PYCON KOREA 2017

How Far Can You Trust Your Deep Neural Networks

Feat. TensorFlow

Sungjoon Choi

Back to the Basic

I made two mistakes

PyCon KR 2017 Call for Proposals

Email address *

sungjoon.choi@cpslab.snu.ac.kr

이름 (Name) *

Sungjoon Choi

세션 제목 (Title of your talk) *

How far can you trust your Deep Neural Networks? (feat. TensorFlow)

세션 카테고리 (Which category best describes your talk?) *

- ○ Best Practices & Patterns
- ○ Python Core (language, stdlib, etc.)
- ◉ Python Libraries
- ○ Science / Data
- ○ Community
- ○ Service and Web Development
- ○ Environment and DevOps
- ○ Other:

세션 난이도 (Level of difficulty) *

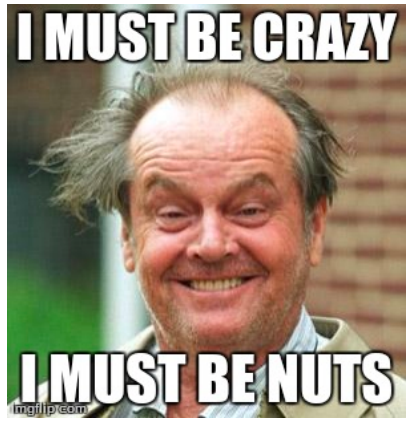
- ○ 초급 (Beginner)
- ○ 중급 (Intermediate)
- ◉ 고급 (Experienced)

세션 길이 (How long would you talk about the topic?) *

- ◉ 25분 (25mins)
- ○ 40분 (40mins)

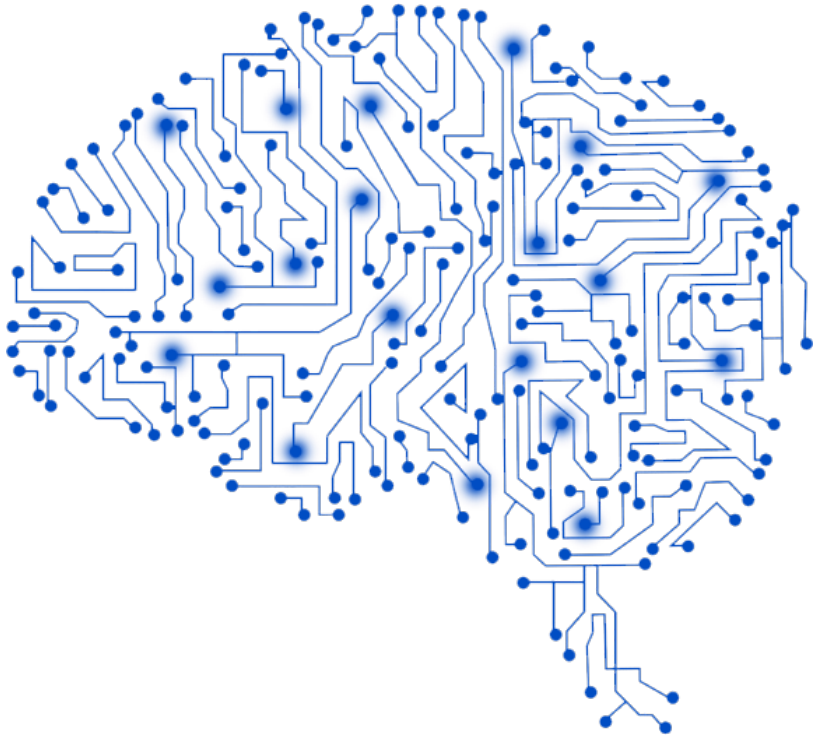
언어 (Language) *

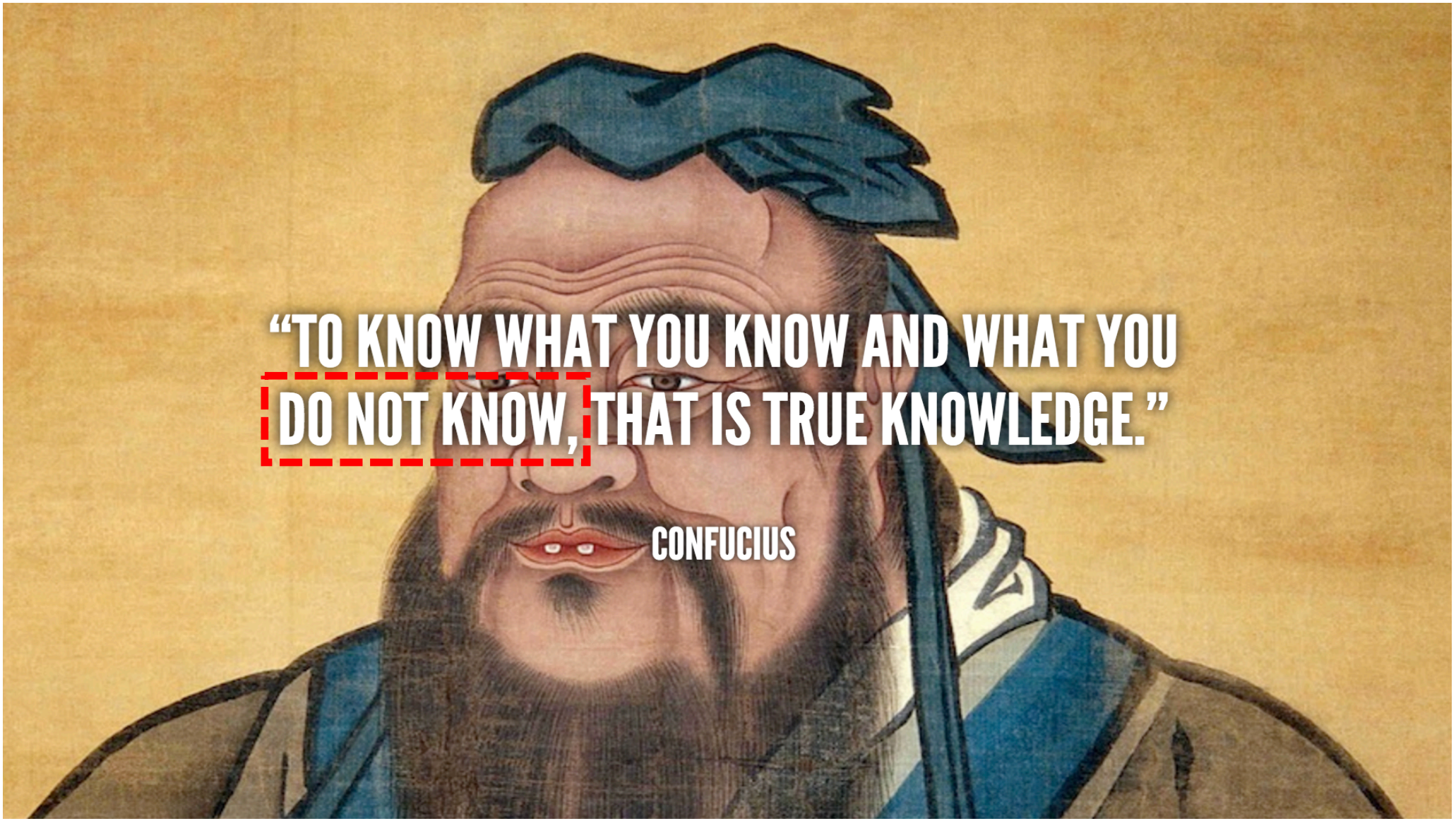
- ◉ 영어 (English)
- ○ 한국어 (Korean)



To be honest,

I will talk about two things

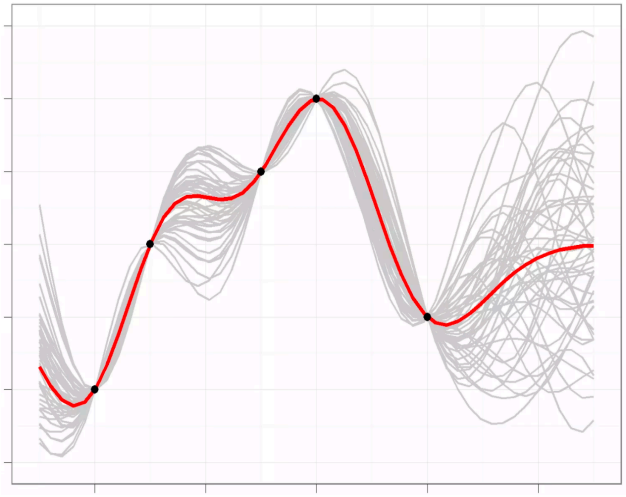



“TO KNOW WHAT YOU KNOW AND WHAT YOU
DO NOT KNOW, THAT IS TRUE KNOWLEDGE.”
CONFUCIUS

Why?

