The Role Of Biology In Deep Learning

Robert Bain
Oregon State University
bainro@oregonstate.edu

May 26, 2022

Abstract

Artificial neural networks took a lot of inspiration from their biological counterparts in becoming our best machine perceptual systems. This work summarizes some of that history and incorporates modern theoretical neuroscience into experiments with artificial neural networks from the field of deep learning. Specifically, iterative magnitude pruning is used to train sparsely connected networks with 33x fewer weights without loss in performance. These are used to test and ultimately reject the hypothesis that weight sparsity alone improves image noise robustness. Recent work mitigated catastrophic forgetting using weight sparsity, activation sparsity, and active dendrite modeling. This paper replicates those findings, and extends the method to train convolutional neural networks on a more challenging continual learning task. The code is publicly available here.

1. Introduction

Sparsity has proven useful numerous times in deep learning (DL) [39, 61, 56, 13, 14, 3, 20]. Unfortunately though, typical hardware does not effectively benefit from sparse weights (e.g. GPUs). FPGAs and neuromorphic hardware do however better leverage sparsity [27, 44, 9, 12]. The hardware lottery hypothesis is worth considering [24]. It is the idea that worse algorithms in AI can become popular because the current hardware favors them more, and not because they are better long term. This is supported by the fact that brains are sparse in both their connections and outputs.

There is a lot of evidence that dendrites (see Fig. 1) are nonlinear filters of voltage signals as they travel to the cell body (i.e. soma) [4, 59, 10], but these active dendrite properties are not typically modeled in deep neural networks (DNNs). [20] recently modeled some of these effects in what they called an *Active Dendrites Network* (ADN). ADNs do well at PermutedMNIST, a continual learning task, by mitigating the effects of catastrophic forgetting. These were feedforward networks with few layers trained on a dataset that allowed for no transfer of knowledge between tasks. This paper extends that work to include convolutional neural networks (CNNs) and tests it on a more challenging continual learning dataset that theoretically enables some knowledge transfer [38].

It is worth reflecting on the history of artificial intelligence (AI) research to study what has worked the best in the past. This work attempts to follow a rather long tradition of using data from biological sciences to create better AI. Part of that history is chronicled in the subsequent *Section 2.1.*

2. Background

2.1. Deep Learning Background

McCulloch and Pitts [43] created the point neuron model most commonly used in deep learning today[1]. They also proposed that biological neural networks (BNNs) were equivalent to binary first order logic and could be implemented in electronic circuits. [55] improved on this by allowing unequal (ie weighted) connection strengths between neurons. Rosenblatt's perceptron [50] tested Hebb's idea that modifying such connection strengths could lead to memory and learning, to great success. Minsky & Papert (1969) are cited as causing the subsequent AI winter by writing that perceptrons could not even learn to do a trivial XOR on 2 inputs. Solutions were already known at the time however [47], by including multiple layers of neurons and non-linear activation functions, but the damage to the perceptron's reputation had already been done by the point Minsky and Papert became aware of such solutions.

Fukushima created convolutional layers with multiple input and output feature maps, downsampling, and weight sharing (ensuring translational invariance) with the neocognitron in 1980 [16].

Lecun's architecture years later was largely the same, but with back-propagation of errors (i.e. backprop) to change weights instead of self-supervised learning [35, 34]. Lecun cites Rumelhart's T-C problem paper for weight sharing [52], which in turn cites Fukushima's neocognitron for using an analogous solution.

All of the aforementioned scenarios involved low data and compute regimes. Lecun was dealing with a dataset of

[1]Lapique is often incorrectly cited, but did nearly invent the modern leaky integrate and fire (LI&F) neuron model [7, 8]. The seemingly only missing component would be a variable resistor to model the cell membrane gates as they open and close.

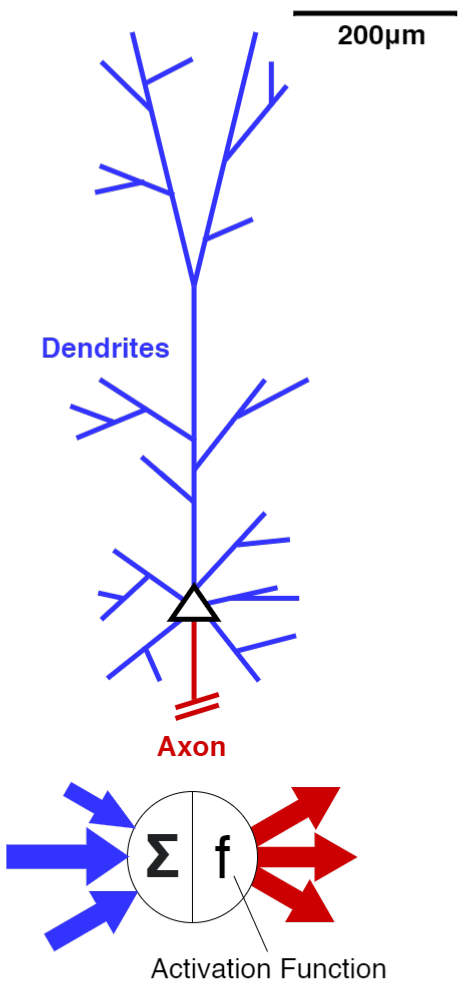

Figure 1: **Top**: Diagram of a human L5 pyramidal neuron. Dendrites receive information from other neurons in the form of voltage spikes. They transform the spikes while relaying them to where the cell body meets the axon. If the voltage across that bit of membrane sums to larger than a certain threshold value, a spike (i.e. action potential (AP)) is generated, and becomes input to other neurons. The axon is truncated as they are usually long and elaborate. **Bottom**: A point neuron, typically used in deep learning, abstracts away all physical structure of a real neuron. The influence a dendritic input has on an action potential being generated is modeled as a multiplicative weight. The dynamics at the soma are represented by a summation of weighted inputs, and an activation function. The latter is most often ReLU, which outputs 0 if the summed input is below 0. This in part represents the threshold of AP firing in real neurons.

only 480 tiny 16x16 pixel grayscale images. Weight sharing was demonstrated to be compute efficient since less parameters had to be learned, and useful to avoid overfitting small datasets.

Fukushima took direct inspiration from Hubel and Weisel's work on the striate cortex (i.e. V1) of cats[2] and monkeys [25, 26]. The neocognitron's S & C cells correspond to simple & complex cells from that work. The nomenclature of *receptive fields* is also borrowed from the same literature. That term is still used heavily today, but many artificial neural network (ANN) practitioners do not realize it comes from investigations into the eyes and brains of animals.

ReLU came about somewhere between Rosenblatt's 1962 book "*Principles of Neurodynamics*,"[47] and Fukushima's 1968 paper about "analog threshold units" [15]. The subsections of Ch. 10 [47] involve transfer functions that each have components of ReLU as used today (e.g. thresholding, non-linearity, and monotonically increasing functions). The neocognitron paper's figure 3 includes the exact input-output response plot we typically see today when learning about ReLU.

