SPARLING: Learning Latent Representations with Extremely Sparse Activations

Kavi Gupta [1] **Osbert Bastani** [2] **Armando Solar-Lezama** [1]

Abstract

Real-world processes often contain intermediate state that can be modeled as an extremely sparse tensor. We introduce SPARLING, a new kind of informational bottleneck that explicitly models this state by enforcing extreme activation sparsity. We additionally demonstrate that this technique can be used to learn the true intermediate representation with no additional supervision (i.e., from only end-to-end labeled examples), and thus improve the interpretability of the resulting models. On our DIGITCIRCLE domain, we are able to get an intermediate state prediction accuracy of 98.84%, even as we only train end-to-end.

1. Introduction

A hallmark of deep learning is its ability to learn useful intermediate representations of data from end-to-end supervision via backpropagation. However, these representations are often opaque, with components not referring to any semantically meaningful concepts. Many approaches have been proposed to address this problem by leveraging extra knowledge in the form of additional supervision or handcrafted constraints on the intermediate representation. For instance, concept bottlenecks leverage labels for the intermediate concepts (Koh et al., 2020), and information bottlenecks impose that that the mutual information between the representation and the input be bounded (Bourlard & Kamp, 1988). Here, we consider the constraint of extreme sparsity, which, when applicable, leads to a particularly effective approach to discovering the true underlying structure.

We introduce SPARLING, a novel technique for learning *extremely sparse representations*, where $\geq$99% of the activations are sparse for a given input. We are motivated by settings where components of the intermediate representation correspond to spatial concepts—which we call *motifs*—that occur in only a small number of locations. For instance, in a character recognition task, each motif may encode whether the center of a given character occurs at a given position. Since even in the worst case, an image of pure text, the image has orders of magnitude fewer characters than pixels, we expect the intermediate representation to be extremely sparse. This pattern is representative of many other prediction tasks—e.g., one could predict economic signals from satellite data by identifying a small number of building types, or detect bird social behavior from nature recordings by analyzing bird chirps.

SPARLING directly enforces sparsity by setting activations below some threshold equal to zero; this threshold is iteratively updated to achieve a target sparsity level (e.g., 99%). A key challenge is that the optimization problem is very unstable for high sparsity values. To address this issue, our optimization algorithm anneals the target sparsity over time. A byproduct of this approach is that we achieve a tradeoff between sparsity values and accuracies during the course of training, enabling the user to post-hoc choose a desired sparsity level.

Example. Figure 1 shows our DIGITCIRCLE task, consisting of noisy images that contain digits placed in a circle. The goal is to list the digits in counterclockwise order starting from the smallest one. In our framework, each digit is a motif, and it occurs at a very sparse number of positions in the input image. The final label can be computed as a function of these motifs and their positions.

Crucially, we want to learn to predict these motifs given no labeled supervision about their positions—i.e., the position of each digit is not provided during training. Despite training only on end-to-end supervision (i.e., input images and labels of the form "072634"), our model is able to act as an effective predictor (up to permutation) of digit positions, identifying the correct digit 98.84% of the time on average.

Additionally, it is able to achieve high end-to-end accuracy of 97.42%, while achieving nearly the maximum sparsity possible (99.9950%; the maximum sparsity possible for this domain is 99.9955%). Alternate sparsity enforcement techniques employing L_1 and KL-divergence loss cannot reproduce these results and either do not produce extreme sparsity or have accuracy close to 0%.

[1]Massachusetts Institute of Technology [2]University of Pennsylvania. Correspondence to: Kavi Gupta <kavig@mit.edu>.

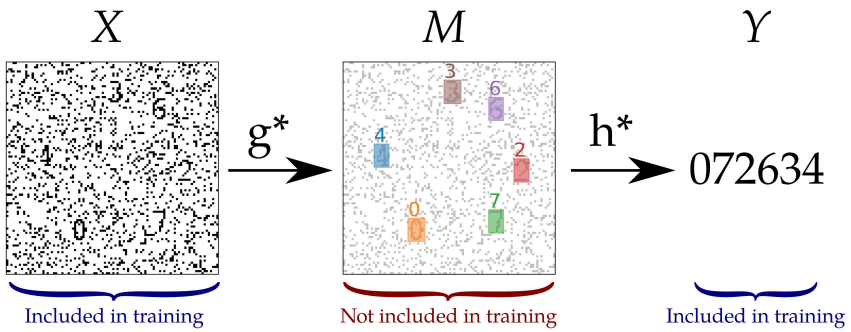

Figure 1. Example of the DIGITCIRCLE domain. The input x is mapped by the ground truth g^* function to a map m of the positions of every digit, which is itself mapped by the ground truth h^* function to the output y, the sequence of symbols `072634`. Only x and y are available during training.

Contributions. Our main contribution is SPARLING, an algorithm for learning intermediate representations with extremely sparse activations, along with an empirical evaluation of the effectiveness of our approach. In particular, we show that our approach can successfully learn the correct latent motifs given only end-to-end supervision.

2. Related Work

Concept bottleneck models. There has been work on learning models with intermediate features that correspond to known variables. Some techniques, such as Concept Bottleneck Models (Koh et al., 2020) and Concept Embedding-Models (Zarlenga et al., 2022), involve additional supervision with existing feature labels. Other techniques, such as Cross-Model Scene Networks (Aytar et al., 2017), use multiple datasets with the same intermediate representation. SPARLING does not require the presence of additional datasets or annotations.

