Uncertainty in Deep Learning PhD Thesis

Variational Inference: A Review for Statisticians

Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference *ICLR*

Dropout as a Bayesian Approximation: Insights and Applications ***ICML***

Deep Bayesian Active Learning with Image data ***ICML* 2017**

What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *NIPS* 2017

2018.4.4

Given training inputs $\{\mathrm{X}_1, \ldots, \mathrm{X}_N\}$ and outputs $\{y_1, \ldots, y_N\}$, estimate that a function $y = \mathrm{f}(x)$ that is likely to have generated outputs

prior: $p(\mathrm{f})$ likelihood: $p(\mathrm{Y}|\mathrm{f}, \mathrm{X})$
posterior: $p(\mathrm{f}|\mathrm{X}, \mathrm{Y})$

We can predict an output for a new input x^* by integrate all possible functions f

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

To approximate the integral, assume that the model depends on a finite set of random variables ω alone

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

Variational inference

Let $\mathrm{X} = x_{1:n}$ be a set of observed variables and $Z = z_{1:m}$ be a set of latent variables. We specify a family of densities over the latent variables. Now our goal is to find a approximation $q(z)$ to the exact conditional $p(z|x)$.

We thus minimize the Kullback–Leibler (KL) divergence, intuitively a measure of similarity between two distributions

$$q^*(\mathbf{z}) = \arg\min_{q(\mathbf{z})\in\mathcal{Q}} \mathrm{KL}\left(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})\right).$$

$$\mathrm{KL}\left(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})\right) = \mathbb{E}\left[\log q(\mathbf{z})\right] - \mathbb{E}\left[\log p(\mathbf{z}\,|\,\mathbf{x})\right],$$

$$\mathrm{KL}\left(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})\right) = \mathbb{E}\left[\log q(\mathbf{z})\right] - \mathbb{E}\left[\log p(\mathbf{z},\mathbf{x})\right] + \log p(\mathbf{x}).$$

Because we cannot compute the KL, we optimize an alternative objective that is equivalent to the KL up to an added constant

$$\text{ELBO}(q) = \mathbb{E}\left[\log p(\mathbf{z}, \mathbf{x})\right] - \mathbb{E}\left[\log q(\mathbf{z})\right].$$

This function is called the evidence lower bound(ELBO)

We rewrite the ELBO as a sum of the expected log likelihood of the data and the KL divergence between the prior $p(z)$ and $q(z)$

$$\begin{aligned}\text{ELBO}(q) &= \mathbb{E}\left[\log p(\mathbf{z})\right] + \mathbb{E}\left[\log p(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{E}\left[\log q(\mathbf{z})\right] \\ &= \mathbb{E}\left[\log p(\mathbf{x} \mid \mathbf{z})\right] - \text{KL}\left(q(\mathbf{z}) \| p(\mathbf{z})\right).\end{aligned}$$

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

We define an approximating variational distribution $q(\omega)$ to be as close as possible to the posterior distribution by minimizing the KL divengence.

$$\mathrm{KL}(q(\omega) \,||\, p(\omega|\mathbf{X}, \mathbf{Y})).$$

Minimising the Kullback–Leibler divergence is equivalent to maximising the log evidence lower bound

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

resulting in the approximate predictive distribution

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

Dropout as Approximate Variational Inference in Bayesian Neural Network

Let $\hat{y}$ by be the output of a NN with L layers and a loss function $E(\cdot,\cdot)$ such as the softmax loss. We denote by W_i the NN' s weight matrices of dimensions $K_i \times K_{i-1}$, and by b_i the bias vectors of dimensions K_i for each layer $i = 1, \ldots, L$. In addition to regularization, we have the optimization problem

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N}\sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L} \left(||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2 \right).$$

Given weight matrices W_i and bias vectors b_i for layer i, we often place standard matrix Gaussian prior distributions over the weight matrices, $p(W_i)$:

$$\mathbf{W}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

We are thus interested in the posterior over the weights given our observables X, Y: $p(\omega|\mathrm{X}, \mathrm{Y})$. This posterior is not tractable for a Bayesian NN, and we use variational inference to approximate it.

