LEARNING REPRESENTATIONS FROM TEMPORALLY SMOOTH DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Events in the real world are correlated across nearby points in time, and we must learn from this temporally "smooth" data. However, when neural networks are trained to categorize or reconstruct single items, the common practice is to randomize the order of training items. What are the effects of temporally smooth training data on the efficiency of learning? We first tested the effects of smoothness in training data on incremental learning in feedforward nets and found that smoother data slowed learning. Moreover, sampling so as to minimize temporal smoothness produced more efficient learning than sampling randomly. If smoothness generally impairs incremental learning, then how can networks be modified to benefit from smoothness in the training data? We hypothesized that two simple brain-inspired mechanisms – leaky memory in activation units and memory-gating – could enable networks to exploit the redundancies in smooth data. Across all levels of data smoothness, these brain-inspired architectures achieved more efficient category learning than feedforward networks. Finally, we investigated how these brain-inspired mechanisms altered the internal representations learned by the networks. We found that networks with multi-scale leaky memory and memory-gating could learn internal representations that "un-mixed" data sources which vary on fast and slow timescales across training samples. Altogether, we identified simple mechanisms enabling neural networks to learn more quickly from temporally smooth data, and to generate internal representations that separate timescales in the training signal.

1 INTRODUCTION

Events in the world are correlated in time: the information that we receive at one moment is usually similar to the information that we receive at the next. For example, when having a conversation with someone, we see multiple samples of the same face from different angles over the course of several seconds. However, when we train neural networks for categorization or reconstruction tasks, we commonly use randomly ordered data. What are the effects of the temporal ordering of training data on the performance of learning systems? In particular, even in the case where a learning system is only asked to categorize or compress one input at a time, how is learning affected by temporal autocorrelation ("smoothness") in the training data?

We investigated this question in three stages. First, we examined the effects of smooth training data on feedforward neural networks performing category learning. Second, we investigated conditions under which these classification networks might take advantage of smooth data. In particular, we tested whether two brain-inspired mechanisms (leaky memory and memory-resetting) would enable a neural network to benefit from the temporal smoothness in the training data. Finally, we studied the internal representations learned by neural networks equipped with these brain-inspired mechanisms. In particular, we tested whether the networks with leaky memory and resetting could learn internal representations that separate data sources with differing levels of smoothness.

2 RELATED WORK

Effects of sampling strategies on learning. The ordering of training examples affects the speed and quality of learning. For example, learning can be speeded when training starts with "easier"

examples, and then proceeds through examples of gradually increasing difficulty (Elman, 1993; Bengio et al., 2009; Kumar et al., 2010; Lee & Grauman, 2011). Similarly, learning can be more efficient if training data is organized so that the magnitude of weight updates increases over training samples (Gao & Jojic, 2016).

In contrast to the prior work, we only explore the effects of ordering similar items nearby in time; we do not manipulate the order based on item difficulty or proximity to category boundaries. Our goal is to identify mechanisms that can aid efficient learning across many levels of temporal autocorrelation, adapting to what is present in the data. This ability to adapt to the properties of the training data is important in real-world settings, where a learner may lack control over the training order, or prior knowledge of item difficulty is unavailable.

Potential costs and benefits of training with smooth data. Temporal smoothness in data may slow learning. For instance, in smooth training sets, consecutive items are similar, and therefore the gradients induced by them will be related, especially early in training. If we consider the average of the gradients induced by the whole training set as the "ideal" gradient, then subsets of similar samples provide a higher-variance (i.e. noisier) estimate of this ideal. Moreover, smoothness in data may slow learning due to *catastrophic forgetting* (French, 1999). Suppose that, for smoother training, we sample multiple times from a category before moving to another category. This means that the next presentation of each category will be, on average, farther apart from its previous appearance. This farther distance subjects the category to a greater amount of forgetting, hence slows learning overall. Smoother training data might also benefit learning. For example, there may be some category-diagnostic features that will not reliably be extracted by a learning algorithm unless multiple weight updates occur for that feature nearby in time; smoother training data would be more liable to present such features nearby in time.

3 Research Questions and Hypotheses

1. How does training with temporally smooth data affect learning in conventional feedforward networks? *In light of the work reviewed above, we hypothesized that temporally smooth data would slow learning.*
2. How can neural networks benefit from temporal smoothing, in terms of either learning efficiency and in learning more meaningfully structured internal representations? *We hypothesized that a combination of two brain-inspired mechanisms – leaky memory and memory-resetting – could enable networks to learn more efficiently from temporally smooth data.*

4 Effects of temporal smoothness in training data on learning in feedforward neural networks

We first explored how smoothness of data affects the speed and accuracy of category learning (classification) in feedforward networks. See Appendix A.1 for similar results with unsupervised learning.

4.1 Methods

4.1.1 Manipulating smoothness in training data

We manipulated smoothness in training data by varying the number of consecutive samples drawn from the same category. We began each training session by generating a random "category order", which was a permutation of the numbers from 1 to N (e.g. the ordering in Figure 1.B is 2-1-3). The same category order was used for all smoothness conditions in that training session.

To sample with minimum smoothness, we sampled exactly one exemplar from each category, before sampling from the next category in the category order (1 repetition) (Figure 1.B). This condition is called "minimum smoothness" because all consecutive items were from different categories, and there were not more examples from a category until all other categories were sampled. We increased smoothness by increasing the number of consecutive samples drawn from each category (3 repetitions and 5 repetitions in Figure 1.B). Finally, we also used the standard random sampling method, in which items were sampled at random, without replacement, from the training set (Figure 1.B).