Neural Input Attribution. SPARLING is useful for identifying the relevant parts of an input. One existing technique that accomplishes this goal is saliency mapping (Simonyan et al., 2013; Selvaraju et al., 2016), which uses a backward propagating algorithm (either the standard backpropagation automatic differentiation algorithm or a variant) to find which parts of the input affect the output most. Another technique, looking at the attention weights of an attention model (Mnih et al., 2014), only works with a single layer of attention and also has well known pitfalls in terms of the validity and completeness of the explanations (Serrano & Smith, 2019). The main benefit a sparse annotation provides over these techniques is the property of unconditional independence. Specifically, when using sparsity, you have the ability to make the claim "region $x[r]$ of the input is not relevant to the output prediction, regardless of what happens in the rest of the input $x[\bar{r}]$". This is a direct result of the fact that if a location is not annotated as a motif, this is a purely local decision and as 0s are overwhelmingly common, they thus carry little information. This property is unavailable using saliency or attention techniques as these techniques condition on the values you provide for $x[\bar{r}]$.

Latent ground truth. While deep neural networks typically have inscrutable latent variables that are not intended to correspond to any understood feature, in other settings, such as graphical models, latent variables can often represent real parts of a known system. A commonly used example is Hidden Markov Models with known states, which are commonly used in genomics (Yoon, 2009), where hidden states represent various hidden features of an observed DNA or RNA sequence. Our work attempts to accomplish the same goal of having an interpretable latent variable, but without having to pre-specify what it means.

Disentangled representations. *Disentangled representations* are ones where different components of the representation encode independent attributes of the underlying data (Desjardins et al., 2012). However, these approaches typically seek to capture all attributes of the data rather than select the ones specialized to a specific downstream prediction problem.

Informational bottleneck. Other work also constrains the information content of the intermediate representation in a neural network. Intuitively, by limiting the mutual information between the input and the the intermediate representation, the model must learn to compress the input in a way that retains performance at the downstream prediction task. Strategies include constraining the dimension of the representation—e.g., PCA and autoencoders with low-dimensional representations (Bourlard & Kamp, 1988), or adding noise—e.g., variational autoencoders (Kingma & Welling, 2014). However, these approaches are not designed to learn interpretable representations. By reducing

dimensionality, you increase the chances that multiple different concepts will share a given activation, and by injecting noise, you promote redundancy between neurons and thus reduce the meaningfulness of any given neuron.

Sparse parameters and sparse activations. One popular measure of interpretability is *sparsity*, where models with fewer nonzero values are considered more interpretable. Thus, there has been work on constraining the information content by encouraging the intermediate representation to have *sparse activations*—i.e., each component of the representation is zero for most inputs. Note that this notion of sparsity differs from *sparse parameters* (Tibshirani, 1996; Scardapane et al., 2017; Ma et al., 2019), where the parameters themselves are sparse. Strategies for achieving sparse activations include imposing an L_1 penalty on the representation or a penalty on the mutual information of the representation with a low-probability Bernoulli random variable (Jiang et al., 2015). However, these techniques typically only achieve 50% to 90% sparsity, versus SPARLING, which achieves 99.995%. As discussed in Section 5.1, we directly compare with these as baselines.

3. Preliminaries

We are interested in settings where the activations are latent variables corresponding to semantically meaningful concepts in the prediction problem. To this end, we consider the case where the *ground truth* is represented as a function $f^* : X \to Y$ composed of two functions $g^* : X \to M$ and $h^* : M \to Y$—i.e., $f^* = h^* \circ g^*$. Our goal is to learn models $\hat{g}$ and $\hat{h}$ that model g^* and h^* well using only end-to-end data, i.e., enforcing only that their composition $\hat{f} = \hat{h} \circ \hat{g}$ models f^* well.

We assume that elements of X are tensors (e.g., elements of $\mathbb{R}^{d_1}, \mathbb{R}^{d_1 \times d_2}, ...$), and Y is an arbitrary label space. We typically think of the last dimension of X representing channels and the rest corresponding to spatial dimensions (e.g., 2D images).

We call the latent space M the *motif* space. We assume it shares spatial dimensions with X, but may have a different number of channels. Importantly, we do not assume that M is known—e.g., we may have little or no labeled data on which components of M are active.

3.1. Sparse Activations Assumption

Our critical assumptions are that the output of g^* is *sparse* (i.e., its output equals zero on nearly all components), and that g^* is *local*. To formalize sparsity, we first define the *density* δ to be the expected fraction of nonzero components of the output of g^*. Letting

$$\mathrm{NZ}(m) = \frac{1}{SC} \sum_{\mathbf{i},c} \mathbf{1}(m[\mathbf{i}, c] \neq 0)$$

be the proportion of nonzero entries of m, where S is the total number of positions in m and C is the number of channels, we define

$$\delta_g = \mathbb{E}_x[\mathrm{NZ}(g(x))],$$

where the expectation is taken over the distribution of inputs $x \in X$. Our Sparse Activations Assumption, parameterized by δ_0, can thus be stated as $\delta_g \leq \delta_0 \ll 1$. We use δ to denote $\delta_{\hat{g}}$ for the rest of this paper.

In addition, locality is the standard property where a component only depends on a small number of inputs; for example, convolution filters are designed to parameterize spatially local linear functions.

