**Abstract**

A widely used paradigm to improve the generalization performance of high-capacity neural models is through the addition of auxiliary unsupervised tasks during supervised training. Tasks such as similarity matching and input reconstruction have been shown to provide a beneficial regularizing effect by guiding representation learning. Real data often has complex underlying structures and may be composed of heterogeneous subpopulations that are not learned well with current approaches. In this work, we design \textsc{ExpertNet}, which uses novel training strategies to learn clustered latent representations and leverage them by effectively combining cluster-specific classifiers. We theoretically analyze the effect of clustering on its generalization gap, and empirically show that clustered latent representations from \textsc{ExpertNet} lead to disentangling the intrinsic structure and improvement in classification performance. \textsc{ExpertNet} also meets an important real-world need where classifiers need to be tailored for distinct subpopulations, such as in clinical risk models. We demonstrate the superiority of \textsc{ExpertNet} over state-of-the-art methods on 6 large clinical datasets, where our approach leads to valuable insights on group-specific risks.

1 Introduction

Overfitting is a common problem in high-capacity models, such as neural networks, that adversely affects generalization performance. A standard approach to reduce overfitting is regularization. Various regularization strategies have been developed such as those based on norm penalties and some specifically designed for neural networks such as dropout and early stopping [Goodfellow et al., 2016]. The use of auxiliary tasks have also been explored for regularization. The rationale comes from multi-task learning, which is known to improve generalization through improved statistical strength via shared parameters [Baxter, 1995; Caruana, 1997].

Since there is no dependence on label acquisition, several unsupervised auxiliary tasks have been explored for regularizing neural networks. Input reconstruction is a common choice, often implemented using autoencoder-like architectures, e.g. [Rasmus et. al., 2015; Zhao, 2015; Zhang et al., 2016; Le et al., 2018]. Such reconstruction-based regularization has the advantage of yielding hidden layer representations closer to the original data topology.

Real data, however, contains complex underlying structures like clusters and intrinsic manifolds that are not disentangled by the bottleneck layer of autoencoders. This has been studied in the context of purely unsupervised models that have explored the joint tasks of clustering and dimensionality reduction (DR) to find “cluster-friendly” representations. E.g., in [Yang, 2017], it is shown that joint clustering and DR outperforms the disassociated approach of independently performing DR followed by clustering. Recognizing the importance of preserving cluster structure in latent embedded spaces, we hypothesize that, in addition to reconstruction, clustering-based regularization could improve generalization performance in supervised models.

Our work is also motivated by the need to develop models tailored to distinct subpopulations in the data. This is a common requirement in clinical risk models since patient populations can show significant heterogeneity. Most previous works adopt a ‘cluster-then-predict’ approach to model such data e.g., [Ibrahim et. al., 2020], where clustering is first done independently to find subpopulations (called sub-types) and then predictive models for each of these clusters are learnt. Such subtype-specific models are often found to outperform models learnt from the entire population [Moshoudnia and Ebrahimpour, 2014]. They also provide a form of interpretability and subsequent clinical decisions can be personalized to each patient group’s characteristics. However, in these approaches, clustering is performed independent of classifier training, and may not discover latent structures that are beneficial for subsequent classification. This leads us to hypothesize that combined clustering with supervised classification will lead to better performance compared to such cluster-then-predict approaches.

We thus design \textsc{ExpertNet}, a deep learning model that performs simultaneous clustering and classification. \textsc{ExpertNet} consists of components that have both local and global view of the data space. The local units specialize in classifying observations in clusters found in the data while the global unit is responsible for generating cluster friendly embeddings based on the feedback given by local units. Apart
from being a supervised model, EXPERTNET can function as an unsupervised model that can perform target-specific clustering. In summary, our contributions are:

1. **Model:** We design EXPERTNET, a neural model for simultaneous clustering and classification with cluster-specific local networks. We introduce novel training strategies to obtain latent clusters and use them effectively to improve generalization of the classifiers.

2. **Theoretical Guarantees:** We analyze the effect of clustering on EXPERTNET’s generalization gap.

3. **Experiments:** Our extensive experiments demonstrate the efficacy of our model over both reconstruction-based regularization and cluster-then-predict approaches. On 6 large real-world clinical datasets EXPERTNET achieves clustering performance that is comparable to, and achieves classification performance that is considerably better than, state-of-the-art methods.

## 2 Background and Related Work

Unsupervised learning techniques are often used to pretrain a neural network before finetuning with supervised tasks [Bengio et al., 2007]. Input reconstruction is commonly used to regularize deep models. [Zhang et al., 2016] jointly train a supervised neural model with an unsupervised reconstruction network. [Zhao, 2015] showed that by adding decoding pathways, to add an auxiliary reconstruction task to the objective, improves the performance of supervised networks. Similarly [Le et al., 2018] append a prediction layer to the embedding layer to improve generalization performance.

A more dissociated way of using clustering to benefit supervised models is to simply find clusters in the data and then train separate models on each of them. Amongst such techniques, there are some algorithms [Finley and Joachims, 2008; Gu and Han, 2013; Fu et al., 2010] that impose a global regularization scheme on all the individual classifiers. In some other works [Reyna et. al., 2019; Lasko et. al., 2013; Suresh et. al., 2018], all the individual classifiers are completely independent. Such analysis is common in clinical data where patient stratification is done followed by predictive modeling. E.g., Deep Mixture Neural Networks (DMNN) [Li et al., 2020b], designed with this aim, is based on an autoencoder architecture while having an additional gating network, which is first trained to learn clusters through $k$-means. Local networks corresponding to each cluster are trained using a weighted combination of losses, where the weights are determined by the gates.

Simultaneous clustering and representation learning have been used in purely unsupervised settings. Examples include neural clustering models such as Deep Embedded Clustering (DEC) [Xie et al., 2016] and Improved DEC (IDEC) [Guo et al., 2017] and Deep Clustering Network (DCN) [Yang, 2017]. Recent self-supervised learning techniques also employ cluster centroids as pseudo class labels in embedded data space to improve the quality of representations [Li et al., 2020a; Caron et al., 2018].

We briefly describe DEC [Xie et al., 2016] as the loss function in EXPERTNET is based on their formulation. Their key idea is to find latent representations $z_i$ for data points using an autoencoder by first pretraining it. Initial cluster centers $\{\mu_j\}$ are obtained by using $k$-means on the latent representations. The decoder is then removed and the representations are fine tuned by minimizing the loss function:

$$L_c = KL(P||Q) = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

(1)

where $q_{ij}$ is the probability of assigning the $i^{th}$ data point ($z_i$) to the $j^{th}$ cluster (with centroid $\mu_j$) measured by the Student’s $t$-distribution [V. D. Maaten and Hinton, 2008] as:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}}$$

(2)

The target distribution $p_{ij}$ is defined as

$$p_{ij} = \frac{q_{ij}/\sum_j q_{ij}}{\sum_j (q_{ij}/\sum_i q_{ij})}$$

(3)

The predicted label of $x_i$ is $\arg\max_j q_{ij}$. Minimizing the loss function $L_c$ is a form of self-training as points with high confidence act as anchors and distribute other points around them more densely.

