CONTEXTUAL EXPLANATION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce contextual explanation networks (CENs)—a class of models that learn to predict by generating and leveraging intermediate explanations. CENs are deep networks that generate parameters for context-specific probabilistic graphical models which are further used for prediction and play the role of explanations. Contrary to the existing post-hoc model-explanation tools, CENs learn to predict and to explain jointly. Our approach offers two major advantages: (i) for each prediction, valid instance-specific explanations are generated with no computational overhead and (ii) prediction via explanation acts as a regularization and boosts performance in low-resource settings. We prove that local approximations to the decision boundary of our networks are consistent with the generated explanations. Our results on image and text classification and survival analysis tasks demonstrate that CENs are competitive with the state-of-the-art while offering additional insights behind each prediction, valuable for decision support.

1 INTRODUCTION

Model interpretability is a long-standing problem in machine learning that has become quite acute with the accelerating pace of widespread adoption of complex predictive algorithms. While high performance often supports our belief in predictive capabilities of a system, perturbation analysis reveals that black-box models can be easily broken in an unintuitive and unexpected manner (Szegedy et al., 2013; Nguyen et al., 2015). Therefore, for a machine learning system to be used in a social context (e.g., in healthcare) it is imperative to provide a sound reasoning for each decision.

Restricting the class of models to only human-intelligible (Caruana, 2015) is a potential remedy, but often is limiting in modern practical settings. Alternatively, we may fit a complex model and explain its predictions post-hoc, e.g., by searching for linear local approximations of the decision boundary (Ribeiro et al., 2016). While such approaches achieve their goal, the explanations are generated a posteriori, require additional computation per data instance, and most importantly are never the basis for the predictions made in the first place which may lead to erroneous interpretations.

Explanation is a fundamental part of the human learning and decision process (Lombrozo, 2006). Inspired by this fact, we introduce contextual explanation networks (CENs)—a class of deep neural networks that generate parameters for probabilistic graphical models. The generated models not only play the role of explanations but are used for prediction and can encode arbitrary prior knowledge. The data often consists of two representations: (1) low-level or unstructured features (e.g., text, image pixels, sensory inputs), and (2) high-level or human-interpretable features (e.g., categorical variables). To ensure interpretability, CENs use deep networks to process the low-level representation (called the context) and construct explanations as context-specific probabilistic models on the high-level features (cf. Koller & Friedman, 2009, Ch. 5.3). Importantly, the explanation mechanism is an integral part of CEN, and our models are trained to predict and to explain jointly.

A motivating example. Consider a CEN for diagnosing the risk of developing heart arrhythmia (Figure 1a). The causes of the condition are quite diverse, ranging from smoking and diabetes to an injury from previous heart attacks, and may carry different effects on the risk of arrhythmia in different contexts. Assume that the data for each patient consists of medical notes in the form of raw text (which is used as the context) and a number of specific attributes (such as high blood pressure, diabetes, smoking, etc.). Further, assume that we have access to a parametric class of expert-designed models that relate the attributes to the condition. The CEN maps the medical notes to the parameters of the model class to produce a context-specific hypothesis, which is further used to make a prediction.
In the sequel, we formalize these intuitions and refer to this toy example in our discussion to illustrate different aspects of the framework. The main contributions of the paper are as follows:

(i) We formally define CENs as a class of probabilistic models, consider special cases (e.g., Jacobs et al., 1991), and derive learning and inference algorithms for simple and structured outputs.

(ii) We prove that post-hoc approximations of CEN’s decision boundary are consistent with the generated explanations and show that, in practice, while both methods tend to produce virtually identical explanations, CENs construct them orders of magnitude faster.

(iii) It turns out that noisy features can render post-hoc methods inconsistent and misleading, and we show how CENs can help to detect and avoid such situations.

(iv) We implement CENs by extending a number of established domain-specific deep architectures for image and text data and design new architectures for survival analysis. Experimentally, we demonstrate the value of learning with explanations for prediction and model diagnostics. Moreover, we show that explanations can act as a regularizer and improve sample efficiency.

2 RELATED WORK

The idea of combining deep networks with graphical models has been explored extensively. To mention a few, recent papers include work on structured prediction (e.g. Collobert et al., 2011; Jaderberg et al., 2014; Belanger & McCallum, 2016), kernel learning (e.g., Wilson et al., 2016; Al-Shedivat et al., 2016), and state-space modeling (e.g., Krishnan et al., 2015; Johnson et al., 2016). The key difference between CENs and the previous art is that the latter proposed to directly integrate neural networks into the graphical models as components (e.g., using neural potential functions). While flexible, the resulting deep graphical models could no longer be clearly interpreted in terms of crisp relationships between specific variables.

The idea of using a network to generate parameters for another model (typically, also a neural net) has been considered most notably for zero-shot learning (Lei Ba et al., 2015) and meta-learning (Ha et al., 2016; Edwards & Storkey, 2016; Vinyals et al., 2016), but is not suitable for interpretability purposes. CENs generate parameters for models from a restricted class and further use the attention mechanism (Xu et al., 2015) to also improve interpretability. Using explanations based on domain knowledge is known to improve generalization (Mitchell et al., 1986) and could be used as scaffolding to help at solving complex downstream tasks such as program induction (Ling et al., 2017).

Our discussion focuses on explanations defined as simple models that locally approximate behavior of a complex model. A few methods that allow to construct such explanations in a post-hoc manner have been proposed recently (Ribeiro et al., 2016; Shrikumar et al., 2017; Lundberg & Lee, 2017). There are multiple other complementary approaches to interpretability ranging from a variety of visualization techniques (Simonyan & Zisserman, 2014; Yosinski et al., 2015; Mahendran & Vedaldi, 2015; Karpathy et al., 2015), to explanations by example (Caruana et al., 1999; Kim et al., 2014; Kim et al., 2016; Koh & Liang, 2017), to natural language rationales (Lei et al., 2016). Finally, we point out that our framework encompasses the class of so-called personalized or instance-specific models that learn to partition the space of inputs and fit local sub-models (e.g., Wang & Saligrama, 2012).
3 Methods

We consider the problem of learning from a collection of data where each instance is represented by three random variables: the context, $C \in \mathcal{C}$, the attributes, $X \in \mathcal{X}$, and the targets, $Y \in \mathcal{Y}$. Our goal is to learn a model, $p_w(Y \mid X, C)$, parametrized by $w$ that can predict $Y$ from $X$ and $C$. We define contextual explanation networks as models that assume the following form (Figure 1b):

$$Y \sim p(Y \mid X, \theta), \quad \theta \sim p_w(\theta \mid C), \quad p_w(Y \mid X, C) = \int p(Y \mid X, \theta)p_w(\theta \mid C)d\theta \quad (1)$$

where $p(Y \mid X, \theta)$ is a predictor parametrized by $\theta$. We call such predictors explanations, since they explicitly relate interpretable variables, $X$, to the targets, $Y$. For example, when the targets are scalar and binary, explanations may take the form of linear logistic models; when the targets are more complex, dependencies between the components of $Y$ can be represented by a graphical model, e.g., a conditional random field (Lafferty et al., 2001).

CENs assume that each explanation is context-specific: $p_w(\theta \mid C)$ defines a conditional probability of an explanation $\theta$ being valid in the context $C$. To make a prediction, we marginalize out $\theta$’s; to interpret a prediction, $Y = y$, for a given data instance, $(x, c)$, we infer the posterior, $p_w(\theta \mid Y = y, x, c)$. The main advantage of this approach is to allow modeling conditional probabilities, $p_w(\theta \mid C)$, in a black-box fashion while keeping the class of explanations, $p(Y \mid X, \theta)$, simple and interpretable. For instance, when the context is given as raw text, we may choose $p_w(\theta \mid C)$ to be represented with a recurrent neural network, while $p(Y \mid X, \theta)$ be in the class of linear models.

Implications of the assumptions made by (1) are discussed in Appendix A. Here, we move on to describing a number of practical choices for $p_w(\theta \mid C)$ and learning and inference for those.

3.1 Contextual Explanation Networks

In practice, we represent $p_w(\theta \mid C)$ with a neural network that encodes the context into the parameter space of explanations. There are multiple ways to construct an encoder, which we consider below.

**Deterministic Encoding.** Consider $p_w(\theta \mid C) := \delta(\phi_w(C), \theta)$, where $\delta(\cdot, \cdot)$ is a delta-function and $\phi_w$ is the network that maps $C$ to $\theta$. Collapsing the conditional distribution to a delta-function makes $\theta$ depend deterministically on $C$ and results into the following tractable conditional log-likelihood:

$$\log p(y_i \mid x_i, c_i; w) = \log \int p(y_i \mid x_i, \theta)\delta(\phi_w(c_i), \theta)d\theta = \log p(y_i \mid x_i, \theta_i = \phi_w(c_i)) \quad (2)$$

Since $p_w(\theta_i \mid y_i, x_i, c_i) \propto p(y_i \mid x_i, \theta_i)\delta(\phi_w(c_i), \theta_i)$, the posterior also collapses to $\theta_i^* = \phi_w(c_i)$, and hence the inference is done via a single forward pass.