While these constraints may appear strict, they fit problems where most of the information can be localized to small regions of the input. In these settings, we can trade a small amount of accuracy in exchange for being able to tell precisely what parts of an input are important. Unlike attention layers, this determination is independent of other parts of the input.

3.2. Motif Identifiability Hypothesis

We can then pose the Motif Identifiability Hypothesis as *If g^* and $\hat{g}$ both satisfy locality and the Sparse Activations Assumption, and $\hat{f} \approx f^*$, we know that $\hat{g} \approx g^*$.* This hypothesis means that for certain kinds of functions, it is possible to recover the underlying motif structure with just end-to-end data. Note that this is a narrower claim than Identifiability in general, as we only claim to identify the ground truth (g^*, h^*) functions rather than any individual parameters of g^* or h^*.

3.3. Motif Model Equivalence

Evaluating our Motif Identifiability Hypothesis requires a formal definition of approximate equivalence between motif models—i.e., what $\hat{g} \approx g^*$ means. For the purposes of this paper, we work in a synthetic domain where during final evaluation we can "unseal" M, and thus get a view of the true motifs. However, we need to deal with two additional challenges: channel permutations and motif alignment. Permutations are easily handled by taking the minimum of our error metric over all possible permutations.

Handling motif alignment is more complex. Specifically, there are many different ways to recognize a given pattern, some of which correspond to different motif positions. To

ensure we account for this flexibility when evaluating models, we only check that the predicted point be within the *footprint* of the true motif, which we define as the smallest cuboid[1] covering the points that influence that motif.

We can then define $P(\hat{m})$ as the set of all predicted motifs, $\mathrm{FPM}(\hat{m}, m^*)$ as the set of predicted motifs that do not overlap the footprints of any true motifs, and $\mathrm{MM}(\hat{m}, m^*)$ as the set of predicted motifs that overlap a footprint of a true motif and have greater activation value than all other motifs overlapping the same footprint.[2] We also define $C((\hat{\mathbf{i}}, \hat{c}), m^*)$ to be the footprint that the predicted motif at location $\hat{\mathbf{i}}, \hat{c}$ matches, or $\emptyset$ if it does not match any. For formal definitions of these functions, see Appendix A.

3.4. Evaluation Metrics

Next, we describe the metrics we use to evaluate different models $\hat{f} = \hat{g} \circ \hat{h}$. First, we use the usual end-to-end evaluation of exact match error:

$$\textsc{EndToEnd}_{\mathcal{D}}(\hat{f}) = \mathbb{E}_{x \sim \mathcal{D}}[\mathbf{1}(f^*(x) \neq \hat{f}(x))].$$

This error metric can be calculated given only end-to-end supervision in the form of (x, y) pairs, and it is the only error used in training and validation.

Beyond this basic error metric, we are interested in evaluating $\hat{g} \approx g^*$ in order to test the Motif Identifiability Hypothesis. We define two motif error metrics.

First, the *false positive error (FPE)* is the percentage of motifs that are false positive motifs.

$$\mathrm{FPE}_{\mathcal{D}}(\hat{g}) = \frac{\sum_{x \in \mathcal{D}} |\mathrm{FPM}(\hat{g}(x), g^*(x))|}{\sum_{x \in \mathcal{D}} |P(\hat{g}(x))|}.$$

Second, the *confusion error (CE)* is defined as follows: (i) permute $\hat{g}$'s channels to best align them with g^*, (ii) compute the percentage of maximal motifs in range of a true motif that do not correspond to the true motif's channel:

$$\mathrm{CE}_{\mathcal{D}}(\hat{g}) = \min_{\sigma \in \Sigma_C} \frac{\sum_{x \in \mathcal{D}} |\mathrm{conf}_\sigma(\hat{g}(x), g^*(x))|}{\sum_{x \in \mathcal{D}} |\mathrm{MM}(\hat{g}(x), g^*(x))|},$$

where $\mathrm{conf}_\sigma(\hat{m}, m^*)$ represents the motifs that do not match ground truth under permutation σ

$$\mathrm{conf}_\sigma(\hat{m}, m^*) = \{t \in \mathrm{MM}(\hat{m}, m^*) : \neg \mathrm{mat}_\sigma(t, C(t, m^*))\}|,$$

and $\mathrm{mat}_\sigma(\hat{t}, t^*)$ is a function that checks whether the two motif index tuples match under channel permutation σ.

A low FPE implies that the motifs you do see are probably referring to something real, while a low CE implies that you can correctly identify which true motif is being referred to.

[1] For images, the cuboid is a rectangle, as drawn in Figure 1.

[2] We ignore motifs that are not maximal in a footprint as these would be trivially ignorable when actually using the intermediate layer.

Algorithm 1 Train Loop $(\hat{f}, \mathcal{D}, M, B, d_T, \delta_{\text{update}})$

```
T_0 ← 1
for t = 1 to ... do
  TrainStep(f̂, D_{Bt:B(t+1)})
  T_t ← T_{t−1} − B d_T
  if bt mod M = 0 then
    A_t ← Validate(f̂)
    if A_t > T_t then
      f̂.δ ← f̂.δ × δ_update
      T_t ← A_t
    end if
  end if
end for
```

3.5. Connection to Information Bound

Finally, we establish a connection between SPARLING and information bottleneck approaches. Sparsity induces an information bound by limiting the amount of information in the intermediate representation. Specifically, if we let $\mathcal{X}$ be a random variable for the input, and $\mathcal{M} = g(\mathcal{X})$ be the motif layer, we have that we can bound the mutual information between inputs and motifs as $I(\mathcal{X}, \mathcal{M}) \leq H(\mathcal{M})$, where $H(\cdot)$ is entropy.

