen, change, and adapt with given correlated tasks (Figures 16, 17, 18, Appendix). As the system evolves, weights become more strongly positive or negative, represented by brighter colors in the plotted data. We notice similar color patterns across all training iterations, likely due to the regularizer solution which penalizes extreme changes in weights; rather, weight values appear to be consolidated across tasks. However, a large portion of the network is still dark, indicating that many weight values remain close to zero as the network is constructed to have extra space to store weight values for different tasks - intuitively, the small number of tasks we trained on is not enough to allow the network to "saturate."

Though studies such as those presented in Zenke et al. and Kirkpatrick et al. attempt to solve catastrophic interference by implementing various architectural solutions that retain learned task information within the weights of a network during training on new tasks, little work has been done to inquire about the dynamics of how catastrophic interference occurs during training progression and what features of tasks aggravate its effect. We have explored the effect of various manipulations to MNIST dataset images on catastrophic interference, showing that controlling task similarity by training networks sequentially on highly correlated tasks or anti-correlated tasks can significantly affect the prediction accuracy and learning progression across a se-

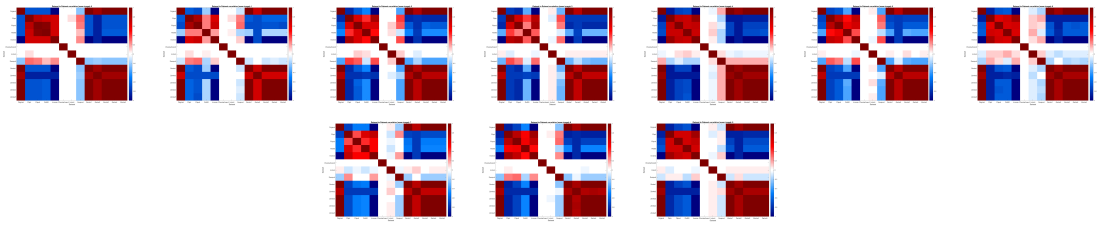

Figure 12: Digit by digit correlations between datasets. The first 8 rows and columns of each correlation matrix are the manipulated MNIST datasets we used initially, last 5 rows and columns are the manipulated MNIST datasets used by Zenke et al. Top row: Digits 0-5; Bottom row: Digits 6-9.

ries of tasks regardless of the solution schema implemented. While the solution system undoubtedly improved performance, correlation between tasks was an important factor in the success of performance.

As we explore the nature of how a network moves in parameter space throughout training, we must recognize that the tasks themselves have a large effect on the network's ability to learn sequentially and perform on various tasks. When we compared our results using manipulated MNIST images against the findings of Zenke et al. using images with injected noise, we find that our predictions, though improved, do not reach the level of performance as predictions on the noisy images. A simple explanation for this result is that correlations between the noisy images would be higher when compared to the correlations between our manipulated data. In essence, the noise caused differentiation between tasks, but did not produce a level of task diversity significant enough to dip performance greatly. This result shows that even small variations in task diversity as a result of the form of manipulation used to create a varied task set, can cause visible change to results and in fact, reveal possible fragility in synaptic solutions such as the one presented in Zenke et al. . Most importantly, this exposes the fact that current solutions would not be capable of crossing task domains. As we seek to develop biologically inspired and complex networks, crossing task space within related tasks and unrelated tasks is necessary. Further work looking in more fine detail at the aspects of tasks that make them difficult to classify or cause extreme anti-correlation with other tasks will provide the required context to develop synaptic solutions that are sensitive to fine task details across domains.

## Acknowledgements

We would like to acknowledge Friedemann Zenke for guiding us in getting his code set up on our computers. We would also like to acknowledge the teaching staff of Neuro 120: Introductory Computational Neuroscience (Kenneth Blum, Luke Rast, Andrew Saxe),and our classmates for providing valuable feedback and discussion.

# References


[1] Marcus K Benna and Stefano Fusi. Computational principles of biological memory. *arXiv preprint arXiv:1507.07580*, 2015.