**Constrained Deterministic Encoding.** The downside of deterministic encoding is the lack of constraints on the generated explanations. There are multiple reasons why this might be an issue: (i) when the context encoder is unrestricted, it might generate unstable, overfitted local models, (ii) explanations are not guaranteed to be human-interpretable per se, and often require imposing additional constraints, such as sparsity, and (iii) when we want to reason about the patterns in the data as a whole, local explanations are not enough. To address these issues, we constrain the space of
explanations by introducing a *global dictionary*, \( D := \{ \theta_k \}_{k=1}^K \), where each atom of the dictionary, \( \theta_k \), is sparse. The encoder generates context-specific explanations using *soft attention* over the dictionary, i.e., each explanation becomes a convex combination of the sparse atoms (Figure 2):

\[
\phi_{w, D}(c) = \sum_{k=1}^K p_w(k \mid c) \theta_k = \alpha_w(c)^\top D, \quad \sum_{k=1}^K \alpha_w^{(k)}(c) = 1, \quad \forall k : \alpha_w^{(k)}(c) \geq 0
\]

where \( \alpha_w(c) \) is the attention over the dictionary. As previously, the encoder is a delta-function, \( p_{w, D}(\theta \mid C) := \delta(\phi_{w, D}(C), \theta) \). The model is trained by learning the weights, \( w \) and the dictionary, \( D \). The log-likelihood is as given in (2), and learning and inference are done via a forward pass.

**Mixture of Experts.** So far, we represented \( p_w(\theta \mid C) \) by a delta-function centered around the output of the encoder. It is natural to extend \( p_w(\theta \mid C) \) to a mixture of delta-distributions, in which case CENs recover the *mixtures of experts* (MoE, Jacobs et al., 1991). In particular, let \( D := \{ \theta_k \}_{k=1}^K \) be now a dictionary of experts, and define the encoder as \( p_{w, D}(\theta \mid C) = \sum_{k=1}^K p_w(k \mid C) \delta(\theta, \theta_k) \). The log-likelihood in such case is the same as for MoE:

\[
\log p_{w, D}(y_i \mid x_i, c_i) = \log \int p(y_i \mid x_i, \theta)p_{w, D}(\theta(c_i))d\theta = \log \sum_{k=1}^K p_w(k \mid c_i)p(y_i \mid x_i, \theta_k)
\]

Note that \( p_w(k \mid C) \) is also represented as soft attention over the dictionary, \( D \), which is now used for combining predictions of each expert, \( \theta_k \), for a given context, \( C \), instead of constructing a single context-specific explanation. Learning is done by either directly optimizing the log-likelihood (4) or via EM. To infer an explanation for a given context, we compute the posterior (see Appendix C).

**Contextual Variational Autoencoders.** Modeling \( p(Y \mid X, C) \) in the form of (1) avoids representing the joint distribution, \( p(\theta, C) \), which is a good decision when the data is abundant. However, incorporating a generative model of the context provides a few benefits: (i) a better regularization in low-resource settings, and (ii) a coherent Bayesian framework that allows imposing additional priors on the parameters of explanations, \( \theta \). We accomplish this by representing \( p(\theta, C) \) with a variational autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) whose latent variables are explanation parameters (Figure 1c). The generative process and the evidence lower bound (ELBO) are as follows:

\[
\begin{align*}
\theta &\sim p_\gamma(\theta), \quad C \sim p_u(C \mid \theta), \quad Y \sim p(Y \mid X, \theta), \\
\log p(Y, C \mid X) &\geq \mathbb{E}_{q_\omega(\theta \mid C)} \left[ \log p(Y, C \mid X, \theta) \right] - \text{KL} \left( q_\omega(\theta \mid C) \parallel p(\theta) \right)
\end{align*}
\]

where \( p(Y, C \mid X, \theta) := p(Y \mid X, \theta)p_u(C \mid \theta), \) and \( q_\omega(\theta \mid C) \) and \( p_u(C \mid \theta) \) the encoder and decoder, respectively. We consider encoders that also make use of the global learnable dictionary, \( D \), and represent \( q_\omega(\theta \mid C) \) in the form of logistic normal distribution over the simplex spanned by the atoms of \( D \). For the prior, \( p_\gamma(\theta) \), we use a Dirichlet distribution with parameters \( \alpha_k < 1 \) to induce sharp attention. Derivations are deferred to Appendix D.

### 3.2 CEN-generated vs. Post-hoc Explanations

In this section, we analyze the relationship between CEN-generated and LIME-generated *post-hoc* explanations. LIME (Ribeiro et al., 2016) constructs explanations as local linear approximations of the decision boundary of a model \( f \) in the neighborhood of a given point \( (x, c) \) via optimization:

\[
\hat{\theta} = \arg\min_{\theta} \mathcal{L}(f, \theta, \pi_{x,c}) + \Omega(\theta),
\]

where \( \mathcal{L}(f, \theta, \pi_{x,c}) \) measures the quality of the linear model \( g_\theta : X \mapsto Y \) as an approximation to \( f \) in the neighborhood of \( (x, c) \), and \( \Omega(\theta) \) is a regularizer. The typical choice for \( \mathcal{L} \) and \( \Omega \) is \( L_2 \) and \( L_1 \) losses, respectively. The neighborhood of \( (x, c) \) is defined by a distribution \( \pi_{x,c} \) concentrated around the point of interest. Given a trained CEN, we can use LIME to approximate its decision boundary and compare the explanations produced by both methods. The question we ask:

*How does the local approximation, \( \hat{\theta} \), relate to the actual explanation, \( \theta^* \), generated and used by CEN to make a prediction in the first place?*

For the case of binary classification, it turns out that when the context encoder is deterministic and the space of explanations is linear, local approximations, \( \theta \), obtained by solving (6) recover the original CEN-generated explanations, \( \theta^* \). Formally, our result is stated in the following theorem.

\footnote{Analysis of the multi-class case can be reduced to the binary in the one-vs-all fashion.}
Theorem 1. Let the explanations and the local approximations be in the class of linear models, \( p(Y = 1 \mid x, \theta) \propto \exp \{ x^\top \theta \} \). Further, let the encoder be L-Lipschitz and pick a sampling distribution, \( \pi_{x,c} \), that concentrates around the point \( (x, c) \), such that \( p_{\pi_{x,c}}(\|z' - z\| > t) < \varepsilon(t) \), where \( z := (x, c) \) and \( \varepsilon(t) \to 0 \) as \( t \to \infty \). Then, if the loss function is defined as

\[
L = \frac{1}{K} \sum_{k=1}^{K} \left( \log p(Y = 1 \mid x_k, c_k) - \log p(Y = 1 \mid x_k, \theta) \right)^2, \quad (x_k, c_k) \sim \pi_{x,c},
\]

the solution of (6) concentrates around \( \theta^* \) as \( \mathbb{P}_{\pi_{x,c}}(\|\tilde{\theta} - \theta^*\| > t) \leq \delta_{K,L}(t), \) for \( \delta_{K,L} \to 0 \).

Intuitively, by sampling from a distribution sharply concentrated around \( (x, c) \), we ensure that \( \tilde{\theta} \) will recover \( \theta^* \) with high probability. The proof is given in Appendix B.

This result establishes an equivalence between the explanations generated by CEN and those produced by LIME post-hoc when approximating CEN. Note that when LIME is applied to a model other than CEN, equivalence between explanations is not guaranteed. Moreover, as we further show experimentally, certain conditions such as incomplete or noisy interpretable features may lead to LIME producing inconsistent and erroneous explanations.

3.3 Structured Explanations for Survival Time Prediction

While CEN and LIME generate similar explanations in the case of simple classification (i.e., when \( Y \) is a scalar), when \( Y \) is structured (e.g., as a sequence), constructing coherent local approximations in a post-hoc manner is non-trivial. At the same time, CENs naturally let us represent \( p(Y \mid X, \theta) \) using arbitrary graphical models. To demonstrate our approach, we consider survival time prediction task where interpretability can be uniquely valuable (e.g., in a medical setting). Survival analysis can be re-formulated as a sequential prediction problem (Chun-Nam et al., 2011). To this end, we design CENs with CRF-based explanations suitable for sequentially structured outputs.