Thus, it is sufficient to bound $H(\mathcal{M})$. We first can break it into per-channel components:

$$H(\mathcal{M}) \leq \sum_{\mathbf{i}, c} H(\mathcal{M}[\mathbf{i}, c]),$$

Then, let $\delta_{\mathbf{i},c}$ denote the density of channel c at position $\mathbf{i}$, and η be a bound on the amount of entropy in each nonzero activation:

$$\eta \geq H(\mathcal{M}[\mathbf{i}, c] | \mathcal{M}[\mathbf{i}, c] \neq 0)$$

Then we apply the chain rule

$$H(\mathcal{M}[\mathbf{i}, c]) \leq H(B(\delta_{\mathbf{i},c})) + \eta \delta_{\mathbf{i},c}.$$

Where $B(p)$ denotes the Bernoulli distribution with parameter p. Thus, we have

$$H(\mathcal{M}) \leq \sum_{\mathbf{i}, c} H(B(\delta_{\mathbf{i},c})) + SC\eta\delta,$$

where S is the size of the image in pixels and C is the number of channels, and δ is defined as in section 3.1. Finally, using Jensen's inequality (as $H(B(t))$ is concave), we have

$$H(\mathcal{M}) \leq SC(H(B(\delta)) + \eta\delta).$$

Section 5.5 discusses techniques to bound η.

4. Methods

In this section, we introduce SPARLING, which is composed of two parts: the Spatial Sparsity Layer and the Adaptive Sparsity Algorithm. The Spatial Sparsity Layer is designed to achieve the extreme sparsity rates described in Section 3. This layer is the last step in the computation of $\hat{g}$ and enforces the sparsity of $\hat{g}$; we compose $\hat{g}$ out of convolutional layers to enforce locality. The Adaptive Sparsity Algorithm is designed to ensure the Spatial Sparsity Layer can be effectively trained.

4.1. Spatial Sparsity Layer

We define a spatial sparsity layer to be a layer with a parameter t that whose forward pass is computed

$$\text{Sparse}_t(z) = \text{ReLU}(z - t)$$

Importantly, t is treated as a constant for the purposes of backpropagation and is not updated by gradient descent. Instead, we update t using an exponential moving average of the quantiles of observed training batches:

$$t_n = \mu t_{n-1} + (1 - \mu) q(z_n, 1 - \delta),$$

where t_n is the value of t on the nth iteration, z_n is the nth batch of inputs to this layer, μ is the momentum (we use $\mu = 0.9$), δ is a target density the layer aims to achieve (described in section 3.1), and q is the quantile function. The quantile function $q : \mathbb{R}^{B \times d_1 \times \ldots \times d_k \times C} \times \mathbb{R} \to \mathbb{R}^C$ is implemented such that

$$\forall c, p \approx \frac{1}{BS} \sum_{b,\mathbf{i}} \mathbf{1}(z[b, \mathbf{i}, c] \leq q(z, p)[c])$$

This enforces that each channel must individually have density δ. Thresholds are set uniformly at all positions in the input. We refer to this as the *multiple thresholds (MT)* approach, as opposed to the *single thresholds (ST)* ablation we describe in Section 5.1's "ablation" paragraph.

Since t_n is computed from the data distribution, we can treat it as the $(1 - \delta)^{\text{th}}$ quantile of the distribution of the outputs of the previous network over the data, enabling this layer to set all but a δ fraction of its outputs to 0.

Finally, we always include an affine batch normalization before this layer. This increases training stability, we believe by allowing for gradient signal to propagate even to areas masked by the thresholding of our Sparse layer. We provide an analysis on the necessity of this addition in Section 5.4.

4.2. Adaptive Sparsity

In practice, we find that applying an extreme sparsity requirement (very low δ) upon initial training of the network leads to bad local minima, with the network being unable to gain any learning signal on the vast majority of inputs. Instead, we use a technique inspired by simulated annealing and learning rate decay, and reduce δ slowly over time.

Specifically, we add a step to our training loop that periodically checks validation accuracy A_t and reduces the density whenever it exceeds a target T_t. The process is as described in Algorithm 1, with the target accuracy dropping slowly. When the validation accuracy reaches the target accuracy, we reduce density and increase the accuracy bar to whatever our model achieved.

Our experiments use evaluation frequency $M = 2 \times 10^5$, batch size $B = 10$, $d_T = 10^{-7}$, and $\delta_{\text{update}} = 0.75$.

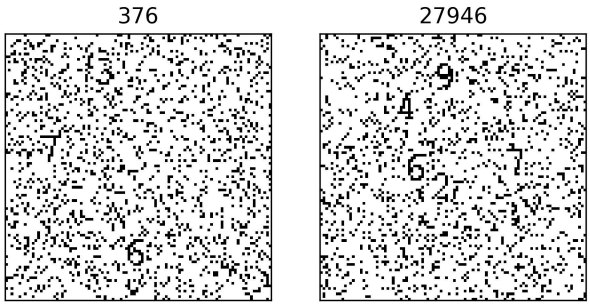

Figure 2. Examples of input/output pairs of the Digit Circle domain. The inputs are the images, and outputs are the sequences of numbers in the title.

