Inference over Distribution of Posterior Class Probabilities for Reliable Bayesian Classification and Object-Level Perception

Vladimir Tchuiev and Vadim Indelman

Abstract—State of the art Bayesian classification approaches typically maintain a posterior distribution over possible classes given available sensor observations (images). Yet, while these approaches fuse all classifier outputs thus far, they do not provide any indication regarding how reliable the posterior classification is, thus limiting its functionality in terms of autonomous systems and robotics. On the other hand, current deep learning based classifiers provide an uncertainty measure, thereby quantifying model uncertainty. However, they do so on a single frame basis and do not consider a sequential framework. In this paper we develop a novel approach that infers a distribution over posterior class probabilities, while accounting for model uncertainty. This distribution enables reasoning about uncertainty in the posterior classification, and therefore is of prime importance for robust classification, object-level perception in uncertain and ambiguous scenarios, and for safe autonomy in general. The distribution of the posterior class probability has no known analytical solution, thus we propose to approximate this distribution via sampling. We evaluate our approach in simulation and using real images fed into a convolutional neural network classifier.

Index Terms—Deep Learning in Robotics and Automation; Recognition

I. INTRODUCTION

CLASSIFICATION and object recognition is a fundamental problem in robotics and computer vision, which plays a significant role in numerous problem domains and applications, including semantic mapping, object-level SLAM, active perception and autonomous driving. Yet, reliable and robust classification in uncertain and ambiguous scenarios is challenging, as object classification is often viewpoint dependent, influenced by environmental visibility conditions such as lighting, clutter, image resolution and occlusions, and limited by the classifier’s training set. In these challenging scenarios, classifier output can be sporadic and highly unreliable. Moreover, approaches that rely on most likely class observations can easily break, as these observations are treated equally regardless if the most likely class has high probability or not, potentially giving large significance to ambiguous observations.

Indeed, modern (deep learning based) classifiers provide much richer information that is being discarded by resorting to only most likely observations. Current convolutional neural network (CNN) classifiers provide not only vector of class probabilities (i.e. probability for each class), but, recently, also output an uncertainty measure, quantifying how (un)certaint each of these probabilities is.

Even though CNN-based classification achieved remarkable results in the last few years, as with any data driven method, actual performance heavily depends on the training set. In particular, if the classified object is represented poorly in the training set, the classification result will be unreliable and vary greatly with slightly different classifier weights. This variation is referred to as model uncertainty. High model uncertainty tends to arise from input that is far from the classifier’s training set, which could be caused by an object not being in the training set or by occlusions. In addition, classification, where each frame is treated separately, is influenced by environmental conditions such as lighting and occlusions. Consequently, it can provide unstable classification results. Various methods were proposed to compute model uncertainty from a single image (see [1]–[3]).

To address this problem, various Bayesian sequential classification algorithms (e.g. [4]–[10]) that maintain a posterior class distribution were developed; however, none of these approaches address model uncertainty. Crucially, while posterior class distribution fuses all classifier outputs thus far, it does not provide any indication regarding how reliable the posterior classification is. In Bayesian inference over continuous random variables (e.g. SLAM problem), this would correspond to getting the maximum a posteriori solution without providing the uncertainty covariances! Clearly, this is highly undesired, in particular in the context of safe autonomous decision making (e.g. in robotics, or for self driving cars), where a key question is when should a decision be made given available data thus far (see e.g. [11]). On the other hand, existing approaches that account for model uncertainty do not consider sequential classification. As a consequence, none of the existing approaches reason about the posterior uncertainty, given images thus far.

In this paper we address this gap and propose to maintain a distribution over posterior class probabilities while accounting for model uncertainty. This distribution enables reasoning about uncertainty in posterior classification, which is crucial for robust classification, and for safe autonomy in general. In particular, we derive equations to sequentially update the distribution over posterior class probabilities. We evaluate our approach in simulation, and using real images fed into a deep learning classifier.
II. RELATED WORK

Several sequential classification algorithms were developed in recent years. Coates and Y. Ng [4] proposed a method that updates posterior class probability by multiplying prior class probability with a new classification measurement, considering an object detection problem. Omidshafiei et al. [5] mention the Static State Bayes Filter (SSBF) algorithm that extends the work by Coats and Y. Ng for multiple possible classes. It assumes the prior, posterior and likelihood all categorically distributed. Patten et al. [10] used a method similar to SSBF in the context of active classification. Atanasov et al. [6] proposed a method that updates categorically distributed posterior pairs of candidate object classes and orientations. This approach utilizes a viewpoint dependent observation model. Omidshafiei et al. [5] developed the hierarchical Bayesian noise inference (HBNI) algorithm. At each time step, the algorithm updates class probabilities with the likelihood of the soft-max classifier output modeled by a Dirichlet distribution, with a noise parameter for each possible class. Mu et al. [7] utilizes a Dirichlet prior and most likely classifier observations, while addressing the challenging problem of data association.