Our setup is as follows. Again, the data instances are represented by contexts, \( C \), attributes, \( X \), and targets, \( \hat{Y} \). The difference is that now targets are sequences of \( m \) binary variables, \( \hat{Y} := (y^1, \ldots, y^m) \), that indicate occurrence of an event. If the event occurred at time \( t \in [t_i, t_{i+1}) \), then \( y^j = 0 \), \( \forall j \leq i \) and \( y^k = 1 \), \( \forall k > i \). If the event was censored (i.e., we lack information for times after \( t \in [t_i, t_{i+1}) \)), we represent targets \( (y^{i+1}, \ldots, y^m) \) with latent variables. Note that only \( m + 1 \) sequences are valid, i.e., assigned non-zero probability by the model. We define CRF-based CEN as:

\[
\theta^t \sim p_{\theta}(\theta^t \mid C), \quad t \in \{1, \ldots, m\}, \quad Y \sim p(Y \mid X, \theta^{1:m}),
\]

\[
p(Y = (y^1, y^2, \ldots, y^m) \mid X, \theta^{1:m}) \propto \exp \left\{ \sum_{t=1}^{m} y^t (x^\top \theta^t) + \omega(y^t, y^{t+1}) \right\}
\]

\[
p_{\theta}(\theta^t \mid C) := \delta(\theta^t, \phi_{w,D}^t(c)), \quad \phi_{w,D}^t(c) := \alpha(h^t) \top D, \quad h^t := \text{RNN}(h^{t-1}, c)
\]

Note that here we have explanations for each time point, \( \theta^{1:m} \), and use an RNN-based encoder \( \phi^t \). The potentials between attributes, \( x \), and targets, \( y^{1:m} \), are linear functions parameterized by \( \theta^{1:m} \); the pairwise potentials between targets, \( \omega(y_i, y_{i+1}) \), ensure that configurations \( (y_i = 1, y_{i+1} = 0) \) are improbable (i.e., \( \omega(1,0) = -\infty \) and \( \omega(0,1) = \omega(0,0) = \omega(1,1) = \alpha_{10} \) are learnable parameters). Given these constraints, the likelihood of an uncensored event at time \( t \in [t_j, t_{j+1}) \) is

\[
p(T = t \mid x, \Theta) = \exp \left\{ \sum_{i=1}^{m} x^\top \theta^i \right\} / \sum_{k=0}^{m} \exp \left\{ \sum_{i=k+1}^{m} x^\top \theta^i \right\}
\]

and the likelihood of an event censored at time \( t \in [t_j, t_{j+1}) \) is

\[
p(T \geq t \mid x, \Theta) = \sum_{k=j+1}^{m} \exp \left\{ \sum_{i=k+1}^{m} x^\top \theta^i \right\} / \sum_{k=0}^{m} \exp \left\{ \sum_{i=k+1}^{m} x^\top \theta^i \right\}
\]

The joint log-likelihood of the data consists of two parts: (a) the sum over the non-censored instances, for which we compute \( \log p(T = t \mid x, \Theta) \), and (b) sum over the censored instances, for which we use \( \log p(T \geq t \mid x, \Theta) \). We provide a much more elaborate discussion of the survival time prediction setup and our architectures in Appendix F.
Table 1: Performance of the models on classification tasks (averaged over 5 runs; the std. are on the order of the least significant digit). The subscripts denote the features on which the linear models are built: pixels (pxl), HOG (hog), bag-or-words (bow), topics (tpc), embeddings (emb), discrete attributes (att).

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST Err (%)</th>
<th>Model</th>
<th>MNIST Err (%)</th>
<th>Model</th>
<th>MNIST Err (%)</th>
<th>Model</th>
<th>IMDB Acc (%)</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR.pxl</td>
<td>8.00</td>
<td>LR.hog</td>
<td>60.1</td>
<td>LR.bow</td>
<td>13.3</td>
<td>LR.att</td>
<td>62.5</td>
<td>68.1</td>
</tr>
<tr>
<td>LR.hog</td>
<td>2.98</td>
<td>LSTM</td>
<td>9.4</td>
<td>VGG</td>
<td>13.2</td>
<td>MLP</td>
<td>77.4</td>
<td>78.7</td>
</tr>
<tr>
<td>CNN</td>
<td>0.75</td>
<td>CEN.pxl</td>
<td>1.23</td>
<td>CEN.hog</td>
<td>13.0</td>
<td>CEN.emb</td>
<td>77.9</td>
<td>85.4</td>
</tr>
<tr>
<td>MoE.pxl</td>
<td>1.23</td>
<td>MoE.hog</td>
<td>13.0</td>
<td>MoE.bow</td>
<td>13.9</td>
<td>MoE.att</td>
<td>77.9</td>
<td>85.4</td>
</tr>
<tr>
<td>MoE.hog</td>
<td>1.10</td>
<td>MoE.hog</td>
<td>11.7</td>
<td>MoE.tpc</td>
<td>12.2</td>
<td>CEN.att</td>
<td>81.5</td>
<td>84.2</td>
</tr>
<tr>
<td>CEN.pxl</td>
<td>0.76</td>
<td>CEN.pxl</td>
<td>9.6</td>
<td>CEN.bow</td>
<td>6.9</td>
<td>CEN.tpc</td>
<td>83.4</td>
<td>84.6</td>
</tr>
<tr>
<td>CEN.hog</td>
<td>0.73</td>
<td>CEN.hog</td>
<td>9.2</td>
<td>CEN.bow</td>
<td>7.8</td>
<td>VCEN.att</td>
<td>83.4</td>
<td>84.6</td>
</tr>
</tbody>
</table>

* Best previous results for supervised learning and similar LSTM architectures: 8.1% (Johnson & Zhang [2010]).

4 EXPERIMENTS

For empirical evaluation, we consider applications that involve different data modalities of the context: image, text, and time-series. In each case, CENs are based on deep architectures designed for learning from the given type of context. In the first part, we focus on classification tasks and use linear logistic models as explanations. In the second part, we apply CENs to survival analysis and use structured explanations in the form of conditional random fields (CRFs).

We design our experiments around the following questions:

(i) When explanation is a part of the learning and prediction process, how does that affect performance of the predictive model? Does the learning become more or less efficient both in terms of convergence and sample complexity? How do CENs stand against vanilla deep nets?

(ii) Explanations are as good as the features they use to explain predictions. We ask how noisy interpretable features affect explanations generated post-hoc by LIME and whether CEN can help to detect and avoid such situations.

(iii) Finally, we ask what kind of insight we can gain by visualizing and inspecting explanations?

Additional details on the setup, all hyperparameters, and training procedures are given in Appendix F.

4.1 CLASSIFICATION TASKS AND LINEAR EXPLANATIONS

Classical datasets. We consider two classical image datasets, MNIST and CIFAR10, and a text dataset for sentiment classification of IMDB reviews (Maas et al., 2011). For MNIST and CIFAR10: full images are used as the context; to imitate high-level features, we use (a) the original images cubically downscaled to $20 \times 20$ pixels, gray-scaled and normalized, and (b) HOG descriptors computed using $3 \times 3$ blocks (Dalal & Triggs, 2005). For IMDB: the context is represented by sequences of words; for high-level features we use (a) the bag-of-words (BoW) representation and (b) the 50-dimensional topic representation produced by a separately trained off-the-shelf topic model. Neither data augmentation, nor pre-training or other unsupervised techniques were used.

Remote sensing. We also consider the problem of poverty prediction for household clusters in Uganda from satellite imagery and survey data (the dataset is referred to as Satellite). Each household cluster is represented by a collection of $400 \times 400$ satellite images (used as the context) and 65 categorical variables from living standards measurement survey (used as the interpretable attributes). The task is binary classification of the households into poor and not poor. We follow the original study of Jean et al., 2016 and use a pre-trained VGG-F network to compute 4096-dimensional embeddings of the satellite images on top of which we build contextual models. Note that this datasets is fairly small (642 points), and hence we keep the VGG-F part of the model frozen to avoid overfitting.

http://yann.lecun.com/exdb/mnist/
http://www.cs.toronto.edu/~kriz/cifar.html
Models. For each task, we use linear regression and vanilla deep nets as baselines. For MNIST and CIFAR10, the networks are a simple convnet (2 convolutions followed by max pooling) and the VGG–16 architecture (Simonyan & Zisserman, 2014), respectively. For IMDB, following Johnson & Zhang (2016) and we use a bi-directional LSTM with max pooling. For Satellite, we use a fixed VGG–P followed by a multi-layer perceptron (MLP) with 1 hidden layer. Our models used the baseline deep architectures as their context encoders and were of three types: (a) CENs with constrained deterministic encoding (b) mixture of experts (MoE), (c) CENs with variational context autoencoding (VCEN). All our models use the dictionary constraint and sparsity regularization.

