UnfoldML: A Cost-Aware 2-D Dynamic Prediction Pipeline for Multi-Stage Classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

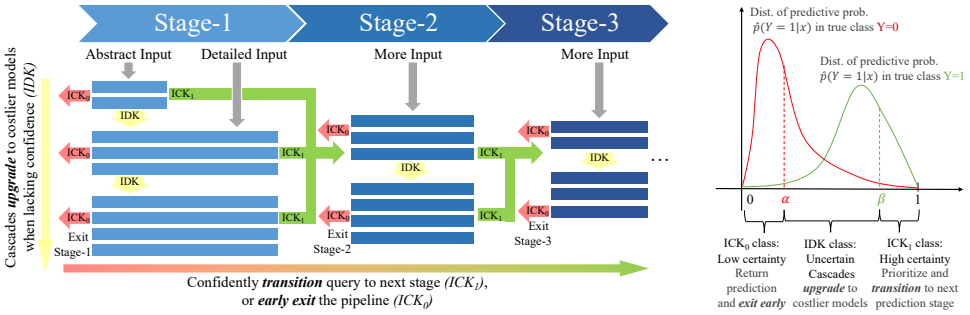
Machine Learning (ML) research has focused on maximizing the accuracy of predictive tasks. ML models, however, are increasingly more complex, resource intensive, and costlier to deploy in resource constrained environments. These issues are exacerbated for prediction tasks with sequential classification on progressively transitioned stages with "happens-before" relation between them.We argue that it is possible to "unfold" a monolithic single multi-class classifier, typically trained for all stages using all data, into a series of single-stage classifiers. Each single-stage classifier can be cascaded gradually from cheaper to more expensive binary classifiers that are trained using only the necessary data modalities or features required for that stage. UnfoldML is a cost-aware 2-D dynamic prediction pipeline for multi-stage classification that achieves this enabling (1) navigation of the accuracy/cost tradeoff space, (2) reducing the spatio-temporal cost of inference by orders of magnitude, and (3) early prediction on proceeding stages. UnfoldML achieves orders of magnitude better cost in clinical settings, while detecting multi-stage disease development in real time. It achieves within 0.1% accuracy from the highest-performing multi-class baseline, while saving close to 20X on spatio-temporal cost of inference and earlier (3.5hrs) disease onset prediction. We also show that UnfoldML generalizes to image classification, where it can predict different level of labels (from coarse to fine) given different level of abstractions of a image, saving close to a 5X cost with as little as 0.4% accuracy reduction. UnfoldML improves the whole frontier of optimality in the accuracy/cost tradeoff space, achieving the highest area under the curve.

1 Introduction

Machine Learning (ML) research has mostly focused on improving prediction accuracy for classification tasks, such as image classification (Foret et al., 2020; Xie et al., 2017), disease risk prediction (Feng et al., 2020; Xu et al., 2018), pedestrian detection (Zhang et al., 2018; Cai et al., 2015), etc. The understandable drive for high accuracy has often resulted in deeper, complex neural networks, which can incur high memory (*spatio* cost) and high latency (*temporal* cost) at inference time. However, the deployment of ML applications must be cost-aware. Run time environment like mobile devices or bedside-patient monitors are commonly resource constrained, and applications that can be offloaded to cloud computing always aim for a reduced cloud bill. In this paper, we focus on developing a pipeline that can balance between high prediction accuracy and low *spatio-temporal* cost in deploying ML classification models.

We consider a scenario of deploying ML classifiers in a multi-stage classification task where one predicted class can progressively transform to a next stage of classes, characterized as "happens-before" relationship between classes. This task is commonly observed in many real-world applications. For instance in clinical settings, disease progression is often identified by a series of stage transitions (Sperling et al., 2011; Singer et al., 2016). Detection on early stage of the disease allows doctors to take

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) The 2-D query propagation mechanism in `UnfoldML`

(b) IDK and ICK classes

Figure 1: A 2-D query propagation in `UnfoldML`: Queries that are in confidently low risk will return ICK_0 and be monitored by cheaper models; queries that are hard to predict will return IDK and be advanced to costlier but more confident models; queries that are in confidently high risk will return ICK_1 and be transited to next stage for making earlier prediction on more severe conditions.

appropriate actions in time before it enters into late severe stages. In image classifications, recognition from super classes to sub classes in a coarse-to-fine manner has shown improved classification performance (Dutt et al., 2017; Lei et al., 2017). In all these applications, general multi-class classifiers (Liu et al., 2019; Fagerström et al., 2019; Foret et al., 2020) have been developed by treating all the stages as multi classes and achieved state-of-the-art prediction performance, but at significantly high spatio-temporal cost. Prior work (Wang et al., 2017) trades off accuracy for low cost but still ignores the key relationship between the classes, so it fails to find the optimal trade-off.

We propose `UnfoldML`: a cost-aware prediction pipeline that leverages the "happens-before" property in multi-stage classification. It "unfolds" a monolithic multi-class classifier into a series of single-stage classifiers, reducing its deployment cost. Each single-stage classifier is then cascaded gradually from cheaper to more expensive binary classifiers, further reducing the cost by dynamically selecting an appropriate classifier for an input query. Figure 1 summarizes the two dimensional (2-D) query propagation *mechanism* designed in `UnfoldML`: Horizontally it allows a query to transition through multiple stages, and vertically it allows the query to progressively upgrade to costlier models constrained by the pre-specified budget limit. It computes a classifier's prediction confidence on a query then directs the query through one the following three gates: 1) "*I confidently know NO*" (ICK_0), which rejects the current query and early exits from the pipeline (exit); 2) "*I don't know*" (IDK), which upgrades the query to a higher accuracy but costlier model, producing a more confident result within the budget (vertical cascading); and 3) "*I confidently know YES*" (ICK_1), which transitions the query to the next stage of the prediction task (horizontal forwarding). The design of ICK_0 gate allows queries to early exit from the `UnfoldML` so it reduces the overall spatio-temporal cost of the pipeline. The ICK_1 gate allows queries to faster transition to next stages so it enables early prediction on late stages, which can be critical in clinical settings (Reyna et al., 2019). Overall, the combined 2-D propagation mechanism uniquely enables the navigation of the cost/accuracy tradeoff space for searching an optimal set of *policies* for dynamic model selection at inference time.