5. Experiments

5.1. Experimental Setup

DIGITCIRCLE domain. To evaluate SPARLING we construct the DIGITCIRCLE domain. The input X is a 100×100 monochrome image with 3-6 unique digits placed in a rough circular pattern, with some noise being applied to the image both before and after the numbers are placed. See Figure 2 for examples. The output Y is the sequence of digits in counterclockwise order, starting with the smallest number. The latent motifs layer M is the position of each digit: we can conceptualize this space as a $100 \times 100 \times 10$ tensor with 3-6 nonzero entries. Note that the model during training and validation has no access to the concept of a digit as an image, nor to the concept of a digit's position.

Architecture and training. Our neural architecture is adapted from that of (Deng et al., 2016). We make our $\hat{g}$ architecture a convolutional network with a 17×17 overall window, by layering four residual units (He et al., 2016), each containing two 3×3 convolutional layers. We then map to a 10-channel bottleneck where our Spatial Sparsity layer

is placed. (We choose 10 channels to match the 10 digits.) Our $\hat{h}$ architecture is a max pooling, followed by a similar architecture to Deng. We keep the LSTM row-encoder, but replace the attention decoder with a column-based positional encoding followed by a Transformer (Vaswani et al., 2017) whose encoder and decoder have 8 heads and 6 layers. Throughout, except in the bottleneck layer, we use a width of 512 for all units. For our experiments, we keep this structure stable, and only modify the bottleneck layer.

We use an entirely random generation technique for the dataset, with seeds 1 through 9 for the 9 different training runs of each model, and seeds -1 and -2 being reserved for validation and testing. We use a batch size of 10 samples and a learning rate of 10^{-5}. Our validation and test sets both contain 10^4 examples.

Baselines. We consider two other approaches to ensuring the creation of sparse motifs, both taking the form of auxiliary regularization losses. In both cases, we vary loss weight to see how that affects error and sparsity. First, we consider L_1 loss. In our implementation, we use an affine batch normalization layer followed by a ReLU. The output of the ReLU is then used in an auxiliary L_1 loss. This approach is discussed in (Jiang et al., 2015). We also consider using KL-divergence loss as in (Jiang et al., 2015). The approach is to apply a sigmoid, then compute a KL-divergence between the Bernoulli implied by the mean activation of the sigmoid and a target sparsity value (we use 99.995% to perform a direct comparison). While this usually is done across the training data (Ng, 2011), in our case, the overall sparsity should be similar in all batches, so we instead enforce the loss per-batch (but across all positions and channels). Our other modification, in order to induce true sparsity, is to, after the sigmoid layer (where the loss is computed), subtract 0.5 and apply a ReLU layer.

Ablations. We consider ablations to test three design decisions. First, is the batch normalization we place before our sparse layer necessary? Second, is the adaptive sparsity algorithm we use necessary? Third, we consider the *single threshold (ST)* sparsity approach, where we take the quantile across the entire input (batch axis, dimensional axes, channel axis). In this case, the channels can have differing resulting densities that average together to the target δ. More precisely, we use the quantile function $q_{\text{ST}} : \mathbb{R}^{B \times d_1 \times \ldots \times d_k \times C} \times \mathbb{R} \to \mathbb{R}$, implemented such that

$$p \approx \frac{1}{BSC} \sum_{b,\mathbf{i},c} \mathbf{1}(z[b, \mathbf{i}, c] \leq q_{\text{ST}}(z, p)).$$

5.2. End-to-End Results

End-to-end errors. Figure 3 shows the end-to-end errors of SPARLING and the ST ablation. At 1.5× the theoretical minimum density, we consistently perform under 5% error,

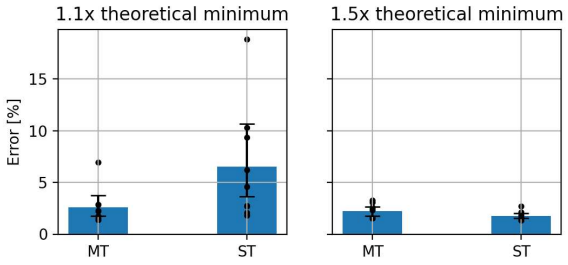

Figure 3. End-to-end error (lower is better). Computed on a test set (different from the validation set used to reduce density). Left plot is computed at 1.1× the theoretical minimum density (one non-zero activation per digit), and right plot is computed at 1.5× the theoretical minimum density. Plotted are 9 separate runs per model, with a 95% bootstrap confidence interval.

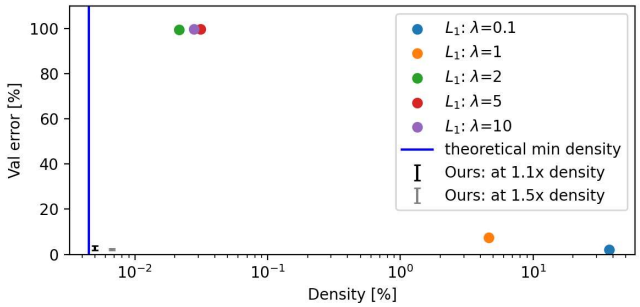

Figure 4. Results of L_1 experiment. Note that the x-axis is log-scaled and the y-axis is error. Our results (same as Figure 3) can be seen on the bottom left.

whereas at 1.1× theoretical minimum density there is a much wider variation from run to run, but the minimum error stays similar, suggesting some instability in SPARLING as it approaches the theoretical minimum density.

