

Lazy learning: a biologically-inspired plasticity rule for fast and energy efficient synaptic plasticity.

Aaron Pache ^{*1} and Mark CW van Rossum ^{†1,2}

¹School of Mathematical Sciences

²School of Psychology, University of Nottingham, Nottingham, UK

March 26, 2023

Abstract

When training neural networks for classification tasks with backpropagation, parameters are updated on every trial, even if the sample is classified correctly. In contrast, humans concentrate their learning effort on errors. Inspired by human learning, we introduce lazy learning, which only learns on incorrect samples. Lazy learning can be implemented in a few lines of code and requires no hyperparameter tuning. Lazy learning achieves state-of-the-art performance and is particularly suited when datasets are large. For instance, it reaches 99.2% test accuracy on Extended MNIST using a single-layer MLP, and does so $7.6\times$ faster than a matched backprop network.

Recent progress in machine learning has been partly attributed to the use of large data sets [LeCun et al., 2015]. Even already large datasets are often augmented to further boost performance. However, repeatedly cycling over large datasets and adjusting the parameters is time and energy consuming. In classification tasks, backprop typically prescribes synaptic updates regardless of whether the classification was correct or incorrect; updating the network to be correct if it was wrong, but also updating to be *more* correct if it was right. Is this incessant updating necessary or do more efficient training rules exist?

Previous work has determined that high accuracies for many datasets can be achieved using only a small number of critical samples, called the coreset [Agarwal et al., 2005]. Methods have been developed to find these samples and focus training [Toneva et al., 2019, Paul et al., 2021, Coleman et al., 2020]. These methods require a model trained beforehand to identify key samples and form a coreset dataset. However, for a new dataset, it won't be clear what the coreset is or even whether the coreset is small or large. A simple algorithm that restricts learning and does not need tuning is lacking.

Humans appear to be more efficient by placing far more importance on mistakes and errors, a phenomenon known as the negativity bias [Rozin and Royzman, 2001]. A familiar example is the sharp 'jolt' we experience when typing a word incorrectly. When such mistakes occur a large event-related potential known as the error-related negativity (ERN) is elicited in the electroencephalogram signal (EEG), which usually precedes behavioural changes like post-error slowing and post-error improvement in accuracy [Kalfaoglu et al., 2017]. The ERN is linked to dopamine [Holroyd and Coles, 2002] - a plasticity modulator required for persistent forms of plasticity [O'Carroll and Morris, 2004]. Indeed, subjects learn better on stimuli that evoke a larger EEG [de Bruijn et al., 2020].

This suggests that, in contrast to backprop, biological learning largely occurs on mistakes, while skipping over correct samples. The reason might be that learning is emerging as a metabolically costly process. For instance, flies subjected to aversive conditioning and subsequently starved, died 20% sooner

*lpxap7@nottingham.ac.uk

†mark.vanrossum@nottingham.ac.uk

than control flies [Mery and Kawecki, 2005]. Similarly, flies doubled sucrose consumption after long term memory formation [Plaçais et al., 2017, Plaçais and Preat, 2013]. While the reason for these costs are unknown, they were recently estimated to be as much as 10mJ/bit [Girard et al., 2023].

The high energetic cost of learning is not unique to biology and is also important in traditional computers and in ANNs. Therefore restricting synaptic updates to incorrect samples could also benefit the training time and energy requirements of computer implementations of neural networks.

Results

With commonly used loss functions backprop updates the network parameters even if an example is classified correctly. For instance when using cross-entropy loss and one-hot encoding, weights are updated unless the network output exactly equals the target one-hot vector. In contrast, the idea in lazy learning is to only update the weights when a sample is misclassified; not unlike the classic perceptron algorithm [Rosenblatt, 1958]. We describe two variants: ‘pure lazy learning’ and ‘lazy learning’. In **pure lazy learning**, the network *only* learns when the presented sample is incorrectly classified; if the sample is correctly classified, no update occurs and the algorithm skips to the next example. This amounts to a single line of code

```
if (pred != target): #do backprop. (1)
```

As is shown below, pure lazy learning can lead to fragile generalization performance. We therefore also introduce **lazy learning** which instead remembers all incorrect samples and learns when the sample is currently misclassified or was at *any previous time* misclassified. Lazy learning maintains a vector of booleans with as many indices as there are samples in the dataset and is initialized to zero. If at any time during learning a sample is misclassified, the index of that sample in the vector is set to 1, which signifies remembering that sample for the rest of the run. The learning rule is

```
if remember[index] == 1: #do backprop (2)
```

Neither rule requires hyperparameter tuning and both are very easily added to existing code.

Intuition from a simple task

We first illustrate the lazy learning rules with a simple, linearly separable classification task of two data clouds using a small learning rate (0.001), Fig.1. Given our biological perspective, we make no use of data normalization, data augmentation, momentum, optimizers or batching as these would raise questions in terms of the biological implementation. While more work is needed, preliminary investigations indicate that such modifications do not negate lazy learning’s benefits.

We track the weight evolution starting from 3 different initial conditions on both the loss and accuracy surface, Fig.1a+b. Starting from initial condition 1, on a coarse level standard backprop follows a typical graded descent trajectory towards the minimum of the continuous loss surface. At a finer scale backprop updates irrespective of whether the sample is correctly classified, in the direction of that sample’s gradient, resulting in a windy path, Fig.1c. Its weight trajectory resembles a random walk with drift.

In contrast, pure lazy learning follows a path that on a coarse level perpendicularly ascends the terraced accuracy surface. It stops as soon as it reaches the 100% accuracy terrace (white, delineated region in Fig.1b). At a finer-scale, the weight trajectory of pure lazy learning gets progressively straighter during training as fewer samples contribute to learning. At the end of learning a single sample dictates the direction, resulting in a straight path Fig.1c. Lazy learning follows an intermediate trajectory. Trajectories from other initial conditions are consistent with this view. Initial condition 3 highlights that in contrast to backprop, lazy learning ascends the accuracy surface.