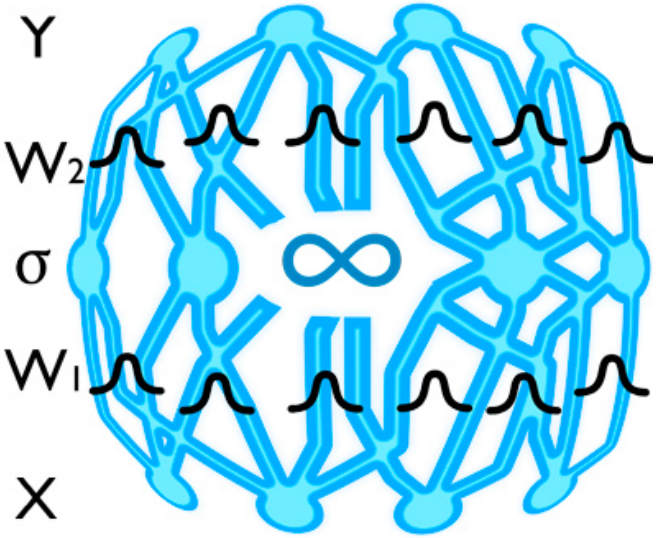
http://www.businessinsider.com/tesla-fatal-crash-speeding-autopilot-ntsb-report-2016-7

Then, How?



Gaussian Process

+



Bayesian Network

+

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) p(\omega|\mathbf{X}, \mathbf{Y}) \, \mathrm{d}\omega.$$

$$\operatorname{argmin}_\theta \mathrm{KL}(q_\theta(\omega) \mid p(\omega|\mathbf{X}, \mathbf{Y})).$$

$$q_\theta(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) q_\theta(\omega) \mathrm{d}\omega$$

$$q_\theta(\mathbf{y}^*|\mathbf{x}^*) \approx \frac{1}{T} \sum^{T} p(\mathbf{y}^*|\mathbf{x}^*, \omega_t)$$

$$\mathcal{L}_{\mathrm{VI}} := \int q_\theta(\omega) \log p(\mathbf{Y}|\mathbf{X}, \omega) \mathrm{d}\omega - \mathrm{KL}(q_\theta(\omega)||p(\omega))$$

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int \mathcal{N}(\mathbf{w}; 0, l^{-2}\mathbf{I}_Q) p(b) \sigma(\mathbf{w}^T\mathbf{x} + b) \sigma(\mathbf{w}^T\mathbf{y} + b) \mathrm{d}\mathbf{w} \mathrm{d}b$$

$$\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \sigma(\mathbf{w}_k^T\mathbf{x} + b_k) \sigma(\mathbf{w}_k^T\mathbf{y} + b_k)$$

$$\mathbf{w}_k \sim \mathcal{N}(0, l^{-2}\mathbf{I}_Q), \quad \mathbf{w}_d \sim \mathcal{N}(0, l^{-2}\mathbf{I}_K), \quad b_k \sim p(b),$$
$$\mathbf{W}_1 = [\mathbf{w}_k]_{k=1}^{K}, \quad \mathbf{W}_2 = [\mathbf{w}_d]_{d=1}^{D}, \quad \mathbf{b} = [b_k]_{k=1}^{K},$$
$$\omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$$

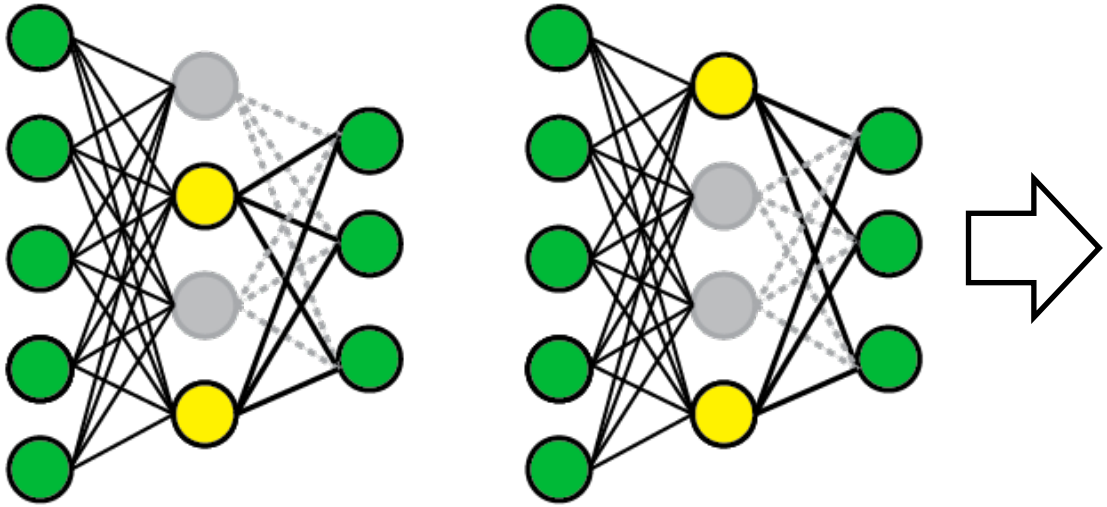
$$p(\mathbf{y}^*|\mathbf{x}^*, \omega) = \mathcal{N}\left(\mathbf{y}^*; \sqrt{\frac{1}{K}} \sigma(\mathbf{x}^*\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2, \tau^{-1}\mathbf{I}_N\right)$$

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) p(\omega|\mathbf{X}, \mathbf{Y}) \mathrm{d}\omega.$$

Equations

http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html

Results are pretty simple!

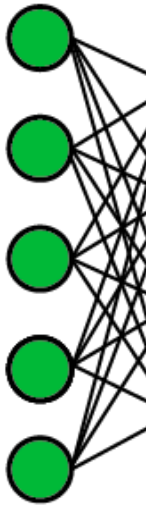


$$\mathbb{E}(\mathbf{y}^*) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t^*(\mathbf{x}^*)$$

$$\mathrm{Var}\left(\mathbf{y}^*\right) \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t^*(\mathbf{x}^*)^T\hat{\mathbf{y}}_t^*(\mathbf{x}^*) - \mathbb{E}(\mathbf{y}^*)^T\mathbb{E}(\mathbf{y}^*)$$

http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html

Results are pretty simple!