[2] Timo Flesch, Jan Balaguer, Ronald Dekker, Hamed Nili, and Christopher Summerfield. Focused learning promotes continual task performance in humans. *bioRxiv*, page 247460, 2018.

[3] Stefano Fusi, Patrick J Drew, and Larry F Abbott. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, 2005.

[4] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[5] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[6] James L McClelland, Bruce L McNaughton, and Randall C O'reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[7] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press, 1989.

[8] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[9] D Wittman. Fisher matrix for beginners. Physics Department, University of California, Davis, CA 95616; dwittman@physics.ucdavis.edu, http://wittman.physics.ucdavis.edu/Fisher-matrix-guide.pdf.

[10] Makoto Yamaguchi. Reassessment of catastrophic interference. *Neuroreport*, 15(15):2423–2426, 2004.

[11] Makoto Yamaguchi. Orthogonality is not a panacea: Backpropagation and catastrophic interference. *Scandinavian journal of psychology*, 47(5):339–344, 2006.

[12] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

# Appendix

Larger figures for easier reading.

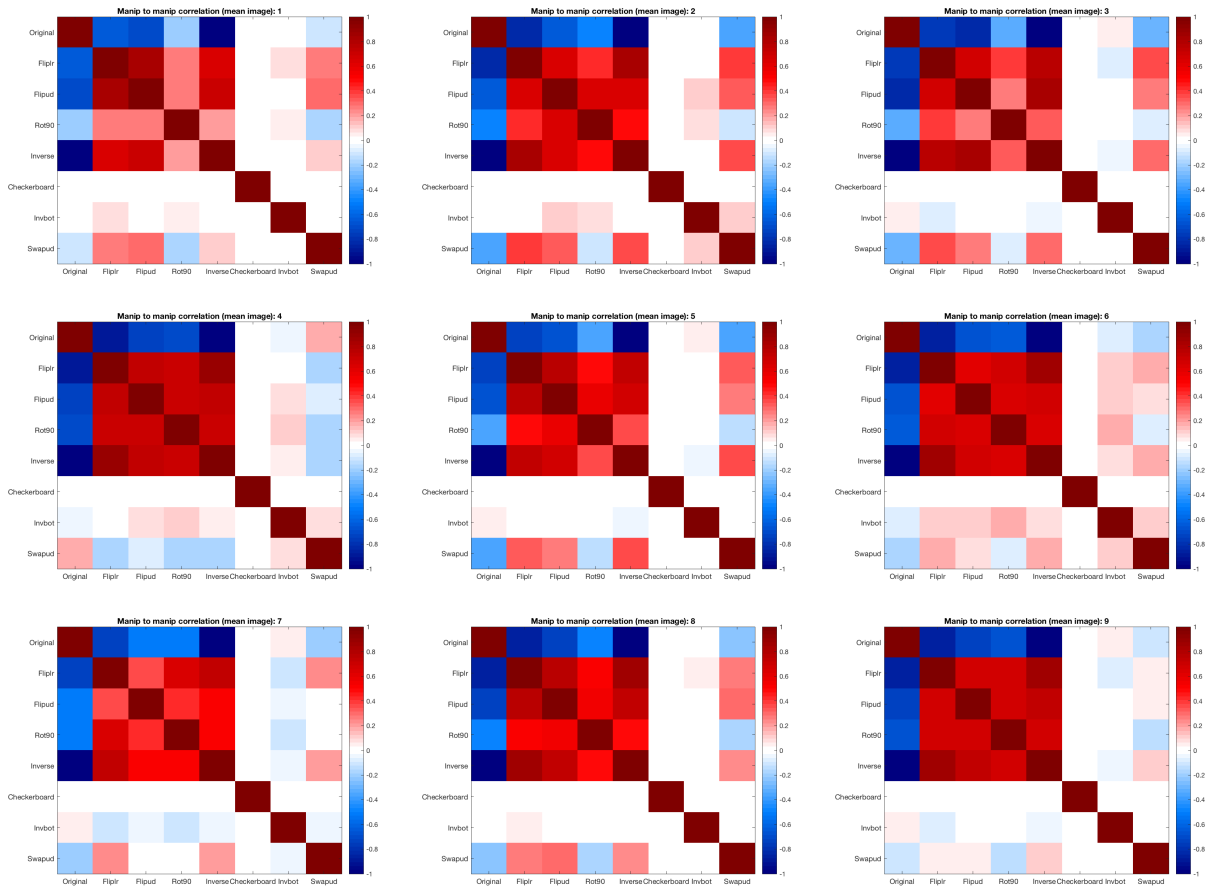


Figure 13: Digit by digit correlations between datasets. Each row and column of each correlation matrix is a manipulated MNIST datasets.