In lazy learning updates are less frequent, which translates to faster training time and fewer updates in computers (see Appendix). In light of our biological motivation, we would also like to know the metabolic

energy savings achieved by lazy learning. As the details of biological energy expenditure associated with learning are not well known, we use a previously proposed energy measure [Li and Van Rossum, 2020],

$$M = \sum_{i=1}^N \sum_{t=1}^T |w_i(t) - w_i(t-1)| \quad (3)$$

where N is equal to the total number of weights in the network and T is equal to the total number of learning steps. The metabolic energy equals the L1-norm of the weight trajectory.

Fig.1e compares this energy measure across the learning rules; pure lazy learning is most efficient, but lazy learning is almost as efficient. The amount of saving depends on the initial condition. For instance, from initial condition 2, the paths of all rules are identical on a coarse level and little saving can be obtained, aside from pure lazy learning stopping earlier.

To further explore the origin of the savings, Fig.1f colors the data such that darker color means that more energy was expended on that sample. In standard backprop the coloring is fairly uniform (there are still variations in the weight changes that are reflected in the energy). In contrast, pure lazy learning automatically concentrates on a few edge samples - similar to Support Vector Machines. For lazy learning, the separating line corresponding to initial condition 1, which rotates anti-clockwise during learning, is visible.

Initial condition 2 in particular highlights that pure lazy learning finds a fragile solution, where a small perturbation rightwards would lead to a large change in accuracy. This reduces the generalization performance of pure lazy learning, it is however rescued by the memory aspect of lazy learning.

Learning from the memory of mistakes: lazy learning of MNIST

Having shown the principle of lazy learning on simple tasks, we apply it the MNIST task. We train MLPs with a single hidden layer, implemented in the PyTorch framework [Paszke et al., 2019], initialized using the PyTorch default for linear layers (kaiming uniform) [He et al., 2015]. Networks contain 100, 200, 500, or 1000 hidden units and use the Leaky Rectified Linear Unit activation function with the default PyTorch negative slope parameter (0.01). The target is a one-hot vector. As cross entropy loss minimization requires more energy (not shown), we use a MSE cost function with a learning rate of 0.01. For biological realism we use stochastic gradient-descent, presenting a sample one at a time. We found that *pure* lazy learning quickly gives good training set performance, but finds fragile solutions that don't generalize as well as backprop (see Appendix).

Instead, lazy learning learns over all samples currently or previously misclassified and should overcome the shortcomings of pure lazy learning. To test this we track both the energy expended and testing accuracy as training proceeds and plot them against each other, Fig.2a. Reaching higher accuracy requires more energy and larger networks can reach better performance. Importantly, lazy learning always expends substantially less energy (note the logarithmic y-scale). Also the total number of parameter updates was always less (not shown).

To characterize the speed of lazy learning, we can not use the number of epochs. Compute time is dominated by parameter updating rather than by inference. While lazy learning requires the entire training dataset for the forward pass, it only uses previously misclassified samples for the backward pass. Thus while lazy learning typically takes more epochs, each epoch takes much less time because it skips updating for most samples. Instead we measure the computing time spent, Fig.2b (see Methods for details). The training times only include the time to perform feedforward and backward updates and exclude the fixed (small) overhead time for testing accuracies and losses. Training times are typically faster for lazy learning, while reaching similar accuracy levels, Fig.2b. Thus by memorizing the critical samples, lazy learning reduces training time by finding more robust solutions than pure lazy learning and maintains similar energy savings.

