Geometry and Uncertainty in Deep Learning for Computer Vision

Alex Kendall, University of Cambridge, March 2017

@alexgkendall  alexgkendall.com  agk34@cam.ac.uk
Why is uncertainty important?
Bayesian SegNet for probabilistic scene understanding

Input Image  Semantic Segmentation  Uncertainty
Outline of Talk

1. What **uncertainty** can we model with deep learning and what are the benefits?

2. How do we model uncertainty using **Bayesian deep learning** for regression and classification tasks?

3. Why should we formulate deep learning models for vision which leverage our knowledge of **geometry**?
Uncertainty
What kind of uncertainty can we model?

1. **Epistemic uncertainty**
   - Measures what you’re model doesn’t know
   - Can be explained away by unlimited data

2. **Aleatoric uncertainty**
   - Measures what you can’t understand from the data
   - Can be explained away by unlimited sensing

What kind of uncertainty can we model?

*Epistemic* uncertainty is *modeling* uncertainty

*Aleatoric* uncertainty is *sensing* uncertainty
Deep learning is required to achieve state of the art results in computer vision applications but doesn’t provide uncertainty estimates.

- **Bayesian neural networks** are a framework for understanding uncertainty in deep learning
- They have **distributions over network parameters** (rather than deterministic weights)
- Traditionally they have been **tricky to scale**

---

We can **model epistemic uncertainty** in deep learning models using Monte Carlo **dropout sampling** at test time.

Dropout sampling can be interpreted as **sampling from a distribution over models**.
# Modeling Aleatoric Uncertainty with Probabilistic Deep Learning

<table>
<thead>
<tr>
<th></th>
<th>Deep Learning</th>
<th>Probabilistic Deep Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>( \hat{y} = f(x) )</td>
<td>( \hat{y}, \sigma^2 = f(x) )</td>
</tr>
<tr>
<td><strong>Regression</strong></td>
<td>( \text{Loss} = |y - \hat{y}|^2 )</td>
<td>( \text{Loss} = \frac{|y - \hat{y}|^2}{2\hat{\sigma}^2} + \log \hat{\sigma}^2 )</td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td>( \text{Loss} = \text{SoftmaxCrossEntropy}(\hat{y}_t) )</td>
<td>( \hat{y}_t = \hat{y} + \epsilon_t \quad \epsilon_t \sim N(0, \hat{\sigma}^2) ) ( \text{Loss} = \frac{1}{T} \sum_t \text{SoftmaxCrossEntropy}(\hat{y}_t) )</td>
</tr>
</tbody>
</table>

---

# Semantic Segmentation Performance on CamVid

<table>
<thead>
<tr>
<th>CamVid Results</th>
<th>IoU Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet (State of the art baseline)</td>
<td>67.1</td>
</tr>
<tr>
<td>+ Aleatoric Uncertainty</td>
<td>67.4</td>
</tr>
<tr>
<td>+ Epistemic Uncertainty</td>
<td>67.2</td>
</tr>
<tr>
<td>+ Aleatoric &amp; Epistemic</td>
<td>67.5</td>
</tr>
</tbody>
</table>

![CamVid Results](image)

---

### NYU Depth Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Rel. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet (State of the art baseline)</td>
<td>0.167</td>
</tr>
<tr>
<td>+ Aleatoric Uncertainty</td>
<td>0.149</td>
</tr>
<tr>
<td>+ Epistemic Uncertainty</td>
<td>0.162</td>
</tr>
<tr>
<td>+ Aleatoric &amp; Epistemic</td>
<td>0.145</td>
</tr>
</tbody>
</table>

Input Video (Monocular)

Predicted Depth

Uncertainty
Aleatoric vs. Epistemic Uncertainty for Out of Dataset Examples

<table>
<thead>
<tr>
<th>Train dataset</th>
<th>Test dataset</th>
<th>RMS</th>
<th>Aleatoric variance</th>
<th>Epistemic variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make3D / 4</td>
<td>Make3D</td>
<td>5.76</td>
<td>0.506</td>
<td>7.73</td>
</tr>
<tr>
<td>Make3D / 2</td>
<td>Make3D</td>
<td>4.62</td>
<td>0.521</td>
<td>4.38</td>
</tr>
<tr>
<td>Make3D</td>
<td>Make3D</td>
<td>3.87</td>
<td>0.485</td>
<td>2.78</td>
</tr>
<tr>
<td>Make3D / 4</td>
<td>NYUv2</td>
<td>-</td>
<td>0.388</td>
<td>15.0</td>
</tr>
<tr>
<td>Make3D</td>
<td>NYUv2</td>
<td>-</td>
<td>0.461</td>
<td>4.87</td>
</tr>
</tbody>
</table>

Aleatoric uncertainty remains constant while epistemic uncertainty increases for out of dataset examples!