The deep CNNs used today bear a great deal of similarity to Fukushima's Neocognitron. Perhaps the biggest software difference is the greater depth, which is attainable through residual (i.e. skip) connections [23]. Alexnet [32] is often credited as having started the latest AI summer in 2012 by introducing ReLU activation functions and leveraging the massively parallel architecture of GPUs. ReLU is quick to calculate in both the forward and backward passes, and blocks certain neurons from contributing to the next layers' neuron's firing. This reduces training time, and induces sparser weight updates respectively. As noted earlier, ReLU was already applied in ANNs during the 1960s, which leaves just hardware changes to credit for the latest wave of interest in ANNs. While the brain is massively parallel like GPUs, the brain is sparse in its connections, connection updates, and activations (i.e. the percentage of neurons sending action potentials at any given moment). GPUs currently do not enable speedups when doing large sparse matrix calculations like those required for DNNs. The performance benefits are usually only double, even in cases of rather extreme weight and activation sparsity.

2.2. Active Dendrites Background

Dendrites have been modeled as passive voltage attenuators for a long time. The majority of excitatory inputs (85%) on L5 pyramidal cells are on dendrites frequently too far away from the cell body to significantly contribute to axonal APs without active dendritic properties [33, 41, 59, 10, 29]. These properties include at least sodium (Na+), potassium (K+), calcium (Ca++), NMDA, and back-propagating action potential activated Ca++ (BAC) spikes.

A lot of discoveries have been been made since the 1990s about NMDA spikes in the dendrites, largely in part due to better realtime imaging [4]. The NMDA channel membrane protein that allows such spikes to occur requires both glutamate molecules to bind the receptor, and high enough voltage levels to evacuate magnesium from the channel's pore.

[2]They euthanized 40 cats. Bring that up in your AI ethics course.

Even with high transmembrane voltages, NMDA spikes do not spread beyond where glutamate is bound. This local type of spike is in contrast to the typical Na+ spikes that travel along axons just through voltage gradients alone. NMDA spikes occur primarily in the thin, distant dendrites, but not in the apical trunk (the black bit of the dendrites in Fig. 3). Dendritic branches typically have 100-400 spines. NMDA spikes are highly localized events that envelop just one small dendritic segment, $10 - 40$ μm in length [4]. As few as 8 individual inputs from other neurons can produce NMDA spikes if they are close enough to each other on the dendrite. NMDA spikes over larger dendritic surface areas require stronger or more concurrent inputs [59, 4].

Point neurons assume that the axonal trigger zone is the only place where spiking occurs. This assumption was made a long time ago, and is no longer assumed to be true by electrophysiologists. Voltage sensitive Na+, K+, and Ca++ channels are embedded in the membrane nearly everywhere, and assist in actively propagating inputs and dendritic spikes towards the soma, as well as axonal APs back into the dendritic arbor [40, 1, 63, 60]. The latter signals are called backwards propagating action potentials, or bAPs (no relation to backprop in DL). The filtering of bAPs can be nonlinearly conditioned on previous neuron activity, as voltage levels can cause bAPs to be attenuated differently [36, 59].

NMDA spikes in the dendrites most proximal to the cell body of L5 pyramidal cells display rather binary voltage responses wrt input intensity, but the relationship between intensity and duration is positively linear [4]. Stronger inputs can thus lead to more prolonged NMDA spikes, which is thought to be useful for integrating information from multiple modalities (i.e. sensor fusion).

It is known today that the apical trunk of pyramidal neurons alone can do XOR [18] via active dendritic processing. Even relatively small compartments of a single biological neuron can compute XOR, where point neurons typically require 2 layers of processing to perform XOR.

Small dendritic spikes often need to sum with other inputs to evoke spiking at the soma, or to make it more likely to fire. Dendritic spike interaction also enables complicated coincidence detection [59, 4].

Many researchers have highlighted the importance of active dendrites in updating connection strengths between biological neurons [1, 4, 59, 10]. Blocking bAPs through chemical means can prevent local updates rules from occurring within pyramidal cells [58, 68].

2.3. Lottery Ticket Hypothesis Background

The lottery ticket hypothesis (LTH) [14] states that densely connected DNNs contain sparse sub-networks that can exceed the test accuracy of the original network after training on at most the same number of instances. These sparse networks are called winning lottery tickets (WLTs) and do much better than the average random sub-network.

Algorithm 1: Iterative Magnitude Pruning

```
1 Train model to convergence
2 Set x% of the smallest magnitude, non-zero weights
  per layer to 0.
3 Set non-zero weights to their original values before
  training.
4 Train model to convergence again
5 Repeat steps 2 through 4 until the desired sparsity is
  met.
```

[14] used iterative magnitude pruning (see Algorithm 1) to create WLTs. They have been discovered in many architectures [72, 14, 69, 71], and in different task domains like reinforcement learning (RL) and natural language processing [69]. Finding WLTs with more than 50% sparsity appears trivial, but what pragmatic benefit do they offer to practitioners? Finding these winning tickets still requires a lot of compute and fine-tuning for larger CNNs.

[72] found that even better than IMP is keeping weights whose values change by the greatest magnitude, instead of just keeping the largest magnitude weights. [14] unintentionally foreshadowed this when noting in their Appendix F that winning tickets' weights move further than other weights. [72] also showed that re-initializing the weights is not as important as retaining the original signs of the weights, lending even more evidence to the idea that re-initializing to the original values is not vital to finding WLTs.

[37] revised some of the experiments from [14] on pruning larger CNNs while training on ImageNet. This domain usually requires a more exhaustive hyperparameter search to discover WLTs, but when discovered the results tend to more impressive [14]. [37] demonstrate that using the more de facto training regime with a larger learning rate and momentum SGD instead of Adam produces better results than even the winning tickets, making LTH even less practical.

As originally noted by [14] their IMP method produces sparse networks in such a way that GPUs do not benefit. The sparsity allows more compression, but not faster inference. In part, this is due to the asynchronous checkpoints that occur while GPUs process matrix calculations in parallel. Multiplications by 0 in a certain batch might be faster (i.e. fewer clock cycles) but the slowest multiplication in that batch of operations has to finish before more calculations can be queued.

3. Experiments

This code was adapted to produce WLTs on MNIST and CIFAR10 (see Algorithm 1). Both datasets were trained on with fully connected (FC) networks, and the former also had a LeNet-5 trained on it. The best network from a max of 100 rounds of training was kept, then 12.5% of the remaining weights were pruned. This train-prune loop occurred until only 1.1% of the original weights were left being non-zero.