DEC, IDEC and DCN are all centroid-based and have objectives similar to that of $k$-means. In such approaches, the encoder can map the centroids to a single point to make the loss zero and thus collapse the clusters. Hence, previous methods employ various heuristics to balance cluster sizes, e.g., through sampling approaches [Caron et al., 2018] or use of priors [Jitta and Klami, 2018].

## 3 EXPERTNET

Given $N$ datapoints $\{x_i \in X\}_{i=1}$, and class labels $y_i \in \{1,...,B\}$, associated with every point $x_i$, our aim is to simultaneously (i) cluster the $N$ datapoints into $k$ clusters, each represented by a centroid $\mu_j$, $j \in \{1,...,k\}$, and (ii) build $k$ distinct supervised classification models within each cluster to predict the class labels. Note that during training, labels are used only for building the classification models and not for clustering.

### Network Architecture

Fig. 1 shows the neural architecture of EXPERTNET that consists of an encoder, a decoder and $k$ local networks $(L_N)$. The encoder $f(\mathcal{U}) : X \rightarrow Z$ is used to obtain low-dimensional representations of the input datapoints. Cluster structure is learnt in this latent space and representations in each cluster are used in local networks, $h(\mathcal{W}_j \mathcal{V}) : Z \rightarrow \hat{y}_j$ for $j = 1 \ldots k$ to train $k$ classification models. In addition, there is a decoder $g(\mathcal{V}) : Z \rightarrow X$ that is used to reconstruct the input from the embeddings. EXPERTNET is parameterized by three sets of weights: $\mathcal{U}, \mathcal{V}, \{\mathcal{W}_j\}_{j=1}^k$ which are learnt by optimizing a combination of losses as described below.

### Loss Function

The overall loss function $L$ is a weighted combination, with coefficients $\beta, \gamma > 0$, of the reconstruction loss $L_r$, clustering loss $L_c$, cluster balance loss $L_{bal}$ and classification loss $L_c$:

$$L = L_r + \beta \cdot L_c + \gamma \cdot L_s + \delta \cdot L_{bal}$$

(4)

$L_c$ is defined as the KL divergence loss (Eq. 1), where the cluster membership distribution $Q$ (Eq. 2) uses repre-
sentations $z_i$ and cluster centroids $\mu_j$ inferred during EXPERTNET training. As suggested in [Guo et al., 2017; Yang, 2017], to prevent distortion in the latent space and improve clustering performance, we add the reconstruction loss measured by mean squared error:

$$L_r = \sum_{i=1}^{N} \| x_i - g(f(x_i)) \|^2$$  \hspace{1cm} (5)

We design a novel cluster balance loss to discourage unevenly distributed cluster sizes. We define the ‘soft’ size of a cluster $C_j$ as $|C_j| = \sum_i q_{ij}$ and cluster support counts $\mathcal{S} = \{|C_1|, |C_2|, \ldots, |C_k|\}$ as a $k$-dimensional probability distribution. Let $U_k = (1/k)U(0, 1)$ denote the $k$-dimensional uniform distribution function. We use the Hellinger distance ($H$), which measures the dissimilarity between two distributions, as the loss $L_{bal} = H(\mathcal{S} || U_k) = \frac{1}{2} \| \sqrt{\mathcal{S}} - \sqrt{U_k} \|_2$. $L_s$ is a weighted cross entropy loss described in the following.

**EXPERTNET Training**

To initialize the parameters $(U, V)$, we first pre-train the encoder and decoder with the input data using only the reconstruction loss $L_r$. This is followed by $k$-means clustering on $\{z_i = f(x_i; U)\}_{i=1}^{N}$ to obtain cluster centroids $\{\mu_j\}_{j=1}^{k}$ which are used to calculate cluster membership and target distributions, $Q, P$. After initialization, we use mini-batch Stochastic Gradient Descent to train the entire network, using the loss $L$ (details are in Appendix E). To stabilize training we update $P$ only after every epoch.

Since the embeddings get updated progressively in every iteration, we train the LNs for a larger number of sub-iterations within every iteration of the main training loop. The number of sub-iterations gradually increases (1 per every 5 epochs until a max-limit we set to 10). This enables the LNs to learn better from the stabilized clustered embeddings than from the intermediate representations.

Further, we design a novel strategy called **stochastic cohort sampling** used in each sub-iteration, which leads to more robust classifiers, as verified in our experiments. Instead of training each LN with a fixed cluster of data points (e.g., using $\arg\max_{j=1}^{k} q_{ij}$), we leverage the probabilistic definition of clusters to obtain multiple, different cluster assignments for the same set of embeddings. Considering $q_{ij}$, the $i^{th}$ row of $Q$, as a cluster probability distribution for $z_i$, we sample a random variable $s_i$ from $q_{ij}$, denoting the cluster assignment for the point $x_i$. We can define a cluster realization $C_m = \{i : s_i = m\}_{i=1}^{N}$ (i.e. index of all the points assigned to cluster $C_m$) and $X_m = \bigcup_{j \in C_m} \{x_j\}$. The LNs are trained on these cluster realizations for $T - 1$ sub-iterations without backpropagating the error to the encoder. The individual errors are collected from all the $k$ LNs only at the last ($T^{th}$) iteration and backpropagated to the encoder to adjust the cluster representations accordingly. The final classification loss is a weighted cross entropy (CE) loss:

$$L_s = \frac{1}{k} \sum_{j=1}^{k} q_{ip} \cdot CE(y_p, h_j(x_p; V_j)).$$  \hspace{1cm} (6)

After training, the encoder network is frozen and the local networks are finetuned on the latent embeddings via stochastic cohort sampling to further improve LN performance.

**EXPERTNET Prediction**

Prediction can be done using the encoder and local networks. For a test point $\hat{x}_p$, the soft cluster probabilities $(\hat{q}_{ip})$ are calculated from Eq. 2. All the local networks (see stochastic cohort sampling above) can be used to predict the class label $\hat{y}_p = \sum_{j=1}^{k} \hat{q}_{ip} h_j(\hat{x}_p)$.