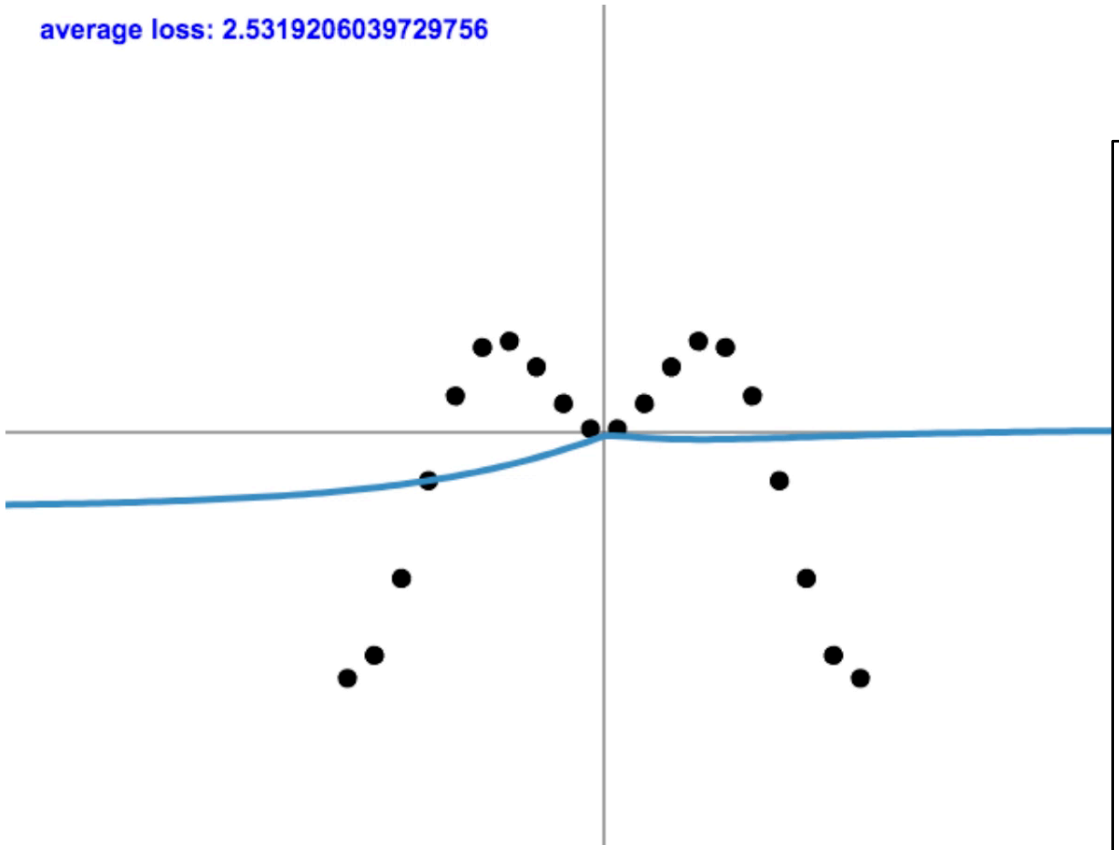


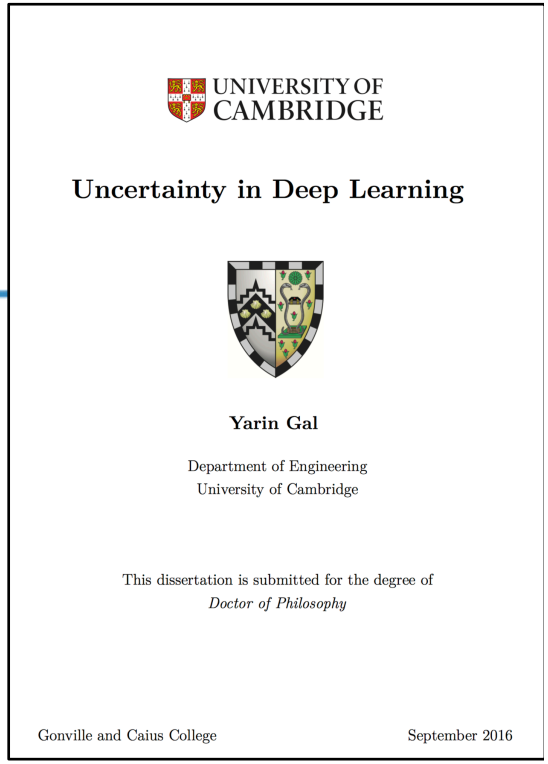
```
probs = []
for _ in xrange(T):
    probs += [model.output_probs(input_x)]
predictive_mean = numpy.mean(prob, axis=0)
predictive_variance = numpy.var(prob, axis=0)
tau = l**2 * (1 - model.p) / (2 * N * model.weight_decay)
predictive_variance += tau**-1
```

$(\mathbf{x}^*)$

http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html

Cool, isn't it?





UNIVERSITY OF CAMBRIDGE

Uncertainty in Deep Learning

Yarin Gal

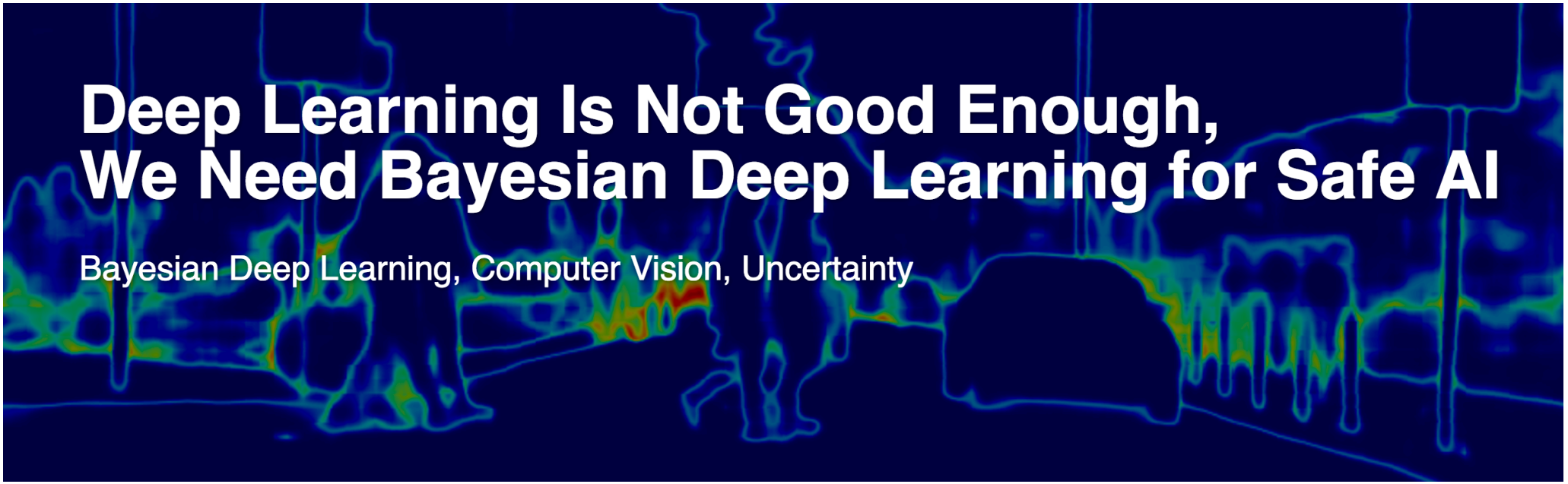
Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Gonville and Caius College — September 2016

Yarin Gal, "Uncertainty in Deep Learning", PhD Thesis, 2016

Recently,



http://alexgkendall.com/computer_vision/bayesian_deep_learning_for_safe_ai/

Two Types of Uncertainty

Aleatoric uncertainty

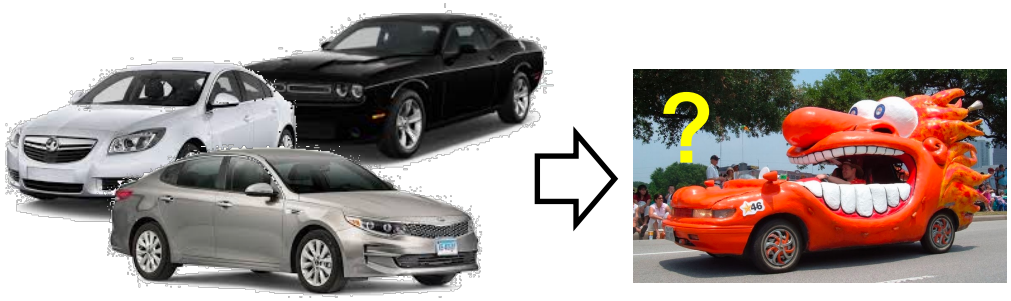
Captures our uncertainty with respect to information which our data cannot explain.