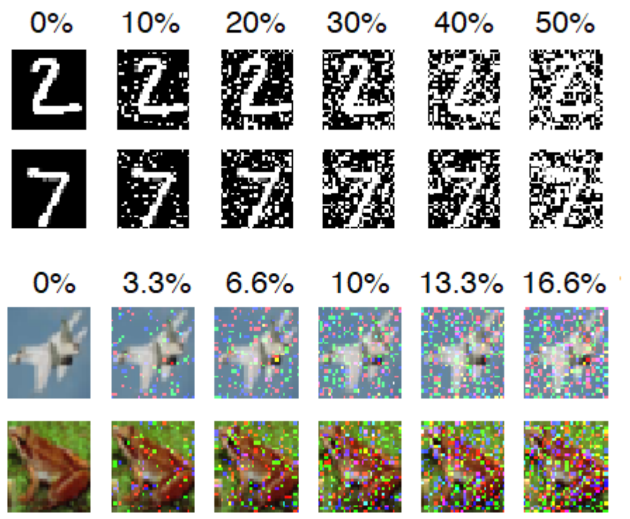

Figure 2: Examples of the images used to test the effects of weight sparsity on noise robustness. The probability of injecting noise was uniform across the image, but the noise was created by setting a pixel's value to near full intensity (specifically 2 standard deviations above the dataset's mean). **Top**: Examples from MNIST. **Bottom**: Examples from CIFAR10 required less noise for me to have difficulty classifying them.

The same networks were randomly pruned as well. The resulting WLTs and random tickets were then tested for noise robustness by adding variable amounts of noise to the test set images. Examples from that process can be visualized in Fig. 6. All trends reported are an average over 8 training runs with different seeds.

Next, I set out to replicate some of the continual learning results on PermutedMNIST reported by [20]. PermutedMNIST [19] is a continual learning benchmark that takes normal MNIST and applies a random shuffling of the pixels (i.e. permutes the pixels), but does the same shuffling across each task. Thus many tasks can be made from just MNIST, but there is no semantic information that can be leveraged across individual tasks.

$$y = \left(\boldsymbol{w}^\top \boldsymbol{x} + b\right) \times \sigma\left(\max_j \boldsymbol{u}_j^\top \boldsymbol{c}\right) \tag{1}$$

Eq. (1) details how the active dendrite model calculates a neuron's output (y) using context ($\boldsymbol{c}$) and feedforward inputs ($\boldsymbol{x}$). There are j sets of dendritic segments, each with a set of weights ($\boldsymbol{u}$). Every segment takes a weighted sum of the context vector. The largest absolute value among segments is passed through a sigmoid, and is then multiplied by the feedforward output of the neuron. This is referred to as context gating. See Fig. 3 for a visualization of this equation.

Their ADNs are detailed in Fig. 4, which are built upon layers of active dendrite neurons as shown in Fig. 3. They used the element-wise average image as the context signal to each ADN neuron. In [28] they used 1-hot encoding instead, which we tried out as well to see the effects of fewer context weights. Every task gets its own unique 1-hot encoded vector in this setting. To further explore the effect of more context weights, we also trained the 1-hot ADN architectures with a context vector full of 0s. In this setting there is effectively no task-specific information given as context, just a single multiplicative gating bias for certain layers' neurons, each of which is calculated from multi-layer perceptrons (MLPs) given 0s as input. For the 1-hot and 0s context vectors, we used the hyperparameters [20] found from their grid searches.

To induce activation sparsity they used k-winners take all (kWTA), which can be seen as a ReLU with a variable threshold [42]. The threshold is set at each inference such that k neurons are always active.

$$k\text{WTA}\left(y_i\right) = \begin{cases} y_i, & \text{if } y_i \text{ is one of the largest k values in } y \\ 0, & \text{otherwise} \end{cases}$$

[20] did not track the test accuracy of the latest trained task while doing continual learning, only the overall average of all tasks trained up to that point. I thought this worth doing to see if perhaps the larger network capacity has an advantage the whole time, as might be the case if it does better on all individual tasks.

All ADNs tested modeled 10 dendritic segments per active dendrite neuron. The number of inputs to each of those segments is determined by the size of the context vector, which is the same length for the 1-hot and 0s vector case (i.e. the number of tasks being trained on). All ADN results came from an average over 5 different seeds.

SplitCIFAR100, a new dataset, was made to test how sparse CNNs and ADNs generalize to more difficult tasks than they were tested in [20]. It is a variant of SplitCIFAR from [70], but only uses CIFAR100 and not CIFAR10. CIFAR100 has 100 classes typically, but here those classes are randomly sampled without replacement to create 10 sequential 10-way classification tasks. To extend ADNs to include CNNs the active dendrite modeling as multiplicative context gating had to be added to 2D convolutional layers as well. Considering that weights are shared across neurons in such layers typically, the dendritic/context weights for each neuron were assumed to be the same as well. This caused whole feature maps (i.e. channels) to be up or down-regulated at a time conditioned on the given context vector.

[20] tested their ADNs without the context gating, leaving just sparse networks. These nets had only activation sparsity however, not weight sparsity. I tested sparse networks as well, because [20] found that activation sparsity alone was better than dendritic segments alone.

Both the convolutional ADNs and sparse networks replaced ReLUs in Fig. 5 with kWTA activation functions, and fully connected layers with sparsely connected ones. The k-winners were taken across all output channels. The convo-

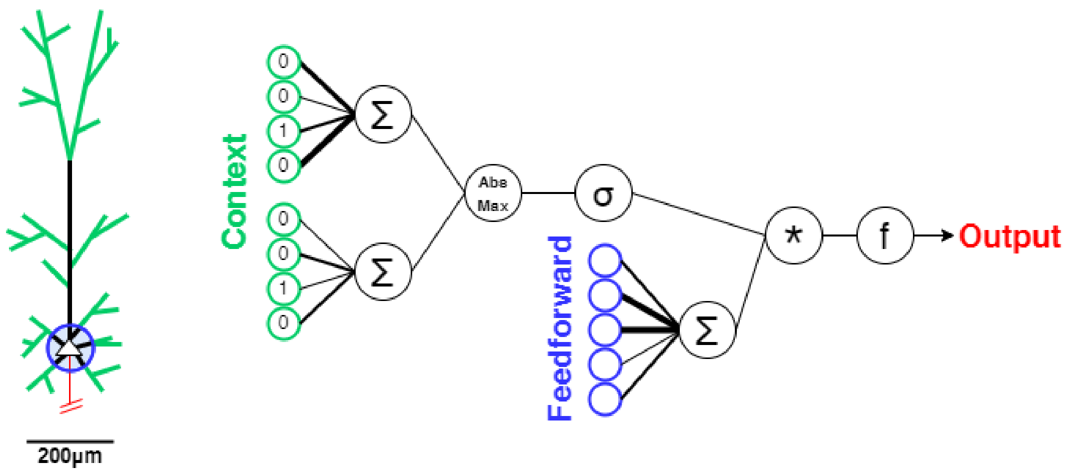

Figure 3: **Left**: Diagram of L5 pyramidal neuron highlighting dendrites that produce NMDA spikes in green. The blue circle encompasses the connections close enough to the cell body to have a large influence on producing APs without active dendritic effects. **Right**: A diagram of a single active dendrite neuron model. This particular one has 2 dendritic compartments. Each gets a copy of the context signal. As shown this would be a network trained on 4 consecutive tasks, the current being the 3rd, as indicated by the 1-hot encoding scheme. [20] used the task averaged input as context, leading to networks with many more weights. The absolute max is taken among dendritic segments, and then scaled between 0 and 1 using a sigmoid function. *f* stands for the chosen activation function. If it is kWTA, then the layer's other neuron's values at this point in the diagram will be taken into consideration as well.