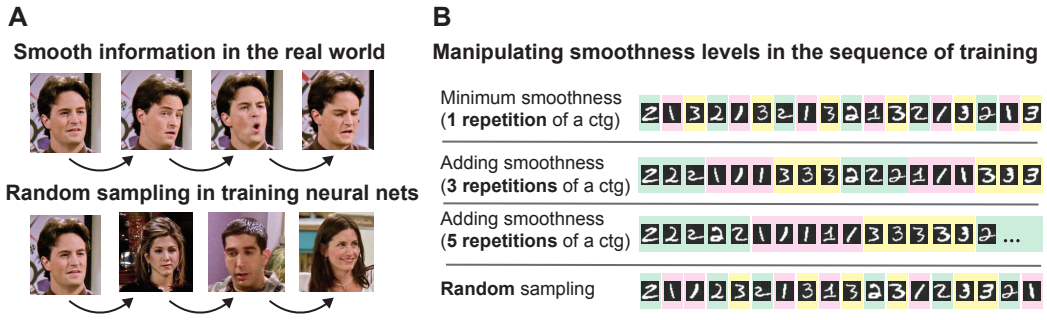

Figure 1: Temporal smoothness in the real world and in neural network training. *A) Top: smooth information in the real world. Bottom: randomly ordered data in training neural networks.*[2] *B) Manipulating smoothness levels in training data using the ordering of training samples. Colored rectangles indicate the amount of smoothness induced by repeating a category.*

The training set was identical across all conditions, as was the order in which samples were drawn from within a category (Figure 1.B).

4.1.2 Feedforward Neural Network

Dataset. We tested MNIST, Fashion-MNIST, and further synthetic datasets containing low category overlap (LeCun et al., 2010; Xiao et al., 2017). An example synthetic dataset is shown in Appendix A.2. For creating synthetic datasets, we used Numpy (Harris et al., 2020). For creating and testing the models, we used PyTorch(Paszke et al., 2019).
Learning rule and objective function. We used backpropagation with both mean squared error (MSE) and cross-entropy (CE) loss functions. The results reported here are using MSE, primarily for the ease of comparison with later reconstruction error measures in this manuscript. However, the same pattern was observed using CE loss, as shown in Appendix A.3. Also, it has been shown MSE loss provides comparable performance to commonly utilized classification models with CE loss function (Illing et al., 2019). To test incremental learning, we employed stochastic gradient descent (SGD), updating weights for each training sample.
Optimization, initialization, and activation function. We tested the model both with and without RMSprop optimization, along with Xavier initialization method (Tieleman & Hinton, 2012; Glorot & Bengio, 2010). We applied ReLU to hidden units and Softmax or Sigmoid to the output units.
Hyperparameters. For MNIST and Fashion-MNIST, we used a 3-layer fully connected network with (784, 392, 10) dimensions and a learning rate of 0.01. The learning rate was not tuned for a specific condition. We used the same learning rate across all conditions so that the smoothness would be the only variable. We also averaged the results over 5 runs, each with a different randomization seed, to control for the potential effects of initialization. For hyperparameters in synthetic dataset see Appendix A.2. When RMSprop was implemented, beta-1 and beta-2 were set to 0.9 and 0.99, respectively (Ruder, 2016).

4.2 Results

Smooth training data slowed incremental learning (Figure 2.A). Moreover, minimum smoothness yielded more efficient learning than random sampling (Figure 2.A). These observations generalized across all tested datasets and across MSE and CE loss, with and without RMSprop optimization.

4.3 Discussion

The superiority of minimum smoothness over other conditions suggests that any level of smoothness slows incremental learning, even the smoothness that can occur by chance in random sampling (Figure 2.A). Therefore, given a fixed time budget for training, a sampling strategy that minimizes smoothness can reach a higher performance than random sampling.

[2]Photos in this section are taken from the FRIENDS TV series, Warner Brothers (Kauffman et al., 1994).

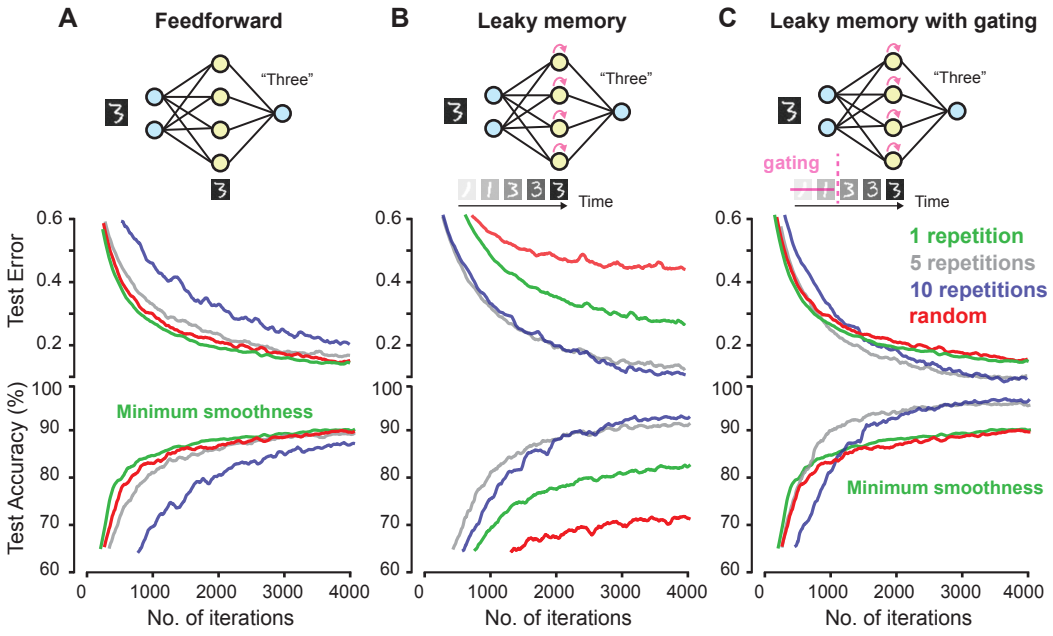

Figure 2: Neural architectures for classifying temporally smooth data. *A) Test error (MSE loss) and test accuracy in SGD training of a feedforward neural network (MNIST data) across different smoothness levels. B) The same as A, for a neural network with leaky memory in internal representations. C) The same as A, for a neural network with leaky memory and gating mechanism. Random sampling in Figure 2.A and random sampling in Figure 2.C are the same and are used for comparison to the commonly practiced method. [Curves in this figure have been averaged over 5 runs with different initialization and were further smoothed using a 100-iteration moving average.]*

