E9: 309 ADL 15-1-2021

http://leap.ee.iisc.ac.in/sriram/teaching/ADL2020/

Housekeeping

- Midterm project III
    - ✓ Evaluation after final exam (1st week of Feb)
- Final Exam (as per IISc schedule)
    - ✓ Jan 23rd afternoon!
- Extra class (Friday 15, nov, 430pm)

Bayesian Deep Learning

- Goal -
  - ➡ Show that the use of dropout (and its variants) in NNs can be interpreted as a Bayesian approximation of a well known probabilistic model.
- Goal -
  - ➡ Develop tools for representing model uncertainty of existing dropout NNs – extracting information that has been thrown away so far. This mitigates the problem of representing model uncertainty in deep learning without sacrificing either computational complexity or test accuracy.

Definition of Gaussian process

- A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions.

- A Gaussian process is fully specified by its mean function m(x) and covariance function k(x, x).

$$f \sim \mathcal{N}(m, k)$$

- This is a natural generalization of the Gaussian distribution whose mean and covariance is a vector and matrix, respectively. The Gaussian distribution is over vectors, whereas the Gaussian process is over functions.

Introduction to Gaussian processes

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix}$$

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad \mathbf{f} \sim \mathcal{N}(\boldsymbol{m}(\mathbf{x}), \boldsymbol{k}(\mathbf{x}, \mathbf{x}'))$$

✳ Mean will be function of x and variance will also be functions of two data points.

Gaussian process - Example

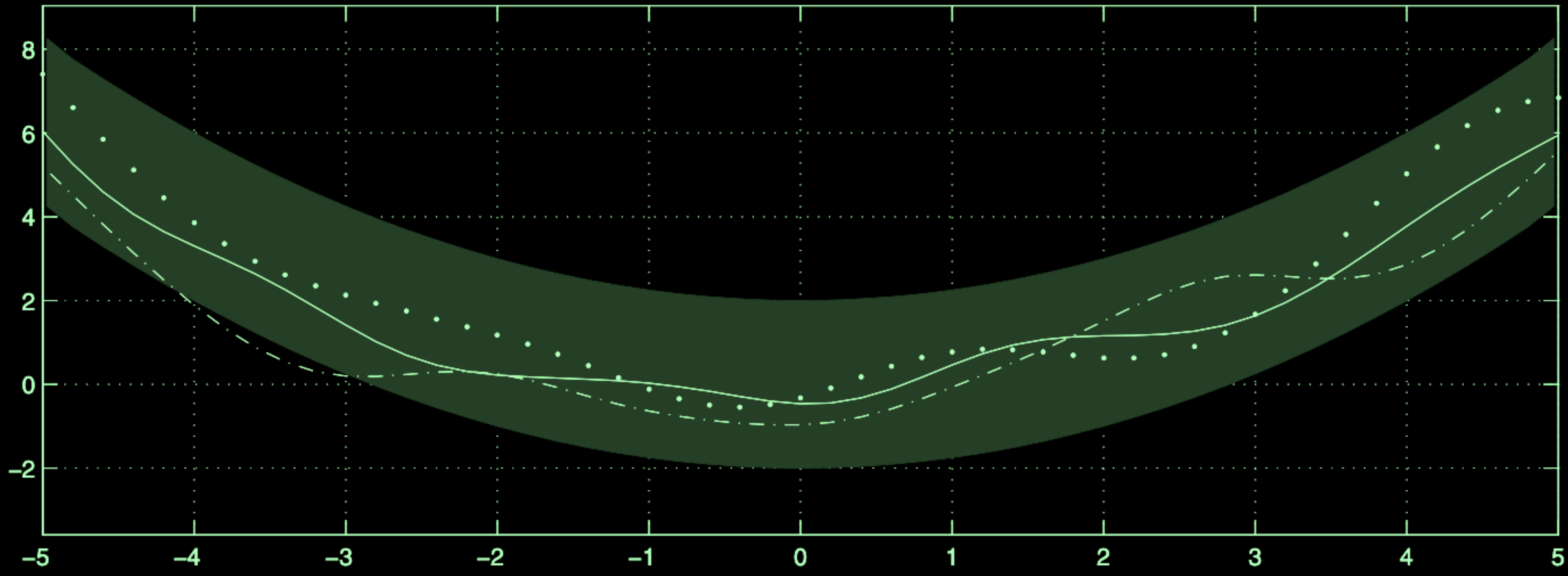


Fig. 1. Function values from three functions drawn at random from a GP as specified in Eq. (2). The dots are the values generated from Eq. (4), the two other curves have (less correctly) been drawn by connecting sampled points. The function values suggest a smooth underlying function; this is in fact a property of GPs with the squared exponential covariance function. The shaded grey area represent the 95% confidence intervals

Gaussian processes for Bayesian inference

- GP will be used as a prior for Bayesian inference.