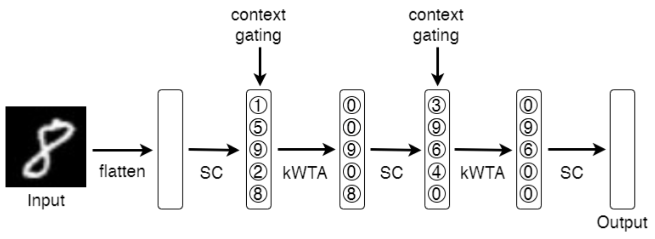

Figure 4: An ADN network architecture similar to those used in [20]. SC stands for sparsely connected feedforward layer, and kWTA stands for a *k-winners take all* activation function. Note that k=2 for the shown kWTA activation functions (i.e. 40% sparse).

lutional ADNs also had channel-wise context gating for the neurons in the convolutional layers.

Following from the results of [3, 20, 27], I constructed grid searches as shown in Appendix A. Their papers reported no impact of convolutional weight sparsity, so I did not test it further. The grid searches for the sparse networks and ADNs tested 12x as many hyperparameter sets relative to the grid search over the base CNN (see Fig. 5).

4. Results

4.1. Iterative Magnitude Pruning Results

Fig. 6 shows the test accuracy evolving with successive train-prune loops, and contrasts them to randomly pruned networks of the same sparsity. WLTs are attained until around 3% sparsity, and random tickets even test as well as the fully dense network until around 30%.[3]

Given this, WLTs of 3% and 30% sparsity, random tickets of 30% sparsity, and dense with 100% sparsity were tested for noise robustness by injecting uniform noise into all the test set images. The results in Fig. 7 show that there is no consensus as to which network does best. w30 does best on MNIST when using fully connected networks, and 2nd best on the other setups. Besides that, there is not much of a trend.

4.2. Active Dendrite Network Results

My experiments were able to replicate a subset of Numenta's [20]. Specifically, their 10 segment ADN network got 94.6% and 78.5% on the 10 and 100 sequential tasks of PermutedMNIST respectively. This work's ADN networks got 0.3% and 0.2% less respectively. [4] The performance difference between the 1-hot and average image ADNs was small for the 10 task setting. The ADNs with 0s context had the majority of improvements wrt the baseline 3 layer MLP. Perhaps doing a separate hyperparameter search on the 100 task setting would reduced the performance gap between the 3 ADNs even more.

The final results for SplitCIFAR100 are presented in Fig. 9.

[3]The middle graph's x-axis is slightly different. Do not know why, could be a trivially small rounding error at each pruning step as the first round of pruning removed 12.4% instead of 12.5%.

[4]They always tested on the same training dataset seed, including their hyperparameter sweep, and thus the same set and order of permutations. This most likely accounts for the minor performance differences.

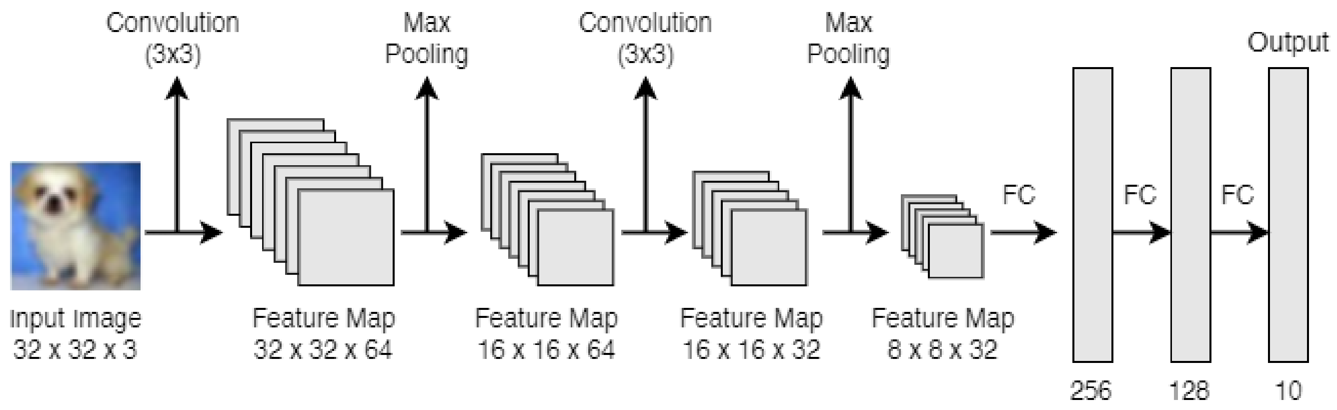

Figure 5: This is the base CNN used for experiments on SplitCIFAR100. Convolutional and fully connected layers are followed by the ReLU activation function for the base CNN shown, and kWTA for the ADNs and sparse networks. The ADNs had either task specific 1-hot encoded vectors, vectors full of 0s, or the task averaged input image flattened as a context signal at each convolutional and sparsely connected feedfoward layer. The hyperparameter grid search axes and resulting values are in *Appendix A*.

The trends are nearly overlapping each other, indicating the base CNN, sparse version, and convolutional ADN all perform similarly well.

5. Discussion

It was suggested by [3] that perhaps weight pruning techniques increase noise robustness. This work suggests that weight sparsity alone in FC and small CNNs does not improve noise tolerance in image classification scenarios. Perhaps the benefits [3] found were from activation sparsity alone, or from combination with weight sparsity. A seemingly easy way to test this would be to add a kWTA layer to the networks tested in this paper to see if noise robustness increases.

The epochs at each train-prune loop were not tracked during IMP, but the test accuracy rose very quickly at the beginning of all subsequent training phases (except for some of the more extremely pruned ones). I also played around with pruning earlier than convergence (e.g. after only 1 epoch of training), and it did seem to produce WLTs as well. This might be an easy way to reduce the amount of compute this method takes to work. In another paper, I explored the shape of the loss landscape using dimensionality reduction techniques. The shape of the loss wrt the weight/configuration space is fairly stable after 100 examples [5]. Perhaps this phenomena can be used to motivate when pruning should occur.

The results on PermutedMNIST indicate that the great majority of the benefits can be had from multiplicative gating alone. I would be curious to know if learning a single multiplicative factor achieves the majority of continual learning benefits observed here and in [28, 20].

