A General Framework for Visualizing Embedding Spaces of Neural Survival Analysis Models Based on Angular Information

George H. Chen GEORGECHEN@CMU.EDU
Carnegie Mellon University, USA

Abstract

We propose a general framework for visualizing any intermediate embedding representation used by any neural survival analysis model. Our framework is based on so-called *anchor directions* in an embedding space. We show how to estimate these anchor directions using clustering or, alternatively, using user-supplied "concepts" defined by collections of raw inputs (e.g., feature vectors all from female patients could encode the concept "female"). For tabular data, we present visualization strategies that reveal how anchor directions relate to raw clinical features and to survival time distributions. We then show how these visualization ideas extend to handling raw inputs that are images. Our framework is built on looking at angles between vectors in an embedding space, where there could be "information loss" by ignoring magnitude information. We show how this loss results in a "clumping" artifact that appears in our visualizations, and how to reduce this information loss in practice.

Data and Code Availability We use the publicly available datasets on predicting time until death from the Study to Understand Prognoses, Preferences, Outcomes, and Risks of Treatment (SUPPORT) (Knaus et al., 1995), the Rotterdam tumor bank (Rotterdam) (Foekens et al., 2000), and the German Breast Cancer Study Group (GBSG) (Schumacher et al., 1994). The SUPPORT dataset is on severely ill hospitalized patients with various diseases whereas the Rotterdam and GBSG datasets are both on breast cancer. We also use the MNIST handwritten digits dataset (LeCun et al., 2010) modified by Pölsterl (2019) to be for survival analysis. Our code is publicly available (links to the datasets we use are in our code): https://github.com/georgehc/anchor-vis/

Institutional Review Board (IRB) Our research does not require IRB approval as we are conducting secondary analyses of existing publicly available datasets that do not have access restrictions.

1. Introduction

Survival analysis models regularly arise in health applications in reasoning about how much time will elapse before a critical event happens, such as death, disease relapse, and hospital readmission. Across many health-related datasets, state-of-the-art survival analysis models commonly use neural networks (e.g., Ranganath et al. 2016; Chapfuwa et al. 2018; Katzman et al. 2018; Lee et al. 2018; Kvamme et al. 2019; Nagpal et al. 2021; Chen 2020, 2022; Li et al. 2020; Zhong et al. 2021, 2022; Manduchi et al. 2022). However, most neural survival analysis models have been developed with a focus on prediction accuracy, often without examining what these models have learned internally. In more detail, these models typically represent individual patients in terms of "embedding vectors". How do these embedding vectors relate to patient characteristics? How do they relate to survival (or time-to-event) outcomes?

Some existing neural survival analysis models have been designed to have interpretable components. For instance, the model by Zhong et al. (2022) uses a partially linear Cox model: variables that the modeler wants to easily reason about are captured by a linear component of the model whereas the rest of the variables are modeled by a neural network. Meanwhile, Chapfuwa et al. (2020), Nagpal et al. (2021), Manduchi et al. (2022), and Chen (2022) all represent a data point (i.e., a patient) in terms of clusters, where we can summarize each cluster's patient characteristics and survival distributions. Along similar lines, Li et al. (2020) introduced a neural topic model with survival supervision, which represents each patient as a combination of "topics", where each topic corresponds to specific patient characteristics being more probable and to either higher or lower survival times.

In this paper, rather than developing a new survival analysis model that aims to be in some sense interpretable, our main contribution is instead to propose a general framework for visualizing intermediate

representations of *any* neural survival analysis model. Crucially, our framework is based on analyzing *angular information*. Specifically, our framework uses what we refer to as *anchor directions* in an intermediate representation space. We specifically show:

- how to estimate anchor directions based on clustering or, alternatively, based on a "concept" that the user provides, where the concept is represented as a collection of data points (e.g., a set of feature vectors all for female patients could represent the concept "female"; this is the same definition of "concepts" as used by Kim et al. (2018));
- how to visualize raw features vs anchor directions, and survival time distributions vs anchor directions;
- how to tell if our visualizations are "losing too much information" by focusing on angular information (and ignoring magnitude information), and how to reduce this information loss.

Our framework could be thought of as a suite of visualization tools with accompanying statistical tests that can help quantify the strength of associations related to the embedding space under examination.

To showcase our framework, we first focus on a tabular dataset on predicting time until death of patients from the Study to Understand Prognoses, Preferences, Outcomes, and Risks of Treatment (SUPPORT) (Knaus et al., 1995). We then show how our visualization ideas extend to working with images, where we use the semi-synthetic Survival MNIST dataset (LeCun et al., 2010; Pölsterl, 2019) with known ground truth structure. Our visualizations help us see to how well a neural network's embedding space captures this known structure. We provide a second tabular data example on survival times of breast cancer patients in Appendix C, using data from the Rotterdam tumor bank (Rotterdam) (Foekens et al., 2000) and the German Breast Cancer Study Group (GBSG) (Schumacher et al., 1994).

The only baseline visualization strategy we are aware of that works with any intermediate representation of any neural survival analysis model is to apply a dimensionality reduction method (such as PCA (Pearson, 1901) or t-SNE (Van der Maaten and Hinton, 2008)) to transform the intermediate representation of interest (which could be high-dimensional) into a 1D, 2D, or 3D representation that is displayed in a scatter plot. Points in the scatter plot could be colored based on, for instance, their median survival times as predicted by the neural survival analysis model. We provide examples of such scatter plots in Appendix H. While this baseline visualization strategy can provide some rough intuition of the geometry of the embedding space, it does not—on its own—do clustering or provide quantitative metrics relating the embedding space to raw features or to predicted survival time distributions. Our framework could be used in addition to this baseline visualization strategy and does relate the embedding space to raw features and to survival time distributions, with the help of anchor directions based on clustering or on concepts.

Separately, even though many visualization tools have been developed for neural network models for classification or for predicting a single scalar output (e.g., Selvaraju et al. 2016; Zhou et al. 2016; Dabkowski and Gal 2017; Lundberg and Lee 2017; Shrikumar et al. 2017; Smilkov et al. 2017; Sundararajan et al. 2017; Kim et al. 2018), these existing visualization tools do not easily extend to the survival analysis setting. A key reason is that these tools aim to quantify how important different input features are in affecting a single output neuron's scalar value. However, in general, the prediction target in survival analysis for a single test data point is not a single scalar value and is instead a probability distribution over time, where time could either be continuous or discrete. When the time is discrete, the number of time steps used is up to the modeler and could even scale with the number of training data points. Quantifying the importance of different input features in predicting such a survival time distribution is not straightforward.

2. Background

We review the standard survival analysis setup in Section 2.1, and then we provide an example of a neural survival analysis model in Section 2.2. For the latter, we specifically review the now-standard DeepSurv model (Katzman et al., 2018). We emphasize that our visualization framework is not limited to only working with DeepSurv and works with any neural survival analysis model. For ease of exposition, we use DeepSurv throughout the paper.

2.1. Survival Analysis

We assume that we have n training patients with data points $\{(x_i, y_i, \delta_i)\}_{i=1}^n$, where the i-th training patient has raw input $x_i \in \mathcal{X}$ (e.g., tabular data, images), observed time $y_i \in [0, \infty)$, and event indicator $\delta_i \in \{0, 1\}$; if $\delta_i = 1$, then this means that the critical event of interest (e.g., death) happened for the i-th training patient so y_i is the time until the critical event happened (also called the "survival time"), whereas if

$\delta_i = 0$, then this means that the critical event did not happen, so y_i is the time until we stopped collecting information on the i-th patient (commonly called the "censoring time").

To model how training points are sampled, we first introduce some notation. We denote the random variable for a generic raw input as X, which has marginal distribution $\mathbb{P}_X$. We denote the random variable for the survival time as T, which depends on raw input X; in particular, there is a conditional distribution $\mathbb{P}_{T|X}$. We also denote the random variable for the censoring time as C, which depends on raw input X via a conditional distribution $\mathbb{P}_{C|X}$. Note that T and C are assumed to be conditionally independent given X. Then each training point (x_i, y_i, δ_i) is assumed to be generated i.i.d. as follows:

1. Sample raw input x_i from $\mathbb{P}_X$.
2. Sample survival time t_i from $\mathbb{P}_{T|X=x_i}$.
3. Sample censoring time c_i from $\mathbb{P}_{C|X=x_i}$.
4. If $t_i \leq c_i$ (the critical event happens before censoring): set $y_i = t_i$ and $\delta_i = 1$. Otherwise, set $y_i = c_i$ and $\delta_i = 0$.

For a patient with raw input $x \in \mathcal{X}$, a survival analysis model estimates a distribution over survival times for x. Specifically, this survival time distribution is specified in terms of the *conditional survival function*

$$S(t|x) \triangleq \mathbb{P}(T > t \mid X = x) \quad \text{for } t \geq 0,$$

or a transformed version of this function, such as the *hazard function* $h(t|x) \triangleq -\frac{\partial}{\partial t} \log S(t|x)$ (by negating, integrating, and exponentiating, one can show that $S(t|x) = \exp(-\int_0^t h(\tau|x)d\tau)$, i.e., having an estimate for $h(\cdot|x)$ yields an estimate for $S(\cdot|x)$).

2.2. Neural Survival Analysis: DeepSurv

The DeepSurv model (Katzman et al., 2018) estimates the hazard function $h(t|x)$ under the standard *proportional hazards assumption* (Cox, 1972):

$$h(t|x) = h_0(t) \exp(f(x;\theta)), \tag{1}$$

where h_0 is called the *baseline hazard function* (which takes as input a nonnegative time $t \geq 0$ and outputs a nonnegative value), and $f(\cdot;\theta)$ is a user-specified neural network with parameters θ (specifically, $f(\cdot;\theta)$ maps a raw input from $\mathcal{X}$ to a single real number that could be thought of as a "risk score", where higher values correspond to the critical event of interest likely happening earlier for feature vector x). For example, when working with tabular data, f could be a multilayer perceptron, and when working with images, f could be a convolutional neural network.

To train a DeepSurv model, we first learn the neural network parameters θ by minimizing the loss

$$L(\theta) \triangleq -\sum_{i=1}^{n} \delta_i \left[f(x_i;\theta) - \log \sum_{\substack{j=1,\ldots,n \\ \text{s.t. } y_j \geq y_i}} \exp(f(x_j;\theta)) \right].$$

After learning θ, we then estimate h_0 using a standard approach such as Breslow's method (Breslow, 1972). Specifically, let $\widehat{\theta}$ denote the learned neural network parameters, let $t_{(1)} < t_{(2)} < \cdots < t_{(\tau)}$ denote the sorted unique times when the critical event happened in the training data (so that the total number of these unique times is τ), and let $d_{(\ell)}$ denote the number of times the critical event happened at time $t_{(\ell)}$ for $\ell = 1, \ldots, \tau$. Then Breslow's method estimates a discretized version of h_0 at time $t_{(\ell)}$ using

$$\widehat{h}_0(t_{(\ell)}) \triangleq \frac{d_{(\ell)}}{\sum_{j=1,\ldots,n \text{ s.t. } y_j \geq t_{(\ell)}} \exp(f(x_j;\widehat{\theta}))}.$$

The conditional survival function $S(t|x)$ can then be estimated by

$$\widehat{S}(t|x) \triangleq \exp\left\{ -\exp(f(x;\widehat{\theta})) \sum_{\substack{\ell=1,\ldots,\tau \\ \text{s.t. } t_{(\ell)} \leq t}} \widehat{h}_0(t_{(\ell)}) \right\}. \tag{2}$$

Note that here, the conditional survival function is estimated along a time grid with a total of τ discretized time points, where τ could scale with the number of training points.

3. Visualization Framework

We now present our visualization framework based on anchor directions. Our framework can work with any neural survival analysis model with a base neural network f and an estimate $\widehat{S}(t|x)$ of the conditional survival function $S(t|x)$. For the rest of the paper, we treat the neural survival analysis model that we aim to provide visualizations for as fixed, meaning that it has already been learned using the training data $\{(x_i, y_i, \delta_i)\}_{i=1}^n$ and its learned parameters $\widehat{\theta}$ will not be modified. Thus, for notational convenience, we now write "$f(x)$" instead of "$f(x;\widehat{\theta})$".

At a high level, our framework works as follows. First, we need to decide what intermediate "embedding space" of the base neural network f to visualize (Section 3.1). Next, we estimate anchor directions in the chosen embedding space (Section 3.2). Our visualizations will then be based on how well embedding

vectors align with anchor directions in terms of cosine similarity (Section 3.3). We explain our visualization strategies via real data examples, first with tabular data (Section 3.4) and then with images (Section 3.5). Our visualization strategy focuses on using angular and not magnitude information within the embedding space, which could result in "information loss" in our visualizations. We provide details on this information loss and how to mitigate it (Section 3.6).

Statistical assumptions and sample splitting. Our exposition will be clear about statistical assumptions, which ensure that the different steps of our visualization framework are theoretically sound (and we will be clear when a step is a heuristic and lacks theoretical justification). For example, we occasionally use statistical tests, which commonly require that the input data to the tests are i.i.d. With such considerations in mind, we assume that we have access to additional data separate from the training data $\{(x_i, y_i, \delta_i)\}_{i=1}^n$: specifically, we assume that we also have "anchor direction estimation" data $\{(x_i^{\mathrm{A}}, y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$ sampled i.i.d. in the same manner as the training data, and also "visualization raw inputs" $\{x_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$ sampled i.i.d. from $\mathbb{P}_X$ that are separate from both training and anchor direction estimation data.[1] The basic workflow is as follows: we learn the neural survival analysis model using training data. Afterward, we estimate anchor directions using anchor direction estimation data. Finally, we produce visualizations based on the estimated anchor directions with the help of visualization raw inputs.

3.1. Which "Embedding Space" to Visualize?

First, we need to specify which space we will try to visualize. To go with the DeepSurv example from earlier, the neural network f used with DeepSurv could be a multilayer perceptron. In this case, we could visualize the representation at one of the intermediate layers such as the representation right before the last fully-connected layer (this last layer outputs a single number corresponding to the risk score). Specifically,

$$f(x) = g(\phi(x)), \tag{3}$$

where the function ϕ maps from the raw input space $\mathcal{X}$ to some intermediate Euclidean space $\mathbb{R}^d$, and the function g maps from $\mathbb{R}^d$ to $\mathbb{R}$. For the rest of the paper, we refer to ϕ as the *encoder*, which converts a raw input into an *embedding vector*.[2]

An embedding vector by itself is not very informative. Instead, our visualizations depend on the distribution of these embedding vectors. Recall from Section 2.1 that the distribution of raw inputs is given by $\mathbb{P}_X$. We define the *embedding space* as follows.

Definition 1 *Let $\mathbb{P}_X$ denote the distribution of raw input data. Suppose that we sample $X \sim \mathbb{P}_X$. Then we set $U = \phi(X)$, where ϕ is the encoder. Then we define the* embedding space *as the distribution of U, denoted as $\mathbb{P}_U$. This distribution is over $\mathbb{R}^d$.*

Crucially, we define the embedding space as a distribution over $\mathbb{R}^d$—not just $\mathbb{R}^d$ without a distribution.

To visualize the embedding space $\mathbb{P}_U$, our framework involves first choosing "interesting" *anchor directions* in the embedding space to look at, which we discuss in detail in the next section. Importantly, we argue that in practice, the d axis-aligned directions of $\mathbb{R}^d$ are not necessarily the "interesting" directions for the application at hand. For example, if $\mathbb{P}_U$ is well-approximated by a clustering model with k clusters, then the "meaningful" directions to consider could be the directions that point toward the k different cluster centers. These directions need not be axis-aligned.

Moreover, the embedding dimension d is often a hyperparameter to be tuned (as part of the neural net architecture of f). If the problem of interest has an underlying ground truth distribution that is based on k clusters, then a "good" choice of neural survival analysis model would be one where for a wide range of values for hyperparameter d (where $d \geq k$), after learning the neural survival analysis model, the resulting learned embedding space $\mathbb{P}_U$ should consist of k sufficiently separated clusters within $\mathbb{R}^d$.

3.2. Choosing Anchor Directions

As stated previously, we estimate anchor directions using the anchor direction estimation data $\{(x_i^{\mathrm{A}}, y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$. We denote the embedding vectors of these points by $u_i^{\mathrm{A}} \triangleq \phi(x_i^{\mathrm{A}})$. Note that ϕ is estimated as part of learning the neural survival analysis model using training data $\{(x_i, y_i, \delta_i)\}_{i=1}^n$. Conditioned on the original training data and on the encoder ϕ, the embedding vectors $u_1^{\mathrm{A}}, \ldots, u_{n^{\mathrm{A}}}^{\mathrm{A}}$ appear

[1] Our framework works even if the training data were sampled differently from the rest of the data; see Appendix C.

[2] Note that for equation (3), our framework does *not* require the function g to output a single real number. This happens to be the case for DeepSurv. For example, DeepHit (Lee et al., 2018) has a base neural network f that outputs a Euclidean vector instead. Our visualization framework trivially also supports these other base neural network functions.

as i.i.d. draws from the embedding space $\mathbb{P}_U$ (this i.i.d. property would fail to hold if the anchor direction estimation data were the same as the training data; see Appendix A.1 for details). We now provide two approaches for estimating anchor directions.

3.2.1. Clustering Approach

We first estimate anchor directions using clustering. This approach is agnostic to the choice of clustering algorithm but assumes that the clustering algorithm chosen only has access to the embedding vectors $u_1^{\mathrm{A}}, \ldots, u_{n^{\mathrm{A}}}^{\mathrm{A}}$. In particular, we ignore the survival information $\{(y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$ for the moment. Note that many clustering algorithms, such as the Expectation-Maximization algorithm for Gaussian mixture models (Bishop, 2006, Section 9.2), are derived under the assumption that the data points to be clustered are i.i.d. Again, this assumption holds since $u_1^{\mathrm{A}}, \ldots, u_{n^{\mathrm{A}}}^{\mathrm{A}}$ appear as i.i.d. samples from $\mathbb{P}_U$ after we condition on the original training data and ϕ.

