Deep Classification with Class-Specific Risk Control With Application to Sleep Staging

CS 498 Project Report: Presentation at https://youtu.be/YOcW3mBE4Qg

Zhen Lin
University of Illinois
zhenlin4@illinois.edu

1. INTRODUCTION

Deep Learning as a method has seen tremendous progress over the recent years. One of the most important application of DL is in the general task of classification, and due to its high discriminative power, DL models have been successfully applied in many important tasks such as sleep staging [2; 14].

One important question yet to be addressed fully is the question about model risk, which is also closely related to model uncertainty. For example, consider an automated sleep staging system. Apart from accuracy, clearly there are two additional goals: The first is *uncertainty quantification* - that is, the model should be aware of its own prediction uncertainty level. This is a topic that has a relatively long history but recently attracted much more attention, and methods include mostly Bayesian approaches [4; 3; 21] and ensemble methods [10]. Secondly, and more importantly, the model should adjust its behavior according to this uncertainty measure, such that when it predicts, the risk (e.g. chance of mistake) should be lower than a predetermined value. In fact, in many cases, including this sleep staging example, what we want is more than controlling the overall risk - we need class-specific risk controls, because different classes (sleep stages) bear different significance. This is a much less studied area, and to the best of our knowledge, the relatively few works focus only on the overall risk control, essentially phrasing this as a binary classification problem on top the original multi-class classification problem [5]. We will review relevant topics in more details in Section 4. As we will later see, controlling only the overall risk induces undesirable behaviors. For example, the model would only predict easier classes and ignore the harder ones.

In this paper we aim to tackle this class-specific risk control problem. The major contribution of this paper is that we propose the first class-specific risk control method for multi-class classification tasks, which can be applied to DL models. This general and theoretically-grounded method has the following desirable properties:

1. It enforces class-specific risk controls.
2. It is *post-hoc*, which means that it does not depend on the model architecture, and won't affect the performance of the base model.
3. It is efficient - in particular, the run-time does not depend the complexity of the base model, but only on the number of classes and, *optionally*, the size of the validation data.
4. When uncertain, it returns a (confidence) set of possible labels. This confidence set is the least ambiguous among all confidence sets with the same coverage level (Theorem 3.1).

In the rest of the paper, we will first formulate the problem mathematically in Section 2, and describe the details of our method, including the parameterization, use of split conformal techniques, and optimization method in Section 3. We will then review in Section 4 some related works and why they cannot apply to our problem. Finally, we will present experiment results on both synthetic data and two real-world sleep-staging datasets in Section 5

2. PROBLEM FORMULATION

2.1 Base Model and Learning Setup

In this work we situate our task in the K-class classification problem, with data space $\mathcal{X}$, label space $\mathcal{Y} = \{1, \dots, K\}$, and the joint distribution of (X, Y) as $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$. We further denote the class-specific distributions $(X, Y = k)$ as $\mathbb{P}_k$, which is effectively a distribution over $\mathcal{X}$. $\mathbb{P}_k\{\cdot\}$ can also be viewed as $\mathbb{P}\{\cdot|Y = k\}$.

We are given a trained model m (potentially a DNN): $\mathcal{X} \mapsto \mathbb{R}^K$, where the k-th output, $m(x)_k$, captures the conditional probability $p(y = k|x)$. More details will be discussed in Section 3.1 (about "confidence score"), but for a simple example, consider $m(x)$ as the Softmax output over the K classes. We are also given $\{m(x)\}_{x\in\mathcal{S}_V}$, where $\mathcal{S}_V$ is a hold-out validation set.

2.2 Problem: Class-Specific Risk Control

We will begin with a few definitions. A *set classifier* is a mapping from data to a set of labels, formally denoted as $\mathbf{H} : \mathcal{X} \mapsto 2^{\mathcal{Y}}$ [1; 16]. This is a generalization of the typical point classifier that outputs the most-probable class. For a multi-class classification problem, ideally, we want an oracle classifier such that $\mathbb{P}\{\mathbf{H}_{oracle}(X) = \{Y\}\} = 1$. However, in most cases this is not possible, and we want something similar to a confidence interval in regressions, such as a *confidence set.* For $\boldsymbol{\alpha} \in [0, 1]^K$, we define the $\boldsymbol{\alpha}$-confidence set to be a $\mathbf{H}$ that satisfies the *class-specific coverage* guarantee:

$$\forall k \in [K], \mathbb{P}_k\{k \in \mathbf{H}_{\boldsymbol{\alpha}}(X)\} \geq 1 - \boldsymbol{\alpha}_k \tag{1}$$

We will refer to $1 - \boldsymbol{\alpha}$ as the class-specific coverage levels, a property of a set-classifier $\mathbf{H}$.