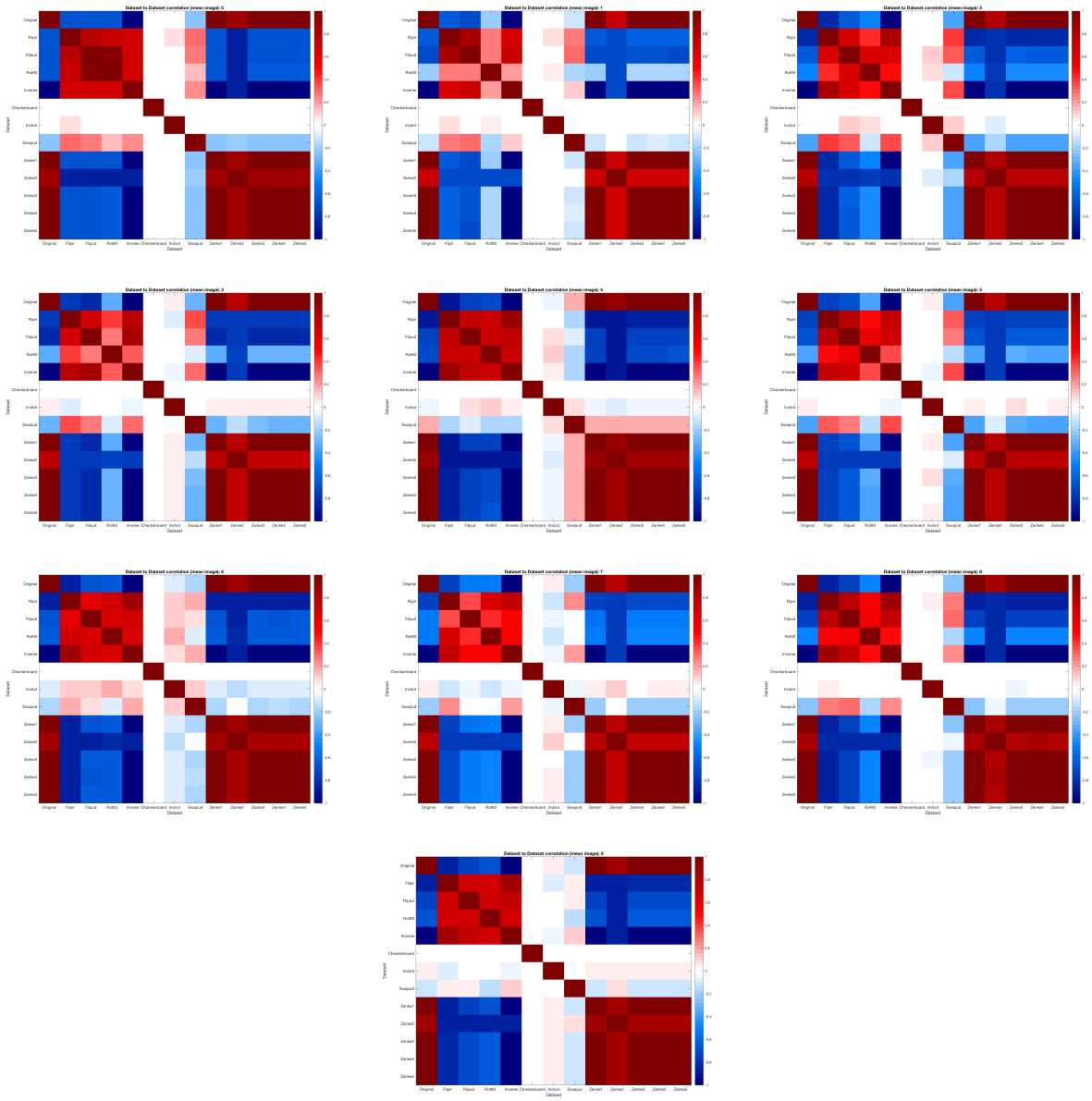

Figure 14: Digit by digit correlations between datasets. The first 8 rows and columns of each correlation matrix are the manipulated MNIST datasets we used initially, last 5 rows and columns are the manipulated MNIST datasets used by Zenke et al.