4.1.1 EXPLANATIONS AS A REGULARIZER

In this part, we compare CENs with the baselines in terms of performance. In each task, CENs are trained to simultaneously generate predictions and construct explanations using a global dictionary. When the dictionary size is 1, they become equivalent to linear models. For larger dictionaries, CENs become as flexible as deep nets (Figure 3a). Overall, CENs show very competitive performance and are able to approach or surpass baselines in a number of cases, especially on the IMDB data (Table 1). Thus, forcing the model to produce explanations along with predictions does not limit its capacity.

Additionally, the “explanation layer” in CENs somehow affects the geometry of the optimization problem, and we notice that it often causes faster convergence (Figure 3b). When the models are trained on a subset of data (size varied between 1% and 20% for MNIST and 2% and 40% for IMDB), explanations play the role of a regularizer which strongly improves the sample efficiency of our models (Figure 3c). This becomes even more evident from the results on the Satellite dataset that had only 500 training points: contextual explanation networks significantly improved upon the sparse linear models on the survey features (known as the gold standard in remote sensing).

Figure 3: (a) Validation error vs. the size of the dictionary. (b) Training error vs. iteration (epoch or batch) for baselines and CENs. (c) Test error for models trained on random subsets of data of different sizes.

4.1.2 CONSISTENCY OF EXPLANATIONS

While regularization is a useful aspect, the main use case for explanations is model diagnostics. Linear explanation assign weights to the interpretable features, $X$, and hence their quality depends on the way we select these features. We consider two cases where (a) the features are corrupted with noise.
additive noise, and (b) the selected features are incomplete. For analysis, we use MNIST and IMDB datasets. Our question is, Can we trust the explanations on noisy or incomplete features?

**The effect of noisy features.** In this experiment, we inject noise into the features $X$ and ask LIME and CEN to fit explanations to the corrupted features. Note that after injecting noise, each data point has a noiseless representation $C$ and noisy $X$. LIME constructs explanations by approximating the decision boundary of the baseline model trained to predict $Y$ from $C$ features only. CEN is trained to construct explanations given $C$ and then make predictions by applying explanations to $X$. The predictive performance of the produced explanations on noisy features is given on Figure 4a. Since baselines take only $C$ as inputs, their performance stays the same and, regardless of the noise level, LIME “successfully” overfits explanations—it is able to almost perfectly approximate the decision boundary of the baselines using very noisy features. On the other hand, performance of CEN gets worse with the increasing noise level indicating that model fails to learn when the selected interpretable representation is of low quality.

**The effect of feature selection.** Here, we use the same setup, but instead of injecting noise into $X$, we construct $X$ by randomly subsampling a set of dimensions. Figure 4b demonstrates the result. While performance of CENs degrades proportionally to the size of $X$, we see that, again, LIME is able to fit explanations to the decision boundary of the original models despite the loss of information.

These two experiments indicate a major drawback of explaining predictions post-hoc: when constructed on poor, noisy, or incomplete features, such explanations can overfit the decision boundary of a predictor and are likely to be misleading. For example, predictions of a perfectly valid model might end up getting absurd explanations which is unacceptable from the decision support point of view.

4.1.3 **Qualitative Analysis**

Here, we focus on the poverty prediction task to analyze CEN-generated explanations qualitatively. After training CEN with a dictionary of size 32, we discover that the encoder tends to sharply select one of the two explanations (M1 and M2) for different household clusters in Uganda (see Figure 5a, also Figure 13a in appendix). In the survey data, each household cluster is marked as either urban or rural. We notice that, conditional on a satellite image, CEN tends to pick M1 for urban areas and M2 for rural (Figure 5b). Notice that explanations weigh different categorical features, such as reliability of the water source or the proportion of houses with walls made of unburnt brick, quite differently. When visualized on the map, we see that CEN selects M1 more frequently around major city areas, which also correlates with high nightlight intensity in those areas (Figures 5c, 5d). High performance of the model makes us confident in the produced explanations (contrary to LIME as discussed in the previous section) and allows us to draw conclusions about what causes the model to classify certain households in different neighborhoods as poor.

---

1. We use Gaussian noise with zero mean and select variance for each signal-to-noise ratio level appropriately.
2. We additionally visualize learned explanations for MNIST and IMDB datasets in Appendix F.
Table 2: Performance of the classical Cox and Aalen models, CRF-based models, and CENs that use LSTM or MLP for context embedding and CRF for explanations. The numbers are averages from 5-fold cross-validation; the std. are on the order of the least significant digit. @K denotes the temporal quantile, i.e., the time point such that K% of the patients in the data have died or were censored before that point.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc@25</th>
<th>Acc@50</th>
<th>Acc@75</th>
<th>RAE</th>
<th>Model</th>
<th>Acc@25</th>
<th>Acc@50</th>
<th>Acc@75</th>
<th>RAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cox</td>
<td>84.1</td>
<td>73.7</td>
<td>47.6</td>
<td>0.90</td>
<td>Cox</td>
<td>93.0</td>
<td>69.6</td>
<td>49.1</td>
<td>0.24</td>
</tr>
<tr>
<td>Aalen</td>
<td>87.1</td>
<td>66.2</td>
<td>45.8</td>
<td>0.98</td>
<td>Aalen</td>
<td>93.3</td>
<td>78.7</td>
<td>57.1</td>
<td>0.31</td>
</tr>
<tr>
<td>CRF</td>
<td>84.4</td>
<td>89.3</td>
<td>79.2</td>
<td>0.59</td>
<td>CRF</td>
<td>93.2</td>
<td>85.1</td>
<td>65.6</td>
<td>0.14</td>
</tr>
<tr>
<td>MLP-CRF</td>
<td>87.7</td>
<td>89.6</td>
<td>80.1</td>
<td>0.62</td>
<td>LSTM-CRF</td>
<td>93.9</td>
<td>86.3</td>
<td>68.1</td>
<td>0.11</td>
</tr>
<tr>
<td>MLP-CEN</td>
<td>85.5</td>
<td>90.8</td>
<td>81.9</td>
<td>0.56</td>
<td>LSTM-CEN</td>
<td>94.8</td>
<td>87.5</td>
<td>70.1</td>
<td>0.09</td>
</tr>
</tbody>
</table>

4.2 Survival Analysis and Structured Explanations

Finally, we apply CENs to survival analysis and showcase how to use our networks with structured explanations. In survival analysis, the goal is to learn a predictor for the time of occurrence of an event (in this case, the death of a patient) as well as be able to assess the risk (or hazard) of the occurrence. The classical models for this task are the Aalen’s additive model (Aalen, 1989) and the Cox proportional hazard model (Cox, 1972), which linearly regress attributes of a particular patient, X, to the hazard function. Chun-Nam et al. (2011) have shown that survival analysis can be formulated as a structured prediction problem and solved using a CRF variant. Here, we propose to use CENs with deep nets as encoders and CRF-structured explanations (as described in Section 3.3). More details on the architectures of our models, the baselines, as well as more background on survival analysis are provided in Appendix E.

Datasets. We use two publicly available datasets for survival analysis of the intensive care unit (ICU) patients: (a) SUPPORT2, and (b) data from the PhysioNet 2012 challenge. The data was preprocessed and used as follows:

SUPPORT2: The data had 9105 patient records and 73 variables. We selected 50 variables for both C and X features (see Appendix F). Categorical features (such as race or sex) were one-hot encoded. The values of all features were non-negative, and we filled the missing values with -1. For CRF-based predictors, the survival timeline was capped at 3 years and converted into 156 discrete intervals of 7 days each. We used 7105 patient records for training, 1000 for validation, and 1000 for testing.

PhysioNet: The data had 4000 patient records, each represented by a 48-hour irregularly sampled 37-dimensional time-series of different measurements taken during the patient’s stay at the ICU. We resampled and mean-aggregated the time-series at 30 min frequency. This resulted in a large number of missing values that we filled with 0. The resampled time-series were used as the context, C, while for the attributes, X, we took the values of the last available measurement for each variable in the series. For CRF-based predictors, the survival timeline was capped at 60 days and converted into 60 discrete intervals.

Models. For baselines, we use the classical Aalen and Cox models and the CRF from (Chun-Nam et al., 2011), where all used X as inputs. Next, we combine CRFs with neural encoders in two ways:

(i) We apply CRFs to the outputs from the neural encoders (the models denoted MLP-CRF and LSTM-CRF, all trainable end-to-end). Similar models have been shown very successful in the natural language applications (Collobert et al., 2011). Note that parameters of the CRF layer assign weights to the latent features and are no longer interpretable in terms of the attributes of interest.

(ii) We use CENs with CRF-based explanations, that process the context variables, C, using the same neural networks as in (i) and output parameters for CRFs that act on the attributes, X.