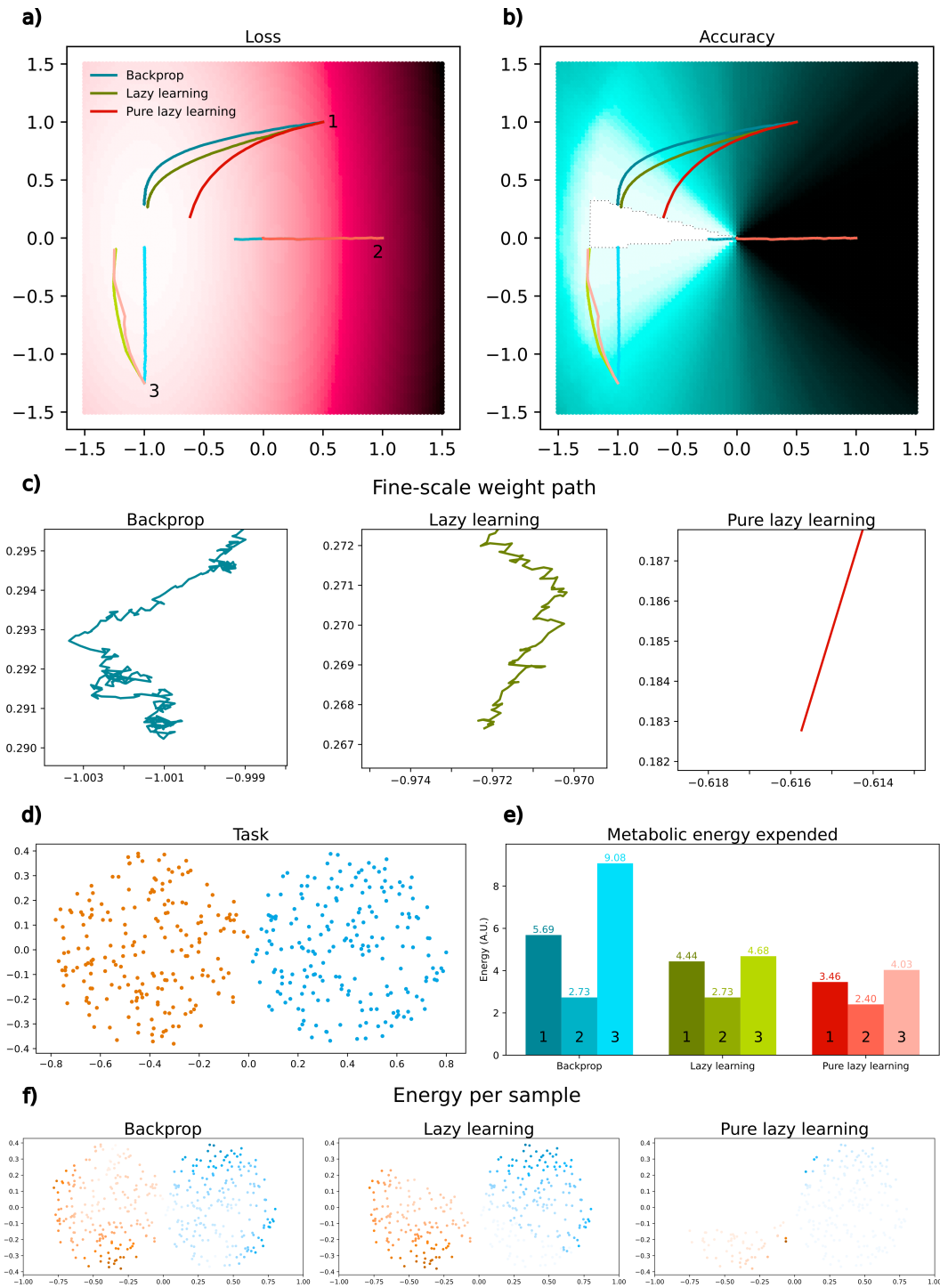


Figure 1: Lazy learning on a two-class problem in two dimensions (panel d). **a, b.** The loss surface (a) and accuracy surface (b) as function of the two synaptic weights, with the weight trajectories of 3 initial conditions for each learning rule: backprop (blue), lazy learning (green) and pure lazy learning (red). On a coarse scale backprop follows the gradient of the loss. Pure lazy learning instead follows the gradient of the accuracy surface. Lazy learning displays both backprop like characteristics, tracing backprop’s path exactly from initial condition 2, as well as pure lazy learning’s ability to trace the accuracy surface in initial condition 3. **c.** Zoom on the final part of the weight path of each learning rule for initial condition 1. **d.** Distribution of the inputs. The two classes are orange and blue. **e.** The energy consumed across initial conditions and learning rules. Regardless of initial conditions, pure lazy learning consumes the least energy. **f.** The energy expended on each sample for initial condition 1. Darker colors correspond to more energy.

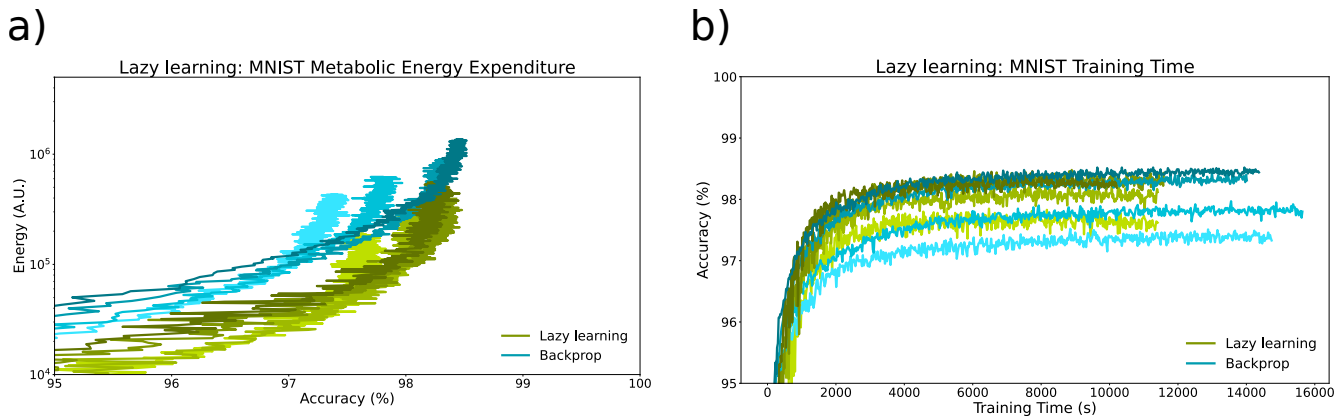


Figure 2: Lazy learning in the MNIST task. **a** - The metabolic energy expenditure of lazy learning for network sizes 100 (light), 200, 500, 1000 (dark) hidden units. Lazy learning saves energy while maintaining the generalization performance of backprop. **b** - Training times of lazy and backprop networks. In particular in small networks, lazy learning reaches similar performance in less time.

Lazy learning of Extended MNIST

We hypothesized that with more data, lazy learning could perform even better since the directions it takes on the loss surface are dictated by misclassified samples. In larger datasets there are more informative samples to the true direction of a 100% accuracy terrace. To test this, we repeat the experiment on the EMNIST digits task which is identical to MNIST, but it contains 240,000 training samples and 40,000 testing samples ($4\times$ the size of MNIST).

Compared to backprop, lazy learning improves accuracy over backprop, particularly in smaller networks, Fig.3a. It has been suggested that as network size increases, the number of local minima increases and further these minimas are no worse than the global minima [Kawaguchi et al., 2018]. We conjecture this is because the loss surface approximates the accuracy surface and the loss surfaces of larger networks are better able to approximate the accuracy surface. However, small lazy networks can approximate the accuracy surface better than small backprop networks.

Moreover, lazy learning also significantly improves the training time across all network sizes. In particular, the 1000 unit backprop network achieves a maximum testing accuracy of 99.165% within the 50 epochs, the lazy network reaches the same accuracy $7.6\times$ faster. When tasked on EMNIST, lazy learning saves even more energy in comparison to the savings of MNIST, Fig.3b. The more samples in a dataset, the more random of a walk backprop takes in weight space and therefore the more energy it consumes.