The primary motivation for extending ADNs to CNNs came from the fact that the number of trainable parameters explodes. [20] trained a rather shallow ADN with over 300 million weights to do a variant of MNIST. While the number of training parameters is large, they pointed out that some post-processing can be done to reduce the parameters to just 1 per active dendrite neuron. As mentioned in the Section 2.1, learning fewer parameters was one of the original motivating factors for sharing convolutional weights. Nowadays we have big data and big compute. It might be time to revisit the biological plausiblity of weight sharing in striate cortex. DNNs in the past decade have benefited from removing weight sharing from convolutional layers [48, 66].[5]

Alexnet made sparks in the community with only 2 GPUs totalling 6GB of VRAM [32]. Techniques today spend tens of thousands of dollars on renting compute, or sometimes millions on renting or building super computers (eg GPT-3, alphastar). The amount of compute thrown into searching the weight space is orders of magnitude larger nowadays [54, 67].

[2] in the 1920s found that hanging heavier weights from a muscle led to faster firing rates from neurons inside the muscles. This is where rate coding originated from, the idea that information in nervous systems are communicated in the form of average firing rates. This is surely one form of communication, and usually assumed to be what the ReLU activation function is modeling, yet several other temporal codes are known to be used in BNNs [21]. As evidence that DNNs use rate coding are the numerous successful ANN-to-SNN [6] conversions methods rely on replacing activation values in DNNs (e.g. ReLU, sigmoid, and tanh) with rate-based coding [62, 11, 46, 51, 9, 12]. Many of these approaches suffer little accuracy loss from the conversion process, and can be implemented much more efficiently on neuromorphic hardware

[5]Yes, I know we won't be guaranteed translational invariance. Perhaps the weights of some feature maps could remain shared.

[6]Spiking Neural Networks, often thought to model more tricks from the brain

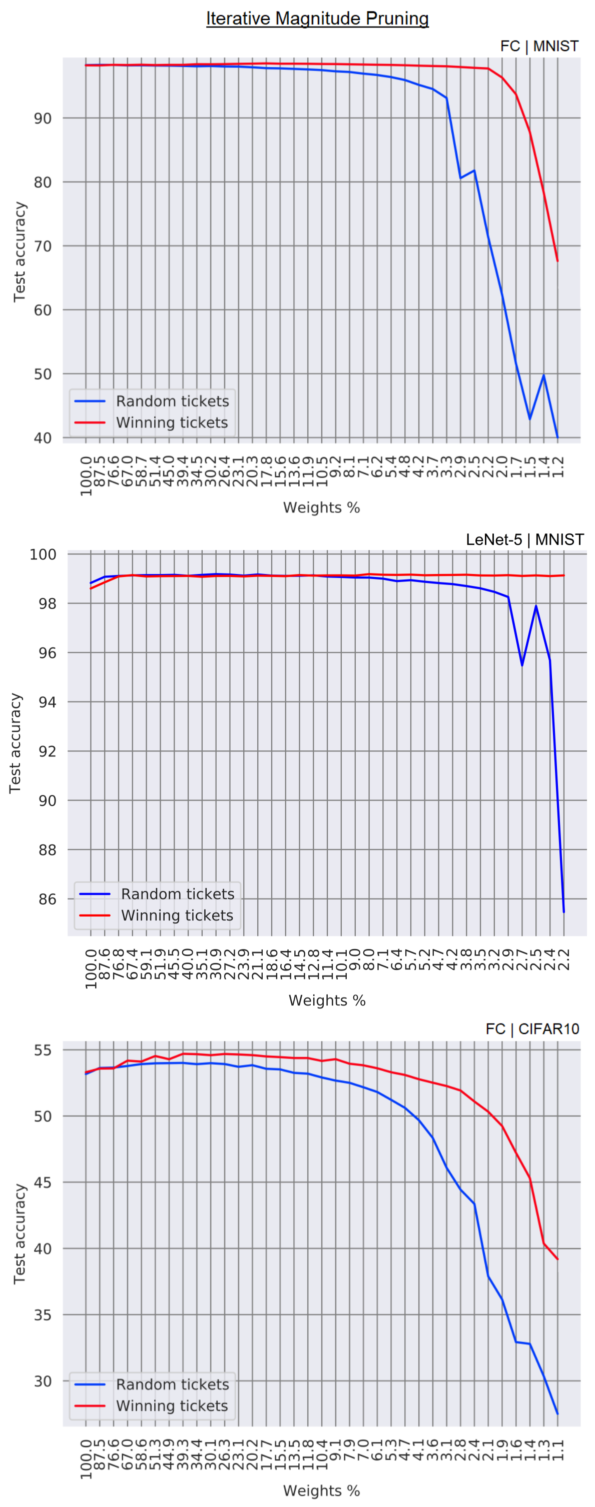

Figure 6: Averaged training curves from 8 randomly pruned networks and 8 winning lottery tickets produced with IMP (see Algorithm 1). 12.5% of the remaining weights are pruned after each training iteration.

[51, 9, 12, 44].

Numenta [27] used FPGAs to benefit from both weight and activation sparsity. Combined with smaller network memory usage, the net effect was a 112x throughput increase on google's speech command dataset. Modern GPUs benefit from dense computations, and do not have very efficient random access memory relative to FPGAs [27].

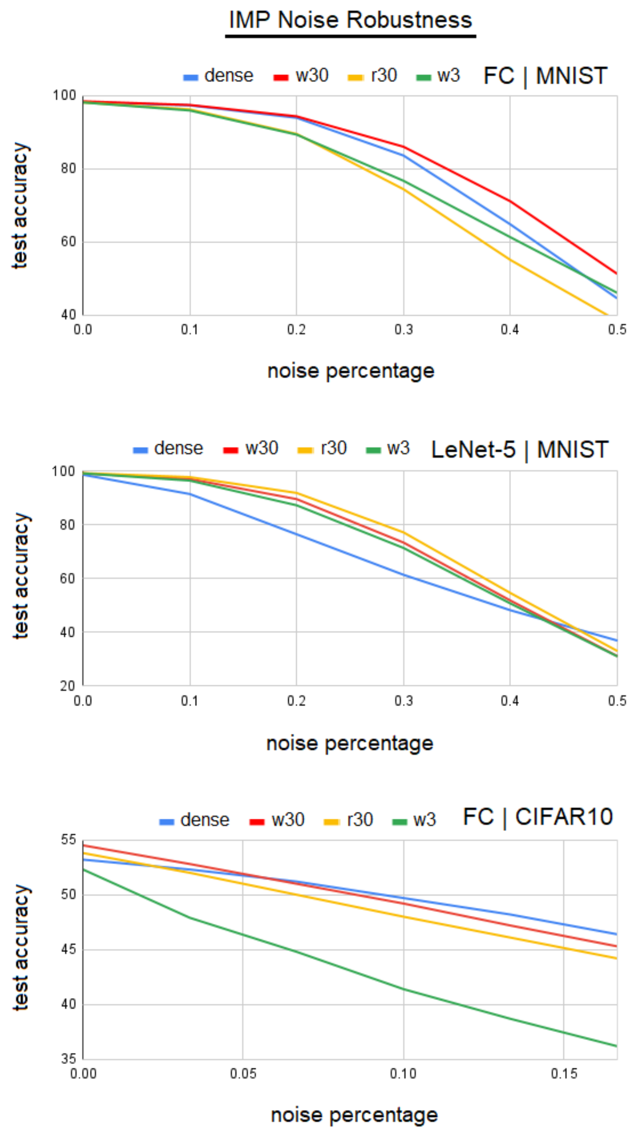

Figure 7: The effects of image noise on randomly pruned networks, winning lottery tickets, and fully dense networks. *r30* are randomly pruned networks with 30% of their weights remaining wrt the fully dense networks. *w3* and *w30* are winning tickets with 3% and 30% weights remaining respectively. Results are an average over 8 trained networks of each type.