Details on the architectures are given in Appendix F.
Figure 7: Weights of the CEN-generated CRF explanations for two patients from SUPPORT2 dataset for a set of the most influential features: dementia (comorbidity), avtisst (avg. TISS, days 3-25), slos (days from study entry to discharge), hday (day in hospital at study admit), ca yes (the patient had cancer), sfdm2 Coma or Intub (intubated or in coma at month 2), sfdm2 SIP (sickness impact profile score at month 2). Higher weight values correspond to higher feature contributions to the risk of death after a given time point.

Metrics. Following Chun-Nam et al. (2011), we use two metrics specific to survival analysis: (a) accuracy of correctly predicting survival of a patient at times that correspond to 25%, 50%, and 75% population-level temporal quantiles (i.e., time points such that the corresponding % of the patients in the data were discharged from the study due to censorship or death) and (b) the relative absolute error (RAE) between the predicted and actual time of death for non-censored patients.

Quantitative results. The results for all models are given in Table 2. Our implementation of the CRF baseline reproduces (and even slightly improves) the performance reported by Chun-Nam et al. (2011). MLP-CRF and LSTM-CRF improve upon plain CRFs but, as we noted, can no longer be interpreted in terms of the original variables. On the other hand, CENs outperform neural CRF models on certain metrics (and closely match on the others) while providing explanations for risk prediction for each patient at each point in time.

Qualitative results. To inspect predictions of CENs qualitatively, for any given patient, we can visualize the weights assigned by the corresponding explanation to the respective attributes. Figure 7 explanation weights for a subset of the most influential features for two patients from SUPPORT2 dataset who were predicted as survivor and non-survivor. These explanations allow us to better understand patient-specific temporal dynamics of the contributing factors to the survival rates predicted by the model (Figure 6).

5 Conclusion

In this paper, we have introduced contextual explanation networks (CENs)—models that learn to predict by generating and leveraging intermediate context-specific explanations. We have formally defined CENs as a class of probabilistic models, considered a number of special cases (e.g., the mixture of experts mode), and derived learning and inference procedures within the encoder-decoder framework for simple and sequentially-structured outputs. We have shown that, while explanations generated by CENs are provably equivalent to those generated post-hoc under certain conditions, there are cases when post-hoc explanations are misleading. Such cases are hard to detect unless explanation is a part of the prediction process itself. Learning to predict and to explain jointly turned out to have a number of benefits, including strong regularization, consistency, and ability to generate explanations with no computational overhead. We believe that the proposed class of models has a potential to be useful not only for improving prediction capabilities, but also for model diagnostics, pattern discovery, and general data analysis, especially when machine learning is used for decision support in high-stakes applications.
REFERENCES


Wang, Joseph & Venkatesh Saligrama (2012). “Local Supervised Learning through Space Partitioning”. In: NIPS.


A  ANALYSIS OF THE ASSUMPTIONS MADE BY CEN

As described in the main text, CENs represent the predictive distribution in the following form:

\[ p(Y \mid X, C) = \int p(Y \mid X, \theta) p(\theta \mid C) d\theta \]

and the assumed generative process behind the data is either:

\begin{align*}
(CEN) & \quad Y \sim p(Y \mid X, \theta), \quad \theta \sim p(\theta \mid C) \text{ for the purely discriminative setting.} \\
(VCEN) & \quad Y \sim p(Y \mid X, \theta), \quad \theta \sim p(\theta), \quad C \sim p(C \mid \theta) \text{ when we model the joint distribution of the explanations, } \theta, \text{ and contexts, } C, \text{ e.g., using encoder-decoder framework.}
\end{align*}

We would like to understand whether CEN, as defined above, can represent any conditional distribution, \( p(Y \mid X, C) \), when the class of explanations is limited (e.g., to linear models), and, if not, what are the limitations?

Generally, CEN can be seen as a mixture of predictors. Such mixture models could be quite powerful as long as the mixing distribution, \( p(\theta \mid C) \), is rich enough. In fact, even a finite mixture exponential family regression models can approximate any smooth \( d \)-dimensional density at a rate \( O(m^{-4/d}) \) in the KL-distance (Jiang & Tanner, 1999). This result suggests that representing the predictive distribution with contextual mixtures should not limit the representational power of the model. The two caveats are:

(i) In practice, \( p(\theta \mid C) \) is limited, e.g., either deterministic encoding, a finite mixture, or a simple distribution parametrized by a deep network.

(ii) The classical setting of predictive mixtures does not separate inputs into two subsets, \((C, X)\). We do this intentionally to produce hypotheses/explanations in terms of specific features that could be useful for interpretability or model diagnostics down the line. However, it could be the case that \( X \) contains only some limited information about \( Y \), which could limit the predictive power of the full model.

We leave the point (ii) to future work. To address (i), we consider \( p(\theta \mid C) \) that fully factorizes over the dimensions of \( \theta \): \( p(\theta \mid C) = \prod_j p(\theta_j \mid C) \), and assume that hypotheses, \( p(Y \mid X, \theta) \), factorize according to some underlying graph, \( G_Y = (V_Y, E_Y) \). The following proposition shows that in such case \( p(Y \mid X, C) \) inherits the factorization properties of the hypothesis class.

**Proposition 1.** Let \( p(\theta \mid C) := \prod_j p(\theta_j \mid C) \) and let \( p(Y \mid X, \theta) \) factorize according to some graph \( G_Y = (V_Y, E_Y) \). Then, \( p(Y \mid X, C) \) defined by CEN with \( p(\theta \mid C) \) encoder and \( p(Y \mid X, \theta) \) explanations also factorizes according to \( G \).

**Proof.** Assume that \( p(Y \mid X, \theta) \) factorizes as \( \prod_{\alpha \in V_Y} p(Y_{\alpha} \mid Y_{MB(\alpha)}, X, \theta_{\alpha}) \), where \( \alpha \) denotes subsets of the \( Y \) variables and \( MB(\alpha) \) stands for the corresponding Markov blankets. Using the definition of CEN, we have:

\[ p(Y \mid X, C) = \int p(Y \mid X, \theta) p(\theta \mid C) d\theta \]

\[ = \int \prod_{\alpha \in V_Y} p(Y_{\alpha} \mid Y_{MB(\alpha)}, X, \theta_{\alpha}) \prod_j p(\theta_j \mid C) d\theta \]

\[ = \prod_{\alpha \in V_Y} \left[ \int p(Y_{\alpha} \mid Y_{MB(\alpha)}, X, \theta_{\alpha}) \prod_j p(\theta_j \mid C) d\theta_{\alpha} \right] \]

\[ = \prod_{\alpha \in V_Y} p(Y_{\alpha} \mid Y_{MB(\alpha)}, X, C), \]

**Remark 1.** All the encoding distributions, \( p(\theta \mid C) \), considered in the main text of the paper, including delta functions, their mixtures, and encoders parametrized by neural nets fully factorize over the dimensions of \( \theta \).

\[ \square \]
Remark 2. The proposition has no implications for the case of scalar targets, Y. However, in case of structured prediction, regardless of how good the context encoder is, CEN will assume the same set of independencies as given by the class of hypotheses, \( p(Y \mid X, \theta) \).

### B Approximating the Decision Boundary of CEN

Ribeiro et al. (2016) proposed to construct approximations of the of the decision boundary of an arbitrary predictor, \( f \), in the locality of a specified point, \( x \), by solving the following optimization problem:

\[
\hat{g} = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g), \tag{15}
\]

where \( \mathcal{L}(f, g, \pi_x) \) measures the quality of \( g \) as an approximation to \( f \) in the neighborhood of \( x \) defined by \( \pi_x \) and \( \Omega(g) \) is a regularizer that is usually used to ensure human-interpretability of the selected local hypotheses (e.g., sparsity). Now, consider the case when \( f \) is defined by a CEN, instead of \( x \) we have \((c, x)\), and the class of approximations, \( G \), coincides with the class of explanations, and hence can be represented by \( \theta \). In this setting, we can pose the same problem as:

\[
\theta = \arg\min_{\theta} \mathcal{L}(f, \theta, \pi_{c,x}) + \Omega(\theta) \tag{16}
\]

Suppose that CEN produces \( \theta^\ast \) explanation for the context \( c \) using a deterministic encoder, \( \phi \). The question is whether and under which conditions \( \theta \) can recover \( \theta^\ast \). Theorem 1 answers the question in affirmative and provides a concentration result for the case when hypotheses are linear. Here, we prove Theorem 1 for a little more general class of log-linear explanations: logit \( p(Y = 1 \mid x, \theta) = a(x)^\top \theta \), where \( a \) is a \( C \)-Lipschitz vector-valued function whose values have a zero-mean distribution when \((x, c)\) are sampled from \( \pi_x \).\( \pi_c \). For simplicity of the analysis, we consider binary classification and omit the regularization term, \( \Omega(\theta) \). We define the loss function, \( \mathcal{L}(f, \theta, \pi_{x,c}) \), as:

\[
\mathcal{L} = \frac{1}{K} \sum_{k=1}^{K} (\logit p(Y = 1 \mid x_k - x, c_k) - \logit p(Y = 1 \mid x_k - x, \theta))^2, \tag{17}
\]

where \((x_k, c_k) \sim \pi_{x,c} \) and \( \pi_{x,c} := \pi_x \pi_c \) is a distribution concentrated around \((x, c)\). Without loss of generality, we also drop the bias terms in the linear models and assume that \( a(x_k - x) \) are centered.