We also propose two training algorithms for learning the optimal *policies* for the designed 2-D query propagation in `UnfoldML`. The key idea of the training algorithms is to learn the optimal thresholds on the gating functions defined for ICK_0, IDK and ICK_1. The first proposed *hard-gating* algorithm assumes the gating functions to be step functions parameterized by deterministic confidence thresholds. To find the optimal thresholds, it performs bottom-up grid search over a topologically-sorted list of all the models and identifies the thresholds to minimize the prediction loss, while following a cost constraint. The secondly proposed *soft-gating* algorithm defines the gating functions to be probabilistic activation functions. It follows a Mixture-of-Expert (MoE) framework to adaptively determine a models' confidence threshold given all the other model's confidence in predicting the same query. To obviate the cost of running all models in MoE selection, we further propose a Dirichlet Knowledge Distillation (DKD) to run only a cheap multi-label classifier that is trained for distilling the Bayesian predictive uncertainties of all models.

We summarize the contributions of this paper as follows:

- We design a novel 2-D query propagation pipeline that "unfolds" multi-stage prediction workflows by leveraging the "happens-before" relationship between the stages, and achieves a lower-cost

prediction pipeline with minimal accuracy degradation.

• We propose two learning algorithms to sufficiently navigate the cost/accuracy tradeoff space and search an optimal set of policies for the designed 2-D query propagation.

• We apply the proposed pipeline to two real-world applications and demonstrate it reduces the spatio-temporal cost of inference by orders of magnitude.

2 Related Work

The most relevant work to our proposed method is the one-step IDK cascade (Wang et al., 2017), which incorporates prior work of "*I don't know*" (IDK) classes (Trappenberg and Back, 2000; Khani et al., 2016) into cascade construction and introduce a latency-aware objective into the construction comparing with previous cascaded prediction frameworks (Rowley et al., 1998; Viola and Jones, 2004; Angelova et al., 2015). Another group of work focus on the problem of feature selection assuming each feature can be acquired for a cost. They train a cascade of classifiers for optimizing the trade-off between the expected classification error and the feature cost. Early solution (Raykar et al., 2010) limits the cascade to a family of linear discriminating functions. Cai et al. (2015) applies boosting method for cascading a set of weak learners. Recent methods (Trapeznikov and Saligrama, 2013; Clertant et al., 2019; Janisch et al., 2019) develop POMDP-based frameworks and incorporate deep Q-learning in training the cascades. In contrast to all of the above work that are only 1-D pipelines for one-step prediction task (can be multi-class classifications), our method extends to a 2-D pipeline that can dynamically forward examples to next steps after they are confidently predicted as passed on the current step. Further, we also develop a more efficient pipeline framework based on Mixture-of-Experts (MoE) modeling and knowledge distillation, which can apply gradient decent algorithms for learning the parameters efficiently.

The idea of MoE was originally introduced by Jacobs et al. ([n. d.]), for partitioning the training data and feeding them into separate neural networks during the learning process. This gate decision design is applied into many domains such as language modeling (Ma et al., 2018), video captioning (Wang et al., 2019), multi-tasking learning (Ma et al., 2018). It is also used in network architecture searching (Eigen et al., 2013) by setting gate activation on network layers. Sparse gates are introduced in MoE so that it can efficiently select from thousands of sub-networks (Shazeer et al., 2017) as well as increases the representation power of large convolutional networks by only using a shallow embedding network to produce the mixture weights (Wang et al., 2020). We incorporate the idea of sparsely gated MoE (Shazeer et al., 2017; Wang et al., 2020) into our prediction framework, and design a soft-gating training algorithm by using ReLU as the sparse gating function and imposing L1-norm regularization on the gating weights for further sparsity.

Confidence criterion has been incorporated into active learning by Li and Sethi (2006) and then extended by Zhu et al. (2010). Lei (2014) proposed confidence based classifiers that identifies the confident region (like ICK class) and uncertain region (like IDK class) in predictions. Confidence are also introduced into word embedding (Vilnis and McCallum, 2015; Athiwaratkun and Wilson, 2018) and graph representations (Orbach and Crammer, 2012; Vashishth et al., 2019). Our method posits thresholds on prediction confidence for activating the gates in pipeline expansion. Bayesian Prior Networks (BPNs) (Malinin and Gales, 2018) have been proposed to estimate the uncertainty distribution in model predictions, which is more computationally efficient than traditional Bayesian approaches (MacKay, 1992; Mackay, 1992; Hinton and Van Camp, 1993). We propose Dirichlet Knowledge Distillation (DKD) based on BPNs for distilling prediction uncertainty in large models so that we only need to run a low-cost multi-head model for producing the weights in MoE efficiently.

3 Multi-Stage Dynamic Prediction Pipeline

We introduce a dynamic 2-D prediction pipeline `UnfoldML`, which learns the optimal policy for making "*I confidently know*" (ICK) predictions on sequential multi-stage classification tasks. An optimal policy will effectively trade off prediction accuracy against spatio-temporal costs in order to maximize the overall system accuracy's AUC while staying under user-imposed cost constraints.

3.1 Problem Formulation

Given $\boldsymbol{x} \in \mathcal{X}$ at time t, a multi-stage pipeline decides whether the individual should maintain at the current stage s or progress into the next stage $s+1$ for time $t+1$. If there are a total number of S stages that need to be detected, we train K number of models m for each specific stage s to form a model zoo $\mathcal{M} = \{\{m_{11}, \cdots, m_{1K_1}\}, \cdots, \{m_{S1}, ..., m_{SK_S}\}\}$. We measure each model's spatio-temporal cost by multiplying the device cost per unit time with the serving time per prediction stage, denoted as $cost(m_{sk})$.

To optimize the limited system resources, we design `UnfoldML` in the following way: (1) start with the simplest model for prediction of the initial stage on incoming data, and (2) **upgrade** to costlier models on those samples where "*I don't know*" (IDK), or (3) **transition** to the next stage of the pipeline samples where "*I confidently know YES*" (ICK_1) that we have correctly identified the sample at the current stage, otherwise (4) **exit** the pipeline for those "*I confidently know NO*" (ICK_0) samples that have been identified and discarded. Figure 1 (a) demonstrates the proposed 2-D architecture of `UnfoldML`. The central problem of `UnfoldML` is learning the optimal policy for each of the three classes of gating functions with the objective of maximizing high system-wide accuracy while minimizing the prediction cost.