Numenta [3] showed that their ADN network learned to activate sparse sub-networks, which facilitated retaining previously learned task skills. While this attempts to make DNN updates more like the brain's, it remains to be demonstrated how the multiplicative contextual gating is implemented in

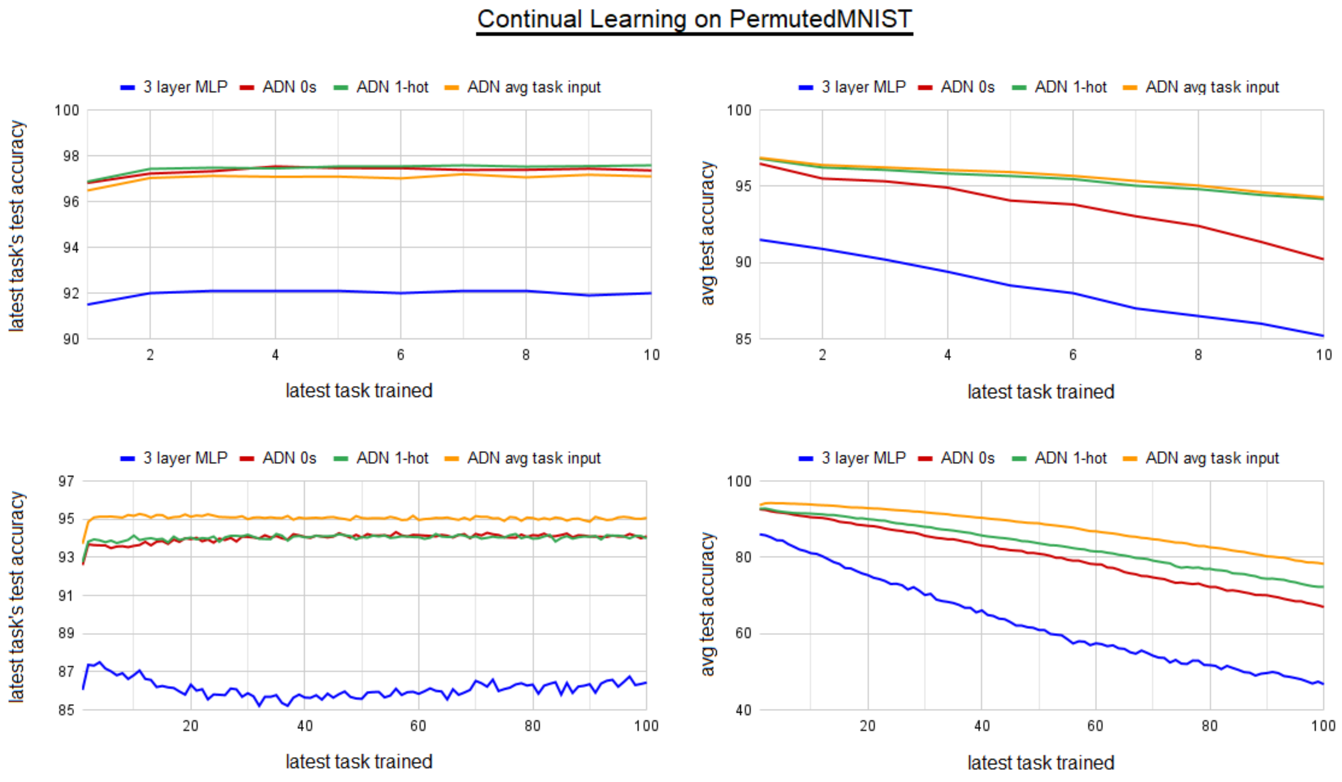

Figure 8: Replicated subset of results from [20]. Additionally evaluated the importance of the context type. *ADN 0s* has the same amount of dendritic weights as *ADN 1-hot*, but always gives a context signal of 0s, essentially proving no task specific information. *ADN avg task input* receives the task averaged input image flattened as context, which results in a much larger network (see Fig. 3 and Fig. 4 for an explanation).

terms of BNNs. The literature on NMDA spikes point to different computations that interact in a non-WTA way, and instead they should combine their ability to produce more APs. Note that smaller dendritic spikes frequently fail to propagate to the soma, and instead rely on combining their effects with other spikes or bAPs to significantly affect the voltage at the soma [53, 64, 17]. It is worth taking into account that DNN's uses rate coding and NMDA spike duration is linear wrt input strength, while imagining what effect dendritic spikes have on neuronal computations. In the future, someone should test 1D convolutions with shared weights as a way to model NMDA spikes since there are overlapping segments of dendrite that can trigger such spikes. Direction and a decaying sum could also be used to represent the spikes starting at a distal point of the dendrites and working their way towards the cell body, allowing the more distant signals to influence the more proximal ones.

Chavlis and Poirazi's recent work [10] also focuses on the potential benefits of modeling active dendritic processes in DNNs. One area they touch on that this work does not is the plasticity rules that govern the dynamic rewiring of synapses (i.e. connections) on dendritic arbors. They believe such rules could improve transfer and continual learning performance. This is a promising future direction given that most neural net architectures are static in their connections, where it is estimated that tens of thousands of connections are being removed or added every second in the adult human brain [65], and at least an order of magnitude more during the first 7 years of childhood [57].

Nearly all mammals studied have abundant amounts of pyramidal neurons. They constitute perhaps the most important cell type in our brains, and are found primarily in areas associated with advanced cognition [59]. The literature shows that they have not yet been accurately modeled at the level of spike timing whether in SNNs or ANNs [45, 31, 30].