Lazy learning saves proportionally more energy by training on proportionally less samples in EMNIST, compared to MNIST, Fig.3c. Interestingly, in absolute numbers lazy learning memorizes twice as much samples from a four times larger dataset. It uses these extra samples to improve performance over MNIST. The training time speed up is roughly proportional to the fraction of memorized samples and the relative gain in generalization these samples provide. It would be of interest to understand the general relation between the dataset size, task complexity, and the number of samples used by lazy learning.

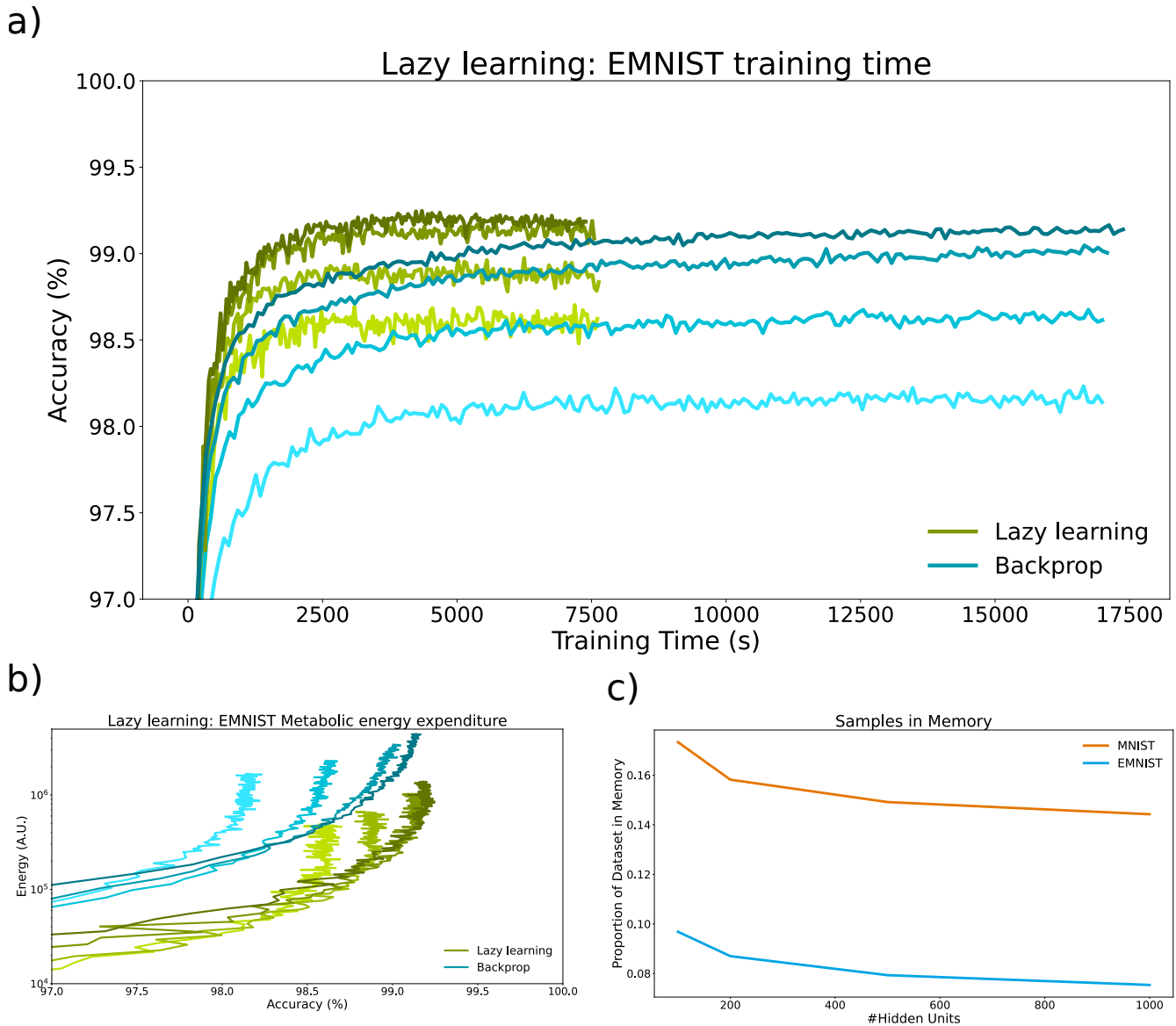


Figure 3: Lazy learning in the extended MNIST task. **a** - The training times of lazy and backprop networks for sizes 100 (light), 200, 500, 1000 (dark) hidden units. To train over the 50 epochs, lazy learning takes less time than backprop networks because they perform synaptic updates on less samples. Note: this training time excludes time taken to pass over the test set since this is a fixed overhead. **b** - The metabolic energy expenditure of lazy learning for network sizes (100, 200, 500, 1000), darker colors indicate larger networks. Lazy learning is able to both save energy and maintain the performance of backprop. **c** - The proportion of samples memorized by lazy learning. When trained on MNIST they memorize at most 17% of the dataset, when trained on EMNIST, they memorize at most 10% of the dataset. This represents the source of the training time improvements.

Discussion

Learning is a costly process, both for biology and computer hardware. To reduce this cost lazy learning draws inspiration from the observation that humans pay more attention to errors when learning. For AI practitioners, lazy learning can lead to faster, energy efficient and more performant synaptic plasticity rules. Code complexity and memory overhead are minimal. We foresee various applications. Lazy learning

acts as a ‘plug-and-play’ method of self-paced curriculum learning [Bengio et al., 2009]. As the network improves, misclassified samples will lie closer to decision boundaries or be an anomalous representation of the class. It has been shown that a dataset can be reduced to the number of repeatedly misclassified samples [Toneva et al., 2019]. Similarly, the set of samples lazy learning uses can be used as a coreset dataset. Here, the loss minima of these samples correspond to the accuracy terrace solution found by the lazy network.