The advantage of sampling with minimum smoothness may relate to reducing the representation overlap across consecutive training items. One of the main approaches in addressing the damage of *catastrophic forgetting* is implementing techniques that reduce the overlap between the learned representations (French, 1999). Such methods mainly rely on orthogonalization of internal representations (French, 1999). Here, we did not explicitly seek to reduce interference, but sampling with minimum smoothness does likely reduce the representation overlap of nearby items.

Additionally, training with minimum smoothness may improve learning by keeping the distribution of sampled categories almost perfectly uniform. It has been shown that training with a "low-discrepancy" sequence, a sequence with uniformly distributed data, avoids classification bias and enhances learning (Iwata & Ishii, 2002; Mishra & Rusch, 2020).

5 Exploiting temporal smoothness in training data for learning in neural networks

Humans must often learn from real-world data, where the order of samples is determined by the environment, and is typically smooth. Therefore, our brains may possess mechanisms for mitigating the costs and maximizing the benefits of temporally smooth training sets. In particular, we focused on two properties of cortical circuits that are relevant to incremental learning: (i) all cortical dynamics exhibit *autocorrelation* on the scale of milliseconds out to many seconds, so that correlation in consecutive internal states is unavoidable (Murray et al., 2014; Honey et al., 2012; Bright et al., 2020); (ii) neural circuits appear to shift state suddenly at *event boundaries*, and this appears to be associated with "resetting" of context representations (DuBrow et al., 2017; Chien & Honey, 2020; Baldassano et al., 2018). Inspired by these two features, we examined the effects of the following mechanisms on learning from smooth data: (i) leaky memory in internal representations; (ii) a memory gating mechanism that resets internal representation at transitions between categories.

5.1 BRAIN-INSPIRED NEURAL ARCHITECTURE FOR SUPERVISED LEARNING

Can brain-inspired architectural tweaks – leaky memory and memory gating – make the learning more flexible in supervised classification tasks? We hypothesized that networks with leaky memory could benefit from repeated structures in data, and gating mechanisms could further enhance learning efficiency by reducing the interference from irrelevant structures.

5.1.1 METHODS

Leaky memory: We added leaky memory to the internal representations (hidden units) by linearly mixing them across consecutive time points. Hidden unit activations were updated according to following function:

$$H(n) = \alpha H(n-1) + (1-\alpha) ReLU(W_{IH} I(n)) \tag{1}$$

where $H(n)$ is the state of the hidden units for trial n, $I(n)$ is the state of the input units for trial n, α is a leak parameter, W_{IH} are the connections from the input layer to the hidden layer, and $ReLU$ is a rectified linear activation. We set α = 0.5 in these experiments.

Memory Gating: In order to reduce the interference between items from different categories in the leaky memory, we employed a gating mechanism to reset memory at the transitions between categories. Therefore, if sample n was drawn from a category other than the category of sample $n-1$, then we set α = 0 in Eq.(1) on that trial n (Figure 3.B).

The learning algorithm, optimization and initialization methods, and the hyperparameters were identical to those used in training and testing feedforward neural networks.

5.1.2 RESULTS

Smoothness in training data increased learning efficiency in learners with leaky memory, as shown in Figure 2.B. This result is in contrast to the detrimental effects of smoothness in memoryless learners (Figure 2.A). Moreover, adding a gating mechanism to the leaky memory units further increased their learning (Figure 2.C). In learners with leaky memory and gating, all levels of smoothness significantly outperformed random sampling and sampling with minimum smoothness (Figure 2.C). These findings generalized across MNIST, Fashion-MNIST, and synthesized datasets [Appendix A.4].

5.1.3 DISCUSSION

When representation at one time point shared some information with previous time points, learners with leaky memory could exploit this property for more efficient category learning (Figure 2.B, C). Importantly, the resetting mechanism prevented the mixing of hidden representations from samples of different categories, allowing the system to benefit most from the smoothness, while not suffering from between-category interference.

Across all levels of smoothness in training data, networks with leaky memory and resetting surpassed the performance of feedforward networks (Figure 2.C, A). This result is notable because the enhanced learning relies only on a linear mixture of activations over time-steps, which is easy to implement, and is found ubiquitously in brain dynamics (Honey et al., 2012; Murray et al., 2014).

Finally, as we note in Appendix A.5, the leaky memory approach (mixing activation across samples) is different from momentum optimization (mixing weight updates across samples) and mini-batch training (mixing of gradients across samples).

5.2 BRAIN-INSPIRED ARCHITECTURES FOR UNSUPERVISED LEARNING ACROSS TEMPORAL SCALES

In the real world, we may need to learn from data with multiple levels of smoothness. For instance, returning to the example of having a conversation, we can see that the features around a person's mouth change quickly, while those of their face outline change more slowly (Figure 3.A). Moreover, there are no pre-defined labels to support the learning of representations in this setting. We hypoth-

esized that neural networks equipped with multi-scale (i.e. fast and slow) leaky memory could learn to effectively represent structures that vary on multiple scales.