Works by Teacy et al. [8] and Velez et al. [9] propose methods to update a viewpoint dependent classifier model to update posterior class probability. Both works are presented in the context of active classification. While Velez et al. only consider correlation aspects of past observations, Teacy et al. also consider correlation aspects for future observations. Crucially, these approaches do not consider model uncertainty, i.e. how reliable is the classifier output.

Recently, several works developed methods for computing model uncertainty for deep learning applications. Grimmett et al. [12] suggested using normalized entropy of class probability as a measure of classification uncertainty. However that approach does not consider model uncertainty. Gal and Ghahramani [1] [2] proposed utilizing neural network dropout to estimate model uncertainty for an input of a single image. Kendal and Cipolla [13] build upon these works, utilizing dropout to compute uncertainty in CNN-based camera relocalization. Both infer model uncertainty from a single image input only. Further, Kendal and Gal [14] analyze the major two types of uncertainty: epistemic uncertainty or model uncertainty, and aleatoric uncertainty that captures noise inherent in observations. Mishkov and Julier [3] compare between multiple methods to predict uncertainty in classifier output, using Hybrid Monte Carlo (HMS) as a baseline. They found that Stochastic Gradient Langevin Dynamics (SGLD) and Dropout (see [2]) methods performed the best in terms of accuracy. Yet, these works consider classification given a single image frame, as opposed to Bayesian sequential classification given multiple images that we consider herein.

III. NOTATION AND PROBLEM FORMULATION

Consider a robot observing a single object from multiple viewpoints, aiming to infer its class while quantifying uncertainty in the latter. Each class probability vector is $\gamma_k \doteq \left[ \gamma_k^1, \ldots, \gamma_k^i, \ldots, \gamma_k^M \right]$, where $M$ is the number of candidate classes. Each element $\gamma_k^i$ is the probability of object class $c$ being $i$ given image $z_k$, i.e. $\gamma_k^i \equiv P(c = i | z_k)$, while $\gamma_k$ resides in the $(M - 1)$ simplex such that

$$\gamma_k^i \geq 0 \quad ||\gamma_k||_1 = 1.$$

Existing Bayesian sequential classification approaches do not consider model uncertainty, and thus maintain a posterior distribution $\lambda_k$ for time $k$ over $c$,

$$\lambda_k \equiv P(c | \gamma_{1:k}),$$

given history $\gamma_{1:k}$ obtained from images $z_{1:k}$. In other words, $\lambda_k$ is inferred from a single sequence of $\gamma_k$, where each $\gamma_k$ for $t \in [1, k]$ corresponds to an input image $z_t$. However, the posterior class probability $\lambda_k$ by itself does not provide any information regarding how reliable the classification result is due to model uncertainty. For example, a classifier output $\gamma_k$ may have a high score for a certain class, but if the input is far from the classifier training set the result is not reliable and may vary greatly with small changes in the scenario and classifier weights.

In this paper we wish to reason about model uncertainty, i.e. quantify how “far” an input image $z_t$ is from the training set $D$ by modeling the distribution $P(\gamma_k | z_t, D)$. Given a training set $D$ and classifier weights $w$, the output $\gamma_t$ is a deterministic function of input $z_t$ for all $t \in [1, k]$:

$$\gamma_t = f_w(z_t),$$

where the function $f_w$ is a classifier with weights $w$. However, $w$ are stochastic given $D$, thus inducing a probability $P(w | D)$ and making $\gamma_t$ a random variable. Gal and Ghahramani [2] showed that an input far from the training set will produce vastly different classifier outputs for small changes in weights. Unfortunately, $P(w | D)$ is not given explicitly. To combat this issue, Gal and Ghahramani [2] proposed to approximate $P(w | D)$ via dropout, i.e. sampling $w$ from another distribution closest to $P(w | D)$ in a sense of KL divergence. Practically, we run an input image $z_t$ through a classifier with dropout multiple times to get many different $\gamma_t$’s for corresponding $w$ realizations, creating a point cloud of class probability vectors. Note that every distribution in this paper is dependent on the training set $D$, so we omit it from further expressions to avoid clutter.

In this paper, a class-dependent likelihood $P(\gamma_k | c = i)$, referred as a classifier model, is utilized. We use a Dirichlet distributed classifier model with a different hyperparameter vector $\theta_i \in \mathbb{R}^{M \times 1}$ per class $i \in [1, M]$, rewriting $P(\gamma_k | c = i)$ as:

$$P(\gamma_k | c = i) = Dir(\gamma_k; \theta_i).$$

This distribution is the conjugate prior of the categorical distribution, thus it supports class probability vectors, particularly $\gamma_k$. Sampling from Dirichlet distribution necessarily satisfies conditions (1), unlike other distributions such as...
Gaussian. The probability density function (PDF) of the above distribution is as follows:

$$\text{Dir}(\gamma_k; \theta_i) = C(\theta_i) \prod_{j=1}^{M} \left( \gamma_k^j \right)^{\theta_i^j - 1},$$