Aleatoric uncertainty captures noise sources such as measurement noise—noises which cannot be explained away even if more data were available.

Epistemic uncertainty

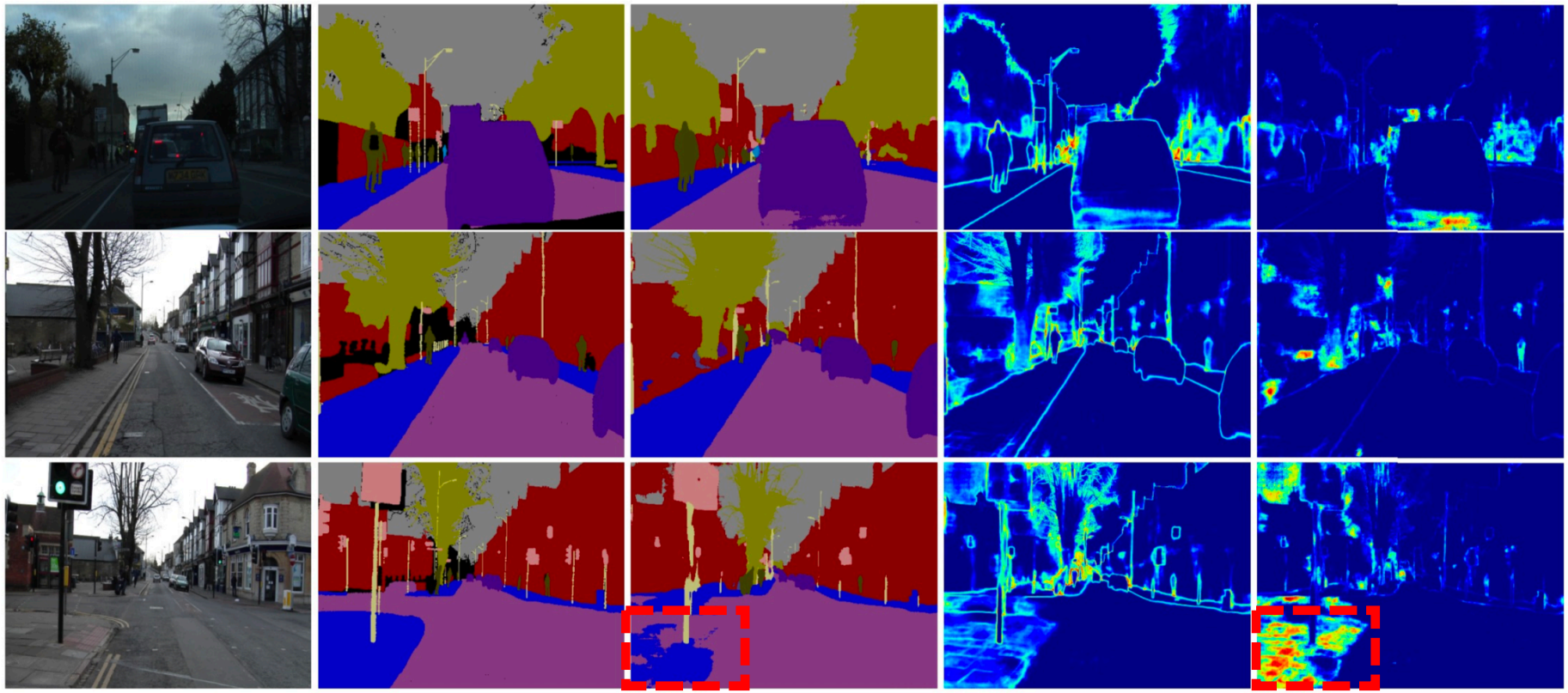
Captures our ignorance about which model generated our collected data.

Epistemic uncertainty reduces as the amount of observed data increases—hence its alternative name "reducible uncertainty".



http://alexgkendall.com/computer_vision/bayesian_deep_learning_for_safe_ai/

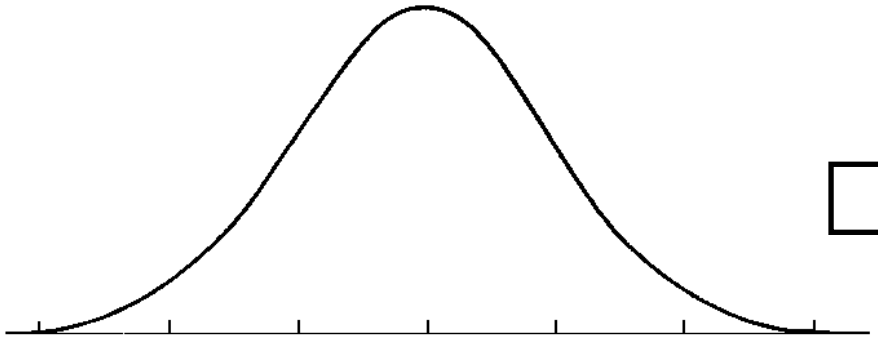
Semantic Segmentation Results



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

http://alexgkendall.com/computer_vision/bayesian_deep_learning_for_safe_ai/

How They Did It?



$$[\hat{x}, \hat{\sigma}^2] = f^{\widehat{\mathbf{W}}}(I)$$

Density Network

$$\mathrm{Var}(x) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{x}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T} \hat{x}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T} \hat{\sigma}_t^2$$

?

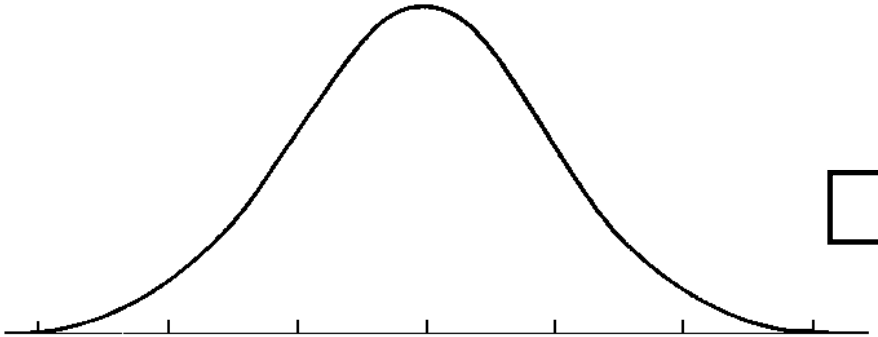
Law of Total Variance

Explained Variance Unexplained Variance

$$V[X] = V(E[X|y]) + E(V[X|y])$$

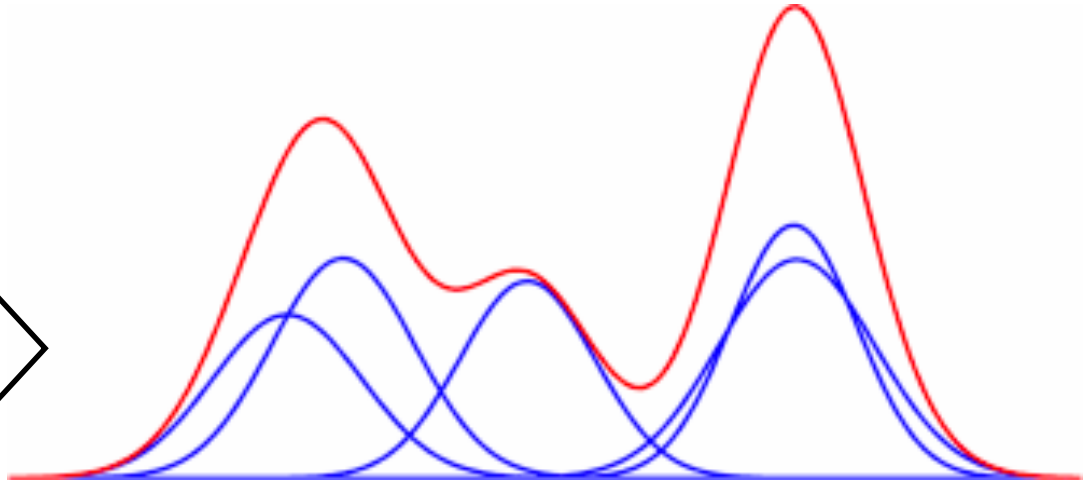
$$\text{Var}(x) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{x}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{x}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2$$