Once we have learned the clustering model of our choice, we obtain a cluster assignment for each embedding vector. In particular, we denote the cluster assignment of the i-th embedding vector u_i^{A} by $z_i^{\mathrm{A}} \in \{1, 2, \ldots, k\}$, where k is the number of clusters. Then we define the j-th cluster's anchor direction as

$$\mu_{\text{cluster } j} \triangleq \underbrace{\frac{\sum_{i=1}^{n^{\mathrm{A}}} u_i^{\mathrm{A}} \mathbb{1}\{z_i^{\mathrm{A}} = j\}}{\sum_{i=1}^{n^{\mathrm{A}}} \mathbb{1}\{z_i^{\mathrm{A}} = j\}}}_{\substack{\text{mean direction of} \\ \text{embedding vectors in} \\ \text{the } j\text{-th cluster}}} - \underbrace{\frac{1}{n^{\mathrm{A}}} \sum_{i=1}^{n^{\mathrm{A}}} u_i^{\mathrm{A}}}_{\substack{\text{mean direction across} \\ \text{embedding vectors} \\ \triangleq\, \overline{u}^{\mathrm{A}}}}, \quad (4)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function that is 1 when its input is true and 0 otherwise. The second term $\overline{u}^{\mathrm{A}}$ is the "center of mass" of the embedding vectors. Thus, by subtracting $\overline{u}^{\mathrm{A}}$, we focus on how the clusters differ from the center of mass. This is essentially changing the center of the coordinate system so that directions are measured treating $\overline{u}^{\mathrm{A}}$ as the "origin" of the new coordinate system. A similar idea is commonly used in principal component analysis, in which data are first centered (Abdi and Williams, 2010).

Choosing the number of clusters based on survival information. Clustering algorithms often have a hyperparameter (or multiple) that affects the number of clusters k. For example, a Gaussian mixture model has a hyperparameter for the number of mixture components, which could be thought of as clusters. We now propose a heuristic for choosing k using survival information $\{(y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$.

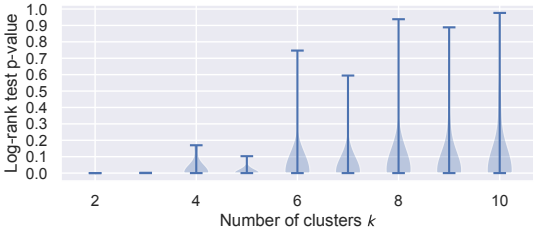

Figure 1: A violin plot to help select the number of clusters k to use. Here, the encoder is from a DeepSurv model trained on the SUPPORT dataset, and the clustering model is a mixture of von Mises-Fisher distributions. We plot the set $\Psi(k)$ (consisting of log-rank test p-values; see equation (5)) vs k. Details on the dataset, encoder, and clustering model are in Section 3.4.

Suppose that we have grouped the data into k clusters. For any two clusters $j, j' \in \{1, \ldots, k\}$ with $j \neq j'$, we can run the log-rank test (Mantel, 1966) to quantify how different these two clusters' survival outcomes are (running this test requires using the y_i^{A} and δ_i^{A} variables of the points in clusters j and j'). We denote the test's resulting p-value as $\psi_{j,j'}(k)$. Then the set of p-values across all pairs of clusters found is

$$\Psi(k) \triangleq \big\{\psi_{j,j'}(k) \text{ for all } j \in \{1, \ldots, k\}, \text{ and } j' \in \{j+1, \ldots, k\}\big\}. \quad (5)$$

We can re-run the clustering algorithm to get cluster assignments that have different numbers of clusters k so that for each k, we have a different set of p-values $\Psi(k)$. We plot the entire set $\Psi(k)$ as a function of k in Figure 1 using a violin plot (i.e., for each k, we see the distribution of $\Psi(k)$ as a "violin"). We can choose k to be a value before the p-values in $\Psi(k)$ become "too large". For example, in Figure 1, the "violins" in the violin plot get much taller (so the p-values in $\Psi(k)$ are overall getting much larger) after 5 clusters, so we could choose $k = 5$.

This procedure for choosing k is a heuristic: we have found it work well in practice but we currently lack theory to justify when it recovers the "correct" number of clusters. We comment on when the log-rank test is theoretically sound to apply (so that the p-values are valid) in Appendix A.2. We further discuss a heuristic for choosing which clusters to focus on when the number of clusters is large in Appendix G; this heuristic is based on estimating a survival time for each anchor direction, ranking anchor directions based

on these survival time estimates, and focusing only on anchor directions with particular ranks (e.g., anchor directions with the highest and lowest survival times).

3.2.2. User-supplied "Concepts"

Next, we consider a scenario where the user provides a "concept" in terms of a collection of example raw input data $\mathcal{C} \triangleq \{x_1^\dagger, \ldots, x_{n^\dagger}^\dagger\}$, where all $n^\dagger$ of these examples exhibit the concept (e.g., to convey a concept corresponding to "female", the user can set $\mathcal{C}$ to be a collection of raw inputs from female patients). This is the same notion of concepts as used by Kim et al. (2018). We assume that the set $\mathcal{C}$ is collected in a manner that is independent of how the visualization raw inputs $\{x_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$ are collected (this assumption will be needed later for statistical tests). Specifically, we use the anchor direction

$$\mu_{\text{concept } \mathcal{C}} \triangleq \frac{1}{|\mathcal{C}|} \sum_{x^\dagger \in \mathcal{C}} \phi(x^\dagger) - \overline{u}^{\mathrm{A}}, \tag{6}$$

where the center of mass $\overline{u}^{\mathrm{A}}$ is defined in equation (4).

3.3. Key Visualization Quantity: Projections onto Anchor Directions

In this section, we use $\mu \in \mathbb{R}^d$ to denote any specific anchor direction that we aim to analyze, such as the ones from equations (4) or (6). Our visualization framework focuses on angular information in the embedding space. Specifically, for any raw input $x \in \mathcal{X}$, we define the following projection onto μ:

$$\text{proj}_\mu(x) \triangleq \frac{\langle \phi(x) - \overline{u}^{\mathrm{A}}, \mu \rangle}{\|\phi(x) - \overline{u}^{\mathrm{A}}\| \|\mu\|}, \tag{7}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product, $\| \cdot \|$ denotes taking the Euclidean norm, and $\overline{u}^{\mathrm{A}}$ is defined in equation (4). This projection is precisely the cosine similarity between vectors $(\phi(x) - \overline{u}^{\mathrm{A}})$ and μ, which looks at how well-aligned these two vectors are, disregarding their magnitudes (since we divide by their norms in equation (7)). We discuss implications of ignoring magnitude information and how to reduce the amount of "information loss" in Section 3.6. Since the cosine similarity of two vectors is always between -1 (the two vectors point in exactly opposite directions) and 1 (the two vectors poin in exactly the same direction), we are guaranteed that $\text{proj}_\mu(x) \in [-1, 1]$.

To create visualizations, we plug in the visualization raw inputs $\{x_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$ into the projection operator proj_μ. For notation, we write $p_i^{\mathrm{V}} \triangleq \text{proj}_\mu(x_i^{\mathrm{V}})$ to be the projection of the i-th visualization input along anchor direction μ. Our 2D visualizations commonly have the x-axis correspond to these projection values while the y-axis will track quantities related to either raw input feature values or time-to-event outcomes.

3.4. Visualization Strategies for Tabular Data

We begin by considering tabular data, where the raw inputs are feature vectors from $\mathcal{X} = \mathbb{R}^D$ and each of the D features is assumed to be "easy to interpret" (e.g., age, gender, cancer status). We show how to relate projection values along an anchor direction to, at first, a single continuous feature (such as age) via a scatter plot (Section 3.4.1). Next, we show how to relate projection values to any continuous or discrete raw feature using a heatmap (Section 3.4.2). The heatmap from Section 3.4.2 reveals that some raw features might be more "important" for an anchor direction than others. We discuss statistical tests that can identify or help rank variables that are "important" for a specific anchor direction (Section 3.4.3). Lastly, we show how to relate projection values to the neural survival analysis model's predicted survival time distributions (Section 3.4.4).

Data and setup. As we progress through our visualization strategies, we apply them to the SUPPORT dataset (Knaus et al., 1995). This dataset has 8,873 data points (patients) and 14 features and is on predicting time until death for severely ill hospitalized patients with various diseases. We use a 70%/30% train/test split. We train a DeepSurv model, where the base neural network f is a multilayer perceptron consisting of the following sequence of layers:

- Fully-connected layer (mapping $\mathbb{R}^D$ to $\mathbb{R}^d$)
- Nonlinear activation: ReLU
- Fully-connected layer (mapping $\mathbb{R}^d$ to $\mathbb{R}^d$)
- Nonlinear activation: ReLU

$\vdots$

- Fully-connected layer (mapping $\mathbb{R}^d$ to $\mathbb{R}^d$)
- Nonlinear activation: Divide each vector by its Euclidean norm
- Fully-connected layer (mapping $\mathbb{R}^d$ to $\mathbb{R}$)

The second-to-last bullet point does not use ReLU activation and instead normalizes vectors to have Euclidean norm 1. We design the architecture in this manner since we shall set the embedding space that we visualize to be the representation immediately after this second-to-last bullet point's layer, i.e., the encoder ϕ consists of all layers except for the last fully-connected layer. Thus, the output of ϕ has information stored purely in terms of angles and not

magnitudes (since $\|\phi(x)\| = 1$ for all $x \in \mathcal{X}$). This helps reduce information loss in our visualizations (more details are in Section 3.6). We also show visualizations in Appendix B.3 where this second-to-last bullet point uses ReLU activation instead.

When training DeepSurv, we hold out 20% of the training set to treat as validation data for hyperparameter tuning. The hyperparameter grid used (e.g., number of fully-connected layers, embedding space dimension d, optimizer learning rate and batch size) is stated in Appendix B.1. After training DeepSurv (including hyperparameter tuning), the learned model achieves a test set concordance index (Harrell et al., 1982) of 0.617. We provide this accuracy metric for reference; our visualization framework can be used regardless of the accuracy of the model (ideally, an accurate model should have an embedding space that "captures" application-specific structure).

As stated above, we take the encoder ϕ to be all the layers of f prior to the last fully-connected layer. We split the test set so that 25% of it is used as the anchor direction estimation data $\{(x_i^{\mathrm{A}}, y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$; the rest is used to obtain visualization raw inputs $\{x_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$.

Anchor directions are estimated via clustering as described in Section 3.2.1. Since we have designed the neural network architecture so that all outputs of encoder ϕ have Euclidean norm 1, clustering can be done using a mixture of von Mises-Fisher distributions (von Mises, 1918), which is the analogue of the Gaussian mixture model for Euclidean vectors with norm 1. We specifically fit this mixture model using the Expectation-Maximization algorithm implementation by Kim (2021) and choose the number of clusters k based on the heuristic we presented in Section 3.2.1. In fact, Figure 1 is precisely the plot we get. Throughout this section, we set the number of clusters to be $k = 5$ and, for illustrative purposes, we only show visualizations for the first cluster found. Visualizations for all 5 clusters and interpretations of these visualizations are in Appendix B.2, where we also discuss results when using other numbers of clusters and a different clustering algorithm altogether.

We separately also apply our visualization framework to tabular data on survival times of breast cancer patients. Specifically, we visualize an embedding space of a DeepSurv model trained using the Rotterdam dataset (Foekens et al., 2000). We treat the GBSG dataset (Schumacher et al., 1994) as the test data (that we split into anchor estimation and visualization data). Due to space contraints, we defer the visualization results for this setup to Appendix C, where we also explain why our framework remains valid when training and test data are independent of each other and come from different distributions.

3.4.1. Visualizing an Anchor Direction with a Single Continuous Raw Feature

Let μ be one of the anchor directions estimated, and let $j \in \{1, 2, \ldots, D\}$ be the raw feature that we want to visualize, where we assume that this feature is continuous-valued. Using the visualization raw inputs $\{x_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$, we compute the projections $p_1^{\mathrm{V}} \triangleq \mathrm{proj}_\mu(x_1^{\mathrm{V}}), \ldots, p_{n^{\mathrm{V}}}^{\mathrm{V}} \triangleq \mathrm{proj}_\mu(x_{n^{\mathrm{V}}}^{\mathrm{V}})$, and we write $(x_i^{\mathrm{V}})_j$ to mean the j-th coordinate of vector x_i^{V}. Then we can make a scatter plot of $(p_1^{\mathrm{V}}, (x_1^{\mathrm{V}})_j), (p_2^{\mathrm{V}}, (x_2^{\mathrm{V}})_j), \ldots, (p_{n^{\mathrm{V}}}^{\mathrm{V}}, (x_{n^{\mathrm{V}}}^{\mathrm{V}})_j)$. As a concrete example of this, for the DeepSurv model trained on the SUPPORT dataset, using the anchor direction corresponding to the first cluster found, and using "age" as the raw continuous feature to be visualized, we obtain the plot in Figure 2. We see that as the projection value increases (where a value of 1 maximally aligns with the anchor direction), the age distribution tends to shift upward, suggesting that this anchor direction is associated with older patients.

3.4.2. Visualizing an Anchor Direction with Discrete or Continuous Raw Features

We can modify the above visualization idea to handle a discrete feature by having the y-axis of the plot correspond to specific discrete values that the feature can take on, and separately discretizing the x-axis into a user-specified number of bins (e.g., evenly spaced bins from the minimum to maximum observed projection values). In doing so, we replace the scatter plot with what we call a *raw feature probability heatmap*, where the intensity at the i-th row and j-th column is the fraction of visualization data patients in the j-th projection value bin who have the i-th row's feature value. Even for a continuous feature, we can discretize it based on a user-specified discretization strategy (e.g., based on quartiles) so that all features (continuous or discrete) can be visualized together as a large heatmap. Using the same anchor direction as in Figure 2, we get the resulting heatmap in Figure 3, where the different underlying features are separated by black horizontal lines in the heatmap.

Along the x-axis of the heatmap, the projection bins in this case are 7 evenly spaced bins between the minimum and maximum observed projection values -0.99

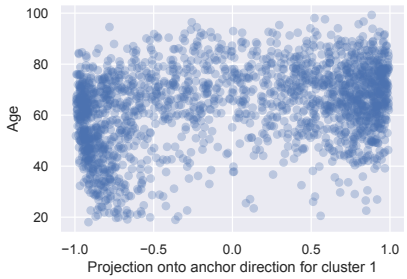

Figure 2: Scatter plot of a single continuous raw feature (age) vs projection values along cluster 1's anchor direction. Plots for all clusters are in Appendix B.2 (Figure B.1).

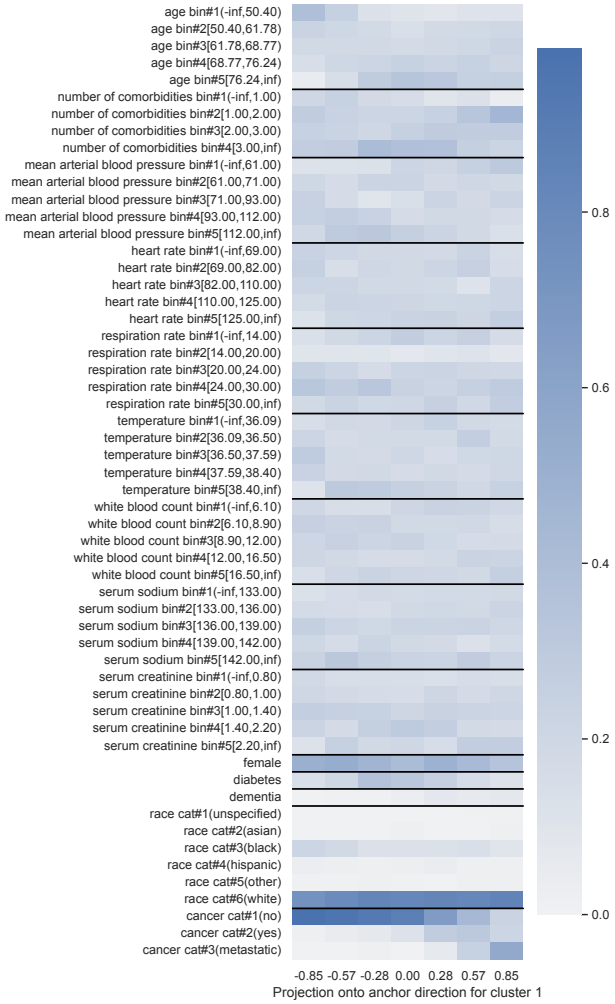

Figure 3: Raw feature probability heatmap: the intensity of the i-row, j-th column indicates the fraction of visualization data in the j-th projection bin that have the i-th row's feature value. Heatmaps for all clusters are in Appendix B.2 (Figure B.2).

and 0.99 respectively. The first (leftmost) bin corresponds to the interval $\mathcal{P}_1 = [-0.99, -0.71)$ with midpoint value $-0.85 = \frac{1}{2}(-0.99 - 0.71)$, the second bin corresponds to the interval $\mathcal{P}_2 = [-0.71, -0.43)$ with midpoint value -0.57, and so forth. The last (rightmost) bin corresponds to the interval $\mathcal{P}_7 = [0.71, 0.99]$ with midpoint value 0.85; only this last bin's interval includes the right endpoint.

We can readily see some trends in Figure 3. As already revealed in Figure 2, age tends to increase as the projection value increases for cluster 1's anchor direction. However, we also see other trends as the projection value increases, such as the number of comorbidities tending to be at least 1 or cancer status tending to be "metastatic". In particular, this cluster seems to correspond to patients who are more ill. Indeed, these patients tend to have shorter predicted survival times, as we show later in Section 3.4.4.