**Proof of Theorem 1** The optimization problem (16) reduces to the least squares linear regression:

\[
\hat{\theta} = \arg\min_\theta \frac{1}{K} \sum_{k=1}^{K} (\logit p(Y = 1 \mid x_k - x, c_k) - a(x_k - x)^\top \theta)^2 \tag{18}
\]

We consider deterministic encoding, \( p(\theta \mid c) := \delta(\theta, \phi(c)) \), and hence \( \logit p(Y = 1 \mid x_k - x, c_k) \) takes the following form:

\[
\logit p(Y = 1 \mid x_k - x, c_k) = \logit p(Y = 1 \mid x_k - x, \theta = \phi(c_k)) = a(x_k - x)^\top \phi(c_k) \tag{19}
\]

To simplify the notation, we denote \( a_k := a(x_k - x) \), \( \phi_k := \phi(c_k) \), and \( \phi := \phi(c) \). The solution of (18) now can be written in a closed form:

\[
\hat{\theta} = \left[ \frac{1}{K} \sum_{k=1}^{K} a_k a_k^\top \right]^{+} \left[ \frac{1}{K} \sum_{k=1}^{K} a_k a_k^\top \phi_k \right] \tag{20}
\]

Note that \( \hat{\theta} \) is a random variable since \((x_k, c_k)\) are randomly generated from \( \pi_{x,c} \). To further simplify the notation, denote \( M := \frac{1}{K} \sum_{k=1}^{K} a_k a_k^\top \). To get a concentration bound on \( \mathbb{E} \| \theta - \theta^\ast \| \), we will use the continuity of \( \phi(\cdot) \) and \( a(\cdot) \), concentration properties of \( \pi_{x,c} \) around \((x, c)\), and some elementary results from random matrix theory. To be more concrete, since we assumed that \( \pi_{x,c} \) factorizes, we further let \( \pi_x \) and \( \pi_c \) concentrate such that \( p_{\pi_x}(\|x - x\| > t) < \varepsilon_x(t) \) and \( p_{\pi_c}(\|c - c\| > t) < \varepsilon_c(t) \), respectively, where \( \varepsilon_x(t) \) and \( \varepsilon_c(t) \) both go to 0 as \( t \to \infty \), potentially at different rates.

\[\text{In case of logistic regression, } a(x) = [1, x_1, \ldots, x_d]^\top.\]
First, we have the following bound from the convexity of the norm:

\[
p(\|\theta - \theta^*\| > t) = p\left(\frac{1}{K} \sum_{k=1}^{K} \left[ M^+ a_k a_k^\top (\phi_k - \phi) \right] > t \right)
\]

\[
\leq p\left(\frac{1}{K} \sum_{k=1}^{K} \left[ M^+ a_k a_k^\top \right] > t \right)
\]

By making use of the inequality \(\|Ax\| \leq \|A\|\|x\|\), where \(\|A\|\) denotes the spectral norm of the matrix \(A\), the \(L\)-Lipschitz property of \(\phi(c)\), the \(C\)-Lipschitz property of \(a(x)\), and the concentration of \(x_k\) around \(x\), we have

\[
p(\|\theta - \theta^*\| > t) \leq p(L \frac{1}{K} \sum_{k=1}^{K} \|M^+ a_k a_k^\top\| \|c_k - c\| > t)
\]

\[
\leq p(CL \|M^+\| \frac{1}{K} \sum_{k=1}^{K} \|a_k a_k^\top\| \|c_k - c\| > t)
\]

\[
\leq p\left(\frac{CL\|x_k - x\|}{\lambda_{\min}(M)} > t\right) + p(\|x_k - x\| \|c_k - c\| > \tau^2)
\]

\[
\leq p\left(\frac{CL\|x\|}{\lambda_{\min}(M)} > t\right) + \|x\| \|c\| > \tau^2)
\]

\[
\leq p\left(\frac{CL\|x\|}{\lambda_{\min}(M)} > t\right) + \|x\| \|c\| > \tau^2)
\]

Note that we used the fact that the spectral norm of a rank-1 matrix, \(a(x_k)a(x_k)^\top\), is simply the norm of \(a(x_k)\), and the spectral norm of the pseudo-inverse of a matrix is equal to the inverse of the least non-zero singular value of the original matrix: \(\|M^+\| \leq \lambda_{\max}(M^+) = \lambda_{\min}^{-1}(M)\).

Finally, we need a concentration bound on \(\lambda_{\min}(M/(C\tau)^2)\) to complete the proof. Note that

\[
\frac{M}{C\tau^2} = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{a_k}{C\tau}\right)^\top \left(\frac{a_k}{C\tau}\right),
\]

where the norm of \(\left(\frac{a_k}{C\tau}\right)\) is bounded by 1. If we denote \(\mu_{\min}(C\tau)\) the minimal eigenvalue of \(\text{Cov} \left[\frac{a_k}{C\tau}\right]\), we can write the matrix Chernoff inequality (Tropp, 2012) as follows:

\[
p(\lambda_{\min}\left(M/(C\tau)^2\right) < \alpha) \leq d \exp\{-KD(\alpha\|\mu_{\min}(C\tau)\})\}, \quad \alpha \in [0, \mu_{\min}(C\tau)],
\]

where \(d\) is the dimension of \(a_k\), \(a := \frac{a}{C\tau}\), and \(D(a\|b)\) denotes the binary information divergence:

\[
D(a\|b) = a \log \left(\frac{a}{b}\right) + (1 - a) \log \left(\frac{1 - a}{1 - b}\right).
\]

The final concentration bound has the following form:

\[
p(\|\hat{\theta} - \theta^*\| > t) \leq d \exp\{-KD\left(\frac{L}{C\tau^2} \|\mu_{\min}(C\tau)\}\right\} + \varepsilon_\mathbf{x}(\tau) + \varepsilon_\varepsilon(\tau)
\]

We see that as \(\tau \to \infty\) and \(t \to \infty\) all terms on the right hand side vanish, and hence \(\hat{\theta}\) concentrates around \(\theta^*\). Note that as long as \(\mu_{\min}(C\tau)\) is far from 0, the first term can be made negligibly small by sampling more points around \((x, c)\). Finally, we set \(\tau \equiv t\) and denote the right hand side by \(\delta_{K, L, C}(t)\) that goes to 0 as \(t \to \infty\) to recover the statement of the original theorem.

**Remark 3.** We have shown that \(\hat{\theta}\) concentrates around \(\theta^*\) under mild conditions. With more assumptions on the sampling distribution, \(\pi_{x,c}\) (e.g., sub-gaussian) one could derive precise convergence rates. Note that we are in total control of any assumptions we put on \(\pi_{x,c}\) since precisely that distribution is used for sampling. This is a major difference between the local approximation setup here and the setup of linear regression with random design; in the latter case, we have no control over the distribution of the design matrix, and any assumptions we make could potentially be unrealistic.

**Remark 4.** Note that concentration analysis of a more general case when the loss \(L\) is a general convex function and \(\Omega(g)\) is a decomposable regularizer could be done by using results from the \(M\)-estimation theory (Negahban et al., 2009), but would be much more involved and unnecessary for our purposes.
C Learning and Inference in the Contextual Mixture of Experts

As noted in the main text, to make a prediction, MoE uses each of the $K$ experts where the predictive distribution is computed as follows:

$$p_{w,D}(Y \mid X, C) = \sum_{k=1}^{K} p_{w}(k \mid C) p(Y \mid X, \theta_k). \quad (29)$$

Since each expert contributes to the predictive probability, we can explain a prediction, $\hat{y}$, for the instance $(x, c)$ in terms of the posterior weights assigned to each expert model:

$$p_{w}(k \mid \hat{y}, x, c) = \frac{p_{w}(\hat{y}, k \mid x, c)}{p_{w}(\hat{y} \mid x, c)} = \frac{p(\hat{y} \mid x, \theta_k) p_{w}(k \mid c)}{\sum_{k=1}^{K} p_{w}(k \mid c) (\hat{y} \mid x, \theta_k)} \quad (30)$$

If the $p(k \mid \hat{y}, x, c)$ assigns very high weight to a single expert, we can treat that expert model as an explanation. Note that however, in general, this may not be the case and posterior weights could be quite spread out (especially, if the number of experts is small and the class of expert models, $p(Y \mid X, \theta)$, is too simple and limited). Therefore, there may not exist an equivalent local explanation in the class of expert models that would faithfully approximate the decision boundary.