Baselines. Figure 4 shows the results of using L_1 as a method for encouraging sparsity. There are two weight regimes, where when $\lambda \leq 1$, we end up with low sparsity (relative to the theoretical minimum) but low error, and when $\lambda \geq 2$, we end up with a model that never learns anything at all (near 100% error). Even in the latter case, the L_1 loss does not consistently push density down to the theoretical minimum or below, suggesting it might be insufficiently strong as a learning signal to achieve the kind of density SPARLING can. In our experiments, the KL-divergence was unable to achieve a density below 0.1%, even when we used a loss weight as high as $\lambda = 10^5$ and 3×10^6 steps (much more than was necessary for convergence of the L_1 model). Thus, we conclude that it is unsuitable for encouraging the kind of sparsity we are interested in.

5.3. Interpretability Results

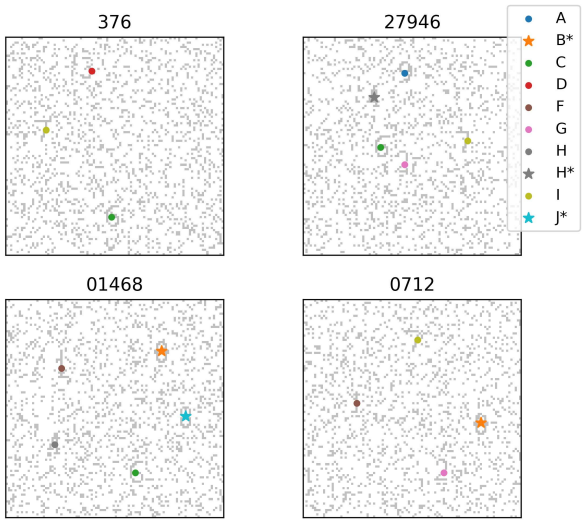

Figure 5. Inputs annotated with the maximal motifs produced by the $\hat{g}$ of the MT model trained with seed=1, at 1.1× the theoretical minimum sparsity. We label our activations A through J to distinguish them from digits. Stars indicate sites where there are non-maximal motifs present as well. These examples are representative and are simply examples 0, 1, 2, and 3 from our dataset.

Examples. Figure 5 shows a few examples for one of our models' intermediate layers. As can be seen, all digits are appropriately identified by our intermediate layer, with very few dots (in these examples, none) falling away from a digit. Most of the slack (here, 10% extra) activations are duplicates on an existing digit. Also, note that the activations are consistent from sample to sample—for example, C is used for digit 6 in all three images where it appears.

Confusion matrix. To provide a more quantitative summary of this effect, consider Figure 6, which shows an analog of a confusion matrix for that model. Note that our model rarely produces an incorrect covering motif, and even more rarely leaves a digit blank.

Motif error. Next, we show our measures of motif error, FPE and CE, in Figure 7 for all the sparsity models.

Errors for our MT model are usually below 10%, and in the 1.1× density case are all below 1% except for in one training run out of the 9. Errors are substantially higher when we have 1.5× minimum density, as our sparsity constraint is less strict so the model is freer to produce an incorrect intermediate representation. The generally low errors on the MT model, despite only having training and validation performed end-to-end, demonstrate that the Motif Identifiability Hypothesis holds for the DIGITCIRCLE domain.

Predicting motif error. Figure 8 shows the relationship between the motif errors and the overall end-to-end error. There is no relationship for FPE, but there is a positive relationship for CE, implying that a strategy where one trains several models and then chooses the one with the best validation error is a good way to reduce CE and thereby improve motif quality.

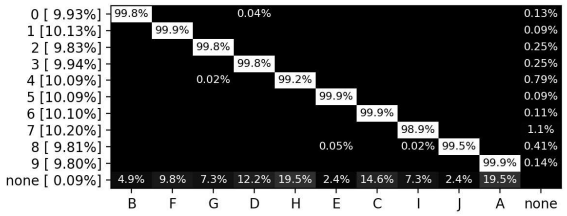

Figure 6. Confusion Matrix of 10k unseen samples (not in training or validation sets). We place false positive motifs into the `none` row and maximal motifs into the rows corresponding to the digit they cover. Each row is labeled by the percentage of motifs falling into the row, and each row's cells are then normalized to add to 1. We also record true motifs that do not have any predicted motifs placed on them as `none` column. The rest of the columns correspond to motifs, labeled A through J and permuted to the permutation that minimizes CE.

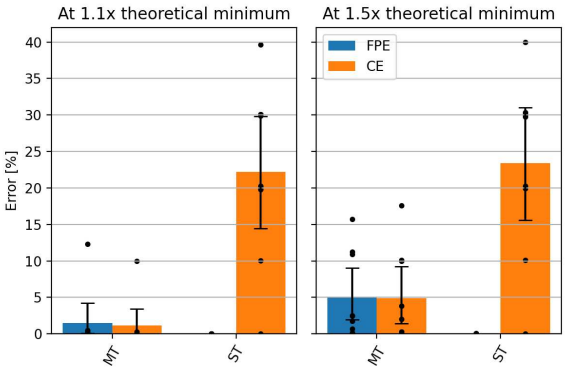

Figure 7. Errors. Bar height depicts the mean across 9 seeds, while individual dots represent the individual values and the error bar represents a 95% bootstrap CI of the mean. The ST model's FPE is so low it does not show up on the chart, all are under 0.012%.

