Journal of Artificial Intelligence Research 72 (2021) 1-37 Submitted 12/2020; published 09/2021

Optimizing for Interpretability in Deep Neural Networks with Tree Regularization

Mike Wu WUMIKE@STANFORD.EDU
Stanford University, Stanford, CA 94305 USA

Sonali Parbhoo SPARBHOO@SEAS.HARVARD.EDU
Harvard University SEAS, Cambridge, MA 02138 USA

Michael C. Hughes MICHAEL.HUGHES@TUFTS.EDU
Tufts University, Medford, MA 02153 USA

Volker Roth VOLKER.ROTH@UNIBAS.CH
University of Basel, Basel, Switzerland

Finale Doshi-Velez FINALE@SEAS.HARVARD.EDU
Harvard University SEAS, Cambridge, MA 02138 USA

Abstract

Deep models have advanced prediction in many domains, but their lack of interpretability remains a key barrier to the adoption in many real world applications. There exists a large body of work aiming to help humans understand these black box functions to varying levels of granularity – for example, through distillation, gradients, or adversarial examples. These methods however, all tackle interpretability as a separate process after training. In this work, we take a different approach and explicitly regularize deep models so that they are well-approximated by processes that humans can step through in little time. Specifically, we train several families of deep neural networks to resemble compact, axis-aligned decision trees without significant compromises in accuracy. The resulting *axis-aligned* decision functions uniquely make tree regularized models easy for humans to interpret. Moreover, for situations in which a single, global tree is a poor estimator, we introduce a *regional* tree regularizer that encourages the deep model to resemble a compact, axis-aligned decision tree in predefined, human-interpretable contexts. Using intuitive toy examples, benchmark image datasets, and medical tasks for patients in critical care and with HIV, we demonstrate that this new family of tree regularizers yield models that are easier for humans to simulate than L_1 or L_2 penalties without sacrificing predictive power.

1. Introduction

Deep models have become the de-facto approach for prediction in many applications like image classification (e.g. (Krizhevsky, Sutskever, & Hinton, 2012)) and machine translation (e.g. (Bahdanau, Cho, & Bengio, 2014; Sutskever, Vinyals, & Le, 2014)) and further seem poised to advance prediction in real-world domains (Miotto, Li, Kidd, & Dudley, 2016; Gulshan, Peng, Coram, Stumpe, Wu, Narayanaswamy, Venugopalan, Widner, Madams, Cuadros, et al., 2016; Ghassemi, Wu, Hughes, Szolovits, & Doshi-Velez, 2017). However, many practitioners still are reluctant to adopt deep models because their predictions are difficult to interpret. Without interpretability, humans are unable to incorporate their domain knowledge and effectively audit predictions.

In this work, we shall seek a specific form of interpretability known as *human-simulability*. A human-simulable model is one in which a human user can "take in input data together with the parameters of the model and in reasonable time step through every calculation required to produce a prediction" (Lipton, 2016). For example, small decision trees with only a few nodes are easy for humans to simulate and thus understand. Human-simulability is valuable in many domains. In particular, despite advances in deep learning for clinical decision support (e.g. (Miotto et al., 2016; Choi, Bahadori, Schuetz, Stewart, & Sun, 2016; Che, Kale, Li, Bahadori, & Liu, 2015)), the clinical community remains skeptical (and rightfully so) of machine learning systems (Chen, Asch, et al., 2017). The black box nature of neural networks prevents the checks-and-balances and quality control that we expect from healthcare providers. Meanwhile, a simulable model would enable clinicians to audit predictions easily: they can manually inspect changes to outputs under perturbed inputs, check substeps against their expert knowledge, and reason about external factors influencing prediction like systemic bias in the data. Similar needs for simulability exist in many decision-critical domains such as disaster response or recidivism prediction.

Despite the clear appeal of human-simulability, popular models are not simulable. Even simple deep models like multi-layer perceptrons with a few dozen units can have far too many parameters and connections for a human to easily step through (successive matrix multiplications quickly becomes difficult to think about). Richer families of neural networks such as those for sequences are essentially impossible for humans to simulate. However, with their expressivity, these rich families allow for significantly more accurate predictions than commonly simulable models (e.g. decision trees or linear models). Thus, the primary question we consider is the following: *Is it possible for a powerful model such as a deep network to be human-simulable, or at least frequently human-simulable?*

Simulability is a rather strict definition for interpretability as it requires full transparency in prediction. As such, current work on the interpretability of black-box models struggle to balance being both simulable and faithful to the model. For instance, Craven and Shavlik (1996) train decision trees that mimic the predictions of a fixed, pre-trained neural network. Other post-hoc interpretations typically evaluate the sensitivity of predictions to local perturbations of inputs or the input gradient (Ribeiro, Singh, & Guestrin, 2016; Selvaraju, Cogswell, Das, Vedantam, Parikh, & Batra, 2017; Adler, Falk, Friedler, Rybeck, Scheidegger, Smith, & Venkatasubramanian, 2016; Lundberg & Lee, 2016; Erhan, Bengio, Courville, & Vincent, 2009). While these post-hoc interpretations come in many sophisticated forms— others include Singh, Ribeiro, and Guestrin (2016), who use programs to explain a model's predictions as a post-hoc step, and Lakkaraju, Bach, and Leskovec (2016), who learn decision sets based on a learned model— it is difficult to simplify the complex logic of an unregularized neural network to a simulable tree, set, or program. As a result, existing methods only explain local behavior or a lower resolution depiction of global logic. In general, distilling an already-trained neural network to a simple medium is a somewhat ill-posed problem: unregularized neural networks have no incentive to be simulable or obey any other notion of human-interpretability. Instead, they will learn complex decision boundaries fit to succeed at the target task. Trying to enforce interpretability post-hoc must understandably make strong assumptions that over-simplify the model's logic.

In contrast, we begin with the observation that since it is well-known that deep models often have multiple optima of similar predictive accuracy (Goodfellow, Bengio, & Courville,

2016), one might hope to directly find "more interpretable" minima with equal predictive power. In other words, if we consider interpretability from the very start i.e. add an "interpretability term" in the objective function, it might be possible to train neural networks to be both performant and simulable. In general however, the field of *optimizing* deep models for interpretability remains largely nascent. In this vein, Ross, Hughes, and Doshi-Velez (2017) penalize input sensitivity to features marked as less relevant, while Lei, Barzilay, and Jaakkola (2016) train deep models that make predictions from text and simultaneously highlight contiguous subsets of words, called a "rationale," to justify each prediction. Unfortunately, while both works optimize deep models to expose relevant features, these lists of features alone are not sufficient to *simulate* the prediction. We draw a stark distinction between explanation and simulation: the former may describe interpretable features whereas the latter requires defining both features and a procedure for translating them into output. In the following, we introduce two contributions: we first discuss how to optimize deep models to expose prediction logic (not just features) using decison trees, and second, how to generalize this method to incorporate human prior knowledge.

Tree Regularization To optimize for interpretability, one must define an objective function that finds deep models that are both accurate and simulable. To do this, we introduce the notion of *tree regularization.* Specifically, we define a novel model-complexity penalty function that favors optima whose decision boundaries can be well-approximated by small decision trees. In effect, this penalizes models that would require many calculations to simulate predictions. Similar to many popular regularizers such as L_2 or L_1, the tree regularizer is a function on the weights of the neural network. Several of our technical contributions surround making this regularizer differentiable such that it is compatible with stochastic gradient descent. Experimentally, we first exemplify how this technique can be used to train simple multi-layer perceptrons to have tree-like decision boundaries. We then focus on time-series applications and show that gated recurrent unit (GRU) models trained with strong tree-regularization reach a high-accuracy-at-low-complexity sweet spot that is not possible with any strength of L_1 or L_2 regularization. Furthermore, we will show that the decision trees (produced during training) can be used as tools for human simulation – they act as distillations of the deep model and can be given to domain experts. Choosing several real world applications, we demonstrate these features of our approach on a speech recognition task and two medical treatment prediction tasks for patients with sepsis in the intensive care unit (ICU) and for patients with human immunodeficiency virus (HIV). Finally, we generalize tree regularization to the visual domain and apply to standard benchmark datasets. Throughout, we also show that standalone decision trees as a baseline are noticeably less accurate than our tree-regularized deep models.

Granularity of Explanation Thus far, we have implicitly assumed that there exists an optima for a deep model that is simulable while maintaining high performance. For many domains, this may not be true – we may rely on the complexity of a deep model where any strong regularization greatly increases error. In such cases, it may not be possible to have a model that is both accurate and well-approximated by a simple decision tree. To remedy this, we consider *regional* explanations that constrain the model independently across a partitioning of the input space. Coincidentally, this form of explanation is consistent with those of humans, whose models are typically context-dependent (Miller, 2018). For example,

physicians in the intensive care unit do not expect treatment rules to be the same across different categories of patients. Constraining each region to be interpretable allows the deep model more flexibility than a global constraint, while still revealing prediction logic that can generalize to nearby inputs (in contrast to works on local explanation—(Ribeiro et al., 2016; Selvaraju, Das, Vedantam, Cogswell, Parikh, & Batra, 2016; Ross et al., 2017)—which cannot indicate whether the same logic revealed for an input x can be used for nearby inputs x', an ambiguity that can lead to mistaken assumptions). In other words, we assume that even the most complex decision boundaries can be decomposed into an ensemble of simpler regional boundaries, each of which can be well-approximated by a decision tree. With *regional tree regularization*, we incorporate expert knowledge to train more simulable models.

2. Related Work

Given the distinction between "global" and "local" definitions of interpretability, we first provide a brief overview of literature in each. Then, we describe how "optimizing for interpretability" relates to similar existing branches of research.

Global Interpretability Given a *trained* black box model, many approaches exist to explain what the model has learned as a whole. Mordvintsev, Olah, and Tyka (2015) expose the features a representation encodes but not the logic. Amir and Amir (2018) and Kim, Rudin, and Shah (2014) provide an informative set of examples that summarize the system. Model distillation compress a source network into a smaller target neural network (Frosst & Hinton, 2017). However, even a small neural model may not be interpretable. Activation maximisation of neural networks (Erhan et al., 2009; Simonyan, Vedaldi, & Zisserman, 2013; Nguyen, Yosinski, & Clune, 2016; Montavon, Samek, & Müller, 2018) tries to find input patterns that produce the maximum response for a quantity of interest. However, a set of input patterns is not necessarily adequate to simulate a model's predictions. Other approaches still, seek to visualize global structure captured by the model by projecting learned embeddings to lower dimensions (Joliffe & Morgan, 1992; Maaten & Hinton, 2008). Again, while these methods can expose clusters, they do not always enable human simulability, since we cannot necessarily step through any calculation that produces a decision.

Local Interpretability In contrast, local approaches provide explanation for a specific input. Ribeiro et al. (2016) show that using the weights of a sparse linear model, one can explain the decisions of a black box model in a small area near a fixed data point. This captures the intuition that even nonlinear functions are locally linear. Similarly, instead of a linear model, Singh et al. (2016) and Koh and Liang (2017) output a simple program or an influence function, respectively. Other approaches have used input gradients (which can be thought of as infinitesimal perturbations) to characterize the local space (Selvaraju et al., 2016). Similarly, Layerwise-Relevance Propagation (Binder, Bach, Montavon, Müller, & Samek, 2016; Bach, Binder, Montavon, Klauschen, Müller, & Samek, 2015) produces a heatmap (per input example) measuring relevant information for prediction. However, the notion of a local region in these works is both very small and often implicit; it does not match with human notions of contexts (Miller, 2018): a user may have difficulty knowing when local explanations apply and how they generalize to nearby inputs.

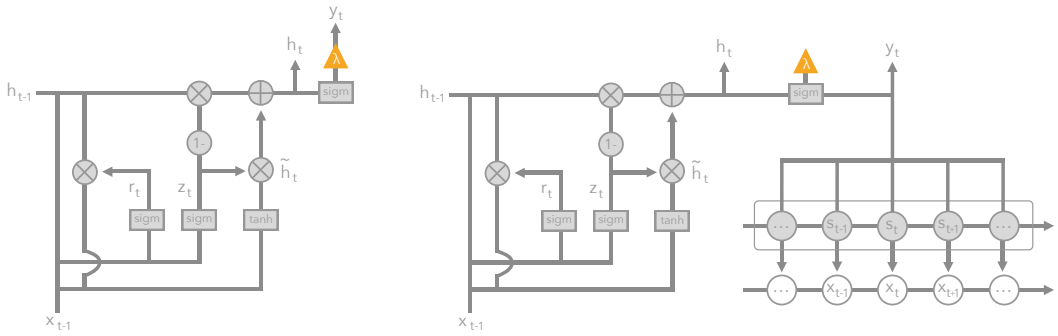
Decision Trees and Neural Networks There have been many attempts to combine decision trees and neural networks to tradeoff accuracy and interpretability. The earliest of which (Craven & Shavlik, 1996) proposes to distill neural networks into decision trees post-training for humans to comprehend. This idea has been recently revisited by Nguyen, Kasmarik, and Abbass (2020) with the same goal of extracting the best of both algorithms. Several other methods have surfaced that "merge" decision trees and neural networks at an architectural level (Kontschieder, Fiterau, Criminisi, & Bulo, 2015; Yang, Morillo, & Hospedales, 2018; Balestriero, 2017), often by imposing a tree structure on neural network nodes. Most recently, neural-backed decision trees (Wan, Dunlap, Ho, Yin, Lee, Jin, Petryk, Bargal, & Gonzalez, 2020) replace a network's final layer with a differentiable decision tree. Similar to our approach, Wan et al. (2020) utilizes a surrogate loss. We view our approach as complementary to this line of work. Rather than impose hard architectural constraints, we use a soft objective to regularize neural networks to be consistent with decision trees.