To learn contextual MoE, we can either directly optimize the conditional log-likelihood, which is non-convex yet tractable, or use expectation maximization (EM) procedure. For the latter, we write the log likelihood in the following form:

$$\log p(y \mid x, c) = \sum_{k=1}^{K} q(k) \log p(y \mid x, c) \quad (31)$$

$$= \sum_{k=1}^{K} q(k) \log \left( \frac{p(y \mid x, \theta_k) p_{w}(k \mid c) q(k)}{p_{w}(k \mid y, x, c) q(k)} \right) \quad (32)$$

$$= \sum_{k=1}^{K} q(k) \log \frac{p(y \mid x, \theta_k) p_{w}(k \mid c) q(k)}{q(k)} + KL(q(k) \mid\mid p_{w}(k \mid y, x, c)) \quad (33)$$

At each iteration, we do two steps:

**(E-step)** Compute posteriors for each data instance, $q_i(k) = p_{w}(k \mid y_i, x_i, c_i)$. 

**(M-step)** Optimize $Q(w) = \sum_{k=1}^{K} q(k) \log p(y \mid x, \theta_k) p_{w}(k \mid c)$.

It is well known that this iterative procedure is guaranteed to converge to a local optimum.

D Contextual Variational Autoencoders

We can express the evidence for contextual variational autoencoders as follows:

$$\log p(Y, C \mid X)$$

$$= \int q_w(\theta \mid C) \log p(Y, C \mid X) d\theta$$

$$= \int q_w(\theta \mid C) \log \left( \frac{p(Y, X, \theta)p_u(C \mid \theta)p(\theta)}{p_u(\theta \mid Y, X, C)} \right) d\theta$$

$$= \int q_w(\theta \mid C) \log \left( \frac{p(Y, X, \theta)p_u(C \mid \theta)p(\theta)}{q_w(\theta \mid C)} \right) d\theta + KL(q_w(\theta \mid C) \mid\mid p_u(\theta \mid Y, X, C))$$

$$\geq \mathcal{L}(w, u; Y, X, C), \quad (34)$$

where $\mathcal{L}(w, u; Y, X, C)$ is the evidence lower bound (ELBO):

$$\mathcal{L}(w, u; Y, X, C)$$

$$= \mathbb{E}_{q_w} \left[ \log \frac{p(Y, X, \theta)p_u(C \mid \theta)p(\theta)}{q_w(\theta \mid C)} \right] \quad (35)$$

$$= \mathbb{E}_{q_w} [\log p(Y \mid X, \theta)] + \mathbb{E}_{q_w} [\log p_u(C \mid \theta)] - KL(q(\theta \mid C) \parallel p(\theta))$$
We notice that the ELBO consists of three terms:

1. the expected conditional likelihood of the explanation, $\mathbb{E}_{q_{w}} [\log p(Y \mid X, \theta)]$,
2. the expected context reconstruction error, $\mathbb{E}_{q_{w}} [\log p_{u}(C \mid \theta)]$, and
3. the KL-based regularization term, $-\text{KL}(q(\theta \mid C) \| p(\theta))$.

We can optimize the ELBO using first-order methods by estimating the gradients via Monte Carlo sampling with reparametrization. When the encoder has a classical form of a Gaussian distribution (or any other location-scale type of distribution), $q_{w}(\theta \mid C) = \mathcal{N}(\theta; \mu_{w}(C), \text{diag}(\sigma_{w}(C)))$, reparametrization of the samples is straightforward (Kingma & Welling, 2013).

In our experiments, we mainly consider encoders that output probability distributions over a simplex spanned by a dictionary, $D$, which turned out to have better performance and faster convergence. In particular, sampling from the encoder is as follows:

$$z \sim \mathcal{N}(\theta; \mu_{w}(C), \text{diag}(\sigma_{w}(C)))$$

$$\gamma_{0} = \frac{1}{1 + \sum_{j=1}^{K} e^{z_{j}}}, \gamma_{i} = \frac{e^{z_{i}}}{1 + \sum_{j=1}^{K} e^{z_{j}}}, i = 1, \ldots, K, \quad (36)$$

$$\theta = D \cdot \gamma$$

The samples, $\theta$, will be logistic normal distributed and are easy to be re-parametrized. For prior, we use the Dirichlet distribution over $\gamma$ with the parameter vector $\alpha$. In that case, the stochastic estimate of the KL-based regularization term has the following form:

$$-\text{KL}(q(\gamma \mid C) \| p(\gamma)) \simeq \frac{1}{L} \sum_{l=1}^{L} \log \mathcal{N} \left( \log \left( \frac{\gamma_{0}^{(l)}}{\gamma_{0}} \right) ; \mu_{w}(C), \text{diag}(\sigma_{w}(C)) \right) - \alpha^T \log(\gamma^{(l)}), \quad (37)$$

where $\gamma^{(l)}_{0}$ is a parameter vector without the first element, and $l$ indexes samples taken from the encoder, $p(\gamma \mid C)$. In practice, we use $L = 1$.

### E Survival Analysis and Contextual Conditional Random Fields

We provide some general background on survival analysis, the classical Aalen additive hazard (Aalen, 1989) and Cox proportional hazard (Cox, 1972) models, derive the structured prediction approach (Chun-Nam et al., 2011), and describe CENs with CRF-based explanations used in our experiments in detail.

In survival time prediction, our goal is to estimate the occurrence time of an event in the future (e.g., death of a patient, earthquake, hard drive failure, customer turnover, etc.). The unique aspect of the survival data is that there is always a fraction of points for which the event time has not been observed (such data instances are called censored). The common approach is to model the survival time, $T$, either for a population (i.e., average survival time) or for each instance. In particular, we can introduce the survival function, $S(t) := p(T \geq t)$, which gives the probability of the event not happening at least up to time $t$ (e.g., patient survived up to time $t$). The derivative of the survival function is called the hazard function, $\lambda(t)$, which is the instantaneous rate of failure:

$$S(t) := -\int_{0}^{t} \lambda(\tau)d\tau \quad (38)$$

This allows us to model survival on a population level. Now, proportional hazard models assume that $\lambda$ is also a function of the available features of a given instance, i.e., $\lambda(t; x)$. Cox’s proportional hazard model assumes $\lambda_{0}(t; x) = \lambda_{0}(t) \exp(x^T \theta)$. Aalen’s model is a time-varying extension and assumes that $\lambda_{0}(t; x) := x^T \theta(t)$, where $\theta(t)$ is a function of time.

Survival analysis is a regression problem as it originally works with continuous time. The time can be discretized (e.g., into days, months, etc.), and hence we can approach survival time prediction as a multi-task classification problem (Efron, 1988). Chun-Nam et al. (2011) went one step further, noted that the output space is structure in a particular way, and proposed a model called sequence of dependent regressors, which is in essence a conditional random field with a particular structure of the
pairwise potentials between the labels. In particular, as we described in Section 3.3, the targets are sequences of binary random variables, \( Y := (y^1, \ldots, y^m) \), that encode occurrence of an event as follows: for an event that occurred at time \( t \in [t_i, t_{i+1}) \), then \( y^j = 0 \), \( \forall j \leq i \) and \( y^k = 1 \), \( \forall k > i \). Note that only \( m + 1 \) sequences are valid, i.e., assigned non-zero probability by the model, which allows us to write the following linear model:

\[
p(Y = (y^1, \ldots, y^m) | x, \Theta) = \frac{\exp \left( \sum_{t=1}^{m} y^t x^T \theta^t \right)}{\sum_{k=0}^{m} \exp \left( \sum_{t=k+1}^{m} x^T \theta^t \right)}
\]  

To train the model, Chun-Nam et al. [2011] optimize the following objective:

\[
\min_{\Theta} C_1 \sum_{t=1}^{m} ||\theta^t||^2 + C_2 \sum_{t=1}^{m-1} ||\theta^{t+1} - \theta^t||^2 - \log \mathcal{L}(Y, X; \Theta)
\]  

where the first two terms are regularization and the last term is the log-likelihood which as:

\[
\mathcal{L}(Y, X; \Theta) = \sum_{i \in \text{NC}} p(T = t_i | x_i, \Theta) + \sum_{j \in \text{C}} p(T > t_j | x_j, \Theta)
\]

where NC is the set of non-censored instances (for which we know the outcome times, \( t_i \)) and C is the set of censored inputs (for which only know the censorship times, \( t_j \)). Expressions for the likelihoods of censored and non-censored inputs are the same as given in Section 3.3.