Recently [6] used a detailed compartmental model to create a synthetic dataset of pyramidal neuron's spiking behavior, and trained a deep CNN to replicate them. They concluded that a network between 5 or 8 layers deep was sufficient to reproduce the spiking dynamics of a single real pyramidal neuron. While this would support the claim that our point neuron models are outdated, the fidelity of the oracle they used from [22] is questionable. [22] had data from 3 real L5b pyramidal neurons. They tried modeling spiking features of both the apical trunk and soma. They successfully created 4 models of 1 cell with a lot of manual intervention. They could not get

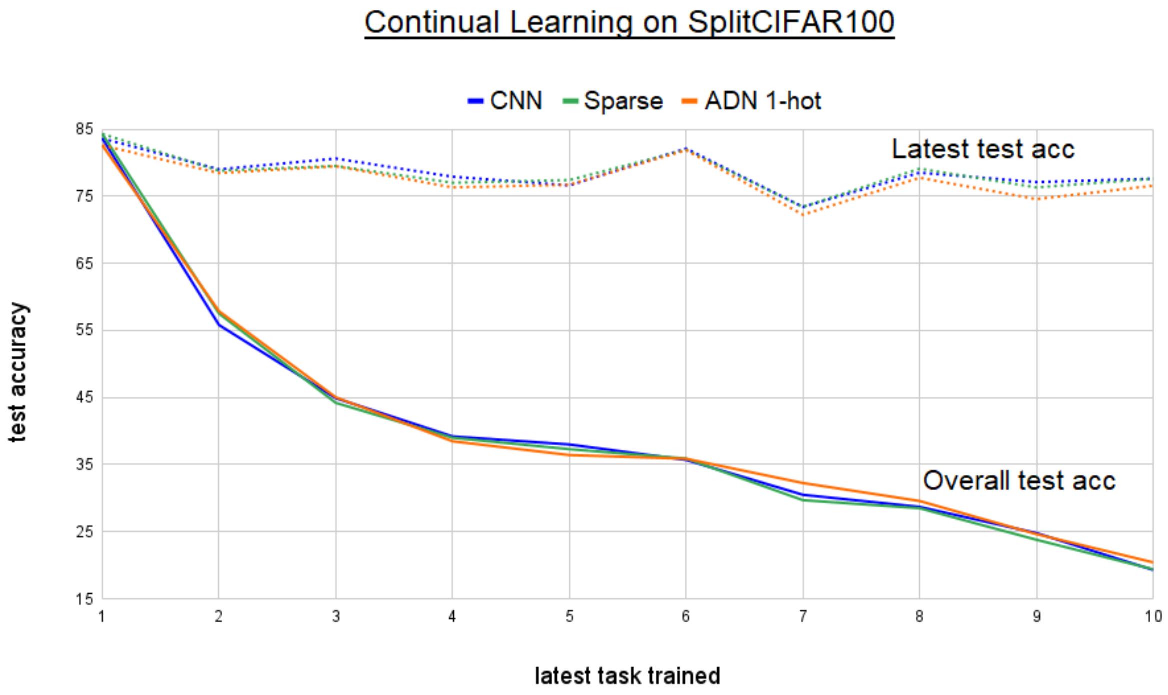

Figure 9: Latest and average task test accuracy while training on SplitCIFAR100, a dataset of random 10-way classification tasks made from CIFAR100. The hyperparameter grid search axes and resulting values are in Appendix A

their method to work for the other 2 cells of which they had data on. Using this modeling method to generate a dataset of 1,000s of synthetic spike trains seems unjustified.

5.1. Let's Stay On The Path.

DNNs still struggles with the following task domains:

- Few shot learning
- Continual learning
- Goal alignment
- Common sense
- Natural language inference
- Sensor fusion
- Adversarial robustness
- Task & motion planning
- Network architecture search (NAS)

Our brains have solutions to all of them, and runs on 20 watts [49]. [7] Humans are often the implicit, yet appropriate, benchmark for artificial general intelligence (AGI). We are the only somewhat generally intelligent things that we can be certain exists. Evolution by natural selection has spent astronomical compute and time discovering some really neat tricks. Some are inside ANNs today, but they could surely use more. We should continue following the path that evolution has already laid down. It is easier to reverse engineer something than it is to reinvent it, even something as complicated as the brain.

[7] Even under seemingly demanding tasks it shows fairly constant usage. Lee Sedol was likely using 25 watts playing AlphaGo, which used 6,800x more. A good portion of AlphaGo's algorithm is attributed to reinforcement learning's TD models from Sutton and Barto, who were modeling behavioral data from mammals (eg Pavlov's dogs).

References

[1] Backpropagating action potentials in neurones: measurement, mechanisms and potential functions. *Progress in Biophysics and Molecular Biology*, 87(1):145–170, 2005. Biophysics of Excitable Tissues.

[2] E. D. Adrian and Y. Zotterman. The impulses produced by sensory nerve-endings. *The Journal of Physiology*, 61(2):151–171, 1926.

[3] S. Ahmad and L. Scheinkman. How can we be so dense? the benefits of using highly sparse representations. *CoRR*, abs/1903.11257, 2019.

[4] S. D. Antic, W.-L. Zhou, A. R. Moore, S. M. Short, and K. D. Ikonomu. The decade of the dendritic nmda spike. *Journal of Neuroscience Research*, 88(14):2991–3001, 2010.

[5] R. Bain. Visualizing the loss landscape of winning lottery tickets. *CoRR*, abs/2112.08538, 2021.

[6] D. Beniaguev, I. Segev, and M. London. Single cortical neurons as deep artificial neural networks. *Neuron*, 109(17):2727–2739.e3, 2021.

[7] N. Brunel and M. van Rossum. Quantitative investigations of electrical nerve excitation treated as polarization. *Biological Cybernetics*, 97:341–349, 12 2007.

[8] N. Brunel and M. van Rossum. Lapicque's 1907 paper: From frogs to integrate-and-fire. *Biological cybernetics*, 97:337–9, 01 2008.

[9] Y. Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2014.

[10] S. Chavlis and P. Poirazi. Drawing inspiration from biological dendrites to empower artificial neural networks. *Current Opinion in Neurobiology*, 70:1–10, 2021. Computational Neuroscience.

[11] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[12] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha. Backpropagation for energy-efficient neuromorphic computing. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[13] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

[14] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[15] K. Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[16] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):826–834, 1983.

[17] S. Gasparini and J. C. Magee. State-dependent dendritic computation in hippocampal ca1 pyramidal neurons. *Journal of Neuroscience*, 26(7):2088–2100, 2006.

[18] A. Gidon, T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I. Vida, and M. E. Larkum. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87, 2020.

[19] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013.

[20] K. Grewal, J. Forest, B. P. Cohen, and S. Ahmad. Going beyond the point neuron: Active dendrites and sparse representations for continual learning. *bioRxiv*, 2021.

[21] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15, 2021.

[22] E. Hay, S. Hill, F. Schürmann, H. Markram, and I. Segev. Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. *PLoS computational biology*, 7:e1002107, 07 2011.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[24] S. Hooker. The hardware lottery. *CoRR*, abs/2009.06489, 2020.

[25] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160, 1962.

[26] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, 195:215–243, 1968.

[27] K. L. Hunter, L. Spracklen, and S. Ahmad. Two sparsities are better than one: Unlocking the performance benefits of sparse-sparse networks. *CoRR*, abs/2112.13896, 2021.

[28] A. Iyer, K. Grewal, A. Velu, L. O. Souza, J. Forest, and S. Ahmad. Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *CoRR*, abs/2201.00042, 2022.