Optimizing for Interpretability While there is little work on optimizing models for interpretability, there are some related threads. The first is *model compression*, which trains smaller models that perform similarly to large, black-box models (e.g. (Bucilua, Caruana, & Niculescu-Mizil, 2006; Hinton, Vinyals, & Dean, 2015; Balan, Rathod, Murphy, & Welling, 2015; Han, Pool, Tran, & Dally, 2015)). Other efforts specifically train very sparse networks via L_1 penalties (Zhang, Lee, & Jordan, 2016) or even *binary* neural networks (Tang, Hua, & Wang, 2017; Rastegari, Ordonez, Redmon, & Farhadi, 2016) with the goal of faster computation. Edge and node regularization is commonly used to improve prediction accuracy (Drucker & Le Cun, 1992; Ochiai, Matsuda, Watanabe, & Katagiri, 2017), and recently Hu, Ma, Liu, Hovy, and Xing (2016) improve accuracy by training neural networks so that predictions match a small list of known domain-specific first-order logic rules. Sometimes, these regularizations—which all smooth or simplify decision boundaries—*can* have the effect of also improving interpretability. However, there is no guarantee that these regularizations will do so. We emphasize that specifically *training* deep models to have easily-simulable decision boundaries is (to our best knowledge) novel.

3. Background and Models

We introduce notation to recall basic feedforward and recurrent neural networks as well as a popular Markov model. Then, we introduce a new joint model with elements of both.

We consider supervised learning tasks given a data set of N labeled examples, $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, where each example (indexed by n) has an input feature vector $\mathbf{x}_n \in \mathbb{R}^p$ and a target output vector $\mathbf{y}_n \in \mathbb{Z}^q$. For example, we will sometimes write $\mathbf{x}_n = [x_n(1), ..., x_n(p)]$, using $(\cdot)$ to indicate indexing into the vector. We shall assume the targets $\mathbf{y}_n$ are binary, though it is simple to extend to the multiclass or regression setting. When modeling time-series, each example sequence n contains T_n timesteps indexed by t which each have a feature vector $\mathbf{x}_{nt}$ and an output $\mathbf{y}_{nt}$. Formally, we write: $\mathbf{x}_n = (\mathbf{x}_{n1} \dots \mathbf{x}_{nT_n})$ and $\mathbf{y}_n = (\mathbf{y}_{n1} \dots \mathbf{y}_{nT_n})$. Each value $\mathbf{y}_{nt}$ could be a prediction about the next timestep (e.g. the next character) or some other task-related annotation (e.g. if the patient became septic). We primarily consider multi-layer perceptrons and recurrent neural networks. That said, our approach generalizes to any architecture.



(a) GRU (b) GRU-HMM

Figure 1: Architecture diagrams for (a) gated recurrent units (GRU) and (b) a GRU and hidden markov model (HMM) hybrid. The orange triangle indicates the output used in surrogate training for tree regularization.

Multi-Layer Perceptrons. A multi-layer perceptron (MLP) predicts probabilities $\hat{\mathbf{y}}_n$ of the binary target $\mathbf{y}_n$ via a function $f : \mathbb{R}^p \times \Theta \to [0,1]^q$ such that $\hat{\mathbf{y}}_n = f(\mathbf{x}_n; \theta)$, where the vector $\theta \in \Theta$ represents all parameters of the network. Given a data set $\mathcal{D}$, our goal is to learn the optimal parameters θ^* to minimize the objective

$$\theta^* = \arg\min_{\theta \in \Theta} \left(\sum_{n=1}^{N} \mathcal{L}(\mathbf{y}_n, \hat{\mathbf{y}}_n) + \lambda \Psi(\theta) \right) \tag{1}$$

For binary targets $\mathbf{y}_n$, the logistic loss (binary cross entropy) is an effective choice for $\mathcal{L}(\cdot)$. The regularization term $\Psi(\theta)$ can represent L_1, L_2 penalties (Drucker & Le Cun, 1992; Goodfellow et al., 2016; Ochiai et al., 2017), or our new family of regularizers.

Recurrent Neural Networks with Gated Recurrent Units. A recurrent neural network (RNN) takes as input an arbitrary length sequence $\mathbf{x}_n = (\mathbf{x}_{n1} \dots \mathbf{x}_{nT_n})$ and produces a "hidden state" sequence $\mathbf{h}_n = (\mathbf{h}_{n1} \dots \mathbf{h}_{nT_n})$ of the same length as the input. Each hidden state vector at timestep t represents a location in a (possibly low-dimensional) "state space" with k dimensions: $\mathbf{h}_{nt} \in \mathbb{R}^k$. RNNs perform sequential *nonlinear* embedding of the form $\mathbf{h}_{nt} = f(\mathbf{x}_{nt}, \mathbf{h}_{nt-1}; \theta)$ in hope that the state space location $\mathbf{h}_{nt}$ is a useful summary statistic for making predictions of the target $\mathbf{y}_{nt}$ at timestep t. As written, $f : \mathbb{R}^p \times \mathbb{R}^k \times \Theta \to \mathbb{R}^k$ is called a *transition* function parameterized by $\theta \in \Theta$. Many different variants of the transition architecture have been proposed to solve the challenge of capturing long-term dependencies. In this paper, we use gated recurrent units, or GRUs (Cho, Gulcehre, Bahdanau, Schwenk, & Bengio, 2014), which are simpler than other alternatives such as long short-term memory units (Hochreiter & Schmidhuber, 1997). While GRUs are convenient, any differentiable RNN architecture is compatible with our approach.

As review, we describe the evolution of a single GRU sequence, dropping the sequence index n for readability. The GRU transition function f produces the state vector $\mathbf{h}_t = (\mathbf{h}_{t1} \dots \mathbf{h}_{tT})$

from a previous state $\mathbf{h}_{t-1}$ and an input vector $\mathbf{x}_t$, via the feed-forward architecture:

$$\text{output state} : \mathbf{h}_t = (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_{t,k}\tilde{\mathbf{h}}_t \tag{2}$$

$$\text{candidate state} : \tilde{\mathbf{h}}_t = \tanh(\mathbf{V}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{3}$$

$$\text{update gate} : \mathbf{z}_t = \sigma(\mathbf{V}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}) \tag{4}$$

$$\text{reset gate} : \mathbf{r}_t = \sigma(\mathbf{V}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}) \tag{5}$$

The internal network nodes include candidate state gates $\tilde{\mathbf{h}}$, update gates $\mathbf{z}$, and reset gates $\mathbf{r}$ which have the same cardinality as the state vector $\mathbf{h}$. Reset gates allow the network to forget past state vectors when set near zero via the logistic sigmoid nonlinearity $\sigma(\cdot)$, and critically adds a multiplicative factor to this model class. Update gates allow the network to either pass along the previous state vector unchanged or use the new candidate state vector instead. This architecture is diagrammed in Figure 1. The predicted probability of the binary target $\mathbf{y}_t$ for timestep t is a sigmoid transformation of the state at time t, $\hat{\mathbf{y}}_t = \sigma(\mathbf{w}^T\mathbf{h}_t)$ where the weight vector $\mathbf{w}$ represents the parameters of this individual output layer. We denote the parameters for the entire model as $\theta = (\mathbf{w}, \mathbf{U}_h, \mathbf{U}_z, \mathbf{U}_r, \mathbf{V}_h, \mathbf{V}_z, \mathbf{V}_r) \in \Theta$, concatenating component parameters. We train GRU models via the following loss minimization:

$$\theta^* = \arg\min_{\theta\in\Theta} \left(\sum_{n=1}^{N} \sum_{t=1}^{T_n} \mathcal{L}(\mathbf{y}_{nt}, \hat{\mathbf{y}}_{nt}) + \lambda\Psi(\theta) \right) \tag{6}$$

where again $\Psi(\theta)$ defines a regularization cost, and θ^* represents the optimal parameters.

Hidden Markov Models with Stochastic Gradient Descent. Besides recurrent neural networks, hidden markov models (or HMMs) are another class of sequence models that are commonly used to describe stochastic processes. Given a sequence of T_n observed variables $\mathbf{x}_n = (\mathbf{x}_{n1} \dots \mathbf{x}_{nT_n})$, we wish to derive a sequence of T_n latent variables $\mathbf{s}_n = (\mathbf{s}_{n1} \dots \mathbf{s}_{nT_n})$. We assume each latent variable, $\mathbf{s}_{nt}$, can take one of k discrete states. In practice, these latent variables can be interpreted as an unsupervised clustering over the observed sequence. For our purposes, one can view the HMM as a stochastic RNN, making it a probabilistic generative model. To be tractable, the HMM makes a set of simplifying assumptions. The free parameters of an HMM define a *prior*, $p(\mathbf{s}_{n0})$, the probability distribution over k states for the first timestep; a *transition matrix*, $p(\mathbf{s}_{nt}|\mathbf{s}_{n,t-1})$ which specifies a probability distribution over states for timestep t given the state at timestep $t-1$; and an *emission matrix*, $p(\mathbf{x}_{nt}|\mathbf{s}_{nt})$ which specifies a probability distribution over (possibly continuous) observations at timestep t given only the latent at timestep t. Critically, this setup makes the Markov assumption – all information required to make a decision at timestep t is present at timestep $t-1$.

In our setting, we also have a sequence of known outputs, $\mathbf{y}_n = (\mathbf{y}_{n1} \dots \mathbf{y}_{nT_n})$. In some sense, we are not interested in the latent states themselves but use them to classify an observation into output. If we decide upfront to specify a simple classifier on top of the latent variables (such as logistic regression), then we explicitly write the joint distribution over latents, observations, and outputs as:

$$p(\mathbf{x}_n, \mathbf{y}_n, \mathbf{s}_n) = p(\mathbf{s}_{n0}; \phi) \prod_{t=1}^{T} p(\mathbf{s}_{nt}|\mathbf{s}_{n,t-1}; \phi) p(\mathbf{x}_{nt}|\mathbf{s}_{nt}; \phi) p(\mathbf{y}_{nt}|\sigma(\sum_{i=1}^{k} w_i f(\mathbf{s}_{nt})_i + b)) \tag{7}$$

where ϕ are the parameters specifying the prior, transition, and emission probabilities; $\{w_i\}_{i=1}^k$ and bias b are the parameters used in logistic regression; $f(\mathbf{s}_{nt}) = p(\mathbf{s}_{nt}|\mathbf{s}_{n,t-1}, \mathbf{x}_{nt}; \phi)$, the posterior distribution over states at timestep t is represented as a vector with k entries where $f(\mathbf{s}_{nt})_i$ is the i-th entry; and σ represents a Sigmoid function. Therefore, we can train the HMM with stochastic gradient descent by choosing $\theta = \{\phi, w_1, ..., w_k, b\} \in \Theta$ to maximize $p(\mathbf{x}_n, \mathbf{y}_n, \mathbf{s}_n)$. In other words, because we only desire maximum-a-posteriori (MAP) inference, we never need to sample from any of the distributions and therefore can differentiate this objective with standard gradient techniques. Note that this is quite similar to the forward pass in the forward-backward algorithm (Rabiner & Juang, 1986).

Modeling the Residuals of a Hidden Markov Models One strength of the HMM is that it is an interpretable model: the discrete latent states have contextual meaning such that we can analyze the predictions of HMM as conditioned completely on its state. However, for complex domains, discrete states might not be able to fully capture the true decision function, resulting in high prediction error. One option is to add a recurrent neural network, which are known to be high performing but un-interpretable, to model the residual errors when predicting the target outputs using the HMM latent states. If we can properly penalize the complexity of the deep model, then high quality predictions do not come at the price of a less interpretable model. In practice, the GRU and HMM can be trained jointly where the parameters of each model are kept independent. We call this model a GRU-HMM and use it in several experiments. Figure 1(b) recap the model architecture.

4. Tree Regularization

As presented in Equation 1 and 6, the regularizer $\Psi(\theta)$ is arbitrary. Common choices include L_2 norms to manage the size of θ and L_1 norms to manage the sparsity of θ. We now come to our core contribution: we replace $\Psi(\theta)$ with a novel *tree regularizer*, denoted $\Omega(\theta)$, that encourages the model θ to be *simulable*. Specifically, we shall encourage our deep models to be well-approximated by (small) decision trees. For clarity, we refer to the deep neural network that we are trying to regularize as the *target neural model* or target network.

To do so, we first fit a binary decision tree which *accurately* reproduces the target network's thresholded binary predictions $\hat{\mathbf{y}}_n$ given input $\mathbf{x}_n$. The accuracy parameter is always kept fixed, so that the tree is forced to model the network well. Next, we penalize the network based on the complexity of learnt tree: a simple decision function can be explained with only a few branches whereas a complex function may need exceedingly large trees. With this in mind, we quantify complexity as the *average decision path length* (shorthand APL), or the average number of decision nodes that must be touched to make a prediction for any input (i.e. the number of nodes from root to leaf). We compute the *average* with respect to some designated reference data set, $\{\mathbf{x}_n\}_{n=1}^N$. Thus, our regularizer is

$$\Omega(\theta) \triangleq \text{APL}(\{\mathbf{x}_n\}_{n=1}^N, f(\cdot;\theta), h) \tag{8}$$

where the APL function is detailed in Algorithm 1; $f(\cdot;\theta)$ represents the target network; and h is a hyperparameter for training decision trees that controls the minimum number of training examples to define a leaf node. This definition of APL generalizes when the input data represents a timeseries. Algorithm 1 requires two subroutines, TrainTree and PathLength.

Algorithm 1 Average decision path length (APL) Cost Function

Require:

$f(\cdot;\theta)$: binary prediction function, with parameters θ

$\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$: reference data set with N examples

h : minimum number of samples required to be a leaf node; a higher h regularizes the tree, resulting in a smaller tree (default: 1)

function APL($\{\mathbf{x}_n\}, f(\cdot;\theta), h$)
    tree $\leftarrow$ TrainTree($\{\mathbf{x}_n, f(\mathbf{x}_n,\theta)\}_{n=1}^N, h$)
    return $\frac{1}{N}\sum_n$ PathLength(tree, $\mathbf{x}_n$)

Firstly, TrainTree trains a binary decision tree to accurately reproduce the provided labeled examples $\{\mathbf{x}_n, \hat{\mathbf{y}}_n\}$. For this we use the `DecisionTree` module distributed in Python's scikit-learn (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, & Duchesnay, 2011a), which fits a tree by maximizing information gain with Gini impurity. Generally, the runtime cost of this module scales superlinearly with the number of examples N and linearly with the number of features F for a total complexity of $O(FN \log N)$. In practice, we found that with $N = 1000$, $F = 10$, fitting a decision tree takes 15.3 microseconds. These trees can give probabilistic predictions at each leaf. Next, PathLength counts how many nodes are visited to make a prediction for a single input and a trained decision tree.