We formulate `UnfoldML` as a decision rule mapping function $m^{casc} : \mathcal{X} \times \mathcal{M} \rightarrow \mathcal{M}$, which takes example $\boldsymbol{x}_t$ coming at time t and the current model choice m_{sk} as an input to determine whether the model for the query should (1) *upgrade* 'vertically' to costlier models in the same stage, (2) *transition* 'horizontally' to the next stage, or (3) return as rejected and *exit* the pipeline. These decisions rules can be realized by two groups of parameters: a confidence criterion $q : \mathcal{X} \times \mathcal{M} \rightarrow [0, 1]$ that measures the confidence score of a model's prediction on a data example, and two gate functions $G^{IDK}, G^{ICK_1} : [0, 1] \times \{\text{True}, \text{False}\}$ that are applied on to the confidence score per each prediction made by model m_{sk}. The two exclusive gates IDK and ICK_1 each respectively decides if the current prediction belongs to either an IDK class such that (s.t.) the system will *upgrade* the query to a costlier model while remaining within the user-defined cost budget, or an ICK_1 class s.t. the system will *transition* the query to the next stage in the pipeline. The third gate G^{ICK_0} is determined by $\neg G^{IDK} \wedge \neg G^{ICK_1}$. We formalize the decision rule used at each prediction stage as follows

$$m^{casc}(\boldsymbol{x}_t, m_{sk}; q, G) = \begin{cases} m_{s(k+1)}, G^{IDK} \wedge \neg G^{ICK_1}\big(q_{sk}(\boldsymbol{x}_t)\big), \\ m_{(s+1)1}, \neg G^{IDK} \wedge G^{ICK_1}\big(q_{sk}(\boldsymbol{x}_t)\big), \\ m_{sk}, \quad \neg G^{IDK} \wedge \neg G^{ICK_1}\big(q_{sk}(\boldsymbol{x}_t)\big) \end{cases}$$

where $q_{sk}(\boldsymbol{x}_t)$ is a short notation for $q(\boldsymbol{x}_t, m_{sk})$ measuring the confidence of model m_{sk}'s prediction on data $\boldsymbol{x}_t$. The goal of configuring an optimal pipeline given a restricted computation resource can be formalized as the following optimization problem, which minimizes an end-to-end prediction loss $\mathcal{L}$ of m^{casc} on an given observed data set $\mathcal{D}$ with respect to (w.r.t) the gate specifications G s.t. the total cost of running the pipeline remains within the user-specified cost-constraints for the system s.t. $\min_G \mathcal{L}\big(m^{\text{casc}}; \mathcal{D}\big) \quad \text{s.t. } cost(m^{\text{casc}}; \mathcal{D}) \leq c$, where G consists of two gate functions G^{IDK} and G^{ICK_1}. Next, we introduce two training algorithms for learning the optimal gate functions in `UnfoldML`.

3.2 Gate Parameters Learning

Given a training data set $\mathcal{D}$, which does not include any data that was used in the training of the models in our model zoo: $\mathcal{D} = \{\big(\boldsymbol{x}_i, (y_i^1, t_i^1), \cdots, (y_i^S, t_i^S)\big)\}_{i=1}^N$, where (1) $\boldsymbol{x}_i = (\boldsymbol{x}_{i1}, \cdots, \boldsymbol{x}_{iT_i})$ is an input sequence observed for individual i, (2) $y_i^s \in \{0, 1\}$ indicates whether the example entered to stage-s, and if *yes*, we use (3) $t_i^s \in \emptyset \cup [1, T_i]$ to determine when it entered. We first partition the multi-stage data into S one-stage data sets: $\mathcal{D}^s = \{(\boldsymbol{x}_{i[t_i^{s-1}:t_i^s]}, y_i^s == 1)\} \cup \{(\boldsymbol{x}_{i[t_i^{s-1}:T_i]}, y_i^s == 0)\}$, then divide the learning of IDK and ICK gate parameters into two separable sub-problems: **Sub-Objective 1:** $\min_{G_s^{IDK}} \mathcal{L}^s\big(m_s^{\text{casc}}; \mathcal{D}^s\big)$ s.t. $cost(m_s^{\text{casc}}; \mathcal{D}^s) \leq c_s, s = 1, \cdots, S$ and **Sub-Objective 2:** $\min_{G^{ICK_1}} \mathcal{L}\big(m^{\text{casc}}; \mathcal{D}, G^{IDK*}\big)$ where $\mathcal{L}^s$ is the one-stage prediction loss on data $\mathcal{D}^s$, and c_s is the cost budget that is pre-allocated for stage-s satisfying $\sum_s c_s = c$. Decomposing the end-to-end optimization problem in Soft-Gating into two sub-problems Hard-Gating allows us to parallelize the training process and efficiently learn each stage's optimal IDK gate parameters by solving Sub-Objective 1. After, we learn the optimal ICK_1 gate parameters by fixing the IDK parameters learned previously and solving Sub-Objective 2.

3.2.1 Hard-gating Training Algorithm

In this algorithm, we assume G^{IDK} to be a *hard-gating* function that is parameterized by cutoff α_{sk} s.t. the gate is only activated if the level of confidence in the prediction is below a threshold. Hard-gating is defined as $G^{IDK}(q_{sk}(\boldsymbol{x}_t)) = \mathbb{I}(q_{sk}(\boldsymbol{x}_t) < \alpha_{sk})$, where $\mathbb{I}(\cdot)$ is an indicator function. Based on the Problem Formulation, a model m_{sk} at stage s can only be activated if $\mathbb{I}(q_{sk}(\boldsymbol{x}_t) \geq \alpha_{sk}) \wedge_{j=1}^{k-1} \mathbb{I}(q_{sj}(\boldsymbol{x}_t) < \alpha_{sj}) \equiv 1$. Now we write the conditional probability of being at stage-s given input $\boldsymbol{x}_t$ as $\Pr(y^s = 1|\boldsymbol{x}_t; m_s^{\text{casc}}) = \sum_{k=1}^{K_s} \mathbb{I}(q_{sk}(\boldsymbol{x}_t) \geq \alpha_{sk}) \cdot \prod_{j=1}^{k-1} \mathbb{I}(q_{sj}(\boldsymbol{x}_t) < \alpha_{sj}) \cdot m_{sk}(\boldsymbol{x}_t)$, where $m_{sk}(\boldsymbol{x}_t) = \Pr\big(y^s = 1 \mid \boldsymbol{x_t}; m_{sk}\big)$. The one-stage loss function in Hard-Gating is defined as the negative log-likelihood loss $\mathcal{L}_{\text{nll}}^s\big(m_s^{\text{casc}}; \mathcal{D}^s\big) = -\sum_{i=1}^{N_s} \sum_{t=1}^{T_i} y_{it}^s \cdot \log p_{it}^s + (1-y_{it}^s) \cdot \log(1-p_{it}^s)$, where $p_{it}^s = \Pr(y^s = 1|\boldsymbol{x}_{it}; m_s^{\text{casc}})$, $y_{it}^s = 1$ only if $y_i^s = 1$ and $t \in [t_i^1 - \delta_t, t_i^1]$ for some δ_t time-steps we wish to early detect the next stage $s+1$.