Our goal is to find a $\mathbf{H}$ that minimizes the *ambiguity* while satisfying *class-specific risk* constraints. For ambiguity, two equally valid definitions are $\mathbb{E}[|\mathbf{H}(X)|]$ [16] and $\mathbb{P}\{|\mathbf{H}(X)| > 1\}$, and each has its pros and cons. For this reason, we will assume the ambiguity measure, denoted as $A(\mathbf{H})$, is be a linear combination of the two definitions. The risk defined in [5] will translate to the following overall risk in our language:

$$R(\mathbf{H}) := \mathbb{P}\{Y \notin \mathbf{H}(X) | |\mathbf{H}| = 1\} \tag{2}$$

We will however bound the class-specific risks:

$$R_k(\mathbf{H}) := \mathbb{P}_k\{k \notin \mathbf{H}(X) | |\mathbf{H}| = 1\} \tag{3}$$

Note that there is only risk when the model is certain, whereas coverage level (E.q. 1) is a simpler unconditional measure. Note also that in the binary classification case, $risk + coverage = 1$ if $\mathbf{H} \neq \emptyset$.

Putting everything together, our goal is to solve the following optimization problem:

$$\min_{\mathbf{H}} \quad A(\mathbf{H}) \tag{4}$$

$$\text{s.t.} \quad \mathbb{P}_k\{k \notin \mathbf{H}(X) | |\mathbf{H}(X)| = 1\} \leq \boldsymbol{r}_k, \forall k \in [K] \tag{5}$$

Where $A(\mathbf{H})$ is the ambiguity measure, and $\boldsymbol{r} \in [0,1]^K$ is the risk control (not to be confused with $\boldsymbol{\alpha}$).

3. CLASS-SPECIFIC RISK CONTROL (CSRC)

Method Overview: In this section we propose an search-based optimization method that solves the specific optimization in Section 2.2 and a much more general class of problems. We will first parameterize $\mathbf{H}$ with K thresholds on model outputs, and show a theoretical justification of it. Then, we will tune the parameters on the validation set (Section 3.2), by using a variant of coordinate descent will be used to optimize the parameters (Section 3.3).

We want to emphasize that although we will be focusing on the specific problem in Section 2.2, what we are proposing here is a **general-purpose** framework to address many optimization problems over the set classifier $\mathbf{H}$ where the objective and constraints can be a wide class of functions.

3.1 H as a Union of Sets

We will approach Eq. (4) by first restricting $\mathbf{H}$ to a specific class of set classifiers. We define $\mathbf{H}$ with K subsets of $\mathcal{X}$, $\{C_y\}_{y=1}^K$:

$$\mathbf{H}(x) := \{k \in [K] : x \in C_k\} \tag{6}$$

Intuitively, we first define K sub-regions in $\mathcal{X}$ where the k-th one encompasses "points that *probably* belong to the k-th class". Then, we assign all classes x might belong to to $\mathbf{H}(x)$. The fact that we "ignored" the potential interaction between classes seems like a bad thing, but such $\mathbf{H}$ is already optimal in the following sense:

THEOREM 3.1. *$\forall k = 1, \ldots, K$, we define $C_k := \{x : p(y = k|x) \geq t_k\}$ for some $t_k \in \mathbb{R}$ Then, the corresponding $\mathbf{H}$ has the minimum ambiguity (define as $\mathbb{E}[|\mathbf{H}|]$) among all set classifiers satisfying the class-specific coverage guarantee with $1 - \boldsymbol{\alpha}$ such that $\boldsymbol{\alpha}_k = \mathbb{P}\{X \notin C_k | Y = k\}$).*

A similar result can also be found in [16].