- The prior does not depend on the training data, but specifies some properties of the functions.

- One of the primary goals computing the posterior is that it can be used to make predictions for unseen test cases.

- Let f be the known function values of the training cases, and let f_* be a set of function values corresponding to the test set inputs, X_*.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma^* \\ \Sigma_*^T & \Sigma_{**} \end{bmatrix} \right)$$

Gaussian processes for Bayesian inference

- Now the quantity of interest is the posterior distribution (for function values)

$$\mathbf{f}_*|\mathbf{f} \sim \mathcal{N}\left(\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_*^T\boldsymbol{\Sigma}^{-1}(\mathbf{f}-\boldsymbol{\mu}), \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_*\right)$$

- Thus,

$$f|\mathcal{D} \sim \mathcal{GP}(m_D, k_D)$$
$$m_D(x) = m(x) + \Sigma(\mathbf{X}, x)^T\boldsymbol{\Sigma}^{-1}(\mathbf{f}-\mathbf{m})$$
$$k_D(x, x') = k(x, x') - \Sigma(\mathbf{X}, x)^T\Sigma^{-1}\Sigma(\mathbf{X}, x')$$

Gaussian Processes

- where $\Sigma(X, x)$ is a vector of covariances between every training case and x. These are the central equations for Gaussian process predictions.

- Let's examine these equations for the posterior mean and covariance. Notice that the posterior variance $k_D(x, x)$ is equal to the prior variance $k(x, x)$ minus a positive term, which depends on the training inputs;

- thus the posterior variance is always smaller than the prior variance, since the data has given us some additional information

Allowing for noise in the model

- Need to address one final issue: noise in the training outputs.
- It is common to many applications of regression that there is noise in the observations6.
- The most common assumption is that of additive i.i.d. Gaussian noise in the outputs.
- In Gaussian process, the effect is that every f(x) has a extra covariance with itself only (since the noise is assumed independent), with a magnitude equal to the noise variance:

Allowing for noise in the model

$$y(x) = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$f \sim \mathcal{GP}(, k), \quad y \sim \mathcal{GP}(m, k + \sigma_n^2 \delta_{i,i'})$$

- Notice, that the indexes to the Kronecker's delta is the identify of the cases, i, and not the inputs xi; you may have several cases with identical inputs, but the noise on these cases is assumed to be independent.