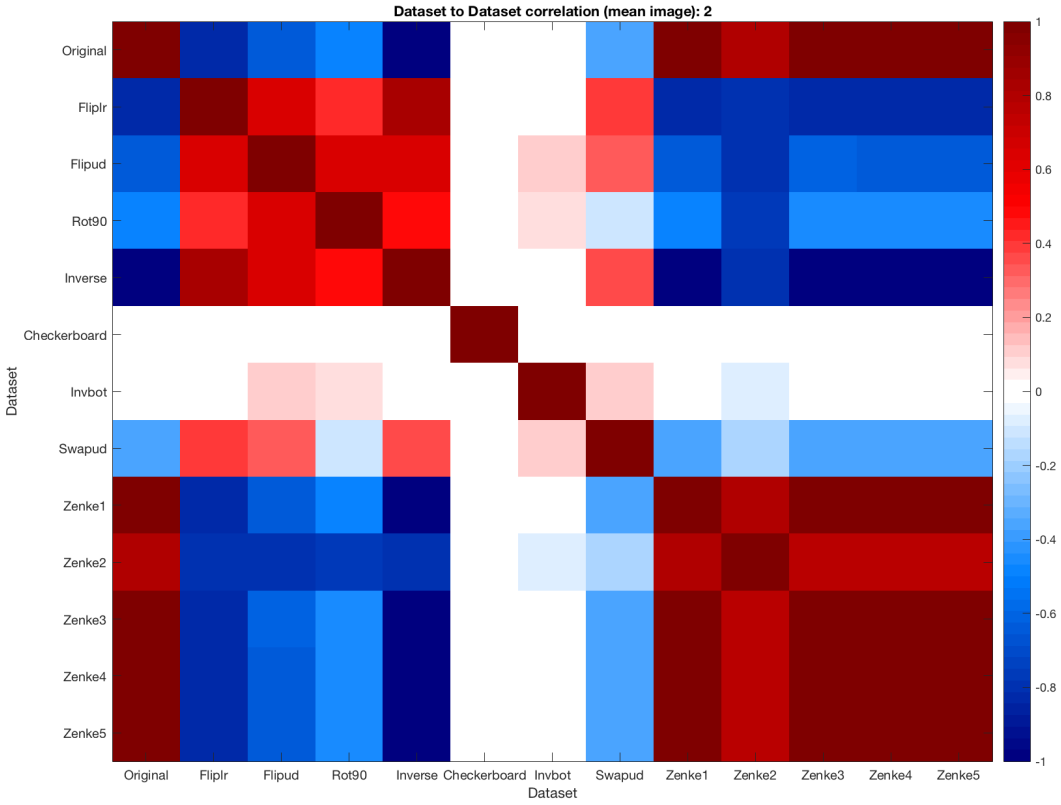


Figure 15: Representative large cross-dataset correlation

