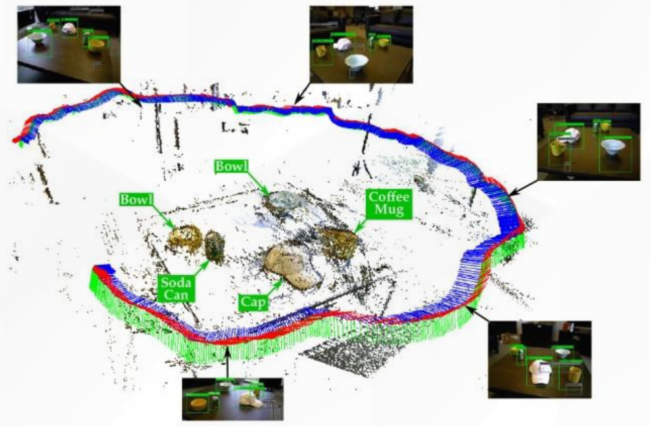
*Inference over Distribution of Posterior Class Probabilities for Reliable Bayesian Classification and Object-Level Perception*

**Vladimir Tchuiev and Vadim Indelman**

# Introduction

- **Autonomous navigation** is a widely researched topic today.
- **Reliable classification** is an important problem for autonomous navigation.
- Can we make classification **"safer"?**




- **Deep learning** based methods provide the best performance.

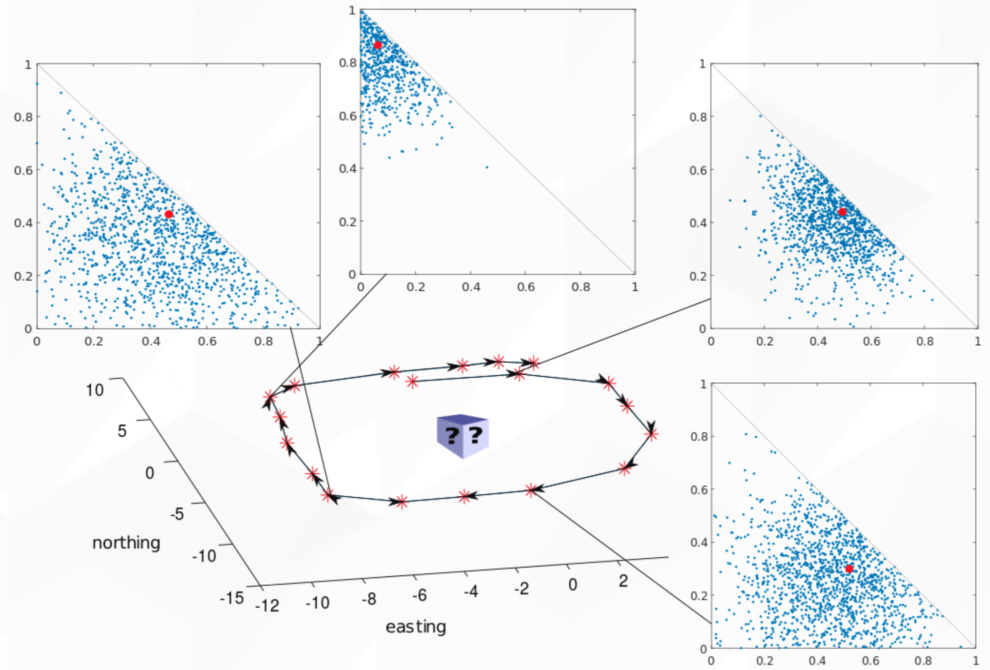
# Contribution

**Previous works**:

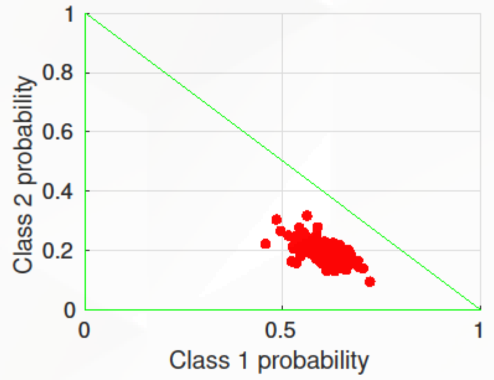
- **Sequential classification** that reasons about posterior class probability $\mathbb{P}(c|\gamma_{1:k})$.
- Infer **uncertainty** in classification from a **single image**.