As suggested by Theorem 3.1, we will parameterize C_k (and thus $\mathbf{H}$) using K quantiles among $p(y = k|x)$. It is then immediately clear that as long as we have access to a "*confidence score*" metric $s_k(x)$ that's rank-preserving - namely, $s_k(x) > s_k(x') \Leftrightarrow p(y = k|x) > p(y' = k|x')$ - then we can choose thresholds on $\{s_k(x)\}_{x \sim \mathbb{P}_k}$ instead of on the conditional probabilities. In practice, the base model usually outputs a vector that can be normalized to and interpreted as a probability distribution, with Softmax being arguably the most popular output layer. Un-calibrated Softmax scores tend to deviate from true probabilities [7] [1] However, basing on the prior discussion, we only want they to be rank-preserving. We will thus use Softmax scores as the class-specific "confidence scores", and in the rest of the paper, unless specified, $s_k(x)$ denote the Softmax confidence score for class k.

Once we have such $s_k(x)$, we parameterize $\mathbf{H}$ using quantiles on $s_k(x)$. Although Theorem 3.1 focuses only on the first definition of ambiguity, there are several benefits to this parameterization: First of all, for any such $\mathbf{H}$, we automatically know the coverage levels (which will carry over to test time per Section 3.2). Secondly, such $\mathbf{H}$ tells us what are the uncertain classes, and it does not include more classes than necessary. Finally, this is a very simple parameterization that is less prone to over-fitting. Therefore, we will use this form for $\mathbf{H}$ even if $A(\mathbf{H})$ is a linear combination of both ambiguities.

3.2 Split Conformal Inference

We will learn choose the thresholds $\{t_k\}_{k \in [K]}$ using the split conformal method [11; 13], which is popular in the general field of conformal inference pioneered by [18]. Split Conformal Inference methods, or in general Conformal Inference methods, are very powerful in that they require very minimal assumptions and apply to various types of models. What it means in our case is very straightforward - we evaluated our $\hat{\mathbf{H}}$ on a hold-out validation set, and hope that these properties will carry over at test time. The idea is that, because the model is independent to all the data in the validation set and test inference data, assuming the data are *exchangeable*, the scores' distribution on the validation set can represent that at test time. Below is a formal definition of exchangeability:

DEFINITION 1. *$\forall n$, a sequence of random variables $\{X_i\}_{i=1}^n$ is exchangeable if for any permutation σ, $X_1, X_2, \ldots, X_n$ and $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(n)}$ follow the same joint distribution.*

It's worth noting that exchangeability is a weaker assumption than the common iid assumption, as it does not say anything about independence.

[1]In addition, the form of the loss function should also be considered.

Now, assume that we have acquired a sequence of data $X_1, X_2, \dots, X_N$, and we randomly split them into training set, on which the base model is trained, and validation set $\mathcal{S}_V$. Suppose we now define a $\hat{\mathbf{H}}$ using the $\{s_k(x)\}_{x\in\mathcal{S}_V, k\in[K]}$. For a new data X_{N+1} at test inference time, $\hat{\mathbf{H}}(X_{N+1})$ will still in expectation satisfy the properties of $\hat{\mathbf{H}}$ on $\mathcal{S}_V$ (such as risk targets), as suggested by the following lemma and theorem:

LEMMA 3.2. *Let $\{X_i\}$ be an exchangeable sequence of R.V., and s be a function independent to these data* [2]. *Let $R \subseteq [0,1]^K$ be a fixed region, then Then, $\mathbb{P}\{s(X_{N+1}) \in R\} = \mathbb{P}\{s(x_{\sigma(N+1)}) \in R\}, \forall \sigma \in S_{N+1}$, where S_{N+1} is the symmetric group of $[N+1]$.*

This is a direct result of exchageability: s being independent of the data means $\{s(X_i)\}$ are also exchangeable.