5.4. Ablation

We compare our approach to ablations to evaluate our design decisions. First, including a batch normalization before the sparsity layer is crucial. Without a batch normalization layer, over 9 runs, the best MT model gets an error of 99.35%, whereas the best ST model gets an error of 97.07%; in essence, neither model is able to learn the task at all. We also analyzed the need for the adaptive sparsity update algorithm (Algorithm 1). When starting from 1.1× or 1.5× the theoretical minimum density, the model converged to an er-

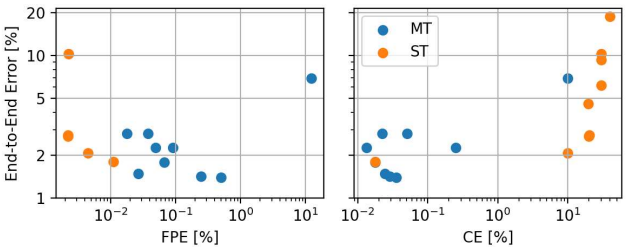

Figure 8. Model error versus FPE and CE, at 1.1× the minimum sparsity. All are log-scaled to highlight the low-error region. Each dot represents a single model training seed.

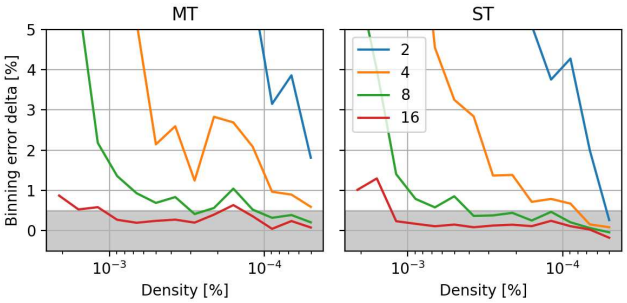

Figure 9. Increase in error when binning. Each series represents a different bin count, as annotated in the legend. Density is log-scaled and reversed to indicate training progress.

ror above 98%. This result suggests that some technique for updating sparsity is necessary to avoid bad local minimia.

Next, as seen in Figure 3, the ST ablation is able to achieve fairly low error end-to-end, but still has a slightly higher average error than MT. In Figure 7, however, we see that it performs substantially worse in terms of CE, while performing better with respect to FPE. Without the constraint that the motifs have equivalent density across each channel, some motifs are being used to represent multiple digits, which substantially increases confusion error, but also reduces false positives. In general, the MT model is superior as it has reasonable FPE and substantially lower CE.

5.5. Entropy upper bound

To compute our entropy upper bound, we must first compute η, as defined in Section 3.5. To compute this, we bin the nonzero activations into 2^k bins by quantile. We set η to be the smallest value of k that does not substantially affect the accuracy of the model (we consider 0.5% to be a reasonable threshold for this purpose). Figure 9 shows the result of this experiment, averaged across 9 seeds. The general downward trend in error caused by binning as density decreases demonstrates that reducing the number of motifs reduces the importance of the precise magnitudes. For the purposes of entropy bounding, we use $\eta = \log(16) = 4$b.

5.6. Error metrics vs Entropy Bound

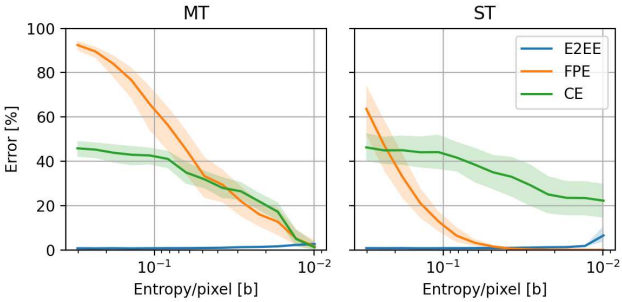

Figure 10. Error metrics used in this project versus entropy per pixel. Note that the x axis is reversed, this is to indicate training progression, which starts with high entropy and narrows it over time. Error region is the 95% confidence interval among 9 seeds.

Figure 10 shows our error metrics plotted against the entropy, with the x-axis reversed to show progression in training time as we tighten the entropy bound. As expected, as entropy decreases, FPE decreases, as there are fewer motifs produced and thus fewer false positives. More interestingly, we find that as entropy decreases, CE decreases while end-to-end error increases. This demonstrates a tradeoff between a more accurate overall model, which benefits from greater information present and a more accurate motif model, which benefits from a tighter entropy bound.[3]

One illuminating result is that even when entropy is about 0.1b/pixel and FPE is very high, CE is still not equivalent to random (which would be about 88% error). This result indicates that the model is mostly choosing the correct motifs to be maximal even at higher levels of entropy, which may explain why Algorithm 1 works: the newly removed activations when the threshold is raised are more likely to be incorrect than not.

6. Conclusion

We have presented SPARLING: a novel spatial sparsity layer and adaptive sparsity training technique that has the ability to learn a highly sparse latent motifs layer for dimensional data, using only an end-to-end training signal. Similar levels of activation sparsity are unachievable by existing strategies. Finally, we demonstrate that SPARLING achieves

[3] While this may seem to contradict the result in Section 5.3, it in fact does not. Within a single model, tightening the density has inverse effects on end-to-end error and CE, but separately, some models are in general more or less accurate.

interpretable and accurate motifs with zero direct training supervision on the motifs.

7. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-1918839 & CCF-1917852. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was enriched by conversations with Christopher Burge, Phillip A. Sharp, Kayla McCue, and Chenxi Yang.