4 Theoretical Analysis

Let $\Omega_{j \neq j'} = \{\Omega_j\}_{j=1}^{k}$ be a partition of $\mathcal{X}$ such that $\mathcal{X} = \bigcup_{j=1}^{k} \Omega_j$ where $\Omega_j \cap \Omega_{j'} = \emptyset$ for $j \neq j'$. This corresponds to the

**Algorithm 1: ExpertNet Training**

**Input:** Training Data: $X \in \mathbb{R}^{n \times d}$, labels $y \in \mathbb{R}^{n \times 1}$, $k$, $f(\cdot; U)$, $g(\cdot; V)$ and $\{h_j(\cdot; W_j)\}_{j=1}^{k}$, $\mathcal{F}$

1. $\triangleright$ **Initialization**
2. Pre-train $f(\cdot; U)$ & $g(\cdot; V)$ via back-propagating loss in eq. 5
3. Compute $\mu(C_j) = \sum_j \mathbb{E}_{y_j \sim X_j} y_j$ for $j \in [k]$ and $b \in [B]$
4. Compute matrices $Q$ and $P$ according to eqs. 2 and 3
5. $\triangleright$ **Algorithm**
6. While Validation AUC increases do
7. for every mini-batch $X_b$ do
8. Pre-train $f(X_b; U)$
9. Calculate $Q_b$ by eq. (2)
10. for $T$ sub-iterations do
11. Sample $\{s_i \sim q_{ij}\}_{i=1}^{N}$
12. Calculate $X_m = \{x_i : s_i = j\}_{i=1}^{N}$
13. Train local classifiers $\{h_j\}_{j=1}^{k}$ on $(X_j, Y_j)$
14. Update $\{W_j\}_{j=1}^{k}$
15. Backpropagate $L = L_r + C_{bal} : L_c + \gamma \cdot L_s + \delta \cdot L_{bal}$ and update $\mu, U, V$ and $W$
16. Update $P$ via eq. (3)
17. $\triangleright$ **Fine tune Local Classifiers**
18. For every cluster $C_j$, train a classifier $h_j$ on $(X_j, Y_j)$.
19. **Output** Trained EXPERTNET model. Cluster centroids $\mu = \{\mu_j\}_{j=1}^{k}$

Figure 1: EXPERTNET Architecture
Suppose that for all \( \Omega_j \), the function \( \sigma_j \) is 1-Lipschitz and positive homogeneous for all \( l \in [\max(L - 1, Q - 1)] \) and \( \|x^2\| \leq B_j \) for all \( x^2 \in \Omega_j \). Let \( F_j = \{x \in \Omega_j \rightarrow (y \circ g)(x) : (\forall l \in [L - 1]) [\|W_l^j\|_F \leq M_l^j \land \|W_l^j\|_F \leq M_l^j]\} \) and \( F^k = \{x \rightarrow f(x) : f(x) = \sum_{j=1}^{k} \{x \in \Omega_j\} f_j(x), f_j \in F_j\} \). Then, for any \( \delta > 0 \), with probability at least \( 1 - \delta \) over an i.i.d. draw of \( m \) i.i.d. test samples \((x_i, y_i)\) \( m \rightarrow \infty \), the following holds: for all maps \( f^k \in F^k \),

\[
E_{x,y}[\ell_{01}(f^k(x), y)] - \hat{E}_{x,y}[\ell_{01}(f^k(x), y)] \\
\leq \zeta_1 \left( \sum_{j=1}^{K} B_j \prod_{l=1}^{L} M_l^j \right) \\
+ 3 \sqrt{\frac{k \ln(2k/\delta)}{2N}}
\]

where \( \zeta_1 = 2 \rho^{-1} \left( \sqrt{2 \log(2)(L + Q)} + 1 \right) \prod_{l=1}^{Q} M_l^0 \) is a \( k \) independent term.

The proof is presented in Appendix A. Theorem 1 shows the generalization bound with the following:

- Increasing \( k \) can reduce the empirical loss \( E_{x,y}[\ell_{01}(f^k(x), y)] \) (since it increases the expressive power), resulting in a tendency for a better expected error \( E_{x,y}[\ell_{01}(f^k(x), y)] \).
- Increasing \( k \) increases the last term \( \sqrt{\frac{k \ln(2k/\delta)}{2N}} \), resulting in a tendency for a worse expected error \( E_{x,y}[\ell_{01}(f^k(x), y)] \).
- Increasing \( k \) can reduce the complexity term \( \sum_{j=1}^{K} B_j \prod_{l=1}^{L} M_l^j \) \( \sqrt{\frac{kN}{2}} \) (since (1) each sub-network only needs to learn a simpler classifier with larger \( k \), resulting in a smaller value of \( \prod_{l=1}^{L} M_l^j \); (2) each domain decrease and hence \( B_j \) decrease as \( k \) increase), resulting in a tendency for a better expected error \( E_{x,y}[\ell_{01}(f^k(x), y)] \).

Not surprisingly, the generalization gap is inversely dependent on \( N \), the total number of data points, indicating that more observations will lead to a more accurate model. There is no simple relationship between \( k \) suggesting that the best number of clusters might have to be found empirically.

# 5 Experiments

We evaluate the classification and clustering performance of ExpertNet on 6 large clinical datasets. We analyze ExpertNet through an ablation study and investigate its sensitivity to model hyperparameters. Finally, we present a case study that illustrates its utility in clinical risk modeling.

## 5.1 Data

We use 6 large clinical datasets: Table 1 lists the number of observations and features in each of them. WID data is from the Women In Data Science challenge [Lee et al., 2020] to predict patient mortality. Diabetes data is from the UCI Repository where the task is readmission prediction. All the remaining datasets have data from the MIMIC III ICU Database [Fei et al., 2020]. CIC and Sepsis have been used for mortality and sepsis prediction challenges in Physionet [Silva, 2012; Reyna et al., 2019]. Kidney and Respiratory data has been extracted by us for the tasks of Acute Kidney Failure and Acute Respiratory Distress Syndrome prediction. All the above tasks are posed as binary classification problems. We also derive a multiclass dataset (CIC-LOS) from the CIC dataset where we predict Length of Stay (LOS), discretized into 3 classes (based on 3 quartiles). We consider a random 57-18-25 split to create training, validation and testing splits in each dataset. More details of data preprocessing and feature extraction are in Appendix D.

| Dataset                  | #Instances | #Features | MAJ |
|--------------------------|------------|-----------|-----|
| WID Mortality [Lee et al., 2020] | 91711      | 241       | 0.91|
| Diabetes [Lichman., 2013] | 100000     | 9         | 0.54|
| Sepsis [Reyna et al., 2019] | 40328      | 89        | 0.93|
| Kidney [Fei et al., 2020] | 14087      | 89        | 0.51|
| Respiratory [Fei et al., 2020] | 22477      | 89        | 0.92|
| CIC [Silva, 2012]        | 12000      | 117       | 0.85|
| CIC-LOS                  | 12000      | 116       | 3 classes |

Table 1: Summary of datasets. MAJ: proportion of majority class instances (for binary labels).