THEOREM 3.3. *For any **fixed** set classifier $\mathbf{H}$, if data are exchangeable, for a new test data $(X_{N+1}, Y_{N+1} = k)$, $\mathbb{P}\{k \notin \mathbf{H}(X_{N+1}) \| \mathbf{H}(X_{N+1})| = 1\} \leq \boldsymbol{r}_k$. Moreover, the expectation of empirical risks on the validation set equal to the true conditional probability.*

Proof The first statement is a direct result of exchangeability. For the second statement, denote $A = \{k \notin \mathbf{H}(X) \wedge |\mathbf{H}| = 1\}$ and $B = \{|\mathbf{H}| = 1\}$. Suppose we repeat the experiment (of drawing data Xs) N times, and denote N_A and N_B as the # of times A or B happen, and suppose we at least observe B once[3]

$$
\begin{aligned}
&E[\frac{N_A}{N_B}|N_B \geq 1] \\
=&\sum_{m=1}^{N} E[\frac{N_A}{m}|N_B = m]\mathbb{P}\{N_B = m|N_B >= 1\} \text{ (Partition Theorem)} \\
=&\sum_{m=1}^{N} \frac{E[\sum_{j=1}^{m} \mathbf{1}\{A|B\}]}{m}\mathbb{P}\{N_B = m|N_B >= 1\} \\
=&\sum_{m=1}^{N} \mathbb{P}\{A|B\}\mathbb{P}\{N_B = m|N_B >= 1\} \\
=&\mathbb{P}\{A|B\}
\end{aligned}
$$

Note that we only used the linearity of expectation which does not require independence.

Remarks Intuitively, Lemma 3.2 is just saying that the distribution of the predictions on the validation set is representative of unseen test data. Each of the class-specific risk controls is in the form of $\frac{\mathbb{P}_k\{s(X)\in R'\}}{\mathbb{P}_k\{s(X)\in R\}}$ where $R' \subseteq R$. For Theorem 3.3, $\{X_i\}_{i=1}^{N+1}$ being exchangeable means $\{s(X_i)\}_{i=1}^{N+1}$ are exchangeable, which further means that $\{s(X_{i'})\}_{s(X_{i'})\in R}$ is also exchangeable. Therefore, if the data is exchangeable, we should expect all risks to carry over at test inference time as well.

3.3 Random Coordinate Descent

After the previous discussions, it's clear that we will need to solve the following optimization problem:

$$\min_{t_1,\dots,t_K} \hat{A}(\mathbf{H}) \tag{7}$$

$$\text{s.t. } \hat{\mathbb{P}}_k\{s_k(X) < t_k \| \mathbf{H}(X)| = 1\} \leq \boldsymbol{r}_k, \forall k \in [K] \tag{8}$$

where $\hat{A}$ and $\hat{\mathbb{P}}$ denote the empirical ambiguity and probability (frequency) on $\mathcal{S}_V$, and t_k corresponds to a quantile of $\{s_k(x)\}_{x\in\mathcal{S}_V}$ which parameterize $\mathbf{H}$.

To solve it, we transform the hard constraints into soft penalties, which leads to the following empirical loss

$$\hat{L} := \underbrace{\hat{A}(\mathbf{H})}_{\text{ambiguity}} + \sum_{k=1}^{K} \underbrace{\lambda_k(\hat{\mathbb{P}}_k\{X \notin C_k \| \mathbf{H}(X)| = 1\} - \boldsymbol{r}_k)_+^2}_{\text{class specific risk control violation penalty}} \tag{9}$$

where $v_+ := \max\{v, 0\}$. Obviously, one can also use the same penalty coefficient for all classes - namely $\lambda_k \equiv \lambda$ for a fixed λ.

Like before, denote the validation set as $\mathcal{S}_V$ and $N = |\mathcal{S}_V|$. At the heard of this problem, we face a discrete optimization problem over the space of N^K quantiles, and the loss function does not have any "nice" form. A brute force algorithm will take $O(N^K)$ loss evaluations, which is in most cases too expensive to compute even for a small dataset (e.g. $N = 10000$, $K = 5$). To optimize the loss function, we combine multiple ideas below, with the full details in Algorithm 1.

1. In each run, search only in a subset of $\mathcal{S}_V$.
2. Search along one coordinate in each step (similar to Coordinate Descent), because evaluating the loss along on dimension is relatively fast.
3. Repeat this process several times, each time with a random starting position (and potentially a different subset), and pick the best run.

In Algorithm 1, the search step in each coordinate descent iteration is given by Algorithm 2. Empirically, we do find that this method works reasonably well if K is not huge - arguably the only case where class-specific risk controls makes sense anyways.