Allowing for noise in the model

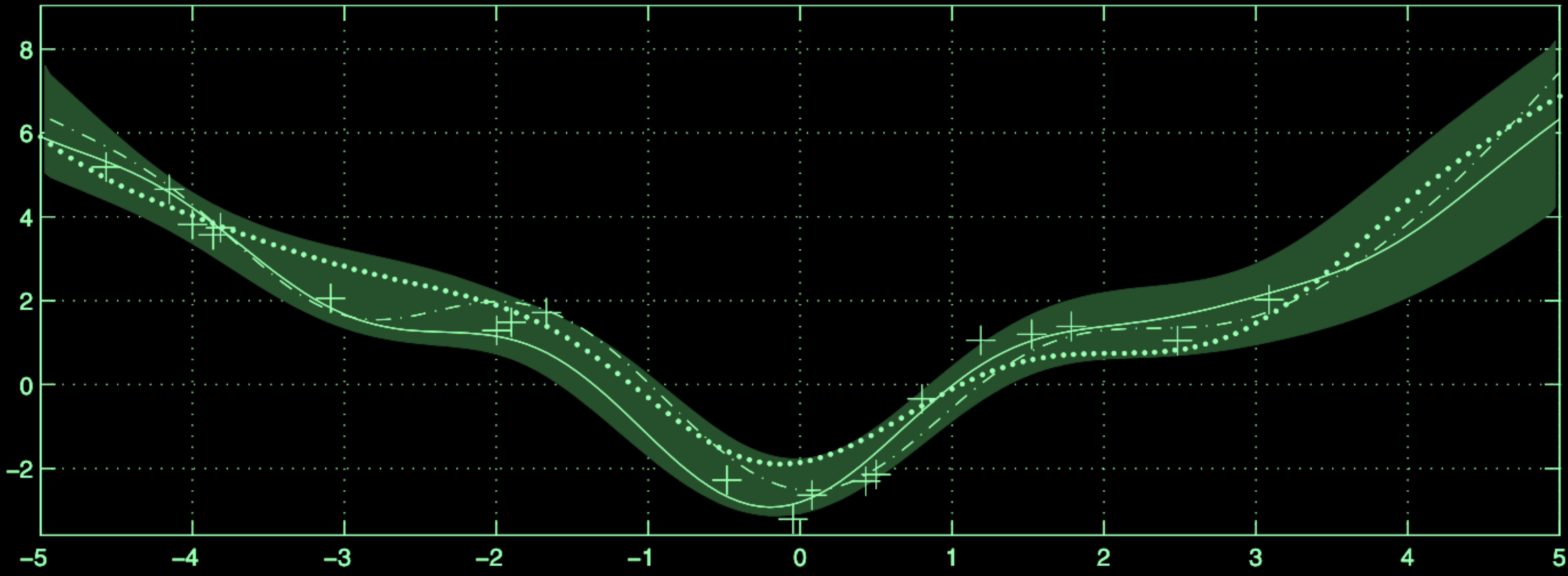


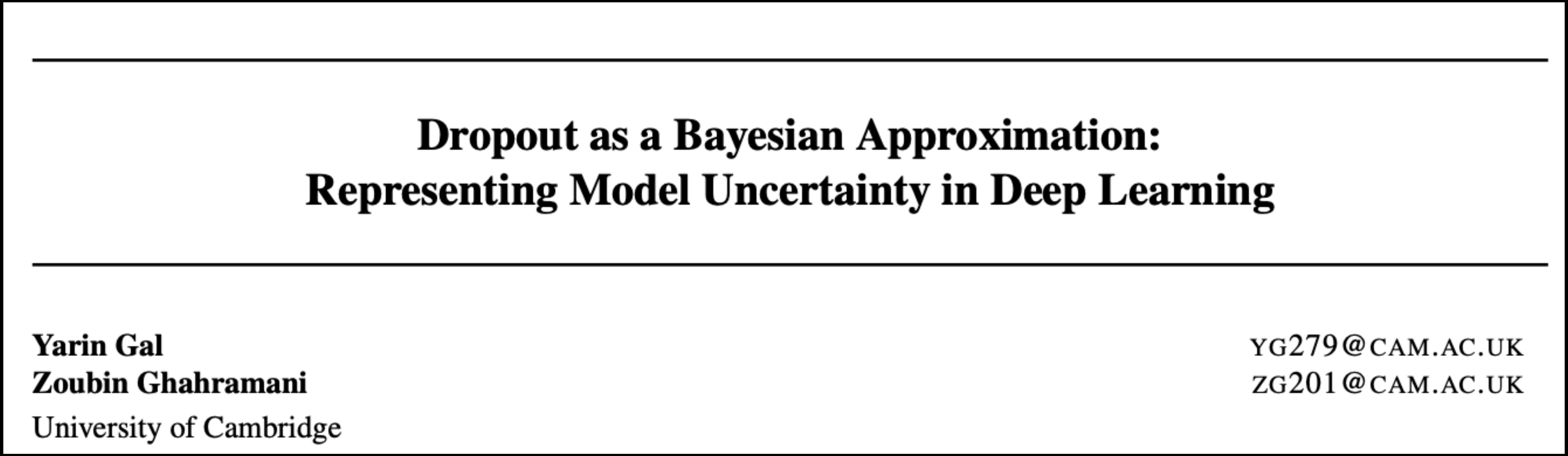
Fig. 2. Three functions drawn at random from the posterior, given 20 training data points, the $\mathcal{GP}$ as specified in Eq. (3) and a noise level of $\sigma_n = 0.7$. The shaded area gives the 95% confidence region. Compare with Figure 1 and note that the uncertainty goes down close to the observations

Dropout and its Bayesian Interpretation

Broad goal

- Interpretation of dropout as a Bayesian model
    - ✓ offers an explanation to some of its properties, such as its ability to avoid over-fitting
    - ✓ our insights allow us to treat NNs with dropout as fully Bayesian models, and obtain uncertainty estimates over their features.
- Mathematically,
    - ➡ we will show that a deep neural network (NN) with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process model

Dropouts



Dropout as a Bayesian Approximation:
Representing Model Uncertainty in Deep Learning

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

Dropout in NN

- ✳ Reviewing the dropout NN model quickly for the case of a single hidden layer NN. This is done for ease of notation, and the generalisation to multiple layers is straightforward.