3.4.3. Statistical Tests to Find "Important" Variables for an Anchor Direction

In Figure 3, some features have noticeable trends as the projection value increases whereas others do not. We may want to focus on features that are the "most important" as they relate to anchor direction μ (especially for datasets with a large number of features, displaying all features would be impractical). We now show how to rank raw features using statistical tests of association between two variables, for which one of the variables we take to be the projection value along μ (or a discretized version of this projection value), and the other variable will be one of the D raw features (or a discretized version of it).

The first test we could use to rank features is based on the heatmap visualization from Figure 3: consider a single raw feature (such as "white blood count") and note that the heatmap restricted to that raw feature (i.e., the heatmap that only looks at the discretized white blood count vs discretized projections) is a contingency table, for which we can run Pearson's chi-squared test to assess whether white blood count and the projection value are independent. We could repeat this for all the different raw features (note that for raw features that are indicator random variables, we would have to add a row corresponding to one minus the indicator before running the statistical test) and rank the raw features based on the p-values obtained. Doing so, we obtain the ranking shown in Table 1. Again, this ranking could be used in constructing raw

Table 1: Ranking of raw features based on the p-value of Pearson's chi-squared test (for cluster 1, the same cluster visualized in Figures 2 and 3); rankings for all clusters are provided in Appendix B.2 (Table B.1).

Rank	Feature	p-value
1	cancer	2.86×10^{-225}
2	age	1.04×10^{-45}
3	number of comorbidities	1.91×10^{-24}
4	mean arterial blood pressure	9.25×10^{-19}
5	diabetes	2.46×10^{-14}
6	dementia	4.15×10^{-11}
7	temperature	1.58×10^{-10}
8	heart rate	3.34×10^{-9}
9	female	1.07×10^{-6}
10	serum creatinine	1.21×10^{-6}
11	race	3.42×10^{-5}
12	white blood count	7.02×10^{-5}
13	respiration rate	4.54×10^{-4}
14	serum sodium	6.14×10^{-2}

feature probability heatmaps, where we choose to only visualize a few top-ranked features.

A limitation of this chi-squared approach is that it requires projection values and raw features to be discretized. If the raw features are all continuous or ordinal, then one could use a different statistical test to compare projection values (without discretization) to each raw feature (also without discretization), such as using Kendall's tau test (Kendall, 1938) (which checks for a monotonic relationship between a pair of variables). If the raw features are all categorical, then instead the Kruskal-Wallis test by ranks could be used (Kruskal and Wallis, 1952).

Importantly, when using any of these statistical tests mentioned above, we suggest that the modeler check the assumptions of the test to see whether the test is appropriate for the particular dataset under examination. One of the common assumptions the tests have are that the input data points given to the tests are independent, which we have taken care to achieve by having the visualization data be separate from the training and anchor direction estimation data.

Separately, note that we have not stated how to pick a threshold for how small a p-value should be to flag a raw feature as "important". From a visualization standpoint, we think that providing a ranking is sufficient; the modeler can arbitrarily decide on how many top features to focus on or to visualize, which is equivalent to choosing an arbitrary p-value threshold. We discuss how to choose a threshold that controls for a desired false discovery rate in Appendix A.3.

Lastly, an important limitation of the general strategy we have stated for ranking raw features is that each statistical test is applied in a manner where we do *not* account for possible interactions between different raw features. We mention two methods that could help determine interactions in Appendix D.

3.4.4. Visualizing an Anchor Direction with Time-to-Event Outcomes

To relate an anchor direction to time-to-event outcomes, we again use a heatmap. We set the x-axis to be the same as in Figure 3. As stated in Section 3.4.2, the x-axis (corresponding to projection values along the anchor direction for cluster 1) has been discretized into projection value bins that are intervals. We specifically had $\mathcal{P}_1 = [-0.99, -0.71), \mathcal{P}_2 = [-0.71, -0.43), \ldots, \mathcal{P}_7 = [0.71, 0.99]$. Then note that the j-th projection interval $\mathcal{P}_j$ corresponds to the following visualization data points:

$$\mathcal{I}_j \triangleq \{i \in \{1, 2, \ldots, n^{\mathrm{V}}\} \text{ s.t. } \mathrm{proj}_{\mu}(x_i^{\mathrm{V}}) \in \mathcal{P}_j\}. \quad (8)$$

Then letting $\widehat{S}(t|x)$ denote the neural survival analysis model's prediction of the conditional survival function for raw input x (e.g., for DeepSurv, we use equation (2)), we can compute the following average predicted survival function for the j-th projection bin:

$$\widehat{S}_j(t) \triangleq \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \widehat{S}(t|x_i^{\mathrm{V}}). \quad (9)$$

We plot the function $\widehat{S}_j$ as the j-th column of the heatmap, where the y-axis uses a discretized time grid (e.g., evenly spaced time points between the minimum and maximum observed times in the training data). The resulting heatmap (which we call a *survival probability heatmap*) is in Figure 4. For high projection values (e.g., looking at the rightmost column), the survival probability decays quickly as time increases (starting from the bottom and going upward in the heatmap), suggesting that the patients whose embedding vectors align the most with this anchor direction tend to have short survival times.

3.5. Handling Images as Raw Inputs

We now turn to working with images as raw inputs, where we intentionally examine a dataset that has known ground truth structure for what the embedding space should capture. This helps us see to what extent the learned embedding space we examine recovers this ground truth structure. Specifically, we use the Survival MNIST dataset, which is the MNIST handwritten digit dataset (LeCun et al., 2010) modified by Pölsterl (2019) to have survival labels, i.e., observed times and event indicators. How Pölsterl

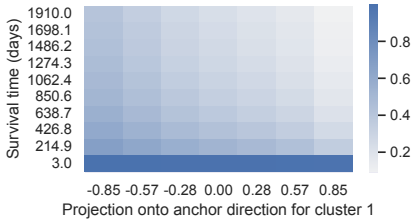

Figure 4: Survival probability heatmap: using the same anchor direction as in Figure 3, we show how projection values along this anchor direction relate to predicted survival probabilities over time; the intensity at the i-th row and j-th column is the survival probability for the j-th projection bin at the i-th discretized time. Heatmaps for all clusters are in Appendix B.2 (Figure B.3).

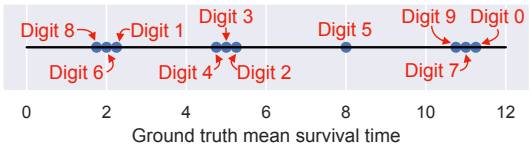

Figure 5: Survival MNIST dataset: each digit has a different ground truth mean survival time.

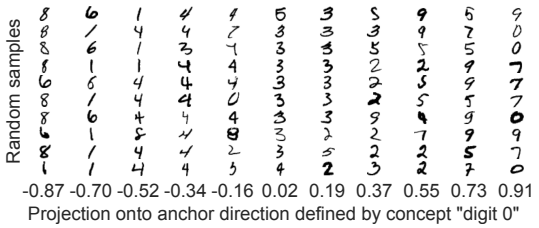

Figure 6: Survival MNIST random input vs projection plot: we display 10 random visualization raw inputs per projection bin.

generates these labels depends on some distributional settings, for which we use the same settings as Goldstein et al. (2020). In particular, each of the 10 digits has a mean survival time shown in Figure 5. Each image's true survival time is sampled based on the digit of the image (e.g., all images of digit 0 have true survival times sampled i.i.d. from a Gamma distribution with mean 11.25 and variance 10^{-3}). The censoring times are sampled from a uniform distribution so that the overall censoring rate is roughly 50%. Also, all images of digit 0 have observed times that are censored. More dataset details are in Appendix E.1.

Using the training set (60,000 data points: each consists of an image, observed time, and event indicator), we learn a DeepSurv model where the base neural network is a convolutional neural network (architecture and training details are in Appendix E.2). Importantly, the digit labels (which digit each image corresponds to) are *not* available during training. We split the test set (10,000 data points), using 25% of it as anchor direction estimation data and the rest as visualization data. For test data, we assume that we have access to their digit labels, which helps us assess whether the embedding space learns what digits are.

First, we use anchor directions defined by concepts of digits. For example, anchor direction estimation data corresponding to digit 0 represents the concept "digit 0". For digit 0's anchor direction (computed using equation (6)), we can discretize the projection values of visualization raw inputs into projection bins $\mathcal{P}_1, \ldots, \mathcal{P}_m$ where m is the number of bins, just as we did with tabular data. Each projection bin $\mathcal{P}_j$ corresponds to a subset $\mathcal{I}_j$ of visualization data (see equation (8)). For projection bin $\mathcal{P}_j$, we can randomly sample, for instance, 10 raw input images corresponding to points in $\mathcal{I}_j$ and display these images along the y-axis. The resulting *random input vs projection plot* is shown in Figure 6.[3] We see that as the projection values get large, the images that get sampled tend to be of digits 0, 7, and 9. These digits have the highest ground truth mean survival times (see Figure 5).

Due to space constraints, we defer additional Survival MNIST visualizations to Appendix E.3. The key findings are as follows. First, note that the digits have mean survival times ranked as 8, 6, 1, 4, 3, 2, 5, 9, 7, 0 (see Figure 5). We refer to two digits as "adjacent" if they are ranked next to each other (e.g., digits 1 and 4 are adjacent). We find that the learned embedding space tends to have the j-th digit's anchor direction align well with embedding vectors of the j-th digit's images as well as those of other adjacent digits (e.g., digit 1 images tend to have high projection values for digit 4's anchor direction). Because the embedding space is not learned in a manner that knows what the different digits are, the 10 digits do not get "disentangled" in the embedding space. Treating data that are censored as a "concept", we find that the embedding space recognizes which digits are more censored than others. Meanwhile, we also show

[3]When the encoder uses a convolutional layer, it is also possible to make a variant of Figure 6 where instead of displaying random raw inputs per projection bin, we display a convolution filter's outputs of these random raw inputs instead.

that if we estimate anchor directions using clustering, the violin plot we use to help select the number of clusters sharply increases in p-values after 9 clusters, as expected (digit 0 is the only one that is always censored making it difficult to learn).

We point out that survival probability heatmaps like the one in Figure 4 are not specific to tabular data as they do not depend on raw features; they can be created the same manner when raw inputs are images. Furthermore, if raw images are converted into a tabular format (e.g., by running object detectors and representing each image as a feature vector specifying how often each object appears), then our tabular data visualization strategies could of course be used.

3.6. Loss of Magnitude Information

Our visualization framework is based on angular information: projection values are cosine similarities, which measure angles between vectors, disregarding their norms (or magnitudes). In the extreme case where the "information content" of the embedding vectors are all in magnitudes and not angles, the only possible projection values are -1 and 1; projection values within the open interval $(-1, 1)$ are not possible. Thus, all our visualizations where the x-axis is based on projection values would only need two projection bins for -1 and 1. We formally state this theoretical result and provide its proof in Appendix F.

When working with real data, the information in the embedding space will typically not be entirely in angles or entirely in magnitudes. As more information is stored in magnitudes, our visualizations based on projection values will start exhibiting this phenomenon where the projection values "clump up" at -1 and 1. An example of this visualization artifact for DeepSurv trained on the SUPPORT dataset (without the nonlinear activation that normalizes the embedding vectors to have norm 1) is in Appendix B.3.

Reducing information loss. Since the modeler can often choose the encoder ϕ when designing the neural network architecture of f, the encoder ϕ could be chosen as to avoid storing information in magnitudes, which would reduce the information lost by using our framework. We had precisely done this in Section 3.4 when we constrained the output of ϕ to have Euclidean norm 1. This "norm 1" constraint alone could still sometimes not lead to enough angular information stored (e.g., if the embedding space $\mathbb{P}_U$ is highly concentrated so that nearly all embedding vectors randomly sampled from it point in almost the same direction). A recent theoretical and empirical insight when working with a "norm 1" constraint (technically referred to as working with vectors on a hypersphere) is to add regularization that encourages the embedding vectors to have angles that are "diverse" (closer to uniformly distributed in all directions). Adding this regularization improves prediction accuracy for a variety of neural network architectures (Wang and Isola, 2020; Liu et al., 2021). This diversity would, in our visualization context, lead to to having the projections onto anchor directions be more dispersed across the interval $[-1, 1]$ instead of being "clumped up" around specific points within $[-1, 1]$.

4. Discussion

We have presented a visualization framework that is meant to help developers of neural survival analysis models better understand what their models have learned. Importantly, the visualizations we have proposed only reveal possible associations related to an embedding space. No causal claims are made. In focusing on examining one embedding space at a time, our framework is not designed to explain how the overall neural survival analysis model actually makes predictions. We believe that our visualization strategies would be helpful in assessing whether a model has internally learned associations that agree with existing clinical literature, or to see if the model surfaces new associations that warrant further investigation.

While we have developed our framework for survival analysis, it can be modified to support other prediction tasks. For example, to support classification, it suffices to make two changes. First, the clustering approach for estimating anchor directions would be unnecessary since we could take the anchor directions to be the average embedding vector for each class, minus the center of mass across embedding vectors. Second, to relate an embedding space to predicted class distributions instead of survival time distributions, we could estimate and visualize the probability of different classes per projection bin instead of using our proposed survival probability heatmaps. Although our framework can easily be adapted to classification, whether it offers any advantages over the many existing visualization tools for classification (e.g., Selvaraju et al. 2016; Zhou et al. 2016; Dabkowski and Gal 2017; Shrikumar et al. 2017; Smilkov et al. 2017; Kim et al. 2018) is unclear. Better understanding the advantages and disadvantages of our framework in prediction tasks beyond survival analysis would be an interesting direction for future research.

References

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

Per K Andersen, Ørnulf Borgan, Richard D Gill, and Niels Keiding. *Statistical Models Based on Counting Processes*. Springer Science & Business Media, 1993.

Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Norman Breslow. Discussion of the paper by David R Cox (1972), cited below. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34: 187–220, 1972.

Paidamoyo Chapfuwa, Chenyang Tao, Chunyuan Li, Courtney Page, Benjamin Goldstein, Lawrence Carin Duke, and Ricardo Henao. Adversarial time-to-event modeling. In *International Conference on Machine Learning*, 2018.

Paidamoyo Chapfuwa, Chunyuan Li, Nikhil Mehta, Lawrence Carin, and Ricardo Henao. Survival cluster analysis. In *Conference on Health, Inference, and Learning*, 2020.

George H Chen. Deep kernel survival analysis and subject-specific survival time prediction intervals. In *Machine Learning for Healthcare*, 2020.

George H Chen. Survival kernets: Scalable and interpretable deep kernel survival analysis with an accuracy guarantee. *arXiv preprint arXiv:2206.10477*, 2022.

David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6970–6979, 2017.

Jack William Dunn. *Optimal Trees for Prediction and Prescription*. PhD thesis, Massachusetts Institute of Technology, 2018.

Thomas R Fleming and David P Harrington. *Counting Processes and Survival Analysis*. John Wiley & Sons, 1991.

John A Foekens, Harry A Peters, Maxime P Look, Henk Portengen, Manfred Schmitt, Michael D Kramer, Nils Brünner, Fritz Jänicke, Marion E Meijer-van Gelder, and Sonja C Henzen-Logmans. The urokinase system of plasminogen activation and prognosis in 2780 breast cancer patients. *Cancer Research*, 60(3):636–643, 2000.

Mark Goldstein, Xintian Han, Aahlad Puli, Adler Perotte, and Rajesh Ranganath. X-CAL: Explicit calibration for survival analysis. In *Advances in Neural Information Processing Systems*, 2020.

Frank E Harrell. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival analysis, 2nd Ed.* Springer, 2015.

Frank E Harrell, Robert M Califf, David B Pryor, Kerry L Lee, and Robert A Rosati. Evaluating the yield of medical tests. *Journal of the American Medical Association*, 247(18):2543–2546, 1982.

Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24, 2018.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, and Fernanda Viegas. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, 2018.

Minyoung Kim. On PyTorch implementation of density estimators for von Mises-Fisher and its mixture. *arXiv preprint arXiv:2102.05340*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

William A Knaus, Frank E Harrell, Joanne Lynn, Lee Goldman, Russell S Phillips, Alfred F Connors, Neal V Dawson, William J Fulkerson, Robert M

Califf, and Norman Desbiens. The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122(3):191–203, 1995.

William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and Cox regression. *Journal of Machine Learning Research*, 2019.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. MNIST handwritten digit database, 2010. URL https://yann.lecun.com/exdb/mnist.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar. DeepHit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Linhong Li, Ren Zuo, Amanda Coston, Jeremy C Weiss, and George H Chen. Neural topic models with survival supervision: Jointly predicting time-to-event outcomes and learning how clinical features relate. In *International Conference on Artificial Intelligence in Medicine*. Springer, 2020.

Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. Learning with hyperspherical uniformity. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777, 2017.

Laura Manduchi, Ričards Marcinkevičs, Michela C Massi, Thomas Weikert, Alexander Sauter, Verena Gotta, Timothy Müller, Flavio Vasella, Marian C Neidert, Marc Pfister, Bram Stieltjes, and Julia E Vogt. A deep variational approach to clustering survival data. In *International Conference on Learning Representations*, 2022.

Nathan Mantel. Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemotherapy Reports*, 50(3):163–170, 1966.

Chirag Nagpal, Xinyu Rachel Li, and Artur Dubrawski. Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

Sebastian Pölsterl. Survival analysis for deep learning, July 2019. URL https://k-d-w.org/blog/2019/07/survival-analysis-for-deep-learning/.

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. In *Machine Learning for Healthcare*, 2016.

Nancy Reid. Estimating the median survival time. *Biometrika*, 68(3):601–608, 1981.