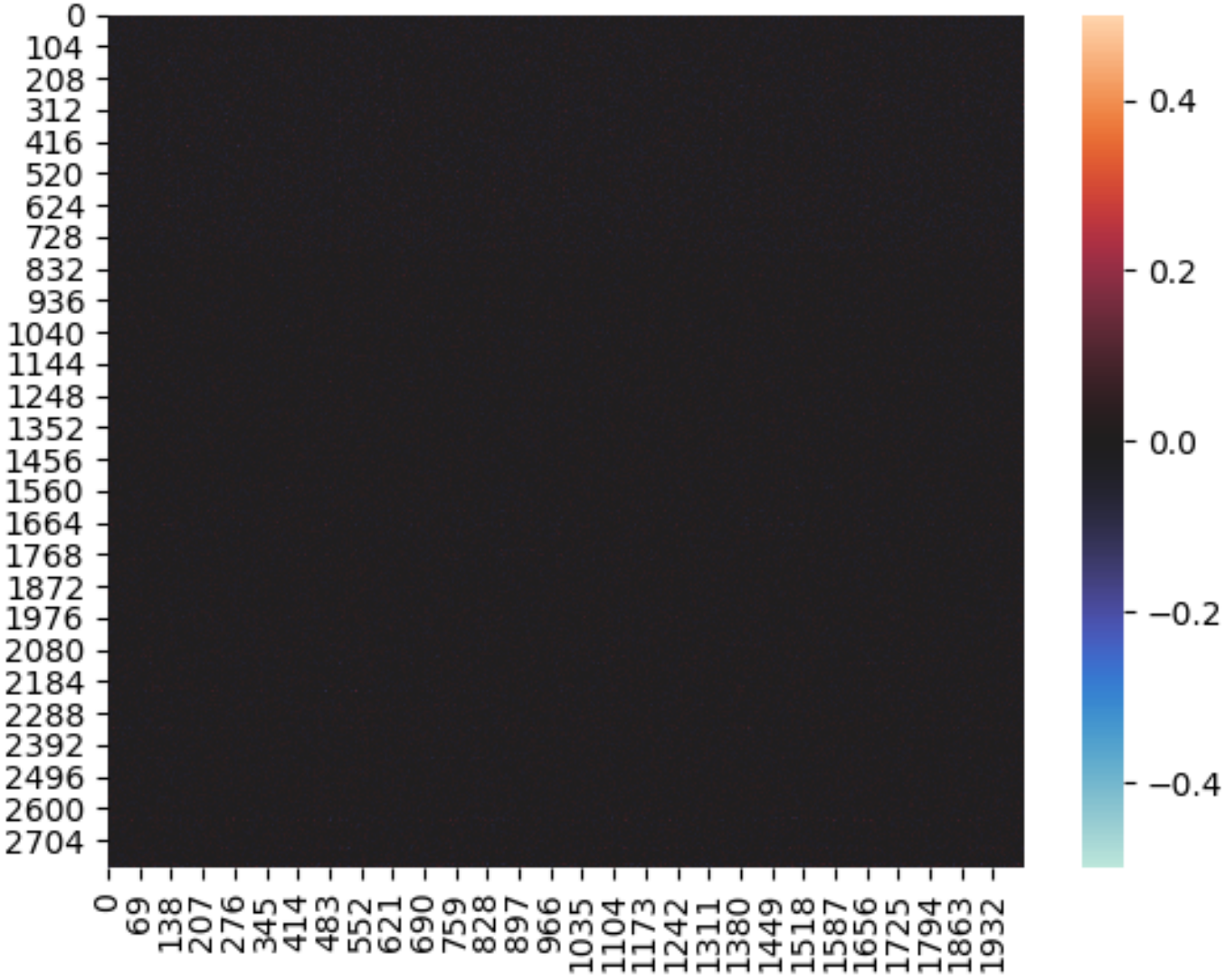


Figure 16: Weightspace after one iteration