It should be noted that Theorem 3.3 does not apply to an $\mathbf{H}$ optimized to lower the risk on the validation set. However, our method also did not optimize on the class-specific risks (our optimization is essentially trying to find a target risk level), and in practice the search space is very small. Moreover, we restricted $\mathbf{H}$ to a very simple class of set classifiers. As a result, we expect the class-specific risks to generalize relatively well to test time, which is true in our experiments. In fact, in order to get a real risk guarantee, we just need to make $\mathbf{H}$ independent to the validation set. For example, we can split validation set into 2, use one to fit $\hat{H}$ and the other to estimate the true risk.

[2]e.g. fitted on a different sample
[3]otherwise our expectation of $\mathbb{P}\{B\} = 0$ and we just need Lemma 3.2

Algorithm 1 Random Coordinate Descent

```
Input: loss function $l(thresholds, sample)$, number of repeats $B$, (optionally $M < N$)
Initialize $b = 1$, $t^*$ as an empty vector, and $l^* = \infty$
repeat
    (optionally) Sample $\mathcal{S}_b$ of size $M$ from $\mathcal{S}_V$
    Pick $x_1, \ldots, x_K \in [M]$ as the initial quantiles
    $l_b \leftarrow l(s_1(x_1), \ldots, s_K(x_K), \mathcal{S}_b)$
    repeat
        for $k = 1$ to $K$ do
            Search for the optimal $x'_k$ in $\mathcal{S}_b$, fixing other quantiles
            $l'_b \leftarrow l(s_1(x_1), \ldots, s_k(x'_k), \ldots, s_K(x_K), \mathcal{S}_b)$
            if $l'_b < l_b$ then
                Update $l_b \leftarrow l'_b$ and $x_k \leftarrow x'_k$
            end if
        end for
    until $l_b$ does not decrease
    $l_b \leftarrow l(s_1(x_1), \ldots, s_K(x_K), \mathcal{S}_V)$
    if $l_b < l^*$ then
        Update $l^* \leftarrow l_b$ and $t^* \leftarrow [s_1(x_1), \ldots, s_K(x_K)]$
    end if
until $b = B$
return $l^*$ and $t^*$
```

Algorithm 2 Search

```
Input: $S = [s_1 \leq s_2 \leq \ldots \leq s_n]$, and a loss function $l(\cdot) : S \mapsto \mathbb{R}$
Initialize $low = 1$, $high = n$
repeat
    Split $[low, high]$ into $P$ equal-sized segments ($P$ is a parameter which is set to 7)
    Evaluate $l(\cdot)$ at the endpoints of these $P$ segments.
    Set the $low$ and $high$ to the endpoints of the segment with the lowest average-endpoint-loss
until $high - low \leq P$
Brute-force search for the optimum in $[s_{low}, \ldots, s_{high}]$ and return
```

4. RELATED WORKS

Uncertainty Quantification for Neural Networks There is a long history of using neural networks with **Bayesian** approaches to provide predictive uncertainty measures [12]. However, due to the computation cost of exact Bayesian inference, methods have been proposed to approximate the posterior, including Monte-Carlo Dropout [4], stochastic gradient MCMC [19] Variational Bayesian methods [3] Other Bayesian methods include using deep models to learn the kernels for Gaussian Process [21]. There are also **non-Bayesian** methods proposed recently, among which using ensembles [10] is one of the most successful.

In addition to their own limitations compared with each other, such as computation cost or deterioration of performance, all methods share two major limitations for our task: First, they do not directly apply to classification problems - the uncertainty score is not calibrated, and does not bear specific meanings. Evaluations of the uncertainty measure, mostly qualitative, typically involve examining prediction accuracy for a few ranges of uncertainty level [10], which leads to the second and more fundamental limitation: A separate method is required to translate these scores to risk measures, to answer questions like "How confident the model is with a prediction" in a meaningful way.

Prediction with Rejection is the most applicable research to our task. These methods use the uncertainty levels to decide whether a predication is reliable or not, while controlling the risk when the model chooses to predict. [5] proposes to reject to predict by thresholding on a confidence function, such as variance sampled from MC-dropout or maximum Softmax output. [6] extends the [5] by training the model and the confidence function jointly. It is worth noting that the similar problem of open category/set detection does not apply to our task, as our goal is not to detect any unknown class, and controlling class-specific risk is not meaningful when we have unknown classes.