What We Did?



$$[\hat{x}, \hat{\sigma}^2] = f^{\widehat{\mathbf{W}}}(I)$$

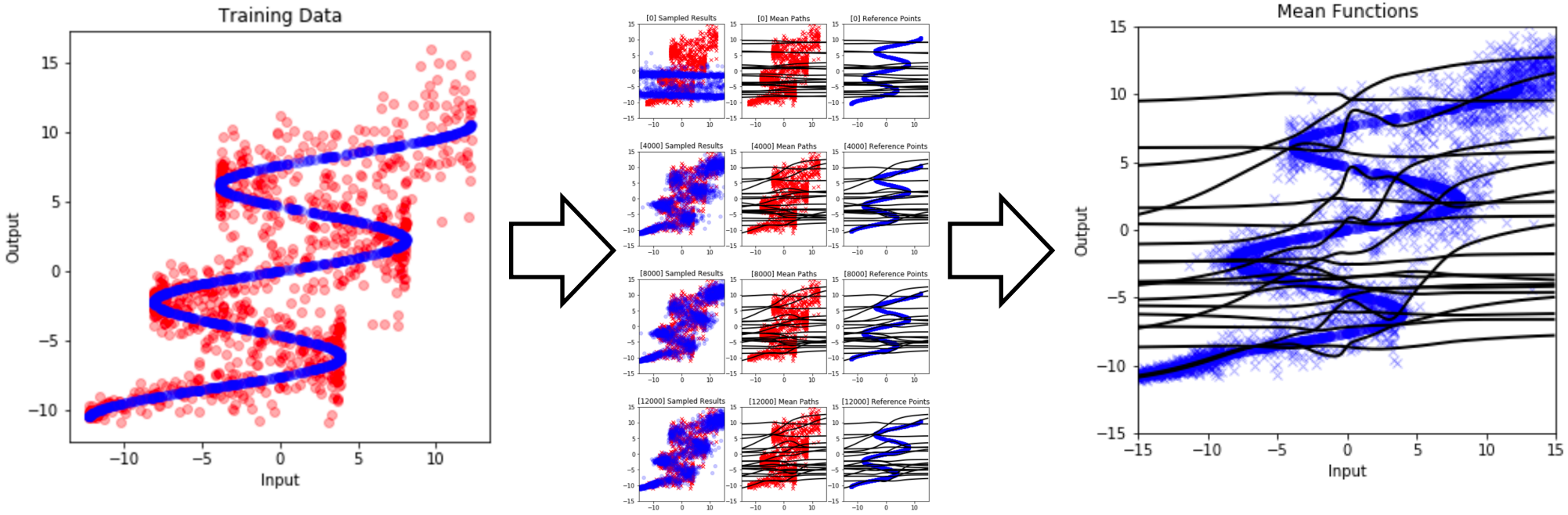
Density Network



$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \Sigma_k)$$

Mixture Density Network

Mixture Density Network



https://github.com/sjchoi86/advanced-tensorflow/blob/master/mdn/hmdn.ipynb

Mixture Density Network

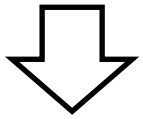
```
# HETEROSCADESTIC GAUSSIAN MIXTURE MODEL NEGATIVE LOG LIKELIHOOD
def hgmm_nll(self, _y, _hmdn_out):
    _pi    = _hmdn_out['pi']
    _mu    = _hmdn_out['mu']
    _sigma = _hmdn_out['sigma'] + self.sigoffset
    _err   = _hmdn_out['err']
    _probs = self.tf_normal(tf.tile(_y, [1, self.kmix]), _mu
                            , sigma+tf.tile(_err, [1, self.kmix]))
    # KRON TO ADL Y-DIM
    _temp  = tf.multiply(self.kron(_pi, self.dimy), _probs)
    _res   = tf.reduce_sum(_temp, 1, keep_dims=True)
    _eps   = 1e-9
    _temp  = _res +_eps
    return tf.reduce_mean(-tf.log( _temp )) # NLL
```

https://github.com/sjchoi86/advanced-tensorflow/blob/master/mdn/hmdn.ipynb

Revisit Law of Total Variance

Explained Variance Unexplained Variance

$$V[X] = V(E[X|y]) + E(V[X|y])$$



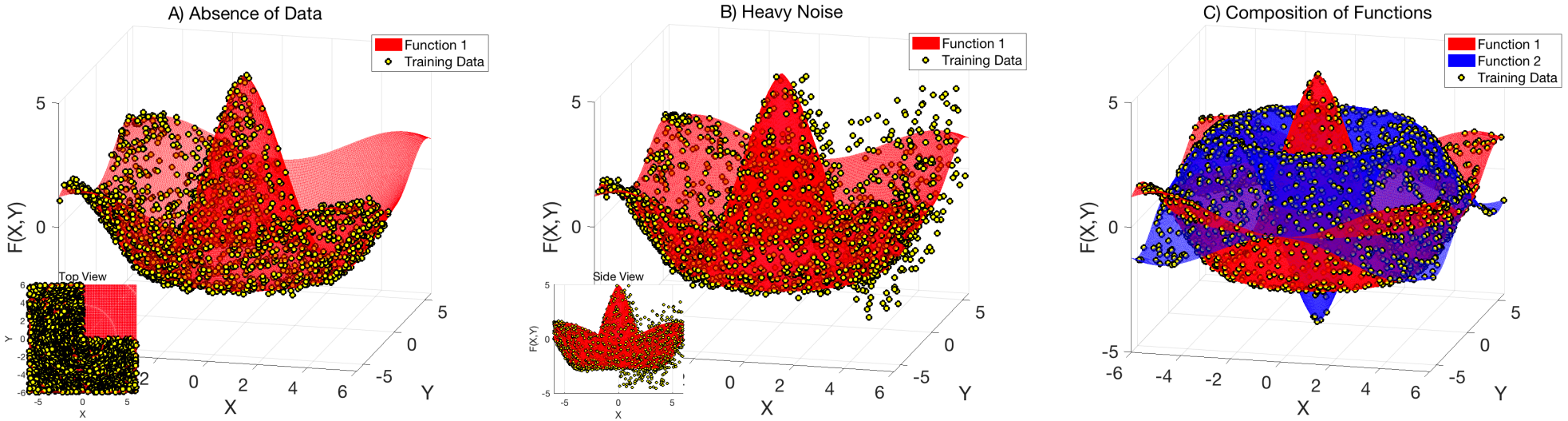
$$V[X] = \sum_y p(y)(E[X|y] - E[X])^2 + \sum_y p(y)\, V(X|y)$$