One reason why computer vision has progressed so rapidly is because we can benchmark and compare algorithms easily.

Often leaderboards rank prediction accuracy and algorithm speed.

Leaderboards should also rank algorithms probabilistically and quantify uncertainty accuracy.
Calibration Plots

- For a prediction with probability $p$, the model should be correct with a frequency of $p$
- Perfect calibration corresponds to the line, $y = x$

Precision Recall Plots

- Uncertainty should correlate well with accuracy

(a) Classification (CamVid)

(b) Regression (Make3D)

Putting it all Together: Multi-Task Learning
Multitask Learning

We want to simultaneously learn multiple tasks: \[ \text{Loss} = \sum_i w_i L_i \]

Task performance is very sensitive to choice of weights, so how do you choose??

Scene Understanding [1]

Localisation [2]

Types of Aleatoric Uncertainty

1. **Heteroscedastic aleatoric uncertainty**
   - Data dependent aleatoric uncertainty

2. **Homoscedastic aleatoric uncertainty**
   - Aleatoric uncertainty which doesn’t depend on the data
   - Task uncertainty

Combine Losses Using Homoscedastic Uncertainty

Homoscedastic uncertainty, $\sigma^2$, captures uncertainty of the entire task itself – not dependant on input data.

We propose to use this to learn a weighting for each loss term.

$$
\text{Loss} = \frac{L_{\text{regression 1}}}{\sigma_1^2} + \log \sigma_1^2 + \frac{L_{\text{regression 2}}}{\sigma_2^2} + \log \sigma_2^2 + \text{SoftmaxCrossEntropy} \left( \frac{y}{\sigma_3^2} \right)
$$

Multi Task Scene Understanding Model

Multitask Learning Results

- Homoscedastic uncertainty can learn the optimal weighting
- Multitask learning can improve performance compared with training separate models for each individual task

<table>
<thead>
<tr>
<th>Loss</th>
<th>Task Weights</th>
<th>Classification IoU [%]</th>
<th>Instance RMS Error [px]</th>
<th>Inverse Depth RMS Error [px]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cls.</td>
<td>Inst.</td>
<td>Depth</td>
<td></td>
</tr>
<tr>
<td>Class only</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>43.1%</td>
</tr>
<tr>
<td>Instance only</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Depth only</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Unweighted sum of losses</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
<td>43.6%</td>
</tr>
<tr>
<td>Approx. optimal weights</td>
<td>0.8</td>
<td>0.05</td>
<td>0.15</td>
<td>46.3%</td>
</tr>
<tr>
<td>2 task uncertainty weighting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>46.5%</td>
</tr>
<tr>
<td>2 task uncertainty weighting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>46.2%</td>
</tr>
<tr>
<td>3 task uncertainty weighting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>46.6%</td>
</tr>
</tbody>
</table>

Qualitative Multitask Learning Results

Geometry
Geometry in Computer Vision?

- Geometry was once the most exciting topic in computer vision
- Now machine learning models are the solution to most tasks
- These black boxes can learn many representations with end-to-end supervised learning
- Often naïve architectures are used
However, geometry provides a rich source of training data.

Motion, pose and depth can be leveraged for supervised and unsupervised training.

Geometric priors and architectural designs can significantly improve model performance.
Naïve deep learning approach to learning camera pose

PoseNet: trained end-to-end to regress camera position, \( x \) and orientation, \( q \)

\[
\text{loss}(I) = ||\hat{x} - x||_2 + \beta \left\| \hat{q} - \frac{q}{||q||} \right\|_2
\]
Camera Pose Regression

training data in green, test data in blue, PoseNet results in red

Train with reprojection loss of 3-D geometry with predicted and ground truth camera poses.

\[
\text{loss}(I) = \frac{1}{|G'|} \sum_{g_i \in G'} \| \pi(q, x, g_i) - \pi(\hat{q}, \hat{x}, g_i) \|_\gamma
\]

Where \( \pi \) is the projection function of 3-D point \( g_i \)