5.2.1 METHODS

Dataset. To test the un-mixing abilities of our networks, we synthesized a simplified training dataset which contained three levels of temporal structure. The input to the model at each timepoint consisted of 3 subcomponents (top, middle, bottom), and each subcomponent had two elements. Each subcomponent was generated to express a different level of smoothness over time: the top, middle and bottom rows changed feature-category every 1, 3 or 5, iterations, respectively. (Figure 3.B). The individual features sampled at each time were generated as the sum of (i) an underlying binary state variable (which would switch every 1, 3 or 5 iterations) and (ii) uniformly-distributed noise (Appendix A.6). As a result, the model was provided with features that varied at 3 time-scales: fast (top row), medium (middle row), and slow (bottom row). For creating the dataset, and designing and analyzing the models we used Numpy(Harris et al., 2020).
Architectures. We used the same brain-inspired mechanisms for unsupervised learning models: leaky memory and gating mechanism. To evaluate the effectiveness of the added mechanisms, we compared 4 types of autoencoder (AE) models: i) Feedforward AE (Figure 3.C); ii) AE with leaky memory in internal representations (Figure 3.D); ii) AE with multi-scale leaky memory in internal representations (Figure 3.E), inspired by evidence showing that levels of processing in the brain can integrate information at different time-scales (Honey et al., 2012; Murray et al., 2014; Bright et al., 2020), and that multiple time-scales are present even within a single circuit (Bernacchia et al., 2011; Ulanovsky et al., 2004); iv) AE with multi-scale leaky memory in internal representations and boundary-sensitive gating (Figure 3.F), motivated by the evidence showing that processing in cortical circuits are sensitive to event-boundaries and these boundaries can shift learned representations (DuBrow et al., 2017; Chien & Honey, 2020).
Learning algorithm, optimization, and initialization. We used backpropagation with MSE loss, both with and without RMSprop optimization method, and Xavier initialization (Tieleman & Hinton, 2012; Glorot & Bengio, 2010). We applied ReLU and Sigmoid as activation functions for hidden and output units, respectively.
Hyperparameters. To implement leaky memory at multiple scales, we varied the time constants across the nodes in the hidden layer. Thus, the variable in Eq.(1) was set to 0, 0.3, and 0.6 for "short memory", "medium-memory", and "long-memory" nodes, respectively. The networks were 3-layer, fully connected autoencoders with (6, 3, 6) dimension. Learning rate was 0.01. In cases where RMSprop was implemented, the beta-1 and beta-2 were set to 0.9 and 0.99. For leaky memory in internal representations in Eq.(1) was set to 0.5.
Learning Efficiency Measure. Learning speed was measured using the reconstruction error of the test data, computed as the MSE across all 3 subcomponents of each data sample.
Un-mixing Measures. We also measured the system's ability to "un-mix" the time-scale of input. By un-mixing time-scale, we refer to the system's ability to not only learn representations that effectively reconstruct the input but to learn representations that selectively track distinct sources that are used to generate each training sample. In other words, we tested whether short-memory, medium-memory and long-memory nodes would track the fast-, medium-, and slow-changing data features, respectively. To this end we measured the Pearson correlation between each hidden unit (short-memory, medium-memory, or long-memory) and all of the data features (fast, medium and slowly changing). We then quantified the "timescale-matching" – e.g. whether the long-memory node was correlated with the slowly-varying data feature – and the "timescale-selectivity" – e.g. whether the long-memory node was more correlated with slowly-varying features than the other features.

5.2.2 RESULTS

We first confirmed that all of the autoencoder (AE) models could learn to reconstruct the input (Figure 4.A). The two most efficient architectures were the multi-scale leaky AE with gating and the memoryless AE. Next, to determine whether each model could "un-mix" data sources that change on different timescales, we tested the relationship between the hidden states of the model and the fast-, medium-, and slow-varying data features.

Multi-scale networks with leaky memory and gating could successfully un-mix fast and slow data sources. The individual hidden state units in this AE were most strongly correlated with their corresponding data features (e.g. long-memory nodes correlated most strongly with slowly-varying data,

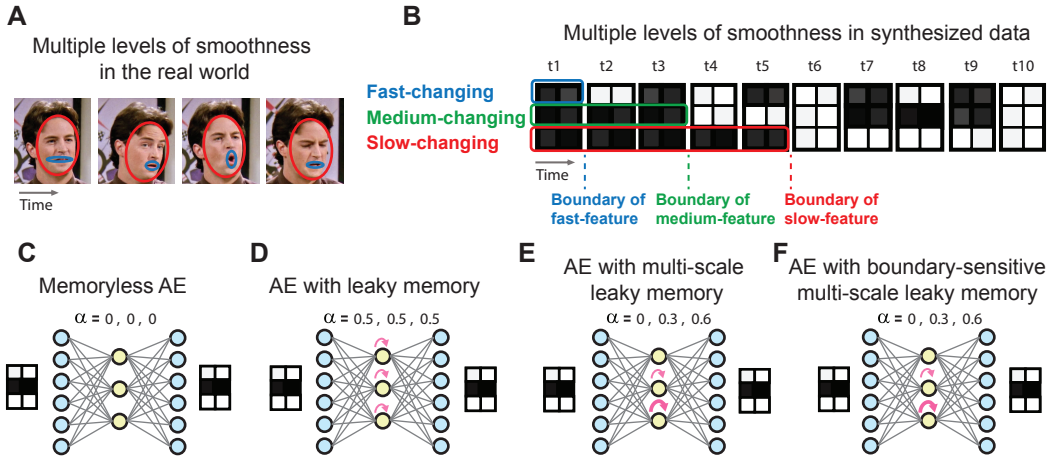

Figure 3: Unsupervised learning from data with multiple levels of smoothness. *A) Example of multiple levels of smoothness in samples from the real world: mouth changing fast, while face-shape changes slowly. B) Multiple levels of smoothness in synthesized data: top row changes every item; middle row changes every 3 items; bottom row changes every 5 items. X-axis shows time, each 3-by-2 item is one sample. C) Memoryless feedforward AE. D) AE with leaky memory. E) AE with multi-scale leaky memory. F) AE with multi-scale leaky memory and a gating mechanism sensitive to feature boundaries. The gating mechanism responds to the state change of each feature (e.g. long-memory nodes were reset at the boundary of slow-changing features).*