We consider APL a good proxy for simulability because human simulation requires stepping through every calculation required to make a prediction. Thus, APL exactly counts the number of true-or-false boolean calculations needed to make the average prediction, assuming the model is a binary decision tree. In contrast, a metric such as the total number of nodes might penalize more accurate trees that have short paths for most examples but need more involved logic for few outliers. While a sensible choice, a few technical innovations are required to efficiently optimize the APL loss.

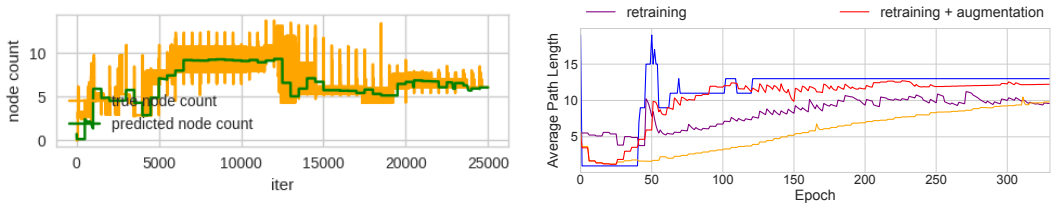
Making Tree Regularization Differentiable Training decision trees is not differentiable, and thus the tree regularization loss $\Omega(\theta)$ from Equation 8 is not differentiable with respect to the network parameters θ. While one could resort to derivative-free optimization techniques (Audet & Kokkolaras, 2016) e.g. search algorithms, gradient descent has been an extremely fast and robust way of training neural networks (Goodfellow et al., 2016).

A key technical contribution of our work is introducing and training a *surrogate* regularization function $\hat{\Omega} : \Theta \rightarrow \mathbb{R}^+$ to map each parameter vector θ of the target neural model to an *estimate* of the APL. Our approximate function $\hat{\Omega}$ is implemented as a standalone multi-layer perceptron network and is critically *differentiable*. Let vector $\xi \in \Xi$ denote the trainable parameters of the surrogate network. We can train $\hat{\Omega}$ to be a good estimator by minimizing a squared error loss function:

$$\min_{\xi \in \Xi} \left(\sum_{j=1}^{J} (\Omega(\theta_j) - \hat{\Omega}(\theta_j, \xi))^2 + \epsilon ||\xi||_2^2 \right) \tag{9}$$

where each θ_j is an instance of the *entire* set of parameters for the target neural model, $\epsilon > 0$ is a regularization strength, and we assume we have a data set of J known parameter vectors and their associated true APLs: $\mathcal{D}^\theta = \{\theta_j, \Omega(\theta_j)\}_{j=1}^J$. This data set can be assembled using

the candidate parameter vectors obtained every gradient step while training our target neural model $f(\cdot, \theta)$. Importantly, one can train the surrogate network in parallel with the target network. In Figure 2a, we show evidence that our surrogate network $\hat{\Omega}(\cdot)$ tracks the true APL effectively as we train the target network.



(a) Surrogate APL vs True APL (b) Effect of Restarts and Augmentation

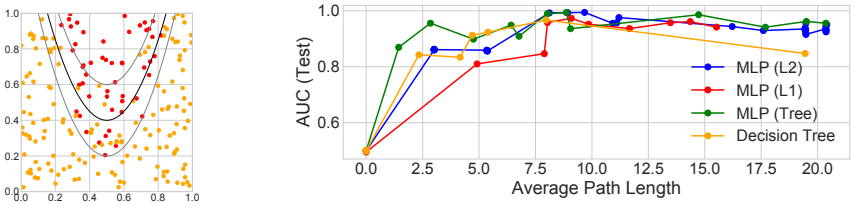
Figure 2: *(a)* True APLs (yellow) and surrogate estimates $\hat{\Omega}$ (green) throughout training (on 2D Parabola). *(b)* Effect of augmentation and random restarts (retraining): The blue line shows the true APL at each epoch. All other lines show predicted APL. By augmenting and restarting, we significantly improve the ability to track the changes in the ground truth.

Training the Surrogate Loss We describe a few more considerations to improve the quality of the surrogate. First, even "small" neural models can have thousands of trainable parameters. Our labeled data set for surrogate training, $\mathcal{D}^{\theta}$, will only obtain one example from each target network gradient step. Even with small batch sizes, this dataset is too small: in early iterations, we will have only few examples from which to learn a good surrogate. We resolve this challenge via *augmenting* our training set with additional examples: We randomly sample weight vectors θ and calculate the true APL $\Omega(\theta)$, and we also perform several random restarts (initializing parameters with different random seeds) on the unregularized target network and use those weights in our training set.

A second challenge arises later in training: as the model parameters shift away from their initial values, parameters from earlier in optimization may poorly characterize the current target neural model. This is a function of the learning rate: a high step size will quickly render recent parameters ineffective for training a surrogate. To address this, for each epoch, we use examples only from the past E iterations, where in practice, E is empirically chosen. Consequently, using examples from a fixed window of iterations also speeds up training. Figure 2b shows a comparison of the importance of these heuristics for efficient and accurate training—empirically, data augmentation for stabilizing surrogate training allows us to scale to neural networks with 100s of nodes. MLPs and GRUs of this size are already sufficient for many real problems, such as those we encounter in healthcare domains.

5. Demonstration: A Tree-Regularized MLP and RNN

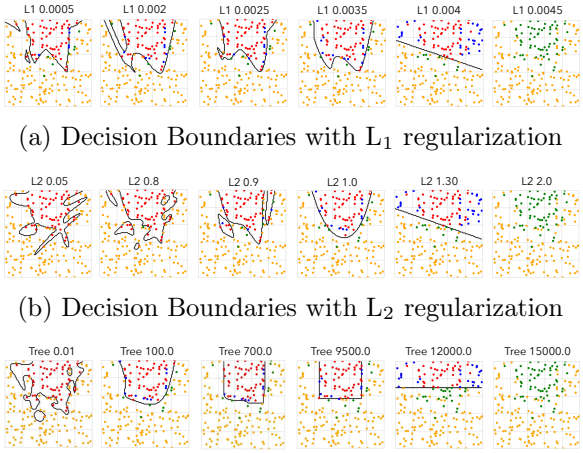
We start by exploring two simple domains intended to build intuition for the tree regularization method. We first test the regularizer on MLPs in a two-dimensional classification task followed by a second prediction task with sequential data.



(a) 2D Parabola (b) Prediction vs Complexity for many λ

Figure 3: *(a)* A visualization of the 2D parabola dataset. The black line shows the true decision boundary; the gray lines define areas where noise is added. *(b)* A comparision of APL versus AUC for many regularizers. In the small average path length regime (0-5), tree-regularization produces models with higher AUC than L_1 or L_2.

Tree-Regularized MLP: Noisy Parabola We first show a binary classification task as demonstration. We call this task the *2D Parabola problem*, because as Figure 3a shows, the training data consists of 2D input points whose two-class decision boundary is a parabola defined by $y = 5 * (x - 0.5)^2 + 0.4$. We sampled 500 input points $\mathbf{x}_n$ uniformly within the unit square $[0, 1] \times [0, 1]$ and labeled those above the decision function as positive. To make it easy for models to overfit to more complex decision boundaries, we flipped 10% of the points in a region near the boundary. A random 30% were held out for testing. For the classifier, we train a 3-layer MLP with 100 first layer nodes, 100 second layer nodes, and 10 third layer nodes. This MLP is intentionally overly expressive to encourage overfitting and expose the impact of different forms of regularization: tree regularization $\Psi(\theta) = \hat{\Omega}(\theta)$, L_2 penalty $\Psi(\theta) = ||\theta||_2$, and L_1 penalty $\Psi(\theta) = ||\theta||_1$. For each regularizer, we train models at many different strengths λ chosen to explore the full range of decision boundary complexities possible under each technique. For tree regularization, we model a surrogate $\hat{\Omega}(\theta)$ with a 1-hidden layer MLP with 25 units. The surrogate is intentionally chosen to be small with few parameters. In practice, we bias towards simpler surrogate networks to ensure faster training. Additionally, too complex of a surrogate would no longer preserve intepretability. Equation 1 is optimized via Adam (Kingma & Ba, 2014) using a batch size of 100 and a learning rate of 1e-3 for 250 epochs. These hyperparameters were found with grid search.

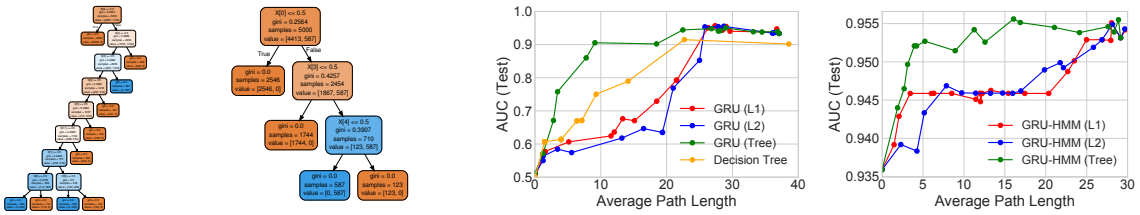


(a) Decision Boundaries with L_1 regularization

(b) Decision Boundaries with L_2 regularization

(c) Decision Boundaries Tree regularization

Figure 4: Decision boundaries (black lines) have qualitatively different shapes for different regularization schemes, as regularization strength λ increases. We color each prediction as true positive (red), true negative (yellow), false negative (green), and false positive (blue). The L_1 boundary appears more sharp, whereas L_2 is more round, and tree reg. is axis-aligned.



(a) GRU $\lambda = 1$ (b) GRU $\lambda = 1\,000$ (c) GRU (d) GRU-HMM

Figure 5: *Toy Signal & Noise Task: (a)-(b)* Decision trees trained to mimic predictions of GRU models at different regularization strengths λ; as expected, increasing λ decreases the size of the learned trees. Decision tree (b) suggests the model learns to predict positive output (blue) if and only if "x[0] == 1 and s[3] == 1 and s[4] == 0". This simple description is consistent with the true rule used to generate labels for our dataset: assign positive label only if first dimension is on (x[0] == 1) and first state is active (the emission probability vector for this state is: [.5 .5 .5 .5 0 ...]). *(c,d)* Tree regularization produces simpler models (as measured by APL) with higher prediction quality (AUC) across range of regularization strengths λ for the GRU (c) and GRU-HMM (d).

To evaluate simulability, we coincidentally use APL. Since Algorithm 1 can compute the APL for *any* fixed deep model given its parameters, we can also use it to measure decision boundary complexity under any regularization, including L_1 or L_2. Figure 4b shows each trained model as a single point in a 2D fitness space: the x-axis measures model complexity with APL, and the y-axis measures AUC (area under the ROC curve). These results show that simple L_1 or L_2 regularization does *not* produce models with both small node count and good predictions at *any* value of the regularization strength λ. As expected, large λ values for L_1 and L_2 only produce far-too-simple decision boundaries with poor accuracies. In contrast, tree regularization directly optimizes the MLP to have simple tree-like boundaries at high λ values which can still yield good predictions. The lower panes of Figure 4 show these boundaries. Tree regularization is uniquely able to create *axis-aligned* functions, because decision trees by definition learn axis-aligned splits. Critically, these axis-aligned functions require very few conceptual nodes but are more effective than L_1 and L_2 regularizers.

Tree Regularized GRU: Toy Signal & Noise Next, we analyze the performance of tree regularization on synthetic timeseries data. We generated a toy data set of $N = 100$ sequences, each with $T = 50$ timesteps. Each timestep has a data vector $\mathbf{x}_{nt}$ of 14 binary features and a single binary output label $\mathbf{y}_{nt}$. The data comes from two separate HMM processes. First, a "signal" HMM generates the first 7 data dimensions from 5 well-separated states. Second, an independent "noise" HMM generates the remaining 7 data dimensions from a different set of 5 states. The transition and emission matrices for both HMMs are shown in Figure 6. The probabilities were chosen to make it difficult for a new HMM to learn. Each timestep's output label $\mathbf{y}_{nt}$ is produced by a rule involving *both* the signal HMM's generated observations and the signal HMM's hidden state: the target is 1 at timestep t only if both the first signal state is active and the first observation is turned on. We deliberately designed the generation so that neither logistic regression on inputs $\mathbf{x}_n$ alone nor a GRU model that makes predictions from hidden states alone can perfectly separate this data.

$$\begin{pmatrix} .5 & .5 & .5 & .5 & 0 & 0 & 0 \\ .5 & .5 & .5 & .5 & .5 & 0 & 0 \\ .5 & .5 & .5 & 0 & .5 & 0 & 0 \\ .5 & .5 & .5 & 0 & 0 & .5 & 0 \\ .5 & .5 & .5 & 0 & 0 & 0 & .5 \end{pmatrix} \quad \begin{pmatrix} .7 & .3 & 0 & 0 & 0 \\ .5 & .25 & .25 & 0 & 0 \\ 0 & .25 & .5 & .25 & 0 \\ 0 & 0 & .25 & .25 & .5 \\ 0 & 0 & 0 & .5 & .5 \end{pmatrix} \quad \begin{pmatrix} .5 & .5 & .5 & 0 & 0 & 0 & 0 \\ 0 & .5 & .5 & .5 & 0 & 0 & 0 \\ 0 & 0 & .5 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & .5 & .5 & .5 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 & .5 \end{pmatrix} \quad \begin{pmatrix} .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \end{pmatrix}$$

(a) Signal: Emission (b) Signal: Transition (c) Noise: Emission (d) Noise: Transition

Figure 6: Emission (5 states vs 7 features) and transition probabilities for the signal HMM (a, b) and noise HMM (c, d). We emphasize that to output 1, the signal HMM must be in the first state and the first input feature must be 1.

As with the MLP, each regularizer (tree, L_2, L_1) is applied to the output node of the GRU across a range of strength parameters λ (see orange triangle in Figure 1). In training, we used 25 hidden dimensions for GRU models and 5 states for the HMM component of the GRU-HMM. All other choices are identical to the 2D Parabola setting. Figure 5 compare the performance of regularized GRU and GRU-HMM models. Since we can no longer easily visualize the decision boundary, we rely on plots like Figure 5(c, d) to measure performance. Many of the same patterns from the 2D Parabola experiments emerge here: tree regularized GRU models achieve much higher (held-out) AUC at lower APL. Further, L_1 and L_2 are quite unreliable at high regularization strengths, doing worse than a decision tree at low APL. All regularized models converge to the same performance as APL approaches 0 (random choice) and infinity (unregularized). Additionally, we include results for the GRU-HMM (d) whose performance is lower bounded by the performance of a standalone HMM (notice the scale of the y-axis). As before, tree regularization on the "GRU component" of the GRU-HMM approaches maximum performance with small APL (around 5). We hypothesize this to be due to the compactly expressive nature of axis-aligned boundaries.