## 5.2 Classification and Clustering Performance

### Baselines

As baselines for the classification task, we use 3 different kinds of techniques. The first is a feedforward neural network (baseline) that only performs classification. It’s architecture is set to be identical to that of a combination of ExpertNet’s encoder and a single local network. The second is the Supervised Autoencoder (SAE) [Le et al., 2018] that uses reconstruction loss as an unsupervised regularizer (similar architecture as baseline). The third category uses the common ‘cluster-then-predict’ approach, where clustering is in-
Table 2: Clustering and Classification performance. Best values in bold. NA indicates no results due to an empty cluster. Note that HTFD values are not defined for \( k = 1 \).

dependently performed first and classifiers are trained on each cluster (denoted by \(-Z\)). In this category, we compare with 3 clustering methods \( k\)-\textit{means} (where we use an autoencoder to get embeddings which are then clustered using \( k\)-\textit{means}), Deep Clustering Network \textit{DCN} [Yang, 2017] and Improved Deep Embedded Clustering (IDEC) [Guo et al., 2017]. We also use Deep Mixture Neural Networks (DMNN) [Li et al., 2020b] that follows this paradigm. Note that their implementation only supports binary classification. We use stochastic cohort sampling during prediction in all baselines that use clustering for a fair comparison. We compare the clustering performance obtained by \textit{EXPER\textbullet\textit{NET}} with that of DMNN, \( k\)-\textit{means}, DCN and IDEC.

**Performance Metrics**

Standard classification metrics, Area Under the Receiver Operating Characteristic (AUC) and F1 scores, are used for binary classification. For multiclass setting, we use one-vs-rest algorithm to calculate AUC.

To evaluate clustering we use Silhouette scores and another score, HTFD, described below, that evaluates feature discrimination in the inferred clusters. A common practice in clustering studies (e.g., [Li et al., 2015]) is to check, for each feature, if there is a statistically significant difference in its distribution across the clusters. A low p-value (< 0.05) indicates significant difference. To aggregate this over all features \( F \), we define, for each cluster in a clustering, the metric Hypothesis Testing based Feature Discrimination (HTFD):

\[
HTFD(C_i) = \frac{1}{|F|} \sum_{f \in F} - \ln \left( \text{p-value}(X_i^f, X^f) \right) > 0.05
\]

where \( X_i^f \) denotes the values of feature \( f \) for data points in cluster \( C_i \) and \( X^f \) denotes the feature values of data points in all clusters except \( C_i \). Student’s t-test is used to obtain the p-value. The negative logarithm of p-value is multiplied by the significance level, to normalize and obtain a measure.
wherein higher values indicate better clustering. To obtain an overall value for the entire clustering, we take the average of individual, cluster-wise HTFD values.

### Hyperparameter Settings

For DCN-Z and IDEC-Z, we use default parameters as suggested by their authors. For EXPERTNET we let \((\beta, \gamma, \delta) = (0.5, 1.5, 1)\). We select these default values by considering the sensitivity analysis (see Fig. 2). The common encoder has layers of size 128-64-32 and local expert network has layers of size 64-32-16-8. The size of latent embeddings is 20. We evaluate all the methods for four different values of \(K = 1, 2, 3, 4\).

### Results

Table 2 shows the performance of all methods compared, on classification and clustering. We observe that cluster-then-predict methods (*Z) and DMNN generally perform better than SAE, which suggests that clustering-based regularization is more effective than reconstruction-based regularization. SAE outperforms the baseline on most datasets. These results align with previous research that show that unsupervised regularization aids supervised learning. EXPERTNET outperforms all methods on all datasets for at least one input cluster size. Also note that EXPERTNET’s performance does not degrade with increasing number of clusters as seen for KMeans-Z. Overall, for classification, EXPERTNET consistently outperforms all the baselines on all datasets.

With respect to clustering, the performance of EXPERTNET is superior to the best baseline in 5 datasets, for some values of \(K\). For other values of \(k\), and on other datasets, the performance values are comparable or lower. F1 scores and Silhouette scores are in Appendix B, where the performance trends are similar. Overall, the results show that EXPERTNET achieves clustering performance that is comparable to, and achieves classification performance that is considerably better than, the state-of-the-art alternatives respectively.

### 5.3 Sensitivity Analysis and Ablation Studies

We evaluate the effect of hyperparameters \(\beta, \gamma\) and \(\delta\) on EXPERTNET. We individually vary each hyperparameter while setting other two to 0 and measure the classification performance. Figures 2a, 2b and 2c show the results for the CIC dataset. We observe that both the F1 and AUC values are fairly robust to changes in their values.

Our stochastic cohort sampling approach may or may not be used independently during training and prediction. To evaluate its effect we evaluate all four combinations through an ablation study. We denote the combinations by TT, TF, FT and FF, where the first and second positions indicate training and prediction respectively. \(T\) indicates use of our approach while \(F\) indicates that it is not used. Figure 2d and 2e show the F1 and AUC scores for all 4 combinations on the CIC dataset (results on Sepsis dataset in Appendix C). The best performance is achieved when the approach is used both in training and prediction (TT). For clustering, there is no prediction. Figure 2f shows that the performance is better with rather than without sampling. All the results shown are averages over 5 runs.

| \(C_i\) | \%D | \%D | \%D |
|---|---|---|---|
| \(|C_1| = 2458\) | \%D = 0.115 | \%D = 0.032 | \%D = 0.254 |
| Age | Age | GCS_last | GCS_last |
| BUN_last | BUN_last | GCS_last | Age |
| RespRate_median | BUN_first | Lactate_last | BUN_last |
| BUN_first | GCS_last | BUN_first | BUN_last |
| Weight_last | GCS_last | Length_of_stay | MechVentDuration |
| HR_highest | MechVentDuration | GCS_median | GCS_median |
| Weight | BUN_last | SOFA | BUN_last |
| Weight | RespRate_median | Weight | BUN_last |
| RespRate_highest | HR_highest | HCO3_last | GCS_last |

Table 3: Important Features for mortality prediction in respective clusters. Features common in all three clusters are highlighted in yellow while those common in two are highlighted in purple. Rest are unique to their respective clusters.