**Contribution:**

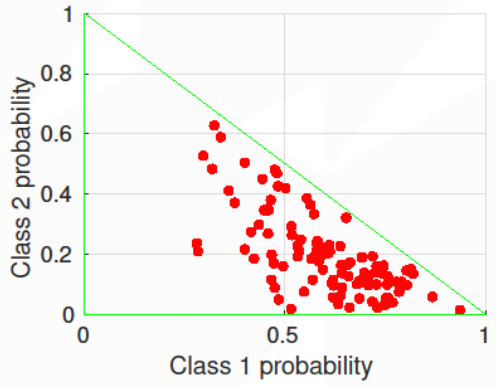
- We present a **sequential classification** method that maintains a **distribution over class probability** $\mathbb{P}(\lambda_k|z_{1:k}, D)$.
- It allows us to reason about **posterior uncertainty** given all data thus far.




- Small uncertainty:




- Large uncertainty:



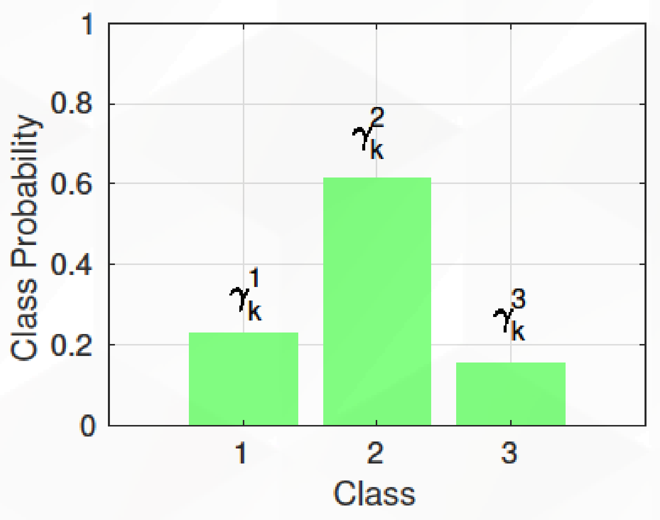
# Definitions

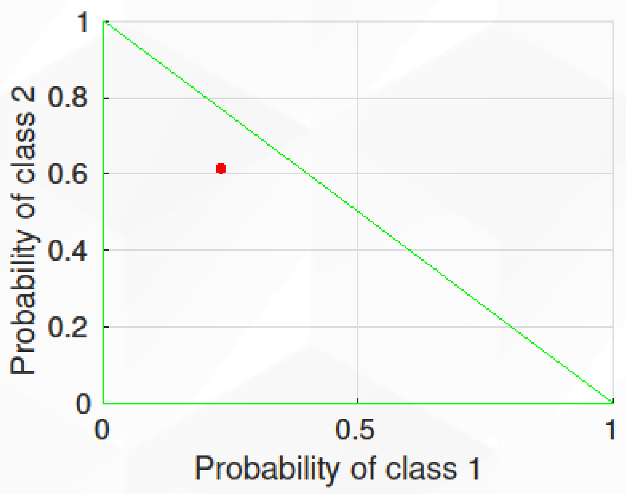
- Class probability:

$$\gamma_k^i \doteq \mathbb{P}(c=i \mid z_k, D) \qquad \gamma_k \doteq \left[\gamma_k^1,\ldots,\gamma_k^M\right]^T$$

- **Posterior** class probability:

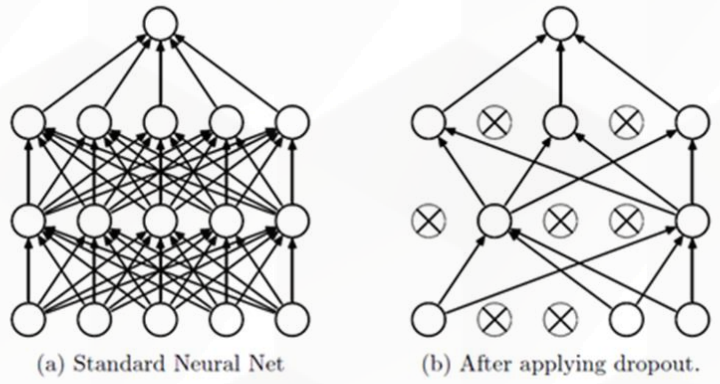
$$\lambda_k^i \doteq \mathbb{P}(c=i \mid \gamma_{1:k}) \qquad \lambda_k \doteq \left[\lambda_k^1,\ldots,\lambda_k^M\right]^T$$




# Dropout and Model Uncertainty

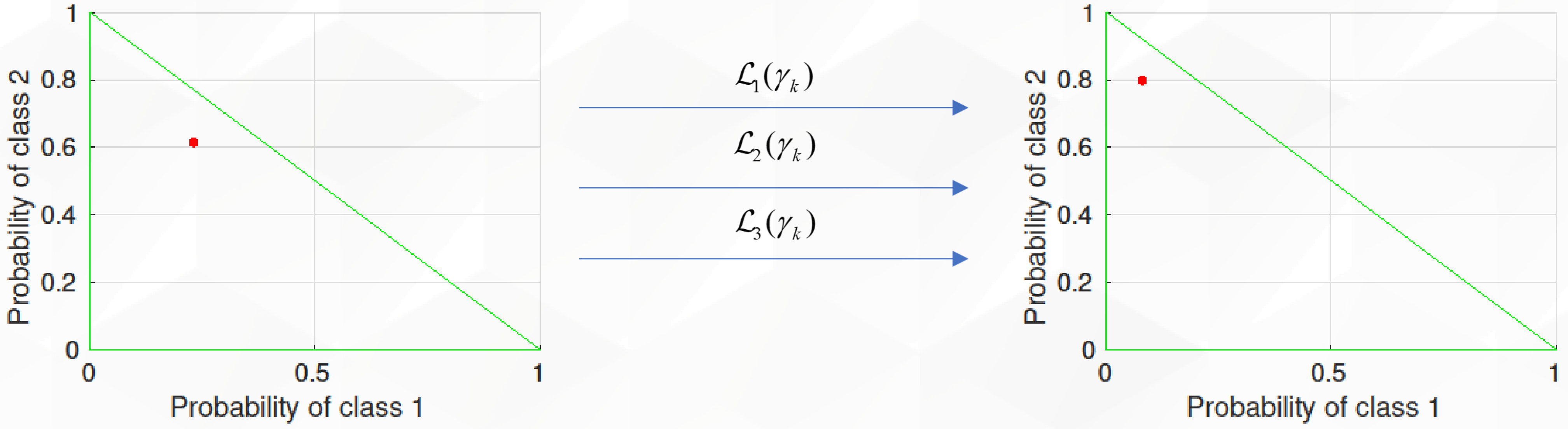
- We use a **convolutional neural network** (CNN) classifier.
- The classifier parameters $w$ are trained from a labeled example image dataset $D$.
- Given fixed weights, the classifier output is **deterministic**: $\gamma_k = f_w(z_k)$.
- Dropout randomly shuts down neurons, making $w$ stochastic, thus $\boldsymbol{\gamma_k}$ **is stochastic** as well.




- **Multiple forward passes** through a network with dropout produces a **point cloud** $\{\boldsymbol{\gamma_k}\}$ that approximates $\mathbb{P}(\gamma_k | z_k, D)$.
- **Model Uncertainty**: How 'far' is an image from training set? Approximated by a CNN via dropout at test time.