Finally, Figure 5(a,b) show two "distilled" decision trees that are used to approximate the deep model in the last epoch of training. We can see that for small regularization strengths (a), the distilled tree is large and difficult to interpret. For larger strengths (b), the tree recovers the true generative process: predict positive output if and only if "x[0] == 1 and s[3] == 1 and s[4] == 0". The first component (x[0] == 1) represents the first observation being 1; the second component (s[3] == 1 and s[4] == 0) represents the first state being active (recall that the emission distribution for this is state is [.5 .5 .5 .5 0 ...]). A decision tree like this can be given to a human to help describe what mappings the deep model as learned. Critically, smaller decision trees are very easy to simulate.

6. Applications: Real-World Timeseries Data

Having explored a few synthetic environments, we now evaluate the tree regularizer on several real-world timeseries models in speech recognition and two sectors of healthcare. For each experiment below, we will compare a tree regularized GRU with an identical GRU regularized with L_1 or L_2. We will also include a decision tree baseline where a tree classifier is fit directly on the observations. Additionally, we will compare the GRU results with GRU-HMM performance to gauge any benefits of residual training. For optimization, we use Adam with a learning rate of 1e-3, a batch size of 256, decision tree hyperparameter $h = 1000$, train for

300 epochs, surrogate datasets of size $J = 100$, and retrain every 25 steps. Like above, we measure performance with AUC and simulability with APL for all models. Before sharing results, we briefly describe each task and domain.

6.1 Tasks

We tested our approach on several real-world tasks: predicting medical outcomes of hospitalized septic patients, HIV therapy outcome prediction, and predicting stop phoneme groups from a selection of English speech recordings. To normalize scales, we independently standardized input features via z-scoring.

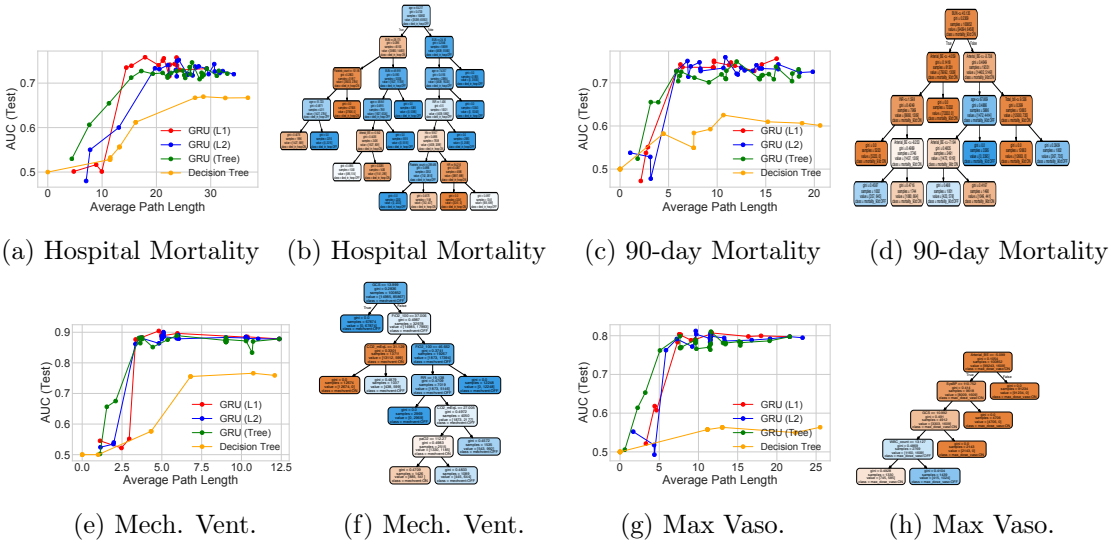
- *Sepsis Critical Care (ICU)*: We study timeseries data for 11,786 septic ICU patients from the public MIMIC III dataset (Johnson, Pollard, Shen, Lehman, Feng, Ghassemi, Moody, Szolovits, Celi, & Mark, 2016). We observe at each hour (timestep) t a data vector $\mathbf{x}_{nt}$ of 35 vital signs and lab results as well as a label vector $\mathbf{y}_{nt}$ of 5 binary outcomes. Hourly data $\mathbf{x}_{nt}$ measures continuous input features such as respiration rate (RR), blood oxygen levels (paO_2), fluid levels, and more. Hourly binary labels $\mathbf{y}_{nt}$ include whether the patient died in hospital, whether the patient died after 90 days, and if mechanical ventilation was applied. Models are trained to predict all 5 output dimensions concurrently from one shared embedding. The average sequence length is 15 hours. 7,070 patients are used in training, 1,769 for validation, and 294 for test.

- *HIV Therapy Outcome (HIV)*: The EuResist Integrated Database (Zazzi, Incardona, Rosen-Zvi, Prosperi, Lengauer, Altmann, Sonnerborg, Lavee, Schulter, & Kaiser, 2012) describes 53,236 patients diagnosed with HIV. We consider 4-6 month intervals as time steps. Each data vector $\mathbf{x}_{nt}$ has 40 features, including blood counts, viral load measurements and lab results. Each output vector $\mathbf{y}_{nt}$ has 15 binary labels, including whether a therapy was successful in reducing viral load to below detection limits, if therapy caused CD4 blood cell counts to drop to dangerous levels (indicating AIDS), or if the patient suffered medical adherence issues. The average sequence length is 14 steps. 37,618 patients are used for training, 7,986 for testing, and 7,632 for validation.

- *Phonetic Speech (TIMIT)*: Timeseries data containing broadband recordings of 630 speakers of American English reading ten phonetically rich sentences (Garofolo et al., 1993). Each sentence contains time-aligned phonetic transcriptions of 60 phonemes. We focus on the problem of distinguishing stop phonemes (those that stop the flow of air, such as "b", "d", or "g") from non-stops. Each timestep has one binary output $\mathbf{y}_{nt}$ indicating whether a stop phoneme occurs. There are 26 continuous features for each input vector $\mathbf{x}_{nt}$ representing the Mel-frequency cepstral coefficients and derivatives of the acoustic signal. There are 6,303 sequences: which we split into 3,697 for training, 925 for validation, and 1,681 for testing. The average length is 614 tokens.

6.2 Results and Analysis

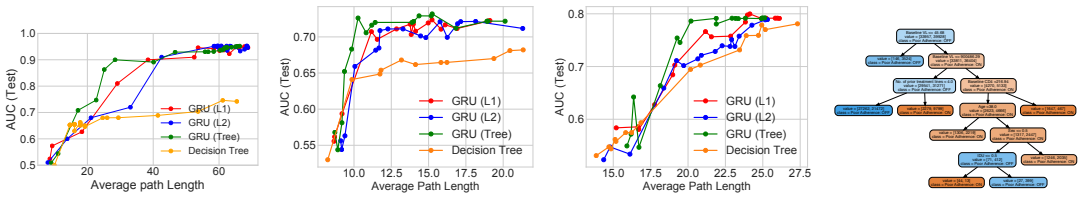
The results on ICU, HIV, and TIMIT share many consistent characteristics. We summarize the many experiments with analysis on common patterns and provide a few takeaways.



(a) Hospital Mortality (b) Hospital Mortality (c) 90-day Mortality (d) 90-day Mortality

(e) Mech. Vent. (f) Mech. Vent. (g) Max Vaso. (h) Max Vaso.

Figure 7: *Sepsis Critical Care Task* – Study of different regularizers for a GRU model with 100 states, trained to jointly predict 5 binary outcomes. Panels *(a,c,e,g)* show AUC vs. APL for 4 of the 5 outcomes; in all cases, tree regularization provides higher accuracy in the target regime of low-complexity decision trees. Panels *(b,d,f,h)* show the associated decision trees for $\lambda = 2\,000$; these were found by clinically interpretable by an ICU clinician.



(a) TIMIT “Stop" (b) HIV: CD4$^+$ (c) HIV Therapy (d) HIV Therapy

Figure 8: *TIMIT and HIV Tasks:* – Study of different regularizers for a GRU model with 75 states. Panels *(a)-(c)* are tradeoff curves showing how predictive power and decision-tree complexity evolve with increasing strength of L_1, L_2 or tree regularization in both TIMIT (stop phoneme prediction) and HIV (CD4$^+$ $\leq$ 200 cells/ml and therapy adherence prediction). The TIMIT task has only one binary outcome. However, for the HIV task, the GRU is trained to jointly predict 15 binary outcomes, of which 2 are shown here in Panels *(b)-(c)*. The decision tree associated with HIV adherence is shown in *(d)*.

Tree regularized models have fewer nodes than other forms of regularization. Across tasks, we see that in the target regime of small decision trees (low APLs), tree regularization achieves higher prediction quality (higher AUCs). In the toy signal/noise task, tree regularization (green line in Figure 5d) achieves AUC values near 0.9 when its trees have an APL of 10. Similar models with L_1 or L_2 regularization reach this AUC only with trees that are nearly double in complexity (APL over 25). On both Sepsis (Figure 7) and TIMIT (Figure 8a), we observe considerable gains in accuracy over other regularizers (AUC

Dataset	Fidelity
signal-and-noise HMM	0.8762
Sepsis (In-Hospital Mortality)	0.8144
Sepsis (90-Day Mortality)	0.8845
Sepsis (Mech. Vent.)	0.9008
Sepsis (Median Vaso.)	0.9166
Sepsis (Max Vaso.)	0.9260
HIV ($CD4^+$ below 200)	0.8426
HIV (Therapy Success)	0.8761
HIV (Mortality)	0.9318
HIV (Poor Adherence)	0.9014
HIV (AIDS Onset)	0.9344
TIMIT	0.8477

Table 1: Fidelity of predictions from our trained deep GRU and its corresponding decision tree. Fidelity is defined as the percentage of test examples on which the prediction made by a tree agrees with the deep model (Craven & Shavlik, 1996).

Dataset	Model	Epoch Time
Sepsis	HMM	589.8 ± 24.1
Sepsis	GRU	822.3 ± 11.2
Sepsis	GRU-HMM	1666.9 ± 147.0
Sepsis	GRU‡	2015.1 ± 388.1
Sepsis	GRU-HMM‡	2443.7 ± 351.2
TIMIT	HMM	1668.9 ± 126.9
TIMIT	GRU	2116.8 ± 438.8
TIMIT	GRU-HMM	3207.2 ± 651.9
TIMIT	GRU‡	3977.0 ± 812.1
TIMIT	GRU-HMM‡	4601.4 ± 805.9

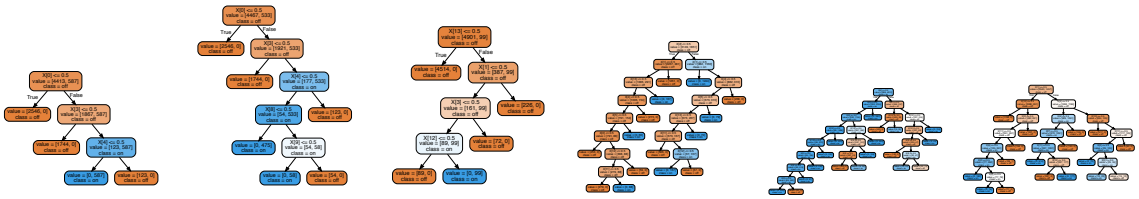
Table 2: Training time for a single epoch in seconds on a single Intel Core i5 CPU. The (‡) symbol represents using tree regularization. The times for tree regularized models include surrogate training expenses. If we retrain sparsely, then the cost is amortized to close to negligible.

differences of 0.05 to 0.15) for APLs of 20-30. On the HIV task in Figure 8b, we see AUC differences of between 0.03 and 0.15 for APLs of 10-15. Similarly, on the other HIV outcomes in Figure 8c-8d, we see AUC differences of between 0.03 and 0.09 for APLs of 20-30. These gains are particularly useful in determining how to administer subsequent therapies. More specifically, in domains where human-simulability is required, these increases in accuracy in the small-complexity regime can mean the difference between models that provide value on a task and models that are unusable, either because their performance is too poor or they are uninterpretable. We emphasize that across all tasks, standalone decision trees (marked by yellow dots in line plots) cannot reach this high-accuracy, low-complexity sweet spot, suggesting that tree regularization still enables neural networks to be nonlinear.

Our learned decision-tree-like boundaries are interpretable. Recall that a consequence of tree regularization is a distillation of the deep model as a decision tree. Across all tasks, these trees which mimic the predictions of tree regularized deep models are small enough to simulate by hand and help users grasp the model's nonlinear prediction logic. We have already seen this to be the case for the toy signal/noise task. Similarly, in Figure 7, we show decision trees for two sepsis prediction tasks. We consulted a clinical expert on sepsis treatment, who noted that the trees helped him understand what the models might be doing and thus determine if he would trust the deep model. For example, he said that using FiO_2, RR, CO_2 and paO_2 to predict need for mechanical ventilation (Figure 7f) was sensible, as these all measure breathing quality. In contrast, the in-hospital mortality tree (Figure 7b) predicts that some young patients with no organ failure have high mortality rates while other young patients with organ failure have low mortality. These counter-intuitive results led to

hypotheses about how uncaptured variables impact the training process. Such reasoning would not be possible from simple sensitivity analyses of the deep model. Moreover, we also found the distilled trees for HIV (Figure 8d) to be interpretable. We observed that the baseline viral load and number of prior treatment lines are crucial factors in predicting whether a patient will suffer adherence issues. This is consistent with several medical studies which show that patients with higher viral loads at baseline tend to have faster disease progression, and hence have to take several drug cocktails to potentially combat resistance. This typically makes it more difficult for these patients to adhere to the medication.