### 5.4 Case Study: Mortality Prediction

As a case study, we illustrate the use of EXPERTNET on the CIC mortality prediction data for \(k = 3\) clusters. Since clustering is done simultaneously with classification, the clusters are influenced by the target label, i.e., mortality indicator, and thus, by design, EXPERTNET is expected to find mortality subtypes. In other words, we expect the inferred clusters to have different risk factors tailored to each underlying subpopulation. To evaluate this, we examine feature importances for each cluster’s local risk model. We distil the knowledge of the local networks into a simpler student model [Gou et al., 2021], a Gradient Boosting Classifier, in our case, that provides feature importance values.

In Table 3, we list the cluster sizes \(C_i\), proportion of patients who do not survive \((\%D)\) and the top 10 most important features in each cluster. We see that, as expected, the risk models created by EXPERTNET deem different sets of features as important for predicting mortality. Out of the 10 most important features, each risk model contains 3 common features, 4, 3 and 1 features common across two models respectively and 3, 4 and 6 features unique to respective clusters. Appendix F presents further analysis of the clusters.

### 6 Conclusion

We design EXPERTNET a model for combined clustering and classification and theoretically analyze its generalization properties. Our model can be viewed as a mixture of expert networks trained on inferred clusters in the data. Our experiments show that both regularization through unsupervised clustering and stochastic sampling strategy during training lead to substantial improvement in classification performance. We demonstrate the efficacy of EXPERTNET on several clinical datasets, for predictive modeling tailored to subpopulations inferred from the data.

The clustering performance of EXPERTNET is comparable to that of other deep clustering methods and may be improved further. This can be explored in future work along with ways to combine hierarchical clustering algorithms with supervised models and applications in other domains.
[Zhang et al., 2016] Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In ICML, pages 612–621. PMLR, 2016.

[Zhao, 2015] J. et. al. Zhao. Stacked what-where auto-encoders. arXiv preprint arXiv:1506.02351, 2015.
A Analysis

A.1 General case

The following lemma is from [Bartlett and Mendelson, 2002; Mohri et al., 2012, Theorem 3.1]:

**Lemma 1.** Let \( \mathcal{F} \) be a set of maps \( x \mapsto f(x) \). Suppose that \( 0 \leq \ell(q, y) \leq \lambda \) for any \( q \in \{ f(x) : f \in \mathcal{F}, x \in X \} \) and \( y \in Y \). Then, for any \( \delta > 0 \), with probability at least \( 1 - \delta \) (over an i.i.d. draw of \( N \) i.i.d. samples \( \{(x_i, y_i)\}_{i=1}^N \)), the following holds: for all maps \( f \in \mathcal{F} \),

\[
\mathbb{E}_{x,y}[\ell(f(x), y)] \leq \frac{1}{N} \sum_{i=1}^N \ell(f(x_i), y_i) + 2 \hat{R}_N(\ell \circ \mathcal{F}) + 3\lambda \sqrt{\frac{\ln(1/\delta)}{2N}},
\]

where \( \hat{R}_N(\ell \circ \mathcal{F}) := \mathbb{E}_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \sigma_i \ell(f(x_i), y_i)] \) and \( \sigma_1, \ldots, \sigma_N \) are independent uniform random variables taking values in \( \{-1, 1\} \).

We can use Lemma 1 to prove the following:

**Theorem 2.** Let \( \mathcal{F}_j \) be a set of maps \( x \in \Omega_j \mapsto f_j(x) \). Let \( \mathcal{F} = \{ x \mapsto f(x) : f(x) = \sum_{j=1}^K 1 \{ x \in \Omega_j \} f_j(x), f_j \in \mathcal{F}_j \} \).

Suppose that \( 0 \leq \ell(q, y) \leq \lambda_j \) for any \( q \in \{ f(x) : f \in \mathcal{F}_j, x \in \Omega_j \} \) and \( y \in Y_j \). Then, for any \( \delta > 0 \), with probability at least \( 1 - \delta \), the following holds: for all maps \( f \in \mathcal{F} \),

\[
\mathbb{E}_{x,y}[\ell(f(x), y)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) \mathbb{E}_{x,y}[\ell(f_j(x), y) \mid x \in \Omega_j)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) \mathbb{E}_{x,y}[\ell(f_j(x), y) \mid x \in \Omega_j)]
\]

Since the conditional probability distribution is a probability distribution, we apply Lemma 1 to each term and take union bound to obtain the following: for any \( \delta > 0 \), with probability at least \( 1 - \delta \), for all \( j \in \{1, \ldots, k\} \) and all \( f_j \in \mathcal{F}_j \),

\[
\mathbb{E}_{x,y}[\ell(f_j(x), y) \mid x \in \Omega_j)] \leq \frac{1}{N_j} \sum_{i=1}^{N_j} \ell(f_j(x_i), y_i) + 2 \hat{R}_{N_j}(\ell \circ \mathcal{F}_j) + 3\lambda_j \sqrt{\frac{\ln(1/\delta)}{2N_j}}
\]

Thus, using (9), we sum up both sides with the factors \( \Pr(x \in \Omega_j) \) to yield:

\[
\mathbb{E}_{x,y}[\ell(f(x), y)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) \mathbb{E}_{x,y}[\ell(f_j(x), y) \mid x \in \Omega_j)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) \left( \frac{1}{N_j} \sum_{i=1}^{N_j} \ell(f(x_i), y_i) + 2 \hat{R}_{N_j}(\ell \circ \mathcal{F}_j) + 3\lambda_j \sqrt{\frac{\ln(1/\delta)}{2N_j}} \right).
\]

\[\square\]

A.2 Special case: multi-class classification

For multi-class classification problems with \( B \) classes and \( y \in \{1, \ldots, B\} \), define the margin loss as follows:

\[
\ell_\rho(f(x), y) = \ell_\rho^{(2)}(\ell_\rho^{(1)}(f(x), y))
\]

where

\[
\ell_\rho^{(1)}(f(x), y) = f(x)_y - \max_{y' \neq y} f(x)_{y'} \in \mathbb{R},
\]
Proof of Theorem 2.

Define the 0-1 loss as:

$$\ell^{(2)}_\rho(q) = \begin{cases} 
0 & \text{if } \rho \leq q \\
1 - q/\rho & \text{if } 0 \leq q \leq \rho \\
1 & \text{if } q \leq 0.
\end{cases}$$

Thus, using (13), we sum up both sides with the factors

$${\mathbb E}_{x,y}[\ell_0(f(x), y)] \leq \frac{1}{N} \sum_{i=1}^N \ell_\rho(f(x_i), y_i) + \frac{2B^2}{\rho} \hat{R}_N(F_B) + \left(1 + \frac{2B^2}{\rho}\right) \sqrt{\frac{\ln(2/\delta)}{2N}},$$

where $F_B = \{ f \mapsto f(x)_k : f \in F, k \in [B] \}$.