Depends on mean functions

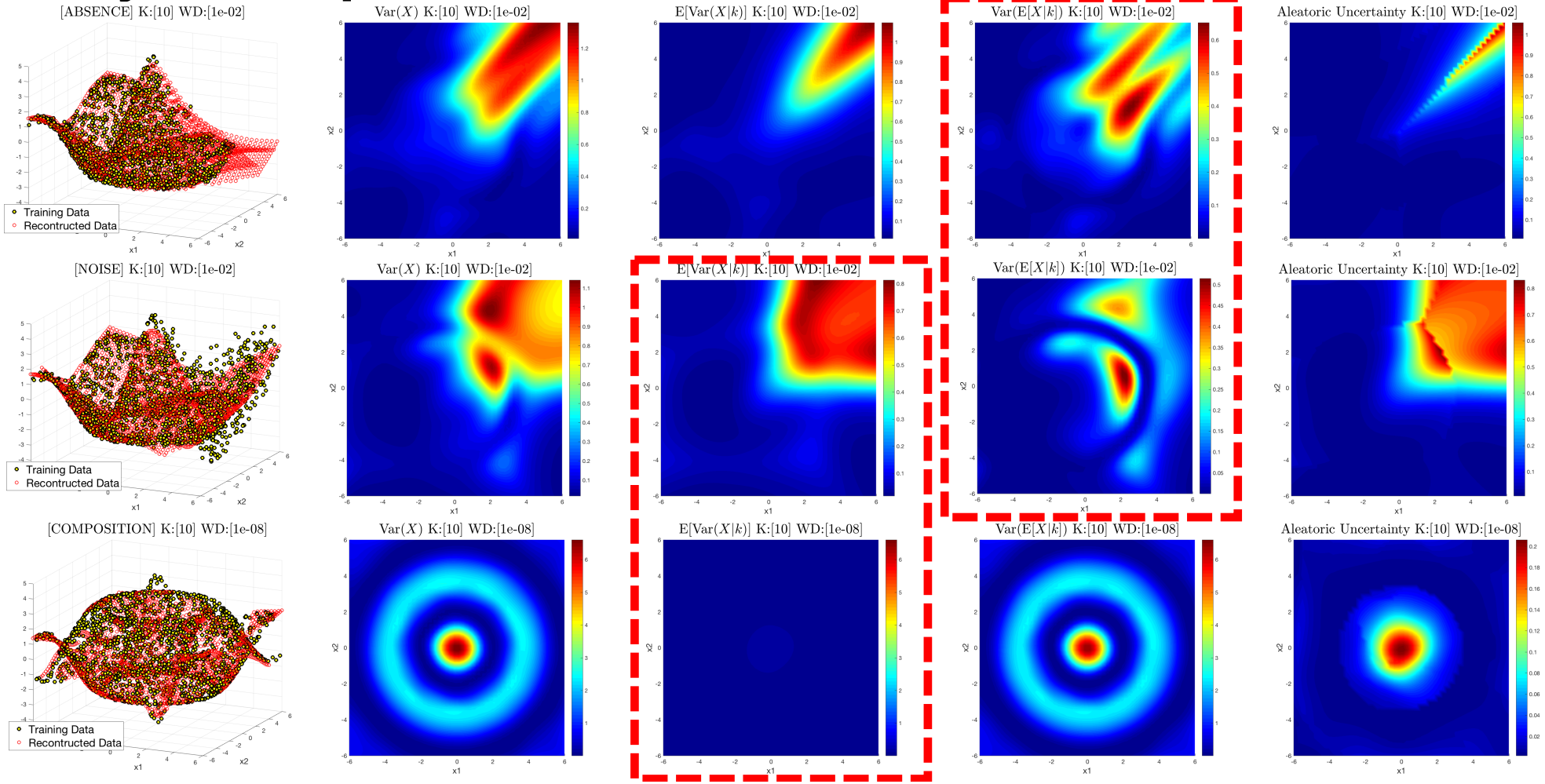
Depends on variance functions

Toy Example

Three Scenarios

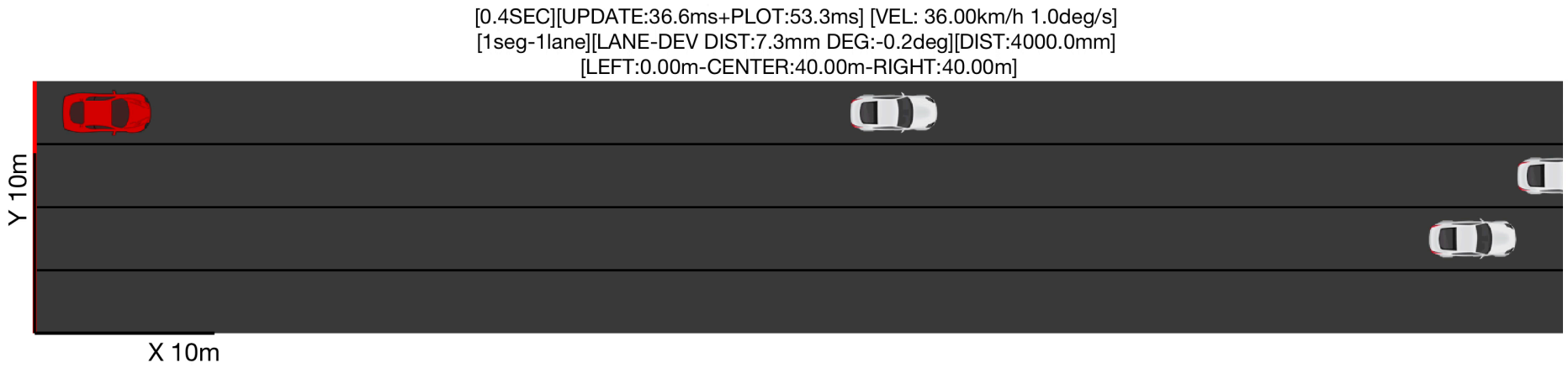


Toy Example


[ABSENCE] K:[10] WD:[1e-02]
Var(X) K:[10] WD:[1e-02]
E[Var(X|k)] K:[10] WD:[1e-02]
Var(E[X|k]) K:[10] WD:[1e-02]
Aleatoric Uncertainty K:[10] WD:[1e-02]
[NOISE] K:[10] WD:[1e-02]
Var(X) K:[10] WD:[1e-02]
E[Var(X|k)] K:[10] WD:[1e-02]
Var(E[X|k]) K:[10] WD:[1e-02]
Aleatoric Uncertainty K:[10] WD:[1e-02]
[COMPOSITION] K:[10] WD:[1e-08]
Var(X) K:[10] WD:[1e-08]
E[Var(X|k)] K:[10] WD:[1e-08]
Var(E[X|k]) K:[10] WD:[1e-08]
Aleatoric Uncertainty K:[10] WD:[1e-08]
Training Data
Recontructed Data
x1
x2