Algorithm 1 in Appendix 5.1 describes a bottom-up grid search algorithm for for learning *hard-gating* parameters. We sort the model list $\mathcal{M}^s = \{m_{s1}, \cdots m_{sK_s}\}$ in a monotonically increasing order w.r.t AUC and cost which discards any sub-optimal models. For any stage s, Algorithm 1 starts by assigning all the N^s samples for that stage to the first model m_{s1}, and performing a grid search on the gate parameters α_{sk}'s for level $k = 1$ to K_s. It gradually assigns IDK samples to the next level's model in the list until the cost exceeds the user-defined budget c_s for that stage. In each iteration of searching the cutoff α_{sk}, we set an upper bound $maxA$ on the maximum searching value to avoid over-upgrading. Without setting this bound, the algorithm could overfit and put all the samples into the IDK class, then assign them to the next level's model. This consumes the cost quota quickly, and prevents high IDK samples from exploring larger models in the list.

3.2.2 Soft-gating Training Algorithm

In contrast to grid search on the thresholds, we propose a *soft-gating* algorithm formulating an objective function that can be efficiently solved using gradient descent algorithms. In this algorithm, we define the gate function G^{IDK} to be a ReLU function parameterized by a pair of coefficients (a_{sk}, b_{sk}) s.t. the gate is only activated if the linear product $a_{sk} \cdot q_{sk}(\boldsymbol{x}_t) - b_{sk} > 0$. Formally we define Soft-gating as $G^{IDK}(q_{sk}(\boldsymbol{x}_t)) = \text{ReLU}(a_{sk} \cdot q_{sk}(\boldsymbol{x}_t) - b_{sk})$. Therefore the conditional probability of being at the stage-s given input $\boldsymbol{x}_t$ is defined as a mixture of the K_s models available for stage-s prediction: $\Pr(y^s = 1|x_t; m^{\text{casc}}) = \sum_{k=1}^{K_s} \frac{G^{IDK}(q_{sk}(\boldsymbol{x}_t)) \cdot m_{sk}(\boldsymbol{x}_t)}{\sum_{j=1}^{K_s} G^{IDK}(q_{sj}(\boldsymbol{x}_t))}$. Now, the negative log-likelihood loss $\mathcal{L}_{\text{nll}}^s$ becomes solvable using gradient descent algorithms. However, the normalization term in the mixture of experts requires running all the candidates models in the zoo, which conflicts with our cost-saving goal. Therefore, we propose using a Dirichlet Knowledge Distillation (DKD) for training a small surrogate model for each stage to quantify the prediction confidence for each model in the zoo. Then we replace the $q_{sj}(\boldsymbol{x}_t)$'s with their estimations $\hat{q}_{sj}(\boldsymbol{x}_t)$'s when selecting the expert models to infer with when real predictions are made. The smaller distilled model only needs to be run once per query, requiring much less cost than running all the models. In our experiment, we utilize the first model m_{s1} from each stage, take the embedding of $h_{s1}(\boldsymbol{x}_t)$ prior to the last activation layers in m_{s1} and feed it into a 4-layer K_s-head Multi-Layer Perceptron (MLP) that is the distilled model.

The idea of DKD is to first posit a Dirichlet prior distribution over the parameters $\boldsymbol{\pi}$ characterizing the output categorical distribution (e.g., binomial in our problem setup) predicted from a model as follows: $\Pr(\boldsymbol{\pi}|\boldsymbol{x}_t; m_{sk}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_{sk})$; $\quad \boldsymbol{\alpha}_{sk} = (\alpha_{sk,0}, \alpha_{sk,1}) = \boldsymbol{f}(\boldsymbol{x}_t; m_{sk})$, A surrogate prior network $\boldsymbol{f}$ is fit to generate the concentration $\boldsymbol{\alpha}_{sk}$ of the Dirichlet distribution.

If the learnt concentration parameters yield a flat prior distribution, then it means high uncertainty in the model prediction; if they yield a sharp prior distribution, it means low uncertainty. Then an estimation $\hat{q}_{sk}(\boldsymbol{x}_t)$ can be computed from the expected predictive probability $\hat{p}_{sk}(\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_{sk})}[\pi_1] = \alpha_{sk,1}/(\alpha_{sk,0} + \alpha_{sk,1})$. For training the DKD model, which is a K_s-head MLP per each stage in this paper, we define the loss function for head k as a Kullback-Leibler (KL) divergence between the prior distribution and empirical observed distribution: $\mathcal{L}(\boldsymbol{\alpha}_{sk}) = \sum_{i=1}^{N_s} \sum_{t=1}^{T_i} \text{KL}\big(\text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_{sk}) \mid\mid p_{sk}(\boldsymbol{x}_{it})\big)$, where $p_{sk}(\boldsymbol{x}_{it})$ are the real predicted probabilities produced from model m_{sk} on input $\boldsymbol{x}_{it}$. Additionally, we also add the cross-entropy loss as an auxiliary loss when training the DKD. Once the distilled model is trained, we replace the $q_{sj}(\boldsymbol{x}_t)$'s with the estimated values from the model. Then for cost constraining, we ensure the selected models do not exceed user-imposed cost constraint by enforcing sparse gating weights over the model

choices. Therefore, we reformulate the objective function in Hard-Gating Sub-Objective 1 as the following Lagrangian function $\min_{G^{IDK}} \mathcal{L}^s_{\text{nll}}(m^{\text{casc}}; \mathcal{D}_s) + \lambda \mathcal{L}^s_{\text{cost}} + \mu \mathcal{L}^s_{\text{sparse}}$, where the second term $\mathcal{L}^s_{\text{cost}}$ takes the cost constraint, controlled by $\lambda > 0$ and the third term $\mathcal{L}^s_{\text{sparse}}$ imposes further sparse regularization on the gating weights w.r.t their L_1 norms, controlled by $\mu > 0$. Formally, we write the last two terms as $\mathcal{L}^s_{\text{cost}} = \big(\max(0, cost(m^{\text{casc}}_s) - c_s)\big)^2$ and $\mathcal{L}^s_{\text{sparse}} = \sum_{i,t} \|G^{IDK}(\hat{q}_{sk}(\boldsymbol{x}_{it}))\|_1$. Now we use stochastic gradient descent algorithms to learn the optimal $\boldsymbol{a}^s$ and $\boldsymbol{b}^s$ minimizing the above loss. Algorithm 2 in Appendix 5.1 summarizes the *soft-gating* training algorithm.