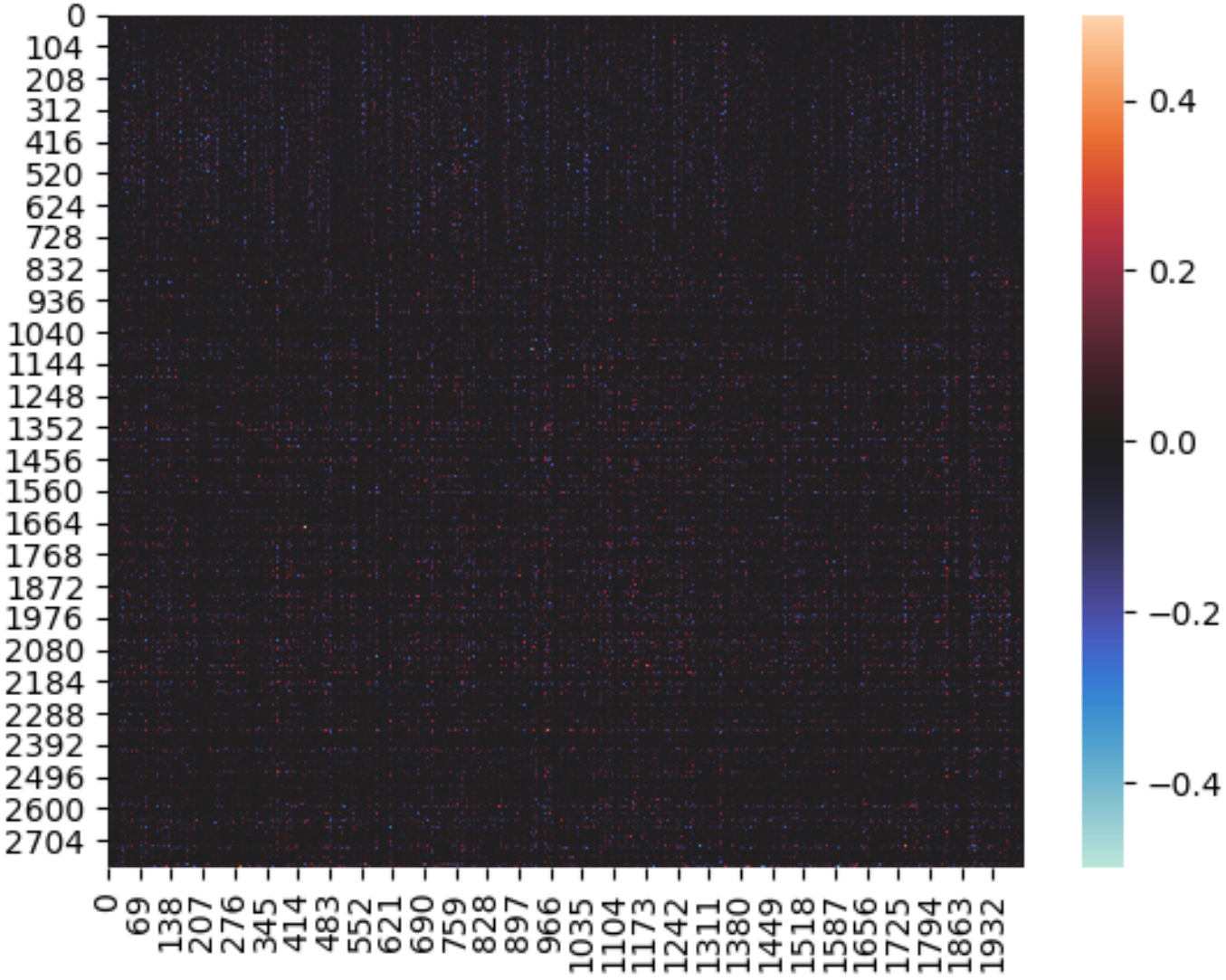


Figure 17: Weightspace after five iterations