We define our approximating variational distribution $q(\omega_i)$ for every layer i as

$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}([\mathbf{z}_{i,j}]_{j=1}^{K_i})$$
$$\mathbf{z}_{i,j} \sim \text{Bernoulli}(p_i) \text{ for } i = 1, ..., L,\ j = 1, ..., K_{i-1}.$$

Here $z_{i,j}$ are Bernoulli distributed random variables with some probabilities p_i, and M_i are variational parameters to be optimized. The binary variable $z_{i,j} = 0$ corresponds to unit j in layer $i - 1$ being dropped out as an input to the i' th layer.

The integral in eq is intractable, and we approximate the integral with Monte Carlo integration over ω.

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

$$\widehat{\mathcal{L}}_{\text{VI}} := \sum_{i=1}^{N} E\left(\mathbf{y}_i, \widehat{\mathbf{f}}(\mathbf{x}_i, \widehat{\omega}_i)\right) - \text{KL}(q(\omega)||p(\omega)) \qquad \widehat{\omega}_i \sim q(\omega)$$

$$\widehat{\mathcal{L}}_{\text{VI}} := \sum_{i=1}^{N} E\left(\mathbf{y}_i, \widehat{\mathbf{f}}(\mathbf{x}_i, \widehat{\omega}_i)\right) - \text{KL}(q(\omega)||p(\omega)) \qquad \widehat{\omega}_i \sim q(\omega)$$

$E(\cdot,\cdot)$ being the softmax loss (for a softmax likelihood).
Note that sampling from $q(W_i)$ is identical to performing dropout on layer i in a network whose weights are $(M_i)_{i=1}^{L}$. The second term can be approximated, resulting in the same objective as dropout.

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N} \sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L} \left(||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2\right).$$

Dropout and Bayesian NNs, in effect, result in the same model parameters that best explain the data.

Predictions in this model follow eq replacing the posterior $p(\omega|D_{train})$ with the approximate Posterior $q(\omega)$. We can approximate the integral with Monte Carlo integration:

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

$$\begin{aligned} p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) &= \int p(y = c|\mathbf{x}, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})\mathrm{d}\boldsymbol{\omega} \\ &\approx \int p(y = c|\mathbf{x}, \boldsymbol{\omega})q_\theta^*(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\ &\approx \frac{1}{T}\sum_{t=1}^{T} p(y = c|\mathbf{x}, \widehat{\boldsymbol{\omega}}_t) \end{aligned}$$

With $\widehat{\omega}_t \sim q_\theta^*(\omega)$, where $q_\theta(\omega)$ is the Dropout distribution.

Deep Bayesian Active Learning

Deep learning poses several difficulties when used in an active learning setting.

- Active learning methods generally rely on being able to learn and update models from small amounts of data.
- Many AL acquisition functions rely on model uncertainty, yet deep learning methods rarely represent such model uncertainty.

Acquisition Functions and their Approximations

Given a model M, pool data D_{pool}, and inputs $x \epsilon D_{pool}$, an acquisition function $a(x, M)$ is a function of x that the AL system uses to decide where to query next:

$$x^* = \text{argmax}_{x \in \mathcal{D}_{\text{pool}}} a(x, \mathcal{M}).$$

For classification, several acquisition functions are available:

1. Choose pool points that maximize the predictive entropy (*Max Entropy*)

$$\mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] := -\sum_c p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) \log p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}).$$

2. Choose pool points that are expected to maximize the mutual information between predictions and model posterior (*BALD*)

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] = \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right]$$

$\mathbb{H}[y|x, w]$ is the entropy of y given model weight ω.

3. Maximize the *Variation Ratios*

$$\text{variation-ratio}[\mathbf{x}] := 1 - \max_y p(y|x, \mathcal{D}_{\text{train}})$$

4. Maximize mean standard deviation (*Mean STD*)

$$\sigma_c = \sqrt{\mathbb{E}_{q(\boldsymbol{\omega})}[p(y=c|\mathbf{x},\boldsymbol{\omega})^2] - \mathbb{E}_{q(\boldsymbol{\omega})}[p(y=c|\mathbf{x},\boldsymbol{\omega})]^2}$$

$$\sigma(\mathbf{x}) = \frac{1}{C}\sum_c \sigma_c$$

5. Random acquisition (baseline)

We can approximate each of these acquisition functions using our approximate distribution $q_\theta^*(\omega)$. For BALD, for example, we can write the acquisition function as follows :

$$\begin{aligned}\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] &:= \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right] \\ &= -\sum_c p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) \log p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) \\ &\quad + \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\sum_c p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log p(y = c|\mathbf{x}, \boldsymbol{\omega})\right],\end{aligned}$$