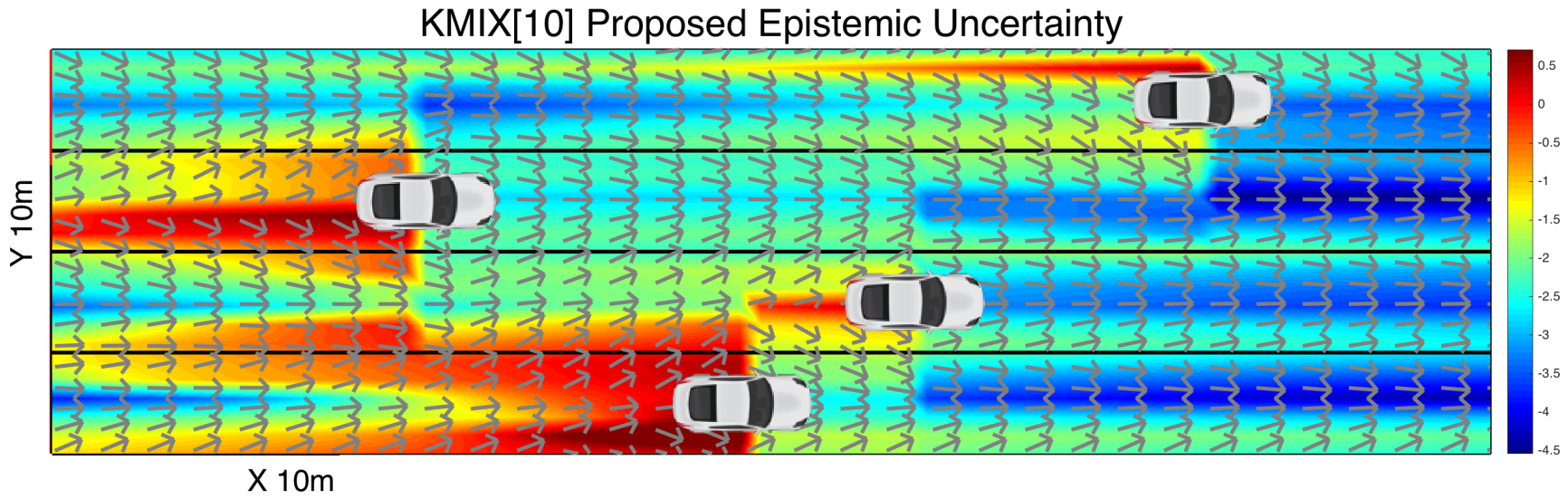
Track Driving Scenario

Demonstration Collection



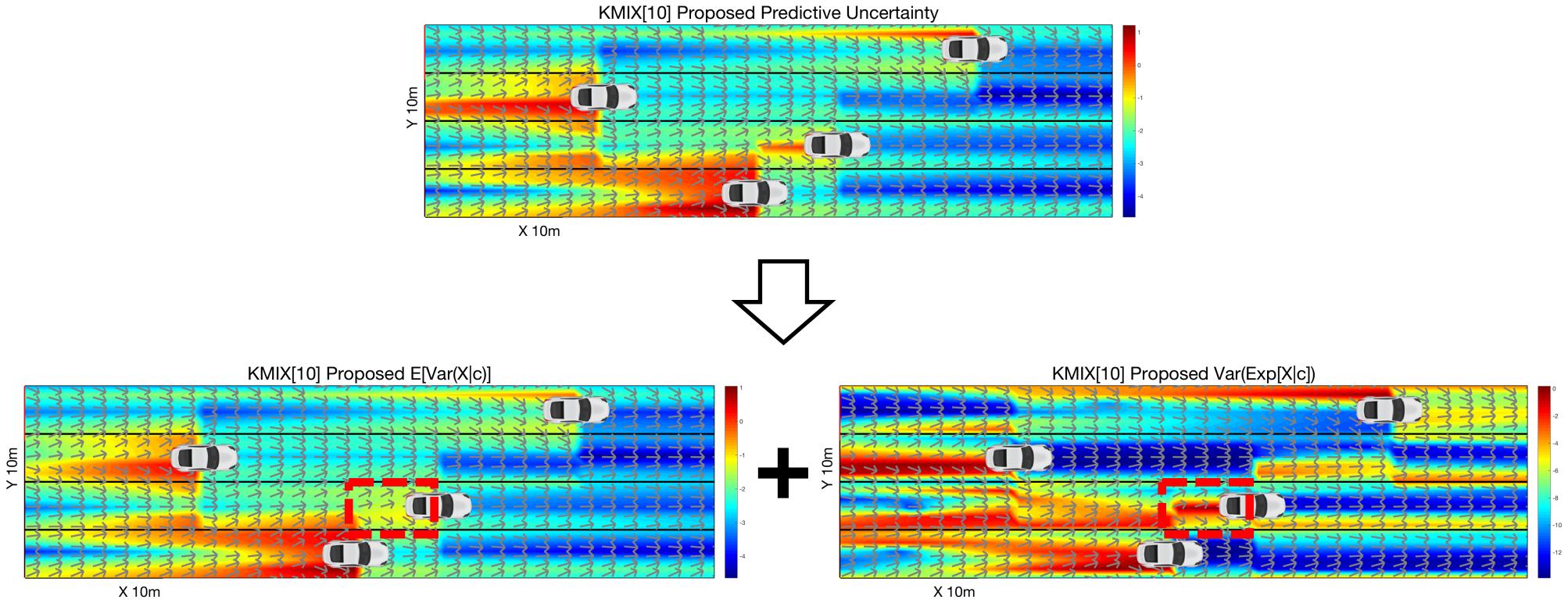
Track Driving Scenario

Predictive Uncertainty



Track Driving Scenario

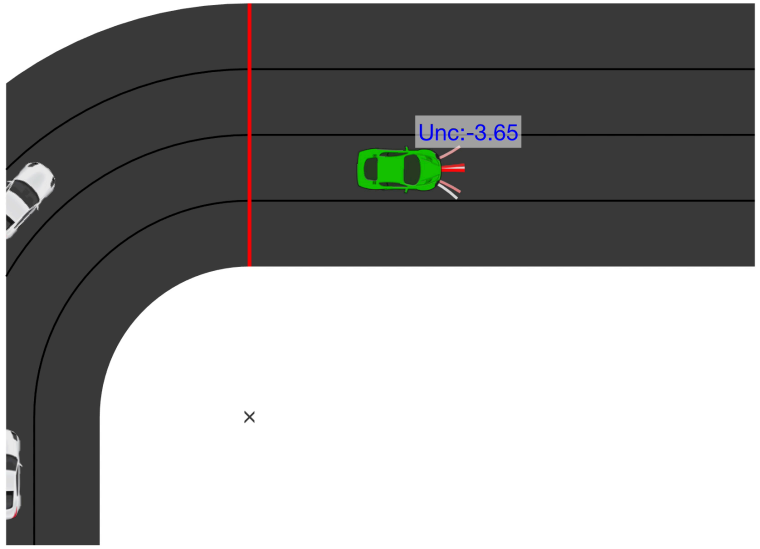
Uncertainty Decompositions



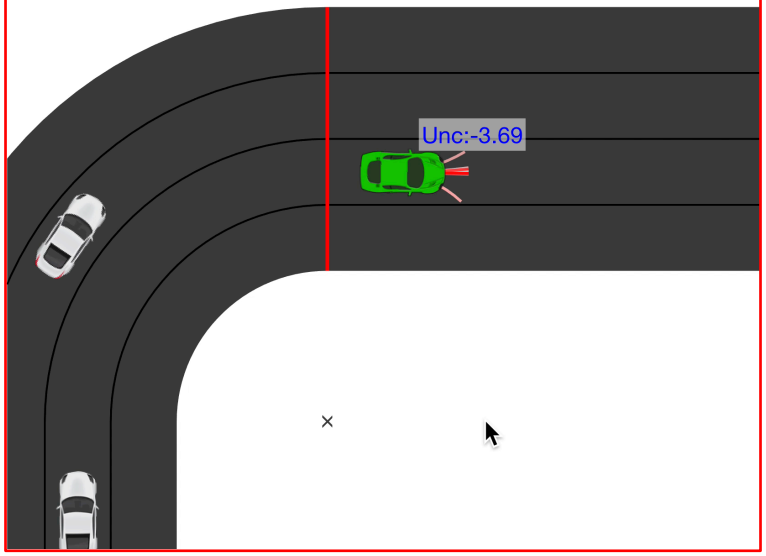
Track Driving Scenario

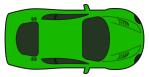
Track Simulator

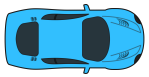
Naïve Navigation

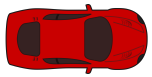


Uncertainty-aware Navigation



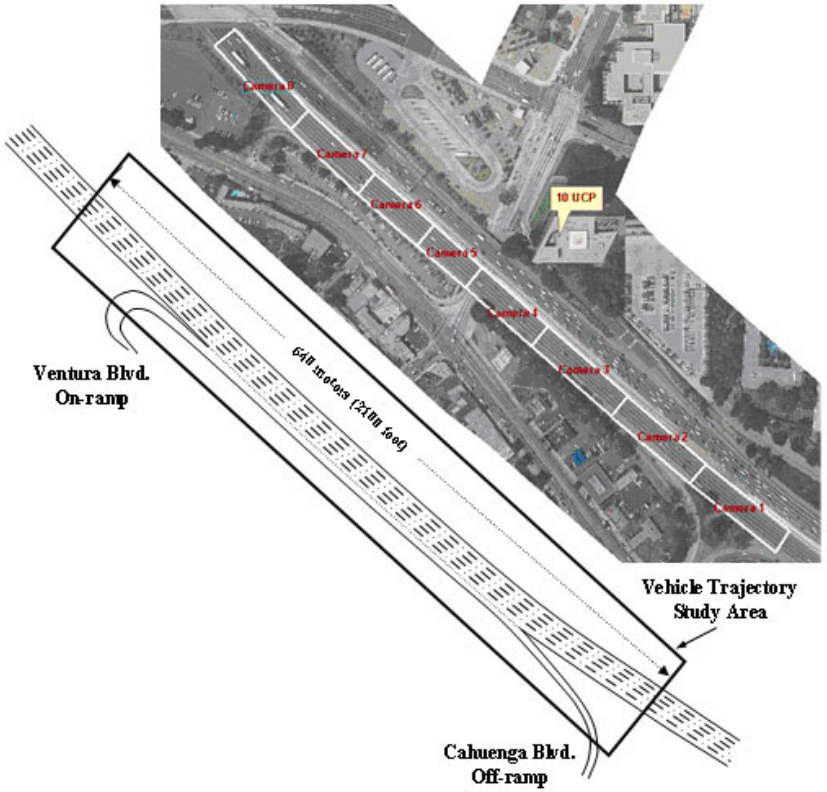

Uncertainty below -1
Directional vel: 20m/s

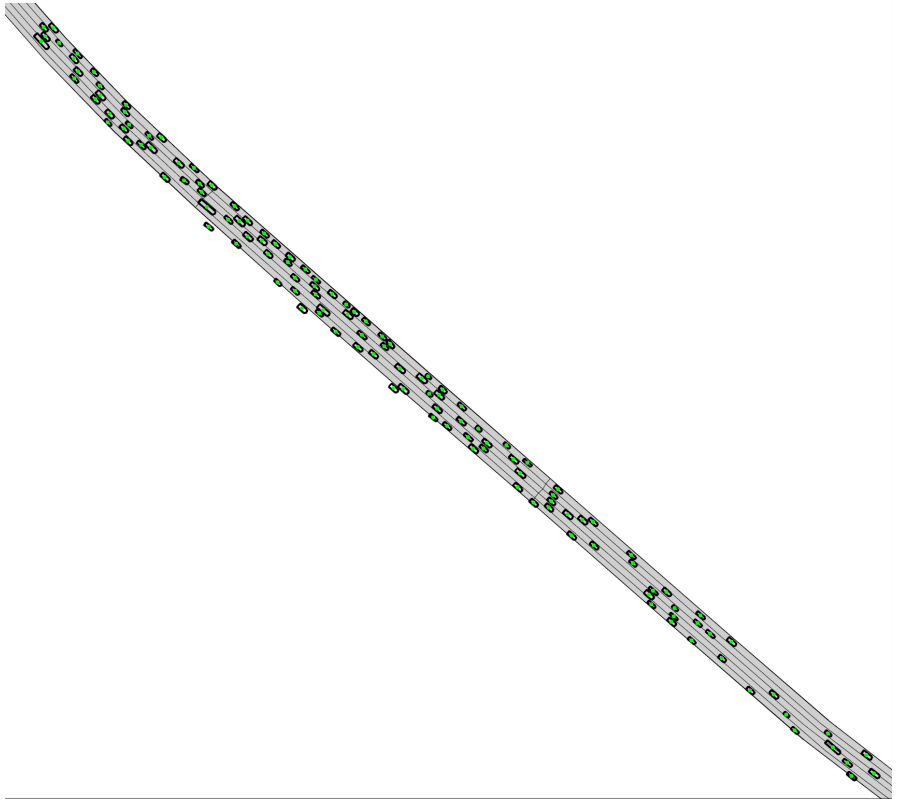