- ✳ Denote by W1,W2 the weight matrices connecting the first layer to the hidden layer and connecting the hidden layer to the output layer respectively. These linearly transform the layers' inputs before applying some element-wise non-linearity $\sigma(\cdot)$. Denote by b the biases by which we shift the input of the non-linearity. We assume the model to output D dimensional vectors while its input is Q dimensional vectors, with K hidden units. Thus W1 is a $Q \times K$ matrix, W2 is a $K \times D$ matrix, and b is a K dimensional vector. A standard NN model would

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$$

Dropout

- Dropout is applied by sampling two binary vectors z1, z2 of dimensions Q and K respectively. The elements of the vectors are distributed according to a Bernoulli distribution with some parameter

$$p_i \in \{0, 1\} \quad i = 1, 2$$

$$z_{1q} \sim Bernoulli(p_1)$$
$$z_{2k} \sim Bernoulli(p_2)$$

- Given an input x, (1 – p1) proportion of the elements of the input are set to zero.

- The output with dropout can be expressed as

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}(\mathbf{Z}_1\mathbf{W}_1) + \mathbf{b})(\mathbf{Z}_2\mathbf{W}_2)$$
$$\mathbf{Z}_1 = diag(\mathbf{z}_1) \quad \mathbf{Z}_2 = diag(\mathbf{z}_2)$$

Loss function

- Loss in regression networks

$$E = \frac{1}{2N}\sum_n ||\mathbf{y}_n - \hat{\mathbf{y}}_n||^2$$

- Loss in classification networks

$$c_n \in [1 ... D]$$

$$\hat{p}_{nd} = \frac{exp(\hat{y}_{nd})}{\sum_{d'} exp(\hat{y}_{nd'})} \qquad E = -\frac{1}{N}\sum_n log\ \hat{p}_{nc_n}$$

- With L2 regularization, the total loss is

$$\mathcal{E}_{dropout} = E + \lambda_1||\mathbf{W}_1||^2 + \lambda_2||\mathbf{W}_2||^2 + \lambda_3||\mathbf{b}||^2$$

Gaussian process

$$\mathbf{f}|\mathcal{X} \sim \mathcal{GP}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$

$$\mathbf{Y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \frac{1}{\tau}\mathbf{I}_N)$$

- To model the data we have to choose a covariance function K(X1, X2) for the Gaussian distribution. This function defines the (scalar) similarity between every pair of input points K(xi , xj).
- Given a finite dataset of size N this function induces an N × N covariance matrix which we will denote K := K(X, X).

Variational Inference

- The output probability distribution on some unseen test data

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

- condition the model on a finite set of random variables **ω**

    ✓ like the weights of the model.

- The distribution $p(\omega|X, Y)$ cannot usually be evaluated analytically. Instead we define an approximating variational distribution $q(\omega)$

Forming a suitable approximation for the weight matrices

$$\mathbf{W}_2 = \mathbf{M}_1 diag(\mathbf{z}_1)$$
$$\mathbf{W}_2 = \mathbf{M}_2 diag(\mathbf{z}_2)$$

- M1 and M2 are full matrices and z1 and z2 are binary vectors with Bernoulli distribution parameterized using p1 and p2.

- In this scenario, maximizing the evidence lower bound gives

$$L_{GP} \approx -\sum_{n=1}^{N} ||\mathbf{y}_n - \hat{\mathbf{y}}_n||^2 - \frac{p1}{2}||\mathbf{M}_1||^2 - -\frac{p2}{2}||\mathbf{M}_2||^2$$

➡ Very similar to the error function optimized in DNN training

$$\mathcal{E}_{dropout} = E + \lambda_1||\mathbf{W}_1||^2 + \lambda_2||\mathbf{W}_2||^2 + \lambda_3||\mathbf{b}||^2$$