Camera Pose Regression

Using geometry in our model structure improves performance

<table>
<thead>
<tr>
<th>Scene</th>
<th>Spatial Extent</th>
<th>PoseNet (GoogLeNet, L2)</th>
<th>Bayesian PoseNet (GoogLeNet, L2)</th>
<th>PoseNet v2 (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td>King’s College</td>
<td>140 × 40m</td>
<td>1.66m, 4.86°</td>
<td>1.74m, 4.06°</td>
<td>0.92m, 0.83°</td>
</tr>
<tr>
<td>Street</td>
<td>500 × 100m</td>
<td>2.96m, 6.00°</td>
<td>2.14m, 4.96°</td>
<td>1.32m, 1.57°</td>
</tr>
<tr>
<td>Old Hospital</td>
<td>50 × 40m</td>
<td>2.62m, 4.90°</td>
<td>2.57m, 5.14°</td>
<td>1.12m, 1.83°</td>
</tr>
<tr>
<td>Shop Façade</td>
<td>35 × 25m</td>
<td>1.41m, 7.18°</td>
<td>1.25m, 7.54°</td>
<td>0.72m, 0.93°</td>
</tr>
<tr>
<td>St Mary’s Church</td>
<td>80 × 60m</td>
<td>2.45m, 7.96°</td>
<td>2.11m, 8.38°</td>
<td>1.62m, 1.84°</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>2.22m, 6.18°</strong></td>
<td><strong>1.96m, 6.02°</strong></td>
<td><strong>1.14m, 1.40°</strong></td>
</tr>
</tbody>
</table>

- Chess              | 3×2×1m        | 0.32m, 6.60°            | 0.37m, 7.24°                     | 0.12m, 3.24°           |
- Fire               | 2.5×1×1m      | 0.47m, 14.0°            | 0.43m, 13.7°                     | 0.13m, 4.20°           |
- Heads              | 2×0.5×1m      | 0.30m, 12.2°            | 0.31m, 12.0°                     | 0.08m, 5.72°           |
- Office             | 2.5×2×1.5m    | 0.48m, 7.24°            | 0.48m, 8.04°                     | 0.16m, 2.38°           |
- Pumpkin            | 2.5×2×1m      | 0.49m, 8.12°            | 0.61m, 7.08°                     | 0.14m, 2.15°           |
- Red Kitchen        | 4×3×1.5m      | 0.58m, 8.34°            | 0.58m, 7.54°                     | 0.16m, 4.24°           |
- Stairs             | 2.5×2×1.5m    | 0.48m, 13.1°            | 0.48m, 13.1°                     | 0.18m, 4.86°           |
| **Average**         |               | **0.45m, 9.94°**        | **0.47m, 9.81°**                 | **0.14m, 3.83°**       |

Epistemic uncertainty to estimate loop closure

We can use epistemic uncertainty to estimate metric relocalisation error.

Determine if the model has seen the landmark before (loop closure).

Increased uncertainty from strong occlusion, motion blur, visually ambiguous landmarks.

End to end deep learning for stereo vision

- Form differentiable cost volume and sub-pixel regression network with soft argmax function
- Use 3-D convolutions to learn to regularise the volume

**Soft ArgMin / ArgMax**

\[
\text{soft argmin} := \sum_{d=0}^{D_{\text{max}}} d \times \sigma(-c_d)
\]

<table>
<thead>
<tr>
<th>Loss</th>
<th>&gt; 3px Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Loss</td>
<td>12.2</td>
</tr>
<tr>
<td>Soft Classification</td>
<td>12.3</td>
</tr>
<tr>
<td>Regression</td>
<td>9.34</td>
</tr>
</tbody>
</table>

Scene Flow Dataset Results

(c) Scene Flow test set qualitative results. From left: left stereo input image, disparity prediction, ground truth.
Probabilistic Deep Learning for Stereo Vision


1st Place on the 2012 & 2015 KITTI Stereo Challenge

Autonomous Drone Prototype

Skydio Inc.  http://www.skydio.com/
Conclusions

1. **Aleatoric** uncertainty is important for
   - Large data situations, where epistemic uncertainty is explained away,
   - Real-time applications, because we can form aleatoric models without expensive Monte Carlo samples,
   - Multitask applications, because we can appropriately weight each loss.

2. **Epistemic** uncertainty is important for
   - Safety-critical applications, because epistemic uncertainty is required to understand examples which are different from training data,
   - Small datasets, where the training data is sparse,
   - Exploratory applications, such as loop closure and reinforcement learning.
Conclusions

3. It is important to quantify the accuracy of uncertainty estimates.

4. We should leverage our knowledge of geometry when designing machine learning models for computer vision:
   - Reprojection loss
   - Stereo cost volume
CVPR Tutorial

Hawaii

July 26th 2017

See you there?
Thank You & References