Practical runtimes for tree regularization are less than twice that of simpler L2. While our tree regularized GRU with 10 states takes 3977 seconds per epoch on TIMIT, an equivalent L_2 regularized GRU takes 2116 seconds per epoch. Thus, our new method has cost less than twice the baseline *even when the surrogate is serially computed*. Because the surrogate, $\hat{\Omega}$, will in general be a much smaller model than the target neural model, we expect one could get much smaller per-epoch times by parallelizing both the creation of $(\theta, \Omega(\theta))$ dataset and also the surrogate training. Additionally, 3,977 seconds includes the time needed to train the surrogate. In practice, we do this sparingly, only once every 25 epochs, yielding an amortized per-epoch cost of 2,191 seconds. More exhaustive runtime results with standard deviations over 10 epochs are in Table 2.

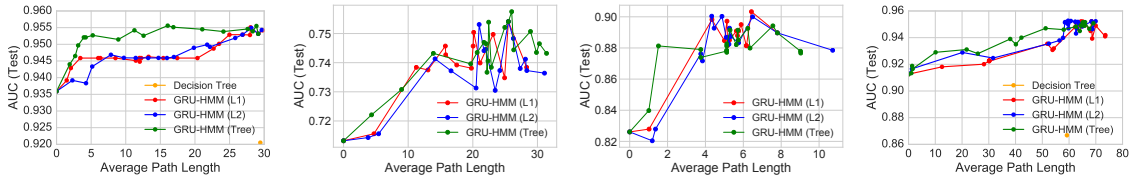


(a) 7 of 10 runs (b) 2 of 10 runs (c) 1 of 10 runs (d) 1 of 10 runs (e) 1 of 10 runs (f) 1 of 10 runs

Figure 9: *(a-c)* Decision trees from 10 independent runs on the toy signal/noise dataset with $\lambda = 1000.0$. Seven of the ten runs resulted in a tree of the same structure. The other three are similar, having additional subtrees but sharing the same splits and features. *(d-f)* Similar experiment with $\lambda = 0.01$. Low regularization causes high variance in tree size and shape. Sub-figures (d-f) show three of many variations.

Decision trees are stable over multiple optimization runs. When tree regularization is strong (high λ), the decision trees trained to match the predictions of deep models are stable. For both the toy signal/noise and Sepsis tasks, multiple runs from different random restarts have nearly identical tree shape and size, perhaps differing by a few nodes. This stability is crucial to building trust in our method. On the toy signal/noise task ($\lambda = 7000$), 7 of 10 independent runs with random initializations resulted in trees of exactly the same structure, and the others closely resembled those sharing the same subtrees and features. On the other hand, with weak regularization (small λ), variability in the distilled decision trees is high. See Figure 9 for example trees under strong (a-c) and weak (d-f) regularization.

Target neural models are faithful to decision trees. *Fidelity* is defined by Craven and Shavlik (1996) as the percentage of examples where the prediction of the target network



(a) Toy S&N $_{20+5 \text{ states}}$ (b) Mortality $_{50+50}$ (c) Mech. Vent. $_{50+50}$ (d) TIMIT "Stop" $_{50+25}$

Figure 10: Fitness curves for the GRU-HMM, showing prediction quality (AUC) vs. complexity (APL) across range of regularization strengths λ. Captions show the number of HMM states plus the number of GRU states. See Figures 7 and 8 to compare these GRU-HMM numbers to simpler GRU and decision tree baselines.

and the decision tree agree. Thus, fidelity is a measurement of how faithful the deep network is to the distilled tree. A fidelity of 1 would indicate perfect agreement, in which the neural network has learned exactly the axis-aligned boundaries of a tree. In some sense, a fidelity of 1 is undesirable as we hope the deep network can make use of nonlinearity on the examples that a simulable tree would struggle with. Table 4 shows that the fidelity is high but not perfect, ranging from 0.80 to 0.94 across datasets.

The deep residual GRU-HMM can achieve high AUC with less complexity. In Figure 10, we show the performance of jointly training the residual model, GRU-HMM, which combines an HMM with a tree regularized GRU to improve its predictions. Here, the ideal APL is zero, indicating only the HMM makes predictions (only the GRU output node is regularized). For small APLs, the GRU-HMM substantially improves the original HMM's predictions *and* has simulability gains over earlier GRUs. On the mechanical ventilation task, the GRU-HMM requires an APL of only 28 to reach AUC of 0.88, while the GRU alone with the same number of states requires a path length of 60 to reach the same AUC. This suggests that jointly-trained deep residual models may provide better interpretability.

7. Applications: Image Classification

Next, we explore the use of tree regularization in image classification on two standard benchmarks: MNIST (LeCun, 1998) and FashionMNIST (Xiao, Rasul, & Vollgraf, 2017). We wish to regularize deep neural networks operating on pixels based on the APL of an approximating decision tree. Naively, we may think to fit decision trees on the raw image pixels themselves. However, one of the strengths of the proposed method was the interpretability of the decision nodes in the tree such that experts could understand them. This was true in the healthcare domain as the input features themselves were interpretable. But in the context of images, this naive approach does not work as a decision tree trained on pixels is not simulable.

Because raw pixels are unsuitable as features, we fit decision trees on region proposals that divide the image into bounding boxes of different sizes. More precisely, for an image x, we tile the image with n^2 disjoint bounding boxes to produce a set of image crops $\{b_i(x)\}_{i=1}^{n^2}$. For each bounding box $b_i(x)$, we convolve it with a fixed filter of the same dimensions to produce a scalar, total n^2 scalars. We then vary n from 2 (for a total of 4 bounding boxes)

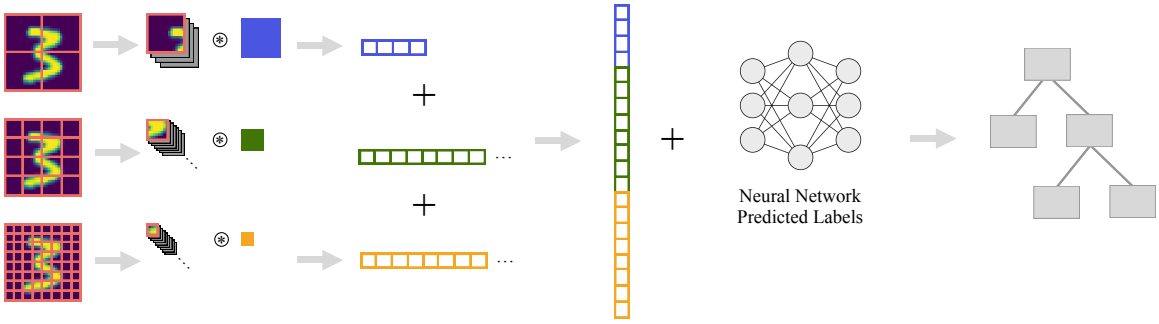
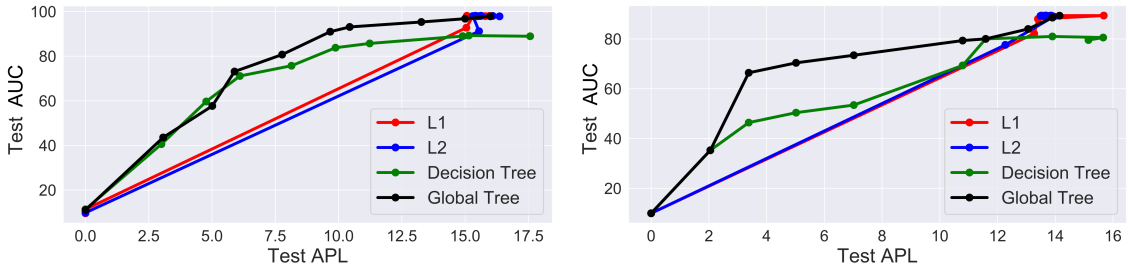


Figure 11: To ensure interpretable features, we distill neural networks into decision trees trained on features extracted from bounding boxes on an input image. We divide the region into disjoint sections, each of which is convolved with a fixed filter to produce a vector of scalars. By varying the resolution of the bounding boxes, we featurize crops of different sizes. These features are all concatenated as the final input to the decision. Critically, each of these final features can be traced back to a region in the original image.

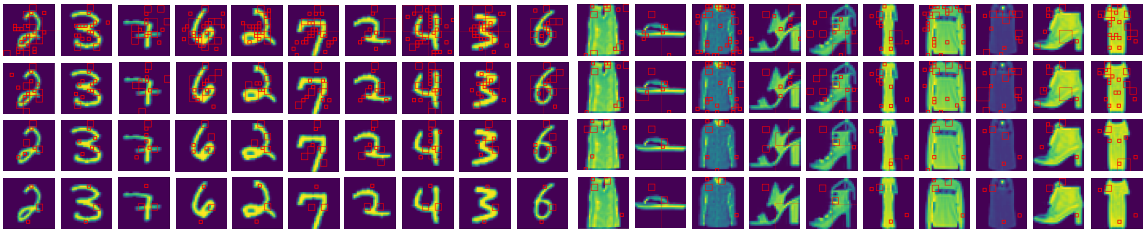


(a) MNIST (b) FashionMNIST

Figure 12: Fitness curves comparing L_1, L_2, and global tree regularization on MNIST (a) and FashionMNIST (b). The green line represents a decision tree baseline. We observe global tree to achieve higher accuracies at lower APLs, consistent with prior experiments.

to 4, 8, 16, etc to featurize crops of different sizes. Finally, the scalar features for each n are concatenated and fed to a decision tree. Thus, when a decision tree chooses a particular feature, it can be mapped to a crop of the original image. For large enough crop sizes, the bounding boxes are interpretable to humans. We summarize the procedure in Figure 11. Note that while all approximating decision trees operate over bounding boxes, the deep neural network still operates over raw pixels.

Results Figure 12 plot the APL versus accuracy on a held-out test set for MNIST and FashionMNIST. We optimize a deep neural network with an L_1, L_2, or tree regularization penalty. For each of these, we vary the strength of the regularization to find different minima. As in prior experiments, we also include a decision tree baseline, in which we vary the minimum number of samples to define a leaf node to fit smaller and larger trees. We make a few observations from Figure 12: First, L_1 and L_2 are poor regularizers — this was true in previous experiments but is exaggerated in the visual domain: increasing the strength of these two regularizations has little effect until it reaches pivotal value that causes the



(a) MNIST (b) FashionMNIST

Figure 13: Visualization of relevant bounding boxes in a decision tree distilled from a tree-regularized neural network for MNIST (a) and FashionMNIST (b). Each column shows a different randomly selected image. The lower rows use more highly regularized neural networks, attributing to the sparser bounding boxes.

network to collapse, trivially reaching 0.0 APL and chance accuracy. We find that although an unregularized deep network network outperforms the decision tree baseline (by more than 10% in both datasets), the decision tree is able to find many more minima at lower non-zero APLs. Next, we find the tree regularized model to achieve the best of both worlds: at low strengths the performance converges to an unregularized model but at high strengths performance converges to that of the baseline. Critically, in the low to middle APL region, we see higher performance than any of the other models, especially in FashionMNIST.

Recall that every tree regularized deep neural network produces a distilled decision tree throughout training. So, we can interpret the neural network by visualizing what sections of an image its distilled decision tree is attending to. Because our decision trees are trained on crops of the original image, it is easy to associate every feature to a collection of pixels. Figure 13 shows 10 randomly chosen examples. The red boxes represent boxes the distilled tree used to classify this image. For each image, we show four versions (each row) where the strength is increased for lower rows. Thus, since the neural network is regularized to be simpler, we observe the distilled decision function to attend to fewer areas of the image, focusing on more critical sections. Qualitatively, the decisions functions seem sensible: the red boxes often focus on prominent features of a digit or article of clothing e.g. curves or object components. Further, many of the boxes also focus on blank areas as they are no less important for distinguishing classes (for example, a "7" from a "9"). These visualizations serve as an explainable artifact that computer vision practitioners and researchers can use to understand the decision function captured by their target network. Unlike related methods e.g. GradCAM (Selvaraju et al., 2017), users can use tree regularization to directly optimize their image classifiers to be interpretable.

Training Details All images are resized to 32 by 32 and normalized using dataset statistics. No additional image augmentations are used. To featurize a region of size $h \times w$, we convolve each box of size with a filter of size $h \times w$ with each entry set to $\frac{1}{hw}$ — this is equivalent to averaging the pixels in the region. The primary neural network is a MLP containing 3 layers, hidden dimension 128, and GELU nonlinearity (Hendrycks & Gimpel, 2016) after the first two layers. For global tree, the surrogate network is another MLP with 3 layers with hidden dimensions of 128, 32, and 16, respectively. After each layer, we add GELU and batch normalization (Ioffe & Szegedy, 2015). We use the Adam optimizer with learning

rate 3e-4, batch size 128, for 30 epochs in training the target neural network. To fit the surrogate, we use Adam with learning rate 1e-3, batch size 256, weight decay 1e-4 for 50 epochs. The surrogate is retrained every epoch of optimizing the main neural network. When measuring average path length, we always separately fit a new decision tree with a minimum sample of 1 to define a leaf node. For the decision tree baseline, we choose the minimum sample threshold from 1, 5, 10, 50, 100, 500, 1000, 5000, 10000, 50000 (note we still fit a new tree with the leaf threshold set to 1 when measuring APL). For all regularizers, we explore strengths of 0, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1.0.

Future Work Although we find promising results on MNIST, we acknowledge that this setup is only a first step towards tree regularization for visual models, leaving many improvements for future research. For instance, more sophisticated approaches like multiple filters per bounding box, or using a pretrained region proposal network (Ren, He, Girshick, & Sun, 2015) are natural extensions. A third direction might explore non-rectangular regions proposals that capture objects in the image (e.g. segmentation maps).

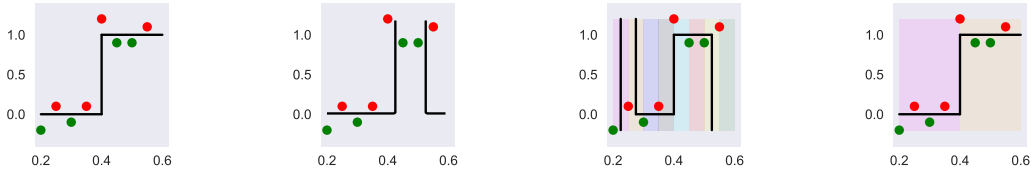
8. Regionally Faithful Explanations

Global summaries such as L_1, L_2, or even tree regularization as presented above face a tough trade-off between human-simulability and being faithful to the underlying model. For instance, if we require a minimum fidelity of 0.95, it simply may not be possible to fit a faithful decision tree that is also human-simulable. For a complex enough domain (or for particularly difficult examples), it is unreasonable to assume that there is a decision tree can be small and faithful. In such a case, tree regularization of a deep network may not be able to find a good compromise between accuracy and complexity. To get the best of both worlds, we will need a *finer-grained* definition of interpretability. Doing so might help find a new wealth of minima with high AUC and low APL (aka powerful yet simulable).