Proof. Using Theorem 8.1 [Mohri et al., 2012], we have that with probability at least $1 - \delta/2$,

$${\mathbb E}_{x,y}[\ell_0(f(x), y)] \leq \frac{1}{N} \sum_{i=1}^N \ell_\rho(f(x_i), y_i) + \frac{2B^2}{\rho} \hat{R}_N(F_B) + \sqrt{\frac{\ln(2/\delta)}{2N}},$$

Since changing one point in $S$ changes $\hat{R}_m(F_B)$ by at most $1/N$, McDiarmid’s inequality implies the statement of this lemma by taking union bound.

Theorem 3. Let $F_j$ be a set of maps $x \mapsto f_j(x)$. Let $F = \{ f \mapsto f(x) : f(x) = \sum_{j=1}^k \mathbbm{1}\{x \in \Omega_j\} f_j(x), f_j \in F_j \}$. Suppose that $0 < \ell(q, y) \leq \lambda_j$ for any $q \in \{ f(x) : f \in F_j, x \in \Omega_j \}$ and $y \in \mathcal{Y}_j$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds: for all maps $f \in F$,

$${\mathbb E}_{x,y}[\ell_0(f(x), y)] \leq \frac{1}{N_j} \sum_{i=1}^{N_j} \ell_\rho(f(x_i^j), y_i^j) + \frac{2B^2}{\rho} \hat{R}_{N_j}(F_{T,j}) + \left(1 + \frac{2B^2}{\rho}\right) \sqrt{\frac{\ln(2k/\delta)}{2N_j}},$$

where $\hat{R}_{N_j}(F_{T,j}) := {\mathbb E}_{x}[\sup_{f_j \in F_{T,j}} \frac{1}{N_j} \sum_{i=1}^{N_j} \sigma_i \ell(f_j(x_i^j), y_i^j)]$ and $\sigma_1, \ldots, \sigma_N$ are independent uniform random variables taking values in $\{-1, 1\}$. Here, $F_{T,j} = \{ f \mapsto f_j(x)_k : f \in F_j, k \in [B] \}$.

Proof of Theorem 2. We have that

$${\mathbb E}_{x,y}[\ell_0(f(x), y)] = \sum_{j=1}^k \Pr(x \in \Omega_j) {\mathbb E}_{x,y}[\ell_0(f(x), y) | x \in \Omega_j]$$

$$= \sum_{j=1}^k \Pr(x \in \Omega_j) {\mathbb E}_{x,y}[\ell_0(f_j(x), y) | x \in \Omega_j].$$

Since the conditional probability distribution is a probability distribution, we apply Lemma 1 to each term and take union bound to obtain the following: for any $\delta > 0$, with probability at least $1 - \delta$, for all $j \in \{1, \ldots, k\}$ and all $f_j \in F_j$,

$${\mathbb E}_{x,y}[\ell_0(f_j(x), y) | x \in \Omega_j] \leq \frac{1}{N_j} \sum_{i=1}^{N_j} \ell_\rho(f_j(x_i^j), y_i^j) + \frac{2B^2}{\rho} \hat{R}_{N_j}(F_{B,j}) + \left(1 + \frac{2B^2}{\rho}\right) \sqrt{\frac{\ln(2k/\delta)}{2N_j}}$$

Thus, using (13), we sum up both sides with the factors $\Pr(x \in \Omega_j)$ to yield:

$${\mathbb E}_{x,y}[\ell_0(f(x), y)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) {\mathbb E}_{x,y}[\ell_0(f_j(x), y) | x \in \Omega_j]$$

$$\leq \sum_{j=1}^k \Pr(x \in \Omega_j) \left( \frac{1}{N_j} \sum_{i=1}^{N_j} \ell_\rho(f(x_i^j), y_i^j) + \frac{2B^2}{\rho} \hat{R}_{N_j}(F_{B,j}) + \left(1 + \frac{2B^2}{\rho}\right) \sqrt{\frac{\ln(2k/\delta)}{2N_j}} \right).$$
A.3 Special case: deep neural networks with binary classification

Note that for any $\rho > 0$, the margin loss $\ell_\rho(f(x), y)$ is an upper bound on the 0-1 loss: i.e., $\ell_\rho(f(x), y) \geq \ell_{01}(f(x), y)$.

The following lemma is from [Mohri et al., 2012, Theorem 4.4]:

**Lemma 3.** Let $F$ be a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an i.i.d. draw of $m$ i.i.d. test samples $(x_i, y_i)_{i=1}^m$, the following holds: for all maps $f \in F$,

$$
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] \leq \frac{1}{m} \sum_{i=1}^m \ell_\rho(f(x_i), y_i) + \frac{2}{\rho} \hat{R}_m(F) + 3 \sqrt{\frac{\ln(2/\delta)}{2m}}.
$$

**Theorem 4.** Let $F_j$ be a set of maps $x \in \Omega_j \mapsto f_j(x)$. Let $F = \{x \mapsto f(x) : f(x) = \sum_{j=1}^K \mathbb{1}(x \in \Omega_j) f_j(x), f_j \in F_j\}$. Suppose that $0 \leq \ell(q, y) \leq \lambda_j$ for any $q \in \{f(x) : f \in F_j, x \in \Omega_j\}$ and $y \in \mathcal{Y}_j$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds: for all maps $f \in F$,

$$
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] \leq \sum_{j=1}^k \Pr(x \in \Omega_j) \left( \frac{1}{N_j} \sum_{i=1}^{N_j} \ell_\rho(f_j(x_i), y_i) + \frac{2}{\rho} \hat{R}_{N_j}(F_j) + 3 \sqrt{\frac{\ln(2/\delta)}{2N_j}} \right),
$$

where $\hat{R}_{N_j}(F_j) := \mathbb{E}_x[\sup_{f_j \in F_j} \frac{1}{N_j} \sum_{i=1}^{N_j} \sigma_i(f_j(x_i), y_i)]$ and $\sigma_1, \ldots, \sigma_N$ are independent uniform random variables taking values in $\{-1, 1\}$.