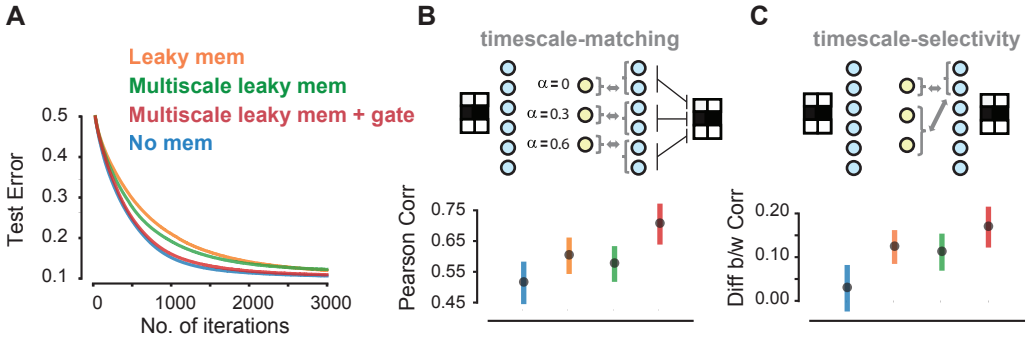

Figure 4: Learning performance and internal representations of autoencoder models with and without memory. *A) Reconstruction test error (MSE loss) during training for individual items across 4 different AE models. All the curves in this plot have been averaged over 20 runs with different initialization. B) Comparing the "timescale-matching" ability of models, as measured by the squared Pearson correlation of short-memory, medium-memory and long-memory internal representations (hidden units) with slow, medium and fast-varying output units. C) Comparing the "timescale-selectivity" of models, ability by computing the difference between the squared Pearson correlations for time-scale matching units and non-matching units, e.g. long-memory correlated with slow features minus long-memory correlated with fast and medium features. In no-memory systems and in leaky-memory systems with uniform memory, we measured these correlations for hidden units in the corresponding position as those in the multi-scale leaky memory. Error bars show the mean and standard deviation across 10,000 bootstraps, with 20 values per bootstrap.*

Figure 4.B), and this effect was also selective (i.e. the long-memory node was more correlated with the slow features than with the other features (Figure 4.C)).

5.2.3 DISCUSSION

The autoencoder model with multi-scale leaky memory and boundary-gating was most successful in learning internal representations which tracked distinct timescales of the input. Slowly (or quickly) varying features were extracted by slowly (or quickly) varying subsets of the network, analogous to a matched filter (see also Mozer (1992)). Features that change on different timescales may correspond to different levels of structure in the world (Wiskott & Sejnowski, 2002). Thus, by adding leaky memory and memory-gating to a simple feedforward AE model, we equip it with an ability to separate different levels of structure in the environment. Moreover, because intrinsic dynamics vary on multiple scales in the human brain (Stephens et al., 2013; Murray et al., 2014; Honey et al., 2012; Raut et al., 2020), this implies that slowly-varying brain circuits may be biased to extract slowly-varying structure from the world (Honey et al., 2017).

6 CONCLUSION

Inspired by temporal properties of the training signal and the learning architectures in primate brains, we investigated how the smoothness of training data affects incremental learning in neural networks.

First, we examined the speed of learning. We found that smoothness slows learning in memoryless learners (feedforward neural net), but speeds learning in systems with leaky memory in internal representations (Figure 2). Moreover, adding a simple gating mechanism to leaky-memory networks enabled them to flexibly adapt to the smoothness in the data, so that they could benefit from repeating structure while not mixing unrelated information.

Second, we examined the representations learned when unlabeled data contained temporal structure on multiple smoothness levels. Neural networks with multi-scale memory and feature-sensitive gating learned representations that un-mixed features varying on different timescales. If distinct timescales in data reflect distinct data generators, these "un-mixed" representations may provide a more "meaningful" description of the input data (Mitchell, 2020; Mahto et al., 2020).

Leaky network memory produced more efficient learning and more interpretable representations, even though the networks were trained with a learning rule that did not employ any temporal information. In particular, all networks were trained incrementally using backpropagation and a loss function that only depended on the immediate state of the network. Architectures with leaky memory and gating can thus exploit temporal structure in a way that is computationally simpler and more biologically plausible than backpropagation through time (Sutskever, 2013; Lillicrap & Santoro, 2019). With respect to biological plausibility, we note that the leaky-memory-plus-gating system works well even for autoencoders (Figure 4), for which there are simple activation-based learning rules that do not require the propagation of partial derivatives (Lee et al., 2015). On the computational side, we highlight that the gradients computed for the leaky memory networks were, in a sense "incorrect", because the update rule was unaware of the recurrent leak connections, and yet learning in the leaky nets was still faster than in feedforward nets, for which the gradients should be more accurate.