References

Aytar, Y., Castrejon, L., Vondrick, C., Pirsiavash, H., and Torralba, A. Cross-modal scene networks. *IEEE transactions on pattern analysis and machine intelligence*, 40 (10):2303–2314, 2017.

Bourlard, H. and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Deng, Y., Kanervisto, A., and Rush, A. M. What you get is what you see: A visual markup decompiler. *arXiv preprint arXiv:1609.04938*, 10:32–37, 2016.

Desjardins, G., Courville, A., and Bengio, Y. Disentangling factors of variation via generative entangling. *arXiv preprint arXiv:1210.5474*, 2012.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Jiang, N., Rong, W., Peng, B., Nie, Y., and Xiong, Z. An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning. In *2015 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2015.

Kingma, D. P. and Welling, M. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, pp. 121, 2014.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Ma, R., Miao, J., Niu, L., and Zhang, P. Transformed l1 regularization for learning sparse deep neural networks. *Neural Networks*, 119:286–298, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.08.015. URL https://www.sciencedirect.com/science/article/pii/S0893608019302321.

Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

Ng, A. Cs294a lecture notes: Sparse autoencoder, Winter 2011. URL https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.02.029. URL https://www.sciencedirect.com/science/article/pii/S0925231217302990.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

Serrano, S. and Smith, N. A. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Yoon, B.-J. Hidden markov models and their applications in biological sequence analysis. *Current genomics*, 10(6): 402–415, 2009.

Zarlenga, M. E., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Shams, Z., Precioso, F., Melacci, S., Weller, A., et al. Concept embedding models. *arXiv preprint arXiv:2209.09056*, 2022.

A. Evaluation Metric Details

We now define our FPM and MM motif sets, along with the C function.

Predicted motifs. For a given predicted motif tensor $\hat{m}$, we define $P(\hat{m}) = \{(\hat{\mathbf{i}}, \hat{c}) : \hat{m}[\hat{\mathbf{i}}, \hat{c}] > 0\}$ to be the set of motifs predicted in $\hat{m}$, where $\mathbf{i}$ is over all sequences of spatial indices (e.g., for images $\mathbf{i} : \mathbb{N}^2$) and c is over the channel indices. Typically, we are interested in the set of motifs $P(\hat{g}(x))$ for our estimated motif model $\hat{g}$.

Footprint. We can formally define the footprint of a motif as follows: Let a motif with $\mathbf{i}$ in channel c have footprint $\mathbf{i} + F_c$. Note that F_c depends on the channel of the true motif—e.g., in the DIGITCIRCLE domain, some digits are slightly larger than others.

Footprint identification. First, we define a way to determine which footprint a motif belongs to. Define the footprint motif function $S(\hat{\mathbf{i}}, m^*)$ to be the set of true motifs whose footprints contain $\hat{\mathbf{i}}$—i.e.,

$$S(\hat{\mathbf{i}}, m^*) = \{(\mathbf{i}, c) : m^*[\mathbf{i}, c] \wedge \hat{\mathbf{i}} - \mathbf{i} \in F_c\}.$$

As a simplification, since motif footprints typically do not heavily overlap, we define $C(\hat{\mathbf{i}}, m^*)$ to be our classification function that gives a relevant true motif center for the input $\hat{\mathbf{i}}$.

$$C(\hat{\mathbf{i}}, m^*) = u(S(\hat{\mathbf{i}}, m^*)),$$

where u is a choice function that picks an arbitrary element of its input if there are multiple and returns the empty set if there are no entries.

False Positive Motifs. We now have the ability to define our first class of motifs: *false positive motifs*. These are predicted motifs that do not correspond to any real motifs:

$$\mathrm{FPM}(\hat{m}, m) = \{(\hat{\mathbf{i}}, \hat{c}) \in P(\hat{m}) : C(\hat{\mathbf{i}}, g(x)) = \emptyset\}.$$

We denote the remaining motifs by

$$P_1(\hat{m}, m^*) = P(\hat{m}) \setminus \mathrm{FPM}(\hat{m}, m^*).$$

Maximal Motifs First, we need to define the set of all predicted motifs that cover the same footprint as a given predicted motif. We do so via the $A_{\hat{m},m^*}$ function, which takes a given predicted motif (assumed to overlap some footprint) and returns all others covering the same footprint:

$$A_{\hat{m},m^*}(\hat{\mathbf{i}}, c) = \{(\hat{\mathbf{i}}', \hat{c}') \in P(\hat{m}) : C(\hat{\mathbf{i}}', m^*) = C(\hat{\mathbf{i}}, m^*)\}$$

Now we can define *maximal motifs* are predicted motifs that are maximal in the footprint they cover:

$$\mathrm{MM}(\hat{m}, m^*) = \{t \in P_1(\hat{m}, m^*) : \hat{m}[t] = \max_{t' \in A_{\hat{m},m^*}(t)} \hat{m}[t']\}$$

We can also define *non-maximal motifs* are predicted motifs that are non-maximal in the footprint they cover:

$$\mathrm{NMM}(\hat{m}, m^*) = \{t \in P_1(\hat{m}, m^*) : \hat{m}[t] \neq \max_{t' \in A_{\hat{m},m^*}(t)} \hat{m}[t']\}$$

However, we ignore non-maximal motifs entirely for the purposes of our analysis, under the reasoning that these are trivially removable in practice.