**Proof of Theorem 4.** We have that

$$
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] = \sum_{j=1}^K \Pr(x \in \Omega_j) \mathbb{E}_{x,y}[\ell_{01}(f(x), y) \mid x \in \Omega_j)]
$$

(17)

Since the conditional probability distribution is a probability distribution, we apply Lemma 3 to each term and take union bound to obtain the following: for any $\delta > 0$, with probability at least $1 - \delta$, for all $j \in \{1, \ldots, k\}$ and all $f_j \in F_j$,

$$
\mathbb{E}_{x,y}[\ell_{01}(f_j(x), y) \mid x \in \Omega_j)] \leq \frac{1}{N_j} \sum_{i=1}^{N_j} \ell_\rho(f_j(x_i), y_i) + \frac{2}{\rho} \hat{R}_{N_j}(F_j) + 3 \sqrt{\frac{\ln(2/\delta)}{2N_j}}
$$

(18)

Thus, using (17), we sum up both sides with the factors $\Pr(x \in \Omega_j)$ to yield:

$$
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] = \sum_{j=1}^K \Pr(x \in \Omega_j) \mathbb{E}_{x,y}[\ell_{01}(f_j(x), y) \mid x \in \Omega_j)]
$$

(19)

We are now ready to prove Theorem 1.

**Proof of Theorem 1.** Using Theorem 4 with $\Pr(x \in \Omega_j) = 1/k$ and $m_j = N/k$, for $f \in F^k$. □
\[
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] \leq \frac{1}{k} \sum_{j=1}^{k} \left( \frac{1}{N/k} \sum_{i=1}^{N/k} \ell_{\rho}(f(x_i^j), y_i^j) + 2\rho^{-1} \hat{R}_{N/k}({\mathcal{F}}_j) + 3\sqrt{\frac{\ln(2/\delta)}{2(N/k)}} \right)
\]

\[
= \frac{1}{k} \sum_{j=1}^{k} \left( \frac{k}{N} \sum_{i=1}^{N/k} \ell_{\rho}(f(x_i^j), y_i^j) + 2\rho^{-1} \hat{R}_{N/k}({\mathcal{F}}_j) + 3\sqrt{\frac{k \ln(2k/\delta)}{2N}} \right)
\]

\[
= \frac{1}{N} \sum_{j=1}^{N/k} \ell_{\rho}(f(x_i^j), y_i^j) + 2\rho^{-1} \sum_{j=1}^{k} \hat{R}_{N/k}({\mathcal{F}}_j) + 3\sqrt{\frac{k \ln(2k/\delta)}{2N}}
\]

Here, using Theorem 1 of [Golowich et al., 2018], we have that

\[
\hat{R}_{N/k}({\mathcal{F}}_j) \leq B_j \sqrt{\hat{k}} \left( \sqrt{2 \log(2)(L + Q)} + 1 \right) \left( \prod_{l=1}^{L} M^l_j \right) \left( \prod_{l=1}^{Q} M^0_l \right).
\]

Thus, for any \( \delta > 0 \), with probability at least \( 1 - \delta \) over an i.i.d. draw of \( N \) i.i.d. test samples \( ((x_i, y_i))_{i=1}^{N} \), the following holds: for all maps \( f \in \mathcal{F}^K \),

\[
\mathbb{E}_{x,y}[\ell_{01}(f(x), y)] - \mathbb{E}_{x,y}[\hat{\ell}_{\rho}(f(x), y)] 
\leq 2\rho^{-1} \sum_{j=1}^{k} B_j \sqrt{\hat{k}} \left( \sqrt{2 \log(2)(L + Q)} + 1 \right) \left( \prod_{l=1}^{L} M^l_j \right) \left( \prod_{l=1}^{Q} M^0_l \right) + 3\sqrt{\frac{k \ln(2k/\delta)}{2N}}
\]

\[
\leq \frac{2\rho^{-1} \left( \sqrt{2 \log(2)(L + Q)} + 1 \right) \left( \prod_{l=1}^{Q} M^0_l \right) \left( \sum_{j=1}^{k} B_j \prod_{l=1}^{L} M^l_j \right)}{\sqrt{kN}} + 3\sqrt{\frac{k \ln(2k/\delta)}{2N}}
\]

By taking union bound over \( M^l_j \), it is straightforward to replace \( \prod_{l=1}^{L} M^l_j \) by \( \prod_{l=1}^{L} \|W^l_i\|_F \) in the bound of Theorem 1 with additional factors that is not dominant and hidden in the \( \tilde{O} \) notation (see the proof of Lemma A.9 of [Bartlett et al., 2017] or the proof of Theorem 2 of [Koltchinskii and Panchenko, 2002]).
## B Extended Results

### Table 4: Silhouette Scores. Best values in bold.

| Dataset | k | KMeans | DCN | DEC | DMNN | EXPERTNET |
|---------|---|--------|-----|-----|------|-----------|
| CIC     | 2 | 0.54   | NA  | 0.51| 0.125| 0.358     |
|         | 3 | 0.297  | NA  | 0.354| 0.092| 0.305     |
|         | 4 | 0.168  | NA  | 0.326| 0.085| 0.231     |
| Sepsis  | 2 | 0.588  | 0.253| 0.325| 0.250| 0.357     |
|         | 3 | 0.277  | 0.242| 0.406| 0.104| 0.349     |
|         | 4 | 0.212  | NA  | 0.307| 0.099| 0.248     |
| AKI     | 2 | 0.401  | 0.596| 0.769| 0.259| 0.638     |
|         | 3 | 0.459  | 0.782| 0.828| 0.285| 0.611     |
|         | 4 | 0.459  | 0.585| 0.622| 0.278| 0.53      |
| ARDS    | 2 | 0.369  | 0.65 | 0.746| 0.295| 0.484     |
|         | 3 | 0.384  | NA  | 0.553| 0.319| 0.628     |
|         | 4 | 0.4    | NA  | 0.856| 0.301| 0.76      |
| WID-M   | 2 | 0.346  | 0.797| 0.423| 0.202| 0.393     |
|         | 3 | 0.189  | NA  | 0.45 | 0.157| 0.282     |
|         | 4 | 0.136  | NA  | 0.544| 0.152| 0.197     |
| Diabetes| 2 | 0.249  | 0.448| 0.378| 0.156| 0.129     |
|         | 3 | 0.201  | 0.181| 0.503| 0.167| 0.154     |
|         | 4 | 0.157  | 0.088| 0.423| 0.183| 0.24      |
| CIC,LOS | 2 | 0.547  | 0.582| 0.546| -    | 0.231     |
|         | 3 | 0.306  | 0.372| 0.447| -    | 0.099     |
|         | 4 | 0.162  | NA  | 0.349| -    | 0.129     |
Table 5: F1 Scores. Best values in bold.

C Extended Ablation Analysis

![F1/AUC vs. β](image1)

![F1/AUC vs. γ](image2)

![SIL](image3)

Figure 3: Ablation study on Sepsis dataset. TT represents that stochastic sampling is turned on in both training and testing phase. Similarly for other combinations. Note that stochastic sampling turned on in the training phase has a strong effect on performance.