M Schumacher, G Bastert, H Bojar, K Huebner, M Olschewski, W Sauerbrei, C Schmoor, C Beyerle, R L Neumann, and H F Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. German Breast Cancer Study Group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-CAM: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad:

removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? Interpretable attribution for feature interactions. In *Advances in Neural Information Processing Systems*, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

Richard von Mises. Über die "Ganzzahligkeit" der Atomgewicht und verwandte Fragen. *Physikal. Z.*, 19:490–500, 1918.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 2020.

Qixian Zhong, Jonas W. Mueller, and Jane-Ling Wang. Deep extended hazard models for survival analysis. In *Advances in Neural Information Processing Systems*, 2021.

Qixian Zhong, Jonas Mueller, and Jane-Ling Wang. Deep learning for the partially linear Cox model. *The Annals of Statistics*, 50(3):1348–1375, 2022.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

Appendix A. Statistical Considerations

A.1. What Happens if Anchor Direction Estimation Data were the Same as the Training Data

Suppose that the anchor direction estimation data $\{(x_i^{\mathrm{A}}, y_i^{\mathrm{A}}, \delta_i^{\mathrm{A}})\}_{i=1}^{n^{\mathrm{A}}}$ were actually the same as the training data $\{(x_i, y_i, \delta_i)\}_{i=1}^{n}$, so that $n^{\mathrm{A}} = n$ and $x_i^{\mathrm{A}} = x_i$ for $i = 1, \ldots, n$. Since the training data were used to learn ϕ, then ϕ itself depends on all the training data. Thus, the embedding vectors of the anchor direction estimation data would be $u_i^{\mathrm{A}} = \phi(x_i^{\mathrm{A}}) = \phi(x_i)$ for each $i = 1, \ldots, n$. This means that $u_1^{\mathrm{A}}, \ldots, u_{n^{\mathrm{A}}}^{\mathrm{A}}$ would no longer be guaranteed to be independent since they each depend on ϕ which in turn depends on all of the training data (and thus all of the anchor direction estimation data).

A.2. Comments on the Log-rank Test

The log-rank test, like other statistical hypothesis tests, is designed under certain assumptions, where checking these assumptions is important if one wants the resulting p-values computed to be statistically valid. One case these assumptions hold is if in addition to the survival analysis setup stated in Section 2.1, we further assume that:

(i) the conditional censoring time distribution $\mathbb{P}_{C|X}$ is independent of raw inputs and is thus equal to the marginal censoring time distribution $\mathbb{P}_C$;

(ii) the true underlying hazard function $h(t|x)$ satisfies the proportional hazards assumption (1).

To provide some intuition, condition (i) ensures that when comparing any two clusters using the log-rank test, the censoring patterns for the two clusters "look the same". To see why this is important, consider an extreme example where two clusters truly have the same underlying survival time distribution but their censoring patterns are so different that one of the clusters always has all its observations censored whereas the other has no observations censored. In this case, we would not be able to tell that these two clusters have the same survival time distribution.

The justification for condition (ii) is more technical. One observation is that the log-rank test for comparing two clusters can be shown to be equivalent to a statistical test for the Cox proportional hazards model (specifically the so-called "score test") that checks for association between the survival time and an indicator variable stating which of the two clusters a data point is in (Harrell, 2015, Section 20.4). For more theoretical justification, see the books by Fleming and Harrington (1991, Chapter 7) and Andersen et al. (1993, Chapter V).

A.3. P-value Thresholding to Control for a Desired False Discovery Rate

For a specific anchor direction μ, we rank raw features by computing p-values of a statistical test (such as the chi-squared test of independence) that quantifies the strength of the association between each raw feature and projections along μ. These tests are not independent of each other because all tests use the same projection values along the same direction μ. To

determine a p-value threshold that appropriately controls for false discovery rate across multiple statistical tests with arbitrary dependence between them, one could use, for instance, the method by Benjamini and Yekutieli (2001).

Appendix B. SUPPORT Dataset Experiment

B.1. Hyperparameter Grid and Optimization Details

We train the neural network using minibatch gradient descent with at most 100 epochs and early stopping (no improvement in the validation concordance index after 10 epochs). We specifically use the Adam optimizer (Kingma and Ba, 2014). We sweep over the following hyperparameters:

- Batch size: 64, 128
- Learning rate: 0.01, 0.001
- Number of fully-connected layers in encoder ϕ: 1, 2, 3, 4
- Embedding dimension d: 5, 6, 7, 8, 9, 10

Our code is written using PyTorch (Paszke et al., 2019).

Compute instance. We ran our code on a Ubuntu 22.04.1 LTS machine with an Intel Core i9-10900K CPU (3.7GHz, 10 cores, 20 threads) with 64GB RAM and a Quadro RTX 4000 GPU (with 8GB GPU RAM).

B.2. Additional Visualizations Using an Encoder With the Euclidean Norm 1 Constraint

In the main paper, we only showed visualizations for the first cluster found out of the 5 clusters used in the 5-component mixture of von Mises-Fisher distributions. We include scatter plots of age vs anchor projections for all 5 clusters in Figure B.1, raw feature probability heatmaps for all 5 clusters in Figure B.2, raw feature rankings for all 5 clusters in Table B.1, and survival probability heatmaps for all 5 clusters in Figure B.3.

Interpreting the visualizations. From the visualizations, we can see some patterns, where for simplicity we mention only a few per cluster (e.g., looking at the top few features per cluster in Table B.1 already provides insight); we list these clusters in order of how fast their survival probability heatmap's rightmost column decays (starting from the fastest decay, indicative of survival times that tend to be the shortest):

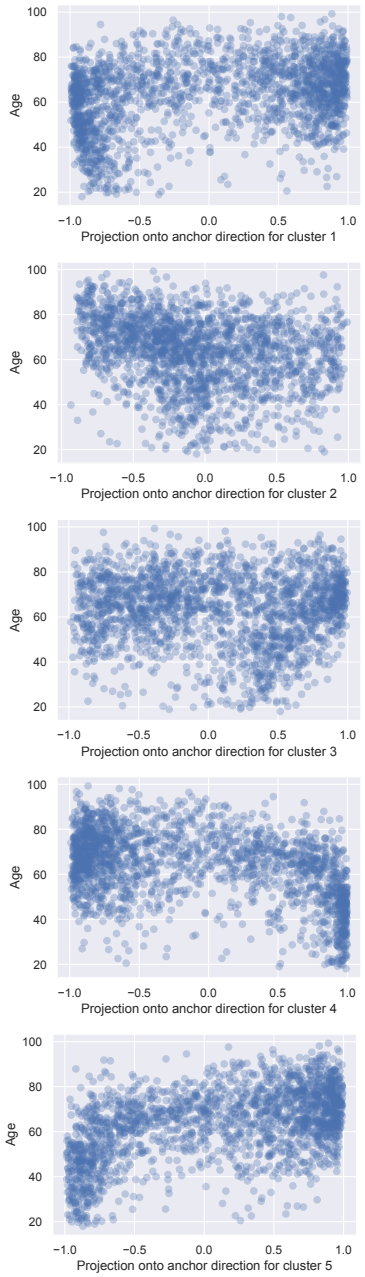

Figure B.1: SUPPORT dataset: scatter plots of age vs projection values for a DeepSurv model where the encoder has a Euclidean norm 1 constraint. The projection values are along each cluster's anchor direction (for all 5 clusters in a 5-component mixture of von Mises-Fisher distributions). The plot only for cluster 1 is in Figure 2.

- (Fastest survival probability decay) Cluster 1 is, as already stated in Sections 3.4.2 and 3.4.4, associated with patients being more elderly and having metastatic cancer and at least one comorbidity.
- Cluster 5 is associated with patients who are elderly, have low or normal temperatures, and often have cancer (non-metastatic or metatstatic). Similar to cluster 1, cluster 5 is largely also associated with patients having at least one comorbidity.
- Cluster 2 is associated with patients having high temperatures (indicative of a fever), high sodium levels, and lower ages.
- Cluster 3 is associated with patients without cancer, with low or normal temperatures, and ages that are neither low nor high.
- (Slowest survival probability decay) Cluster 4 is associated with patients who are young, do not have cancer, and (compared to patients with high projection values for the other clusters) often do not have any comorbidities.

These interpretations are not surprising in that being elderly, having cancer, and having at least one comorbidity intuitively should be associated with a patient being more ill and tending to have shorter survival times. Similar findings for the same dataset but using a different neural survival analysis model have been reported previously by Chen (2022). Note that the above ranking of clusters was determined qualitatively by looking at the survival probability heatmaps. In fact, an approach we suggest for ranking clusters/anchor directions by median survival time estimates in Appendix G yields the same ranking.

Using different numbers of clusters and a different clustering algorithm. We have also separately tried using different numbers of clusters (aside from $k = 5$) with the mixture of von Mises-Fisher distributions. As the resulting visualizations do not convey much more insight than what we have already presented, we defer these to our code repository. Qualitatively, we found the following: using $k < 5$ results in "coarser-grain" clusters, each of which look like a combination of the clusters we found with $k = 5$, and using $k > 5$ results in "finer-grain" clusters although some of these finer-grain clusters have raw feature probability (and, separately, survival probability heatmaps) that look very similar (so that some of these clusters should probably be merged into a single cluster as they correspond to similar raw feature patterns and survival time distributions).

We also repeated this exercise of trying different numbers of clusters where we cluster using a Gaussian mixture model instead. Again, we defer the resulting visualizations to our code repository except for the violin plot, which we show in Figure B.4. Qualitatively, the clusters found for the same choice of k was somewhat similar to what we get using a mixture of von Mises-Fisher distributions, although of course the clusters are not entirely the same. The violin plot ends up looking a bit different: we see in Figure B.4 that the p-values tend to be very small for $k = 2$ and $k = 3$, and then they increase a bit at $k = 4$, and even more at $k = 5$ (the highest point in the violin plot significantly increases from $k = 4$ to $k = 5$ and then it stays very high for all values of $k > 5$ that we tried).

Ultimately, we have left the choice of which clustering algorithm to use up to the user. We suspect that a "good" choice of clustering algorithm would be able to find a larger number of clusters while keeping the p-values low in the violin plot. For instance, using a mixture of von Mises-Fisher distributions, the violin plot has very low p-values for up to $k = 5$ in Figure 1. In contrast, the violin plot we get using a Gaussian mixture model for the same embedding vectors has low p-values only up to $k = 3$ as shown in Figure B.4. This suggests that the clusters found for the Gaussian mixture model do not distinguish the embedding vectors as well in terms of survival outcomes compared to the mixture of von Mises-Fisher distributions.

B.3. Visualizations Using an Encoder *Without* the Euclidean Norm 1 Constraint

We now present results using the exact same setup as described in Section 3.4 and detailed in Appendices B.1 and B.2, where the only differences are that: (i) the final nonlinear activation layer in the encoder ϕ is ReLU instead of dividing the intermediate representation by its Euclidean norm, and (ii) the clustering model used in the embedding space is a Gaussian mixture model. For reference, this model achieves a test set concordance index of 0.615, which is close to what was achieved with the model that includes the Euclidean norm 1 constraint.

The violin plot for selecting the number of clusters is shown in Figure B.5, where we choose the number of clusters to be $k = 3$. For this 3-component Gaussian mixture model, we find the anchor directions corresponding to its 3 clusters and then show scatter plots of age vs anchor projections of all 3 clusters in Figure B.6, raw feature probability heatmaps for all 3 clusters in Figure B.7, raw feature rankings in Table B.2, and survival probability heatmaps for all 3 clusters in Figure B.8.

Figure B.2: SUPPORT dataset: raw feature probability heatmaps for a DeepSurv model where the encoder has a Euclidean norm 1 constraint. These heatmaps are for all 5 clusters' anchor directions (clusters are from a 5-component mixture of von Mises-Fisher distributions). The heatmap only for cluster 1 is also shown in Figure 3.

Table B.1: SUPPORT dataset: rankings of raw features based on the p-value of Pearson's chi-squared test for a DeepSurv model where the encocder has a Euclidean norm 1 constraint. These tables are for all 5 clusters in a 5-component mixture of von Mises-Fisher distributions. The ranking table only for cluster 1 is also in Table 1.

Cluster 1		
Rank	Feature	p-value
1	cancer	2.86×10^{-225}
2	age	1.04×10^{-45}
3	number of comorbidities	1.91×10^{-24}
4	mean arterial blood pressure	9.25×10^{-19}
5	diabetes	2.46×10^{-14}
6	dementia	4.15×10^{-11}
7	temperature	1.58×10^{-10}
8	heart rate	3.34×10^{-9}
9	female	1.07×10^{-6}
10	serum creatinine	1.21×10^{-6}
11	race	3.42×10^{-5}
12	white blood count	7.02×10^{-5}
13	respiration rate	4.54×10^{-4}
14	serum sodium	6.14×10^{-2}

Cluster 2		
Rank	Feature	p-value
1	temperature	6.73×10^{-181}
2	age	2.42×10^{-40}
3	serum sodium	4.06×10^{-37}
4	female	6.42×10^{-32}
5	cancer	2.09×10^{-27}
6	mean arterial blood pressure	3.01×10^{-18}
7	respiration rate	2.43×10^{-12}
8	number of comorbidities	3.80×10^{-8}
9	heart rate	4.04×10^{-8}
10	white blood count	2.87×10^{-5}
11	serum creatinine	2.89×10^{-3}
12	race	1.78×10^{-2}
13	diabetes	3.56×10^{-2}
14	dementia	6.41×10^{-1}

Cluster 3		
Rank	Feature	p-value
1	cancer	6.26×10^{-183}
2	temperature	2.93×10^{-75}
3	age	1.75×10^{-37}
4	female	1.35×10^{-24}
5	number of comorbidities	3.05×10^{-20}
6	white blood count	6.74×10^{-16}
7	heart rate	1.58×10^{-12}
8	respiration rate	1.04×10^{-10}
9	mean arterial blood pressure	1.28×10^{-9}
10	serum sodium	1.39×10^{-9}
11	serum creatinine	1.19×10^{-8}
12	race	2.44×10^{-7}
13	dementia	6.06×10^{-5}
14	diabetes	6.16×10^{-3}

Cluster 4		
Rank	Feature	p-value
1	cancer	6.12×10^{-185}
2	age	1.48×10^{-99}
3	number of comorbidities	1.09×10^{-18}
4	mean arterial blood pressure	4.24×10^{-15}
5	dementia	1.29×10^{-13}
6	diabetes	1.71×10^{-8}
7	temperature	1.85×10^{-6}
8	serum sodium	1.37×10^{-4}
9	serum creatinine	1.65×10^{-4}
10	respiration rate	1.69×10^{-4}
11	race	4.96×10^{-4}
12	female	7.54×10^{-4}
13	white blood count	4.61×10^{-3}
14	heart rate	1.53×10^{-2}

Cluster 5		
Rank	Feature	p-value
1	age	4.47×10^{-130}
2	cancer	4.88×10^{-114}
3	temperature	1.09×10^{-42}
4	serum sodium	2.35×10^{-17}
5	mean arterial blood pressure	6.81×10^{-17}
6	number of comorbidities	9.90×10^{-14}
7	dementia	1.66×10^{-13}
8	race	6.59×10^{-4}
9	heart rate	2.06×10^{-3}
10	respiration rate	2.05×10^{-2}
11	serum creatinine	7.25×10^{-2}
12	diabetes	8.99×10^{-2}
13	white blood count	2.60×10^{-1}
14	female	3.98×10^{-1}

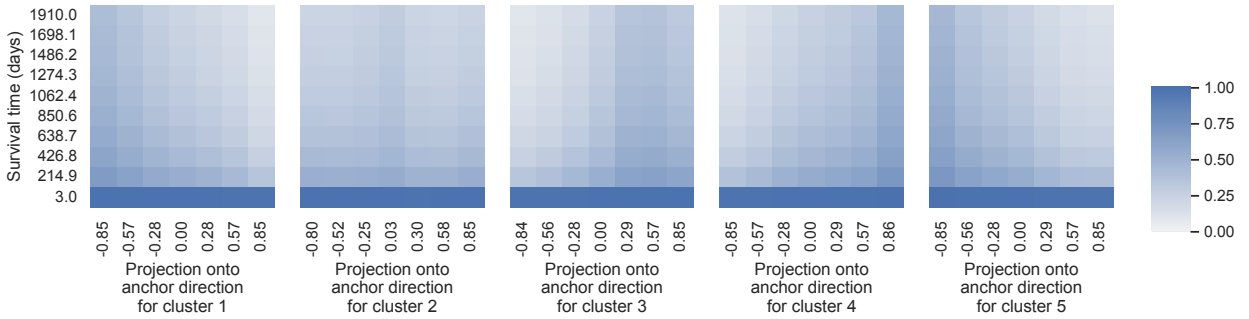

Figure B.3: SUPPORT dataset: survival probability heatmaps for a DeepSurv model where the encoder has a Euclidean norm 1 constraint. These heatmaps are for all 5 clusters' anchor directions (clusters are from a 5-component mixture of von Mises-Fisher distributions). The heatmap only for cluster 1 is also in Figure 4.

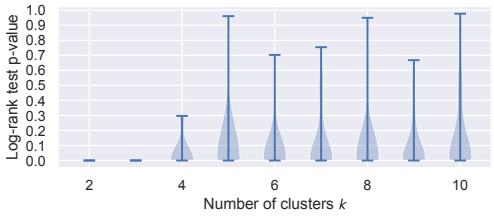

Figure B.4: SUPPORT dataset: a violin plot to help select the number of clusters (and thus the number of anchor directions) to use with a clustering model. Here, the encoder used is from a DeepSurv model that has a Euclidean norm 1 constraint, and the clustering model is a Gaussian mixture model.

A key point we want to emphasize is that in the scatter plots (Figure B.6), we can see the "clumping up" artifact we mentioned in Section 3.6 that indicates that there is likely a lot of magnitude information lost (specifically, a lot of the points in the scatter plot "clump up" around -1 an 1).