These works, however, aim to improve the trade-off between *overall* risk and accuracy, and tend to sacrifice certain classes' performance in an uncontrolled manner, which is undesirable. Furthermore, when a rejection happens, we cannot know what are the competing predictions that confuse the model.

Model Calibration tries to calibrate the output of a model to a meaningful probability measure. Most of the works focus on calibrating the probability for the predicted class using methods such as temperature scaling or vector & matrix scaling [15; 7], and Gaussian process based methods [20]. Recently, a few works started focusing on the evaluation [17] and calibration [9] of all classes. While a calibrated output is desirable, calibration still doesn't give us decision rules to control the prediction risks.

5. EXPERIMENTS

We could not find a method that solves the particular class-specific risk control problem we face. As a result, we present two baseline methods, SGR and LABEL, that we consider most relevant to our task. A comparison of all methods can be found in Table 1.
We would like to emphasize that our method is solving a *more general* problem, with the optimization in Eq. (4) being just an instance.

Baselines Selective Guaranteed Risk (SGR) [5] is a post-hoc method that can achieve a overall risk guarantee target. It picks a "confidence-rate function", which is *class-agnostic*, and only make a prediction if such function value is higher than a threshold. For each threshold, SGR can evaluate the empirical risk $\hat{r}$ on the training data, and bounds the probability of the true risk r assuming binomial distribution. It then searches for a threshold such that $P(r > r^*) < \delta$ given a risk bound r^* and probability δ. A follow-up work, SelectiveNet [6], trains the base model and rejection criteria jointly. However, it slightly breaks the post-hoc nature, and does not overcome the limitations of SGR, so we will mainly focus on SGR in our comparisons.

[4]Risks are defined differently for SGR and LABEL. Neither method can handle the other definition.

[4]Risks are defined differently for SGR and LABEL. Neither method can handle the other definition.

Table 1: Comparison of different methods.

PROVIDES:	SGR	CSRC	LABEL
RISK CONTROLS WHEN *certain*	√	√	×
CONFIDENCE SET **H** WHEN *uncertain*	×	√	√
OVERALL RISK CONTROLS[4]	√	√	√
CLASS-SPECIFIC CONSTRAINTS	×	√	√
FLEXIBILITY FOR OTHER PROBLEM SETUPS	×	√	×

LABEL is a multi-class confidence set method proposed by [16]. It also uses quantiles on the validation set to construct $\mathbf{H}$, with a focus on filling the null region (when $\forall k, p_k(x) < t_k$). It minimizes the ambiguity defined as $\mathbf{E}[|\mathbf{H}|]$, and instead of risks controls, it enforces class-specific *coverage* constraints $(1-\boldsymbol{\alpha})$ (E.q. 1). This specific problem setup allows for an almost analytic solution that is fast to compute. However, this setup tends to "spread" the ambiguity to a vast region in $\mathcal{X}$, and it cannot control the risks (E.q. 3). In other words, satisfying the coverage constraints is easy, but translating a set of risk controls ($\boldsymbol{r}$) to coverage levels is very difficult. In experiments we will show how the risks tend to deviate from our expectation with LABEL.

Evaluation The evaluation will include several metrics: the two definitions of ambiguity (when applicable), the overall risk, and the class-specific risks on validation and test set. It should be noted that as none of the baselines control class-specific risk, arguably only the overall risks are comparable. Thus, we will show that our method can control overall risks just like SGR, with comparable ambiguity and added benefits. We show class-specific risks to show that controlling overall risks leads to bad behaviors, and how our method can fix it.

5.1 Synthetic Data

We start by a dataset where the true conditional probability $p(y|x)$ is known. This allows us to assess the performance of the uncertainty quantification method independently, without the complication from the base model (which might not fit the conditional probability well). Specifically, for each data point, we

1. generate $p' \sim \mathcal{N}(0, I_K)$, where I_K is the $K \times K$ identity matrix.
2. sample a class and increase $p'_k \leftarrow p'_k + S$, where S is a parameter denoting the signal strength. A higher S can be thought of a easier/clearer classification, as there are more "sure" predictions.
3. pass p' through Softmax to get a probability distribution p over the K classes.
4. assign label y by sampling from the K classes with probability p.