3.2.3 Overall Training Algorithm

To complete the overall training algorithm, we need to learn the optimal gate parameter G^{ICK_1} in Hard-Gating (3) Sub-Objective 2. Given the model m_{sk} picked by one-stage IDK cascade (either using *hard-gating* or *soft-gating* algorithms) for a data example $\boldsymbol{x}_t$ at stage s, we grid search for the optimal thresholds θ_{sk}'s on the predictive probabilities s.t. the type I error on class ICK_1 and type II error on ICK_0 are both minimized. Several existing methods (Liu, 2012; Perkins and Schisterman, 2006; Unal, 2017; Miller and Siegmund, 1982) have been proposed for minimizing both type I error and type II error in various ways, we pick the *Closet-to-(0,1)* (Perkins and Schisterman, 2006) method that finds the optimal threshold achieving the most left upper corner in the ROC curve. Finally, Algorithm 3 in Appendix 5.1 gives the end-to-end training Algorithm for `UnfoldML`, where we can use either Algorithm 1 or 2 to learn the optimal gating policy.

4 Experiments

We evaluate `UnfoldML` on two real-world tasks. The first task is to predict if and when a patient who was newly admitted into the Intensive Care Unit (ICU) of a hospital will develop sepsis (Stage-1), which can then progress into septic shock (Stage-2). The second task is detection of the *subcategory-of-interest* in a label hierarchy that filters out queries that are not of interest in Stage-1 and then refines the predictions into fine classes in Stage-2.

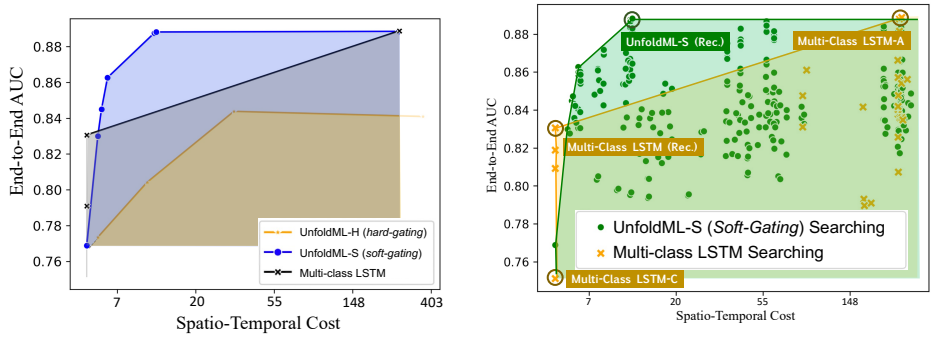
4.1 Task 1: Sepsis-Septic Shock prediction

We use MIMIC-III Critical Care Database (Johnson et al., 2016). The database consists of deidentified health records from over $50,000$ critically ill patients who stayed in the ICUs of the Beth Israel Deaconess Medical Center between 2001 and 2012. We detail our data preparation in Appendix 5.2. The final cohort includes a total of $34,475$ ICU patients, from which $2,370$ (6.8%) presented with Sepsis, from which a total of 229 (9.7%) progressed into Septic Shock. We randomly split our cohort of patient data into a training set (70%), validation set (20%) and test set (10%). First, we use the training set to train a set of models to formulate a model zoo. Next, we use the validation set to train the `UnfoldML` policy. Finally, we use the test set to evaluate performance.

4.1.1 Experimental Setup

Model Zoo. We construct our model zoo by training two sets of binary classifiers for Stage-1 (Sepsis) and for Stage-2 (Septic Shock) using a variety of architectures and features. We select CPU-based models such as Logistic Regression, Decision Tree and Random Forest, and GPU-based models such as LSTM (Long Short-Term Memory) that is thus far the state-of-art approach in the early detection of Sepsis and Septic Shock (Fagerström et al., 2019; Liu et al., 2019). For LSTM, we vary the hidden size ranging from 100 to 400, the number of layers from 1 to 4. We also vary the window of patient data we input over 1, 6, and 12 hours. For each stage and model architecture we train models with different combinations of feature sets based on the collection modality. All models start with with basic demographic features and vital signs, and are then extended with additional features including results from lab tests, beside monitoring, and medication/IV treatments.

Spatio-Temporal Cost. Given the trained model zoo, we profile spatio-temporal costs for each of the models. The spatio-temporal cost is the total time spent in each hardware due to inference/forward-pass calls of the models (temporal cost) multiplied with the hardware's cost per unit time (spatial cost). This cost serves as a proxy for the real dollar cost as it is the basis for pricing models in cloud offerings such as AWS On-Demand, Lambda, or Spot.



(a) Convex hull comparison for the set of points searched by different methods in the End-to-End AUC vs. Spatio-Temporal Cost tradeoff space.

(b) Search the trade off between End-to-End AUC and Spatio-Temporal cost by varying the cost constraint of the system (x-axis is in a logarithmic scale).

Figure 2: Tradeoff space of the End-to-End AUC vs. Spatio-Temporal Cost.

Baseline. To our knowledge, `UnfoldML` is the first system to provide 2-dimensional cascading predictions. A reasonable baseline for our method are models that frame the multi-stage sequential task as a multi-class classification task by ignoring the *happen-before* relationship between the stages. The baseline's goal is to classifying patients into one of the three classes: Non-Septic, Septic, and Septic Shock. We use a state-of-the-art LSTM which is widely used in prediction of clinical time-series. We evaluate performance LSTM's across a variety of spatio-temporal costs by changing the number of layers and hidden sizes in the architecture.

• *Multi-class LSTM*: One unified multi-class model works end to end for predicting the multi classes. Prior work (Wang et al., 2017) implements a 1-D prediction cascade on *IDK* classes. It uses a similar *hard-gating* algorithm to learn hard thresholds on prediction entropy for making decisions if the system should cascade to costlier models. So we reduce `UnfoldML` to a 1-D pipeline by removing its horizontal *ICK* transition and compare the *soft-gating* algorithm with this baseline. We evaluate them on the two single-stage classification tasks on sepsis and septic shock predictions respectively.