Future work should test whether these results generalize to larger architectures and datasets, and should include a broader search of the hyperparameter space. The results may have some generality, because we used simple architectures and made few domain-specific assumptions, but at present the results serve as demonstrations of basic phenomena that we hope to generalize. It will also be interesting to investigate the broader consequences of learning with leaky memory: for example, human internal representations of natural sensory input sequences appear to be smooth in time, in contrast to the representations of most feedforward nets (Hénaff et al., 2019), and training with smooth data and leaky memory could potentially reduce this difference.

In sum, we identified simple mechanisms enabling neural networks to learn more quickly from temporally smooth data and to generate internal representations that separate distinct timescales of the data.

REFERENCES

Christopher Baldassano, Uri Hasson, and Kenneth A Norman. Representation of real-world event schemas during narrative perception. *Journal of Neuroscience*, 38(45):9689–9699, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Alberto Bernacchia, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A reservoir of time constants for memory traces in cortical neurons. *Nature neuroscience*, 14(3):366–372, 2011.

Ian M Bright, Miriam LR Meister, Nathanael A Cruzado, Zoran Tiganj, Elizabeth A Buffalo, and Marc W Howard. A temporal record of the past with a spectrum of time constants in the monkey entorhinal cortex. *Proceedings of the National Academy of Sciences*, 117(33):20274–20283, 2020.

Hsiang-Yun Sherry Chien and Christopher J Honey. Constructing and forgetting temporal context in the human cerebral cortex. *Neuron*, 2020.

Sarah DuBrow, Nina Rouhani, Yael Niv, and Kenneth A Norman. Does mental context drift or shift? *Current opinion in behavioral sciences*, 17:141–146, 2017.

Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Tianxiang Gao and Vladimir Jojic. Sample importance in training deep neural networks. 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Olivier J Hénaff, Robbe LT Goris, and Eero P Simoncelli. Perceptual straightening of natural videos. *Nature neuroscience*, 22(6):984–991, 2019.

Christopher J Honey, Thomas Thesen, Tobias H Donner, Lauren J Silbert, Chad E Carlson, Orrin Devinsky, Werner K Doyle, Nava Rubin, David J Heeger, and Uri Hasson. Slow cortical dynamics and the accumulation of information over long timescales. *Neuron*, 76(2):423–434, 2012.

Christopher J Honey, Ehren L Newman, and Anna C Schapiro. Switching between internal and external modes: a multiscale learning principle. *Network Neuroscience*, 1(4):339–356, 2017.

Bernd Illing, Wulfram Gerstner, and Johanni Brea. Biologically plausible deep learning—but how far can we go with shallow networks? *Neural Networks*, 118:90–101, 2019.

Kazunori Iwata and Naohiro Ishii. Discrepancy as a quality measure for avoiding classification bias. In *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, pp. 532–537. IEEE, 2002.

Marta Kauffman, David Crane, and Kevin S Bright. Friends. season 1, 1994. Produced by Warner Home Video, [Online; Retrieved on 10th September, 2020].

M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in neural information processing systems*, pp. 1189–1197, 2010.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015.

Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR 2011*, pp. 1721–1728. IEEE, 2011.

Timothy P Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current opinion in neurobiology*, 55:82–89, 2019.

Shivangi Mahto, Vy A. Vo, Javier S. Turek, and Alexander G. Huth. Multi-timescale representation learning in lstm language models, 2020.

Siddhartha Mishra and T Konstantin Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *arXiv preprint arXiv:2005.12564*, 2020.

Melanie Mitchell. On crashing the barrier of meaning in artificial intelligence. *AI Magazine*, 41(2): 86–92, 2020.

Michael C Mozer. Induction of multiscale temporal structure. In *Advances in neural information processing systems*, pp. 275–282, 1992.

John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, et al. A hierarchy of intrinsic timescales across primate cortex. *Nature neuroscience*, 17(12):1661–1663, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Ryan V Raut, Abraham Z Snyder, and Marcus E Raichle. Hierarchical dynamics as a macroscopic organizing principle of the human brain. *Proceedings of the National Academy of Sciences*, 117 (34):20890–20897, 2020.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Greg J Stephens, Christopher J Honey, and Uri Hasson. A place for time: the spatiotemporal structure of neural dynamics during natural audition. *Journal of neurophysiology*, 110(9):2019–2026, 2013.

Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Nachum Ulanovsky, Liora Las, Dina Farkas, and Israel Nelken. Multiple time scales of adaptation in auditory cortex neurons. *Journal of Neuroscience*, 24(46):10440–10453, 2004.

Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A APPENDIX

A.1 EFFECTS OF SMOOTHNESS ON CATEGORIZATION VERSUS RECONSTRUCTION TASKS

Classification networks (performing categorization) and autoencoder networks (performing reconstruction) were similarly affected by temporal smoothness of training data (Figure A.1). Increased smoothness decreased learning efficiency. Also, "minimum smoothness" sampling exhibited the best performance across both types of networks. We used 3-layer fully connected networks for both classification and reconstruction. The network dimension for classification was (16, 8, 4) and for reconstruction was (16, 4, 16). Learning rate was 0.01 for all conditions in classification, and 0.005 for all conditions in reconstruction.