Since lazy learning only learns on mistakes it may also be useful for class imbalanced datasets [Johnson and Khoshgoftaar, 2019], since imbalanced classes could be classified incorrectly more often. In this case, lazy learning would naturally balance the dataset. Lazy learning also regularizes the weights which could improve generalization performance.

Lazy learning could serve a unique purpose in Difference Target Propagation [Lee et al., 2014] and Direct-Difference Target Propagation [Meulemans et al., 2020]. These have been proposed as a biological implementation of backprop, with separate forward and backward weights. The backward weights attempt to recreate the hidden activities achieved by the forward weights, effectively inverting the higher unit activities to the lower unit activities. Here, lazy learning could also signal separate forward and backward synaptic updates. When the network is incorrect, both forward and backward updates occur. However, when the network is correct, only backward updates occur, which may help stabilize the backward inversions.

Recently, there is evidence that Transformer networks store the weights of a smaller neural network within them and perform gradient-descent from a sequence of presented examples in the input without updating its own weights [Akyürek et al., 2022]. Furthermore, it has been proposed that the hippocampus of the brain mimics the representations found by that of a transformer [Whittington et al., 2021]. Lazy learning effectively memorizes a small subset of a much larger dataset through an attention mechanism. Taken together with the results presented here, this suggests the brain may learn from memorized examples within the hippocampus and that this is energy efficient.

Lazy learning is also of importance for neuroscientists. The hippocampus has long been suggested as a temporary buffer to allow for offline learning. Lazy learning suggests that the content of the hippocampus is carefully titrated. On one hand, only events that evoke an error signal need to be stored. On the other hand, the result of pure lazy learning show that such events should be kept in memory, even after they do not evoke an error anymore.

Though neural networks train on millions of examples within a dataset, we show this training can be reduced with an attentional mechanism that focusses misclassified samples.

Methods

For the simple task, a simple, single-layer perceptron consisting of 2 weights, w_i , is used. It takes two inputs, x_i , that represent the cartesian vector of a data point. The task is to determine a linearly separable binary decision, between two circles as shown in Fig.1d. We used a MSE loss function between target and $h = \mathbf{w} \cdot \mathbf{x}$, and classify using $y = H(h)$, where $H()$ is the Heaviside function.

The networks are trained on the MNIST dataset [Lecun et al., 1998] which consists of 28×28 grey-scale pixel images of handwritten digits 0 - 9. There are 60,000 training samples which are presented in a random order and 10,000 testing images. Every 5,000 training samples, various metrics including loss, accuracy and energy are measured on the test and training set. All lazy learning simulations ran for 50 epochs.

For timing purposes, all compared simulations of backprop and lazy learning networks run on the same machine. This was either an Nvidia RTX A6000, Threadripper 3970X, 128GB of RAM or an Nvidia GTX 1080Ti, Ryzen 1800X and 32GB of RAM. Times were obtained using the `timeit` python library. In the case of lazy learning, the MNIST simulations ran using the Nvidia GTX 1080Ti setup on the GPU. We also use the EMNIST digits dataset [Cohen et al., 2017], which used the Nvidia RTX A6000 setup on the GPU and measured metrics every 60,000 samples. Depending on the hardware and various

hyperparameter setups, different training times may be obtained. If all samples are misclassified then lazy learning will approach an identical solution to backprop. Generally lazy learning is faster than backprop but will otherwise be just as fast.

For pure lazy learning on MNIST (see Appendix) we set a cut-off accuracy of 97.5% or alternatively, gave up at 50 epochs. These simulations used the Nvidia A6000 setup on the GPU.

Acknowledgments

It is a pleasure to thank Claudia Danielmeier and Josh Khoo for discussion. This research was supported by a grant from NVIDIA and some simulations utilized an NVIDIA RTX A6000.

Appendix

Pure lazy learning in complex tasks

Here we show the results of pure lazy learning on the MNIST task.

In addition to the energy listed in Eq. 3, we also define a minimal energy. The lowest energy solution would follow a straight path in weight space to the solution closest to the initial condition. However, since finding the optimal solution is intractable, we can designate a candidate ideal solution to be the L1-norm of the weights at a given point in time. This would assume the network traced a straight line to a given level of performance.

$$M_{\min} = \sum_{i=1}^N |w_i(T) - w_i(0)| \quad (4)$$

While similar to Eq. 3, it only uses the initial and final weights. Eq. 3 and Eq. 4 define an energetic inefficiency,

$$\text{Ineff} = \frac{M}{M_{\min}} \quad (5)$$

Conceptually, it describes how tortuous the weight path is compared to the straight path. Consequently there are two methods to save energy: solutions that have a smaller norm (relative to the initial conditions) and less windy paths.

Pure lazy learning saves significant energy over backprop for all network sizes and for all accuracies attained by pure lazy learning, Fig.4. The energy savings fall between 2.2-4.5x that of backprop and this variance is likely a result of the potentially fragile solutions pure lazy learning finds, as demonstrated in Fig.1. Since the minimum energy of pure lazy learning is less than backprop, the final L1-norm of the weights is smaller and therefore saves energy, Fig.4c. It also shows that pure lazy learning has a regularizing effect on the weights.

The inefficiency of lazy learning is also less than that of backprop, indicating the weights follow a less tortuous path, Fig.4d. Pure lazy learning thus saves energy by following a less tortuous trajectory to a closer solution.