## *Assumptions*

- A **single object** is observed multiple times.
- **Classifier output** of $\{\boldsymbol{\gamma_k}\}$ that approximates $\mathbb{P}(\gamma_k|z_k,D)$.
- Uninformative prior for $\mathbb{P}(c)$.
- A Dirichlet distributed **classifier model** with known parameters.




## *Classifier Model*

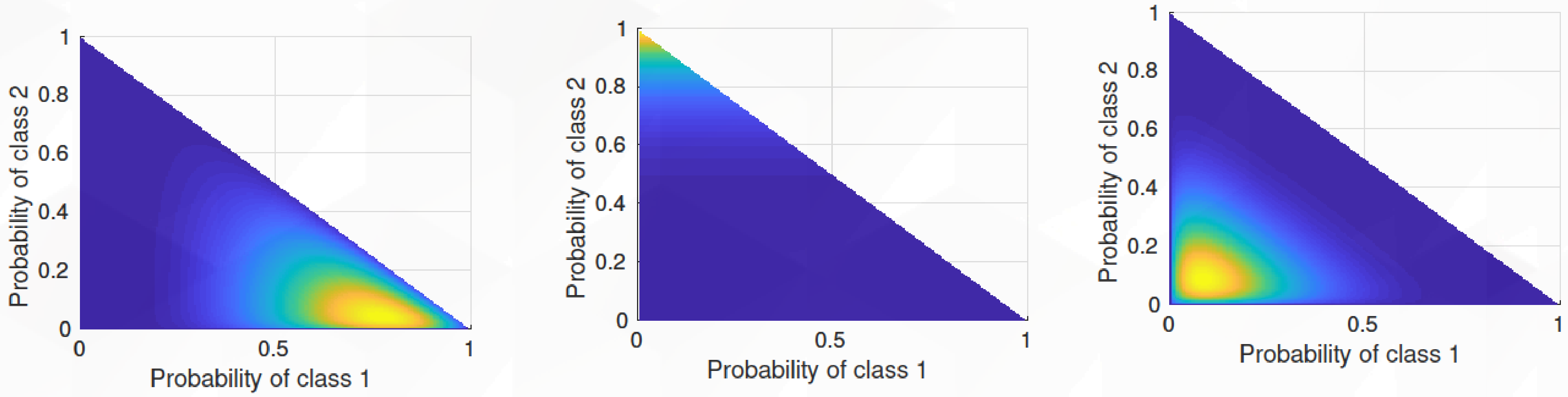
- **Likelihood** of $\gamma_k$ given object class i:

$$\mathcal{L}_i(\gamma_k) \doteq \mathbb{P}(\gamma_k \mid c=i, D) \qquad \mathcal{L}(\gamma_k) \doteq [\mathcal{L}_1(\gamma_k),\ldots,\mathcal{L}_M(\gamma_k)]^T$$

- $\mathcal{L}(\gamma_k)$ is referred to as the ***classifier model***.
- **Dirichlet distributed** in our case with a-priori known parameters $\theta^i$:

$$\mathcal{L}_i(\gamma_k) = Dir(\gamma_k;\theta^i)$$

- Probability vector $\gamma_k$ is projected via the classifier model to vector $\mathcal{L}(\gamma_k)$.
- Classifier model example:

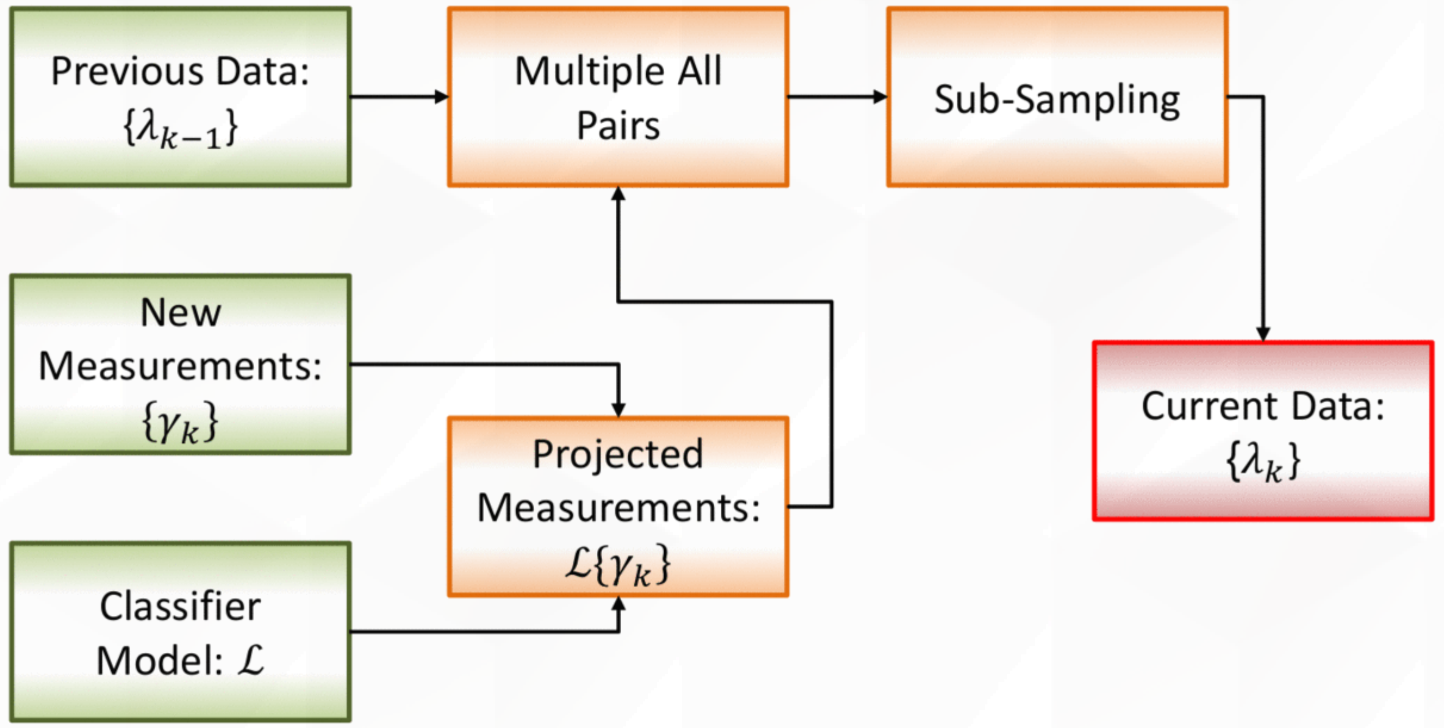


# *Our Method*

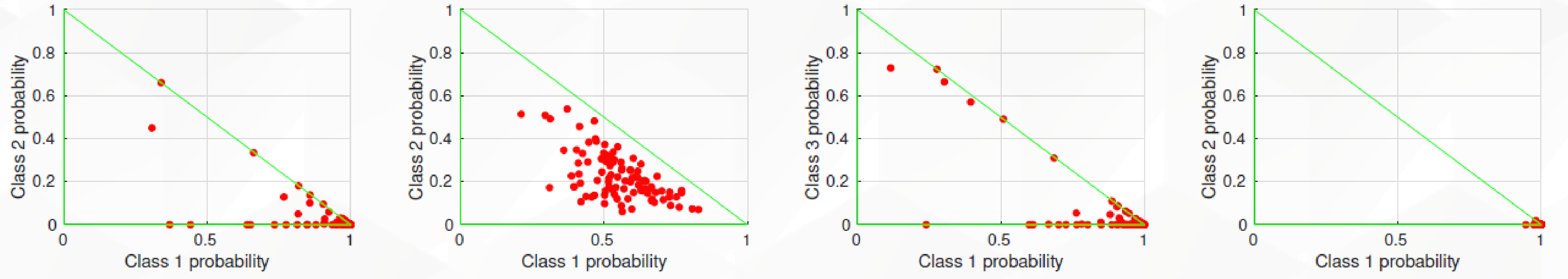
- Our goal is to **maintain** $\mathbb{P}(\boldsymbol{\lambda_k}|\boldsymbol{z_{1:k}}, \boldsymbol{D})$.
- Using Bayes rule, $\boldsymbol{\lambda_k}$ **is updated by**:

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

- All $\gamma$ up to time $k$ are **random variables**, thus $\lambda_{k-1}$ and $\lambda_k$ are **random variables**.
- We represent the distribution of each $\lambda$ by a **point cloud** $\{\lambda\}$.
- Multiplying all permutations of $\lambda_{k-1}$ and $\gamma_k$ **is computationally expensive**, thus we use **sub-sampling** to reduce computational effort.

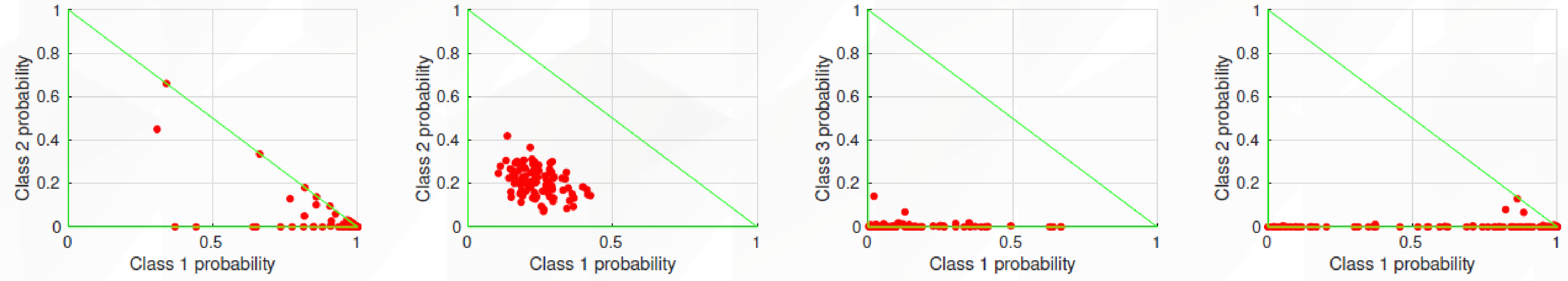



- Point cloud development for a single step: **uncertainty decreases**.




(a) $\{\lambda_{k-1}\}$ (b) $\{\gamma_k\}$ (c) $\{\mathcal{L}(\gamma_k)\}$ (d) $\{\lambda_k\}$

- Point cloud development for a single step: **uncertainty increases**.



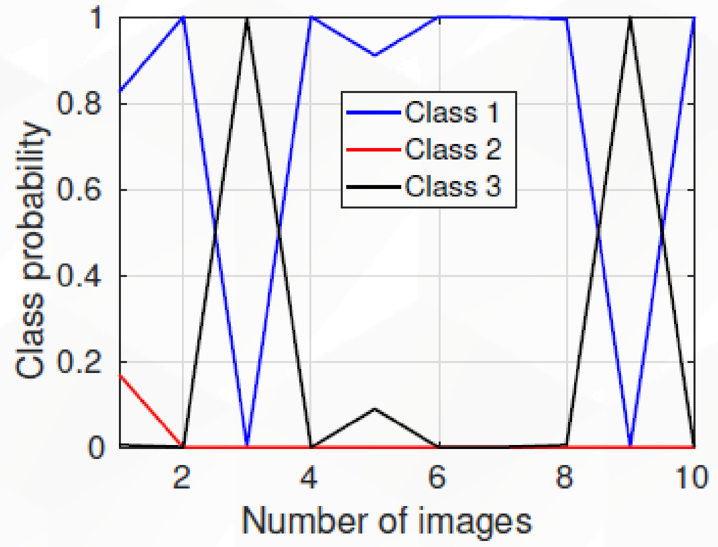

(a) $\{\lambda_{k-1}\}$ (b) $\{\gamma_k\}$ (c) $\{\mathcal{L}(\gamma_k)\}$ (d) $\{\lambda_k\}$