(5)

where $C(\theta_i)$ is a normalizing constant dependent on $\theta_i$, and $\theta_i^j$ is the $j$-th element of vector $\theta_i$. To shorten notation, we will write this likelihood as:

$$P(\gamma_k|c = i) = \mathcal{L}_i(\gamma_k), \quad P(c = i) = \mathcal{L}_i.$$  

(6)

We denote the likelihood vector as $\mathcal{L}(\gamma_k) = [\mathcal{L}_1(\gamma_k) \cdots \mathcal{L}_M(\gamma_k)]$. For simplicity, we consider these hyperparameter vectors to be known or inferred. Furthermore, in this paper we assume an uninformative prior $P(c = i) = 1/M$.

We must distinguish between the classifier model $\mathcal{L}_i(\gamma_k)$, and the model uncertainty derived from $P(\gamma_k|z_k)$ for class $i$ and time step $k$. The classifier model $\mathcal{L}_i(\gamma_k)$ is the likelihood of a single $\gamma_k$ given a class hypothesis $i$; it is computed prior to the scenario for each class from the training set, and it is assumed constant within the scenario. On the other hand, $P(\gamma_k|z_k)$ is the probability of $\gamma_k$ given an image $z_k$, and is computed during the scenario. Note that if the true object class is $i$ and it is “close” to the training set, the probabilities $P(\gamma_k|z_k)$ and $\mathcal{L}_i(\gamma_k)$ will be “close” to each other as well.

A key observation is that $\lambda_k$ is a random variable, as it depends on $\gamma_{1:k}$ (see Eq. (2)) while each $\gamma_t$, with $t \in [1, k]$, is a random variable distributed according to $P(\gamma_t|z_t, D)$. Thus, rather than maintaining the posterior Eq. (2), our goal is to maintain a distribution over posterior class probabilities for time $k$, i.e.

$$P(\lambda_k|z_{1:k}).$$  

(7)

This distribution allows to calculate the posterior class distribution, $P(c|z_{1:k})$, via expectation

$$P(c = i|z_{1:k}) = \int_{\lambda_k} P(c = i|\lambda_k, z_{1:k}) P(\lambda_k|z_{1:k}) d\lambda_k$$

$$= \int_{\lambda_k} P(c = i|\lambda_k) P(\lambda_k|z_{1:k}) d\lambda_k = \mathbb{E}[\lambda_k]$$

(8)

where we utilized the identity $P(c = i|\lambda_k) = \lambda_k^i$.

Moreover, as will be seen, Eq. (7) allows to quantify the posterior uncertainty, thereby providing a measure of confidence in the classification result given all data thus far. At this point, it is useful to summarize our assumptions:

1) A single object is observed multiple times.
2) $P(\gamma_t|z_t, D)$ is approximated by a point cloud $\{\gamma_t\}$ for each image $z_t$.
3) An uninformative prior for $P(c = i)$.
4) A Dirichlet distributed classifier model with with designated parameters for each class $c \in [1, \ldots, M]$. These parameters are constant and given (e.g. learned).

IV. APPROACH

We aim to find a distribution over the posterior class probability vector $\lambda_k$ for time $k$, i.e. $P(\lambda_k|z_{1:k})$. First, $\lambda_k$ is expressed given some specific sequence $\gamma_{1:k}$. Using Bayes’ law:

$$\lambda_k^i = P(c = i|\gamma_{1:k}) \propto P(c = i|\gamma_{1:k-1})P(\gamma_k|c = i, \gamma_{1:k-1}).$$  

(9)

We assume, for simplicity, classifier outputs are statistically independent$^1$ and re-write Eq. (9) as

$$\lambda_k^i \propto P(c = i|\gamma_{1:k-1})P(\gamma_k|c = i).$$  

(10)

Per the definition for $\lambda_{k-1}$ (Eq. (2)) and $P(\gamma_k|c = i)$ (Eq. (6)), $\lambda_k^i$ assumes the following recursive form:

$$\lambda_k^i \propto \lambda_{k-1}^i \mathcal{L}_i(\gamma_k).$$  

(11)

We now recall that $\gamma_t$ (for each time step $t \in [1, k]$) is a random variable, making also $\lambda_{k-1}^i$ and $\lambda_k^i$ random variables. Thus, our problem is to infer $P(\lambda_k|z_{1:k})$, where, according to Eq. (11), for each realization of the sequence $\gamma_{1:k}$, $\lambda_k$ is a function of $\lambda_{k-1}^i$ and $\gamma_k$.

