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 $P(c|y_{1:k})$.
• Infer uncertainty in classification from a single image.

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

• Small uncertainty:

• Large uncertainty:
**Definitions**

- **Class probability:**
  \[ \gamma_i^k = \mathbb{P}(c = i \mid z_k, D) \]
  \[ \gamma_k = \left[ \gamma_1^k, \ldots, \gamma_M^k \right]^T \]

- **Posterior class probability:**
  \[ \lambda_i^k = \mathbb{P}(c = i \mid \gamma_k) \]
  \[ \lambda_k = \left[ \lambda_1^k, \ldots, \lambda_M^k \right]^T \]

**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 \( \gamma_k \) is **stochastic** as well.

- **Multiple forward passes** through a network with dropout produces a **point cloud** \( \{ \gamma_k \} \) that approximates \( \mathbb{P}(\gamma_k \mid z_k, D) \).
- **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 \( \{\gamma_k\} \) that approximates \( P(\gamma_k | z_k, D) \).
• Uninformative prior for \( P(c) \).
• A Dirichlet distributed classifier model with known parameters.

Classifier Model

• Likelihood of \( \gamma_k \) given object class \( i \):
  \[
  L_i(\gamma_k) \triangleq P(\gamma_k | c = i, D) \quad L(\gamma_k) \triangleq [L_i(\gamma_k), ..., L_M(\gamma_k)]^T
  \]

• \( L(\gamma_k) \) is referred to as the classifier model.

• Dirichlet distributed in our case with a-priori known parameters \( \theta^i \):
  \[
  L_i(\gamma_k) = \text{Dir}(\gamma_k; \theta^i)
  \]

• Probability vector \( \gamma_k \) is projected via the classifier model to vector \( L(\gamma_k) \).

• Classifier model example:
Our Method

- Our goal is to maintain \( \mathbb{P}(\lambda_k | z_{1:k}, D) \).
- Using Bayes rule, \( \lambda_k \) is updated by:
  \[
  \lambda_k^i \propto \lambda_{k-1}^i \mathcal{L}(\gamma_k)
  \]
- 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.
  \[\{\lambda_{k-1}\}\]
  \[\{\gamma_k\}\]
  \[\mathcal{L}(\gamma_k)\]
  \[\{\lambda_k\}\]

- Point cloud development for a single step: uncertainty increases.
  \[\{\lambda_{k-1}\}\]
  \[\{\gamma_k\}\]
  \[\mathcal{L}(\gamma_k)\]
  \[\{\lambda_k\}\]
Experiment

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

- **Compared 4 approaches:**
  - $P(c|y_{1:k})$, no classifier model.
  - $P(c|y_{1:k})$, with classifier model.
  - $P(\lambda_k|z_{1:k})$, all pairs considered.
  - $P(\lambda_k|z_{1:k})$, 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.
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!