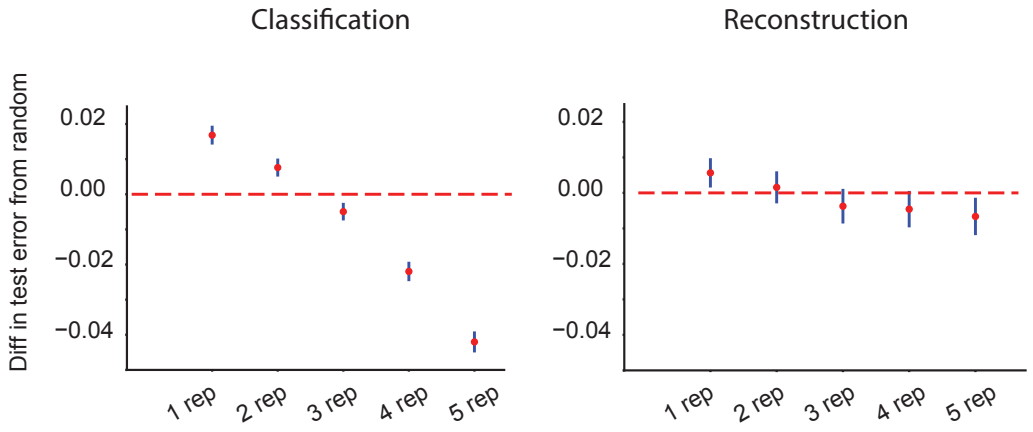

Figure A.1: Comparing the effects of smoothness on classification and reconstruction for a synthetic dataset (Figure A.2). *Left: Difference between the test error with random sampling and the test error in other sampling conditions (e.g. Error(random) - Error(1 repetition)) in the classification task. The dashed line shows the baseline for random sampling. Test errors were computed at the end of the first training epoch. We ran 100 runs with different weight initializations. Error bars show the mean and standard deviation of bootstrapping 10,000 times on 100 values from 100 runs. Right: As for the left panel, but for the reconstruction task.*

A.2 SYNTHETIC DATASET

We synthesized a dataset with low between-category overlap. The dataset consisted of 4 categories, each with 300 training items. Each item was a 1-by-16 vector. Different exemplars of a category were created by adding uniform noise to the template of the category.

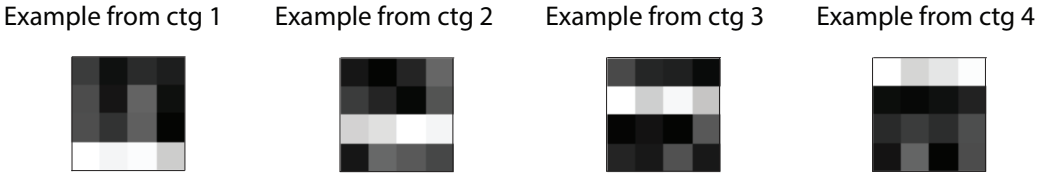

Figure A.2: Example items from each of the 4 categories in the synthetic dataset.

A.3 SMOOTHNESS EFFECTS FOR CLASSIFICATION USING CROSS-ENTROPY LOSS

Similar to mean-square error (MSE) loss, we found that temporally smooth data slowed category learning with training with cross-entropy (CE) loss. Figure A.3 shows this effect for the MNIST

dataset. We used the same neural architecture and hyperparameters as those in MSE, explained in section 4.1.2.

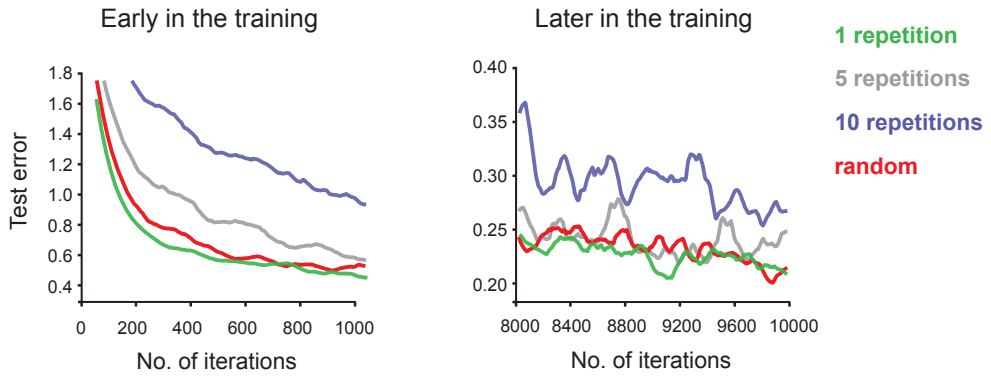

Figure A.3: Test error (CE loss) for SGD training of feedforward neural network on MNIST dataset, early vs later in the training process. *Each color shows a different smoothness condition in the training data.*

A.4 Categorization of synthetic data by a neural network with leaky memory and gating

Figure A.4 shows the effects of temporal smoothness in training data for neural network models equipped with leaky memory and gating for the synthetic dataset. Similar to the pattern observed in Figure2.B, we can see that, in the network with leaky memory, higher levels of smoothness generate better performance. Moreover, adding a gating mechanism enhanced learning, such that all levels of smoothness surpassed the "minimum smoothness" (1 repetition) condition, as was observed in Figure2.C.

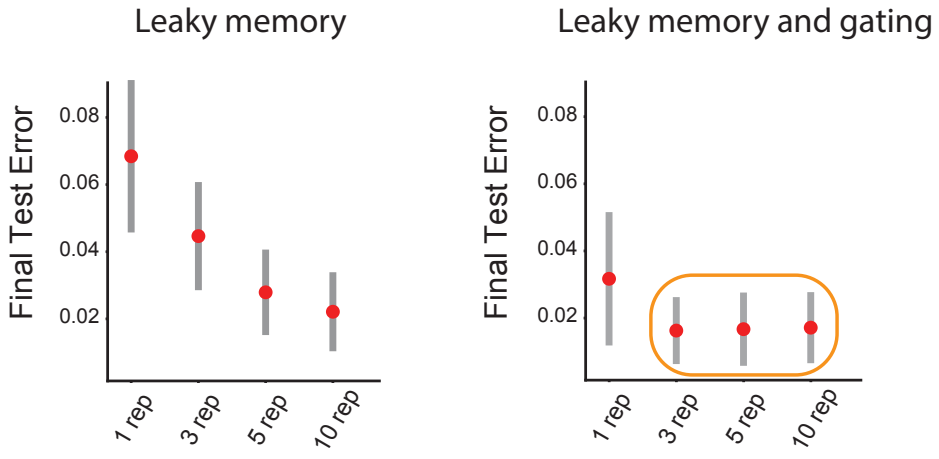

Figure A.4: Effects of temporal smoothness on categorization of synthetic data in networks with leaky memory. *Left: Test error (MSE loss) at the end of first training epoch with SGD on synthetic dataset, for network with leaky memory in internal representation. Right: As for the left panel, but for networks with leaky memory in internal representations and gating. Error bars show the mean and standard deviation of bootstrapping 10,000 times on 100 values from 100 runs with different weight initialization.*