We present our approach in Algorithm 1. At each time step $t$, a new image $z_t$ is classified using multiple forward passes through a CNN with dropout, yielding a point cloud $\{\gamma_t\}$. Each forward pass gives a probability vector $\gamma_t \in \{\gamma_t\}$, which is used to compute the class likelihood $\mathcal{L}(\gamma_t)$, that is modeled as a Dirichlet distribution. In addition, we have a point cloud $\{\lambda_{t-1}\}$ from the previous step. We multiply all possible pairs of $\lambda_{t-1}$ and $\mathcal{L}(\gamma_t)$, as in Eq. (11). Finally $N_{ss,n}$ pairs are chosen for the next step, in a sub-sampling algorithm that will be detailed in Section IV-B. We eventually get a point cloud $\{\lambda_t\}$ that approximates $P(\lambda_t|z_{1:k})$.

Algorithm 1: $P(\lambda_k|z_{1:k})$ inference algorithm with sub-sampling.

1 Inputs:
2 $z_{1:k}$: $k$ images of an object.
3 $P(c = i) \forall i = 1, \ldots, M$: a prior for object class.
4 $\mathcal{L}_i \forall i = 1, \ldots, M$: a classifier model.
5 $N_{ss,n}$ maximum points per time step
6 Outputs:
7 $P(\lambda_k|z_{1:k})$:
8 Initialize: $\lambda_0 = P(c = i)$
9 for $t = 1 : k$ do
10 Classify image $z_t$, and produce a point cloud $\{\gamma_t\}$.
11 for All possible $\gamma_t$ and $\lambda_{t-1}$ pairs: do
12 $\lambda_t^i \propto \mathcal{L}(\gamma_t)\lambda_{t-1}^i$
13 end
14 Select randomly $N_{ss,n}$ pairs to form $\{\lambda_t\}$
15 end
16 return $\{\lambda_t\}$

We need to initialize the algorithm for the first image. Recalling Eq. (2), we define $\lambda_1^i$ (first image) for class $i$ and time $k = 1$ as:

$$\lambda_1^i = P(c = i|\gamma_1).$$

(12)

$^1$In this paper we do not consider viewpoint-dependent classifier models and thus model all $\gamma_{1:k}$ to be statistically independent from each other. We refer to our recent work [15] that relaxes this assumption while investigating complimentary aspects to this work.
Using Bayes law:
\[ P(c = i | \gamma_1) = \frac{P(\gamma_1 | c = i)P(c = i)}{P(\gamma_1)} \]  
(13)

where \( P(c = i) \) is a prior probability of class \( i \), \( P(\gamma_1) \) serves as a normalizing term, and \( P(\gamma_1 | c = i) \) is the classifier model for class \( i \). Per definition Eq. (6), Eq. (13) can be written as:
\[ \lambda_i^1 \propto P(c = i) \mathcal{L}_i(\gamma_1), \]  
(14)

thus \( \lambda_i^1 \) is a function of prior \( P(c = i) \) and \( \gamma_1 \), and in the subsequent steps we can use the update rule of Eq. (11) to infer \( P(\lambda_k | z_{1:k}) \).

Remark: There is a numerical issue where \( \lambda_k^i \) for sufficiently large \( k \) can practically become 0 or 1, preventing any possible change for future time steps. In our implementation, we overcome this by calculating \( \log \lambda_k^i \) instead of \( \lambda_k^i \).

In the next section we discuss the properties of \( P(\lambda_k | z_{1:k}) \), analyze the corresponding posterior uncertainty versus time, and consider two inference approaches that approximate this PDF.

A. Inference over the Posterior \( P(\lambda_k | z_{1:k}) \)

In this section we consider how the distribution \( P(\lambda_k | z_{1:k}) \) develops and seek to find an inference method to track this distribution over time. As discussed in Section III, we consider all \( \gamma_t \) as random variables; hence, according to Eq. (11), \( P(\lambda_k | z_{1:k}) \) accumulates all model uncertainty data from all \( P(\gamma_t | z_k) \) up until time step \( k \), with \( t \in [1, k] \).

Fig. 1 illustrates an example for inference of \( P(\lambda_k | z_{1:k}) \) from \( P(\gamma_k | z_k) \) and \( P(\lambda_k - 1 | z_{1:k}) \) using a known classifier model, considering three possible classes. Fig. 1a-1c present example distributions for the classifier model. Fig. 1d presents a point cloud that describes the distribution of \( \lambda_{k-1} \). Fig. 1e presents \( P(\gamma_k | z_k) \) represented by a point cloud of \( \gamma_k \) instances. Each \( \gamma_k \) is projected via \( \mathcal{L}(\gamma_k) \) to a different cloud in the simplex, presented in Fig. 1f. Finally, based on Eq. (11), the multiplication of points from Fig. 1d and 1f creates a \( \{\lambda_k\} \) point cloud, shown in Fig. 1g. In the presented scenario, the spread of \( \{\lambda_k\} \) (Fig. 1g) point cloud was smaller than \( \{\lambda_{k-1}\} \) (Fig. 1d), because both point clouds \( \{\lambda_{k-1}\} \) and \( \{\mathcal{L}(\gamma_k)\} \) are near the same simplex edge. In general, classifier models with large parameters (see Eq. 5) create \( \{\mathcal{L}(\gamma_k)\} \) point clouds that are closer to the simplex edge. In turn, the \( \{\lambda_k\} \) point cloud (updated via Eq. (11)) will converge faster to a single simplex edge.