[29] T. Jarsky, A. Roxin, W. Kath, and N. Spruston. Conditional dendritic spike propagation following distal synaptic activation of hippocampal ca1 pyramidal neurons. *Nature neuroscience*, 8:1667–76, 01 2006.

[30] R. Jolivet, R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, and W. Gerstner. A benchmark test for a quantitative assessment of simple neuron models. *Journal of neuroscience methods*, 169:417–24, 05 2008.

[31] R. Jolivet, F. Schürmann, T. Berger, R. Naud, W. Gerstner, and A. Roth. The quantitative single-neuron modeling competition. *Biological cybernetics*, 99:417–26, 12 2008.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[33] A. U. Larkman. Dendritic morphology of pyramidal neurones of the visual cortex of the rat: Iii. spine distributions. *Journal of Comparative Neurology*, 306(2):332–343, 1991.

[34] Y. Lecun. Une procedure d'apprentissage pour reseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). In *Proceedings of Cognitiva 85, Paris, France*, pages 599–604, 1985.

[35] Y. Lecun. *Generalization and network design strategies*. Elsevier, 1989.

[36] D. Ledergerber and M. Larkum. Properties of layer 6 pyramidal neuron apical dendrites. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30:13031–44, 09 2010.

[37] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning, 2019.

[38] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840, 2017.

[39] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization, 2017.

[40] J. C. Magee and D. Johnston. Synaptic activation of voltage-gated channels in the dendrites of hippocampal pyramidal neurons. *Science*, 268(5208):301–304, 1995.

[41] G. Major, M. E. Larkum, and J. Schiller. Active properties of neocortical pyramidal neuron dendrites. *Annual Review of Neuroscience*, 36(1):1–24, 2013. PMID: 23841837.

[42] A. Makhzani and B. Frey. k-sparse autoencoders, 2013.

[43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[44] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. pages 1 – 4, 10 2011.

[45] R. Naud, B. Bathellier, and W. Gerstner. Spike-timing prediction in cortical neurons with active dendrites. *Frontiers in Computational Neuroscience*, 8, 2014.

[46] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7:178, 10 2013.

[47] J. Orbach. Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. *Archives of General Psychiatry*, 7(3):218–219, 09 1962.

[48] J. Ott, E. Linstead, N. LaHaye, and P. Baldi. Learning in the machine: To share or not to share? *CoRR*, abs/1909.11483, 2019.

[49] M. Raichle and D. Gusnard. Appraising the brain's energy budget. *Proceedings of the National Academy of Sciences of the United States of America*, 99:10237–9, 09 2002.

[50] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[51] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11, 2017.

[52] D. E. Rumelhart and J. L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

[53] J. Schiller, Y. Schiller, G. Stuart, and B. Sakmann. Calcium action potentials restricted to distal apical dendrites of rat neocortical pyramidal neurons. *The Journal of Physiology*, 505(3):605–616, 1997.

[54] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos. Compute trends across three eras of machine learning, 2022.

[55] G. L. Shaw. Donald hebb: The organization of behavior. In G. Palm and A. Aertsen, editors, *Brain Theory*, pages 231–233, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[56] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[57] J. Silbereis, S. Pochareddy, Y. Zhu, M. Li, and N. Sestan. The cellular and molecular landscapes of the developing human central nervous system. *Neuron*, 89(2):248–268, 2016.

[58] P. J. Sjöström and M. Häusser. A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons. *Neuron*, 51(2):227–238, 2006.

[59] N. Spruston. Spruston n. pyramidal neurons: dendritic structure and synaptic integration. nat rev neurosci 9: 206-221. *Nature reviews. Neuroscience*, 9:206–21, 04 2008.

[60] N. Spruston, Y. Schiller, G. Stuart, and B. Sakmann. Activity-dependent action potential invasion and calcium influx into hippocampal ca1 dendrites. *Science*, 268:297–300, 05 1995.

[61] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[62] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco. An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Frontiers in Neuroscience*, 11, 2017.

[63] G. J. Stuart and B. J. Sakmann. Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. *Nature*, 367:69–72, 1994.

[64] G. J. Stuart, N. Spruston, B. J. Sakmann, and M. Häusser. Action potential initiation and backpropagation in neurons of the mammalian cns. *Trends in Neurosciences*, 20:125–131, 1997.

[65] T. Südhof. The cell biology of synapse formation. *Journal of Cell Biology*, 220, 07 2021.

[66] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. 09 2014.

[67] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[68] S. Watanabe, D. A. Hoffman, M. Migliore, and D. Johnston. Dendritic k<sup>+</sup> channels contribute to spike-timing dependent long-term potentiation in hippocampal pyramidal neurons. *Proceedings of the National Academy of Sciences*, 99(12):8366–8371, 2002.

[69] H. Yu, S. Edunov, Y. Tian, and A. S. Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp, 2020.

[70] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. 03 2017.

[71] S. Zhang, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong. Why lottery ticket wins? a theoretical perspective of sample complexity on sparse neural networks, 2021.

[72] H. Zhou, J. Lan, R. Liu, and J. Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask, 2020.

A. Experimental Details

A.1. LTH Experiments

The LeNet-5 architecture used 3x3 kernels, max pooling, and ReLU activation functions after all 3 fully connected layers. The first convolutional layer took in a 1 channel image and outputs 64 feature maps. The latter convolutional layer takes in and outputs 64 channels.

The fully connected nets are the same for MNIST and CIFAR10, except for the variable length flattened input image dimensions: 784 and 3,072 respectively. The first fully connected layers output 300 neurons, and the final layer output 10, one for each output class.

A.2. LTH Hyperparameters

All LTH experiments used the adam optimizer with a weight decay constant of 1e-4 and learning rate (LR) of 1.2e-3. Prune percent per training loop was 12.5%. Batch size was 60.

A.3. ADN Experiments

The adam optimizer was used for all experiments. All parameters of models used to replicate Numenta's [20] previous findings were taken from their hyperparameter sweeps (i.e. the same values they used).

Network	LR	batch size	weight decay	conv activation sparsity	ff activation sparsity	ff weight sparsity
CNN	1e-3	64	1e-5	100%	100%	100%
Sparse	1e-3	32	0	20%	10%	50%
ADN	1e-3	128	0	20%	30%	50%

A.4. ADN hyperparameter search

Base CNN: LRs [1e-5, 1e-4, 1e-3],
batch sizes [32, 64, 128],
adam weight decays [0, 1e-5, 1e-4]

Sparse: LRs [1e-5, 1e-4, 1e-3],
batch sizes [32, 64, 128],
conv activation sparsities [0.1, 0.2, 0.3],
ff activation sparsities [0.1, 0.3],
ff weight sparsities [0.3, 0.5, 0.7]

ADN: LRs [1e-5, 1e-4, 1e-3],
batch sizes [32, 64, 128],
conv activation sparsities [0.1, 0.2, 0.3],
ff activation sparsities [0.1, 0.3],
ff weight sparsities [0.3, 0.5, 0.7]