A.5 EFFECTS OF SMOOTH DATA ON MINI-BATCH TRAINING

We explored how smooth data affects learning when weights were updated using mini-batch training. We used the MNIST dataset and trained each network with batches of size 16. Network dimension and other hyperparameters were identical to those used in incremental SGD. Our results showed that smoothness does not influence mini-batch training similarly to SGD training. Early in the training, minimum smoothness showed the fastest learning and higher levels of smoothness showed slower learning. However, later in the training, another pattern was observed: the condition with the smoothness level equal to the batch size (e.g. 16 repetitions for batch of 16) showed the greatest learning efficiency compared to both lower levels of smoothness (e.g. 10 repetitions) and higher levels of smoothness (e.g. 24 repetitions).

In connecting the mini-batch data to the results reported in Figure (2), consider that "smoothness" can happen at 2 levels: samples can be similar to one another within a batch ("smooth within a batch") and the composition of samples can be similar across consecutive mini-batches ("smooth across batches"). It seems that early in the training, the condition with minimum "within-batch smoothness" have the highest learning speed; this makes sense as the composition of each mini-batch is most reflective of the overall composition of the test data. However, later in the training, the condition with minimum "across-batch smoothness" have the best learning speed. Minimum across-batch smoothness refers to the condition where each batch consists of items from only one category (e.g. 16 repetitions for batch of 16). Note that when each individual batch contains items all from one category, this also implies that consecutive batches will not contain any items from the same categories, leading to a "minimum smoothness at the batch level". Thus, the advantage for minimum across-batch smoothness may be analogous to what is observed in Figure 2 at the single item level, but occurring at the batch level.

Future work is required to investigate how smooth data interact with mini-batch training.

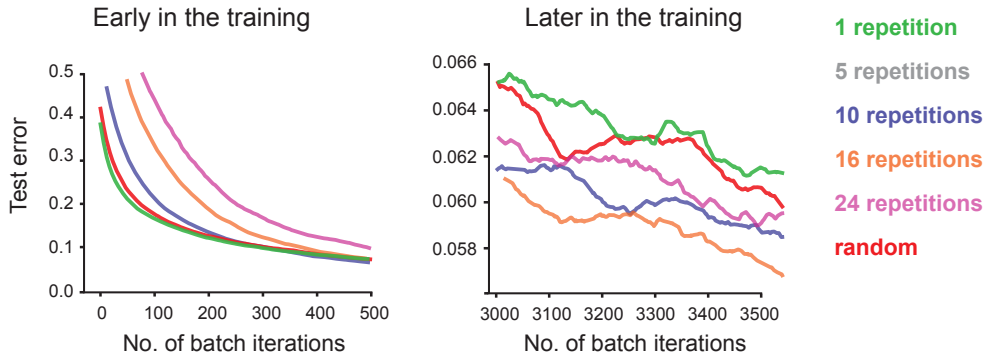

Figure A.5: *Left: Test error (MSE loss) in different levels of smoothness in data, early in mini-batch training of MNIST dataset for classification. Right: The same as left, for later in the training, toward the end of first epoch.*

A.6 SYNTHETIC DATASET WITH COMBINED MULTIPLE LEVELS OF SMOOTHNESS

We created training items composed of 3 "features". Each feature consisted of 2 elements, forming a 3-by-2 item. Note that each 3-by-2 data item was converted to a 6-by-1 vector before being input to the model, but in order to make clear the "features" in the data, we preserve the 3-by-2 arrangement in our illustrations.

Features could have two values: high or low. For each sample, the elements of each feature were generated as the sum of an average feature-value ("high" or "low" depending on the state) plus noise (sampled from a uniform distribution). The average feature-values associated with the "high" or "low" state of the feature was the same for all levels of smoothness. Therefore the only difference between levels of smoothness was the rate of change in the features.

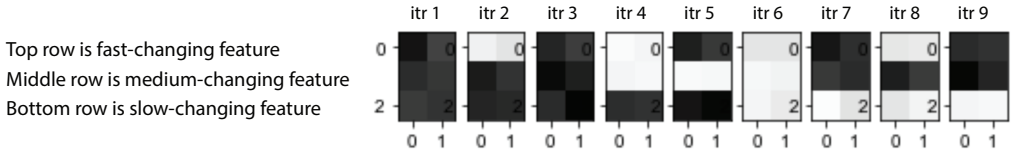

Figure A.6: First 9 training items from the synthetic data set. *Top, middle, and bottom row change every 1, 3, and 5 items, which results in 3 levels of smoothness. In the unsupervised learning setting, each item was both the input and the desired output of the model at each iteration.*

To form a training sequence with multiple levels of smoothness, we ordered training items such that different features varied at different scales: the top row changed every item, representing a feature changing at a fast timescale; the middle row changed every 3 items, representing a feature changing at a medium timescale; the bottom row changed every 5 items, representing a feature changing at a slow timescale.

A.7 Open-Source

The source code will be shared on authors' github after the reviewing process.