In this paragraph we discuss the behavior of \( P(\lambda_k) \), dependent on both \( \lambda_{k-1} \) and \( \gamma_k \). The spread of \( \{\lambda_k\} \) is indicative of accumulated model uncertainty, and is dependent on the expectation and spread of both \( \{\lambda_{k-1}\} \) and \( \{\gamma_k\} \). For specific realizations of \( \lambda_{k-1} \) and \( \gamma_k \), as seen in Eq. (11), \( \lambda_k \) is a multiplication of \( \lambda_{k-1} \) and \( \mathcal{L}_i(\gamma_k) \). Therefore, when \( \mathcal{L}(\gamma_k) \) is within the simplex center, i.e. \( \mathcal{L}_i(\gamma_k) = \mathcal{L}_j(\gamma_k) \) for all \( i, j = 1, ..., M \), the resulting \( \lambda_k \) will be equal to \( \lambda_{k-1} \).

On the other hand, when \( \mathcal{L}(\gamma_k) \) is at one of the simplex' edges, its effect on \( \lambda_k \) will be the greatest. Expanding to the probability \( P(\lambda_k | z_{1:k}) \), there are several cases to consider.

If \( P(\lambda_{k-1} | z_{1:k-1}) \) and \( \{\mathcal{L}(\gamma_k)\} \) “agree” with each other, i.e. the highest probability class is the same, and both are far enough from the simplex center, the resulting \( P(\lambda_k | z_{1:k}) \) will have a smaller spread compared to \( P(\lambda_{k-1} | z_{1:k-1}) \) and its expectation will have the dominant class with a high probability. On the other hand, if \( P(\lambda_{k-1} | z_{1:k-1}) \) and \( \{\mathcal{L}(\gamma_k)\} \) “disagree” with each other, i.e. they are close to the same simplex corner, the spread of \( P(\lambda_k | z_{1:k}) \) will become larger; an example for this case is illustrated in Fig. 2. In practice such a scenario can occur when an object of a certain class is observed from a viewpoint where it appears like a different class. If both \( P(\lambda_{k-1} | z_{1:k-1}) \) and \( \{\mathcal{L}(\gamma_k)\} \) are near the simplex center, the spread of \( P(\lambda_k | z_{1:k}) \) will increase as well. Finally, if only one of \( P(\lambda_{k-1} | z_{1:k-1}) \) and \( \{\mathcal{L}(\gamma_k)\} \) is near the simplex center, \( P(\lambda_k | z_{1:k}) \) will be similar to the one that is farther from the simplex center.

From \( P(\lambda_k | z_{1:k}) \) we can infer the expectation \( \mathbb{E}(\lambda_k) \) (computed as in Eq. (8)) and covariance matrix \( \text{Cov}(\lambda_k) \) of \( \lambda_k \). As \( \mathbb{E}(\lambda_k) \) takes into account model uncertainty from each image, unlike existing approaches (e.g. [5]), we can achieve a posterior classification that is more resistant to possible aliasing. The covariance matrix \( \text{Cov}(\lambda_k) \) represents the spread of \( \lambda_k \), and in turn accumulates the model uncertainty from all images \( z_{1:k} \). In general, lower \( \text{Cov}(\lambda_k) \) values represent smaller \( \lambda_k \) spread, and thus higher confidence with
the classification results. Practically, this can be used in a
decision making context, where higher confidence answers
are preferred. In this paper we compare between values of
\(V\text{ar}(\lambda_k^i)\) for all classes \(i = 1, \ldots, M\), as it is simpler to
describe the uncertainty per class.

There is a correlation between the expectation \(\mathbb{E}(\lambda_k^i)\) and
\(\text{Cov}(\lambda_k^i)\). The largest covariance values will occur when
\(\mathbb{E}(\lambda_k^i)\) is at the simplex’ center. In particular, it is not difficult
to show that the highest possible value for \(V\text{ar}(\lambda_k^i)\) for
any \(i\) is 0.25; it can occur when \(\lambda_k^i = 0.5\). In general, if
\(\mathbb{E}(\lambda_k)\) is close to the simplex’ boundaries, the uncertainty
is lower. Therefore, to reduce uncertainty, \(\mathbb{E}(\lambda_k)\) should be
concentrated in a single high probability class.

To the author’s knowledge, the expression \(P(\lambda_k|z_{1:k})\),
where the expression for \(\lambda_k\) is described in Eq. (11), has no
known analytical solution. The next most accurate method
available is multiplying all possible permutations of point
clouds \(\{\gamma_t\}\), for all images at times \(t \in [1, k]\). This method is
computationally intractable as the number of \(\lambda_k\) points grows
exponentially. In the next section we propose a simple sub-
sampling method to approximate this distribution and keep
computational tractability.