In this experiment, $K = 5$, $S = 3$ for $k = 0$ and $S = 1$ otherwise. We generated 10000 data for the validation set, and evaluated the performance on another 10000 test data generated in the same fashion. For the CSRC method, $B = 5$.

Results The results are presented in Table 2. Note that SGR is equivalent to the *theoretical optimal* solution in this case by Theorem 3.1, because its essentially answering a binary classification problem, and the input is the *true* conditional probability. Thus, we should never expect it performing worse than CSRC. However, this experiment suggests that, even in the most ideal cases, CSRC is pretty comparable.

Table 2: The rows denote the two definitions of ambiguity and overall risk, in order. Target overall risk is 0.3. For every metric here, a lower value is better.

	VALIDATION		TEST	
METRIC	SGR	CSRC	SGR	CSRC
$\mathbf{P}\{|\mathbf{H}| > 1\}$	0.698	0.700	0.691	0.692
$\hat{\mu}(|\mathbf{H}|)$	N/A	2.14	N/A	2.13
$\hat{r}$	0.283	0.287	0.270	0.282

5.2 ISRUC

ISRUC is a publicly available PSG dataset for sleep staging task, and its sub-group 1 data contains PSG recordings of 100 subjects, with one all-night (around 8 hours) recording session per subject. Each recording consists signals from 19 channels, and were sampled at 200Hz [**?**]. Notably, each recording was scored by **two** different sleep experts per 30 seconds, allowing us to benchmark the model's classifications with experts' results. The task is to classify each segment into five classes, including W, REM, $N1$, $N2$ and $N3$.

Experiment We set the overall risk target for SGR to be 0.15. We set the class-specific risk of CSRC to be $[0.1, 0.1, 0.3, 0.3, 0.1]$ in order to get a relatively even risk distribution, and B is set to 10. For LABEL, we set the target mis-coverage rates to $[0.1, 0.1, 0.3, 0.3, 0.1]$, but there is no way to directly specify the risk. All methods use the output of a 1D CNN model with the same structure as in [14], with the decoder replaced with 2 256-node fully connected layers and a softmax output. The model was trained with 30 epochs. The data was split into 3 sets: 84:6:14 for training:valid:test sets

Results In a real-world application, we see that to control the overall risk target, SGR essentially avoids predicting certain classes that are harder to predict ($N1$ and $N2$). On the other hand, we see that the CSRC method does control the risks well with the target $\alpha = [0.1, 0.1, 0.3, 0.3, 0.1]$. The risks are somewhat different on the test set. However, it should be noted that the class-specific risks of SGR tend to change in the same directions, and are much more volatile. As a result, this is more likely due to the small size of test and validation set (validation set has only 6 patients).

5.3 Sleep EDF

Sleep EDF [8] is a similar sleep-staging dataset containing 197 whole-night PSG sleep recordings. We still phrase this as a 5-class classification problem with the same classes as ISRUC.

Table 3: Class-specific risks and ambiguities for different methods. Note that $\hat{r}_k$s are not controllable for SGR, and the risk targets for CSRC and LABEL are set to $[0.1, 0.1, 0.3, 0.3, 0.1]$.

	Validation			Test		
Metric	SGR	CSRC	LABEL	SGR	CSRC	LABEL
$\mathbf{P}\{\lvert\mathbf{H}\rvert > 1\}$	0.595	0.563	0.188	0.559	0.509	0.193
$\hat{\mu}(\lvert\mathbf{H}\rvert)$	N/A	1.58	1.19	N/A	1.53	1.19
$\hat{r}_{\text{OVERALL}}$	0.134	0.152	0.232	0.112	0.145	0.238
$\hat{r}_W$	0.101	0.098	0.121	0.008	0.023	0.051
$\hat{r}_R$	0.002	0.098	0.075	0.005	0.085	0.057
$\hat{r}_{N1}$	0.773	0.299	0.451	0.837	0.402	0.442
$\hat{r}_{N2}$	0.528	0.295	0.375	0.355	0.232	0.426
$\hat{r}_{N3}$	0.000	0.094	0.097	0.004	0.129	0.099

Experiment We repeat a similar experiment like for ISRUC, this time adding an additional experiment asking CSRC to mimic SGR. Specifically, we first run SGR with the target overall risk level of 0.15. Then, we set the risk targets for CSRC to the (uncontrolled) class-specific realized risks from the SGR experiment - we call this method CSRC $\rightarrow$SGR. This is to show that our method is more flexible and can capture almost any class-specific risk targets.
Like before, we chose the class-specific risks such that they are even and the overall risk is close to 0.15.