To start, we take advantage of the fact that domain experts may already have notions about how regions of the input space operate differently. For example, a clinical intensivist may already cognitively consider patients in the surgical intensive care unit (ICU) as different from patients in the cardiac ICU. Analogously, biologists may be happy with different models for classifying diseases in deciduous versus in coniferous plants. In fact, this way of partitioning thinking into independent compartments is a very general phenomena. Cognitive science literature tells us that people build context-dependent models of the world; they do not expect the same rule to apply in all circumstances (Miller, 2018).

Using this intuition, we divide the input space into exclusive regions. We assume that this division is available *a priori* via domain knowledge. In fact, this is a good opportunity to inject expert priors into the model. Formally, this translates into R exclusive regions $\mathcal{X}_1, \dots \mathcal{X}_R$, where $\cup_{r=1}^{R} \mathcal{X}_r \subseteq \mathbb{R}^p$. We denote the observed dataset belonging to region r as $X_r = \{\mathbf{x}_n : \mathbf{x}_n \in \mathcal{X}_r\}$. Thus, we shall apply a *regionally-faithful regularization* that encourages the target neural model to be "simple" in *every* region (where a region corresponds to a human context). This partitioning of the input space into regions allows a regularized neural model to approximate very complex decision boundaries with simple components, thereby remaining simulable. We emphasize that our regional explanations are distinct from local explanations (Ribeiro et al., 2016): the latter concerns itself with behavior within an ϵ-ball around a single example and makes no claims about general behavior across examples. In contrast,

regional explanations are faithful over a subspace. As a preview, Figure 14 highlights the



(a) True (b) Global (c) Local (d) Regional

Figure 14: We show the differences between global (b), local (c), and regional (d) tree regularization using a synthetic classification task. (a) shows the true decision boundary. Red and green points represent the training dataset. Lightly colored areas represent regions. In (b), the model is over-regularized and ignores underlying structure. In (c), regions are made as small as possible to simulate locality—resulting in highly variable rules for nearby points. Regional tree regularization (d) provides an interpretable middle ground.

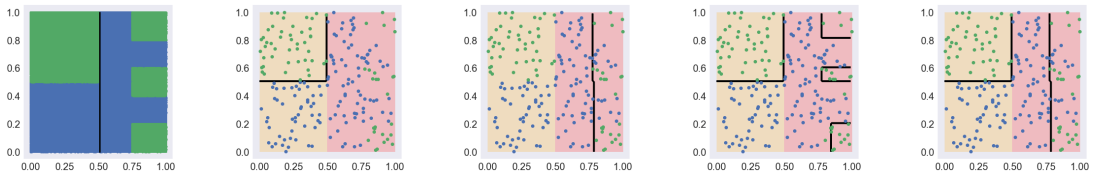
distinctions between global, local, and regional tree regularization on a two-dimensional toy dataset where the true decision boundary is divided in half at $x = 0.4$. We see that global explanations (b) lack information about the input space and have to choose from a large set of possible solutions, converging to a different boundary. On the other hand, local explanations (c) produce simple boundaries around each data point but fail to capture global relationships, resulting in a complex overall decision function. Finally, regional explanations (d) over two regions divided at 0.4 share the benefits of (b) and (c).

8.1 Regional Tree Regularization

We introduce regional tree regularization, which will require that the target neural model $f(\cdot;\theta)$ is well-approximated by a separate compact decision tree in *every* region. In contrast, we will rename the tree regularizer presented above as *global tree regularization*. Regionally simple decision boundaries are particularly hard to achieve with global tree regularization as the global APL metric may allow some human-relevant regions to be complex as long as most are simple. In particular, global tree regularization has an incentive to "ignore" simpler regions in order to minimize the regularization term (i.e. trivially predict a single label). In many contexts, this behavior is undesirable. For example, if a clinician splits his/her patients by severity of illness, regularizing for simple global explanations can completely ignore a group of patients, rendering the machine learning system useless. To ensure that some regions cannot be made simple at the expense of others, we propose the novel regularizer:

$$\Omega^{\texttt{regional}}(\theta) \triangleq \texttt{max}_r(\{\Omega_r(\theta)\}_{r=1}^R) \text{ where } \Omega_r(\theta) = \text{APL}(\mathcal{X}_r, f(\cdot;\theta)). \tag{10}$$

Equation 10 is a L_0 norm over $\{\Omega_r\}_{r=1}^R$. The choice of L_0 produces different (and desirable) behavior than if we had simply used, for example, L_1, which is equivalent to simply regularizing APL in a global tree that first branches by region. As a nonlinear regularizer, L_0 keeps *all* regions simple, while not penalizing regions that are already simple. We show an example in Figure 15: (a) shows a toy dataset with two regions (split by the black line): the left has a simple decision boundary dividing the region in half; the right has a more complex



(a) True (b) L_1 (c) L_1 (d) L_0 (e) L_0

Figure 15: An L_1 penalty on per-region APLs can over-penalize, resulting in an entire region with far too simple predictions. Subplots (b) and (c) show results from two different initializations using the L_1 norm, while (d) and (e) show the same using the L_0 norm.

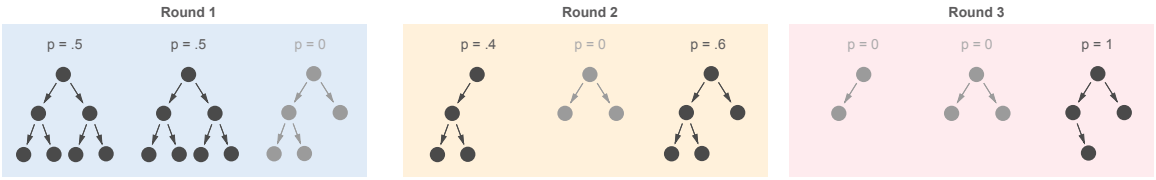


Figure 16: Illustration of L_0 regional tree regularization. Each round contains three trees representing regions. Light gray color indicates regions ignored by `sparsemax`: these regions do not contribute to the objective function. Over the three rounds, different regions are given priority while other regions are given no weight. The ability to disregard regions of low complexity makes for smoother learning.

boundary. Figure 15 (b) and (c) then show two minima using L_1. In both cases, one of the regions collapses to a trivial decision boundary to minimize the overall sum of APLs. On the other hand, since L_0 is sparse, simple regions are not included in the objective, resulting in more "balanced" boundaries across regions. Unfortunately, the `max` operation is not easy to differentiate. While we could utilize a straight-through estimator, we found this to face optimization challenges in real world contexts. Namely, learning was slow and prone to being trapped in "oscillatory minima" where optimization alternates between minimizing the APL of one region at the cost of increasing APL of the other. Ideally, we would like to get the sparsity of the `max` but consider more than a single region at once.

A continuous approximation to `max` that is also sparse is a tall order. Common differentiable approximations like `softmax` are dense, which makes it difficult to focus on the most complex regions as `max` does. Thus, we must use the recently-proposed `sparsemax` (Martins & Astudillo, 2016), which encourages sparsity while remaining smooth and differentiable almost everywhere. Intuitively, `sparsemax` corresponds to a Euclidean projection of an input vector $\hat{\mathbf{\Omega}}$ with R entries to an R-length vector $\mathbf{p}$ of non-negative entries that sums to one (i.e. the $(R-1)$-dimensional probability simplex). When the projection lands on a boundary in the simplex (which is likely), then the resulting vector will be sparse. Efficient implementations of this projection are well-known (Duchi, Shalev-Shwartz, Singer, & Chandra, 2008), as are Jacobians for automatic differentiation (Martins & Astudillo, 2016). We refer to using `sparsemax` in Equation 10 as L_0 regional tree regularization. In practice, we find `sparsemax` to be produce more stable and faster optimization than `max`.

We also note that the regional APL $\Omega_r(\theta)$ is also not differentiable just like in the global case. We again employ surrogate networks $\hat{\Omega}_r^{\texttt{regional}}$: that map a parameter vector $\theta \in \Theta$ to an *estimate* of $\Omega_r(\cdot)$, the APL in region r. This process is identical but restricted to observations lying in region r. Each surrogate $\hat{\Omega}_r$ has its own parameters ϕ_r. We fit each surrogate by minimizing $\min_{\phi_r} \left(\sum_{j=1}^{J} (\Omega_r(\theta_j) - \hat{\Omega}_r(\theta_j, \phi_r))^2 \right)$ for each $r = 1, ..., R$ where θ_j is sampled from a data set of J known parameter vectors and their true APLs, $\mathcal{D}_r^\theta = \{\theta_j, \Omega_r(\theta_j)\}_{j=1}^J$. For R regions, we curate one such data set for each surrogate model.

8.2 Innovations for Optimization Stability

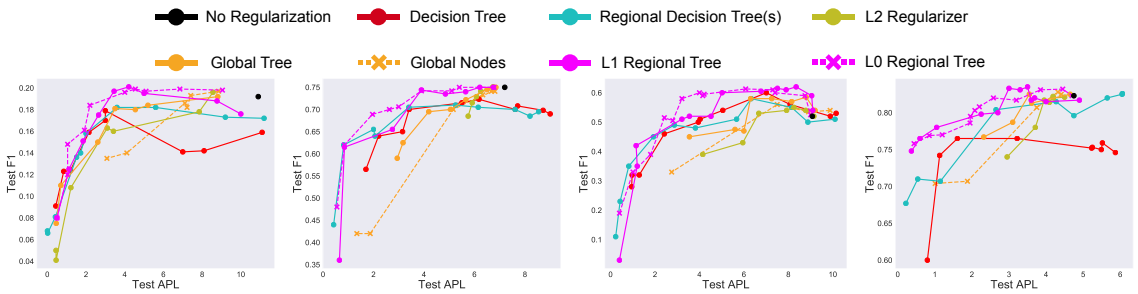
Optimizing multiple surrogate networks is a delicate operation. We found that depending on hyperparameters, the regional surrogates were unable to accurately predict the APL, causing regularization to fail. Further, repeated runs also often found different minima, making regional tree regularization feel unreliable. In short, it presents a much more difficult technical challenge than training a single surrogate as in global tree regularization. Below, we list optimization innovations that are essential to stabilize training, identify consistent minima, and achieve good APL prediction—all of which enable robust regularization.

Augmentation makes for a robust surrogate. Especially for regional explanations, relatively small changes in the underlying model can cause large changes in the learned decision function in a specific region. As such, the surrogates need to be retrained frequently (e.g. every 50 gradient steps). The practice used in global tree regularization of computing the true APL for a dataset $\mathcal{D}^\theta$ of the most recent θ is insufficient to learn the mapping from a thousand-dimensional weight vector to the APL. Using stale (very old) θ from previous epochs, however, would result in a poor surrogate model given outdated information. Previous heuristics as in random restarts or arbitrarily sampling random weights introduced more noise than signal. Thus, we supplement the dataset with randomly sampled weight vectors *from the convex hull defined by the recent weights*. Specifically, to generate a new θ, we sample from a Dirichlet distribution with J categories and form a new parameter as a convex combination of the elements in $\mathcal{D}^\theta$. For each of these samples, we compute its true APL to train the surrogate. Table 3 shows this to reduce noise.

Experiment	Mean MSE	Max MSE
No data aug.	0.069	0.987
With data aug.	0.015	0.298
Randomized	0.116	1.731
Deterministic	0.024	0.371

Table 3: Comparison of the average and max mean squared error (MSE) between surrogate predictions and true average path lengths over 500 epochs. Non-deterministic training and lack of data introduces large errors.

Decision trees should be pruned. Given a dataset, $\mathcal{D}$, even with a fixed seed, there are many decision trees that can fit $\mathcal{D}$. One can always add additional subtrees that predict the same label as the parent node, thereby not effecting performance. This invariance again introduces difficulty in learning a surrogate model. To remedy this, we use *reduced error pruning*, which removes any subtree that does not effect performance as measured on a portion of the data set not used in TrainTree. Intuitively, pruning collapses the set of possible trees describing a single classifier to a singleton.



(a) Bank (b) Gamma (c) Adult (d) Wine

Figure 17: (a-d) Comparison of regularizers (L_2, global tree, regional tree) on four datasets from the UCI repository. Each subfigure plots the average APL over 5 regions (computed on a held-out test set) against the test F1 score. The ideal model is with high accuracy and low APL i.e. the upper left diagonal of each plot. In each setting, regional tree regularized models are able to find more low APL minima than global explanations and consistently achieves the highest performance at low APL. In contrast, the performance of global tree and L_2 regularization quickly decays as the regularization strength increases.

Trees should be trained deterministically. CART is a common algorithm to train a decision tree. However, it has poor complexity in the number of features as it enumerates over all unique values per dimension. To scale efficiently, many open-source implementations e.g. Scikit-Learn (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, et al., 2011b) randomly sample a small subset of features. As such, independent training instances can lead to different decision trees of varying APL. For tree regularization, unexplained variance in APL means difficulty in training the surrogate model, since the function from model parameters to APL is no longer many-to-one. The error is compounded when there are many surrogates. To remedy this, we fix the random seed.

9. Applications: UCI Machine Learning Benchmarks

We apply regional tree regularization to a suite of four popular machine learning data sets from the UC Irvine repository (Dheeru & Karra Taniskidou, 2017). Without loss of generality, we focus on feedforward networks, or MLPs. The same ideas of regional explanation using decision trees can be trivially extended to sequential models or convolutional models. For the experiments below, we set the target neural model to a 6 layer MLP with 128, 128, 128, 64, 64, and q dimensional hidden layers respectively. The final layer contains a node for each output dimension. We use leaky ReLU nonlinearities in between each layer. Each surrogate remains a very shallow MLP. In each of the following dataset, the target neural model is trained for 500 epochs with 1e-4 learning rate using Adam and a minibatch size of 128. We train under 20 different λ between 0.0001 and 10.0. We do not do early stopping to preserve overfitting effects. We use 250 samples from the convex hull and retrain every 50 gradient steps. We set $h = 25$ for Wine and $h = 100$ otherwise. As baselines, we compare L_0 regional tree regularization to L_1 regional tree regularization, "global" tree regularization, a global decision tree classifier, an ensemble of decision tree classifiers with one per region, L_2 regularization,

and no regularization at all. Many of these can be considered ablation experiments to emphasize the importance of sparsity or regions for simulability and performance.