B. Sub-Sampling Inference

As mentioned previously in section III, each measurement
we receive a cloud of \(N_k\) probability vectors \(\{(\gamma_k)^n\}_{n=1}^{N_k}\).
Each probability vector is projected via the classifier model
to a different point with the simplex, which provides a new
point cloud \(\{L(\gamma_k)^n\}_{n=1}^{N_k}\). We assume that \(P(\lambda_{k-1}|z_{1:k-1})\)
is described by a cloud of \(N_{k-1}\) points. Given the data for
\(\gamma_k\) and \(\lambda_{k-1}\), the most accurate approximation to \(P(\lambda_k|z_{1:k})\)
is given by multiplying all possible pairs of \(\lambda_{k-1}\) and \(L(\gamma_k)\).
Thus, \(P(\lambda_k|z_{1:k})\) is described with a cloud of \(N_{k-1} \times N_k\)
points. For subsequent steps the cloud size grows exponentially,
making it computationally intractable. We address this
problem by randomly sampling from the point cloud for \(\lambda_k\)
a subset of \(N_{ss,n}\) points and use them for the next time step.
In practice, we keep \(N_{ss,n}\) constant across all time steps, see
line 16 in Algorithm 1.

V. EXPERIMENTS

In this section we study our method in simulation and using
real images fed into an AlexNet [16] CNN classifier. We used a
PyTorch implementation of AlexNet for classification, and
Matlab for sequential data fusion. Our hardware is an Intel
i7-7700HQ CPU running at 2.8GHz, and 16GB of RAM. We
compare between four different approaches:

1) Method-\(P(c|z_{1:k})\)-w/o-model: Naive Bayes that
infers the posterior of \(P(c|z_{1:k})\) where the classifier
model is not taken into account (SSBF in [5]).

2) Method-\(P(c|z_{1:k})\)-w-model: A Bayesian approach
that infers the posterior of \(P(c|z_{1:k})\) and uses a clas-
sifier model; essentially using Eq. (11) with a known
classifier model.

3) Method-\(P(\lambda_k|z_{1:k})\)-AP: Inference of \(P(\lambda_k|z_{1:k})\)
multiplying all possible combinations of \(\lambda_{k-1}\) and
\(\mathcal{L}(\gamma_k)\). Note that the number of combinations grows
exponentially with \(k\), thus the results are presented up
until \(k = 5\).

4) Method-\(\mathbb{P}(\lambda_k|z_{1:k})\)-SS: Inference of \(\mathbb{P}(\lambda_k|z_{1:k})\) using
the sub-sampling method.

Our proposed approaches are 3 and 4.

A. Simulated Experiment

This experiment is a simulation to demonstrate the algo-
rithm’s performance. This simulation is designed to emulate
a scenario of a robot traveling in a predetermined trajectory
and observing an object from multiple viewpoints. This
object’s class is one of three possible candidates. We infer
the posterior over \(\lambda\) and display the results as expectation
\(\mathbb{E}(\lambda_k^i)\) and standard deviation per class \(i\):

\[
\sigma_i = \sqrt{\text{Var}(\lambda_k^i)}. \quad (15)
\]

This simulation is a study on the effect of using classifier
model in the inference for highly ambiguous measurements.
In addition, we analyze the uncertainty behavior for this
scenario. We use a categorical uninformative prior of \(P(c =
i) = 1/M\) for all \(i = 1, \ldots, M\).

Each of the three classes has its own (known) classifier
model Eq. (16), as shown in Figures 3a-3c. This classifier
model is assumed Dirichlet distributed with the following
hyperparameters \(\theta_i\) for all \(i \in [1, 3]\):

\[
\begin{align*}
\theta_1 &= [6 1 1] \\
\theta_2 &= [2 7 2] \\
\theta_3 &= [1 1 5.2].
\end{align*}
\]

In this experiment the true class is 3. These hyperparameters
were selected to simulate a case where the \(\gamma\) measurements
are spread out (corresponds to ambiguous appearance of
the class), thus leading to incorrect classification without
a classifier model. The classifier model for this class \(\mathcal{L}_3\)
predicts highly variable \(\gamma\)’s using the training data (Fig. 3c).
The \(\{\gamma_t\}\) point clouds for each \(t \in [1, k]\) are different from
each other (Fig. 3e), representing an object photographed by
a robot from multiple viewpoints.