Finally, CENs additionally take the context variables, \( C \), as inputs and generate \( \theta^t \) for each time step using a recurrent encoder. In our experiments, we considered datasets where the context was represented by a vector or regularly sampled time series. Architectures for CENs used in our experiments are given in Figure 8. We used encoders suitable for the data type of the context variables available for each dataset. Each \( \theta^t \) was generated using a constrained deterministic encoder with a global dictionary, \( D \) of size 16. For details on parametrization of our architectures see tables in Appendix F.

Importantly, CEN-CRF architectures are trainable end-to-end (as all other CEN architectures considered in this paper), and we optimized the objective using stochastic gradient method. For each mini-batch, depending on which instances were censored and which were non-censored, we constructed the objective function accordingly (to implement this in TensorFlow we used masking and the standard control flow primitives for selecting between parts of the objective for censored and non-censored inputs).

### F Experimental Details

This section provides details on the experimental setups including architectures, training protocols and procedures, etc. Additionally, we include complete dictionary visualizations learned by CENs.
Figure 9: Visualization of the top 3 prediction rationalizations provided by the CEN_pxl model for correctly classified (left), misclassified (middle), and adversarial (right) digits. Adversarial examples were generated using the fast gradient sign method (FGSM) (Papernot et al., 2016).

F.1 ADDITIONAL DETAILS ON THE DATASETS AND EXPERIMENT SETUPS

MNIST. We used the classical split of the dataset into 50k training, 10k validation, and 10k testing points. All models were trained for 100 epochs using the Adam optimizer with the learning rate of $10^{-3}$. No data augmentation was used in any of our experiments. HOG representations were computed using $3 \times 3$ blocks.

CIFAR10. For this set of experiments, we followed the setup given Zagoruyko (2015), reimplemented in Keras with TensorFlow backend. The input images were global contrast normalized (a.k.a. GCN whitened) while the rescaled image representations were simply standardized. Again, HOG representations were computed using $3 \times 3$ blocks. No data augmentation was used in our experiments.

IMDB. We considered the labeled part of the data only (50,000 reviews total). The data were split into 20,000 train, 5,000 validation, and 25,000 test points. The vocabulary was limited to All models were trained with the Adam optimizers with $10^{-2}$ learning rate. The models were initialized randomly; no pre-training or any other unsupervised/semi-supervised technique was used.

Satellite. As described in the main text, we used a pre-trained VGG-16 network to extract features from the satellite imagery. Further, we added one fully connected layer network with 128 hidden units used as the context encoder. For the VCEN model, we used dictionary-based encoding with Dirichlet prior and logistic normal distribution as the output of the inference network. For the decoder, we used an MLP of the same architecture as the encoder network. All models were trained with Adam optimizer with 0.05 learning rate. The results were obtained by 5-fold cross-validation.

Medical data. We have used minimal pre-processing of both SUPPORT2 and PhysioNet datasets limited to standardization and missing-value filling. We found that denoting missing values with negative entries (−1) often led a slightly improved performance compared to any other NA-filling techniques. PhysioNet time series data was irregularly sampled across the time, so we had to resample temporal sequences at regular intervals of 30 min (consequently, this has created quite a few missing values for some of the measurements). All models were trained using Adam optimizer with $10^{-2}$ learning rate.

F.2 MODEL ARCHITECTURES

Architectures of the model used in our experiments are summarized in Tables 3, 4, 5.

F.3 DICTIONARIES LEARNED BY CENS

We visualize the full dictionaries learned by CENS on MNIST, IMDB, and Satellite tasks (see Figures 9, 11, 12, 10, 13a) and additional correlation plots between the selected models and the survey variable values (Figure 13b).

9 The model was taken from https://github.com/nealjean/predicting-poverty.
Table 3: Top-performing architectures used in our experiments on MNIST and IMDB datasets.

<table>
<thead>
<tr>
<th>Convolutional Encoder</th>
<th>Sequential Encoder</th>
<th>Contextual Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(a) MNIST</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>layer</td>
<td>Conv2D</td>
<td>model</td>
</tr>
<tr>
<td># filters</td>
<td>32</td>
<td>Logistic reg.</td>
</tr>
<tr>
<td>kernel size</td>
<td>3 × 3</td>
<td>HOG (3, 3)</td>
</tr>
<tr>
<td>strides</td>
<td>1 × 1</td>
<td># of features</td>
</tr>
<tr>
<td>padding</td>
<td>valid</td>
<td>729</td>
</tr>
<tr>
<td>activation</td>
<td>ReLU</td>
<td>dictionary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>layer</td>
<td>l1 penalty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 · 10^{-6}</td>
</tr>
<tr>
<td></td>
<td>pool size</td>
<td>2 × 2</td>
</tr>
<tr>
<td>dropout</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dense</td>
<td></td>
</tr>
<tr>
<td>units</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td># of blocks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td># params</td>
<td>1.2M</td>
<td></td>
</tr>
</tbody>
</table>

| **(b) IMDB**          |                    |                         |
| model                 | VGG-16             |                         |
| pretrained            | No                 |                         |
| fixed weights         | No                 |                         |
| reference             | Zagoruyko.         |                         |
|                      | 2015               |                         |
|                      | layer              | Logistic reg.           |
|                      |                    | HOG (3, 3)              |
|                      | # of features      | 1024                    |
|                      | dictionary         |                           |
|                      |                    | 256                     |
|                      | l1 penalty         | 5 · 10^{-5}             |
|                      | l2 penalty         | 1 · 10^{-6}             |
|                      | dropout            | 0.25                    |
|                      | Dense              |                         |
| units                 | 128                |                         |
| dropout               | 0.25               |                         |
| # params              | 20.0M              |                         |

Table 4: Top-performing architectures used in our experiments on CIFAR10 and Satellite datasets. VGG-16 architecture for CIFAR10 was taken from [https://github.com/szagoruyko/cifar](https://github.com/szagoruyko/cifar) but implemented in Keras with TensorFlow backend. Weights of the pre-trained VGG-F model for the Satellite experiments were taken from [https://github.com/nealjean/predicting-poverty](https://github.com/nealjean/predicting-poverty).

**Table 5: Top-performing architectures used in our experiments on SUPPORT2 and PhysioNet 2012 datasets.**

**(a) SUPPORT2**

<table>
<thead>
<tr>
<th>MLP Encoder</th>
<th>Contextual Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>layer</td>
<td>Dense</td>
</tr>
<tr>
<td>pretrained</td>
<td>No</td>
</tr>
<tr>
<td>fixed weights</td>
<td>No</td>
</tr>
<tr>
<td>units</td>
<td>64</td>
</tr>
<tr>
<td>dropout</td>
<td>0.50</td>
</tr>
<tr>
<td>activation</td>
<td>ReLU</td>
</tr>
</tbody>
</table>

| layer       | LSTM                   |
| pretrained  | No                     |
| fixed weights | No                     |
| units       | 32                     |
| dropout     | 0.25                   |
| activation  |                         |

**(b) PhysioNet Challenge 2012**

<table>
<thead>
<tr>
<th>Sequential Encoder</th>
<th>Contextual Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>layer</td>
<td>LSTM</td>
</tr>
<tr>
<td>bidirectional</td>
<td>Yes</td>
</tr>
<tr>
<td>max length</td>
<td>150</td>
</tr>
<tr>
<td>dropout</td>
<td>0.25</td>
</tr>
<tr>
<td>activation</td>
<td></td>
</tr>
</tbody>
</table>

| layer              | LSTM                   |
|                   |                         |
| pretrained        | No                     |
| fixed weights      | No                     |
| units              | 32                     |
| dropout            | 0.25                   |
| activation         |                         |
Figure 10: Histograms of test weights assigned by CEN to 6 topics: Acting- and plot-related topics (upper charts), genre topics (bottom charts). Note that acting-related topics are often bi-modal, i.e., contributing either positive, negative, or zero weight to the sentiment prediction in different contexts. Genre topics almost always have negligible contributions. This allows us to conclude that the learned model does not have any particular biases towards or against any a given genre.
Figure 11: Visualization of the model dictionary learned by CEN on MNIST. Each row corresponds to a dictionary element, and each column corresponds to the weights of the model voting for each class of digits. Images visualize the weights of the models. Red corresponds to high positive values, dark gray to high negative values, and white to values that are close to 0.
Figure 12: The full dictionary learned by CEN<sub>1pc</sub> model: rows correspond to topics and columns correspond to dictionary atoms. Very small values were thresholded for visualization clarity. Different atoms capture different prediction patterns; for example, atom 5 assigns a highly positive weight to the \{kid, child, disney, family\} topic and down-weighs \{sexual, violence, nudity, sex\}, while atom 11 acts in an opposite manner. Given the context of the review, CEN combines just a few atoms to make a prediction.
(a) Full visualization of models M1 and M2 learned by CEN on Satellite data.

(b) Correlation between the selected model and the value of a particular survey variable.

Figure 13: Additional visualizations for CENs trained on the Satellite data.
REFERENCES