One shortcoming of pure lazy Learning is the maximal performance it can reach. While backprop networks were able to achieve the 97.5% threshold testing accuracy within 50 epochs, pure lazy learning networks were not. We highlight this performance in Fig.5, which depicts the test and training loss alongside the test and training accuracy for both backprop and pure lazy learning (up to 30 epochs). Significantly, pure lazy learning reach a loss 10-fold larger than that of backprop networks. This occurs since pure lazy learning can tolerate low confidence but correct neuron outputs to a given sample. They end up finding discrete, sharp terraces where tiny weight perturbations can cause large differences in accuracy.

The loss demonstrates that for MNIST, some 95% accuracy terraces lie around a mean-squared error loss of 0.07 and is significantly higher than the loss backprop requires to reach the same accuracy. This evinces how high some of the 95% terraces might be on the loss surface.

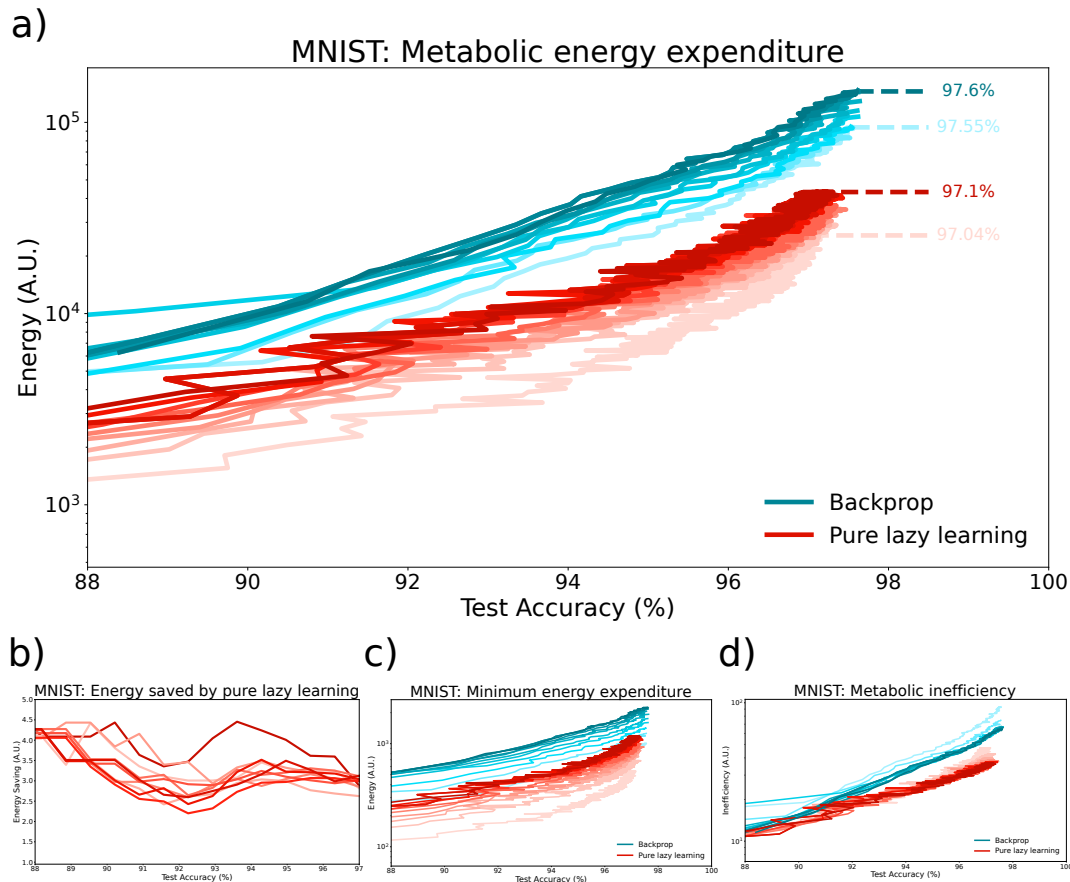


Figure 4: Pure lazy learning in the MNIST task. **a** - The metabolic energy consumed by backprop and lazy learning networks for network sizes 100, 200, ..., 1000. Increasing darkness represents increasing network size. The testing accuracy reached by the end of training for the smallest and largest networks under each learning regime are indicated. Training stops after 50 epochs or reaching 97.5% testing accuracy. For all network sizes and for all accuracies achieved by lazy learning, lazy learning saves considerable metabolic energy over backprop but is unable to achieve the same performance. **b** - The energy saved by lazy learning for all achieved accuracies. Lazy learning can save between 2.2-4.5x of the metabolic energy expended by backprop. **c** - Lazy learning achieves a lower minimum energy in comparison to the equivalent backprop networks, indicating that lazy learning networks find a shorter weight path to attain a given accuracy. **d** - The energy inefficiency of backprop and lazy learning networks. Lazy learning is more efficient than backprop, indicating that lazy learning follows a less windy path to a given accuracy.