# *Experiment*

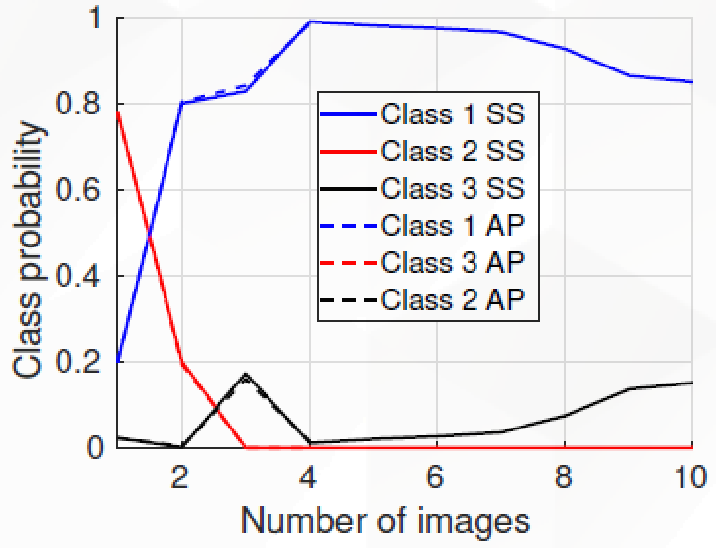
- Images of an object with occlusions, blur, different color filters. 3 possible classes, class 1 is the correct.

- **Compared 4 approaches**:
  - $\mathbb{P}(c|\gamma_{1:k})$, no classifier model.
  - $\mathbb{P}(c|\gamma_{1:k})$, with classifier model.
  - $\mathbb{P}(\lambda_k|z_{1:k})$, all pairs considered.
  - $\mathbb{P}(\lambda_k|z_{1:k})$, sub-sampling.
- Note that sub-sampling produces **very close** results to the approach that considers all pairs.
- Provides **superior classification** results, and provides access to **posterior model uncertainty.**

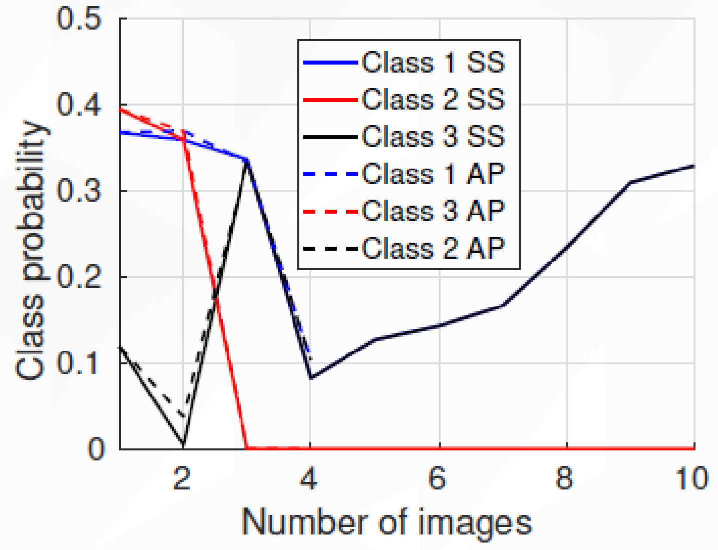

No dropout, no model




No dropout, with model




With dropout, expectation




With dropout, deviation

# *Conclusions*

- We proposed maintaining a **distribution over the posterior class probabilities** for classification and extracting uncertainty.
- We utilize a **cloud** of class probability vectors as a **classifier output**.
- To reduce computational effort, we proposed using a simple **sub-sampling method.**
- We showed **superior results** to common used approaches for classification.
- Our method provides access to **model uncertainty**.
- **Future work** may include utilizing this approach for multi-robot and active planning applications.

# *Thank you for listening!*