In this case, the top features across clusters are largely the same (Table B.2). From looking at the raw feature probability heatmaps (Figure B.7), focusing on the largest projection bin per heatmap, we find that clusters 1 and 2 actually appear quite similar: both appear to be associated with older patients who often have cancer and at least one comorbidity. In contrast, cluster 3 appears to be associated with younger patients without cancer. Meanwhile, the survival probability heatmap (Figure B.8) indicates that indeed clusters 1 and 2 are associated with survival functions that decay quickly (indicative of the survival times tending to be shorter) whereas cluster 3 is associated with a survival function that decays slowly. These are the same main findings as for the version of the model that used the Euclidean norm 1 constraint. In fact, even if we used 2 clusters, we get the same main findings.

We point out that the major challenge when there is a lot of information loss due to magnitudes being ignored is that our visualization heatmaps will end up each consisting of essentially only two projection bins along the x-axis (that have enough data in them: one that contains projection values around -1 and the other that contains projection values around 1). In this case, we could still of course find interesting relationships of how a raw feature changes with respect to an anchor direction but we would only be seeing what this change looks like (if there is any) at two x-axis values. We would only be able to check for monotonic trends between how a raw feature relates to two projection values along a specific anchor direction.

Appendix C. Rotterdam/GBSG Experiment

As mentioned in this main paper, we also have visualizations where we train on the Rotterdam dataset (using the exact same neural network architecture as we used for SUPPORT, including the Euclidean norm 1 constraint; we specifically use the same hyperparameter grid and optimization strategy as detailed in Appendix B.1). For reference, when we test on the GBSG dataset, we get a concordance index of 0.677. For the visualizations to follow, we randomly choose 25% of the GBSG dataset to treat as the anchor direction estimation data, and we use the feature vectors from the rest of the data as the visualization raw inputs $x_1^{\mathrm{V}}, \ldots, x_{n_{\mathrm{V}}}^{\mathrm{V}}$. Note that technically how we have set up the DeepSurv model here violates

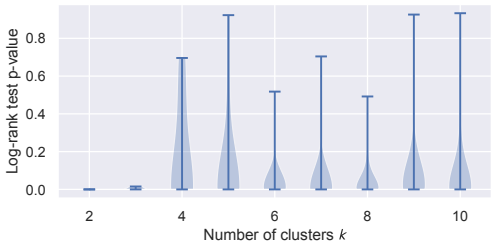

Figure B.5: SUPPORT dataset: a violin plot to help select the number of clusters (and thus the number of anchor directions) to use with a clustering model. Here, the encoder used is from a DeepSurv model that does *not* have a Euclidean norm 1 constraint, and the clustering model is a Gaussian mixture model.

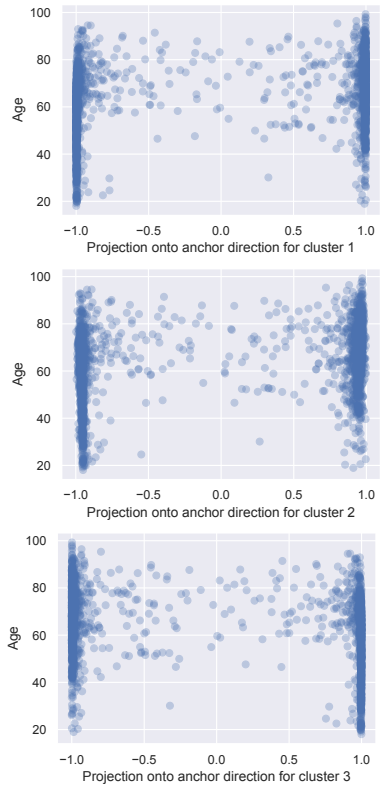

Figure B.6: SUPPORT dataset: scatter plots of age vs projection values for a DeepSurv model where the encoder does *not* have a Euclidean norm 1 constraint. The projection values are along each cluster's anchor direction (for all 3 clusters in a 3-component Gaussian mixture model).

Table B.2: SUPPORT dataset: rankings of raw features based on the p-value of Pearson's chi-squared test for a DeepSurv model where the encoder does *not* have a Euclidean norm 1 constraint. These tables are for all 3 clusters in a 3-component Gaussian mixture model.

Cluster 1		
Rank	Feature	p-value
1	cancer	4.35×10^{-176}
2	age	2.90×10^{-34}
3	dementia	9.15×10^{-15}
4	number of comorbidities	4.76×10^{-14}
5	heart rate	6.93×10^{-9}
6	mean arterial blood pressure	2.48×10^{-8}
7	diabetes	3.43×10^{-5}
8	female	4.77×10^{-5}
9	white blood count	5.63×10^{-4}
10	temperature	5.94×10^{-3}
11	respiration rate	1.31×10^{-1}
12	serum creatinine	3.13×10^{-1}
13	serum sodium	3.32×10^{-1}
14	race	5.28×10^{-1}

Cluster 2		
Rank	Feature	p-value
1	cancer	3.41×10^{-175}
2	age	8.46×10^{-36}
3	dementia	6.14×10^{-15}
4	number of comorbidities	1.38×10^{-12}
5	heart rate	7.64×10^{-10}
6	mean arterial blood pressure	1.98×10^{-8}
7	diabetes	6.49×10^{-5}
8	female	9.97×10^{-4}
9	temperature	8.66×10^{-3}
10	white blood count	1.71×10^{-2}
11	serum sodium	2.14×10^{-2}
12	race	2.45×10^{-1}
13	serum creatinine	2.45×10^{-1}
14	respiration rate	4.42×10^{-1}

Cluster 3		
Rank	Feature	p-value
1	cancer	1.34×10^{-175}
2	age	4.95×10^{-34}
3	dementia	9.15×10^{-15}
4	number of comorbidities	3.29×10^{-14}
5	heart rate	1.48×10^{-8}
6	mean arterial blood pressure	8.29×10^{-8}
7	diabetes	6.40×10^{-6}
8	female	1.25×10^{-4}
9	white blood count	1.49×10^{-3}
10	temperature	1.31×10^{-2}
11	respiration rate	1.59×10^{-1}
12	serum creatinine	1.60×10^{-1}
13	race	2.18×10^{-1}
14	serum sodium	3.04×10^{-1}

Figure B.7: SUPPORT dataset: raw feature probability heatmaps for a DeepSurv model where the encoder does *not* have a Euclidean norm 1 constraint. These heatmaps are for each cluster's anchor direction (for all 3 clusters in a 3-component Gaussian mixture model).

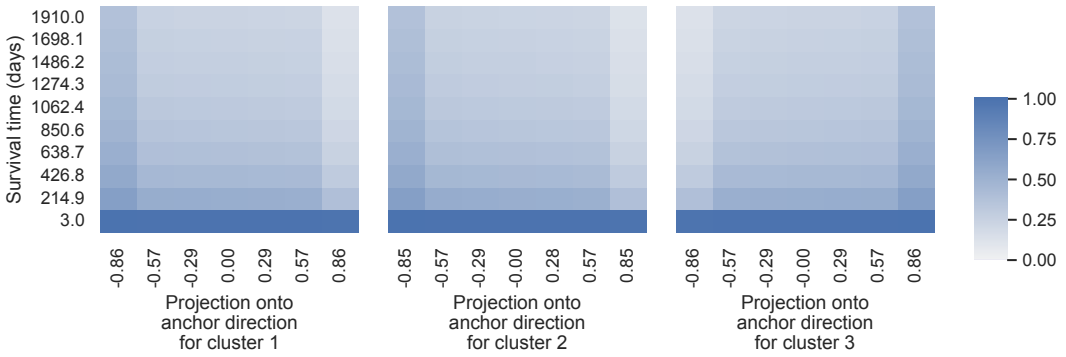

Figure B.8: SUPPORT dataset: survival probability heatmaps for a DeepSurv model where the encoder does *not* have a Euclidean norm 1 constraint. These heatmaps are for each cluster's anchor direction (for all 3 clusters in a 3-component Gaussian mixture model).

Table C.1: Rotterdam/GBSG datasets: rankings of raw features based on the p-value of Pearson's chi-squared test for a DeepSurv model where the encoder has a Euclidean norm 1 constraint. These tables are for all 3 clusters from a 3-component mixture of von Mises-Fisher distributions.

Cluster 1		
Rank	Feature	p-value
1	age	6.22×10^{-50}
2	postmenopausal	1.36×10^{-40}
3	number of positive nodes	3.51×10^{-27}
4	estrogen receptor	2.67×10^{-14}
5	progesterone receptor	1.99×10^{-7}
6	tumor size	4.11×10^{-5}
7	hormonal therapy	8.39×10^{-3}

Cluster 2		
Rank	Feature	p-value
1	number of positive nodes	1.58×10^{-98}
2	hormonal therapy	4.32×10^{-14}
3	tumor size	2.52×10^{-9}
4	progesterone receptor	6.66×10^{-6}
5	age	1.47×10^{-5}
6	estrogen receptor	5.02×10^{-4}
7	postmenopausal	6.31×10^{-2}

Cluster 3		
Rank	Feature	p-value
1	age	1.07×10^{-35}
2	postmenopausal	5.56×10^{-33}
3	number of positive nodes	2.61×10^{-30}
4	hormonal therapy	2.47×10^{-10}
5	estrogen receptor	2.05×10^{-3}
6	tumor size	2.61×10^{-1}
7	progesterone receptor	2.62×10^{-1}

the i.i.d. assumption between training and anchor direction estimation data, as well as the assumption that the visualization raw inputs come from the same distribution as the raw inputs of the training data. However, our visualization framework still actually works when the training data are sampled differently. We discuss this in a bit more detail next before going over the resulting visualizations.

A different distribution for training data. In Section 3, when we discussed statistical assumptions and sample splitting, we had, for simplicity, assumed that the training data and anchor direction estimation data were sampled i.i.d. from the distribution (which technically is a joint distribution $\mathbb{P}_{X,Y,\Delta}$ defined for raw input X with observed time Y and event indicator Δ; here, X, Y, and Δ are random variables), and that the visualization raw inputs are drawn from the marginal raw input distribution $\mathbb{P}_X$. In fact, the training data could be sampled differently so long as the anchor direction estimation data and visualization raw inputs are sampled in a manner that is independent of the training data, which ensures that we do not encounter the issue stated in Appendix A.1.

To be more precise, our visualization framework still holds if the anchor direction estimation data are sampled i.i.d. from $\mathbb{P}_{X,Y,\Delta}$ and the visualization raw inputs are sampled from $\mathbb{P}_X$, but now the training data are sampled i.i.d. from some other distribution $\mathbb{Q}_{X,Y,\Delta}$ and the training data are independent from the anchor direction estimation data and the visualization raw inputs. After all, the statistical analyses we conduct are all conditioned on the training data and the encoder ϕ; we just needed to ensure that conditioning on the training data and ϕ did not result in dependence between anchor direction estimation data or the visualization raw inputs.

Visualization results and interpretations. We show the violin plot for selecting the number of clusters for a mixture of von Mises-Fisher distributions in Figure C.1(a), where we choose the number of clusters to be $k = 3$. For this 3-component mixture model, we find the anchor directions corresponding to its 3 clusters and then show raw feature probability heatmaps for all 3 clusters in Figure C.1(b), raw feature rankings in Table C.1, and survival probability heatmaps for all 3 clusters in Figure C.1(c).

The main findings from our visualizations are as follows: clusters 1 and 3 are both associated with patients who tend to have very few lymph nodes that contain cancer, which in turn is associated with longer survival times (the survival probability functions in the rightmost of the survival probability heatmaps for clusters 1 and 3 decay slowly). Interestingly, clusters 1 and 3 differ largely in that cluster 1 is for older women (being postmenopausal is flagged as being very probable for cluster 1 which is of course related to age) and cluster 3 is for younger women (who have not undergone menopause). Meanwhile, the anchor direction for cluster 2 is associated with women who have a large number of lymph nodes that have cancer and the tumor sizes tend to be large as well. This of course means that cancer has advanced quite significantly and, unsurprisingly, cluster 2 is associated with shorter survival times (its survival probability heatmap's rightmost column decays quickly).

Appendix D. Tabular Data: Finding Interactions Between Raw Features in Predicting Projection Values Along an Anchor Direction

We mention two methods that can be used to probe raw feature interactions in predicting projection values along an anchor direction μ.

Fitting a regression model that is "easy to interpret" and accounts for feature interactions. One way to find possible interactions is to fit any regression model that is straightforward to interpret and that can surface possible feature interactions, using feature vectors $x_1^{\mathrm{V}}, \ldots, x_{n^{\mathrm{V}}}^{\mathrm{V}}$ and target regression labels $p_1^{\mathrm{V}}, \ldots, p_{n^{\mathrm{V}}}^{\mathrm{V}}$. For example, we could fit a so-called *optimal regression tree* using mixed-integer optimization (Dunn, 2018). An example of such a tree is shown in Figure D.1. The reason why this tree encodes feature interaction information is apparent when we look at any leaf. For example, the leftmost leaf with predicted projection value -0.6306 corresponds to the intersection of the constraints "cancer = no", "age $<$ 68.44", and "serum creatinine $<$ 2.55", showing an interaction between the variables "cancer", "age", and "serum creatinine" that depends on them satisfying specific inequalities.

Note that this sort of approach should be used with care. Specifically, by using different splits of data (e.g., different train/validation splits) to train the tree, it is possible that the resulting trees look different and might suggest different raw features to matter, and the raw features that interact might vary across experimental repeats. Another issue is that such tree

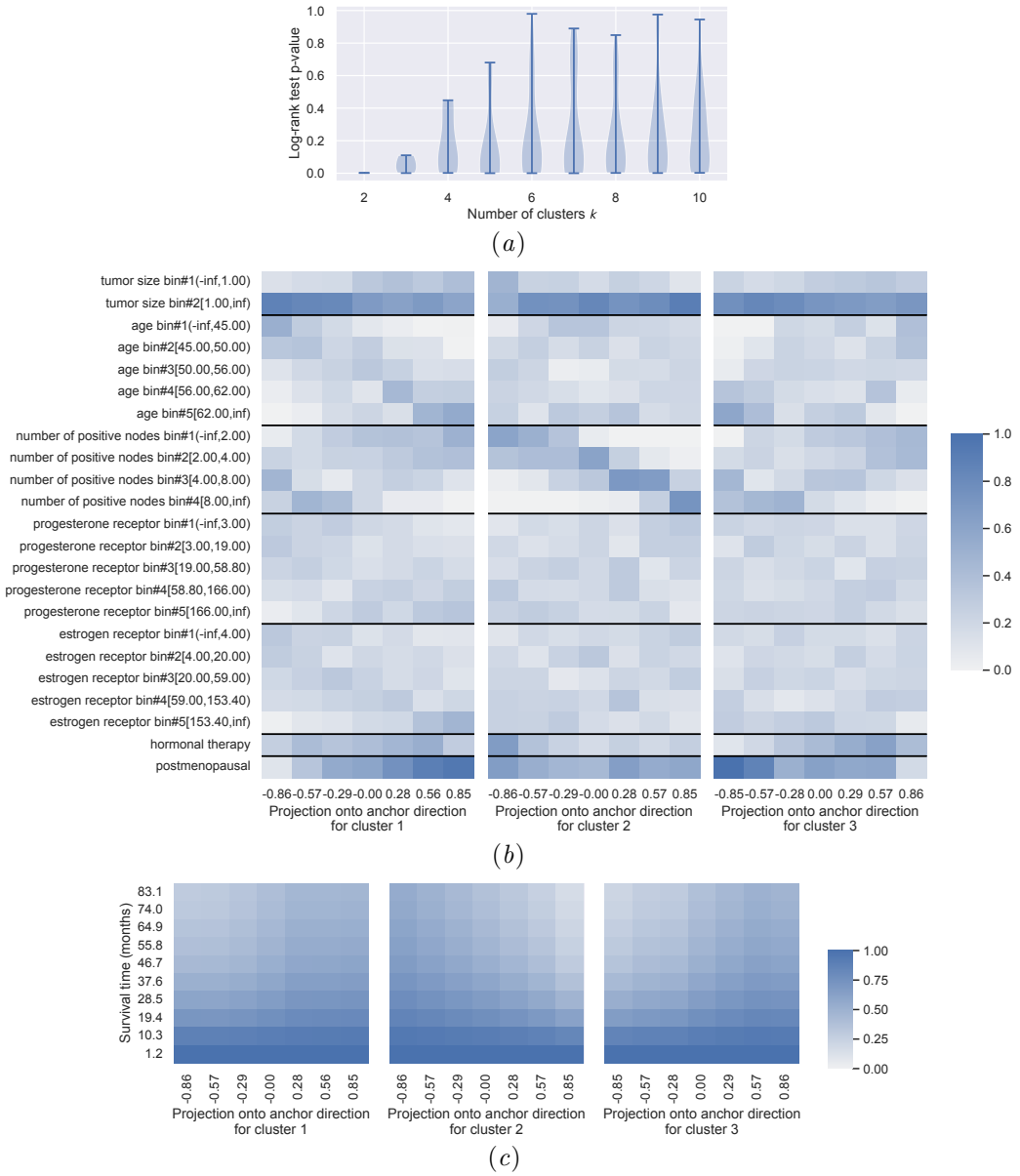

Figure C.1: Rotterdam/GBSG datasets: visualizations for a DeepSurv model where the encoder has a Euclidean norm 1 constraint. Panel (a) shows a violin plot to help select the number of clusters (and thus the number of anchor directions) for use with a mixture of von Mises-Fisher distributions, where we choose $k = 3$ for the subsequent panels. Panel (b) shows raw feature probability heatmaps for the three clusters' estimated anchor directions. Panel (c) shows the survival probability heatmaps for the same anchor directions as panel (b). Details on the dataset, encoder, and clustering model are in Appendix C.