D Data Preprocessing and Feature Extraction

We evaluate the results on 6 real world clinical datasets out of which 3 are derived from the MIMIC III dataset derived from Beth Israel Deaconess Hospital. CIC dataset is derived from 2012 Physionet challenge [Silva, 2012] of predicting in-hospital
mortality of intensive care units. The Sepsis dataset is derived from the 2019 Physionet challenge of sepsis onset prediction. We manually extract the Acute Kidney Failure (AKI) and Acute Respiratory Distress Syndrome (ARDS) datasets from the larger MIMIC III dataset. WID Mortality dataset is extracted from the 2020 Women In Data Science challenge to predict patient mortality. The Diabetes Readmission prediction dataset consists of 100000 records of patients. This is a multiclass version of the CIC dataset where the Length of Stay (LOS) feature is discretized into 3 classes using quartiles. The task is to predict the LOS of ICU patients.

- **CIC**: The CIC dataset is derived from the 2012 Physionet challenge [Silva, 2012] of predicting in-hospital mortality of intensive care units (ICU) patients at the end of their hospital stay. The data has time-series records, comprising various physiological parameters of 120000 patient ICU stays. We follow the data processing scheme of [Johnson, 2012] (the top ranked team in the competition) to obtain static 117-dimensional features for each patient.

- **Sepsis**: The Sepsis dataset is derived from the 2019 Physionet challenge of sepsis onset prediction. The dataset has time-series records, comprising various physiological parameters of ~ 40000 patients. We follow the data processing scheme of the competition winners [Morrill, 2019] to obtain static 89 dimensional features for each patient.

- **AKI and ARDS**: We manually extract the Acute Kidney Failure (AKI) and Acute Respiratory Distress Syndrome (ARDS) datasets from the larger MIMIC III dataset. We follow the KDIGO criteria to determine kidney failure onset time and the Berlin criteria for ARDS. The challenge is to predict kidney failure onset ahead of time. Similar to the above datasets, we derive static vectors from the time series data.

- **WID Mortality**: This dataset is extracted from the 2020 Women In Data Science challenge where the objective is to create a model that uses data from the first 24 hours of intensive care to predict patient survival. We perform standard data cleaning procedures before using the dataset.

- **Diabetes**: The Diabetes Readmission prediction dataset consists of 100000 records of patients from 130 hospitals in the US from the year 1998 to 2008.

- **CIC-LOS**: This is a multiclass version of the CIC dataset where the Length of Stay (LOS) feature is discretized into 3 classes using quartiles. The task is to predict the LOS of ICU patients.

### E ExpertNet Optimization Details

#### E.1 Updating ExpertNet parameters

Optimizing \( (U, V) \) is similar to training an SAE - but with the additional loss terms \( L_c, L_a \) and \( L_{bal} \). We code our algorithm in PyTorch which allows us to easily backpropagate all the losses simultaneously. To implement SGD for updating the network parameters, we look at the problem w.r.t. the incoming data \( x_i \):

\[
\min_{U, W} L^I = \ell(g(f(x_i)), x_i) + \beta \cdot \text{KL}(P, Q) \\
+ \gamma \cdot \sum_{j=1}^{k} q_{i,j} \cdot \text{CE}(y_i, h_j(x_i; V_j)) + \delta \cdot L_{bal}
\]

The gradient of the above function over the network parameters is easily computable. Let \( \mathcal{J} = \{U, V, W\} \) be the collection of network parameters, then for a fixed target distribution \( P \), the gradients of \( L_c \) w.r.t. embedded point \( z_i \) and cluster center \( \mu_j \) can be computed as:

\[
\frac{\partial L_c}{\partial z_i} = 2 \sum_{j=1}^{k} \left(1 + \|z_i - \mu_j\|^2\right)^{-1} (p_{i,j} - q_{i,j})(z_i - \mu_j)
\]

\[
\frac{\partial L_c}{\partial \mu_j} = 2 \sum_{i} \left(1 + \|z_i - \mu_j\|^2\right)^{-1} (q_{i,j} - p_{i,j})(z_i - \mu_j)
\]

The above derivations are from [Xie et al., 2016]. We leverage the power of automatic differentiation to calculate the gradients of \( L_{bal} \) and \( L_a \) during execution. Then given a mini batch with \( m \) samples and learning rate \( \lambda, \mu_j \) is updated by:

\[
\mu_j = \mu_j - \frac{\tau}{m} \sum_{i=1}^{m} \frac{\partial L_c}{\partial \mu_j}
\]

(20)

The decoder’s weights are updated by:

\[
V = V - \frac{\tau}{m} \sum_{i=1}^{m} \frac{\partial L_r}{\partial V}
\]

(21)
and the encoder’s weights are updated by:

$$\mathcal{U} = \mathcal{U} - \frac{\tau}{m} \sum_{i=1}^{m} \left( \frac{\partial L_r}{\partial \mathcal{U}} + \frac{\partial L_c}{\partial \mathcal{U}} + \frac{\partial L_s}{\partial \mathcal{U}} + \frac{\partial L_{bal}}{\partial \mathcal{U}} \right)$$

(22)

where $\tau$ is the diminishing learning rate.

F Case Study: Additional Analysis

| Cluster 1       | Cluster 2              | Cluster 3     |
|-----------------|------------------------|---------------|
| SAPS-I          | Length_of_stay         | SAPS-I        |
| SOFA            | CCU                    | SOFA          |
| GCS_first       | CSRU                   | Length_of_stay|
| GCS_lowest      | DiasABP_first          | Weight        |
| CSRU            | GCS_first              | CSRU          |
| Creatinine_last | Creatinine_last        | MechVentLast8Hour |
| Creatinine_last | Glucose_first          | DiasABP_first |
| GCS_last        | HR_first               | Glucose_first |
| BUN_last        | MAP_first              | HR_first      |
| Creatinine_first| NIDiasABP_first        | MAP_first     |
| Lactate_first   | NIMAP_first            | NIDiasABP_first|

Table 6: Cluster wise features sorted by HTFD values.

We study the clusters found by EXPERTNET by analyzing the HTFD metric in table 6. SAPS-I and SOFA have high HTFD in clusters 1 and 3. This indicates that the two clusters have different distributions of SAPS-I and SOFA. It (SOFA) is also an important feature for cluster 3’s risk model (Table 3) but in distribution, it is similar to SOFA values of clusters 2 but not 1. GCS_Last (Glasgow Coma Score) is an important feature for all the three risk models but not significantly different across the three clusters (cluster averages 12.744, 14.489, 10.226). Normal GCS is 15.