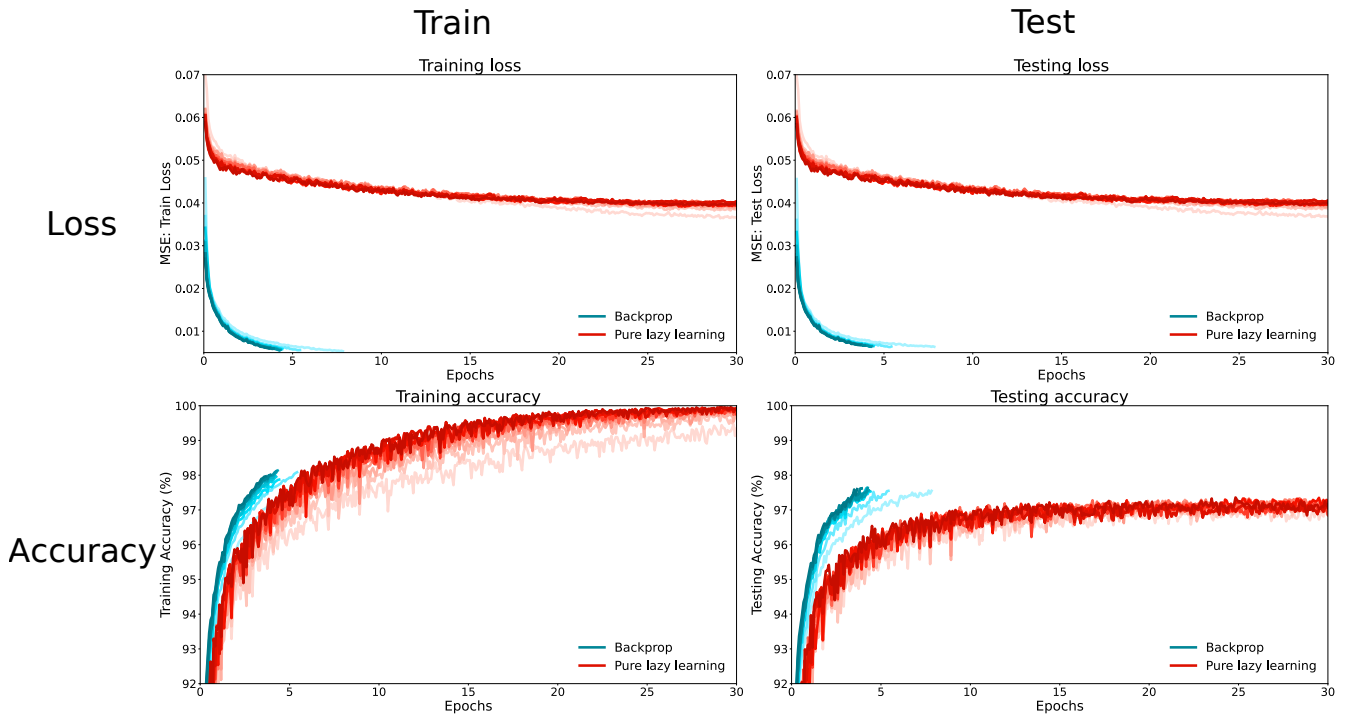


Figure 5: The test and training, loss and accuracy metrics of pure lazy learning for network sizes 100, 200, ..., 1000. Pure lazy learning has a significantly higher loss than backprop in both testing and training (top). This stems from its ability to allow low confidence but correct classifications. Also significant is lazy learning’s training accuracy, which continues to improve towards 100%, but the testing accuracy levels off. Since the testing accuracy levels off, as opposed to dropping, this suggests lazy learning is not necessarily overfitting. In all cases, lazy learning is unable to achieve the threshold testing accuracy of 97.5%, while equivalent backprop networks reach this within 10 epochs. This performance marks a deficit to pure lazy learning.

While the simple example shown in Fig.1 provides intuition to distinguish between the terraced accuracy surface and the loss surface. In more complicated tasks, there are many different samples with different varying tolerances for their accuracy terrace and this results in many sharp fragile solutions at the intersection of these terraces. Regardless, in a more complex task, pure lazy learning is able to proficiently locate high test accuracy terraces using less energy in comparison to backprop.

The more correct pure lazy learning is, the less samples are evaluated in an epoch. One incorrectly labelled sample could dominate the gradients prescribed by gradient-descent and under the pure lazy learning rule this moves pure lazy learning outside of better test accuracy terrace. For example, index 59915 of MNIST labels a ‘7’ looking digit as a ‘4’ and would guide lazy learning towards a fragile accuracy terrace representative of the training set accuracy terrace instead of the global terrace.

Though pure lazy learning is unable to achieve the 97.5% testing accuracy, the training accuracy continues to improve. It is important to note the test accuracy does not begin to fall which would indicate overfitting. We infer that pure lazy learning continues to seek the 100% accuracy terrace of the training set and this remains on the outskirts of the test set when seeking minimal energy consumption from a particular initial condition.

Number of synaptic updates and CPU time

The energy measure used in the main results is useful as biological model and to indicate the tortuousness of a weight path. However, it is not as useful for computational efficiency. An alternative energy measure

is the number of synaptic updates. Under this energy measure, lazy learning also saves energy; even more than using the biological measure used in the main text.

In the main results, networks are trained using GPUs. To show the generality of the result we here use the CPU time taken. On a CPU, small networks train faster, but large networks take significantly longer to train.