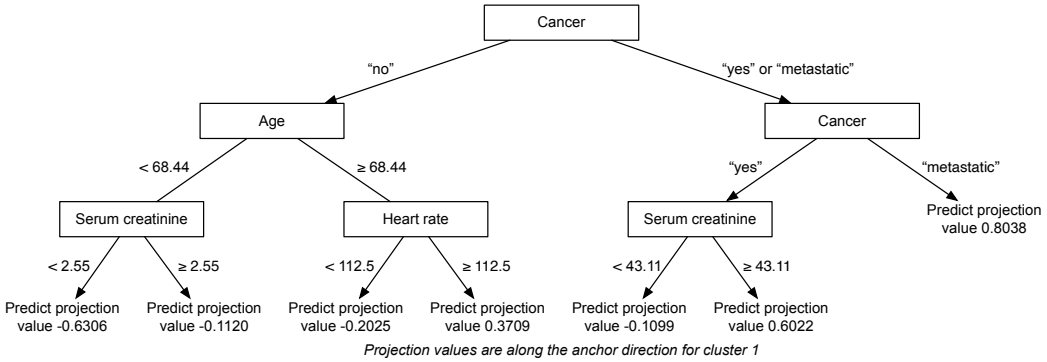

Figure D.1: SUPPORT dataset: using a DeepSurv model where the encoder has a Euclidean norm 1 constraint, we show an optimal regression tree trained using visualization raw inputs (feature vectors) to predict projection values (regression target labels) along cluster 1's anchor direction. Cluster 1 is the same cluster that we provided visualizations for throughout Section 3.4 and in Appendix B.2.

learning algorithms have hyperparameter(s) for controlling the tree complexity, such as the max tree depth, and for different such hyperparameter choices, the raw features that are found to be important or that are shown to interact might vary.

Archipelogo. As an alternative approach to finding possible feature interactions, we point out that one could use the Archipelogo framework (Tsang et al., 2020) that is designed to find feature interactions of a black-box model (in this case, the encoder ϕ) when provided with specific raw inputs (e.g., $x_1^{\mathrm{V}}, \ldots, x_{n^{\mathrm{V}}}^{\mathrm{V}}$).

Appendix E. Survival MNIST Experiment

E.1. Dataset

The Survival MNIST dataset builds off of the original MNIST classification dataset (LeCun et al., 2010), which consists of 60,000 training images and 10,000 test images. All images are 28-by-28 pixel grayscale images of handwritten digits. Each image has a target label corresponding to which of the 10 digits the image corresponds to. Pölsterl (2019) modified the MNIST dataset so that the labels for training and test images are instead survival labels (observed times and event indicators) that are synthetically generated. There is some flexibility in this synthetic generation process. We specifically use the same synthetic survival label generation procedure as Goldstein et al. (2020), who also use the Survival MNIST dataset. Specifically, for each digit $j \in \{0, 1, \ldots, 9\}$, we let m_j denote the true mean survival time for digit j, where:

- $m_0 = 11.25$
- $m_1 = 2.25$
- $m_2 = 5.25$
- $m_3 = 5.0$
- $m_4 = 4.75$
- $m_5 = 8.0$
- $m_6 = 2.0$
- $m_7 = 11.0$
- $m_8 = 1.75$
- $m_9 = 10.75$

These mean survival times are also shown in Figure 5. The digits are ranked (in increasing order of mean survival time) as: 8, 6, 1, 4, 3, 2, 5, 9, 7, 0. Note that the 10 digits are grouped into four ground truth risk groups: $\{0, 7, 9\}, \{1, 6, 8\}, \{2, 3, 4\}, \{5\}$; within each risk group, the digits in the group have very similar ground truth mean survival times.

For each training image $x_i \in \mathcal{X}$ with digit label $\eta_i \in \{0, 1, \ldots, 9\}$, we sample its true survival time t_i from a Gamma distribution with mean m_{η_i} and variance 10^{-3}. After we generate survival times $t_1, \ldots, t_n$ for all training data, we then sample the censoring times $c_1, \ldots, c_n$ i.i.d. from a uniform distribution between $\min\{t_1, \ldots, t_n\}$ and the 90th percentile value of $t_1, \ldots, t_n$. Finally, we set the training data's observed times and event indicators to be $y_i = \min\{t_i, c_i\}$ and $\delta_i = \mathbb{1}\{t_i \leq c_i\}$ respectively. This results in a censoring rate $(1 - \frac{1}{n}\sum_{i=1}^{n} \delta_i)$ of approximately 50%. The test data are generated separately from the training data but in the same manner.

As a reminder, when training a survival analysis model with the Survival MNIST dataset, the training data are $\{(x_i, y_i, \delta_i)\}_{i=1}^{n}$. The true digit labels $\eta_1, \ldots, \eta_n$ are not available to the training procedure. For test data, we also have access to their digit labels.

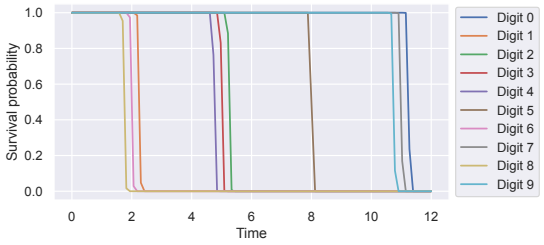

Figure E.1: Ground truth survival functions for the different digits.

Some important remarks regarding this dataset are in order:

- *The proportional hazards assumption does not hold for the underlying survival distributions of the different digits.* One way to see this is that by plotting the true survival functions of the different digits (per digit, we can get the true survival function by looking at 1 minus the CDF of the Gamma distribution associated with the digit), these survival functions are not powers of one another; we show these ground truth survival functions in Figure E.1. (When the proportional hazards assumption holds, all survival functions are powers of an underlying "baseline survival function".) This means that if we fit a model with a proportional hazards assumption, such as the DeepSurv model, we should not expect the predicted conditional survival functions to be correct although these should be able to "correctly order" the survival times of the different digits. Recall that the digits are ordered by true mean survival time (in increasing order) as 8, 6, 1, 4, 3, 2, 5, 9, 7, 0. We would like the "average predicted survival function for digit 8" to be lower than that of digit 6, which should be lower than that of digit 1, and so forth.
- *By how the censoring mechanism is set up, digits with higher mean survival times have higher censoring rates.* This is because the uniform distribution for censoring times has its maximum set to be the 90th percentile value of randomly generated survival times, so digits with high ground truth mean survival times get censored more often. For example, for the test data we generated, we have the following censoring rates for the different digits (we have ordered the digits in increasing order of ground truth mean survival time):
  - Digit 8: 1.23%
  - Digit 6: 2.92%
  - Digit 1: 5.64%
  - Digit 4: 34.42%
  - Digit 3: 35.25%
  - Digit 2: 37.98%
  - Digit 5: 71.19%
  - Digit 9: 96.53%
  - Digit 7: 99.22%
  - Digit 0: 100.00%

  Digits 0, 7, and 9 have censoring rates higher than 96%, with digit 0's censoring rate at 100% (in the training and test sets we generated, digit 0 is always censored). This means that that their randomly generated observed times and event indicators often look identical. Thus, when learning a neural survival analysis model using the training data, the learned embedding space (that we aim to visualize) would likely have trouble distinguishing between these three digits. Having part of the embedding space correspond only to digit 0 and not 7 or 9 would be particularly difficult.

E.2. Neural Survival Analysis Model and Encoder Setup

We set the base neural network f to be a convolutional neural network (CNN) consisting of the following sequence of layers:

- Conv2D layer with 32 filters (each 3-by-3)
- Nonlinear activation: ReLU
- MaxPool2D layer (2-by-2)
- Conv2D layer with 16 filters (each 3-by-3)
- Nonlinear activation: ReLU
- MaxPool2D layer (2-by-2)
- Flatten
- Fully-connected layer (that maps to d outputs)
- Nonlinear activation: Divide each vector by its Euclidean norm
- Fully-connected layer (map d inputs to 1 output)

We take the encoder ϕ to be everything excluding the last fully-connected layer.

Note that the above choice of CNN is somewhat arbitrary. The goal of our paper here is not to provide visualizations for the very best CNN possible for Survival MNIST. Rather, we just aim to show that for a choice of CNN that is straightforward to implement, we can readily provide visualizations for one of its intermediate representations.

Just as in the tabular data setup, we train the neural network using minibatch gradient descent with at most 100 epochs and early stopping (no improvement in the validation concordance index after 10 epochs). We use Adam to optimize, and we sweep over the following hyperparameters:

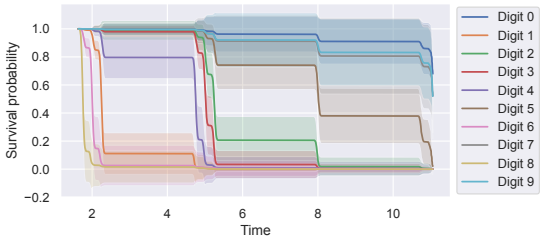

Figure E.2: Predicted survival functions for the different digits (mean ± standard deviation at each time point).

- Batch size: 64, 128
- Learning rate: 0.01, 0.001
- Embedding dimension d: 10, 20, 30, 40, 50

We ran the experiments for this dataset on the same compute instance mentioned in Appendix B.1. After training the DeepSurv model, the model achieved a test set concordance index of 0.953.

E.3. Additional Visualizations

E.3.1. Predicted Survival Functions

We begin with a visualization that is not actually related to our anchor direction visualization framework and instead just looks at how well the trained DeepSurv model predicts survival functions. For the i-th visualization raw input x_i^{V}, we denote its true digit label as $\eta_i^{\mathrm{V}} \in \{0, 1, \ldots, 9\}$. Then for digit j, we can compute the mean predicted survival function

$$\widehat{S}_{\text{digit } j}(t) \triangleq \frac{\sum_{i=1}^{n^{\mathrm{V}}} \mathbb{1}\{\eta_i^{\mathrm{V}} = j\} \widehat{S}(t|x_i^{\mathrm{V}})}{\sum_{i=1}^{n^{\mathrm{V}}} \mathbb{1}\{\eta_i^{\mathrm{V}} = j\}}.$$

We could also compute its standard deviation

$$\widehat{S}^{\text{std}}_{\text{digit } j}(t) \triangleq \sqrt{\frac{\sum_{i=1}^{n^{\mathrm{V}}} \mathbb{1}\{\eta_i^{\mathrm{V}} = j\} \big(\widehat{S}(t|x_i^{\mathrm{V}}) - \widehat{S}_{\text{digit } j}(t)\big)^2}{\sum_{i=1}^{n^{\mathrm{V}}} \mathbb{1}\{\eta_i^{\mathrm{V}} = j\}}}.$$

We plot each mean predicted survival function $\widehat{S}_{\text{digit } j}(t)$ with error bars given by $\widehat{S}^{\text{std}}_{\text{digit } j}(t)$ in Figure E.2. As expected, these survival functions do not resemble the ground truth ones as the neural survival analysis model fitted assumes a proportional hazards model. However, the ranking of the digits is approximately correct: looking at the mean predicted survival functions, the ranking of these (going from lower to higher) is: 8, 6, 1, 4, 3, 2, 5, 7, 9, 0. The only error in this ranking is that digits 7 and 9 are swapped. As a reminder, digits 0, 7, and 9 are more difficult as they have censoring rates over 96%.

Importantly, the survival functions $\widehat{S}_{\text{digit } j}$ are estimated with the help of ground truth digit labels. The DeepSurv model in this case never received ground truth digit labels and, in particular, its embedding space (the output space of encoder ϕ) was not explicitly trained to be able to distinguish between digits. That said, we can try to understand to what extent this embedding space captures information regarding the 10 digits. We proceed to do this next.

E.3.2. Treating Each Digit as a Concept for Anchor Direction Estimation

We now show random input vs projection plots (like the one in Figure 6) for all 10 digits in Figure E.3. From these plots, we see that as the projection value gets large for digit $j \in \{0, 1, \ldots, 9\}$, the raw inputs that achieve these large projection values for digit j tend to be of digit j itself or of other "adjacent" digit(s), where by "adjacent", we mean one(s) ordered next to digit j in terms of the ranking of ground truth mean survival times.

Using these same anchor directions, we produce the survival probability heatmaps (like the ones in Figure 4 and Figure B.3) in Figure E.4. Note that these heatmaps convey information similar to what is shown in Figure E.2. For instance, when we look at the rightmost column of the survival probability heatmap for digit 0's anchor direction, we see that the survival function barely decays for all the observed times, indicative of the survival time tending to be large, as expected. This rightmost column's survival function resembles the predicted survival curve for digit 0 in Figure E.2. A similar finding holds for the other digits.

The embedding space does not appear to capture the ground truth risk groups. We previously pointed out that the digits are grouped into four risk groups (each risk group has ground truth mean survival times that are very close by to each other; see Figure 5). It is not the case, however, that only the digits within the same risk group end up with high projection values for each other's anchor directions (see Figure 6). We *do* see this happen for the risk group with the lowest mean survival times (consisting of digits 1, 6, and 8) as well as the risk group with the highest mean survival times (consisting of 0, 7, and 9) but this does not entirely hold for the other risk groups.

To give a concrete example of how the embedding space does not correctly "capture" a risk group, con-

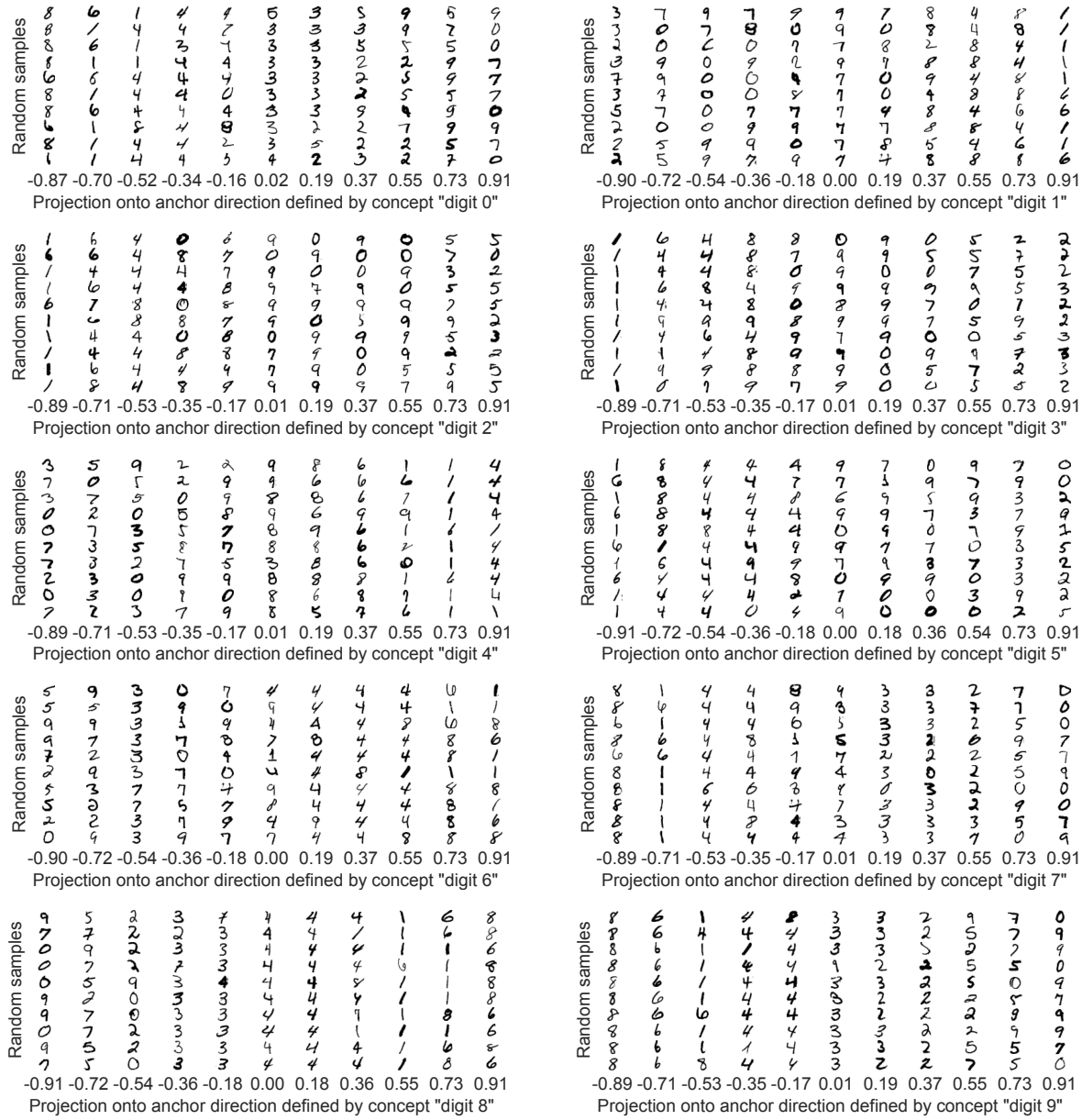

Figure E.3: Survival MNIST: we treat each of the 10 digits as a concept that we compute an anchor direction for, and then we produce random input vs projection plots for the 10 anchor directions. For each plot, per projection bin, we sample 10 random visualization raw inputs.