Result Results are in Table 4. In addition to what we saw in the experiment of ISRUC, we see that CSRC $\rightarrow$SGR achieves a very similar set of evaluation metrics (on top of providing the extra information of what's in $\mathbf{H}$) to the SGR method.

6. CONCLUSION

In this paper, we proposed a general and theoretically grounded framework to make predictions with class-specific risk controls. Unlike previous methods that control the overall risk, which tend to sacrifice certain harder to predict classes, our method is much more flexible in controlling the behavior of the predictor, with trust-able risk expectations when the data is exchangeable.

7. REFERENCES

[1] Rudolf Beran. Prepivoting to reduce level error of confidence sets. *Biometrika*, 1987.

[2] Siddharth Biswal, Joshua Kulas, Haoqi Sun, Balaji Goparaju, M. Brandon Westover, Matt T. Bianchi, and Jimeng Sun. SLEEPNET: Automated Sleep Staging System via Deep Learning, 2017.

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning, ICML 2016*, 2016.

[5] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, 2017.

[6] Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *36th International Conference on Machine Learning, ICML 2019*, 2019.

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017.

[8] Bastiaan Kemp, Aeilko H. Zwinderman, Bert Tuk, Hilbert A.C. Kamphuisen, and Josefien J.L. Oberyé. Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 2000.

[9] Meelis Kull, Miquel Perello-Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration, 2019.

[10] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

[11] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 2018.

Table 4: Class-specific risks and ambiguities for different methods on the Sleep-EDF dataset. Note that SGR controls for overall risk of 0.15, and the risk targets for CSRC and LABEL are set to $[0.1, 0.4, 0.4, 0.4, 0.4]$.
CSRC $\rightarrow$SGR tries to mimic the realized risk profiles by simply setting the risk targets correspondingly, and the result is very close.

	Validation				Test			
Metric	SGR	CSRC $\rightarrow$SGR	CSRC	LABEL	SGR	CSRC $\rightarrow$SGR	CSRC	LABEL
$\mathbf{P}\{\lvert\mathbf{H}\rvert > 1\}$	0.01	0.03	0.26	0.07	0.00	0.03	0.25	0.07
$\hat{\mu}(\lvert\mathbf{H}\rvert)$	N/A	1.10	1.37	1.07	N/A	1.09	1.37	1.06
$\hat{r}_{\text{OVERALL}}$	0.143	0.131	0.152	0.147	0.145	0.132	0.145	0.148
$\hat{r}_W$	0.024	0.019	0.087	0.047	0.027	0.021	0.088	0.050
$\hat{r}_R$	0.774	0.777	0.392	0.564	0.778	0.769	0.462	0.589
$\hat{r}_{N1}$	0.191	0.187	0.379	0.301	0.187	0.182	0.357	0.288
$\hat{r}_{N2}$	0.798	0.796	0.393	0.607	0.855	0.858	0.376	0.652
$\hat{r}_{N3}$	0.494	0.485	0.390	0.591	0.492	0.481	0.396	0.532

[12] R. Neal. Bayesian Learning for Neural Networks. *LECTURE NOTES IN STATISTICS -NEW YORK-SPRINGER VERLAG-*, 1996.

[13] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002.

[14] Mathias Perslev, Michael Hejselbak Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-Time: A fully convolutional network for time series segmentation applied to sleep staging. In *Advances in Neural Information Processing Systems*, 2019.

[15] John Platt and others. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 1999.

[16] Mauricio Sadinle, Jing Lei, and Larry Wasserman. Least Ambiguous Set-Valued Classifiers With Bounded Error Levels. *Journal of the American Statistical Association*, 114(525), 2019.

[17] Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B. Schön. Evaluating model calibration in classification, 2019.

[18] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world.* 2005.

[19] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011.

[20] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. Non-parametric calibration for classification, 2019.

[21] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, 2016.