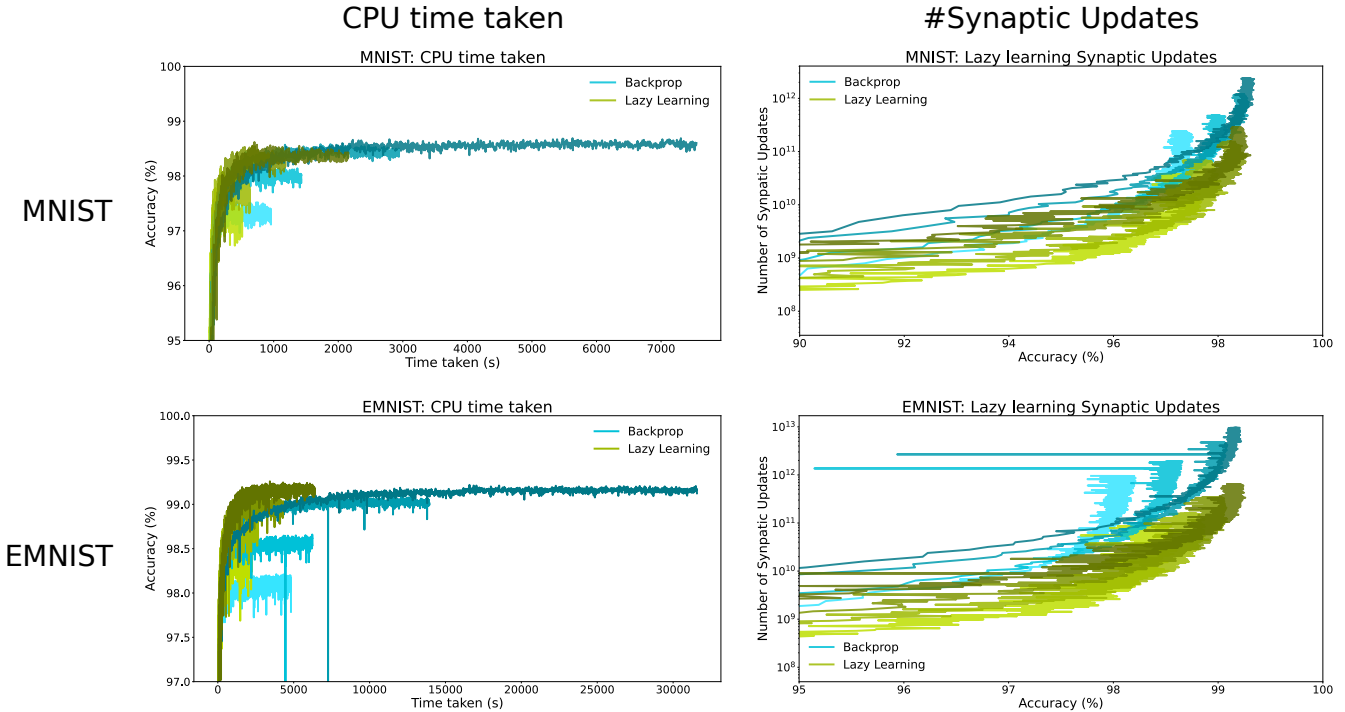


Figure 6: The CPU time taken and number of synaptic updates energy of backprop and lazy learning for network sizes (100, 200, 500, 1000). On a CPU, larger networks take longer to train over the 50 epochs. Since lazy networks update less often, it takes significantly less time to train. Lazy learning uses around 10-fold less synaptic updates than backprop to reach an equivalent accuracy.

References

- P. K. Agarwal, S. Har-Peled, K. R. Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52(1):1–30, 2005.
- E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou. What learning algorithm is in-context learning? investigations with linear models, 2022. URL <https://arxiv.org/abs/2211.15661>.
- Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009*, pages 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL <https://doi.org/10.1145/1553374.1553380>.
- G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters, 2017. URL <https://arxiv.org/abs/1702.05373>.
- C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia. Selection via proxy: Efficient data selection for deep learning, 2020.

- E. R. A. de Bruijn, R. B. Mars, and R. Hester. Processing of performance errors predicts memory formation: Enhanced feedback-related negativities for corrected versus repeated errors in an associative learning paradigm. *European Journal of Neuroscience*, 51(3):881–890, 2020. doi: <https://doi.org/10.1111/ejn.14566>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/ejn.14566>.
- M. Girard, J. Jiang, and M. C. van Rossum. Estimating the energy requirements for long term memory formation. *arxiv*, page 2301.09565, 2023.
- K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL <https://arxiv.org/abs/1502.01852>.
- C. B. Holroyd and M. G. H. Coles. The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity. *Psychol Rev*, 109(4):679–709, Oct. 2002.
- J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, Mar 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL <https://doi.org/10.1186/s40537-019-0192-5>.
- Ç. Kalfaoğlu, T. Stafford, and E. Milne. Frontal theta band oscillations predict error correction and posterror slowing in typing. *J Exp Psychol Hum Percept Perform*, 44(1):69–88, Apr. 2017.
- K. Kawaguchi, J. Huang, and L. P. Kaelbling. Effect of depth and width on local minima in deep learning. *CoRR*, abs/1811.08150, 2018. URL <http://arxiv.org/abs/1811.08150>.
- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 0028-0836. doi: 10.1038/nature14539. Insight.
- D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation, 2014. URL <https://arxiv.org/abs/1412.7525>.
- H. Li and M. C. Van Rossum. Energy efficient synaptic plasticity. *Elife*, 9:e50804, 2020.
- F. Mery and T. J. Kawecki. A cost of long-term memory in drosophila. *Science*, 308(5725):1148, May 2005. doi: 10.1126/science.1111331.
- A. Meulemans, F. S. Carzaniga, J. A. Suykens, J. a. Sacramento, and B. F. Grewe. A theoretical framework for target propagation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- C. M. O’Carroll and R. G. Morris. Heterosynaptic co-activation of glutamatergic and dopaminergic afferents is required to induce persistent long-term potentiation. *Neuropharmacology*, 47(3):324–332, 2004. ISSN 0028-3908. doi: <https://doi.org/10.1016/j.neuropharm.2004.04.005>. URL <https://www.sciencedirect.com/science/article/pii/S0028390804001108>.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

- M. Paul, S. Ganguli, and G. K. Dziugaite. Deep learning on a data diet: Finding important examples early in training, 2021.
- P.-Y. Plaçais and T. Preat. To favor survival under food shortage, the brain disables costly memory. *Science*, 339(6118):440–442, 2013. ISSN 0036-8075. doi: 10.1126/science.1226018.
- P.-Y. Plaçais, È. de Tredern, L. Scheunemann, S. Trannoy, V. Goguel, K.-A. Han, G. Isabel, and T. Preat. Upregulated energy metabolism in the drosophila mushroom body is the trigger for long-term memory. *Nature Communications*, 8(15510), June 2017. doi: 10.1038/ncomms15510.
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- P. Rozin and E. B. Royzman. Negativity bias, negativity dominance, and contagion. *Personality and Social Psychology Review*, 5(4):296–320, 2001. doi: 10.1207/S15327957PSPR0504_2.
- M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon. An empirical study of example forgetting during deep neural network learning, 2019.
- J. C. R. Whittington, J. Warren, and T. E. J. Behrens. Relating transformers to models and neural representations of the hippocampal formation, 2021. URL <https://arxiv.org/abs/2112.04035>.