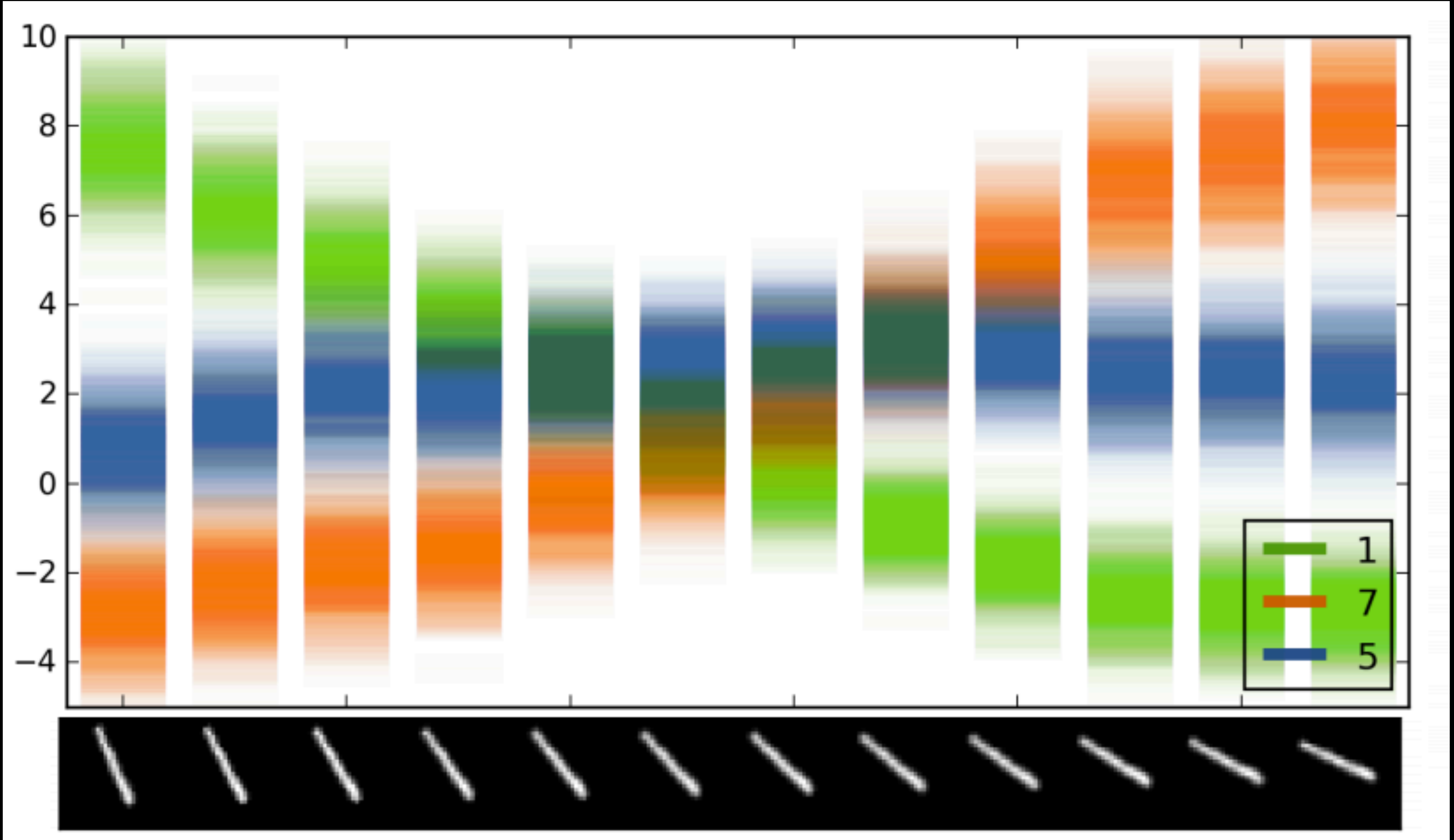
Obtaining the model uncertainity

- ✳ Train the model using dropout and L2 regularization
- ✳ Under the assumed q distribution
  - ✓ Estimate the first order and second statistics of the output given the input
  - ✓ Approximately equal to
    - ◉ First order and second statistic of the output with different dropouts for the given the input.
    - ◉ The first order moment is $\mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \approx \frac{1}{Q}\sum_{q=1}^{Q} \hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}_1^t, \mathbf{W}_2^t)$