$\mathbb{I}[y, w|x, D_{train}]$ can be approximated in our setting using

$$p(y = c|\mathbf{x}, \mathcal{D}_{\text{train}}) = \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega}$$

then,

$$\begin{aligned}&\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}] = \\ &\quad -\sum_c \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega} \\ &\qquad \cdot \log \int p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}}) \mathrm{d}\boldsymbol{\omega} \\ &\quad + \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D}_{\text{train}})}\left[\sum_c p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log p(y = c|\mathbf{x}, \boldsymbol{\omega})\right].\end{aligned}$$

Swapping the posterior $p(w|D_{train})$ with our approximate posterior $q_\theta^*(\omega)$ and through MC sampling, we then have

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

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

$$+ \mathbb{E}_{q_\theta^*(\boldsymbol{\omega})} \left[\sum_c p(y=c|\mathbf{x},\boldsymbol{\omega}) \log p(y=c|\mathbf{x},\boldsymbol{\omega}) \right]$$

$$\approx -\sum_c \left(\frac{1}{T} \sum_t \widehat{p}_c^t \right) \log \left(\frac{1}{T} \sum_t \widehat{p}_c^t \right)$$

$$+ \frac{1}{T} \sum_{c,t} \widehat{p}_c^t \log \widehat{p}_c^t =: \widehat{\mathbb{I}}[y, \boldsymbol{\omega}|\mathbf{x}, \mathcal{D}_{\text{train}}]$$

with $\hat{p}_c^t$ the probability of input x with model parameters $\widehat{w}_t \sim q_\theta^*(\omega)$ to take class c:

$$\widehat{\mathbf{p}}^t = [\widehat{p}_1^t, ..., \widehat{p}_C^t] = \mathrm{softmax}(\mathbf{f}^{\widehat{\boldsymbol{\omega}}_t}(\mathbf{x})).$$

We then have

$$\widehat{\mathbb{I}}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}] \xrightarrow[T \to \infty]{} \mathbb{H}[y | \mathbf{x}, q_\theta^*] - \mathbb{E}_{q_\theta^*(\boldsymbol{\omega})} \left[\mathbb{H}[y | \mathbf{x}, \boldsymbol{\omega}] \right]$$
$$\approx \mathbb{I}[y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}_{\text{train}}],$$

Comparison of various acquisition functions

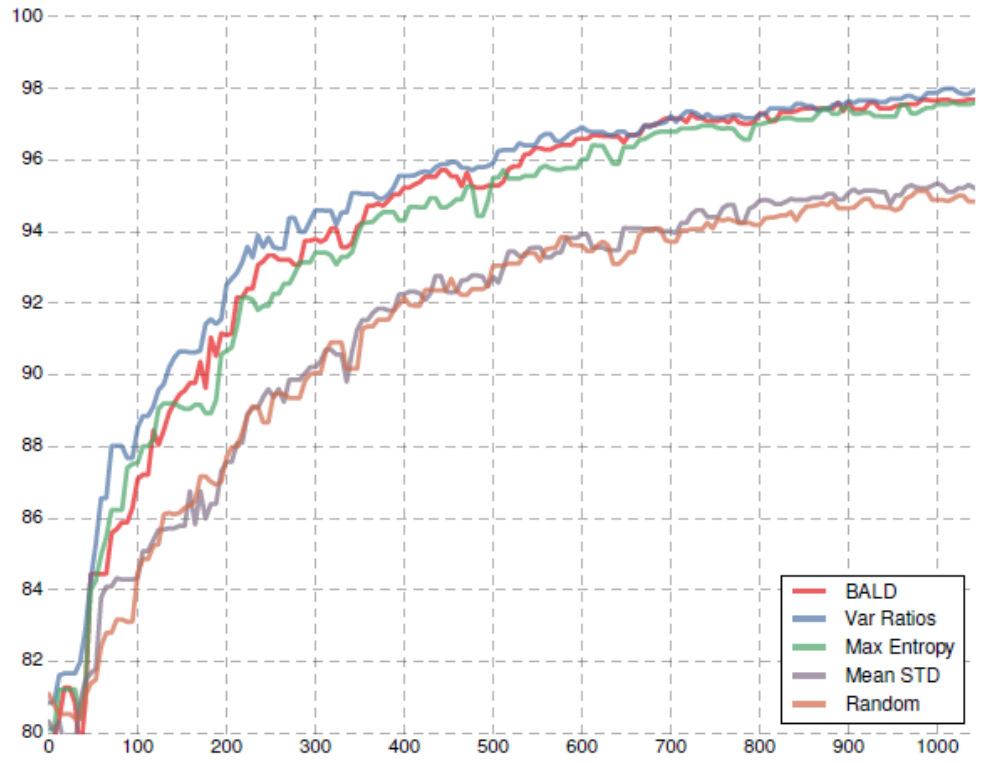

Figure 1. **MNIST test accuracy as a function of number of acquired images from the pool set** (up to 1000 images, using validation set size 100, and averaged over 3 repetitions). Four acquisition functions (*BALD*, *Variation Ratios*, *Max Entropy*, and *Mean STD*) are evaluated and compared to a *Random* acquisition function.

Comparison to current active learning techniques with image data

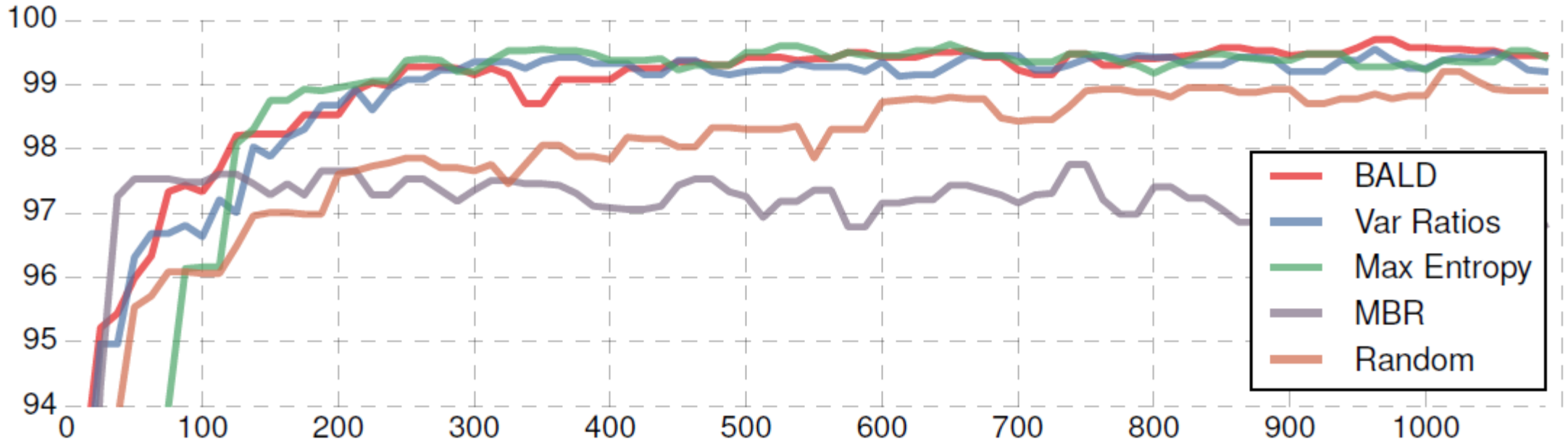

Figure 3. MNIST test accuracy (two digit classification) as a function of number acquired images, compared to a current technique for active learning of image data: MBR (Zhu et al., 2003).

Comparison to semi-supervised learning

Technique	Test error
Semi-supervised:	
Semi-sup. Embedding (Weston et al., 2012)	5.73%
Transductive SVM (Weston et al., 2012)	5.38%
MTC (Rifai et al., 2011)	3.64%
Pseudo-label (Lee, 2013)	3.46%
AtlasRBF (Pitelis et al., 2014)	3.68%
DGN (Kingma et al., 2014)	2.40%
Virtual Adversarial (Miyato et al., 2015)	1.32%
Ladder Network (Γ-model) (Rasmus et al., 2015)	1.53%
Ladder Network (full) (Rasmus et al., 2015)	0.84%
Active learning with various acquisitions:	
Random	4.66%
BALD	1.80%
Max Entropy	1.74%
Var Ratios	1.64%