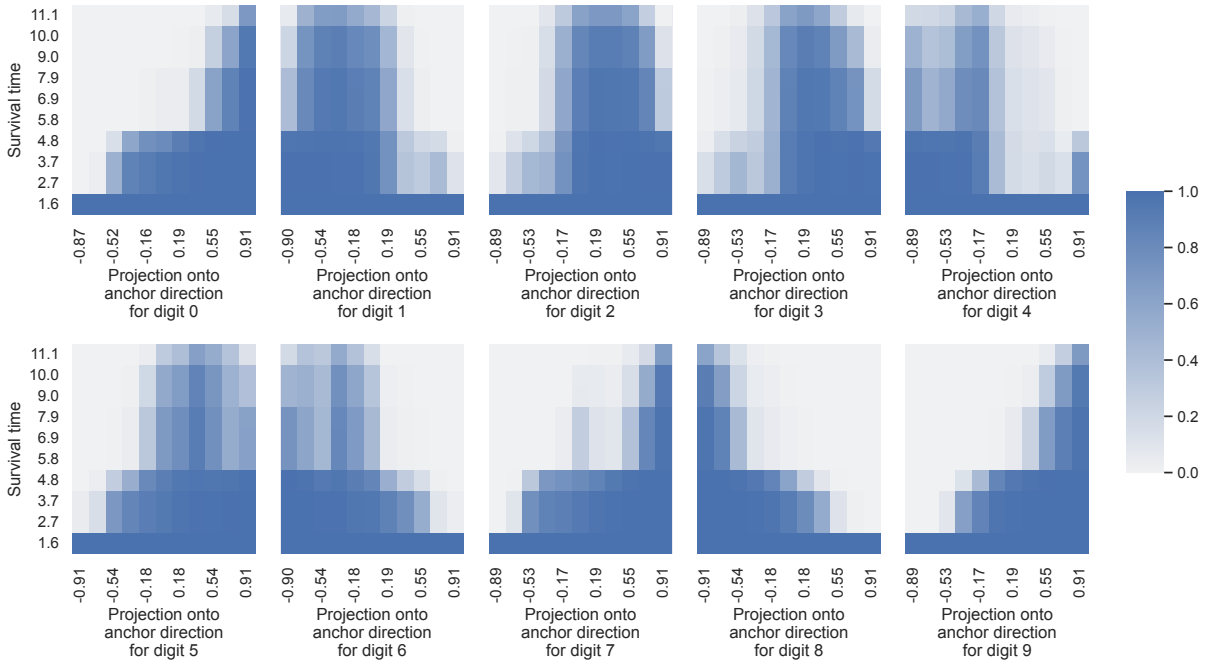

Figure E.4: Survival MNIST: we treat each of the 10 digits as a concept that we compute an anchor direction for, and then we plot survival probability heatmaps for the 10 anchor directions.

sider digit 5. The ground truth has digit 5 in its own risk group: no other digit's mean survival time is very close to that of digit 5. However, when we look at digit 5's random input vs projection plot in Figure E.3, when we look at the rightmost two projection bins, we see that many digits (that are not the digit 5) have high projection values for digit 5.

We suspect that what is causing the problem is censoring. We said that the digits in risk group $\{1, 6, 8\}$ tend to have high projection values for each other's anchor directions, and similarly for the digits in the risk group $\{0, 7, 9\}$. Note that all digits in risk group $\{1, 6, 8\}$ have censoring rates below 6%. All digits in risk group $\{0, 7, 9\}$ have censoring rates over 96%. In contrast, the digit 5 has a censoring rate of about 71%, which is neither very low nor very high. The digits with high projection values for digit 5 are ones that all have censoring rates over 35%. Basically, although the embedding space does not appear to be capturing risk groups well, it seems to recognize what censoring means. We examine this next.

E.3.3. Treating "Censored" as a Concept for Anchor Direction Estimation

We treat anchor direction estimation data that are censored (i.e., their event indicator variables are equal to 0) as a concept, which we then compute the anchor direction for using equation (6). We produce a random input vs projection plot for this "censored" concept's anchor direction in Figure E.5. From the plot, as we progress from the most negative projection values to the most positive, the random samples clearly correspond to digits going from the lowest to the highest ground truth mean survival times, which also corresponds to going from the lowest to the highest censoring rates.

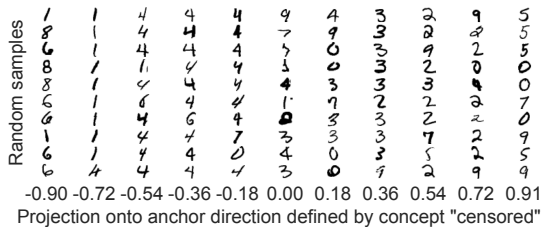

Figure E.5: Survival MNIST: after computing an anchor direction for the concept "censored", we produce a random input vs projection plot for this anchor direction.

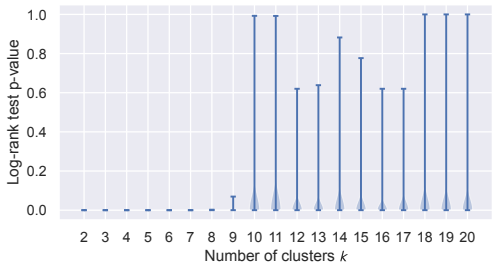

Figure E.6: Survival MNIST: a violin plot to help select the number of clusters for use with a mixture of von Mises-Fisher distributions (which determines the number of anchor directions to use).

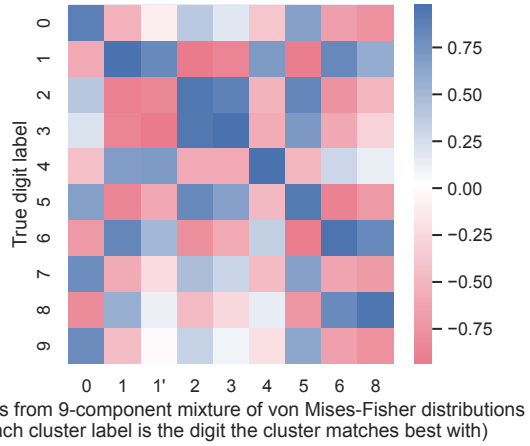

Figure E.7: Survival MNIST: average projection heatmap for seeing how well a clustering assignment with 9 clusters (using a mixture of von Mises-Fisher distributions) aligns with the 10 ground truth digit labels. The intensity at the i-th row and j-th column corresponds to average projection value along the j-th cluster's anchor direction across visualization data with ground truth digit label i. Note that the clusters with labels 1 and 1' both match best with digit 1.

E.3.4. Estimating Anchor Directions via Clustering

Lastly, we consider estimating anchor directions via clustering, where we use a mixture of von Mises-Fisher distributions as the clustering model. We show the violin plot for selecting the number of clusters in Figure E.6. From this violin plot, we see that the log-rank test p-values have a sharp increase after 9 clusters. We examine the clustering results using 9 clusters and, separately, also using 4 clusters and 10 clusters. The reason we look at the 4 clusters case is because there are 4 underlying risk groups, and we can check to what extent these can be recovered from a clustering solution with 4 clusters. As for looking at a model with 10 clusters, this is because we know that in reality there are 10 digits, each with its own ground truth survival function.

Results for a clustering model with 9 clusters. We first use anchor directions from a 9-component mixture of von Mises-Fisher distributions, where each component is treated as a cluster. We check how well the 9 clusters' anchor directions align with the anchor directions of the digit concepts. For visualization purposes, we label each cluster with the digit that the cluster matches best to, where we determine a match as follows: for the j-th cluster, we find whichever digit's anchor direction (computed using equation (6); these were the anchor directions used in Appendix E.3.2) is most similar to the j-th cluster's anchor direction (computed using equation (4)) according to cosine similarity. It is possible that multiple clusters match best with the same digit. Then, we can create what we call an *average projection heatmap* where the entry at the i-th row and j-th column corresponds to the average projection value along the j-th cluster's anchor direction across visualization data that have the ground truth digit label i. We show the resulting heatmap in Figure E.7. From this heatmap, we see that the cluster with label 0 has high projection values for digits 0, 5, 7, and 9, which are the most censored digits (digits 0, 7, and 9 in particular tend to have the highest projection values for cluster 0); digit 5, however, also has its own cluster that it is matched to whereas digits 7 and 9 do not. Another observation is that digits 1, 6, and 8 tend to have high projection values for clusters with labels 1, 6, and 8 (although the cluster with label 1 has highest projection values for digit 1, and similarly for the clusters with labels 6 and 8).

We create a random input vs projection plot for each of these 9 clusters' anchor directions in Figure E.8, where we see the same sort of phenomenon we had pointed out in Appendix E.3.2: when we look at an anchor direction roughly corresponding to digit j, then images of digit j as well as images of digits that have a true mean survival time adjacent to that of digit j will often have high projection values. We also plot survival probability heatmaps for these anchor directions in Figure E.9. Each cluster's survival probability heatmap is similar to the one for the digit

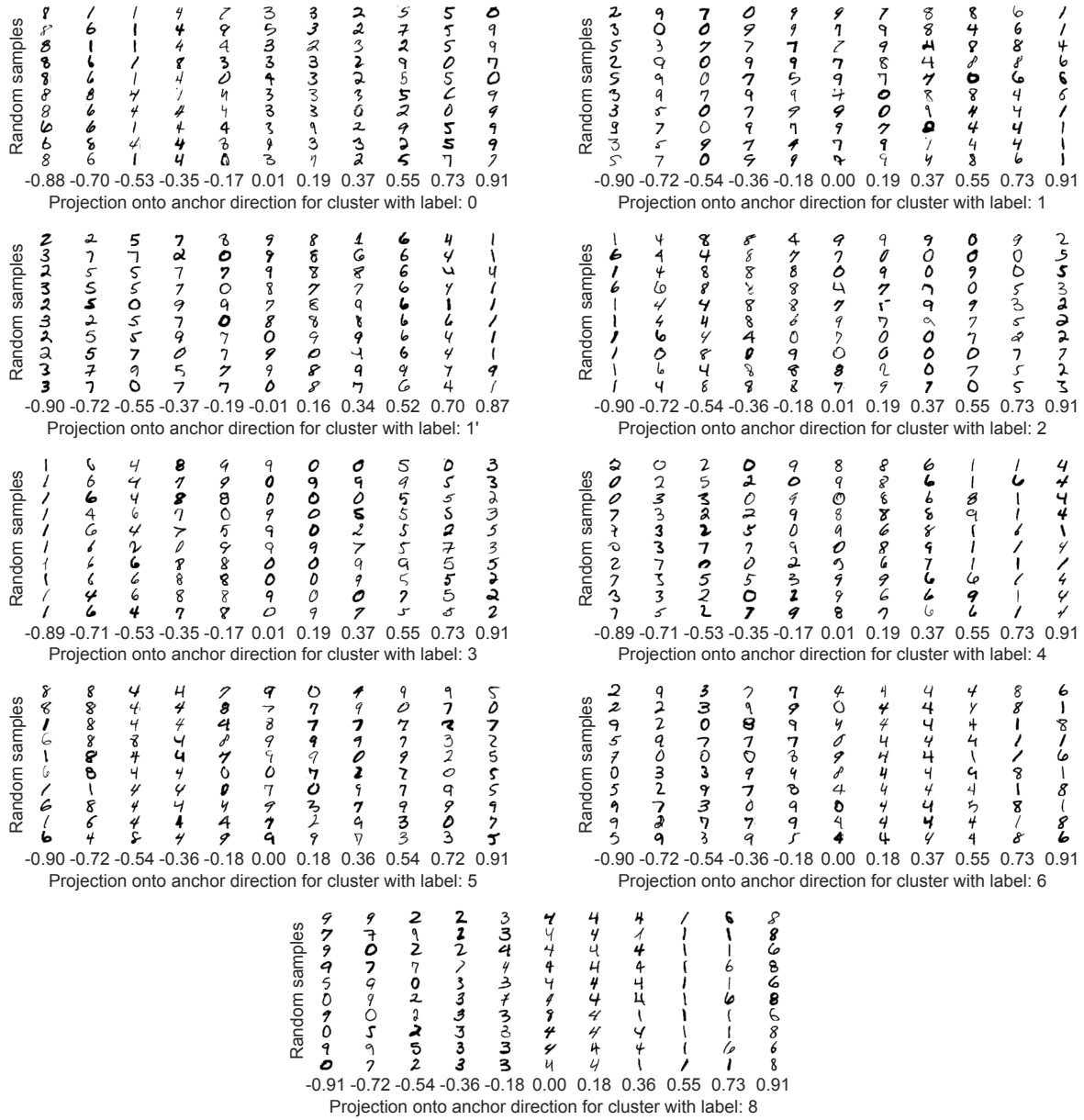

Figure E.8: Survival MNIST: using anchor directions estimated from a clustering model with 9 clusters, we produce random input vs projection plots for the resulting 9 anchor directions. The cluster labels are the same as the ones along the x-axis of Figure E.7.

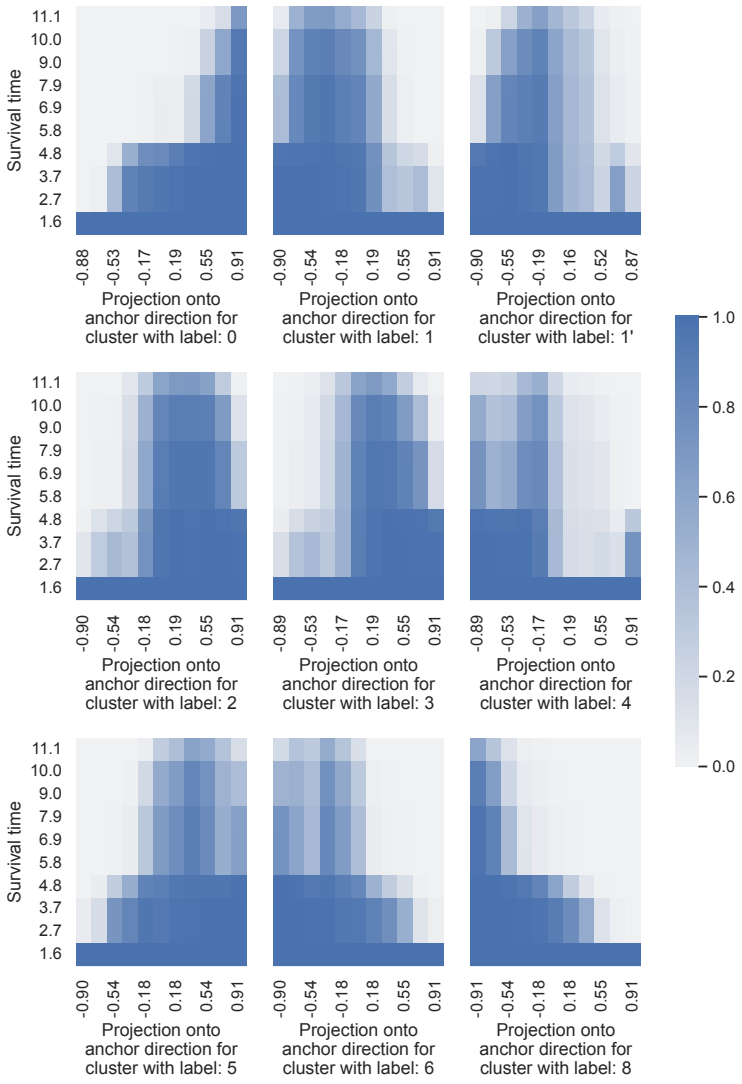

Figure E.9: Survival MNIST: using anchor directions estimated from a clustering model with 9 clusters, we produce survival probability heatmaps for the resulting 9 anchor directions. The cluster labels are the same as the ones along the x-axis of Figure E.7.

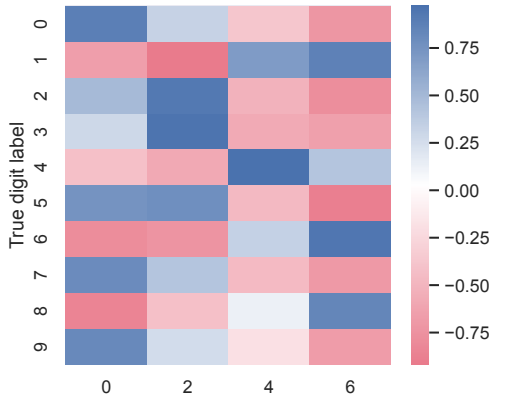

Figure E.10: Survival MNIST: average projection heatmap for a clustering assignment with 4 clusters (using a mixture of von Mises-Fisher distributions). The intensity at the i-th row and j-th column corresponds to average projection value along the j-th cluster's anchor direction across visualization data with ground truth digit label i. Just as in Figure E.7, we label each cluster based on the single digit that it best matches to.

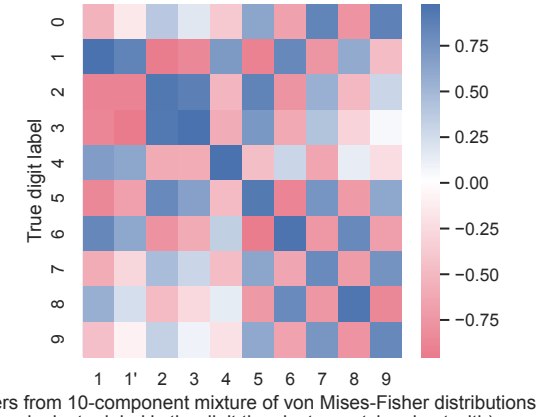

Figure E.11: Survival MNIST: average projection heatmap for a clustering assignment with 10 clusters (using a mixture of von Mises-Fisher distributions). The intensity at the i-th row and j-th column corresponds to average projection value along the j-th cluster's anchor direction across visualization data with ground truth digit label i. Just as in Figure E.7, we label each cluster based on the single digit that it best matches to.

that the cluster best matches to (see Figure E.4 for comparison).

Results for a clustering model with 4 clusters. We next use anchor directions from a 4-component mixture of von Mises-Fisher distributions, again treating each component as a cluster. We show an average projection heatmap in Figure E.10. Note that although we use the same way of labeling each cluster as we did when we used 9 clusters, here we would actually like each cluster to correspond to the ground truth risk groups (so that each cluster does not necessarily only correspond to a single digit).