Application to uncertaininty modeling in MNIST

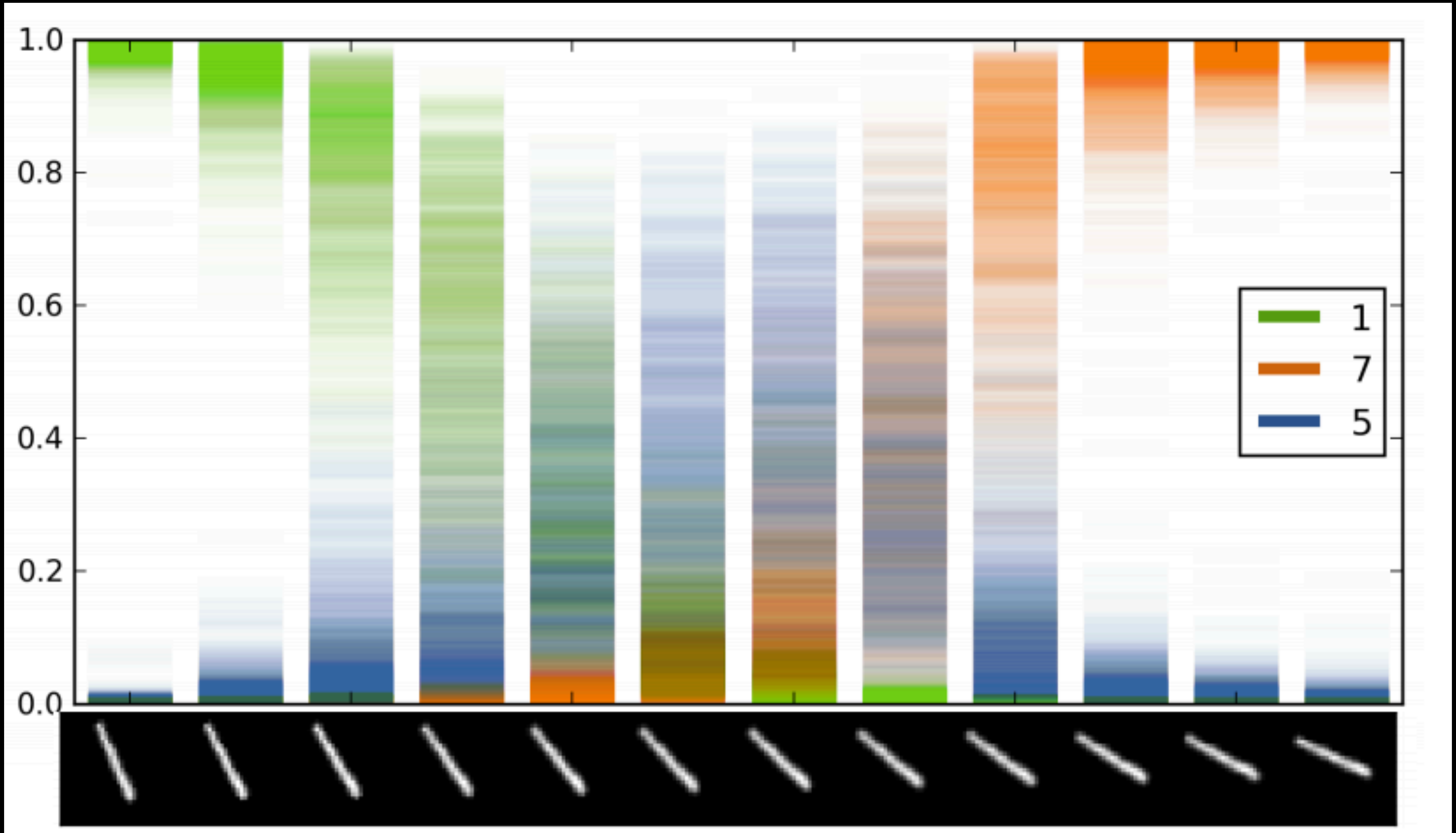
- Train the MNIST model
  - With dropout and regularization
- Obtain the output on a new test sample
  - Using different realizations of dropout on the test data
  - Find the first and second moment of the output for each class
    - ✓ Denoted as uncertainty in the model.

Uncertainty in MNIST



(a) Softmax *input* scatter

Uncertainty in the model output



(b) Softmax *output* scatter

Summary of ADL course …

- Visual and Time Series Modeling: Semantic Models, Recurrent neural models and LSTM models, Encoder-decoder models, Attention models.
- Unsupervised Learning: Restricted Boltzmann Machines, Variational Autoencoders, Generative Adversarial Networks.
- Representation Learning, Causality And Explainability: t-SNE visualization, Hierarchical Representation, semantic embeddings, gradient and perturbation analysis, Topics in Explainable learning, Structural causal models. Uncertainty modeling in deep learning.
- New Architectures: Capsule networks, End-to-end models, Transformer Networks.
- Applications: Applications in in NLP, Speech, Image/Video domains in all modules.

Thanks