We simulate a series of 5 images. Each image at time
step \(t\) has its own different \(P(\gamma_t|z_t)\). For the approaches that
infer \(P(c|z_{1:k})\), we sample a single \(\gamma_t\) per image \(z_t\) for all
\(t \in [1, k]\) (Fig. 3f, also we present the \(\gamma_t\) order). This sample
simulates the usual single classifier forward pass that is used.
For our approaches we sample 10 \(\gamma_t\)’s from each \(P(\gamma_t|z_t)\),
except for the first step \(t = 1\) where we sample 100 \(\gamma_t\)’s. For
Method-\(P(\lambda_k|z_{1:k})\)-SS each \(\{\lambda_t\}\) point cloud is capped
at 100 points. The expectation of these generated measurements
are presented in Fig. 3d, along with the cloud order. In Fig. 3e
\(\{\gamma_t\}\) point clouds for three different \(t\)’s are presented in
distinct colors. The input for methods 1 and 2 is shown in
Fig. 3f, and some of the input for methods 3 and 4 is shown in
Fig. 3e.

Fig. 4 presents results obtained with our algorithm, in
terms of expectation \(\mathbb{E}(\lambda_k^i)\) and \(\sqrt{\text{Var}(\lambda_k^i)}\) for each class \(i\),
as a function of classifier measurements. In Fig. 4a and 4b we use a single sampled $\gamma_i$ for $z_t$ (see Fig. 3f), while in Fig. 4c and 4d we create a $\{\gamma_i\}$ point cloud for $z_t$ (see Fig. 3e). In Fig. 4a and 4b results for Method-$P(c|z_{1:k})$-w/o-model and Method-$P(c|z_{1:k})$-w-model respectively. Without classifier model the results generally favor class 2 incorrectly, as the measurements tend to give that class the higher chances. With classifier models the results favor class 3, the correct class. Because the classifier model for class 3 is more spread out than for the other classes, $\gamma_i$’s in the simplex middle (as in Fig. 3e) have higher $L_3(\gamma)$ values than $L_1(\gamma)$ and $L_2(\gamma)$. While method Method-$P(c|z_{1:k})$-w-model gives eventually correct classification results, it does not account for model uncertainty, i.e. uses a single classifier output $\gamma$ obtained with a forward run through the classifier without dropout. In this simulation we sample a single $\gamma$ from each point cloud to simulate this forward run.

Figs. 4c and 4d present the results for Method-$P(\lambda_k|z_{1:k})$-SS and Method-$P(\lambda_k|z_{1:k})$-AP, expectation and standard deviation respectively. Throughout the scenario class 3 has the highest probability correctly, and the deviation drops as more measurements are introduced. Compared to Fig. 4b where class 3 has high probability only at time step $t = 3$, in Fig. 4c class 3 is the most probable from time step $t = 1$. Both Method-$P(\lambda_k|z_{1:k})$-SS and Method-$P(\lambda_k|z_{1:k})$-AP behave similarly. Note that class 1 has much smaller deviation than the other two because its probability is close to 0 through the entire scenario.

Fig. 5 presents the development of $\{\lambda_k\}$ point clouds for Method-$P(\lambda_k|z_{1:k})$-SS at different time steps. Those figures show the gradual decrease in $\{\lambda_k\}$’s spread, coinciding with the corresponding standard deviation at Fig. 4d.

### B. Experiment with Real Images

Our algorithm is tested using a series of images of an object (space heater) with conflicting classifier outputs when observed from different viewpoints. This corresponds to a scenario where a robot in a predetermined path observes an object that is obscured by occlusions and different lighting conditions. The experiment presents our algorithm’s robustness to these difficulties in classification, and addressing them is important for real-life robotic applications.

The database photographed is a series of 10 images of a space heater with artificially induced blur and occlusions. Each of the images is run through an AlexNet convolutional neural network [16] with 1000 possible classes. Similar to Section V-A, we use an uninformative classifier prior on $P(c)$ with $P(c = i) = 1/M$ for all $i = 1, ..., M$ classes. Our algorithm is used to fuse the classification data into a posterior distribution of the class probability and infer deviation for each class. As in the previous section, we
present results with and without classifier model. Fig. 6 presents four of the dataset images, exhibiting occlusions, blur and different colored filters in a monotone environment.

![Image](a) (b) (c) (d)

Figure 6: Four of the 10 images used in the dataset with occlusions and different viewpoints. Blurring and colored filters were introduced to some images artificially.

We compare between the same methods that are used in the previous sub-sections. For Method\(-\mathbb{P}(c|z_{1:k})\)-w/o-model and Method\(-\mathbb{P}(c|z_{1:k})\)-w-model, we forward the images through the classifier without dropout and use a single output $\gamma$ for each image. For Method\(-\mathbb{P}(\lambda_k|z_{1:k})\)-SS, we run each image 10 times through the classifier with dropout, producing a point cloud $\{\gamma\}$ per image. The cap for number of $\lambda_k$ points with Method\(-\mathbb{P}(\lambda_k|z_{1:k})\)-SS is 100. For Method\(-\mathbb{P}(\lambda_k|z_{1:k})\)-AP method, we present results only for the first five images as the calculations become infeasible due to the exponential complexity.