Uncertainty $-1 \sim 0$
Directional vel: 15m/s


Uncertainty above 0
Directional vel: 10m/s

Track Driving Scenario

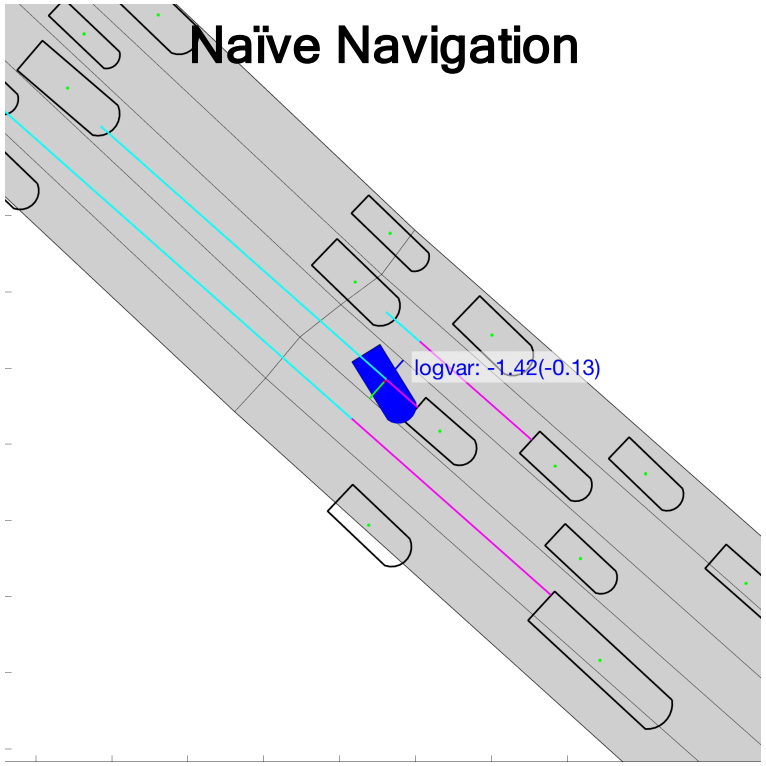
US Highway 101

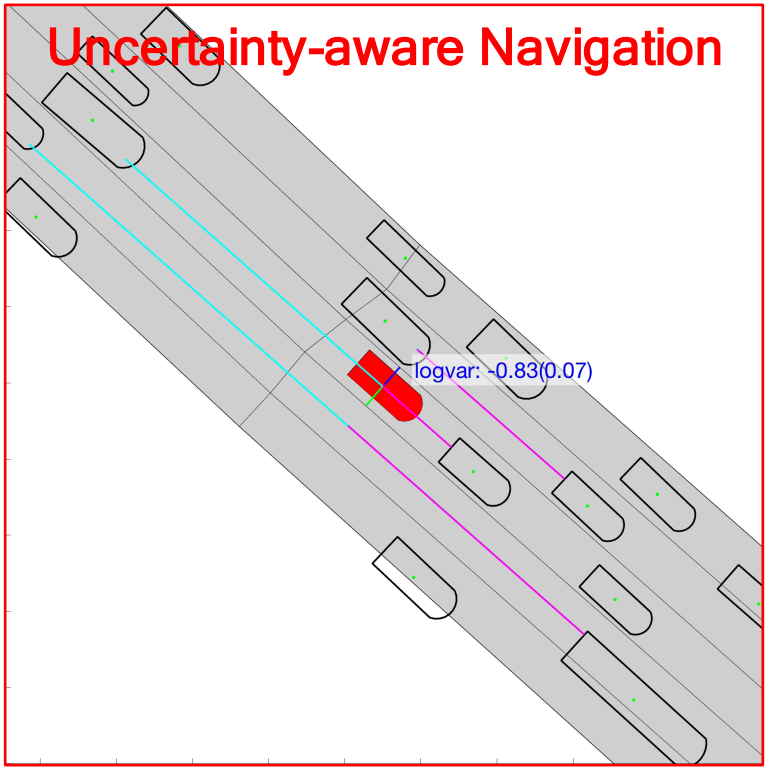




Track Driving Scenario

US Highway 101





Conclusion

A simple but powerful uncertainty estimation method is presented with a mixture density network.

Successfully applied to uncertainty-aware learning from demonstration.

Currently, writing a paper about this topic. ☺

Special thanks to Sungbin Lee and Kyungjae Lee.

Uncertainty-Aware Learning from Demonstration with Mixture Density Networks

Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh