Dropout as Bayesian Appoximation: Representing Model Uncertainty in Deep Learning - ICML 2016
Yarin Gal, Zoubin Ghahramani, University of Cambridge

MSDMA Journal Club
Nicolas Thome
Prenom.Nom@cnam.fr
http://cedric.cnam.fr/~thomen/

April, 28th 2017

Outline

The origins

The formal neuron, basis of the neural networks

- **1943:** The formal neuron [MP43]

x_i: inputs
w_i, b: weights
f: activation function
y: output of the neuron

$$y = f(w^\top x + b)$$

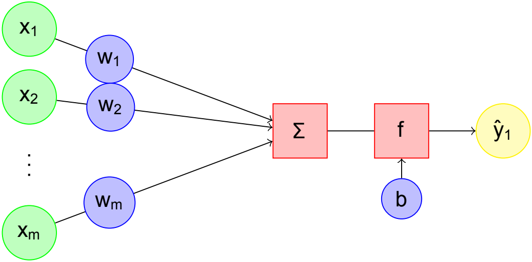


Figure: The formal neuron – **Credits: R. Herault**

The Multi-Layer Perceptron (MLP) & Deep Learning

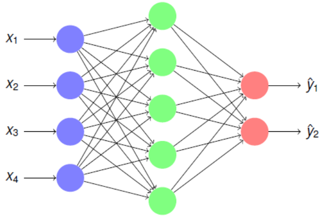


Figure: Perceptron with 1 hidden layer – **Credits: R. Herault**

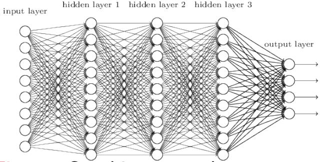
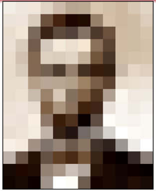


Figure: Stacking more layers, toward "deep learning" – **Credits: M. Nielsen**

- Basis of the "deep learning" field
- Principle: Stacking layers of neural networks to allow more complex and rich functions
- **Not limited to linear prediction**
- With a hidden layer, **can approximate any function** given enough hidden units [Cyb89]
- Can be seen as **different levels of abstraction** from low-level features to the high-level ones

Recognition of low-level signals

Challenge: filling the semantic gap



What we perceive *vs*
What a computer sees

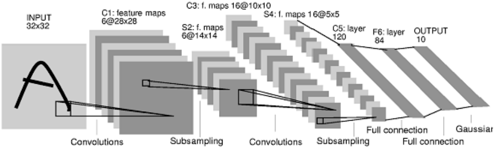
243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

- Illumination variations
- View-point variations
- Deformable objects
- intra-class variance
- *etc*

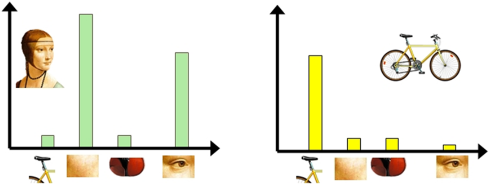
⇒ **How to design "good" intermediate representation ?**

History: Trends and methods in the last four decades

- 80's: training Convolutionnal Neural Networks (CNN) with back-propagation ⇒ postal code reading [LBD+89]

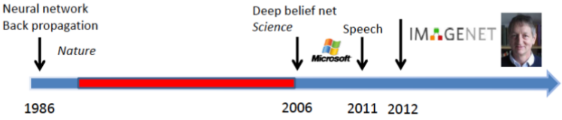


- 90's: golden age of kernel methods, NN = black box
- 2000's: BoW + SVM : state-of-the-art CV



History: Trends and methods in the last four decades

- Deep learning revival: unsupervised learning (DBN) [HOT06]



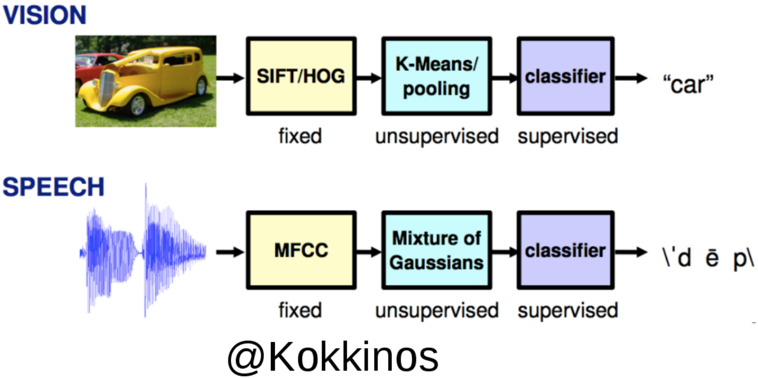
- 2012: CNN outstanding success in ImageNet [KSH12]

Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

  - Huge number of labeled images (10^6 images)
  - GPU implementation for training

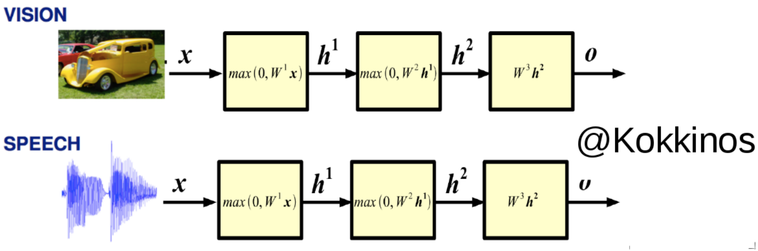
Deep Learning (DL) & Recognition of low-level signals

- DL: breakthrough for the recognition of low-level signal data
- Before DL: handcrafted intermediate representations for each task
  - ⊖ Needs expertise (PhD level) in each field
  - ⊖ Weak level of semantics in the representation



Deep Learning (DL) & Recognition of low-level signals

- DL: breakthrough for the recognition of low-level signal data
- Since DL: automatically **learning intermediate representations**
  - $\oplus$ Outstanding experimental performances >> handcrafted features
  - $\oplus$ Able to learn high level intermediate representations
  - $\oplus$ Common learning methodology $\Rightarrow$ field independent, no expertise

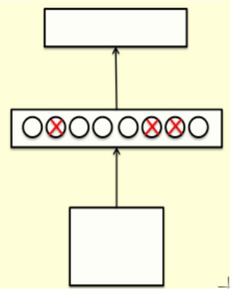


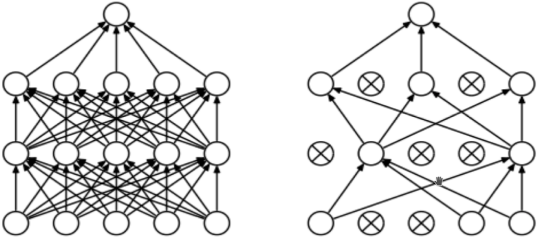
Outline

Deep Learning Modules

Training: droupout

- Randomly omit each hidden unit with probability 0.5
- **Regularization technique**, limits over-fitting (better generalization)
  - Pulls the weights towards what other models want, useful to prevent co-adaptation (feature only helpful when other specific features present)
  - May be viewed as averaging over many NN
  - Slower convergence





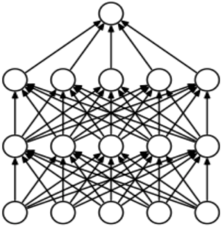
Standard Neural Net After applying dropout.

Credits: Geoffrey E. Hinton, NIPS 2012

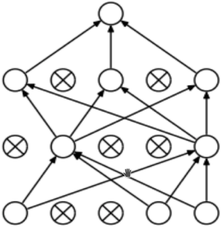
Deep Learning Modules

Training: droupout

- What to do at test time ?
  - Sample many different architectures and take the geometric mean of their output distributions
  - Faster alternative: use all hidden units (but after halving their outgoing weights)
  - Equivalent to the geometric mean in case of single hidden layer
  - Pretty good approximation for multiple layers



Standard Neural Net



After applying dropout.

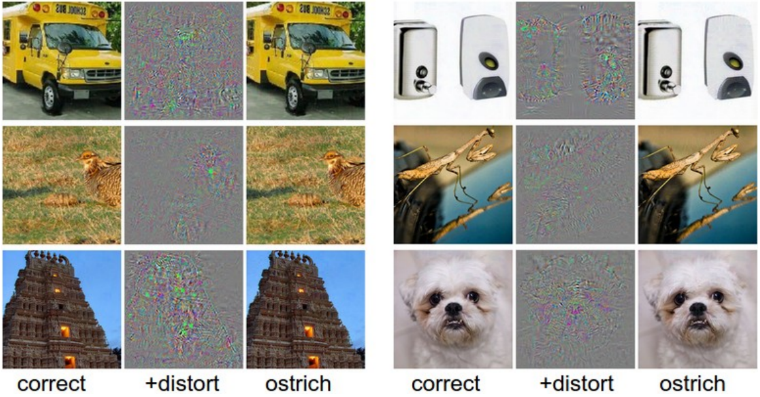
Credits: Geoffrey E. Hinton, NIPS 2012

Outline

Deep Learning (DL) & Uncertainty

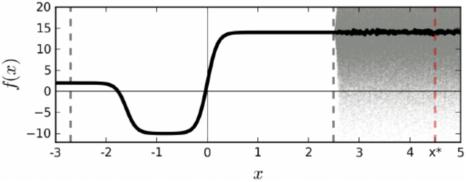
Problem

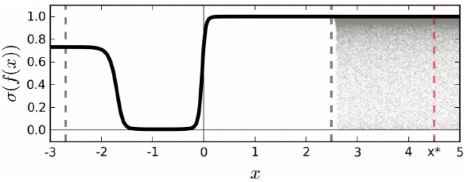
- Deep Models not necessarily robust to input variations
- Deep Models do not naturally capture uncertainty
- Ex: Adversarial Examples



Deep Learning (DL) & Uncertainty: Problem

Softmax output in neural network ≠ confidence (uncertainty) measure !





- x input, f(x) neural net function (left), $\sigma(f(x))$ softmax output (right)
- Solid black line : model pointwise function estimate
- Training data : between dashed gray lines
- Red dashed line: test point x^* (far from training)
- Shaded gray area: uncertainty
- **Conclusions:**
  - Model $\sigma(f(x))$: extrapolations with unjustified high confidence for points far from the training data (probability of 1 to x^*).
  - However, passing the distribution through a softmax (shaded area 1b) better reflects classification uncertainty far from the training data.

Uncertainty Modeling

Modeling uncertainty is crucial in many contexts:

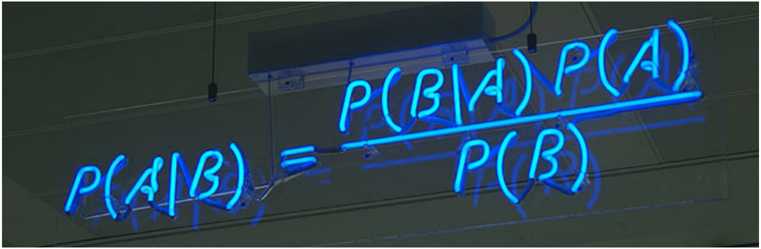
- When error in prediction can have a huge impact, *e.g.*
  - Diagnostic: pass input to an expert
  - Autonomous driving
- Training from few data, *e.g.* active learning (must select informative samples for annotation, *e.g.* based on uncertainty)
- Reinforcement Learning: uncertainty helps in improving the exploitation / exploration tradeoff

Bayesian Models

- Observed inputs $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and outputs $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$
- Prior $p(\mathbf{w})$, likelihood $p(\mathbf{Y}/\mathbf{w}, \mathbf{X})$
- Posterior: Bayes $\Rightarrow p(\mathbf{w}/\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}/\mathbf{w},\mathbf{X})p(\mathbf{w})}{p(\mathbf{Y}/\mathbf{X})} \propto p(\mathbf{Y}/\mathbf{w}, \mathbf{X})p(\mathbf{w})$
- Predictive distribution given new input $\mathbf{x}^*$

$$p(\mathbf{y}^*/\mathbf{x}^*, \mathbf{w}, \mathbf{Y}, \mathbf{X}) = \int p(\mathbf{y}^*/\mathbf{x}^*, \mathbf{w})p(\mathbf{w}/\mathbf{X}, \mathbf{Y})dw$$

- This is what we want !
  - Prob distribution of outputs
  - Naturally gives a measure of uncertainty



Bayesian Models and Variational Inference (VI)

- **BUT...** Posterior $p(\mathbf{w}/\mathbf{X}, \mathbf{Y})$ quickly becomes intractable
  - Closed form solution for very simple models, *i.e.* Bayesian linear regression (or when likelihood conjugate to prior)
  - For moderately complex models: no closed form (*e.g.* a neural network with a single hidden unit)
- Popular solution: approximate $p(\mathbf{w}/\mathbf{X}, \mathbf{Y})$ by $q_\theta(\mathbf{w})$
- Find parameters θ st Kullback-Leibler divergence $KL(q_\theta(\mathbf{w}), p(\mathbf{w}/\mathbf{X}, \mathbf{Y}))$ is minimized
- Minimizing $KL(q_\theta(\mathbf{w}), p(\mathbf{w}/\mathbf{X}, \mathbf{Y}))$ equivalent to maximizing the log evidence lower bound (ELBO):

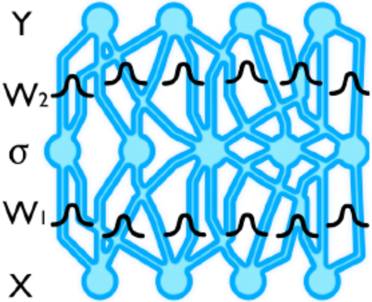
$$\mathcal{L}_{\mathrm{VI}}(\theta) := \int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} - \mathrm{KL}(q_\theta(\boldsymbol{\omega}) || p(\boldsymbol{\omega})) \leq \log p(\mathbf{Y}|\mathbf{X}) = \log \text{evidence},$$

- Resulting in the approximate predictive distribution:

$$p(\mathbf{y}^*/\mathbf{x}^*, \mathbf{w}, \mathbf{Y}, \mathbf{X}) \approx \int p(\mathbf{y}^*/\mathbf{x}^*, \mathbf{w}) q_\theta(\mathbf{w}) dw$$

Bayesian Deep Learning

- Bayesian deep neural networks:
  - Prior on neural network weight, *e.g.* $p(\mathbf{w}_{ik}) \propto e^{-\frac{1}{2}\mathbf{w}_{ik}^T\mathbf{w}_{ik}}$ $\forall$ layer i neuron k
  - $\hat{\mathbf{y}}_i = f_{\mathbf{W}}(\mathbf{x}_i) = \mathbf{W}_L\sigma(...\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}))$
  - $p(\mathbf{y}_i/\mathbf{x}_i, \mathbf{w}) = softmax(f_{\mathbf{W}}(\mathbf{x}_i))$



- **BUT evaluate posterior $p(\mathbf{w}/\mathbf{X}, \mathbf{Y})$ challenging ...**
- Recall: Variational Inference (VI)

$$\mathcal{L}_{\mathrm{VI}}(\theta) := -\sum_{i=1}^{N}\int q_\theta(\boldsymbol{\omega})\log p(\mathbf{y}_i|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i))\mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$

Bayesian Deep Learning Variational Inference (VI)

- Modern solutions: approximate integral with MC integration $\hat{\mathbf{w}} \sim q_\theta(\mathbf{w})$

Algorithm 1 Minimise divergence between $q_\theta(\boldsymbol{\omega})$ and $p(\boldsymbol{\omega}|X, Y)$

1: Given dataset $\mathbf{X}, \mathbf{Y}$,
2: Define learning rate schedule η,
3: Initialise parameters θ randomly.
4: **repeat**
5: Sample M random variables $\widehat{\boldsymbol{\epsilon}}_i \sim p(\boldsymbol{\epsilon})$, S a random subset of $\{1, .., N\}$ of size M.
6: Calculate stochastic derivative estimator w.r.t. θ:

$$\widehat{\Delta\theta} \leftarrow -\frac{N}{M}\sum_{i\in S}\frac{\partial}{\partial\theta}\log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon}_i)}(\mathbf{x}_i)) + \frac{\partial}{\partial\theta}\mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})).$$

7: Update θ:

$$\theta \leftarrow \theta + \eta\widehat{\Delta\theta}.$$

8: **until** θ has converged.

- ⊖ Prohibitive computational cost. To represent uncertainty, the number of parameters in these models is doubled for the same network size.
- ⊖ Requires more time to converge and do not improve on existing techniques.

Dropout and Bayesian Deep Learning Variational Inference (VI)

- Now let us specify some specific $q_\theta(\mathbf{w})$
  - Given variational parameters $\theta = \{\mathbf{m}_{ik}\}_{i,k}$:

$$q_\theta(\omega) = \prod_i q_\theta(\mathbf{W}_i)$$
$$q_\theta(\mathbf{W}_i) = \prod_k q_{\mathbf{m}_{ik}}(\mathbf{w}_{ik})$$
$$q_{\mathbf{m}_{ik}}(\mathbf{w}_{ik}) = p\delta_{\mathbf{0}}(\mathbf{w}_{ik}) + (1-p)\delta_{\mathbf{m}_{ik}}(\mathbf{w}_{ik})$$

  $\rightarrow$ k'th column of the i'th layer is a mixture of two components

  - Or, in a more compact way:

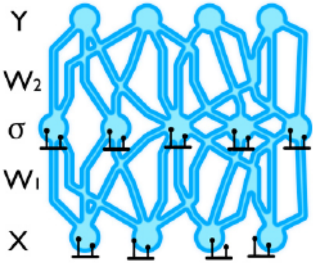
$$\mathbf{z}_{ik} \sim \text{Bernoulli}(p_i) \text{ for each layer } i \text{ and column } k$$
$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}([\mathbf{z}_{ik}]_{k=1}^{K})$$

  with $\mathbf{z}_{ik}$ Bernoulli r.v.s.

Dropout and Bayesian Deep Learning Variational Inference (VI)

The big result

Sounds familiar?



$$\widehat{\mathcal{L}}(\theta) = \overbrace{-\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega})}^{=\text{ loss}} + \overbrace{\mathrm{KL}(q_\theta(\omega) \mid\mid p(\omega))}^{=\, L_2 \text{ reg}}$$

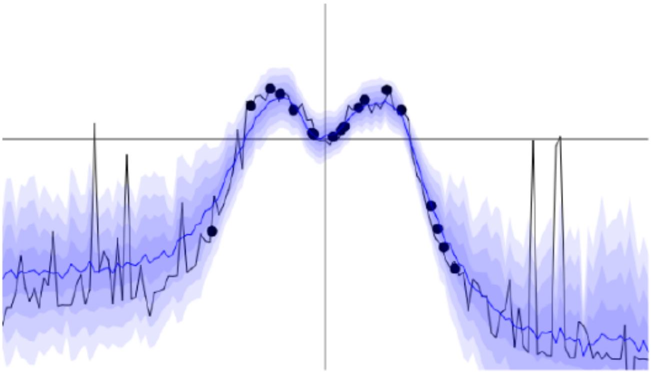
Implementing VI with $q_\theta(\cdot)$ above = implementing dropout in deep network

Dropout and Bayesian Deep Learning Variational Inference (VI)

The big result

- Dropout applied before every weight layer equivalent to variational inference in Bayesian NNs !
- Can be used to get **uncertainty estimates in the network** !

We fit a **distribution**...



Dropout and Uncertainty Estimates

We fit a **distribution**...

- Use first moment for **predictions**:

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

with $\widehat{\mathbf{y}}_t \sim \text{DropoutNetwork}(\mathbf{x}^*)$.

- Use second moment for **uncertainty** (in regression):

$$\text{Var}(\mathbf{y}^*) \approx \frac{1}{T}\sum_{t=1}^{T} \widehat{\mathbf{y}}_t^T \widehat{\mathbf{y}}_t - \mathbb{E}(\mathbf{y}^*)^T \mathbb{E}(\mathbf{y}^*) + \tau^{-1}\mathbf{I}$$

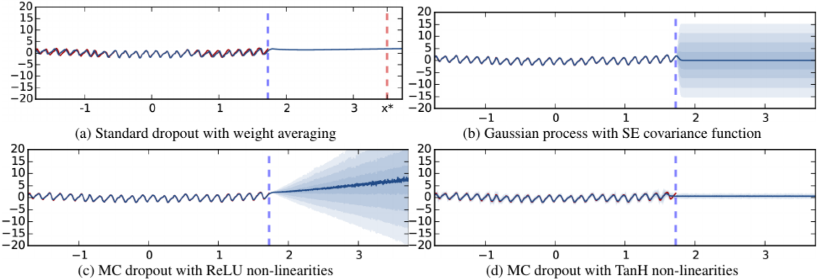
with $\widehat{\mathbf{y}}_t \sim \text{DropoutNetwork}(\mathbf{x}^*)$.

- Drop units at test time and look at mean and sample variance

```
y = []
for _ in xrange(10):
    y.append(model.output(x, dropout=True))
y_mean = numpy.mean(y)
y_var = numpy.var(y)
```

Outline

Model Uncertainty in Regression Tasks



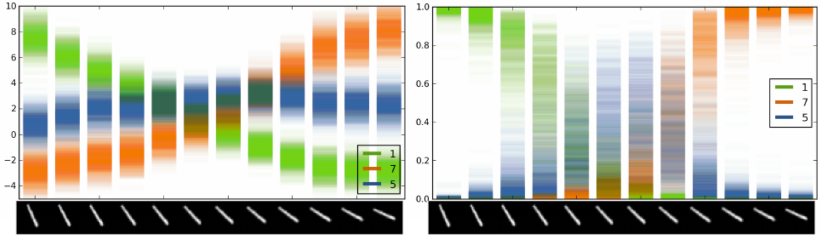
(a) Standard dropout with weight averaging

(b) Gaussian process with SE covariance function

(c) MC dropout with ReLU non-linearities

(d) MC dropout with TanH non-linearities

Model Uncertainty in Classification Tasks



(a) Softmax *input* scatter (b) Softmax *output* scatter

Figure 4. **A scatter of 100 forward passes of the softmax input and output for dropout LeNet.** On the X axis is a rotated image of the digit 1. The input is classified as digit 5 for images 6-7, even though model uncertainty is extremly large (best viewed in colour).

Predictive Performance

	Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
Dataset	**VI**	**PBP**	**Dropout**	**VI**	**PBP**	**Dropout**
Boston Housing	4.32 ±0.29	3.01 ±0.18	**2.97 ±0.85**	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.25**
Concrete Strength	7.19 ±0.12	5.67 ±0.09	**5.23 ±0.53**	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.09**
Energy Efficiency	2.65 ±0.08	1.80 ±0.05	**1.66 ±0.19**	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.09**
Kin8nm	**0.10 ±0.00**	**0.10 ±0.00**	**0.10 ±0.00**	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.03**
Naval Propulsion	**0.01 ±0.00**	**0.01 ±0.00**	**0.01 ±0.00**	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.05**
Power Plant	4.33 ±0.04	4.12 ±0.03	**4.02 ±0.18**	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.05**
Protein Structure	4.84 ±0.03	4.73 ±0.01	**4.36 ±0.04**	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.01**
Wine Quality Red	0.65 ±0.01	0.64 ±0.01	**0.62 ±0.04**	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.06**
Yacht Hydrodynamics	6.89 ±0.67	**1.02 ±0.05**	1.11 ±0.38	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.12**
Year Prediction MSD	9.034 ±NA	8.879 ±NA	**8.849 ±NA**	-3.622 ±NA	-3.603 ±NA	**-3.588 ±NA**

Table 1. **Average test performance in RMSE and predictive log likelihood** for a popular variational inference method (VI, Graves (2011)), Probabilistic back-propagation (PBP, Hernández-Lobato & Adams (2015)), and dropout uncertainty (Dropout).

Conclusion

- Best of both worlds
- Deep learning:
  - $\oplus$ Complex and powerful model for prediction
  - $\oplus$ Dropout scales well to big data
  - $\ominus$ No uncertainty measure
- Bayesian Deep Learning:
  - $\oplus$ Uncertainty Measure
  - $\ominus$ Computation issues, does not scale well and is practice is not used with deep learning

References I

- George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems 2 (1989), no. 4, 303–314.
- Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, Neural Comput. 18 (2006), no. 7, 1527–1554.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105.
- Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural computation 1 (1989), no. 4, 541–551.
- Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics 5 (1943), no. 4, 115–133.