Evaluation We wish to compare models with global and regional explanations. However, given $\theta \in \Theta$, $\Omega^{\texttt{regional}}(\theta)$ and $\Omega^{\texttt{global}}(\theta)$ are not directly comparable: subtly, the APL of a global tree is often an overestimate for data points in a single region. To reconcile this, for any globally regularized model, we separately compute $\Omega^{\texttt{regional}}(\theta)$ as an evaluation criterion. In this context, $\Omega^{\texttt{regional}}$ is used only for evaluation; it does not appear in the objective nor training. We do the same for baselines, L_2 regularized models, and unregularized models.

Datasets We provide context for each UCI dataset. In each, we choose a generic method for defining regions to showcase the wide applicability of regional regularization: we fit a k-means clustering model with $k = 5$. Each example $\mathbf{x}_n \in \mathcal{D}$ is then assigned a number, $s_n \in \{1, 2, 3, 4, 5\}$. We define $X_r = \{\mathbf{x}_n | s_n = r\} \subseteq \mathbb{R}^p$.

- *Bank Marketing* (Bank) contains 45,211 rows collected from marketing campaigns for a bank (Moro, Cortez, & Rita, 2014). An example $\mathbf{x}_n$ has 17 features describing a recipient of the campaign (age, education, etc). There is one binary ouput indicating whether the recipient subscribed to the campaign.
- *MAGIC Gamma Telescope* (Gamma) contains 19,020 samples from a simulator of high energy Gamma particles in an Cherenkov telescope. There are 11 input features for afterimages of photon pulses, and one binary output discriminating signal and noise.
- *Adult Income* (Adult) contains 48,842 data points with 14 input features describing personal information (age, sex, etc.) and a binary output indicating if an individual's income exceeds \$50,000 per year (Kohavi, 1996).
- *Wine Quality* (Wine) contains 4,898 examples describing Portuguese wine. Each wine has a quality score from 0-10 and 11 features describing acidity, sugar, pH level, etc. We binarize the target where a positive label indicates a score of at least 5.

Node Count Baseline We also compare models to a new baseline called "global node regularization". Similar to the global tree regularization, we optimize a neural network to be faithfully approximated by a small decision tree. However, instead of regularizing APL, we regularize the total number of nodes in the distilled decision tree. A surrogate network is used to predict node count. Intuitively, simpler trees have fewer nodes: a smaller node count should indicate a more simpler model. In practice, we find sub-par performance from this baseline: in Figure 17, we find global nodes to underperform global and regional tree regularization. Moreover, this baseline finds few optima in the medium APL region: increased regularization strength collapses from a high-APL, high-F1 solution to a low-APL, low-F1 solution. We discuss a few takeaways. First, minimizing the total nodes instead of APL makes it difficult to find expressive, simulable trees. Whereas trees with low APL may still have a high branching factor, trees with low node count are more constrained. Thus, forcing node count to be small risks producing trivial trees with little to gain: branching does not change the simulability of a decision tree since for any example, the number of decisions to be made remains constant. Second, total node count has higher variance than APL meaning it is a more difficult function to learn using a surrogate.

Results Figure 17 (a-d) compare regional tree regularization the the baselines. The points plotted show minima from 3 independent runs (different random seeds). We make a few remarks. First, an unregularized model (black) does poorly due to overfitting to a complex decision boundary, as the neural network is over-parameterized. Second, we find that L_2 is *not* a desirable regularizer for simulability as it is unable to find many minima in the low APL region (see Gamma, Adult, and Wine under roughly 5 APL). Any increase in regularization strength quickly causes the target neural model to decay to an F1 score of 0, in other words, one that predict a single label. We see similar behavior with global tree regularization, suggesting that finding low complexity minima is challenging under global constraints. Third, regional tree regularization achieves the highest test accuracy in all datasets. We find that in the lower APL area, regional explanations surpasses global explanations in performance. For example, in Bank, Gamma, Adult, and Wine, we can see this at 3-6, 4-7, 5-8, 3-4 APL respectively. This suggests that it is easier to regularize groups rather than the entire input space as a whole. In fact, unlike global regularization, models constrained regionally are able to reach a wealth of minima in the low APL area. Lastly, we note that with high strengths, regional tree regularization converges in performance with regional decision trees, which is sensible as the target network prioritizes distillation over performance.

10. Applications: Healthcare

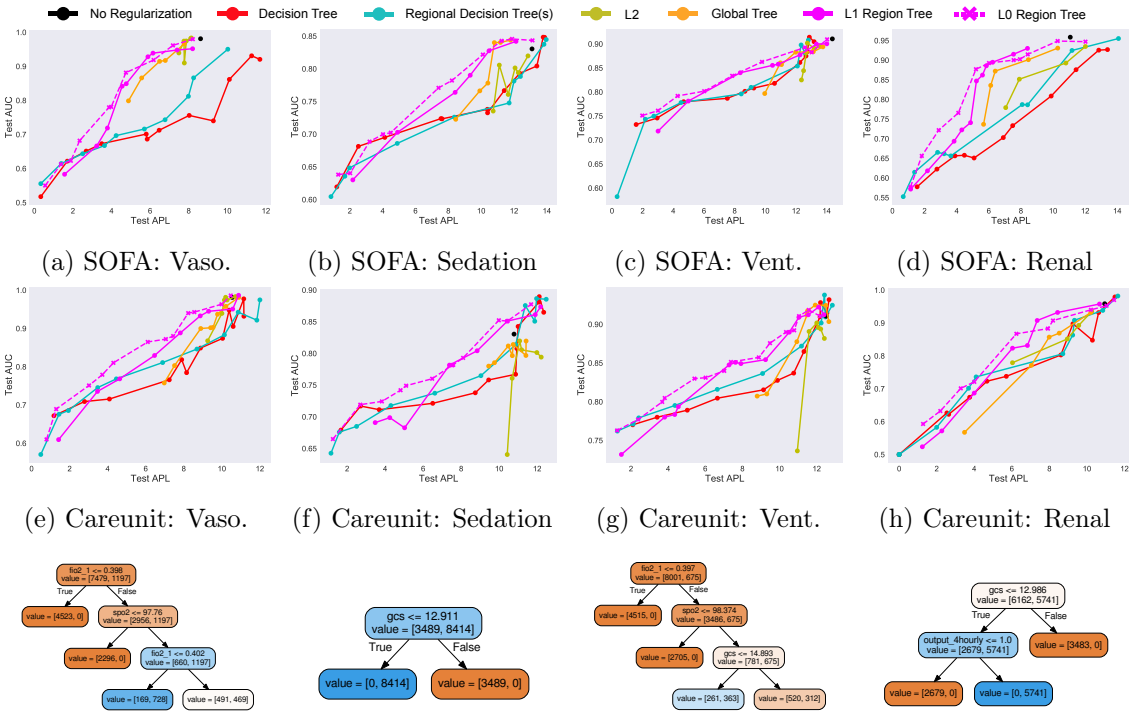
Next, we return to the critical care and HIV datasets explred in Section 6 to study the impact of regional tree regularization.

10.1 Sepsis Critical Care

We revisit the Sepsis Critical Care data set, only now we apply regional tree regularization and compare to other regularizers, including global tree regularization. The UCI data sets had only 1 binary output while Critical Care has 5. So, we take the sum of APLs for each output dimension as the measure of complexity. This requires fitting $q \times R$ trees.

We explore two methods of defining regions, both suggested by ICU physicians. The first defines three regions by sequential organ failure assessment (SOFA), a summary statistic that has historically been used for predicting ICU mortality. The groups are defined by more than one standard deviation below the mean, one standard deviation from the mean, and more than one standard deviation above the mean. Intuitively, each group should encapsulate a very different type of patient. The second method clusters patients by the his/her careunit into five groups: MICU (medical), SICU (surgical), TSICU (trauma surgical), CCU (cardiac non-surgical), and CSRU (cardiac surgical). Again, patients who undergo surgery should behave differently than those with less-invasive operations.

Results Figure 18 compares regional regularizers using SOFA regions (a-d) and careunit regions (e-h). Overall, the patterns we discussed in the UCI datasets are consistent in this application. We especially highlight the inability of global explanation to find as many low complexity solutions. For example, in Figure 18 (a,c,e), the minima from global constraints stay very close to the unregularized minima. In other cases (f, g), global regularization finds very poor optima: reaching low accuracy with high APL. In contrast, region regularization consistently finds a good compromise between complexity and performance. In each subfigure,



(a) SOFA: Vaso. (b) SOFA: Sedation (c) SOFA: Vent. (d) SOFA: Renal

(e) Careunit: Vaso. (f) Careunit: Sedation (g) Careunit: Vent. (h) Careunit: Renal

(i) Low SOFA: Vent. (j) High SOFA: Vent. (k) Low SOFA: Sedation (l) High SOFA: Sedation

Figure 18: Comparison of regularization methods on the Critical Care dataset. Each output represents a form of medication given in the ICU (e.g. vasopressor, sedation, mechanical ventilation, and renal replacement therapy). Each subfigure compares APL and test accuracy. *(a-d)* compute APL based on three regions defined using SOFA scores; *(e-h)* instead, compute APL on five regions, one for each careunit (e.g. medical vs. surgical ICU). In each experiment, regional tree regularized finds the best performing models at low complexity. Finally, *(i-l)* show distilled decision trees (split by SOFA) that best approximate a regionally regularized target neural model with a low APL and good test accuracy. As confirmed by a physician in the ICU, distilled trees are simulable and capture statistical nuances specific to a region.

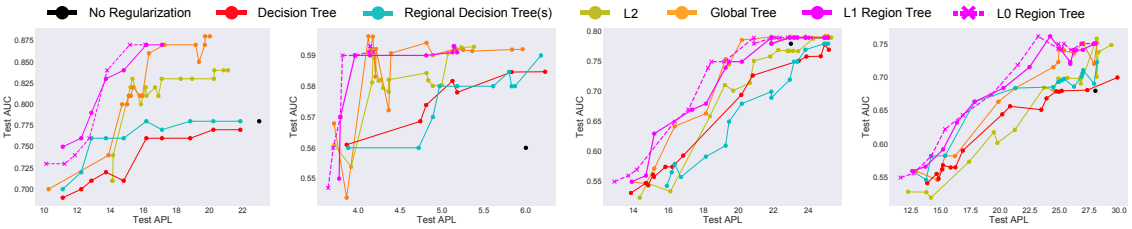
we can point to a span of APL at which the pink curve is much higher than all others. These results are from 3 runs, each with 20 different strengths.

A consequence of tree regularization is that every minima is associated with a set of distilled trees. We can extract the trees that best approximate the target neural model, and rely on it for explanation. Figure 18 (i,j) show an example of two trees predicting ventilation plucked from a low APL - high AUC minima of a regional tree regularized model. We note that the composition of the trees are different, suggesting that they each capture a decision function biased to a region. Moreover, we can see that while Figure 18i mostly predicts 0, Figure 18k mostly predicts 1; this agrees with our intuition that SOFA scores are correlated with risk of mortality. Figure 18(k, l) show similar findings for sedation. If we were to capture this behavior with a single decision tree, we risk losing granularity or simplicity.

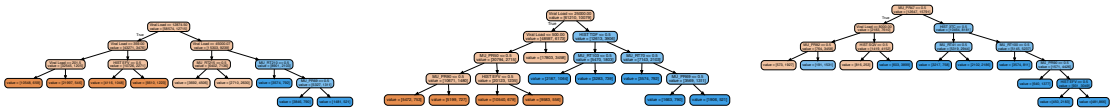
We presented a set of 9 distilled trees (1 for each output and SOFA region) to an expert intensivist for interpretation. Broadly, he found the regions beneficial as it allowed him to connect the model to his cognitive categories of patients—including those unlikely to need interventions. He verified that for predicting ventilation, GCS (mental status) should have been a key factor, and for predicting vasopressor use, the logic supported cases when vasopressors would likely be used versus other interventions (e.g. fluids if urine output is low). He was also able to make requests: for example, he asked if the effect of oxygen could have been a higher branch in the tree to better understand its effects on ventilation choices, and, noticing the similarities between the sedation and ventilation trees, pointed out that they were correlated and suggested defining new regions by both SOFA and ventilation status. We highlight that this kind of reasoning about what the model is learning and how it can be improved is very valuable. Very few notions of interpretability in deep models offer the level of granularity and simulatability as regional tree explanations do.

10.2 Human Immunodeficiency Virus

Finally, we revisit the EuResist data set to compare global and regional explanations. We define regions based on the advice of medical experts. This is performed using a patient's degree of immunosuppression at baseline (known as CDC staging). These groups are defined as: <200 cells/mm^3, 200 - 300 cells/mm^3, 300 - 500 cells/mm^3 and >500 cells/mm^3 (Organization et al., 2005). This choice of regions should characterize patients based on the severity of the infection; the lower the initial cell count, the more severe the infection.



(a) Mortality (b) AIDS Onset (c) Adherence (d) Viral Suppression



(e) High Mortality (f) Mid Mortality (g) Low Mortality

Figure 19: Comparison of regularizers on 4 output dimensions of EuResist. We apriori define four regions corresponding to the level of immunosuppression at baseline (e.g. <200 cells/mm^3). Subfigure (a-d) compares APL and test accuracy in a region whereas (e-g) show distilled decision trees with a low APL that approximate the target network per region.

Results Figure 19 compares regularizers against baseline models across levels of immunosuppression. Overall, regional tree regularization produces more accurate predictions and provides simpler explanations across all outputs. For the case of predicting patient mortality in Figure 19a, we tend to find more suitable optima across different patient groupings and

can provide better regional explanations for these patients as a result. Here, we observe that patients with lower levels of immunosuppression tend to have lower risk of mortality. We also observe that patients with lower immunity at baseline are more likely to progress to AIDS. Similar inferences can be made for the other outputs. In each subfigure, we reiterate that there is a span of APL at which the pink curve is much higher than all others.