As AlexNet has 1000 possible classes (one of them is "Space Heater"), it is difficult to clearly present results for all of them. Because we wish to compare to the most likely classes, we select 3 likely classes by averaging all $\gamma$ classifier outputs and selecting the three with highest probability. The probabilities for those classes are then normalized, and utilized in the scenario. All other classes outside those three are ignored. We require a classifier model for each class; assuming the classifier model is Dirichlet distributed, we classified multiple images unrelated to the scenario for each class with the same AlexNet classifier but without dropout. The classifier produced multiple $\gamma$’s, one per image, and via a Maximum Likelihood Estimator [17] we inferred the Dirichlet hyperparameters for each class $i \in [1, 3]$. The classifier model $\mathbb{P}(\gamma_k|c = i) = Dir(\gamma_k; \theta_i)$ was used with the following hyperparameters $\theta_i$:

$$
\begin{align*}
\theta_1 &= [5.103, 1.699, 1.239] \\
\theta_2 &= [0.143, 208.7, 5.31] \\
\theta_3 &= [0.993, 14.31, 25.21] 
\end{align*}
$$

In this experiment, class 1 is the correct class (i.e. "Space Heater”). Fig. 7 presents the simplex representation of the classifier model per class, and a normalized simplex of classifier outputs for three high probability classes, similarly to Fig. 3. The classifier model for class 1 is much more spread than the other two (Fig. 7a), therefore the likelihood of measurements within a larger area will be higher for this class. Interestingly, the classifier model for class 3 predicts $\mathbb{P}(\gamma_k|c = 3)$ will be between classes 2 and 3 (Fig. 7c). Fig. 7e presents 4 of the 10 $\{\gamma_t\}$ point clouds used in the scenario. Fig. 7d presents the expectation of each $\{\gamma_t\}$ point cloud for $t \in [1, 10]$. Fig. 7f presents classifier outputs without dropout, i.e. a single $\gamma_t$ per image. Both Fig. 7d and 7f have indices that represent the images order.

![Image](a) (b) (c) (d)

Figure 7: A simplex representation of the classifier model for (a) class 1, (b) class 2, and (c) class 3. In (b), note the distribution is very tight centered at the top left corner of the simplex. (d) $\mathbb{E}(\gamma_t)$ for $t \in [1, 10]$ (i.e. 10 images). (e) Pointcloud $\{\gamma_t\}$ for 4 images. (f) CNN classifier output without dropout. In (d) and (f), image indices are shown.

Fig. 8 presents the classification results for all the methods presented. Fig. 8a and 8b show results for Method\(-\mathbb{P}(c|z_{1:k})\)-w/o-model and Method\(-\mathbb{P}(c|z_{1:k})\)-w-model respectively. Without a classifier model, i.e. the former method, incorrectly indicates class 2 as the most likely, because the classifier outputs often show class 2 as the most likely (see Fig. 7f). With a classifier model, the results jump between classes 1 and 3 as most probable. This can be explained by the likelihood vector $\mathcal{L}$ from Eq. (17) that projects the $\gamma$’s from different images approximately to different simplex edges (e.g. $\gamma_2$ and $\gamma_4$ for class 1, and $\gamma_3$ and $\gamma_5$ for class 3).

Figs. 8c and 8d present results for Method\(-\mathbb{P}(\lambda_k|z_{1:k})\)-SS and Method\(-\mathbb{P}(\lambda_k|z_{1:k})\)-AP, expectation and standard deviation respectively. Fig. 8c presents class 1 as most likely correctly in both methods from $k = 2$ onwards, and the results are smoother than in Fig. 8b because our algorithm takes into account multiple realizations of $\gamma_1$ to $\gamma_{10}$ - we recall that for each image we use a point cloud of $\gamma$’s. In addition, we can reason about the standard deviation of $\lambda_k$, representing the posterior uncertainty, as seen in Fig. 8d. Note that starting from the 4th image, the uncertainty increases, as later measurement likelihoods do not agree with $\lambda_{k-1}$ about the most likely class at those time steps, similar to the example presented in Fig. 2.

Fig. 9a presents the computational time comparison be-
and with real images fed into a deep learning classifier, providing classification posterior along with uncertainty estimates for each time instant. Future research might explore active classification aspects via belief space planning.

VII. ACKNOWLEDGMENTS

The authors thank Yuri Feldman from the Technion’s Department of Computer Science for numerous useful discussions.

REFERENCES


VI. CONCLUSIONS

We proposed a method that infers a distribution over posterior class probabilities with a measure of uncertainty using a modern, deep learning classifier. As opposed to state of the art, our approach enables quantification of uncertainty in posterior classification given all data thus far, and as such is important for robust classification, object-level perception and safe autonomy. In particular, we showed that the current posterior class probability vector is a function of the previous, accounting for model uncertainty. We used a sub-sampling approximation to obtain a point cloud that approximates the function’s distribution. Our approach is studied in simulation,