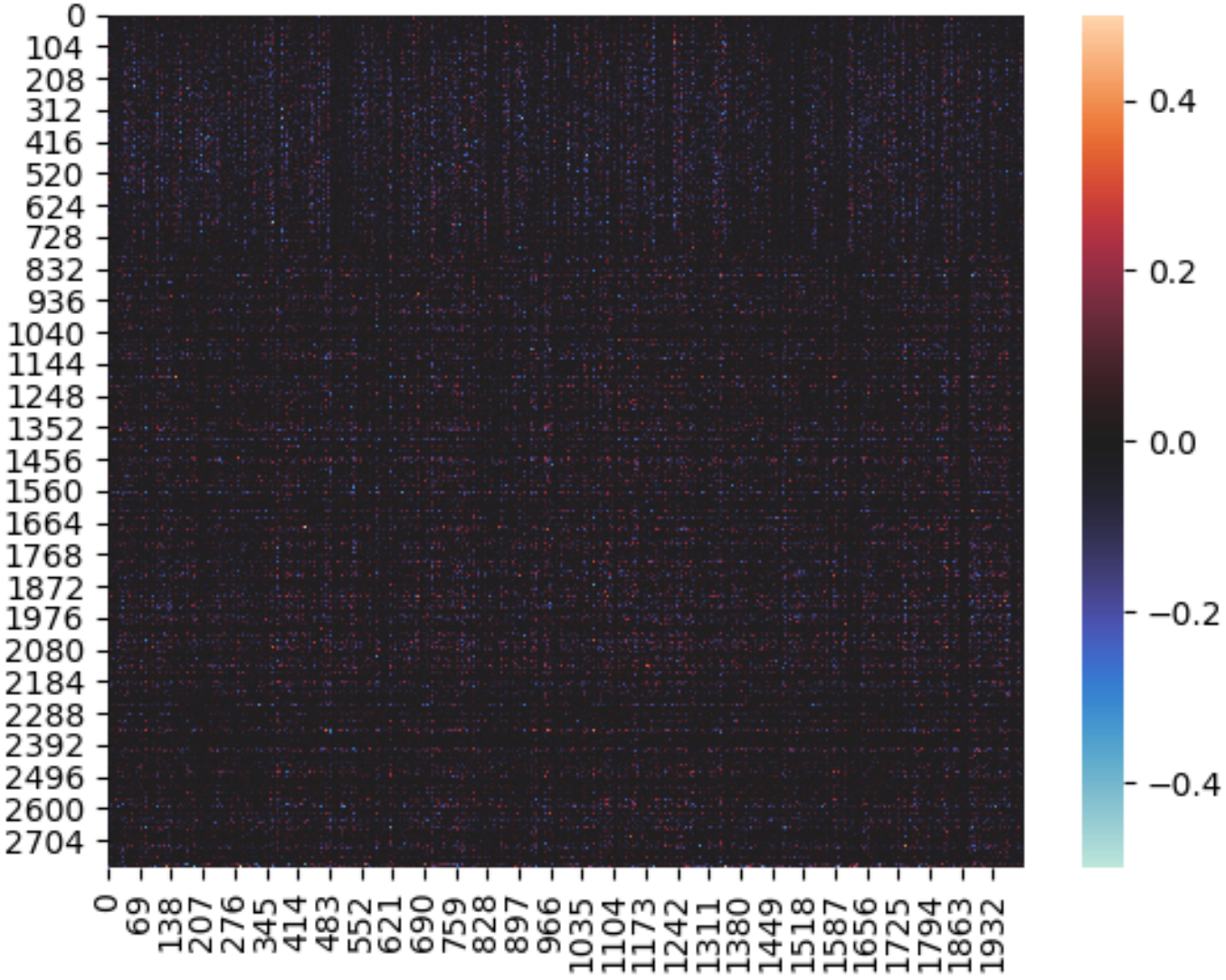


Figure 18: Weightspace after ten iterations