We extract decision trees that approximate the target model for multiple minima and use these as explanations. Fig 19 (e-g) show three trees where we have low APL and high AUC minima from a regional tree regularized model. Again, the trees look significantly different based on the decision function in a particular region. In particular, we observe that lower levels of immunity at baseline are associated with higher viral loads (lower viral suppression) and higher risk of mortality. The trees were shown to a physician specializing in HIV treatment. He was able to simulate the model's logic, and confirmed our observations about relationships between viral loads and mortality. In addition, he noted that when patients have lower baseline immunity, the trees for mortality contain several more drugs. This is consistent with medical knowledge, since patients with lower immunity tend to have more severe infections, and require more aggressive therapies to combat drug resistance.

11. Discussion of Regional Tree Regularization

We summarize several of our experimental takeaways below.

We seek the low APL, high AUC regime. The ideal model is one that is highly performant and simulable. This translates to high F1/AUC scores near medium APL. Too large of an APL would be hard for an expert to understand. Too small of an APL would be too restrictive, resulting in no benefit from using a deep model. Across all experiments, we see that L_0 region regularization is most adept at finding low APL and high AUC minima.

	Bank	Gamma	Adult	Wine	Crit. Care	HIV
Fidelity	0.892	0.881	0.910	0.876	0.900	0.897

Table 4: Fidelity is the percentage of examples on which the prediction made by a tree agrees with the deep model (Craven & Shavlik, 1996).

Regional tree regularization are regionally faithful. Table 4 shows the fidelity of a deep model to its distilled tree (using a minima with low APL and high AUC). A score of 1.0 indicates that both models learned the same decision function. With an average fidelity of 89% over six datasets, the regularized model is "simple" in most cases, but can take advantage of deep nonlinearity with difficult examples. We also emphasize that we have full knowledge of when a tree-regularized neural network is making a "simulable" prediction by checking when it agrees with its distilled decision tree. This enables us to take more precaution in real-world contexts when the predictions between the network and its tree differ.

To investigate this further, we compare the performance of this distilled tree (1) to the deep neural network and (2) to a decision tree trained with CART on the raw data. Figure 20 plots the performance and complexity of several minima under global- and regional-tree regularized models on the Sepsis vasopressor prediction task. The blue and red lines

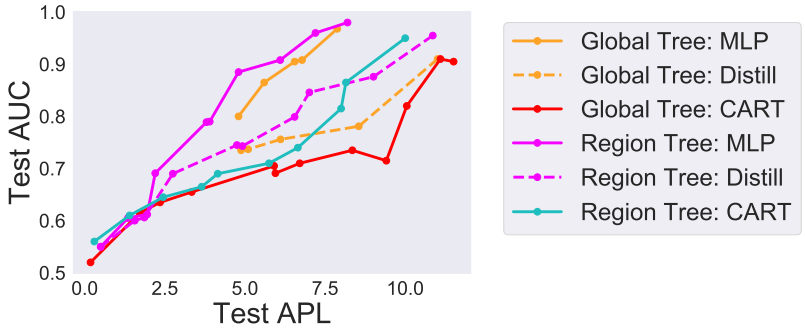


Figure 20: Comparing the performance on the Critical Care dataset of distilled decision trees from regional- or global- tree regularized models to decision trees trained by CART.

represent decision trees trained with CART whereas the dotted lines represent distilled trees. We observe that at low regularization strengths (or high APL), the distilled trees are less faithful to the neural network whereas at higher strengths (low APL), distilled trees find more similar minima to the neural network. Moreover, we observe that distilled trees reach higher performance at lower APL than CART, especially in the mid-APL region.

Regional tree regularization is not computationally expensive. Over 100 trials on Sepsis, an L_2 model takes 2.393 ± 0.258 sec. per epoch; a global tree model takes 5.903 ± 0.452 sec. and 21.422 ± 0.619 sec. to (1) sample 1000 convex samples, (2) compute APL for $\mathcal{D}^\theta$, (3) train a surrogate model for 100 epochs; a regional tree model takes 6.603 ± 0.271 sec. and 39.878 ± 0.512 sec. for (1), (2), and training 5 surrogates. The increase in base cost is due to the extra forward pass through R surrogate models to predict APL. The surrogate cost(s) are customizable depending on the size of $\mathcal{D}^\theta$, the number of training epochs, and the frequency of re-training. If R is large, we need not re-train each surrogate. The choice of which regions to prioritize can be treated as a bandit problem.

Distilled decision trees are interpretable by domain experts. Much like in the global setting, we again asked physicians to analyze the distilled decision trees from regional regularization for both the ICU and HIV applications. They were able to quickly understand the learned decision function per region, suggest improvements, and verify the logic.

Gradient-free methods are sub-optimal. For comparison, we tried alternative optimization methods that do not require differentiating through training a decision tree: (1) estimate gradients by perturbing inputs, (2) search algorithms like Nelder-Mead. However, we found these methods to either be unreasonably expensive, or easily stuck in local minima based on initialization such that results were difficult to reproduce.

Sparsity over regions is important. We experimented with different "dense" norms: L_1, L_2, and a softmax approximation to L_0, all of which faced issues where regions with simpler decision boundaries a priori were over-regularized to trivial decision functions. Only with L_0 (i.e. `sparsemax`) did we avoid this problem. As a consequence, in toy examples, we observe that `sparsemax` finds minima with more axis-aligned boundaries. In real world studies, we find `sparsemax` to lead to better performance in low/mid APL regimes.

12. Limitations of Tree Regularization

In the previous sections, we have demonstrated how we can effectively and efficiently regularize deep networks to prefer simulable forms—decision trees with short APLs—as much as possible. That said, every method has its limitations.

First, simulability from tree regularization relies on the input features being intepretable. Indeed, decision trees are generally applied to data sets with interpretable (but perhaps high-dimensional) inputs. When this is not the case—such as with our experiments on images in Section 7—interpretable higher-order features must first be extracted from the raw input; creating such higher-order features is a domain and data-specific process. For example, naively dividing the image into disjoint boxes will likely not suffice for richer image datasets like ImageNet, and completely different kinds of features will be needed for frequency data such as speech or tactile measurements. However, we emphasize both that such featurization is possible when needed, and also there exist many compelling domains like healthcare where data come in high-dimesional vectors of interpretable features.

Second, large decision trees remain difficult to interpret. While a decision tree of an APL up to 10 may be simulable, decision trees with APL beyond 20 would be extremely costly, if not impossible, for a human to understand. In our sepsis and critical care applications, we are able to find minima that achieves higher accuracy at low APLs (≤ 10). However, in other settings such as the TIMIT dataset, despite using tree regularization, we only reduce the APL from 60 to 30 before sacrificing too much performance. While this is not a negligible difference, we acknowledge that an APL of 30 is still not interpretable.

Third, we have strong empirical demonstrations on the robustness of our approach, but tree-regularization does not have the same kind of theoretical guarantees as more traditional regularizers like the L_1 and L_2 norm on model parameters. This is because efficient tree regularization relies the predictions of the surrogate neural network, makes it difficult to preserve any meaningful theoretical statements. However, we emphasize that this surrogate network is repeatedly retrained at every epoch. Thus, suppose, either due to a poor initialization or stochasticity in optimization, the target network weights suddenly change significantly and the surrogate, which maps weights to APL, cannot generalize. While we may make poor steps at that point, the surrogate will adapt and correct itself following retraining. Also recall that we take convex combinations of the weights collected in the data set $\mathcal{D}^\theta$. Thus, even if we find an outlier weight, upon retraining, the surrogate will observe many "augmentations" of this outlier. Thus, while any single surrogate may not be robust, frequently retraining the surrogate practically ameliorates problems.

We see this empirically: over several runs of different models and datasets, we do not see high variance in the optima that tree regularized models converge to. Recall that in Section 6.2, we consistently find the same distilled trees over repeated runs.

13. Conclusion

Interpretability is a bottleneck preventing widespread acceptance of deep learning. We have introduced a family of novel tree-regularization techniques that encourages the complex decision boundaries of any differentiable model to be well-approximated by human-simulable functions, allowing domain experts to quickly understand and approximately compute what

the model is doing. Overall, our training procedure is robust and efficient. Across three complex, real-world domains (HIV treatment, sepsis treatment, and human speech processing) our tree-regularized models provide gains in prediction accuracy in the regime of simpler, human-simulable models. Finally, we then showed how to extend tree regularization to more regional-specific approximations of a loss, where experts can add prior knowledge about the structure of their domain. More broadly, our general training procedure could apply tree regularization or other procedure-regularization to a wide class of popular models, helping us move beyond sparsity toward models humans can easily simulate and thus trust.

14. Acknowledgments

An early version of this work has appeared as a conference paper in the proceedings of the 32nd AAAI Conference on Artificial Intelligence. We thank the associate editor of JAIR and anonymous reviewers for their helpful comments. A substantial portion of this work was done while Sonali Parbhoo was a graduate student at the University of Basel. This work was further supported by the Stanford Interdisciplinary Graduate Fellowship.

References

Adler, P., Falk, C., Friedler, S. A., Rybeck, G., Scheidegger, C., Smith, B., & Venkatasubramanian, S. (2016). Auditing black-box models for indirect influence. In *ICDM*.

Amir, D., & Amir, O. (2018). Highlights: Summarizing agent behavior to people. In *Proc. of the 17th International conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Audet, C., & Kokkolaras, M. (2016). *Blackbox and derivative-free optimization: theory, algorithms and applications*. Springer.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Balan, A. K., Rathod, V., Murphy, K. P., & Welling, M. (2015). Bayesian dark knowledge. In *NIPS*.

Balestriero, R. (2017). Neural decision trees. *arXiv preprint arXiv:1702.07360*.

Binder, A., Bach, S., Montavon, G., Müller, K.-R., & Samek, W. (2016). Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA) 2016*, pp. 913–922. Springer.

Bucilua, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *KDD*.

Che, Z., Kale, D., Li, W., Bahadori, M. T., & Liu, Y. (2015). Deep computational phenotyping. In *KDD*.

Chen, J. H., Asch, S. M., et al. (2017). Machine learning and prediction in medicine-beyond the peak of inflated expectations. *N Engl J Med*, *376*(26), 2507–2509.

Cho, K., Gulcehre, B. v. M. C., Bahdanau, D., Schwenk, F. B. H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMLNP*.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Doctor AI: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*.

Craven, M., & Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pp. 24–30.

Dheeru, D., & Karra Taniskidou, E. (2017). UCI machine learning repository..

Drucker, H., & Le Cun, Y. (1992). Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, *3*(6), 991–997.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279. ACM.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. Tech. rep. 1341, Department of Computer Science and Operations Research, University of Montreal.

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

Garofolo, J. S., et al. (1993). TIMIT acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, *10*(5).

Ghassemi, M., Wu, M., Hughes, M. C., Szolovits, P., & Doshi-Velez, F. (2017). Predicting intervention onset in the icu with switching state space models. *AMIA Summits on Translational Science Proceedings*, *2017*, 82.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, *316*(22), 2402–2410.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In *NIPS*.

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hu, Z., Ma, X., Liu, Z., Hovy, E., & Xing, E. (2016). Harnessing deep neural networks with logic rules. In *ACL*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data*, *3*.

Joliffe, I. T., & Morgan, B. (1992). Principal component analysis and exploratory factor analysis. *Statistical methods in medical research*, *1*(1), 69–95.

Kim, B., Rudin, C., & Shah, J. A. (2014). The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pp. 1952–1960.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid.. In *KDD*, Vol. 96, pp. 202–207. Citeseer.

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulo, S. R. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pp. 1467–1475.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1675–1684. ACM.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lei, T., Barzilay, R., & Jaakkola, T. (2016). Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Lipton, Z. C. (2016). The mythos of model interpretability. In *ICML Workshop on Human Interpretability in Machine Learning*.

Lundberg, S., & Lee, S.-I. (2016). An unexpected unity among methods for interpreting model predictions. *arXiv preprint arXiv:1611.07478*.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Martins, A., & Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pp. 1614–1623.

Miller, T. (2018). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*.

Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, *6*, 26094.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15.

Mordvintsev, A., Olah, C., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, *20*(14), 5.

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, *62*, 22–31.

Nguyen, A., Yosinski, J., & Clune, J. (2016). Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*.

Nguyen, T. D., Kasmarik, K. E., & Abbass, H. A. (2020). Towards interpretable deep neural networks: An exact transformation to multi-class multivariate decision trees. *arXiv e-prints*, arXiv–2003.

Ochiai, T., Matsuda, S., Watanabe, H., & Katagiri, S. (2017). Automatic node selection for deep neural networks using group lasso regularization. In *ICASSP*.

Organization, W. H., et al. (2005). Interim who clinical staging of hvi/aids and hiv/aids case definitions for surveillance: African region. Tech. rep., Geneva: World Health Organization.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011a). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011b). Scikit-learn: Machine learning in python. *Journal of machine learning research*, *12*(Oct), 2825–2830.

Rabiner, L., & Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, *3*(1), 4–16.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-Net: ImageNet classification using binary convolutional neural networks. In *ECCV*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM.

Ross, A. S., Hughes, M. C., & Doshi-Velez, F. (2017). Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391v3*.

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

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

Singh, S., Ribeiro, M. T., & Guestrin, C. (2016). Programs as black-box explanations. *arXiv preprint arXiv:1611.07579*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Tang, W., Hua, G., & Wang, L. (2017). How to train a compact binary neural network with high accuracy?. In *AAAI*.

Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Jin, H., Petryk, S., Bargal, S. A., & Gonzalez, J. E. (2020). Nbdt: neural-backed decision trees. *arXiv preprint arXiv:2004.00221*.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yang, Y., Morillo, I. G., & Hospedales, T. M. (2018). Deep neural decision trees. *arXiv preprint arXiv:1806.06988*.

Zazzi, M., Incardona, F., Rosen-Zvi, M., Prosperi, M., Lengauer, T., Altmann, A., Sonnerborg, A., Lavee, T., Schulter, E., & Kaiser, R. (2012). Predicting response to antiretroviral treatment by machine learning: the euresist project. *Intervirology*, *55*(2), 123–127.

Zhang, Y., Lee, J. D., & Jordan, M. I. (2016). l1-regularized neural networks are improperly learnable in polynomial time. In *ICML*.