• *Single-Stage binary classifiers*: CPU-based models such as Logistic Regression, Decision Tree, Random Forest and TREWScore (Henry et al., 2015), a cox proportional hazards model that is well-known in early detection of septic shock; GPU-based models such as LSTMs.

• *IDK-cascade* (Wang et al., 2017): Prior work of 1-D prediction cascade on *IDK* classes using a hard-gating like algorithm.

4.1.2 Evaluation and Results

End-to-End Classification Performance. To evaluate the prediction performance on our multi-stage task, we treat `UnfoldML` as a multi-class classifier which predicts one of the same three classes as defined in Baseline. `UnfoldML` generates predictions at time-step for every patient, we take the maximum of the predictive probabilities along the prediction horizon for each patient and normalize them with a sum of 1. We compute the multi-class ROC AUC scores by averaging the pairwise ROC AUCs (known as one-vs-one) of each classes.

Better Cost-AUC tradeoff. We evaluate the model performance in trading off between the end-to-end AUC and spatio-temporal cost as follows: we vary the user-defined cost constraint c in Eq. **??**, and train `UnfoldML` repeatedly while searching the trade off in a 2-D space. Each run contributes a point in the scatter plot of Figure 3. We compute the convex hull of the searched points in the set and demonstrate the resulting area of the hull in Figure 2a. We train our baseline method repeatedly by varying the LSTM architecture and plot in the same way as in Figure 2a. In comparison, `UnfoldML`-S (*soft-gating*) searches significantly higher AUC regions given different cost values than the multi-class LSTM baseline and performs more consistently than the *hard-gating* algorithm. To quantify the trade-off evaluation in Figure 2a, we calculate the area of the convex hull searched by each method (Table 4.1.2).

`UnfoldML`-S presents the highest score comparing to the baseline with `UnfoldML`-H. In addition, we pick several critical points from the searched convex hull in Figure 2b and present them in

	Sepsis Prediction		Septic Shock Prediction	
	Single-Stage AUC (%)	Spatio-Temporal Cost Per Inference Call ($)	Single-Stage AUC (%)	Spatio-Temporal Cost Per Inference Call ($)
TREWScore (Henry et al., 2015)	-	-	83.0	**2.3**
Logistic Regression	74.5	**2.3**	87.0	**2.3**
Decision Tree	70.4	2.4	83.1	2.4
Random Forest	75.7	148	85.2	148
LSTM-C	88.6	5.1	90.3	5.1
LSTM-A	**92.8**	268	**96.9**	268
IDK-Cascade (Wang et al., 2017)	91.7	28.1	95.2	34.0
`UnfoldML`-**S**	*91.1*	*8.3*	*96.2*	*10.0*

Table 3: Single-stage performance

Table 2. The recommended configuration for systems that effectively trade off spatio-temporal cost and end-to-end AUC is to select models along the frontier line drawn through the points that form the top-left corner of the hull. As shown in Table 2, an optimal configuration for our pipeline `UnfoldML`-S (Rec.) can achieve comparable end-to-end AUC score (88.9%) with the most accurate baseline Multi-Class LSTM-A (88.8%), while operating at a 19.6X spatio-temporal cost reduction. In comparison with `UnfoldML`-A, which naively uses the most accurate single-stage models in each stage, `UnfoldML`-S outperforms the end-to-end AUC while providing a 22.7X reduction in spatio-temporal cost. Alternatively, `UnfoldML`-S (Rec.) outperforms Multi-class LSTM (Rec.) at an AUC gain of 5.3% with only 1.3X more cost. The improvement in end-to-end AUC scores from `UnfoldML` is a result of the dynamic nature of inference pipeline, which selects the optimal pathway for each patient. The savings in spatio-temporal cost can be attributed to the usage of low cost models when they are confident enough, in contrast, baselines (`UnfoldML`-A, Multi-Class LSTM-A) always use the most accurate model.

	Approx. area of the convex hull
`UnfoldML`-H	17.58
`UnfoldML`-**S**	**31.17**
Multi-class LSTM	24.20

Table 1: Numerical evaluation on Cost-AUC tradeoff by computing the approximated area of the convex hull searched by different methods in Figure 2a.

	Sepsis-Septic Shock Prediction		
	end-to-end AUC (%)	Spatio-Temporal Cost Per Inference Call ($)	Early Prediction on Septic Shock (hr)
Multi-class LSTM-C	75.1	**5.1**	12.6
Multi-class LSTM-A	**88.9**	269.0	24.0
Multi-class LSTM (Rec.)	83.5	6.0	15.4
`UnfoldML`-C	76.9	5.8	22.9
`UnfoldML`-A	87.2	311.8	16.4
`UnfoldML`-H (Rec.)	84.4	34.2	14.2
`UnfoldML`-**S (Rec.)**	88.8	13.7	**26.1**

Table 2: End-to-end performance comparison on two-stage prediction ('-C' denotes the model choice of *the cheapest and least accurate*; '-A' denotes the model choice of *the costliest and most accurate*; '-Rec.' denotes the recommended model choice with a good *trade-off between cost and accuracy* in Figure 2b); '-S' denotes *soft-gating*; '-H' denotes *hard-gating*.

Single-Stage Performance. We report single-stage performance on Sepsis and Septic Shock predictions in Table 3. For both predictions, `UnfoldML` achieves a significant reduction in costs with only a marginal loss of AUC compared to the highest accuracy baseline LSTM-A — 32.3x lower spatio-temporal cost with 1.7% lower AUC for Sepsis prediction and 26.8x lower costs with 0.7% lower AUC for Septic Shock prediction. The baseline IDK-Cascade can achieve comparable AUC as `UnfoldML`-S, but requires 3.4x higher costs.

Better Early-hour Prediction. Table 2 also reports the average early prediction, showing `UnfoldML`-S predicts septic shock earlier than all other methods. It predicts 2.1 hrs prior to the strongest baseline Mutli-class LSTM-A. It benefits from the multi-stage cascaded prediction in `UnfoldML` such that the system can exit early from the sepsis stage once it is in the ICK_1 class and proceed to shock prediction before sepsis is truly diagnosed.

4.2 Task 2: Subcategory Classification

Our second experiment is a computer vision task demonstrating the generalizability of `UnfoldML`. Our goal identification of images from a subcategory in a dataset with hierarchical label structure. For our experiment we use the CIFAR-100 dataset (Krizhevsky et al., 2009), and define two distinct *subcategory-of-interest* tasks based on real-world computer vision systems. We want to identify images with the subcategory from from one of two categories: 'people' or 'vehicles . CIFAR-100 consists of 60,000 32x32 colour images with a total of 20 'coarse' labels and 100 'fine' labels. For

each 'fine' class there are 500 training and 100 testing images. In our experiments Stage-1 predicts the 'coarse' label, and Stage-2 predicts the 'fine' label.

4.2.1 Experimental Setup

Our model zoo is made of WideResNets (WRN) using Sharpness-Aware Minimization (Foret et al., 2020) which is cited as having state-of-the-art performance on the CIFAR-100 image recognition task. For each stage we train models with widths from $[2, 4, 6, 8, 10]$ and depths from $[16, 22, 24, 28]$. For Stage-1 we train multi-class models using full dataset with 'coarse' labels as the target. For Stage-2 we target the 'fine' subcategory that *happens-after* the 'coarse' subcategory from Stage-1. We train Stage-2 models using data from the coarse-granularity label. For Stage-1 we use the 32x32 image, or a random crop of size 24x24 or 16x16. For Stage-2 we train using the 32x32 image or random crop of 24x24. We profile each model to determine the number of Multiply-Accumulate Operations (MACs) (using toolkit[1]) which we use as proxy measure of spatio-temporal cost.

4.2.2 Evaluation and Results

We compare `UnfoldML` against a baseline multi-class WRN classifier which we train for directly predicting labels at the fine-granularity. We train our baseline with a width factor of 10 and a depth of 28 using the full-sized 32x32 image. We evaluate accuracy on our *subcategory-of-interest* task and report our results in Table 4. Our results show that `UnfoldML` is able to find the optimal gating policy for the pipeline and achieve a spatio-temporal cost-savings of 6.9X with 1% reduction in accuracy for 'people' subcategory classification; we achieve a cost-savings of 4.7X with 0.4% reduction in accuracy for 'vehicle' *subcategory-of-interest* identification.

	Multi-class Accuracy (%)	Macs (10M)	Multi-class Accuracy (%)	Macs (10M)
	Classify 'People'		Classify 'Vehicles'	
WRN-A	98.1	525.0	99.3	667.0
`UnfoldML`-**S**	97.1	77.0	98.9	142.3

Table 4: Performance on subcategory classification

An alternative method is Classification with Costly Features (CwCF) (Janisch et al., 2019), which navigates trade-off space using Deep Q-Learning while varying cost parameters which restricts using features when training models. CWCF requires a pre-trained cost-unaware classifier, which is used when no features are restricted. As a fair baseline against `UnfoldML`, we used the highest accuracy WRN in CwCF. We vary the cost parameter and search for the optimal trade-off point in the accuracy-cost space. CWCF returns two extreme data points with either poor or high accuracy as the WRN-A shows in Table 4. We observed similar failure in traversing the trade-off space for CIFAR-10 in Figure 4(c) of the original paper, and therefore we do not include CWCF as a baseline for this task.

5 Conclusions

ML models, including for healthcare applications, are growing exponentially in size and cost of inference. This is problematic for resource constrained hospital environments. Costlier, monolithic models require expensive hardware, and can not fit on bed-side compute or even on site compute clusters with clinical implications.`UnfoldML` proposes a set of mechanisms and policies to address the growing cost of monolithic classifiers for healthcare applications. It implements a query propagation mechanism that "unfolds" a monolithic multi-class classifier into a sequence of single-class classifiers, each with its own cascade progressively more complex models. Each query is allowed to (1) confidently exit the pipeline with an ICK_0, (2) transition horizontally to the next stage in the pipeline with an ICK_1, or (3) upgrade vertically to a more complex model within the horizontal stage with an IDK. This mechanism is coupled with a set of policies, such as soft-gating, that set the thresholds at which the state transitions occur for queries to the system. `UnfoldML` builds on a fundamental insight that classes may have a "happens before" relationship between them, which can be leveraged to "unfold" a classifier, leading to savings in spatio-temporal cost (how much resource used for how long) and clinically significant earlier onset prediction. `UnfoldML` improves the frontier of optimality in the cost/accuracy tradeoff space and is able to nearly match (within 0.1%) SoTA AUC performance for septic shock prediction at the $\frac{1}{20}^{th}$ of the baseline cost. `UnfoldML` and the application of the "happens before" insight generalizes to computer vision tasks with 5x cost savings gained for a mere 0.4% drop in accuracy. Limitations of this work include demonstrations on more than 2 stages tasks.

[1] https://github.com/sovrasov/flops-counter.pytorch

References

Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit Ogale, and Dave Ferguson. 2015. Real-time pedestrian detection with deep network cascades. (2015).

Ben Athiwaratkun and Andrew Gordon Wilson. 2018. On Modeling Hierarchical Data via Probabilistic Order Embeddings. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJCXZQbAZ

Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. 2015. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 3361–3369.

Matthieu Clertant, Nataliya Sokolovska, Yann Chevaleyre, and Blaise Hanczar. 2019. Interpretable cascade classifiers with abstention. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2312–2320.

Anuvabh Dutt, Denis Pellerin, and Georges Quénot. 2017. Improving image classification using coarse and fine labels. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. 438–442.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314* (2013).

Josef Fagerström, Magnus Bång, Daniel Wilhelms, and Michelle S Chew. 2019. LiSep LSTM: a machine learning algorithm for early detection of septic shock. *Scientific reports* 9, 1 (2019), 1–8.

Chen Feng, Paul Griffin, Shravan Kethireddy, and Yajun Mei. 2020. A boosting inspired personalized threshold method for sepsis screening. *Journal of Applied Statistics* (2020), 1–22.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412* (2020).

Katharine E Henry, David N Hager, Peter J Pronovost, and Suchi Saria. 2015. A targeted real-time early warning score (TREWScore) for septic shock. *Science translational medicine* 7, 299 (2015), 299ra122–299ra122.

Geoffrey E Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*. 5–13.

RA Jacobs, MI Jordan, SJ Nowlan, and GE Hinton. [n. d.]. Adaptive mixtures of local experts, Neural Computation3 (1991), 79-87. *Google Scholar Google Scholar Digital Library Digital Library* ([n. d.]).

Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. 2019. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3959–3966.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016), 160035.

Fereshte Khani, Martin Rinard, and Percy Liang. 2016. Unanimous Prediction for 100% Precision with Application to Learning Semantic Mappings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 952–962. https://doi.org/10.18653/v1/P16-1090

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

Jing Lei. 2014. Classification with confidence. *Biometrika* 101, 4 (2014), 755–769.

Jie Lei, Zhenyu Guo, and Yang Wang. 2017. Weakly supervised image classification with coarse and fine labels. In *2017 14th Conference on Computer and Robot Vision (CRV)*. IEEE, 240–247.

Mingkun Li and Ishwar K Sethi. 2006. Confidence-based classifier design. *Pattern Recognition* 39, 7 (2006), 1230–1240.

Ran Liu, Joseph L Greenstein, Stephen J Granite, James C Fackler, Melania M Bembea, Sridevi V Sarma, and Raimond L Winslow. 2019. Data-driven discovery of a novel sepsis pre-shock state predicts impending septic shock in the ICU. *Scientific reports* 9, 1 (2019), 1–9.

Xinhua Liu. 2012. Classification accuracy and cut point selection. *Statistics in medicine* 31, 23 (2012), 2676–2686.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472.

David John Cameron Mackay. 1992. *Bayesian methods for adaptive models*. Ph. D. Dissertation. California Institute of Technology.

Andrey Malinin and Mark Gales. 2018. Predictive Uncertainty Estimation via Prior Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 7047–7058.

Rupert Miller and David Siegmund. 1982. Maximally selected chi square statistics. *Biometrics* (1982), 1011–1016.

Matan Orbach and Koby Crammer. 2012. Graph-based transduction with confidence. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 323–338.

Neil J Perkins and Enrique F Schisterman. 2006. The inconsistency of "optimal" cutpoints obtained using two criteria based on the receiver operating characteristic curve. *American journal of epidemiology* 163, 7 (2006), 670–675.

Vikas C Raykar, Balaji Krishnapuram, and Shipeng Yu. 2010. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 853–860.

Matthew A Reyna, Chris Josef, Salman Seyedi, Russell Jeter, Supreeth P Shashikumar, M Brandon Westover, Ashish Sharma, Shamim Nemati, and Gari D Clifford. 2019. Early prediction of sepsis from clinical data: the PhysioNet/Computing in Cardiology Challenge 2019. In *2019 Computing in Cardiology (CinC)*. IEEE, Page–1.

Andrew Rhodes, Laura E Evans, Waleed Alhazzani, Mitchell M Levy, Massimo Antonelli, Ricard Ferrer, Anand Kumar, Jonathan E Sevransky, Charles L Sprung, Mark E Nunnally, et al. 2017. Surviving sepsis campaign: international guidelines for management of sepsis and septic shock: 2016. *Intensive care medicine* 43, 3 (2017), 304–377.

Henry A Rowley, Shumeet Baluja, and Takeo Kanade. 1998. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence* 20, 1 (1998), 23–38.

Christopher W Seymour, Vincent X Liu, Theodore J Iwashyna, Frank M Brunkhorst, Thomas D Rea, André Scherag, Gordon Rubenfeld, Jeremy M Kahn, Manu Shankar-Hari, Mervyn Singer, et al. 2016. Assessment of clinical criteria for sepsis: for the Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3). *Jama* 315, 8 (2016), 762–774.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. https://openreview.net/pdf?id=B1ckMDqlg

Mervyn Singer, Clifford S Deutschman, Christopher Warren Seymour, Manu Shankar-Hari, Djillali Annane, Michael Bauer, Rinaldo Bellomo, Gordon R Bernard, Jean-Daniel Chiche, Craig M Coopersmith, et al. 2016. The third international consensus definitions for sepsis and septic shock (sepsis-3). *Jama* 315, 8 (2016), 801–810.

Reisa A Sperling, Paul S Aisen, Laurel A Beckett, David A Bennett, Suzanne Craft, Anne M Fagan, Takeshi Iwatsubo, Clifford R Jack Jr, Jeffrey Kaye, Thomas J Montine, et al. 2011. Toward defining the preclinical stages of Alzheimer's disease: Recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease. *Alzheimer's & dementia* 7, 3 (2011), 280–292.

Kirill Trapeznikov and Venkatesh Saligrama. 2013. Supervised sequential classification under budget constraints. In *Artificial intelligence and statistics*. PMLR, 581–589.

Thomas P Trappenberg and Andrew D Back. 2000. A classification scheme for applications with ambiguous data. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, Vol. 6. IEEE, 296–301.

Ilker Unal. 2017. Defining an optimal cut-point value in ROC analysis: an alternative approach. *Computational and mathematical methods in medicine* 2017 (2017).

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 2579-2605 (2008), 85.

Shikhar Vashishth, Prateek Yadav, Manik Bhandari, and Partha Talukdar. 2019. Confidence-based graph convolutional networks for semi-supervised learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1792–1801.

Luke Vilnis and Andrew McCallum. 2015. Word Representations via Gaussian Embedding.. In *ICLR*. http://arxiv.org/abs/1412.6623

Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *International journal of computer vision* 57, 2 (2004), 137–154.

Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. 2017. Idk cascades: Fast deep learning by learning not to overthink. *arXiv preprint arXiv:1706.00885* (2017).

Xin Wang, Jiawei Wu, Da Zhang, Yu Su, and William Yang Wang. 2019. Learning to compose topic-aware mixture of experts for zero-shot video captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 8965–8972.

Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. 2020. Deep mixture of experts via shallow embedding. In *Uncertainty in Artificial Intelligence*. PMLR, 552–562.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated Residual Transformations for Deep Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 5987–5995.

Yanbo Xu, Siddharth Biswal, Shriprasad R Deshpande, Kevin O Maher, and Jimeng Sun. 2018. RAIM: Recurrent Attentive and Intensive Model of Multimodal Patient Monitoring Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2565–2573.

Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and S. Li. 2018. Occlusion-aware R-CNN: Detecting Pedestrians in a Crowd. In *ECCV*.

Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. 2010. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing (TSLP)* 6, 3 (2010), 1–24.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The primary goal of this work to offer a better trade-off AUC and cost trade-off space and significantly better early hour prediction of sepsis and shock
   (b) Did you describe the limitations of your work? [Yes] Section 5 provides limitations of proposed approaches.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Refer to Section 4
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The details of experimental setup are provided in Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]