From Figure E.10, we see that the cluster with label 0 is most like the ground truth risk group $\{0, 7, 9\}$ but also includes digit 5 (which also has a relatively high censoring rate among the different digits). Meanwhile, the cluster with label 6 is most like the risk group $\{1, 6, 8\}$. The clusters with labels 2 and 4 together correspond to the ground truth risk group $\{2, 3, 4\}$. Overall, the ground truth risk groups are not correctly recovered although the clusters found, qualitatively, pick up on some of the ground truth structure. We suspect that the difficulty in determining that digit 5 should be in its own cluster has to do with how often it is censored (over 70%). The risk group corresponding to digits 1, 6, and 8 should be the easiest to recover as these three digits have the lowest censoring rates (all below 6%).

We omit random input vs projection plots and survival probability heatmaps for the different clusters' anchor directions since the findings from these visualizations are qualitatively similar to the findings we just pointed out from looking at Figures E.10 and E.9 (note that for each cluster in this 4-cluster model, the digits that the cluster matches well with tend to have very similar survival probability heat maps).

Results for a clustering model with 10 clusters. When we use 10 clusters for the mixture of von Mises-Fisher distributions, we obtain the average projection heatmap in Figure E.11. In this case, when we match each cluster to a digit, the only digit that does not get matched is digit 0, although digit 0 itself has high projection values for the clusters with labels 7 and 9. Qualitatively, the clustering result here is not too different from the one where we used 9 clusters. The main difference now is that we can somewhat distinguish better between the digits in the risk group $\{0, 7, 9\}$ consisting of digits with the highest censoring rates. We omit the random input vs projection plots and the survival probability heatmaps for the 10 different clusters' anchor directions as these visualizations

do not provide additional insight at this point over the other findings we have already reported.

Appendix F. Theoretical Result on Projection Values When Information Content in Embedding Vectors is All in Magnitudes

Proposition 2 *(*Extreme example where the embedding space information is all in magnitudes*) Suppose that the embedding vectors (of anchor direction estimation and visualization data) are i.i.d. of the form* $(\Xi, 0, \ldots, 0) \in \mathbb{R}^d$ *(i.e., all coordinates are 0 except the first), where* Ξ *is a continuous random variable with positive variance. In this setup, the only direction in the embedding space that matters is along the vector* $\mu = (1, 0, \ldots, 0) \in \mathbb{R}^d$*, which we can take to be the anchor direction of interest. Then the only possible projection values* p_i^V *are* -1 *or* 1*; projection values in the open interval* $(-1, 1)$ *are not possible.*

Proof Let $\mu = (1, 0, \ldots, 0)$; we treat this as the anchor direction as it is the only direction in which the embedding vectors even vary in this proposition's setup. Our visualizations involve plugging in the visualization raw inputs $x_1^{\mathrm{V}}, \ldots, x_{n^{\mathrm{V}}}^{\mathrm{V}}$ into ϕ. We denote $u_i^{\mathrm{V}} \triangleq \phi(x_i^{\mathrm{V}})$, so that the projection value p_i^{V} defined in equation (7) is equal to

$$p_i^{\mathrm{V}} = \mathrm{proj}_{\mu}(x_i^{\mathrm{V}}) = \Big\langle \frac{u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}}}{\|u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}}\|}, \mu \Big\rangle, \qquad (10)$$

where we have used the fact that $\|\mu\| = 1$.

The key observation is that we can write each u_i^{V} as $u_i^{\mathrm{V}} = (\Xi_i^{\mathrm{V}}, 0, \ldots, 0)$ and similarly each u_i^{A} as $u_i^{\mathrm{A}} = (\Xi_i^{\mathrm{A}}, 0, \ldots, 0)$ where the $\{\Xi_i^{\mathrm{V}}\}_{i=1}^{n^{\mathrm{V}}}$ and $\{\Xi_i^{\mathrm{A}}\}_{i=1}^{n^{\mathrm{A}}}$ are all i.i.d. continuous random variables with positive variance. Therefore,

$$\begin{aligned} u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}} &= u_i^{\mathrm{V}} - \frac{1}{n^{\mathrm{A}}} \sum_{i=1}^{n^{\mathrm{A}}} u_i^{\mathrm{A}} \\ &= \Big(\underbrace{\Xi_i^{\mathrm{V}} - \frac{1}{n^{\mathrm{A}}} \sum_{i=1}^{n^{\mathrm{A}}} \Xi_i^{\mathrm{A}}}_{\spadesuit}, 0, \ldots, 0 \Big), \end{aligned}$$

where $\spadesuit$ is a sum of independent continuous random variables with positive variance, so $\spadesuit$ itself is still a continuous random variable with positive variance (note that the variance of the sum of two independent variables is the sum of their variances). This implies that $\spadesuit$ is 0 with probability 0, which in turn implies that with probability 1, $\|u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}}\|$ is nonzero, so $\frac{u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}}}{\|u_i^{\mathrm{V}} - \overline{u}^{\mathrm{A}}\|}$ is well-defined and, in particular, it is either μ or $-\mu$. Then by using equation (10), p_i^{V} is either equal to $\langle \mu, \mu \rangle = 1$ or $\langle -\mu, \mu \rangle = -1$. ■

Appendix G. Handling a Large Number of Clusters/Anchor Directions With the Help of Ranking

When using our heuristic from Section 3.2.1 for choosing the number of clusters to use, it is possible that the number of clusters could be very large — so large that examining visualizations for anchor directions corresponding to all the clusters would be too tedious for model debugging purposes. Of course, one could simply choose to not set the number of clusters to be so large. Put another way, when using our violin plot visualization to help select the number of clusters, one could simply choose a smaller p-value threshold, which would result in fewer clusters being used. However, if for whatever reason, one wants to use a number of clusters that is larger with the goal of having clusters that are more "fine-grain", we now suggest an approach for handling this situation as to reduce the amount of clusters to look at. Note that this approach actually applies more generally to the setting where there are many anchor directions that are under consideration, where the anchor directions need not be estimated from our clustering approach. For example, the anchor directions could be computed based on concepts as in Section 3.2.2, where there is a very large number of concepts under consideration.

The basic idea is to rank the anchor directions. If we have a ranking of the anchor directions, then we could focus on, for instance, a few of the highest and a few of the lowest ranked anchor directions. Alternatively, we could also, for instance, take anchor directions that are "diverse" across ranks: for example, we could choose the 0th percentile-ranked anchor direction (lowest ranked), the 25th percentile, the 50th percentile (median), the 75th percentile, the 100th percentile (highest ranked). In this manner, we can focus on visualizing only a subset of all the anchor directions. We describe one approach to rank anchor directions next.

Ranking anchor directions based on predicted median survival times. One heuristic approach is to compute a median survival time estimate for each anchor direction, and then rank anchor directions based on this median survival time estimate.

Per anchor direction μ, we first determine the visualization data that are in the top α fraction of the projection values along μ (e.g., if $\alpha = 0.1$, then this means that we consider data points with projection values that are within the top 10%). Formally, this set of visualization data points can be written as follows. First, recall that the visualization data have projection values $p_i^{\mathrm{V}} = \mathrm{proj}_\mu(x_i^{\mathrm{V}})$, for $i = 1, \ldots, n^{\mathrm{V}}$. Suppose that we sort these projection values and denote the sorted projection values as $p_{(1)}^{\mathrm{V}} < p_{(2)}^{\mathrm{V}} < \cdots < p_{(n^{\mathrm{V}})}^{\mathrm{V}}$. Then the top α projection value can be estimated by

$$q_\alpha \triangleq p_{(\lceil (1-\alpha) n^{\mathrm{V}} \rceil)}^{\mathrm{V}}.$$

Then the visualization data with projection values in the top α percentile of projection values along μ are the ones in the set

$$\mathcal{I}_\mu^{\mathrm{top}\,\alpha} \triangleq \{i \in \{1, 2, \ldots, n^{\mathrm{V}}\} \text{ s.t. } p_i^{\mathrm{V}} \geq q_\alpha\}.$$

Note that this equation is similar to that of equation (8). We then compute the survival curve for the data points in $\mathcal{I}_\mu^{\mathrm{top}\,\alpha}$ using an equation analogous to equation (9):

$$\widehat{S}_\mu^{\mathrm{top}\,\alpha}(t) \triangleq \frac{1}{|\mathcal{I}_\mu^{\mathrm{top}\,\alpha}|} \sum_{i \in \mathcal{I}_\mu^{\mathrm{top}\,\alpha}} \widehat{S}(t|x_i^{\mathrm{V}}).$$

By a standard result in survival analysis, the time t where the survival curve $\widehat{S}_\mu^{\mathrm{top}\,\alpha}(t)$ crosses 1/2 corresponds to a median survival time estimate (see, for instance, Reid 1981). In particular, we denote this median survival time estimate as

$$\widehat{\mathrm{med}}_\mu^{\mathrm{top}\,\alpha} \triangleq \inf\{t \geq 0 \text{ s.t. } \widehat{S}_\mu^{\mathrm{top}\,\alpha}(t) \leq 1/2\}.$$

In practice, to compute the infimum, commonly a discrete time grid is used, such as using all the unique observed times in the training data (i.e., the unique Y_i values), and if the survival curve never crosses 1/2 over this discrete time grid, then for simplicity we just take the median survival time estimate to be a special value specifying that it is greater than the maximum observed time in the training data.

Note that what we stated above is for any anchor direction μ. Thus, if we have k anchor directions denoted as $\mu_1, \ldots, \mu_k$, then we can rank these anchor directions by $\widehat{\mathrm{med}}_{\mu_1}^{\mathrm{top}\,\alpha}, \ldots, \widehat{\mathrm{med}}_{\mu_k}^{\mathrm{top}\,\alpha}$.

SUPPORT dataset example. Consider the data and setup from Section 3.4, which is the same setting as the additional results in Appendix B.2. By using the above approach for ranking clusters/anchor directions based on estimated median survival times and setting $\alpha = 0.1$, we get the following ranking of the five clusters (in ascending order of estimated median survival times):

1. Cluster 1: median survival time estimate 46 days
2. Cluster 5: median survival time estimate 105 days
3. Cluster 2: median survival time estimate 236 days
4. Cluster 3: median survival time estimate 452 days
5. Cluster 4: median survival time estimate 1895 days

Appendix H. Baseline Visualization Strategy: Use Dimensionality Reduction to Plot the Embedding Space

We present PCA and t-SNE plots using the baseline visualization strategy described at the end of Section 1. We show plots for the DeepSurv embedding space for the SUPPORT dataset (using the setup in Section 3.4/Appendix B.2) in Figure H.1(*a*), the Rotterdam/GBSG datasets (using the setup in Appendix C) in Figure H.1(*b*), and the Survival MNIST dataset (using the setup in Section 3.5/Appendix E) in Figure H.1(*c*). In particular, the embedding spaces under examination all have a norm 1 constraint. Each scatter plot is made using the visualization data (and not the training data used to train the neural survival analysis model nor the anchor direction estimation data). For each visualization data point x_i^{V}, we compute its median survival time estimate by looking at the time t where $\widehat{S}(t|x_i^{\mathrm{V}})$ crosses 1/2 (similar to what we had discussed in Appendix G), and we color scatter plot points based on these median survival times. From these scatter plots, we can get a rough sense of the geometry of the embedding space. For instance, whereas there are clear clusters of points that show up for Survival MNIST (in fact, one could check that these correspond to different groups of digits; again, as we pointed out in Section 3.5/Appendix E, the embedding space does not appear to disentangle all 10 digits neatly), we do not see this clustering behavior for the SUPPORT and Rotterdam/GBSG DeepSurv embedding spaces.

Importantly, as we already pointed out in Section 1, these scatter plots from dimensionality reduction do

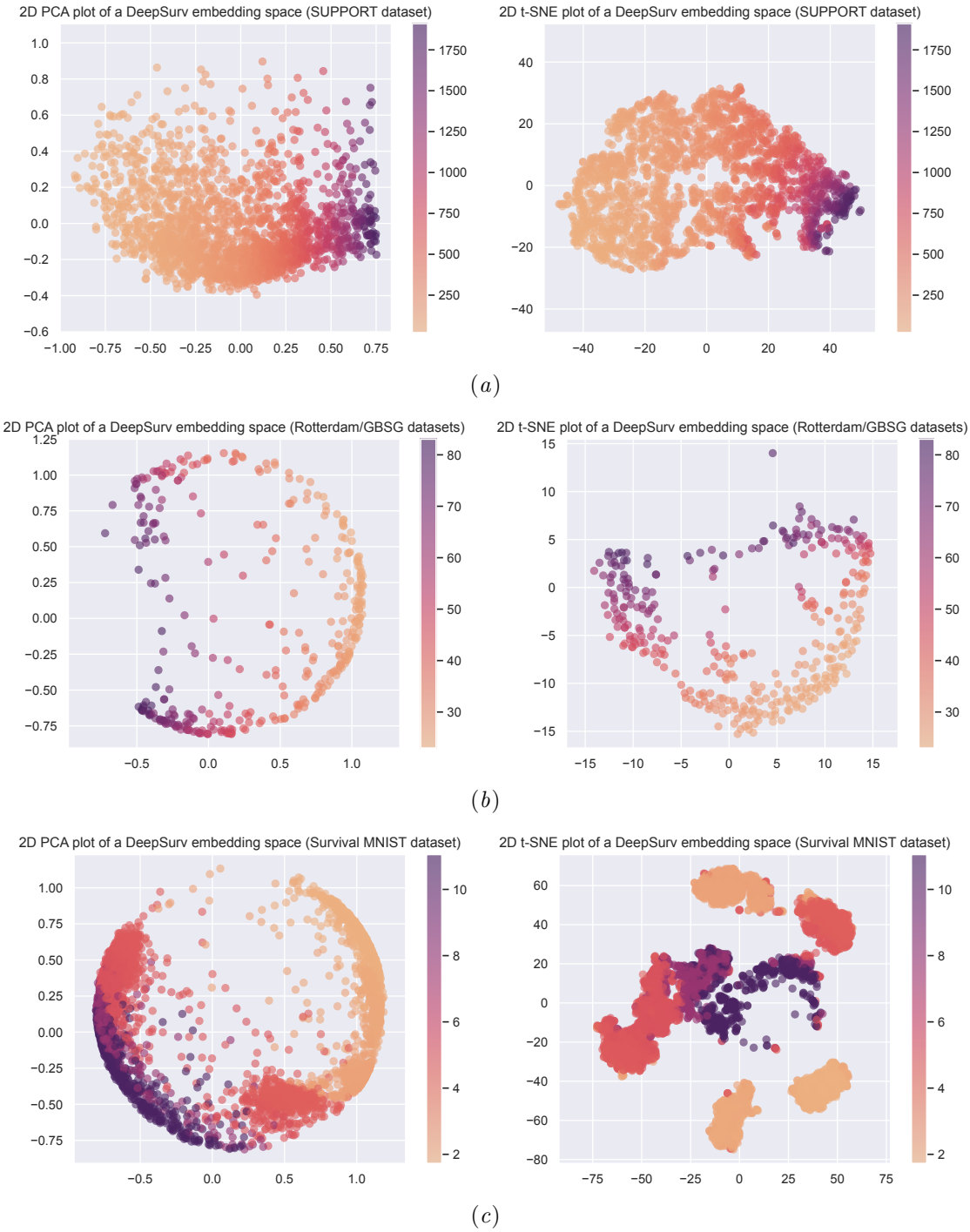

Figure H.1: 2D PCA and t-SNE plots of the visualization data's embedding vectors from a DeepSurv model (with a norm 1 constraint) for the (a) SUPPORT, (b) Rotterdam/GBSG, and (c) Survival MNIST datasets. The colors indicate estimated median survival times.

not tell us how the embedding space relates to raw features. Even PCA, which is easier to interpret than nonlinear dimensionality reduction methods (e.g., t-SNE), does not relate the embedding space to raw features in this setting since PCA here is directly applied to vectors from the embedding space (and not vectors from the raw feature space). While the t-SNE plot for Survival MNIST shows clustering behavior, note that t-SNE itself does not actually estimate cluster assignments for different data points, i.e., t-SNE is inherently *not* a clustering algorithm.

Note that the PCA plots can actually give us a sense of whether information in the embedding space is stored more in magnitudes vs more in angles. As a reminder, Euclidean vectors with norm 1 reside on what is called the "unit hypersphere" $\mathcal{S}^{d-1} \triangleq \{v \in \mathbb{R}^d \text{ s.t. } \|v\| = 1\}$. When we take data on the unit hypersphere and plot their 2D PCA plot, the resulting 2D PCA plot will always look like points that are within a 2D circle (since PCA is a linear dimensionality reduction method, it retains the hyperspherical structure but projects down to 2D, where points can be projected inside the circle rather than only along th shell of the circle). This plot could be helpful. We can readily tell if the data appear uniformly distributed over a hypersphere or not. For example, for the SUPPORT dataset's 2D PCA plot in Figure H.1(a), the points largely bunch up on one side of a circle, meaning that in the embedding space (that in this case is actually 10-dimensional), the points largely are concentrated around a hyperspherical cap (i.e., the embedding vectors are largely all pointed in a similar direction). In contrast, the 2D PCA plots for the Rotterdam/GBSG datasets (Figure H.1(b)) and the Survival MNIST dataset (Figure H.1(c)) clearly show more of a circle shape, indicating that the DeepSurv embedding vectors are more uniformly distributed for Rotterdam/GBSG and Survival MNIST (i.e., they have information stored more in angles than in magnitudes) than for SUPPORT.

We remark that it is possible to color the scatter plot points using other quantitative values. For example, we could use the mean (instead of the median) survival time estimate, which corresponds to the area under a data point's predicted survival curve, we could use an indicator value for whether the point is censored or not (to get a sense of whether some parts of the embedding space correspond to more censored points), or we could use cluster labels as estimated using any of the clustering approaches we had used to estimate anchor directions.